UNIVERSIDAD DE MÁLAGA

Escuela Técnica Superior de Ingeniería Informática
Programa de Doctorado en Tecnologías Informáticas

Doctoral Thesis

Secure Multi-Party Computation: Constructions and Applications in Blockchain

Author:

Daniel Morales Escalera

Supervisors:

Isaac Agudo Ruiz
Francisco Javier López Muñoz

March 2025

UNIVERSIDAD DE MÁLAGA

AUTOR: Daniel Morales Escalera

https://orcid.org/0000-0002-6932-7437

EDITA: Publicaciones y Divulgación Científica. Universidad de Málaga

Esta obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-SinObraDerivada 4.0 Internacional:
https://creativecommons.org/licenses/by-nc-nd/4.0/legalcode
Cualquier parte de esta obra se puede reproducir sin autorización
pero con el reconocimiento y atribución de los autores.
No se puede hacer uso comercial de la obra y no se puede alterar, transformar o hacer obras derivadas.

Esta Tesis Doctoral está depositada en el Repositorio Institucional de la Universidad de Málaga (RIUMA): riuma.uma.es

UNIVERSIDAD DE MÁLAGA

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD DE LA TESIS PRESENTADA PARA OBTENER EL TÍTULO DE DOCTOR

D./Dña DANIEL MORALES ESCALERA
Estudiante del programa de doctorado TECNOLOGÍAS INFORMÁTICAS de la Universidad de Málaga, autor/a de la tesis, presentada para la obtención del título de doctor por la Universidad de Málaga, titulada: SECURE MULTI-PARTY COMPUTATION: CONSTRUCTIONS AND APPLICATIONS IN BLOCKCHAIN

Realizada bajo la tutorización de ISAAC AGUDO RUIZ y dirección de ISAAC AGUDO RUIZ Y FRANCISCO JAVIER LOPEZ MUÑOZ (si tuviera varios directores deberá hacer constar el nombre de todos)

DECLARO QUE:

La tesis presentada es una obra original que no infringe los derechos de propiedad intelectual ni los derechos de propiedad industrial u otros, conforme al ordenamiento jurídico vigente (Real Decreto Legislativo 1/1996, de 12 de abril, por el que se aprueba el texto refundido de la Ley de Propiedad Intelectual, regularizando, aclarando y armonizando las disposiciones legales vigentes sobre la materia), modificado por la Ley 2/2019, de 1 de marzo.

Igualmente asumo, ante a la Universidad de Málaga y ante cualquier otra instancia, la responsabilidad que pudiera derivarse en caso de plagio de contenidos en la tesis presentada, conforme al ordenamiento jurídico vigente.

En Málaga, a 07 de MARZO de 2025

Fdo.: DANIEL MORALES ESCALERA Doctorando/a	Fdo.: ISAAC AGUDO RUIZ Tutor/a
Fdo.: ISAAC AGUDO RUIZ Y FRANCISCO JAVIER LOPEZ MUÑOZ	

UNIVERSIDAD DE MÁLAGA

Edificio Pabellón de Gobierno. Campus El Ejido. 29071
Tel.: 952 13 10 28 / 952 13 14 61 / 952 13 71 10
E-mail: doctorado@uma.es

D. Francisco Javier López Muñoz, Catedrático de Universidad del área de Ingeniería Telemática del Departamento de Lenguajes y Ciencias de la Computación de la Universidad de Málaga, y D. Isaac Agudo Ruiz, Profesor Titular de Universidad del área de Ingeniería Telemática del Departamento de Lenguajes y Ciencias de la Computación de la Universidad de Málaga

CERTIFICAN QUE:

D. Daniel Morales Escalera, Graduado en Ingeniería Telemática, ha realizado en el Departamento de Lenguajes y Ciencias de la Computación de la Universidad de Málaga, bajo su dirección, el trabajo de investigación correspondiente a su Tesis Doctoral, titulada:

Secure Multi-Party Computation: Constructions and Applications in Blockchain

Revisado el presente trabajo, estiman que puede ser presentado al tribunal que ha de juzgarlo y autorizan la presentación de esta Tesis Doctoral en la Universidad de Málaga. Además, autorizan la lectura de la Tesis y declaran que las publicaciones que avalan la Tesis no han sido utilizadas en Tesis anteriores.

Málaga, a 07 de marzo de 2025

Fdo: D. Francisco Javier López Muñoz
Catedrático de Universidad
Departamento de Lenguajes y Ciencias de la Computación

Fdo: D. Isaac Agudo Ruiz
Profesor Titular de Universidad
Departamento de Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA

Abstract

Protection of data in use is a new paradigm that aims to provide data owners with mechanisms to ensure that their data is not exposed, even when it is part of a computation. Within this paradigm, Privacy Enhancing Technologies (PETs) in general, and Secure Multi-Party Computation (MPC) in particular, offer interesting possibilities for security, mainly by relying on cryptographic techniques. Among the wide range of applications that can benefit from MPC, we identify one of great interest that lacks privacy by default: blockchain. Blockchain technologies require data to be publicly available in order to guarantee the verifiability and correctness of their results, but this implies an obstacle for many use cases to be adopted in the decentralized environment.

This thesis is devoted to the study of MPC from a practical point of view, and mainly to the relations that exist between MPC and blockchain. This work is carried out at different levels, from the more general to the more specific. More specifically, we provide an analysis and discussion of different MPC protocols and their peculiarities when used in real systems. We also provide specific solutions to achieve privacy-preserving applications in the blockchain ecosystem by relying on MPC. We approach this not from a specific application, but from a general computational approach, where MPC serves as a confidentiality layer that can be invoked from smart contracts. On the other hand, at the cryptographic level, we provide a new solution to achieve an efficient and dynamic delivery of secret shared data to outsourced MPC committees. This allows data owners to benefit from sending a single ciphertext that can be shared publicly, and also from selecting the specific committee to share the data after it has been encrypted and stored. This new construction, namely Oblivious Sharing Re-Encryption (OSRE), benefits some scenarios, such as those in which IoT devices send periodic data, or those in blockchain that rely on MPC for confidential computations. Finally, we also provide three applications where PETs in general and MPC in particular offer privacy and usability improvements: Private Set Intersection (PSI) computations in consortium blockchains, a privacy-preserving use of location data for a crowd counting application, and a decentralized mixer service in Bitcoin with reversible unlinkability that achieves accountability and compliance while ensuring that payments remain private.

UNIVERSIDAD DE MÁLAGA

UNIVERSIDAD
DE MÁLAGA

A mi abuelo Lorenzo, "*el cabecita*",
de quien me hubiera gustado aprender.

UNIVERSIDAD
DE MÁLAGA

UNIVERSIDAD
DE MÁLAGA

Agradecimientos

Realizar una tesis doctoral conlleva un proceso agitado. Si algo no es, es una línea recta, monótona y sosegada. Por el contrario, es una maraña de altibajos, emocionales, de carga de trabajo, de incertidumbre, e incluso de conocimiento. Pero si algo tengo claro es que la persona que escribe esto hoy no es la misma que comenzó el proceso. Y vaya proceso.

Es de justicia mencionar a todas las personas que han formado parte de este camino, pues sin ese soporte difícilmente podría haber llegado a buen puerto. En primer lugar, agradecer a mi director Isaac, pues él es el principal culpable tanto de las maravillas de este proceso como de las horas de estrés que llevo a mis espaldas. Gracias por haber confiado en mi más allá de lo que yo mismo habría sido capaz de hacerlo y gracias por guiarme durante todo el proceso. Agradecer también a mi director Javier, por todas las oportunidades que me ha facilitado y por la increíble experiencia aportada a cada decisión. Y por supuesto, cómo no, agradecer a todo el grupo de NICS Lab, del que me he sentido uno más desde el primer día: Anto, Carmen, Cristina, Davide, Fran, Jesús, Monte, Onieva, Rodri y Rubén. No cambiéis nunca, pues ese ambiente de trabajo vale oro. Por supuesto debo añadir a todos mis chicos junior, con los que he compartido momentos muy divertidos. Me perdonaréis que no os mencione uno a uno, pero sería inviable añadir tantos nombres.

Por supuesto, mi siguiente agradecimiento es para mi guerrera, mi pilar fundamental, mi Elenita. Sin duda alguna nuestros procesos por separado se han combinado en un solo cauce de esfuerzo y soporte mutuo. Tú más que nadie me has comprendido y apoyado, y tú más que nadie te has comido mis interminables explicaciones de cosas que nadie estaba dispuesto a escuchar. Te quiero.

Agradecer también a mis padres por su amor incondicional, pues sostener a un hijo estudiando tantos años no es tarea fácil. Sé de sobra todo el esfuerzo que lleváis hecho, y sé de sobra que tengo a los mejores padres del mundo.

Finalmente, agradecer también a todos mis amigos, aquellos a los que no he podido ver todo lo que me hubiera gustado durante estos años. Han sido años dispersos, pero cuando nos vemos parece que el tiempo no ha pasado en absoluto.

A todos los que habéis sido parte de esto, gracias. De corazón.

UNIVERSIDAD DE MÁLAGA

UNIVERSIDAD
DE MÁLAGA

Contents

1 Introduction **1**
- 1.1 Motivation and Scope . . . 2
  - 1.1.1 Protection of data in usage . . . 3
  - 1.1.2 Decentralization of trust . . . 3
  - 1.1.3 Secure Multi-Party Computation . . . 4
- 1.2 Goals and Contributions . . . 5
- 1.3 Thesis outline . . . 6
- 1.4 Publications and Funding . . . 9

2 Preliminaries **11**
- 2.1 Secure Multi-Party Computation . . . 11
  - 2.1.1 Problem formulation and main characteristics . . . 11
  - 2.1.2 MPC basic primitives . . . 14
    - 2.1.2.1 Oblivious Transfer . . . 14
    - 2.1.2.2 Secret Sharing . . . 15
    - 2.1.2.3 Commitments . . . 17
    - 2.1.2.4 Zero-Knowledge Proofs . . . 18
    - 2.1.2.5 Homomorphic Encryption . . . 19
  - 2.1.3 Main MPC protocols . . . 21
    - 2.1.3.1 Functionality representation . . . 21
    - 2.1.3.2 Foundational works . . . 22
    - 2.1.3.3 Modern paradigms in MPC . . . 26
    - 2.1.3.4 New dynamic approaches for MPC . . . 28
- 2.2 Blockchain . . . 29
  - 2.2.1 Blockchain classifications . . . 31
    - 2.2.1.1 Access to the blockchain . . . 31
    - 2.2.1.2 Ledger structure and interaction . . . 32
    - 2.2.1.3 Consensus mechanisms . . . 33
  - 2.2.2 The privacy problem in blockchain . . . 34

3 Analysis of MPC instances and protocols **36**
- 3.1 Abstract models and trust relations . . . 36
- 3.2 Relevant protocols and applications in MPC . . . 41
  - 3.2.1 Distributed Random Number Generator . . . 41
  - 3.2.2 Threshold cryptography . . . 43
    - 3.2.2.1 Threshold block ciphers . . . 43
    - 3.2.2.2 Threshold signatures . . . 45

UNIVERSIDAD DE MÁLAGA

3.2.2.3 Threshold decryption . . . 46
3.2.3 Private Set Operations . . . 47
3.3 A review on Private Set Intersection . . . 48
3.3.1 Systematic Literature Review methodology . . . 48
3.3.2 Scenarios and applications . . . 50
3.3.3 PSI taxonomy and main works . . . 51
3.3.3.1 Encoding data with hash-based structures . . . 52
3.3.3.2 General PSI . . . 53
3.3.3.3 PSI-CA and C-PSI . . . 57
3.3.3.4 MP-PSI . . . 58
3.3.3.5 O-PSI . . . 59
3.3.4 Discussion and conclusions . . . 59
3.4 Summary . . . 63

4 Approaches for MPC in the blockchain setting 64
4.1 Relations between MPC and blockchain . . . 64
4.2 Approaches and challenges . . . 66
4.3 MPC in Hyperledger Besu . . . 68
4.3.1 Private transactions in Besu . . . 69
4.3.2 MPC extension in Besu . . . 72
4.3.2.1 Asymmetric state . . . 74
4.3.2.2 Extended private transactions . . . 74
4.3.2.3 Smart contract interface . . . 75
4.3.2.4 MPC execution and message delivery . . . 77
4.3.3 Threat analysis and discussion . . . 79
4.4 Framework for delegated MPC in blockchain . . . 80
4.4.1 Architecture and workflow . . . 81
4.4.1.1 Atomic embedding of asynchronous operations . . . 84
4.4.1.2 Committee selection and interaction . . . 86
4.4.1.3 Public verifiability . . . 87
4.4.1.4 Implementation details . . . 89
4.4.2 Theoretical evaluation . . . 90
4.4.3 Threat analysis and discussion . . . 92
4.5 Summary . . . 93

5 Dynamic and compact sharing of secrets 94
5.1 Introduction . . . 94
5.1.1 Approaches to efficient communication in SS . . . 95
5.1.2 General conceptualization . . . 95
5.1.3 Use cases . . . 96
5.1.3.1 IoT setting . . . 96
5.1.3.2 Blockchain setting . . . 97
5.2 Homomorphic Encryption with Decryption to Shares . . . 97
5.3 Oblivious Sharing Re-Encryption . . . 99
5.3.1 Intermediate solutions . . . 100
5.3.1.1 Approach 1 - Interactive one-time pad . . . 100
5.3.1.2 Approach 2 - Ciphertext space sampling . . . 101
5.3.1.3 Approach 3 - Plaintext space sampling . . . 103
5.3.2 Secure OSRE based on NTRU for Shamir SS . . . 104

UNIVERSIDAD DE MÁLAGA

5.3.2.1 Making dealer's communication constant . . . 106
5.3.2.2 Security proof . . . 106
5.3.2.3 Implementation . . . 107
5.4 Cost analysis and comparison . . . 109
5.4.1 Theoretical analysis . . . 109
5.4.2 Experimental analysis . . . 112
5.5 Summary . . . 113

6 Applications of MPC 114
6.1 Private Set Intersection in blockchain . . . 114
6.1.1 PSI in consortium blockchains . . . 114
6.1.2 Implementations of PSI in blockchain . . . 116
6.1.2.1 Implementation in Besu . . . 116
6.1.2.2 Design for public blockchain with MPC framework . 117
6.1.3 OSRE for input delivery . . . 119
6.2 Location based services with privacy . . . 120
6.2.1 Contact tracing and crowd counting . . . 121
6.2.1.1 Contact tracing approaches . . . 122
6.2.1.2 Crowd counting . . . 123
6.2.2 Privacy preserving space transformation of location data . . . 128
6.3 Private and accountable transactions in cryptocurrency . . . 131
6.3.1 Technical background . . . 133
6.3.1.1 Blockchain mixers . . . 133
6.3.1.2 Mixnets . . . 134
6.3.1.3 Decentralized mixers . . . 135
6.3.2 A decentralized Bitcoin mixer with selective transparency . . . 136
6.3.2.1 Roles and protocol design . . . 136
6.3.2.2 Security requirements . . . 139
6.3.3 Implementation of proofs and evaluation . . . 140
6.3.3.1 Design of proofs . . . 140
6.3.3.2 Proof of concept implementation . . . 143
6.3.3.3 Integration in Bitcoin . . . 146
6.3.3.4 Evaluation . . . 146
6.4 Summary . . . 148

7 Conclusions 149
7.1 MPC and real deployments . . . 149
7.2 Blockchain and privacy . . . 149
7.3 Contributions . . . 150
7.3.1 Analysis of real world drawbacks . . . 150
7.3.2 Confidential computations in blockchain . . . 151
7.3.3 Efficient and dynamic delivery of private data . . . 151
7.3.4 New applications . . . 152
7.4 Future work . . . 153

A Resumen 155
A.1 Motivación, retos y contribuciones . . . 155
A.2 Análisis de arquitecturas y protocolos en el paradigma de la Computación Segura Multiparte . . . 156

UNIVERSIDAD DE MÁLAGA

A.3 Computación Segura Multiparte en entornos blockchain 157
A.4 Compartición dinámica y compacta de secretos 157
A.5 Aplicaciones de la Computación Segura Multiparte 158
A.6 Conclusiones y trabajo futuro 159

UNIVERSIDAD DE MÁLAGA

List of Figures

1.1 A mind map of the main contributions of this thesis. 7

2.1 Example of a boolean and an arithmetic circuit. 22
2.2 General model for dynamic MPC with ephemeral committees. 28
2.3 High-level representation of blockchain blocks. 31

3.1 Abstract MPC model based on roles. 38
3.2 MPC outsourced models regarding trust domains. 40
3.3 Time (ms) for signature generation in threshold signature schemes. . 46
3.4 SLR methodology for screening. 49
3.5 Classification of building blocks to solve PSI. 52
3.6 OPRF-based PSI from *Hemenway Falk et al.* 55

4.1 Representation of public and private states in a Besu blockchain. . . . 70
4.2 Network topology of Besu and Tessera nodes. 71
4.3 Sequence diagram: create PMT. 72
4.4 Sequence diagram: retrieve private transaction. 72
4.5 Overview of the design for MPC in blockchain using Besu clients. . . 73
4.6 Fields of an extended private transaction. 75
4.7 Functional blocks for Besu with MPC. 77
4.8 Sequence diagram: handling a extended private transaction in Besu . 78
4.9 MPC-based confidential computing framework in blockchain. 82
4.10 Sequence diagram of the framework's workflow. 83
4.11 Race condition issue of MPC in blockchain. 84

5.1 Two settings for secret sharing data. 94
5.2 Comparison of the minimum n needed for scheme A to outperform scheme B ($A = B$), as a function of N_c. 111
5.3 Performance evaluation based on simulations. 112

6.1 Uploading confidential data to a SC using OSRE. 120
6.2 High-level overview of a contact tracing app. 122
6.3 Simulation with 200 people in a 50x50 map with different Δ_s values, speed 1 m/s and 20x20 area queried. 127
6.4 Simulation in a 50x50 map and 40x40 area queried, with different values for device density d. 128
6.5 Notification of location data with MPC proxy. 129
6.6 The relation between privacy and transparency. 133
6.7 Graphical description of a centralized blockchain mixer. 134

UNIVERSIDAD DE MÁLAGA

6.8 Roles and interactions for a mixer with private and relinkable payments. 137
6.9 Performance metrics to compute a proof of compliance 147

UNIVERSIDAD DE MÁLAGA

List of Tables

2.1 Computation and communication costs in OTe. 15
2.2 Comparison between different zk-SNARKs protocols. 19

3.1 Latency comparison (in ms/op) in MPC-based PRFs (from *Grassi et al.*). 44
3.2 Running Time (in seconds) for different PSI protocols and set sizes. . 61

4.1 On-chain storage cost for a UINT32 value. 90
4.2 Computational cost to add and multiply UINT32. 91

5.1 Comparison of the required features for the different solutions. 109
5.2 Asymptotic cost comparison 109
5.3 Length of one ciphertext and re-encryption key with $\lambda = 128$. 110

6.1 RME values obtained from the simulations 128
6.2 Related work comparison. * Unless the whole mixnet cheats. ** As long as 2/3 of the nodes are honest. † If *pk* of a Zether account can be linked to the ETH address, then anonymity is lost. ‡ Only briefly mentioned. **n.a.** Not applies. 135
6.3 Performance of decryption-based and secret-sharing-based ZKPs for transaction compliance. Terminology: *constr.:* number of constraints in the circuit, τ_{min}*:* minimum size of the tau ceremony (setup) needed to compute the circuit, *wt:* witness generation time, and *pt:* proof generation time. All times are indicated in seconds. 147

UNIVERSIDAD DE MÁLAGA

List of Acronyms

AHE Additive Homomorphic Encryption

BFT Byzantine Fault Tolerant

BF Bloom Filter

BLE Bluetooth Low Energy

CA Contract Account

CE Commutative Encryption

CF Cuckoo Filter

CRS Common Random String

DAG Directed Acyclic Graph

DKG Distributed Key Generation

DLT Distributed Ledger Technology

DoS Denial of Service

DRNG Distributed Random Number Generator

ECC Elliptic Curve Cryptography

EOA Externally Owned Account

EVM Ethereum Virtual Machine

FHE Fully Homomorphic Encryption

GBF Garbled Bloom Filter

GC Garbled Circuit

HE Homomorphic Encryption

HEDS Homomorphic Encryption with Decryption to Shares

HSM Hardware Security Module

IPFS Interplanetary File System

ISC Interface Smart Contract

ISP Internet Service Provider

KYC Know Your Customer

LAN Local Area Network

LBS Location Based Service

LSB Least Significant Bit

MAC Message Authentication Code

MHE Multiplicative Homomorphic Encryption

MPC Secure Multi-Party Computation

MSB Most Significant Bit

NI-ZKP Non-Interactive Zero-Knowledge Proof

OKVS Oblivious Key Value Store

OPE Oblivious Polynomial Evaluation

OPRF Oblivious Pseudo-Random Function

OPV Oblivious Punctured Vector

OSRE Oblivious Sharing Re-Encryption

OT Oblivious Transfer

PEP Policy Enforcement Point

PET Privacy Enhancing Technology

PHE Partially Homomorphic Encryption

PMT Private Membership Test

PMT Privacy Marker Transaction

UNIVERSIDAD DE MÁLAGA

PoA Proof of Authority

PoS Proof of Stake

PoW Proof of Work

PRE Proxy Re-Encryption

PRF Pseudo-Random Function

PRG Pseudo-Random Generator

PSI Private Set Intersection

PSO Private Set Operation

PSU Private Set Union

P2P Peer-to-Peer

ROI Return of Investment

SC Smart Contract

SLR Systematic Literature Review

SNARK Succinct Non-Interactive Argument of Knowledge

SRS Structured Reference String

SS Secret Sharing

STS State Transition System

SWHE Somewhat Homomorphic Encryption

TEE Trusted Execution Environment

TRNG Truly Random Number Generator

TTP Trusted Third Party

UTXO Unspent Transaction Output

VOLE Vector Oblivious Linear Evaluation

WAN Wide Area Network

ZKP Zero-Knowledge Proof

UNIVERSIDAD DE MÁLAGA

UNIVERSIDAD
DE MÁLAGA

UNIVERSIDAD
DE MÁLAGA

Chapter 1

Introduction

Protecting data from unauthorized accesses has been a major concern since the Internet started to grow. The ease for attackers to break into the normal functioning of communications protocols and computer systems put on display that security is a key aspect to enhance within the digital space. One of the fundamental pillars of cybersecurity has been cryptography, which central role has led to a large work of formalization and development. Regarding the first applications within the Internet ecosystem, such as the web, the focus was mainly set in protecting data while being sent through networks and also while stored. However, the way that users and services interact between them have shifted nowadays from the very centralized architectures in the original Internet to much more pervasive applications, thanks to technologies such as the cloud, edge, or mobile networks. In addition, services have shifted from a passive receiver model to one that is much more user-centric. It is clear that gathering and using data has become much more important, where some of them are even sensitive or highly personal to users. Therefore, in order to improve security in these scenarios, a new paradigm has arisen: protection of data in usage.

This thesis targets *Secure Multi-Party Computation* (MPC), a wide family of cryptographic protocols that allow a set of parties to jointly compute a function on their private data without exposing anything else than what it can be obtained from the output results. Therefore, these protocols are identified as powerful solutions to enhance privacy in different applications. MPC has been proven to be useful in diverse scenarios, mainly in the last decade where research on it has been more extensive and both its security and performance have been improved. In addition, it introduces new security considerations which are based on the capabilities of attackers to corrupt more than one device at the same time, therefore introducing tools interesting for decentralized scenarios where there are no single points of trust.

The contributions of this thesis are framed within different layers, ranging from new proposals for decentralized architectures or ecosystems where MPC has a central role, to new cryptographic designs or applications based on MPC. In the reminder of this chapter, we present the motivation of this thesis, together with its goals and contributions. Finally, we provide an outline of the following chapters.

UNIVERSIDAD DE MÁLAGA

1.1 Motivation and Scope

Cybersecurity is a central area in the digital space, and has had a large evolution since the Internet started to grow and become a fundamental tool in our day-to-day lives. In particular, cryptography has played a central role in solving many of the problems, mainly related to the unauthorized exposure of data and the enhance of trust in digital systems against failures or intended attacks. In fact, the problem of hiding information from unauthorized observers dates back thousands of years. However, the need for protection in the Internet era has boosted the evolution of cryptographic mechanisms and digital security tools, which have been embraced by formal analysis and standards, in order to establish understandable and universally accepted tools for protection.

Both the problem of external passive observers, called eavesdroppers, and unauthorized modification of information, were mainly solved by symmetric cryptography, where different families of schemes remain as well-known standards nowadays, such as AES, HMAC, etc. In addition, asymmetric cryptography enhanced and dynamized the secure capabilities of digital protocols, mainly introducing new ways of interaction and facilities to instantiate symmetric schemes without such a strong reliance on trusted third parties, at least for the instantiating of secure communication channels. These solutions, when being analyzed from the point of view of how data is being manipulated, fit within two main traditional categories: protection of data in transit and in rest. While the first category solves the problem of communication between different entities in the digital space through secure channels, the second one targets the problem of keeping data secure when stored in digital systems. The latter has been most notorious because of the large amount of data breaches produced in the last decade[1].

In fact, data as an asset to protect is a notion that has become much more present these days. The main reason is the central role that data as acquired, and mostly sensitive and personal identification data. Big data and AI are new technological areas that have emerged with great impact and that allow enterprises to process data and obtain fine grained insights to improve business models. This is different from the first steps in the Internet because its origins were much more passive from the point of view of the user. The original web was mainly a tool for information consulting and used as a long-range showcase, but ecosystem has evolved to a user-centric approach, where users do not only rely on the web in particular, and Internet in general, for consulting third party information, but for a direct use on its own data, which is involved in many day to day processes, such as personal documents storage, online shopping, digital payments, real life task organization, etc. However, the great amount of data that can be collected from users is not only interesting for enterprises to enhance their businesses, but also for hackers, who can obtain great revenues by selling stolen data. This has led users to become the center of a threatened scenario where different actors may try to abuse on them.

[1] `https://www.upguard.com/blog/biggest-data-breaches`

UNIVERSIDAD DE MÁLAGA

1.1.1 Protection of data in usage

Above, we introduced the notions of protection of data in transit and in rest, which have been very well established and standardized. However, motivated by the importance that users' data have acquired, there has appeared a new paradigm called protection of data *in usage*. Apart from the new uses of data in large analysis tools, the evolution of the services consumed in the Internet has modified the way in that data must be protected. Old approaches such as perimeter protection are not enough nowadays, mostly because everything has become more pervasive. Services and applications are not restricted to a single infrastructure now, where a highly-secured perimeter could imply a certain amount of trust in that service or application. New approaches such as the cloud or the edge imply a diffuse line of location for data, at least for users, who must fully trust in providers that their data will not be shared with other service providers or that it will only be used within highly protected environments. All these evolution have contributed to put the focus on data from the point of view of protection, where regulations such as the GDPR try to provide the users with legal guarantees that they will not be abused when interacting with digital services.

It is also clear that the weak point in the chain is data usage, because even if a system is very well designed in terms of security, data will need to be in its plaintext form when being used inside a computation. For that reason, protection of data in usage has emerged as an interesting paradigm that puts the focus on protecting data during its whole life-cycle, instead of relying in the service providers' good intentions to provide secure-enough mechanisms that protect users' data the whole time. Thanks to this new approach, users can share their data in a more reliable way, while not impeding enterprises from obtaining the benefits of data processing.

1.1.2 Decentralization of trust

In parallel with the new advances in data protection, the way users trust digital services is also changing. Trust is a key aspect in the digital space and, although one of the main objectives of cryptography is to minimize trust, having no trust in any entity is virtually impossible. For any service or infrastructure, some amount of trust must be deposited somewhere, otherwise digital services through the Internet could not be used. However, the concept of Trusted Third Parties (TTPs) are less accepted each time. Roughly speaking, a TTP is a fully-trusted entity, external to the user/enterprise area of control. A TTP can range from a service that generates cryptographic keys and delivers them to different users, to a cloud service provider where users upload their data and can perform computations on them. The large amount of data leaks, breaches, and the general feeling of not being able to control what happens to owned data in the digital space, has led to new approaches where trust is decentralized, instead of being held by a single entity. Decentralizing trust implies that more than one entity have to collude in order to be able to abuse on data, and therefore users can rely more on the services provided while still not fully trusting the different providers involved.

From these new approaches, Distributed Ledger Technologies (DLTs) and, more specifically Blockchain, have become the most widely accepted. These technologies target new models of interaction and computation where transparency is the central

UNIVERSIDAD DE MÁLAGA

aspect. Since there is not single party that rules the whole system, there exists a general feeling of security even if a subset of the involved parties become malicious or dysfunctional. Blockchain technologies offer great amounts of accountability and immutability, which can help to increase security in use cases where critical operations are involved, such as digital payments. However, they lack in providing privacy and confidentiality of data, which diminishes their adoption for several use cases where data must be both protected in terms of integrity and confidentiality.

1.1.3 Secure Multi-Party Computation

MPC can be described as a general problem in the literature, but is also widely understood as the set of solutions that solve the general problem. Roughly speaking, the problem considers a set of parties that do not trust each other but want to jointly compute a function on their private inputs without exposing anything else but what can be inferred from the output of the computation[2]. MPC is quite old within the paradigm of modern cryptography and has belong only to theoretical research during many years. However, this has changed during the last decades because it has received great attention from the research community and more realistic and efficient solutions have arisen. The amount of solutions that exist within MPC is quite large and diverse, and selecting one or another normally implies an in depth analysis of the requirements needed and the trade-offs provided by the specific solutions. While some solutions for MPC are somehow flexible and dynamic, they imply large amounts of network communication between the different parties involved in the MPC instance. Others move the load to computation, but may become more difficult to embed in a functional system.

MPC is identified as a solution to the TTP problem. Wihtout MPC, in a scenario where a set of parties have a set of private inputs and want to compute a function of these inputs without exposing them, the only approach possible is to send all the inputs to a TTP, which is trusted to compute the correct output and also not to expose any of the private inputs. MPC allows to replace the TTP by a protocol that fulfills the same role, but with cryptographic guarantees. In such setting, the parties provide their inputs to the protocol in a protected way that also enables to compute on them, and where the output is the only plaintext value provided in the end.

Despite such a solution seems very promising, instantiating protocols where there is no single point of trust, even for coordination, is very difficult. In fact, despite MPC has become much more practical the last decade, mainly in terms of efficiency, it still presents several difficulties at the time of deploying truly secure instances where no single point has the control (either total or partial) of the different parties involved. In fact, MPC has also been proposed by different works as a solution to protect outsourced environments for computations, such as a cloud instance, but this emphasized more the fact that a TTP is still present. Even when an outsourced MPC solution can help in protecting data in usage from external adversaries, at least at the level of infrastructure), the manager of the infrastructure (i.e., the service provider) is still a TTP.

[2]This is very related to a close definition called *Secure Function Evaluation*.

UNIVERSIDAD DE MÁLAGA

Despite their limitations, MPC protocols are very interesting because they bring together private computations and decentralization (if deployed correctly), which are not straightforward to achieve together. It was previously said that blockchain technologies offer new trust assumptions by distributing it among a set of parties, but they do not provide privacy. However, despite MPC is not intended for integrity or immutability, as it happens with blockchain, it enables privacy within a set of mutually distrustful parties. We identify this aspect as one that brings closer both technologies, and will be a major point of discussion along this thesis.

1.2 Goals and Contributions

The previous sections have provided an overview of the initial motivation of this thesis. We identified MPC as a cryptography subfield with plenty of research left, mainly regarding usability and integration in new environments. Since MPC is very wide, it is key to understand it well from the point of view of applicability, which is not a straightforward notion for all types of MPC. We noticed that it is common in the research literature of MPC to start from a couple of assumptions to develop new protocols, but these assumptions may not always be realistic or practical. In addition, from all the approaches that are being proposed to provide some sort of privacy and/or confidentiality in blockchain systems, we also identified MPC as a key technology that, even if already noticed by the research community to be useful for blockchain, has not been covered in terms of a practical integration and deployment in real systems. Following this line of focus on the practical side, we also noticed that there exist generalizations that are normally done within MPC protocols that are not addressed from the point of view of the user, and mainly those with a limited amount of resources available. Despite we identified outsourced MPC as a path to follow, mainly because of the ease of adoption for a wider range of devices and applications, we also noticed that instantiating it is not trivial, and some considerations must be taken into account. In addition, we noticed that even when the MPC is done in an outsourced setting, and therefore avoiding the input providers to hold the burden of an intense computing process, private inputs must still be provided to computing nodes in a secure way, and this can introduce significant costs for the users and also a lack of dynamism regarding the input data sets. Following these new lines of integration and interaction with MPC systems, we also identified new applications that can benefit from having secure and decentralized computation instances, therefore enhancing the amount of computation that can be done in outsourced scenarios, in general, and in blockchain applications, in specific. In the end, these new approaches for MPC and blockchain with confidential computations allow the rise of new models which are more dynamic and secure.

With these goals in mind, the following list gathers the main contributions of this thesis, which address the goals from different levels of generalization within the MPC ecosystem:

- We review the basic and general notions for MPC, as a general purpose problem, and the main constructions that have been proposed in the literature. We also introduce blockchain technology as an enabler for decentralization and trust.

UNIVERSIDAD DE MÁLAGA

- We review the MPC paradigm from a point of view of practical usability, making emphasis in the characteristics to consider when deploying these protocols in real applications. Some of these aspects are trust relations, interaction topology or specific considerations about protocols, such as efficiency or security models.
- We analyze the case of Private Set Intersection (PSI) in depth, which is one of the most important special-purpose protocols in the MPC paradigm, mainly because of the large amount of applications it enables. We develop a systematic analysis that provides a taxonomy of the most important recent PSI contributions and make emphasis in efficiency and the real applicability it presents in different scenarios or applications.
- We examine the relations that exist between MPC and the blockchain technology, providing some insights about how each technology can benefit the other. In addition, we provide two solutions to integrate MPC capabilities in blockchain, one design and implementation for Hyperledger Besu, following its private transaction model, and one framework for smart contract-based public blockchains such as Ethereum, where the MPC is delegated and the result is publicly verifiable.
- We provide a new cryptographic construction named Oblivious Sharing Re-Encryption (OSRE) that allows an input provider to secret share data to a committee of holders with the delivery of a single ciphertext. We identify this construction as a useful solution to share data more efficiently in certain scenarios such as IoT or blockchain, and which also provides flexibility to first encrypt data and later select the committee for data to be shared with. We note this as relevant within MPC applications because most of the general-purpose MPC frameworks rely on Secret Sharing techniques and simply assume the existence of confidential channels between the data provider and the computing committee, without taking into account the cost for it.
- We propose several applications that fit both in blockchain and MPC. The first one is the execution of PSI in blockchain, mostly in consortium ones, where different and mutually distrustful organizations can benefit from each other's data through a confidential computation using MPC. In particular, PSI has several applications that fit well within such a collaborative ecosystem, such as medical data. The second application targets the protection of Location Based Services (LBS) in general, and an application for privacy-preserving crowd counting in pandemic scenarios in particular. Finally, the last application provides a solution for Bitcoin blockchain mixers that preserve privacy of the user's transaction but also enables selective disclosure of payment data with verifiability and compliance.

1.3 Thesis outline

In this chapter, we have begun by introducing the motivation of this thesis, which is enabling systems where computations can be done on data provided by several users but while ensuring protection during its entire life cycle. That means not only protecting data in rest or in transit, but also on use. From the set of technologies that

UNIVERSIDAD DE MÁLAGA

fall into the category of PETs, we identify MPC as a very interesting one, which is also dynamic and highly secure. However, MPC still lacks research in terms of ease of use when being deployed in real scenarios. In addition, we also identify MPC as one of the technologies that can offer the most to blockchain scenarios, which are a very hot topic nowadays to achieve secure, resilient, and publicly verifiable infrastructures for data anchoring, but lack privacy or confidentiality. This has led us to consider the integration of these two technologies, as well as new efficient and easy-to-use ways of interacting with them, and new applications that can benefit from them. Before outlining the content of the chapters, Figure 1.1 provides a mind map where the main areas of contribution of this thesis are shown. A key remark is that MPC is addressed from different layers in the network stack, i.e., protocol analysis, integration and implementation, cryptographic contributions and applications, and that these layers coincide with different chapters of this thesis, as shown in the figure.

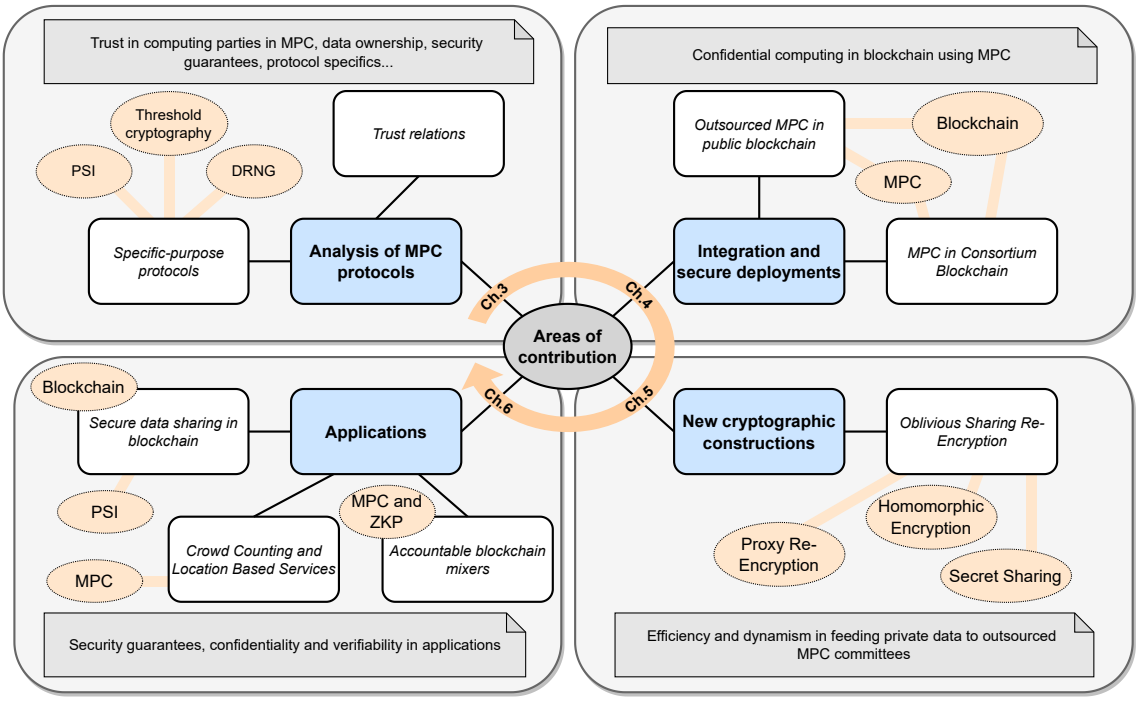


Figure 1.1: A mind map of the main contributions of this thesis.

Before delving into the proper contributions of this thesis, Chapter 2 overviews basic concepts, definitions, and protocols for MPC and blockchain. MPC is very wide, so we cover the main basic building blocks that enable it, and then the main contributions in the literature to the general-purpose MPC space. Within this, we cover the main foundational work, the standard frameworks which are most used nowadays, and some new approaches that target a better dynamism and security. Regarding blockchain, we cover the main definitions from an abstract point of view and some characteristics that can be present for different blockchain instances. Finally, we introduce the problem of lack of privacy in blockchain, together with some of the main solutions that have been provided by the research community.

Prior to introducing our specific solutions for confidential computations, new constructions, and applications, it is fundamental to understand the diversity that exist

UNIVERSIDAD DE MÁLAGA

in MPC protocols, and also their relation with the concept of trust. Chapter 3 discusses about these aspects, where trust is a central axis that is sometimes simplified. This happens in many contributions about MPC that just assume a secure deployment of MPC nodes, without even specifying who owns control of each node and how they are effectively deployed. In addition, we cover some of the main specific-purpose MPC protocols that serve as basic building blocks to many infrastructures where trust is distributed. Finally, we identify one of these protocols, namely Private Set Intersection (PSI), as one of the most researched. This is mainly because of the wide range of applications it presents, and also because cryptographic constructions specifically designed for PSI imply a better performance than general-purpose ones. Since we identified that the literature in PSI is very extensive and there were no research work gathering PSI proposals in detail, we performed a Systematic Literature Review (SLR) of PSI with a focus on its practicability, which is also included in Section 3.3.

In Chapter 4 we delve in the specific problem of providing confidentiality in blockchain applications, and address it first by discussing about the relations that exist between MPC and blockchain technologies. After analyzing the characteristics that facilitate the integration of these technologies, but also the points of conflict they present, we introduce two solutions to achieve confidential computations in blockchain by relying on MPC, in Sections 4.3 and 4.4, respectively. The first solution provides a design to embed MPC capabilities into Hyperledger Besu nodes, which are Ethereum clients that allow to deploy both public and private blockchains, and that already present some sort of privacy in transactions. We target a solution that minimizes the modifications in the way that standard private transactions are used in Besu, and that allow to interact with MPC instances through calls to smart contracts. In addition, this design targets a setting where each Besu node is a participant of the MPC instance. This differs from the second solution provided in this chapter, which targets an outsourced MPC model in public blockchains, and where MPC nodes are selected from a pool of special available nodes, different from the users that deploy and consume decentralized applications. This solution also relies on smart contracts to specify the computations to be carried on confidential data, and provides a mechanism for public verifiability of the results, even when the inputs are never disclosed.

In Chapter 5, the focus is shifted from the architectural point of view to the specific cryptographic constructions. More specifically, we address the problem of providing confidential inputs to MPC committees that rely on protocols where Secret Sharing is their core building block. We notice that the dealing phase of a secret shared value always implies to establish n independent confidential channels from the dealer to the MPC committee, and therefore we propose a new general approach called Oblivious Sharing Re-Encryption (OSRE) that allows a dealer to deliver shares to a committee of holders by providing a single ciphertext, which reduces communication to a constant cost. In this chapter we define OSRE, provide different intuitions and intermediate results from where it can be addressed, and finally provide a concrete construction based on the NTRU encryption scheme, which relies on its properties of being additively homomorphic and having a solution for Proxy Re-Encryption (PRE). We also provide an implementation and a detailed evaluation of costs.

Regarding Chapter 6, it is devoted to provide different applications which are mainly

UNIVERSIDAD DE MÁLAGA

based on MPC and blockchain technologies, which allow to maintain or even expand their usability while ensuring a proper level of security and privacy of the involved data. More specifically, we provide three applications. The first one, provided in Section 6.1, targets PSI in consortium blockchains, which presents several use cases where one of the main benefited is the medical one. The second one, in Section 6.2, is related with Location Based Services (LBS), where MPC is used as a tool to protect users' location data. More specifically, we provide an application for crowd counting in pandemic scenarios, based on contact tracing. Finally, the third application is introduced in Section 6.3, and targets cryptocurrency private payments by using a decentralized blockchain mixer. Our solution provides reversible unlinkability and proofs of compliance to ensure that transactions can be selectively disclosed to specific viewers while ensuring protection against the public view.

Finally, Chapter 7 summarizes the contributions of this thesis and gathers some lines of future work and open problems that can be solved with further research.

1.4 Publications and Funding

The work in this thesis has led to several publications in international journals and conferences. We provide a list of these contributions below, classified by the type of publication.

Articles in JCR Journals

- D. Morales, I. Agudo, and J. Lopez. Private set intersection: A systematic literature review. In *Computer Science Review*, vol. 49, no. 100567, Elsevier, 2023. JCR Impact Factor 2023: 13.3.
- D. Morales, I. Agudo, and J. Lopez. Toward a Framework for Cost-Effective and Publicly Verifiable Confidential Computations in Blockchain. In *IEEE Communications Magazine*, Early Access, 2024. JCR Impact Factor 2023: 8.3.
- D. Morales, I. Agudo, and J. Lopez. A lightweight mechanism for dynamic secret-sharing of private data by constrained devices. In *IEEE Internet of Things Journal*, Forthcoming, 2025. JCR Impact Factor 2023: 8.2.

International Conference papers

- D. Morales, I. Agudo, and J. Lopez. Integration of MPC into Besu through an extended private transaction model. In IEEE International Conference on Metaverse Computing, Networking and Applications (IEEE MetaCom), pp. 266-273, 2023.
- D. Morales, I. Agudo, and J. Lopez. Real-time Crowd Counting based on Wearable Ephemeral IDs. In 19th International Conference on Security and Cryptography (SECRYPT), pp. 249-260, 2022.

Papers under submission at this stage

- D. Morales, I. Agudo, and J. Lopez. Zero-Knowledge Bitcoin Mixer with Reversible Unlinkability. In Blockchain: Research and Applications Journal.

UNIVERSIDAD DE MÁLAGA

JCR Impact Factor 2023: 6.9 (Currently: *minor revision*).

Funding

For the realization of this thesis, the author has been funded by the Spanish Ministry of Education under the National F.P.U. Program (FPU19/01118).

UNIVERSIDAD DE MÁLAGA

Chapter 2

Preliminaries

This chapter provides a brief overview of most of the basic cryptographic tools relevant to the rest of the thesis. In particular, we analyze the core building blocks that enable MPC protocols, together with the main constructions to achieve MPC, either the most important foundational works and the most widely used modern protocols. This chapter also introduces blockchain technologies and their main characteristics, together with the main problems they pose in terms of lack of privacy and some of the solutions provided by the literature.

2.1 Secure Multi-Party Computation

MPC is a broad area with many different strategies to be solved. In this section, it is introduced from the general to the specific. The main core results introduced in this section come from two broad compilation works about MPC [1, 2].

2.1.1 Problem formulation and main characteristics

Secure Multi-Party Computation (MPC) is a general problem in the area of data security and cryptography. This problem formulates the scenario where a set of n parties $\{P_1, ..., P_n\}$, each P_i with its own private input x_i, want to securely compute some function $f(\cdot)$ on their inputs without exposing to the other parties anything else than the output $y = f(x_1, ..., x_n)$. Since no party trust the others, some of them may be assumed to behave in a selfishly way, either trying to expose other parties inputs or forcing the protocol to behave wrongly. There exist some illustrative examples in the literature, where one of the most extended ones is the *average salary problem*. It basically sets a group of workers that want to learn what is the average salary of the group but without revealing each individual salary to anyone (worker or not), so they can learn if they are receiving too little money with respect to the rest. It is important to remark that an MPC solution must not expose anything else than the output of the functionality, but it does not provide protection against what can be inferred from the output. For example, in the average salary case, if there are only two workers, each of them can learn the salary of the other one by using its own salary and the average result.

The definition above captures the essence that the output y is given to all parties

UNIVERSIDAD DE MÁLAGA

once computed. However, MPC also allows partial outputs to be calculated so that what each participant receives is not the same as what others receive, i.e., $(y_1, ..., y_n) = f(x_1, ..., x_n)$. Then, the case with a common output is just a particular case where for all $i \in [n]$, $y_i = y$.

MPC protocols can present a different set of fulfilled properties depending on the specific setup and application. However, there are two main properties that must be fulfilled by any MPC protocol to be useful: *correctness* and *privacy*. Correctness ensures that the output corresponds with the correct output as if the function $f(\cdot)$ was locally computed by a Trusted Third Party (TTP) using the plaintext inputs, while privacy ensures that private inputs (and intermediate private values) are not disclosed to unauthorized parties, whether they are involved in the MPC or not. However, how these properties are ensured is very dependent on the capabilities that are given to the adversary, which is the abstract entity that controls all the dishonest parties in the protocol. Modeling the set of dishonest parties as D, we have that $D \subset \{P_1, ..., P_n\}$. There are two main corruption models that determine the behavior of the corrupted parties:

Semi-honest or passive adversary. This adversary is assumed to see the internal state of the corrupt parties, i.e., messages delivered and received, and the computation performed. However, these parties must follow the protocol as it is described. Based on the state seen, the adversary tries to infer more than it is supposed to.

Malicious or active adversary. This adversary has the capability of modifying the behavior of the corrupt parties with respect to the description of the protocol, e.g., by making them send incorrect messages, not responding when required, or computing wrong data. This type of corruption is more powerful than semi-honest, and achieving security against it is more difficult and resource-demanding.

The analysis of how many parties can be corrupted (and sometimes, which specific subset) is of paramount importance, because it conditions what can be achieved by the MPC protocol. A common way to formalize this in MPC is by *adversarial access structures*, which are mathematical objects that defines the specific subsets of parties that can perform specific actions on secrets, such as revealing them. One of the most common ways of defining these subsets in MPC proposals is by using *threshold access structures*, which homogenizes the capability of accessing or manipulating secrets to any sufficiently large subset. More specifically, given P the total set of parties involved in the MPC, with $|P| = n$, a (t, n)-access structure, with $t \leq n$, is one where any subset $A \subset P$ with $|A| \geq t$ is granted with these capabilities of accessing or modifying secrets. In general purpose theoretical solutions for MPC, there are three general settings used to capture the threshold security that protocols can achieve by using a specific approach. These are, from more to less restrictive: two-thirds honest majority ($t < n/3$), honest majority ($t < n/2$), and dishonest majority ($t < n$).

Regarding privacy, since MPC typically involve many parties interacting between them, security proofs are normally formulated in terms of indistinguishability, and more specifically using the *simulator* strategy in the *real/ideal* model. In such a model, the protocol is analyzed in two scenarios: in the real one, the parties interact

UNIVERSIDAD DE MÁLAGA

between them following the specification of the MPC design, while in the ideal model, the function is executed by a TTP and the simulator provides the view to the adversary as if it was interacting with real parties. Then, the protocol is said to be secure if the adversary has no advantage more than 1/2 in distinguishing if it is the real or the ideal model execution which it is interacting with. Designing simulators to prove security is an interesting area of provable security, where a nice starting point is the work of Lindell [3]. However, it is not covered in this thesis as it is out of scope.

Another important aspect to consider for privacy guarantees is the type of security to be addressed. Achieving exactly no more advantage than 1/2 is very hard and sometimes, the capacity of the adversary can be decreased in order to achieve secure protocols more easily. Therefore, three main models are normally considered:

Perfect security. The real and the ideal execution follows exactly the same distribution, and therefore the advantage of the adversary in distinguishing is exactly 1/2.

Statistical security. In this case, the distributions produced by the real and the ideal execution are statistically close, i.e., there is a very small probability that the adversary guess the correct execution with more chance than random guessing.

Computational security. In this case, the adversary is computationally bounded, i.e., the distributions of the real and the ideal executions are indistinguishable as long as the adversary does not run faster than the specified bound. Adversaries are normally bounded to run in Probabilistic Polynomial Time.

As previously stated, correctness and privacy are the two main properties for MPC protocols, but there are others. Regarding the output, apart from correctness, it is also interesting to know which guarantees of receiving the actual result the different parties have, because it may happen that only some parties obtain the result and prevent the others from receiving it. There are three main options for this:

Guaranteed output delivery. All parties involved in the protocol are ensured to receive the output.

Fairness. The adversary can cause the honest parties to not obtain the output, but if this happens, the malicious parties do not obtain it either.

Security with abort. This is the most relaxed scenario, because the adversary can learn the output while preventing the honest parties from obtaining it.

These notions are related to the type of adversary. For example, when the adversary is passive, the result is always correct and guaranteed output delivery is obtained, because corrupt parties will follow the protocol exactly as it is defined. On the contrary, when the adversary is active, different strategies must be used to ensure guaranteed output delivery or fairness. However, achieving these properties is not always easy or possible.

UNIVERSIDAD DE MÁLAGA

2.1.2 MPC basic primitives

The previous section has introduced MPC from a general perspective, without detailing the specific constructions that exist. In fact, MPC is such a broad area of cryptography that different technologies exist to solve it, with different assumptions, guarantees and performance trade-offs. Before delving into specific protocols that solve MPC, this section introduces some of the main cryptographic building blocks that serve as basic functional units to construct the main MPC protocols that have been proposed in the literature.

2.1.2.1 Oblivious Transfer

Oblivious Transfer (OT) is a 2-party protocol that does not seem to be very useful by itself, but turns to be a key and fundamental building block for more complex MPC protocols. Its base form is shown in Functionality 1, and was introduced by Even et al. [4] and Rabin [5].

Functionality 1: Oblivious Transfer

Sender's private input: two messages m_0 and m_1.
Receiver's private input: a choice bit $c \in \{0, 1\}$.
Sender's output: nothing.
Receiver's output: the selected m_c.

The functionality from Functionality 1 receives the name of "*1-out-of-2*" or $\binom{2}{1}$-OT, and its fundamental property is that the sender does not learn which message is being received by the receiver, and the receiver does not learn anything about the message m_{1-c} that was not chosen.

After those initial works, OT was generalized to $\binom{n}{1}$-OT, where the sender owns n messages and only one can be selected by the receiver, and $\binom{n}{k}$-OT, where the receiver can select $k < n$ messages [6, 7].

OT has been extensively researched with complex and efficient protocols, but for the sake of completeness, we introduce in Protocol 1 a very naive way of achieving $\binom{2}{1}$-OT proposed by Chou and Orlandi [8] under the assumption that the DLP is hard.

OT typically implies public key cryptography in its core, therefore it may turn somehow heavy if the set of input messages owned by the sender is large. Generalizing to estimate costs, it can be approximated to one exponentiation per OT instance; however the work by Naor and Pinkas [9] shows that one exponentiation can be consumed to achieve some OT, using a trade-off technique that reduces computation but increases communication cost.

However, what really made OT efficient for large scale protocols (and specially in terms of computational cost) is OT extensions (OTe), such as the work by Ishai et al. [10]. Given that OT_l^μ means performing μ OT instances with l-bit items being sent, and OTe protocol allows to compute OT_l^μ from OT_κ^κ plus some cheaper computations such instances of Pseudo-Random Generators (PRG) and Hash functions. Regarding

UNIVERSIDAD DE MÁLAGA

Protocol 1: Simplest OT

1. The sender owns two messages m_0 and m_1, and the receiver selects a choice bit $c \in \{0, 1\}$.
2. The sender randomly samples $a \leftarrow \mathbb{Z}_p$ and sends $A = g^a$ to the receiver.
3. The receiver randomly samples $b \leftarrow \mathbb{Z}_p$ and computes:
   - $B = g^b$, if $c = 0$,
   - $B = Ag^b$, if $c = 1$,

   and then sends B to the sender and computes a symmetric key $k_r = H(A^b)$, with $H(\cdot)$ a cryptographically secure hash function.
4. The sender computes the two possible symmetric keys:
   - $k_0 = H(B^a)$,
   - $k_1 = H((B/A)^a)$,

   and then sends the encryption of the two messages to the receiver:
   - $e_0 \leftarrow Enc_{k_0}(m_0)$,
   - $e_1 \leftarrow Enc_{k_1}(m_1)$.
5. Finally, the receiver can receive m_c by decrypting $Dec_{k_r}(e_c) = m_c$.

notation, $\mu = \kappa^c$, κ is the OTe security parameter, and $c > 1$ is a fixed constant. Note that since $\mu > \kappa$, OTe reduces the amount of exponentiations performed.

Table 2.1: Computation and communication costs in OTe.

Protocol		Costs	Security Parameter
[10] IKNP $\binom{2}{1}$-OT_l^μ	Comp.	$\text{OT}_\kappa^\kappa + 2\kappa PRG_\mu + 2\mu HASH$	κ
	Comm.	$2\mu\kappa + 2\mu l$	
[11] KK $\binom{n}{1}$-OT_l^μ	Comp.	$\text{OT}_{\kappa'}^{\kappa'} + 2\kappa' PRG_\mu + \mu n HASH$	$\kappa' \approx 2\kappa$
	Comm.	$2\mu\kappa' + n\mu l$	

Finally, Table 2.1 shows some specific costs of OTe protocols, where the value for the security parameter is typically $\kappa = 128$ and PRG_μ means that the PRG generates μ-bit items. We note that, despite κ' is twice as large in KK [11] than in IKNP [10], it can be amortized when performing OT with short secrets because it achieves $\binom{n}{1}$-OT directly instead of $\binom{2}{1}$-OT, with $n \leq \kappa$.

There are other variations of OT, such as Random Oblivious Transfer (ROT), where the functionality does not receive a set of inputs from the sender, but it outputs a set of pseudo-random values to him. Another variation is Correlated Oblivious Transfer (COT), which is like a ROT but the outputs from the functionality to the sender are correlated by some parameter provided as an input by the sender.

2.1.2.2 Secret Sharing

Secret Sharing (SS) is another fundamental primitive in MPC, since most extended protocols are based on it. While OT served as the base for 2-PC, SS allowed to extend protocols to the n-party setting, with $n > 2$; a great example is the first

UNIVERSIDAD DE MÁLAGA

n-party protocol by Golderich et al. [12], that extended the Garbled Circuit (GC) approach using SS. We will introduce the basic notions of SS in this section, but we refer to Beimel's work [13] for in depth details.

An SS scheme allows a dealer, i.e., the secret owner, to share a secret with a set of n holders $P = \{P_1, ..., P_n\}$, such that only an allowed subset of holders (defined by an access structure) can reveal the secret. It is formally defined in Definition 1.

Definition 1 (SS scheme)**.** *A Secret Sharing scheme is composed of a tuple of algorithms* (Share, Reconstruct[1])*:*

- $\mathsf{Share}(s, n, A) \rightarrow \{s_1, s_2, ..., s_n\}$*. On input a secret s, a number of shares n, and an access structure $A \subseteq P$, it outputs a set of shares, where s_i is P_i's share. Shares are normally delivered through secure channels to each party.*
- $\mathsf{Reconstruct}(\{s_j\}_{j \in A}) \rightarrow s$*. On input a valid subset of shares according to the access structure A, the algorithm outputs the original share s.*

An SS scheme must fulfill two main properties: *correctness* and *privacy*. The first one captures the notion that any authorized subset of holders, i.e., any subset $A \subseteq P$ that belongs to the access structure, must be able to reconstruct the secret with probability 1 (or large probability in the statistical setting). The second notion implies that an unauthorized set cannot learn anything about s from their shares, i.e., given two unauthorized set of shares S_a and S_b of two secrets a and b, S_a is indistinguishable from S_b.

Most extended SS schemes in MPC are those linear and with threshold access structures. We define a (t, n)-SS scheme as one where at least $t+1 \leq n$ shares are needed to reconstruct the secret s. We use $[\![s]\!]_t$ to represent a secret shared value s with threshold t. Through the rest of this document, $[\![s]\!]$ will imply a threshold t, unless specified otherwise. On the other hand, it is somehow required for MPC that shares have some sort of homomorphism, i.e., operations can be performed on shares such that the results are mapped to the secrets once reconstructed. Fortunately, this is implied by linear SS, since linear reconstruction implies that linear operations applied on individual shares, such as addition and scalar multiplication, are translated to the secret once reconstructed.

A basic full-threshold (i.e., $t = n$) and linear SS scheme is additive SS, where a secret $s \in \mathbb{F}_p$ is split in a set of shares $\{s_1, ..., s_n\}$ such that $s = \sum_{i=1}^{n} s_i$. Notice that given $n-1$ shares, the secret cannot be guessed with probability larger than $1/|\mathbb{F}_p|$ and, since any possible result could be valid, the scheme is information-theoretically secure. A common way of generating these shares is by sampling $n-1$ random shares from $\mathbb{F}_p$ and computing the last share as $s_n = s - \sum_{i=1}^{n-1} s_i$. Finally, Equation 2.1 shows that the scheme is additively homomorphic thanks to the associative property (this implies also scalar multiplication), i.e., that local shares x_i, y_i of two secrets x, y can be added together such that they finally reconstruct the secret $x + y$. Additive SS is not multiplicatively homomorphic, but we will show in Section 2.1.3 that there are tricks to perform multiplications on such shares.

$$x + y = (x_1 + ... + x_n) + (y_1 + ... + y_n) = (x_1 + y_1) + ... + (x_n + y_n) \tag{2.1}$$

[1] This is also known as Open.

UNIVERSIDAD DE MÁLAGA

On the other hand, Shamir SS [14] is the most extended linear scheme. It is based on polynomial interpolation and is also information-theoretically secure. The basic idea is that a polynomial of degree k can encode a secret and that at least $k+1$ points of the polynomial are needed to correctly interpolate the polynomial, otherwise nothing can be learned from the points. This leads to the following scheme: given a polynomial $P(X) = \sum_{i=0}^{k} p_i \cdot X^i$, p_0 is set to be the secret s, and the rest of coefficients $\{p_1, ..., p_k\}$ are randomly sampled. Then, given the set of fixed indices $I = \{id_{P_1}, ..., id_{P_n}\}$ for the holders, the evaluation $P(id_{P_i})$ outputs the share for P_i. Equation 2.2 shows how homomorphic addition is fulfilled by linear addition between polynomials. Homomorphic multiplication is also supported, since polynomial multiplication of $P(X)$ and $Q(X)$ leads to a polynomial $R(X)$ where the degree 0 coefficient (the result secret) is the product of the original secrets, i.e., $r_0 = p_0 \cdot q_0$. However, given k_p the degree of $P(X)$ and k_q the degree of $Q(X)$, the degree of $R(X)$ will be $k_r = k_p + k_q$. This implies that $k_r + 1$ points are needed to reconstruct the secret, therefore each multiplication of Shamir SS values imply a larger set of holders. There exist different techniques for Shamir SS multiplication that pursue a degree reduction step where, if k is the fixed degree for secret encondings, the polynomials that result from multiplication are transformed from a $2k$ to a k degree polynomial.

$$a + b = \sum_{i=0}^{k} a_i \cdot X^i + \sum_{i=0}^{k} b_i \cdot X^i = \sum_{i=0}^{k} (a_i + b_i) \cdot X^i \tag{2.2}$$

2.1.2.3 Commitments

A commitment scheme is a 2-party protocol between a sender and a verifier, which is formally introduced in Definition 2.

Definition 2 (Commitment scheme)**.** *A commitment scheme is composed of a tuple of algorithms* (Commit, Verify)*.*

- $\mathsf{Commit}(x) \rightarrow c$*. On input a secret value x, the algorithm outputs a commitment c that does not expose anything about x.*
- $\mathsf{Verify}(x, c) \rightarrow \{0, 1\}$*. On input the secret value x and its commitment c, the algorithm outputs* 1 *with high probability if x generated c.*

A commitment scheme must fulfill two main properties: *hiding*, which ensures that c does not expose anything to the verifier without knowledge of x, and *binding*, which ensures that, once committed, the sender will not be able to find a different x' that correctly verifies c in the opening phase. These schemes are very useful overall to achieve maliciously secure MPC protocols, since binding allows to anchor data in some phases of the protocols such that they can be verified later.

Commitment schemes arise in different forms, such as commitments for bits, for integer data, or even for vectors. They are achieved by some sort of one-wayness, e.g., through one-way functions or permutations. Therefore, commitments can be achieved from hash functions [15], where a basic construction provided by a secure hash function H with preimage resistance (one-wayness) would be $c = H(x, r)$, where r is a randomly chosen value that avoids chosen plaintext attacks. Other

UNIVERSIDAD DE MÁLAGA

schemes have been proposed from different assumptions such as RSA, DLP [16], or CRS [17].

A very interesting and widely used commitment scheme is Pedersen's [18], which is based on the DLP. Given a secret x, and two random public generators g and h of a group G of large order q, a Pedersen commitment is generated as $Comm(x, r) = g^x h^r$, with $r \in \mathbb{Z}_q$. Pedersen commitments have the attractive property that they are additively homomorphic, such as shown in Equation 2.3.

$$Comm(x_1, r_1) \cdot Comm(x_2, r_2) = (g^{x_1} h^{r_1}) \cdot (g^{x_2} h^{r_2}) = \\ = (g^{x_1+x_2} h^{r_1+r_2}) = Comm(x_1 + x_2, r_1 + r_2) \quad (2.3)$$

2.1.2.4 Zero-Knowledge Proofs

A ZKP protocol allows a prover to convince a verifier that a private witness w and a public statement x belong to a relation R, i.e., that the statement is true but without revealing the private data in the witness. There are many different schemes for ZKP, from where the oldest proposals are interactive protocols between the prover and the verifier. However, in this section we will focus on Non-Interactive ZKPs (NI-ZKPs), which accomplish the same task via a single message sent from the prover to the verifier. These are particularly interesting for many applications (e.g., blockchain) and have received a lot of attention in recent days. A formal definition of NI-ZKP protocol is presented in Definition 3.

Definition 3 (NI-ZKP scheme)**.** *A NI-ZKP scheme is composed by a tuple of algorithms* (Setup, Prove, Verify)*.*

- $\mathsf{Setup}(1^\lambda) \rightarrow crs$*. On input a security parameter* λ*, it outputs a common random string* crs*.*
- $\mathsf{Prove}(crs, x, w, R) \rightarrow \rho$*. On input* crs*, a public statement* $x \in L$*, a private witness* w*, and a relation* R*, it outputs a proof* ρ*.*
- $\mathsf{Verify}(crs, x, R, \rho) \rightarrow \{0, 1\}$*. On input* crs*, the public statement* x*, the relation* R*, and the proof* ρ*, it outputs 1 with high probability if the proof* ρ *was correctly generated with* (crs, x, R) *on some unknown* w*.*

ZKPs must fulfill three main properties, which are *completeness*, *soundness*, and *zero-knowledge*. Completeness (see Equation 2.4) means that, given a true statement for R, an honest prover with a valid witness should always convince the verifier. As for soundness (see Equation 2.5), it means that, given a false statement for R, it is very hard to compute a valid proof. Regarding zero-knowledge, it means that the proof reveals nothing about w except the statement's veracity.

$$Pr\left[\mathsf{Verify}(crs, x, \rho) = 1 \middle| \begin{matrix} crs \leftarrow \mathsf{Setup}(1^\lambda) \\ \rho \leftarrow \mathsf{Prove}(crs, x, w) \end{matrix}\right] = 1 \quad (2.4)$$

$$Pr\left[\begin{matrix} \mathsf{Verify}(crs, x, \rho) = 1 \\ \wedge (x, w) \notin R \end{matrix} \middle| \begin{matrix} crs \leftarrow \mathsf{Setup}(1^\lambda) \\ \rho \leftarrow \mathsf{Prove}(crs, x, w) \end{matrix}\right] = \frac{1}{poly(\lambda)} \quad (2.5)$$

UNIVERSIDAD DE MÁLAGA

Table 2.2: Comparison between different zk-SNARKs protocols.

Protocol	Setup	Prover work	Proof size
Groth16 [19]	SRS (circuit specific)	$O(n+m)$	$2\mathbb{G}_1 + 1\mathbb{G}_2$
Sonic [20]	SRS (universal)	$O(n\ log\ n)$	$20\mathbb{G}_1 + 16\mathbb{F}_q$
Marlin [21]	SRS (universal)	$O(m\ log\ m)$	$13\mathbb{G}_1 + 8\mathbb{F}_q$
Plonk [22]	SRS (universal)	$O((n+a)\ log\ (n+a))$	$7\mathbb{G}_1 + 6\mathbb{F}$
Fractal [23]	CRS (transparent)	$O(M\ log\ M)$	$O(M)\mathbb{F}$
Bulletproof [24]	CRS (transparent)	$O(n\ log\ n)$	$O(log\ n)\mathbb{G} + 5\mathbb{Z}_p$

There exist quite a large literature for ZKP schemes but, for NI-ZKP, those that have attracted the attention of most researchers are Succinct Non-Interactive Arguments of Knowledge (SNARKs), which also fulfill the *succinctness* property, i.e., that the crs and proof size are $poly(\lambda)$, and the verifier executes in $poly(\lambda + |x|)$ time, with λ being the security parameter. Table 2.2 presents a comparison between some of the most famous zk-SNARKs schemes. Regarding notation, m stands for the number of wires, n for the number of multiplication gates, a is the number of addition gates, and M is the number of non-zero coefficients in the matrices used by the scheme.

Table 2.2 focuses on the asymptotic cost of the prover and the size of the proof generated. A key aspect to consider first is the setup used, because there exist different types that conditions the way that the proofs can be generated. More specifically, there are three types of setups: (1) *trusted setups*, which generate secret randomness (toxic waste) to create a circuit specific Structured Reference String (SRS), (2) *universal setups*, which also require toxic waste but allow the reuse of the SRS with arbitrary circuits, and (3) *transparent setups*, which create a public Common Reference String (CRS) that does not utilize toxic waste. The problem with trusted setups is that they must be executed each time that a different circuit, i.e., the argument inside the ZKP, is to be proven. Since setups are normally quite expensive in terms of resources, this may pose a problem for applications where the circuit to be proven is not fixed. In such cases, universal or transparent setups are better options, but at the expense of larger proofs.

2.1.2.5 Homomorphic Encryption

Homomorphic Encryption (HE) schemes are those encryption schemes that allow a third party to perform some computation on ciphertexts, without decrypting them. We will introduce some of the main results and characteristics in this section, but we refer to the following surveys [25, 26] for in depth details. Definition 4 formalizes a HE scheme that relies on public key cryptography.

Definition 4 (PK-based HE scheme)**.** *A PK-based HE scheme is composed of a tuple of algorithms* (GenKey, Enc, Dec, Eval)*.*

- $\mathsf{GenKey}(1^\lambda) \rightarrow (sk, pk, ek)$*. On input the security parameter* λ*, it outputs a new tuple of keys* (sk, pk, ek)*.*
- $\mathsf{Enc}(m, pk) \rightarrow c$*. On input a message* m *and the public key* pk*, it outputs a ciphertext* c*.*

UNIVERSIDAD DE MÁLAGA

- $\mathsf{Dec}(c, sk) \rightarrow m$. *On input a ciphertext c that encrypts a valid message under the public key pk, and the secret key sk associated to that pk, it outputs the plaintext m.*
- $\mathsf{Eval}(\{c_1, ..., c_n\}, f(\cdot), ek) \rightarrow c_{eval}$. *On input a set of ciphertexts $\{c_1, ..., c_n\}$ generated under the same public key, a function $f(\cdot)$, and the evaluation key ek, it outputs a new ciphertext c_{eval} such that $Dec(c_{eval}, sk) \rightarrow f(m_1, ..., m_n)$, with m_i the plaintext of ciphertext c_i.*

The main application of HE has been letting a client to outsource some computation on his private data to an untrusted server, such that the server does not learn anything about the inputs or the output. An important aspect is that there are three main categories of HE schemes, based on the amount of computation they allow to perform with the Eval algorithm.

The first group is *Partially HE* (PHE), which are schemes that support a single type of operation in Eval, either addition or multiplication. Formally speaking, an Additively HE (AHE) scheme has an operation $\oplus$ on the ciphertext space such that $\mathsf{Enc}_{pk}(m_1) \oplus \mathsf{Enc}_{pk}(m_2) = \mathsf{Enc}_{pk}(m_1 + m_2)$. On the other hand, a Multiplicatively HE (MHE) scheme has an operation $\otimes$ on the ciphertext space such that $\mathsf{Enc}_{pk}(m_1) \otimes \mathsf{Enc}_{pk}(m_2) = \mathsf{Enc}_{pk}(m_1 \cdot m_2)$. Despite there are a few, two of the most adopted in the literature are Paillier scheme [27], which is AHE, and ElGamal scheme [28], which is MHE. For the sake of completeness, Equations 2.6 and 2.7 show correctness for the homomorphism in Paillier's and ElGamal's schemes respectively.

$$(g^{m_1} {r_1}^n) \times (g^{m_2} {r_2}^n) \; (mod \; n^2) = (g^{m_1+m_2} (r_1 r_2)^n) \; (mod \; n^2) \tag{2.6}$$

$$(g^{r_1}, m_1 h^{r_1}) \times (g^{r_2}, m_2 h^{r_2}) = (g^{r_1+r_2}, m_1 m_2 h^{r_1+r_2}) \tag{2.7}$$

The second group is *Somewhat HE* (SWHE), which are schemes that support more than one operation but only a limited number of times. The major disadvantage of the main early works about SWHE is that the ciphertext size grows with the number of operations performed [29]. However, the BGN scheme [30] managed to achieve constant size ciphertexts, where and unlimited number of additions and one multiplication can be evaluated. This was a great achievement that set the seed for the next group of HE protocols.

Such a group, the most complete one, is *Fully HE* (FHE), where there is no limit in the number of operations that can be applied to ciphertexts. The first feasible FHE scheme was introduced by Gentry [31], based on lattice cryptography, together with a general framework to obtain an FHE scheme. One of the main problems when evaluating functions on encrypted data is that the "noise" becomes higher after each operation, such that the result can become unrecoverable after decryption. Although Gentry's solution is quite inefficient, he introduced a key procedure to avoid the noise threshold in FHE: *bootstrapping*. It is basically a re-encryption procedure to obtain a new ciphertext of the same plaintext with a lower degree of noise. This is achieved by homomorphically evaluating its own decryption algorithm, i.e., $\mathsf{Eval}(\{c_1 = \mathsf{Enc}_{pk_1}(m), c_2 = \mathsf{Enc}_{pk_1}(sk_1)\}, f = \mathsf{Dec}) \rightarrow c'$, and then re-encrypting

UNIVERSIDAD DE MÁLAGA

to $c'' = \mathsf{Enc}_{pk_2}(c') = \mathsf{Enc}_{pk_2}(m)$. The first evaluation (homomorphic decryption) removes the noise, and the second encryption adds new small noise.

From this work, many others FHE solutions have been proposed by following the same concept, but from different approaches. These schemes can be classified into four main groups: (1) FHE over the integers [32], (2) Leveled FHE[2] [33], (3) NTRU-based FHE [34], and (4) fast bootstrapping FHE [35].

We notice that there is a close relation between FHE and MPC. In fact, FHE has been proposed as one of the main building blocks to solve MPC problems, either as a support building block or as the main one. Note that naive FHE based on public key cryptography implies some sort of MPC: given the public key of a server, a set of users can encrypt their data to that key and let the server compute the result and decrypt it for the users. However, this implies a trusted server and also that it can see the result. Another option is to use one user's public key to encrypt data, but this will imply that data from other users can be decrypted by the owner of that key pair. In the end, naive FHE is useful for outsourced evaluation when there is a single input provider. Nevertheless, more advanced constructions have been proposed to achieve a middleground solution between FHE and MPC. These are Multikey FHE, such as the scheme of López-Alt et al. [36], and Threshold FHE, such as that of Jain et al. [37]. In Multikey FHE, there is a set of independent key pairs, each one held by a different user. The server evaluates ciphertexts under an evaluation key which is derived from the set of public keys, and the result can be jointly decrypted by the set of private keys through an interactive MPC procedure. On the other hand, in Threshold FHE, there is a single public key for all encryptions and evaluations, but the private key is split through the set of users by some sort of secret sharing. Then, after evaluating the function on encrypted inputs, a valid subset of the users must collaborate to decrypt the result.

2.1.3 Main MPC protocols

2.1.3.1 Functionality representation

When one thinks of a functionality, it is easy to visualize a black box where some inputs are provided and a result is obtained from it. When working with computers, what one imagines inside the box is a set of instructions that are executed in an ordered way, such that they transform the inputs to an output. However, this way of representing computations does not allow an easy manipulation for privacy preserving cryptographic techniques. In the area of MPC (and other approaches in cryptography), functionalities are represented by circuits, which are mathematical models largely used in the area of complexity analysis.

Definition 5 (Circuits)**.** *A circuit is a finite directed acyclic graph $C = (V, E)$, where the set of vertices V represent inputs and gates, while the set of edges E represent connection wires. Each gate $g_i \in G \subset V$ takes one or more input values through wires and produces an output to another wire, which can also be the input to a different gate g_j. The final set of gates whose outputs are not inputs to other gates is called the set of output gates.*

[2]Can evaluate circuits with a bounded depth.

UNIVERSIDAD DE MÁLAGA

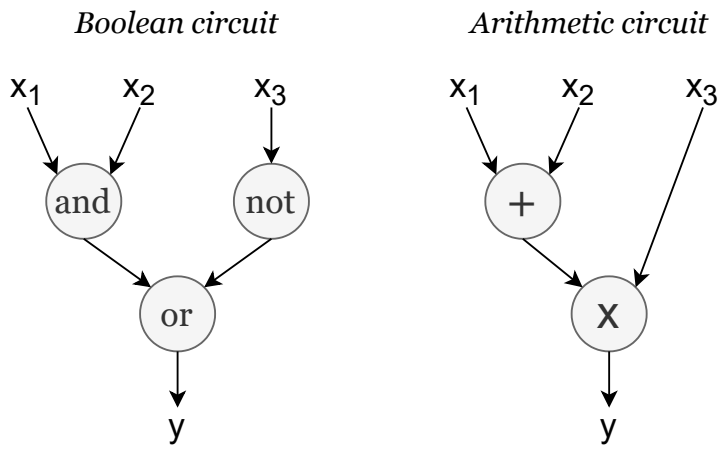


Figure 2.1: Example of a boolean and an arithmetic circuit.

Protocols in MPC rely on one of two type of circuits: boolean or arithmetic (see Figure 2.1 for a graphical representation). While a boolean circuit handle binary data and its gates compute boolean operations, such as OR, AND, or NOT, an arithmetic circuit works over a field $\mathbb{F}$ and has addition and multiplication gates. Selecting one or another to design a protocol is related with how inputs are encoded and which type of functionalities are desired to be computed. While some of them are inherently binary, others are more natural and efficient in arithmetic circuits.

There are other details about functionalities that must be considered when computing them with MPC, such as branching. Having in mind that the objective is not to expose anything about the inputs, one could think about an *if* statement that splits a circuit into two branches. If only one of the two branches is evaluated at running time, that exposes information about the input (maybe being greater than a certain value). Therefore, the straightforward but inefficient approach that has been used to evaluate circuits with branches is evaluating all branches, and then multiplexing to the active one, without exposing the result of the multiplexer. Designing efficient MPC protocols that do not need to evaluate all branches in order to avoid exposing the path selection is an interesting but complex area of research.

2.1.3.2 Foundational works

In this section we briefly introduce the main foundational works for MPC. Even when they are somehow inefficient now, they served as the base for the following contributions to MPC and have been highly optimized. First solutions are based on *Garbled Circuits* (GC), which were originally proposed by Yao [38, 39] for the 2-party case, and later generalized by Goldreich et al. for the n-party case [12]. Garbled Circuits allow to compute functionalities described on binary circuits and the base cryptographic primitive they use is *Oblivious Transfer* (OT). In fact, OT has been proven to be complete for MPC [40].

Garbled Circuits

Garbled Circuits (GC) [38, 39] are the first construction that enabled general purpose MPC. They enable 2-PC and can be somehow interpreted as a SWHE scheme for non-constant size ciphertexts, where one party encodes the inputs and the other

UNIVERSIDAD DE MÁLAGA

evaluates the function. GC are not efficient in communication, but they achieve a constant number of rounds, which reduces latency. With respect to function evaluation, they allow to evaluate boolean circuits.

In GC, there are two roles: the *garbler*, with input x, and the *evaluator*, with input y. The general idea is to represent the function as a look-up table that takes all possible inputs and its corresponding results, one per row. However, instead of doing this for the entire function, the garbling is normally done gate-by-gate. We now depict the process for a single AND gate with a 2-bit input and a 1-bit output, but this can be extended to any gate:

$$\begin{bmatrix} \mathbf{w_a} & \mathbf{w_b} & \mathbf{w_c} \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \quad \begin{bmatrix} \mathbf{w_a} & \mathbf{w_b} & \mathbf{w_c} \\ k_a^0 & k_b^0 & k_c^0 \\ k_a^0 & k_b^1 & k_c^0 \\ k_a^1 & k_b^0 & k_c^0 \\ k_a^1 & k_b^1 & k_c^1 \end{bmatrix} \quad \begin{bmatrix} \textbf{garbled rows} \\ Enc_{k_a^0}(Enc_{k_b^0}(k_c^0)) \\ Enc_{k_a^0}(Enc_{k_b^1}(k_c^0)) \\ Enc_{k_a^1}(Enc_{k_b^0}(k_c^0)) \\ Enc_{k_a^1}(Enc_{k_b^1}(k_c^1)) \end{bmatrix}$$

From the tables represented above, the first one is the truth table of the gate, where $\mathbf{w_a}, \mathbf{w_b}$ are the input wires, and $\mathbf{w_c}$ is the output wire. We assume here that the input for $\mathbf{w_a}$ is provided by the garbler, while the input for $\mathbf{w_b}$ is provided by the evaluator. The second table represents the truth table but with each value substituted by a random key of length κ, which is the security parameter. More specifically, k_i^j is the key assigned for wire $\mathbf{w_i}$ when its value is j. Finally, the last table represents the garbling, which is the double encryption of each output's key with both inputs' keys. This last table is generated by the garbled, who then sends the encryptions to the evaluator, in permuted order, together with the input key k_a^x of the garbler, i.e., the key assigned to its input bit x. The next phase requires the evaluator to learn his input key k_b^y, given his input bit y, but without learning the key k_b^{1-y}. This can be achieved by a $\binom{2}{1}$-OT, where the inputs k_b^0 and k_b^1 are provided by the garbler and the choice bit, provided by the evaluator, is y. Finally, once the evaluator has both k_a^x and k_b^y, he can decrypt a single row of the garbled table, obtaining k_c^z. This value is then notified to the garbler, who knows if it corresponds to 0 or to 1, therefore obtaining the result of the gate.

Notice that this process can easily be repeated for the rest of the gates until the circuit is fully evaluated, no matter how many inputs and outputs it has. Also, despite this is the most basic and simplest form of GC, there are several improvements for efficiency. Some examples are the Point-and-Permute technique, which allows the evaluator to know which exact row has to decrypt, or the free XOR technique, which allows to evaluate XOR gates without requiring OT, which incurs in communication costs.

GMW

After GC, the work proposed by Goldreich et al. [12] (GMW) presented a way to extend 2-PC to n-PC. However, its number of rounds depends on the depth of the circuit to be evaluated, although it supports both boolean and arithmetic circuits. GMW is based on secret sharing and OT, and proposes a protocol for 2-PC evaluation of gates that is used within the n-PC case.

UNIVERSIDAD DE MÁLAGA

First, given w_x, w_y, and w_k the two inputs and output wires of a gate, respectively, secret sharing allows each party to hold a share of each input, i.e., $w_x = s_x^1 \oplus s_x^2$ and $w_y = s_y^1 \oplus s_y^2$, such that s^i is a share held by party P_i. First, both NOT and XOR gates have non-interactive evaluation. To evaluate NOT gates, it just suffices with P_1 flipping its share of the wire value. To evaluate XOR gates, each P_i computes $s_k^i = s_x^i \oplus s_y^i$. Equation 2.8 shows correctness for the XOR case.

$$s_k^1 \oplus s_k^2 = (s_x^1 \oplus s_y^1) \oplus (s_x^2 \oplus s_y^2) = (s_x^1 \oplus s_x^2) \oplus (s_y^1 \oplus s_y^2) = w_x \oplus w_y \tag{2.8}$$

With respect to AND gates evaluation, the process must be interactive by using $\binom{4}{1}$-OT. Given $S(s_i^2, s_j^2) = (s_i^1 \oplus s_i^2) \wedge (s_j^1 \oplus s_j^2)$ the evaluation of the AND gate under the fixed input of P_1, a secret share must be prepared for each possible input of P_2. For that, P_1 chooses a random r and computes the following vector,

$$\begin{pmatrix} r \oplus S(0,0) \\ r \oplus S(0,1) \\ r \oplus S(1,0) \\ r \oplus S(1,1) \end{pmatrix}$$

from where P_2 can retrieve the corresponding share (one row) by using his input bits as selector for the OT. Finally, revealing the output just implies both parties revealing their shares.

We now explain the extension for the n-PC case. Given a set $\{P_i\}_{i=1}^n$, each P_j secret shares his input by sending a share r_i to each P_i. As shown in the 2-PC case, XOR can be locally evaluated by XORing shares. On the other hand, the AND evaluation takes two steps here: a local phase and an interactive phase. Equation 2.9 shows the description of the AND evaluation of two inputs x, y. For the local phase, P_j computes $x_j \wedge y_j$. Then, for the interactive phase, each pair (P_i, P_j) use the 2-PC GMW procedure to evaluate $x_i \wedge y_j$.

$$z = x \wedge y = \left(\bigoplus_{i=1}^{n} x_i \right) \wedge \left(\bigoplus_{i=1}^{n} y_i \right) = \left(\bigoplus_{i=1}^{n} x_i \wedge y_i \right) \oplus \left(\bigoplus_{i \neq j} x_i \wedge y_j \right) \tag{2.9}$$

BGW

BGW [41] is a protocol for n-PC that forms the basis of today's most widely adopted frameworks for MPC. This protocol has a number of rounds linear in the depth of the circuit, and can be applied on boolean or arithmetic circuits.

The intuition behind BGW is very straightforward, because it just takes advantage of linear SS schemes and their homomorphic properties. Section 2.1.2.2 introduced Shamir SS, which is based on polynomials, and where the evaluation $P(0)$ encodes the secret s. We recall that addition is easy to compute and only needs local computation. Given $[\![x]\!], [\![y]\!]$ two secretly shared values, and $P_x(i), P_y(i)$ the shares held by P_i, it just suffices to compute $P_x(i) + P_y(i)$ to obtain a share of $[\![x + y]\!]$. The same happens with scalar multiplication, where locally computing $\lambda P_x(i)$ produces a share for P_i of $[\![\lambda x]\!]$.

UNIVERSIDAD DE MÁLAGA

As it was also introduced in Section 2.1.2.2, Shamir SS allows multiplication, but the product of two polynomials of degree t results in a polynomial of degree $2t$, i.e., $[\![x]\!]_t \cdot [\![y]\!]_t = [\![z]\!]_{2t}$. We now explain how parties can apply a degree reduction step to the product polynomial with an extra round of sharings. First, $P_z(X) = P_x(X) \cdot P_y(X)$ represents the product polynomial of degree $2t$. By Lagrange interpolation method, the secret $z = xy$ can be represented by $P_z(0) = \sum_{i=1}^{2t+1} \lambda_i(0) P_z(i)$, where $\lambda_i(0)$ are the corresponding Lagrange coefficients evaluated in 0 and $P_z(i)$ is P_i's share. Then, for degree reduction, the first step is for P_i generating and distributing a t-share $[\![P_z(i)]\!]_t$. The second step leads every P_j to locally compute $[\![P_z(0)]\!] = \sum_{i=1}^{2t+1} \lambda_i(0)[\![P_z(i)]\!]$. Since $[\![P_z(i)]\!]$ has threshold t, $[\![P_z(0)]\!]$ has it too.

Pre-processed Multiplication Triples

Pre-processed triples is a method introduced by Beaver [42] that allows to generate some correlated randomness first, that can be used later to compute multiplications on secret shared data. The benefits of Beaver triples are twofold: first, they can be generated in a pre-processing phase, which is independent of the inputs provided by the parties, and second, it is a method which is agnostic regarding the secret sharing method used, as long as it is linear and allow local evaluations of addition and scalar multiplication. Therefore, this method works with both Shamir and Additive SS. The pre-processing notion has lead to the well-known paradigm of off-line/on-line MPC, where an off-line phase is entirely dedicated to pre-compute triples, and the on-line phase means consuming those triples to evaluate multiplication gates on inputs, besides addition and scalar multiplication gates.

A Beaver triple has the form $([\![a]\!], [\![b]\!], [\![c]\!])$ such that $c = a \cdot b$. Note that they can be generated by a trusted dealer, by the set of parties using BGW multiplication, or by more advanced protocols that allow efficient generation of batches of triples [43, 44].

Given that all parties hold shares of the triple $([\![a]\!], [\![b]\!], [\![c]\!])$ and the inputs $[\![x]\!], [\![y]\!]$, the protocol to perform a multiplication works as follows:

1. Each party locally compute $[\![x - a]\!]$, and everyone open $d = x - a$.
2. Each party locally compute $[\![y - b]\!]$, and everyone open $e = y - b$.
3. Each party locally compute $[\![xy]\!] = de + d[\![b]\!] + e[\![a]\!] + [\![c]\!]$.

$$xy = (x - a + a)(y - b + b) = (d + a)(e + b) = \\ = de + db + ae + ab = de + db + ae + c \quad (2.10)$$

Equation 2.10 shows correctness for the last step of the protocol, i.e., it generates a correct share of the product. Notice that it can be computed because d, e are publicly known values. This method allows to perform multiplications in the on-line phase with just two openings and local computation, instead of using sharings of n elements by private channels, as it happens in standard BGW multiplication.

BMR

UNIVERSIDAD DE MÁLAGA

The protocols introduced before have served as the basis for most recent MPC frameworks. However, despite not researched that much, BMR [45] started a different line of work for n-PC protocols with a constant number of rounds and boolean circuit evaluation.

BMR is a bit more complex than the other protocols. Its basic idea is to perform a distributed GC generation that can be parallelized, i.e., generating all wire labels and then all garbled gate tables. After the joint effort to generate the GC, it is sent to one evaluator that will output the result. However, the main difficulty is how to sent the active input labels to the evaluator, i.e., those associated with the actual inputs of the parties.

The idea is that for wire w_a, its label w_a^v is computed as the concatenation of sublabels $w_{a,1}^v||...||w_{a,n}^v$, where $w_{a,j}^v$ is generated by P_j. Then, for a gate G_i with input and output labels $w_a^{v_a}, w_b^{v_b}, w_c^{v_c}$ respectively, the garbled row is computed as described in Equation 2.11, where F is a PRF function that extends $\kappa + 1$ bits to $n(\kappa + 1)$.

$$e_{v_a,v_b} = w_c^{v_c} \bigoplus_{j=1}^{n} \left(F(i, w_{a,j}^{v_a}) \oplus F(i, w_{b,j}^{v_b}) \right) \tag{2.11}$$

In summary, each P_j locally computes $F(i, w_{a,j}^{v_a}) \oplus F(i, w_{b,j}^{v_b})$ and submits it to the MPC. Finally, to avoid the evaluator learning the correspondence between its active label and the plaintext input value, "flip" values are used. For that, P_j adds $f_{a,j}$ to each wire, such that $f_a = \bigoplus_{j=1}^{n} f_{a,j}$ determines which plaintext corresponds to w_a^v.

2.1.3.3 Modern paradigms in MPC

Since the base protocols introduced before, many new protocols have been proposed. Although the essence has not changed a lot, the focus has been both achieving better performance and providing better security guarantees. In this section, we present two widely known approaches which are inspired by the BGW approach: GSZ [46] for semi-honest security, and SPDZ [47, 48, 49] for malicious security.

GSZ

GSZ is based on Shamir SS and focuses on the problem of achieving unconditional MPC with guaranteed output delivery for $t < n/2$ such that the communication complexity per multiplication gate is linear in n. Note that this guaranteed output delivery notion is stricter than what the works in Section 2.1.3.2 offer.

The starting strategy for GSZ is the following: (1) run a semi-honest protocol until the output phase, (2) check the correctness of the computation, and (3) reconstruct the output only if the check passes. First, multiplications are based on double sharing consumption, which are pairs $([\![r]\!]_t, [\![r]\!]_{2t})$. To perform a multiplication on $[\![x]\!]_t, [\![y]\!]_t$, the following phases are followed:

1. All parties locally compute $[\![xy + r]\!]_{2t} = [\![x]\!]_t \cdot [\![y]\!]_t + [\![r]\!]_{2t}$.
2. The value $xy + r$ is opened.
3. All parties locally compute $[\![xy]\!]_t = xy + r - [\![r]\!]_t$.

UNIVERSIDAD DE MÁLAGA

Another fundamental step for this line of protocols is Batch-Wise Multiplication Verification, where given a list of m triples $(\llbracket x^{(i)} \rrbracket_t, \llbracket y^{(i)} \rrbracket_t, \llbracket z^{(i)} \rrbracket_t)_{i=\{1,...,m\}}$, the purpose is to check whether $x^{(i)} \cdot y^{(i)} = z^{(i)}$ for all i. The idea from where GSZ starts is to construct three polynomials $F(X), G(X), H(X)$, such that $F(i) = x^{(i)}$, $G(i) = y^{(i)}$, and $H(i) = z^{(i)}$ for all i, and then check if $F(X) \cdot G(X) = H(X)$. This allows to compress from m checks of multiplication tuples to a single check of $(\llbracket F(X) \rrbracket_t, \llbracket G(X) \rrbracket_t, \llbracket H(X) \rrbracket_t)$.

Upon these ideas, GSZ introduces a set of new techniques that enhances performance. In addition, they propose two new protocols, one with security-with-abort, and other with guaranteed output delivery. However, delving into details is out of the scope of this work, and we refer to the original paper [46] for that.

SPDZ

SPDZ is a general purpose MPC protocol that achieves security against active adversaries for a dishonest majority $t < n$, works on arithmetic circuits over $\mathbb{F}_{p^k}$, and employs additive secret shares. It works in the offline/online model, where the offline phase produces pre-processed independent data using SWHE and with quadratic complexity in n. For the online phase, it achieves unconditional security and linear complexity in n.

We will briefly sketch some notions of the online phase from the first SPDZ paper [47]. Subsequent improvements were proposed [48, 44, 49], but we will not cover them for being out of the scope of this work. While prior research for maliciously secure MPC focused on authenticating secret shares independently, SPDZ put the focus on authenticating the secret with a global secret shared key. It works somehow similar to GSZ for the verification strategy: it first executes the computation, and performs a final verification step in the end after revealing the global secret shared key.

First, from the pre-processing, and relying in SWHE and ZKP, each party holds a share of the global authentication key α (in addition to pre-processed pairs and triples that we omit for simplicity). The secret shared form of α is shown in Equation 2.12, from where P_i holds α_i, β_i and $\{\gamma(\alpha)^i_j\}_{j\in\{1,...,n\}}$.

$$\llbracket \alpha \rrbracket = ((\alpha_1, ..., \alpha_n), (\beta_i, \gamma(\alpha)^i_1, ..., \gamma(\alpha)^i_n)_{i\in\{1,...,n\}}) \tag{2.12}$$

The idea is that $\alpha = \sum_{i=1}^{n} \alpha_i$ is the share, $\gamma(\alpha)_i = \sum_{j=1}^{n} \gamma(\alpha)^j_i = \alpha\beta_i$ is the MAC authenticating α under P_i's private key β_i.

On the other hand, secret shared inputs use a different notation, as shown in Equation 2.13 for input value x, from where P_i holds $x_i, \gamma(x)_i$, and δ, which is public.

$$\langle x \rangle = (\delta, (x_1, ..., x_n), (\gamma(x)_1, ..., \gamma(x)_n)) \tag{2.13}$$

It must hold that $x = \sum_{i=1}^{n} x_i$, and $\alpha(x + \delta) = \sum_{i=1}^{n} \gamma(x)_i = \gamma(x)$, which is the MAC that authenticates x under the global key α.

UNIVERSIDAD DE MÁLAGA

2.1.3.4 New dynamic approaches for MPC

There is a recent line of research within the MPC field that has gained a huge attention in just a few years. This line of research could be tagged with the name of *dynamic MPC*, regarding the computing parties involved in the protocol. In traditional approaches, a functionality F is designed to be executed by a fixed set of computing parties from the beginning to the end of F. However, in dynamic MPC, not all the steps in F are executed by the same set of parties.

The ideas for these works can be somehow originated to *Dynamic Proactive Secret Sharing* (DPSS) schemes, where a secret shared secret does not live forever in the same committee, but it is moved from C_i to C_{i+1} from time to time using a secure handover mechanism. This is interesting because it constrains the amount of time that an adversary has to corrupt a sufficiently large subset of holders in C_i, otherwise the secret is sent to C_{i+1} and deleted from C_i.

Following this idea, Benhamouda et al. [50] proposes a protocol called *Evolving Committee PSS* (ECPSS), which is a special type of DPSS that does not only consider the way that the committees hold share and perform the handover, but also include the functionality that performs the committee selection for the next epoch. This is important because a corrupted functionality for committee selection can select a proper subset for the next epoch such that the adversary controls a sufficiently large subset of parties, therefore exposing the secret. Benhamouda et al. proposes a nominating committee for this, and relies on anonymous PKE and selective opening, where a blockchain provides authenticated broadcast and PKI. A similar approach is followed by Goyal et al. [51], that proposes a Fast Batched DPSS (FaB-DPSS) protocol that is more efficient than prior works and supports multiple clients to store and release secrets. This work also relies on blockchain that, combined with DPSS, provides a conditional secret storage application, where secrets are stored along with a release condition.

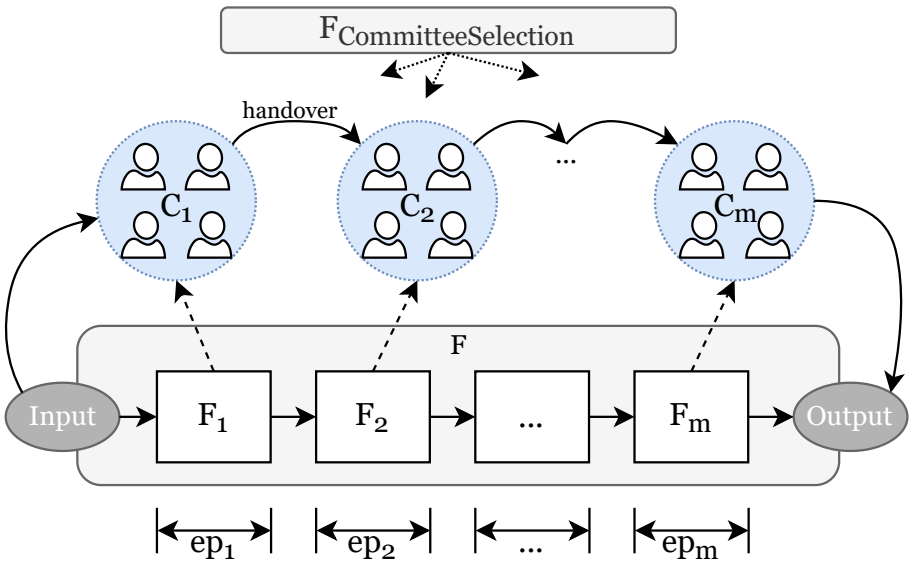


Figure 2.2: General model for dynamic MPC with ephemeral committees.

While the previous works set the focus on storing secrets, subsequent works rapidly noticed that this notion of dynamic committees could be used for general purpose

UNIVERSIDAD DE MÁLAGA

MPC applications. The general model is presented in Figure 2.2, where if we assume that a functionality F is split in a set of substeps $\{F_1, ..., F_m\}$, where step F_{i+1} takes as input intermediate data produced by step F_i, then each committee C_i is assigned with the substep F_i. Per each epoch step, instead of just passing a static secret in the handover phase, C_i secretly sends to C_{i+1} the intermediate results from F_{i+1}.

Fluid MPC [52] formalized this notion of general purpose MPC with dynamic participants, i.e., that can join and leave the computation. It introduces the notion of *fluidity*, which is the minimum number of communication rounds within an epoch, and targets protocols with maximal fluidity, i.e., with a single communication round per epoch. It also provides information-theoretic fluid MPC protocols based on an advanced version of BGW. Another work, called YOSO [53] generalizes this notion of dynamic MPC where there is a single message sent per each party. This allows an additional layer of protection against adaptive adversaries because it is not known in advance which party will be the next sending a message, and once done (and identified by the adversary), that party will never speak again. YOSO defines two layers of design: (1) the protocol design with these abstract roles, which is where this work is focused, and (2) the way that roles are assigned to physical machines. Finally, SCALES [54] modifies YOSO in order to preserve the resilience against adaptive adversaries, but with additional efficiency. It is focused on the outsourced MPC model, where input providers do not participate in the computation. SCALES also provides a new construction which serves as its base for MPC, named *Rerandomizable Garbling Schemes* (RGS), which is a generalization of GC to multiple garblers.

A final remark is that all these works introduced in this section rely somehow in blockchain for some of their steps, and mainly for role assignment/committee selection features.

2.2 Blockchain

Blockchain technologies are a specific type of *Distributed Ledger Technology* (DLT). They essentially store information in a sequentially ordered way, using a *Directed Acyclic Graph* (DAG) with no branches[3]. Every data update, typically initiated by a *transaction*, forms a new batch of data called *block*, where each block can store one or more transactions, and all blocks are cryptographically linked sequentially, hence the name of blockchain. According to Garay et al. [55], a blockchain can be roughly formalized as a *State Transition System* (STS), as shown in Definition 6.

Definition 6 (Blockchain as STS)**.** *A blockchain, modeled as a State Transition System, is a read/write system that maintains a unified view of its state, i.e., the data it has stored in a specific moment in time. It parts from a initial state* st_0 *that may contain specific data associated and is composed of a tuple of algorithms* $(\mathsf{Input}, \mathsf{Validate}, \mathsf{Read})$*:*

- $\mathsf{Input}(st_{i-1}, \mathbf{C}, i, x) \rightarrow st_i$*. On input new data* x *to be added in round* i*, with* $i \in \mathbb{N}$*, the state is updated from* st_{i-1} *to* st_i*, and where* $\mathbf{C}$ *represents the chain.*

[3]This is the main difference between blockchain and other DLTs: blockchain does not allow state branching, while other DLTs do, even if they include mechanisms for branching reunification.

UNIVERSIDAD DE MÁLAGA

- $\mathsf{Validate}(\mathbf{x_C}) \rightarrow \{0,1\}$. *On input the contents of the chain* $\mathbf{x_C}$*, it outputs a boolean result depending on whether if they are consistent with the way that the application works or not.*
- $\mathsf{Read}(p) \rightarrow \mathbf{d}$. *On input a pointer* p *to some data in* $\mathbf{x_C}$*, it outputs the queried data* $\mathbf{d}$ *(or some representation of* $\mathbf{d}$*), where* $\mathbf{d} \subseteq \mathbf{x_C}$.

Far from being just tools for storing data, blockchains have brought a new way to share data with some security principles in mind. Therefore, the interesting thing is how they implement the functions previously described and, most of all, Validate and Input. Their particularities have enabled the growth of many different use cases, from where decentralized digital currencies, or *cryptocurrencies* have been the most extended ones. Blockchains are based on a few of core design principles, which are briefly introduced below.

Decentralization. There is no single centralized entity in a blockchain that controls interactions or data. Participants (sometimes called *blockchain nodes*) are equally from a hierarchical point of view, and it is typical to have blockchains form a partial meshed topology, i.e., a *Peer-to-Peer* (P2P) network. This introduces some difficulties to regulate data and achieve a common view of the chain state, which are solved by consensus protocols.

Replicated storage. Blockchains are not technologies to split data in different stores, but they require all data to be replicated at each node (in practice, there can be nodes storing just parts of the whole data). This is a requirement for decentralization, data immutability and public verifiability, since every node must be able to verify that a transaction, i.e., a modification in the blockchain state, is correct regarding the previous state of the blockchain.

Immutability. Immutability means that blockchain data cannot be modified (nor erased) once appended to the state. This is achieved thanks to replicated storage and public data verification, which allow any node in the blockchain to be aware of the different states of the blockchain history and verify if a transaction is correct with respect to a specific state. Modifying a previous state would require a huge amount of effort, as long as 2/3 of the total amount of blockchain nodes remain honest. In addition, since states are cryptographically linked, modifying a previous state gets more difficult each time a new state update is appended.

Pseudonimity. Blockchains working are based on public key pairs. Each participant generate a new key pair, where the private key is used to sign transactions. On the other hand, the public key allows to verify correctness of the signature. In addition, blockchain identities are called *wallet addresses*, which are pseudo-random values generated from the public keys. This allows to hide the users' real identities from the blockchain. However, relating wallets to real identities is not impossible, and is the reason that blockchains are not considered anonymous.

The main problem that arises from having no centralized entity to coordinate the different nodes is how to achieve *Byzantine Fault Tolerance* (BFT). This notion comes from the original concept of *Byzantine Generals* problem, introduced by Lamport et al. [56]. A system with BFT is said to work correctly as long as 2/3 of the nodes

UNIVERSIDAD DE MÁLAGA

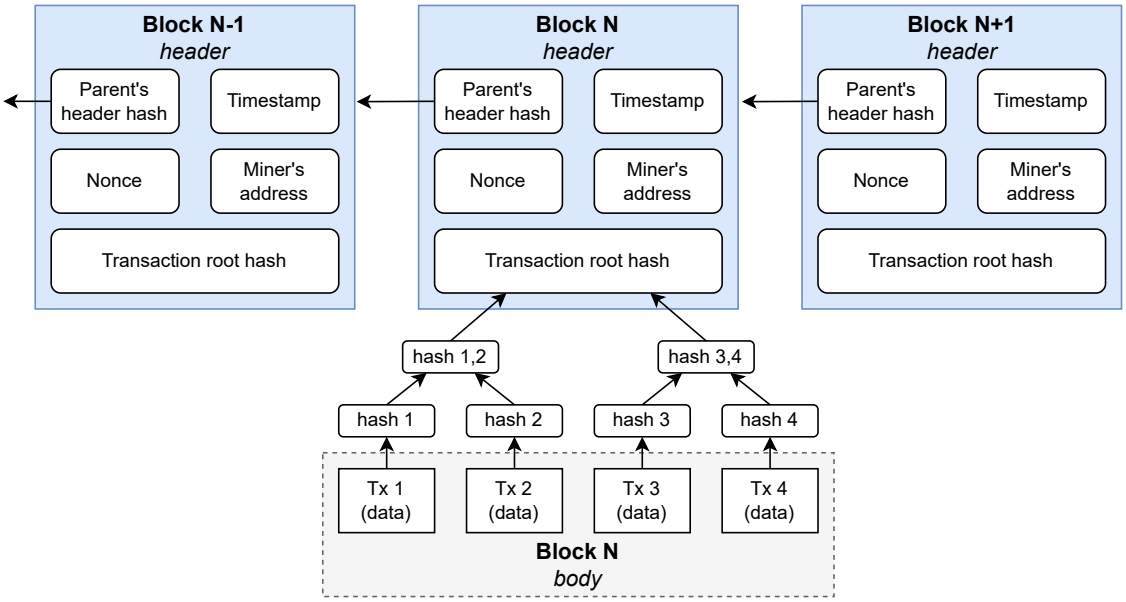


Figure 2.3: High-level representation of blockchain blocks.

involved are honest and do not fail.

Figure 2.3 shows a high-level representation of how blockchain data is structured in blocks. Each block is linked to the previous one (the parent) by incorporating its hash. The transaction root hash is a common mechanism used to decouple the header of a block from the content of the transactions it contains. Therefore, a single hash is included in the header such that it works as a commitment of the body of that block, i.e., the set of all transactions included in it. This is typically done by Merkle Trees, as shown in the figure, where individual hashes are hashed together by pairs until a single hash for all the set of transactions is obtained.

2.2.1 Blockchain classifications

2.2.1.1 Access to the blockchain

A first classification is made based on who can access the blockchain, i.e., the data it stores and the capabilities to send requests to the blockchain. The first type is *permissionless* blockchain, where anyone can become a participant. A participant can be understood as a user that consumes or modifies data which is stored in the blockchain, but also those nodes that maintain the state. This means that anyone that is suitable to participate in the consensus can become a member and propose new blocks. In addition, permissionless blockchains have all data publicly available from the nodes that participate in the replicated storage, and any client with access to a blockchain node can retrieve data from the blockchain. Permissionless blockchains, such as Ethereum [57] or Bitcoin [58], are those that follows the decentralization philosophy in its maximal exponent, since there is no single manager that makes decisions or orchestrates the way that the blockchain works.

The opposite concept is *permissioned* blockchain, where there exist a control access layer. This means that, in order to become a blockchain node or to query the blockchain contents, access must be provided by a manager. Permissioned

UNIVERSIDAD DE MÁLAGA

blockchains are also called private, and are normally used in enterprise contexts where it is desired to deploy a reliable core for verifiability and resilience based on the deployment of a set of blockchain nodes. However, permissioned blockchains are typically managed by a central operator, which moves away from the original philosophy of decentralization and may introduce additional security issues, mainly if the target for an attack is the operator itself.

There is another type which is *hybrid* or *consortium* blockchain, that also targets a more or less isolated environment where access to data is not public, but where there is no single point of control of the blockchain. In consortium blockchains, a set of entities (enterprises, organizations, managers, etc.) are assumed to make decisions based on decentralization, where all of them take part in the process. The target scenarios of consortium blockchains are normally collaborative environments where different organizations want to share some data and procedures, but do not fully trust each other, for example if they are competitors.

2.2.1.2 Ledger structure and interaction

Another classification can be made based on how the ledger stores data and transactions are generated. The two main approaches are *UTXO*-based blockchains (introduced by Bitcoin) and *account*-based blockchains (introduced by Ethereum).

An *Unspent Transaction Output* (UTXO) is a set of unspent notes where each one can only be consumed by a specific user. In these blockchains, a transaction has a set of inputs, which are unspent notes in which the transaction sender is the one with ability to spend them. In addition, the transaction includes a set of outputs, which are new unspent notes with new receivers. A key aspect is that, given $\{I_i\}_{i\in[n]}$ the set of amounts spent in the inputs of a transaction and $\{O_j\}_{j\in[m]}$ the set of amounts being sent in the outputs of that transaction, the total balance must be maintained, as shown in Equation 2.14, where f is the fee spent to have the transaction accepted.

$$\left(\sum_{i=1}^{n} I_i\right) + f = \sum_{j=1}^{m} O_j \tag{2.14}$$

On the other hand, in account-based blockchains, the ledger maintains a list of all accounts that exist and their balances. Each transaction that sends cryptocurrency must specify the origin account from which the funds are taken and the list of accounts that will receive the funds. As it happened in the UTXO case, the total amount of funds sent must be equal to the amount subtracted from the sender's balance plus the fee. In this case, the ledger does not maintain input and output notes, but updates the balances of the accounts. Another feature that Ethereum introduced is the concept of storage. There are two types of accounts: Externally Owned Accounts (EOA), which are controlled by users through their signing keys and can deploy smart contracts, and Contract Accounts (CA), which can interact with the on-chain storage and process data.

UNIVERSIDAD DE MÁLAGA

2.2.1.3 Consensus mechanisms

Since there is no centralized entity, an additional mechanism is needed to allow every decentralized peer agreeing in the order of the transactions. This is needed to ensure a single chain of blocks, because blockchain does not allows having multiple active branches.

The first blockchain proposals were based on Proof of Work (PoW). A set of peers named miners work in forming blocks, which are packages containing a set of new transactions. For a block to be accepted in state st_i, it must contain only valid transactions, must be cryptographically linked to the previous block accepted in state st_{i-1}, and must contain a valid PoW. Having only valid transactions means that all transactions included in the proposed block allow the correct transit from st_{i-1} to st_i when applying the input contribution algorithm Input. With respect to the PoW value, its main purpose is to make the creation of a new valid block computationally hard to prevent sybil attacks that remake the entire blockchain. Essentially, in each round, the double-SHA256 hash of the block proposed must be less than a dynamically adjusted target 2^n. To find a suitable value, the block proposer must add a specific number, called nonce, to a field in the block such that the resulting hash value satisfies the requirement. Since SHA256 acts as a one-way PRF, the only way to obtain a valid PoW is by trial and error, modifying the nonce value each time. The first valid block proposed will in most cases be the one accepted by the majority of the blockchain peers as the new block added for the st_i state. One of the problems of PoW is that it is computationally heavy. It requires a huge amount of work that not only impacts in the whole system performance (by large delays in block production and therefore lowering user experience) but also in the environment due to high electricity consumption (miners deploy large amount of computationally-capable devices to maximize the probability of computing a valid hash). This has lead to the development of other consensus mechanisms, such as Proof of State (PoS), which was adopted by the Ethereum blockchain in 2022.

In PoS, the existing miners (called validators in Ethereum) stake a specific amount of cryptocurrency into a smart contract. Validators are then in charge of proposing new blocks and checking that the new blocks proposed by other peers are valid. If validators are detected misbehaving by proposing invalid blocks or by sending incorrect attestations, they can loose some or all of their staked cryptocurrency. There are different mechanisms to select block proposers for each round, which can be more or less centralized, for example, by a random selection based on the amount staked. This is needed because PoS does not present the competitive procedure to solve a problem as it happens in PoW.

Another group of consensus mechanism is that based on Proof of Authority (PoA), which is mainly intended for permissioned blockchains. In PoA, there is a fixed list of *authorities* that are allowed to propose blocks, and a strategy to select proposers in each round, which is typically based on some form of round robin selection. Authorities participate in supporting new blocks, and authorities detected proposing invalid blocks can be penalized and excluded from the system.

UNIVERSIDAD DE MÁLAGA

2.2.2 The privacy problem in blockchain

One of the main benefits of blockchain technologies is that they offer public verifiability of all data and computations that are processed on-chain. However, in order to achieve that, all data must be public. This implies that one of the main benefits of blockchains is also one of their main caveats that hinders a widespread adoption. There exist many use cases and applications that work with sensitive data and where users do not want that data to be exposed, or even that regulations do not allow it. For these reasons, providing mechanisms that enable some sort of privacy or confidentiality in blockchain, while maintaining most of the properties that make them to be adopted as an interesting technology, is a trend in both research and industry.

Since blockchain was first adopted for cryptocurrency and this is one of its most extended use cases, most of the first works in blockchain privacy targets the context of private payments [59, 60]. There exist several solutions for different blockchains, some of them that make use of mixing providers and others by private coins. These solutions may lie in both public blockchains already deployed, such as Ethereum, or in new blockchains that consider a privacy-by-design approach. We will elaborate more on privacy in the cryptocurrency context in Section 6.3. However, privacy in cryptocurrency has the problem of lack of programmability, i.e., these solutions do not scale well for more complex dynamics and interactions inside the blockchain, apart from private payments. This has led to the emerge of another line of research that targets confidentiality in smart contracts from a wider point of view, where part or the whole computation done by smart contracts is desired to be protected from the public view.

One of the main approaches that have been proposed to achieve flexible confidentiality in blockchains is the use of Trusted Execution Environments (TEEs). TEEs are a hardware-based solution that isolates code execution from the host that performs the computation and its environment. The typical approach is by letting data to travel encrypted the whole time, and is only decrypted inside the TEE, where the computation can be performed in a secure way. It can be somehow considered as a solution for outsourced MPC, where multiple clients can submit their encrypted inputs to a host that holds a TEE. Security of TEEs is normally ensured by an attestation service, that guarantees that encrypted outputs come from a specific TEE. In blockchain, TEEs have been adopted to add a confidential layer for smart contracts, as analyzed in the SoK of Li et al. [61]. Many works have been proposed following this approach since 2016, and we refer to the SoK for insight details. As a general classification, Li et al. proposes two types of TEE-based blockchains: (1) *layer-one*, where all the nodes that participate in consensus must also include a TEE to perform confidential computations on smart contracts, and (2) *layer-two*, where the set of consensus nodes and those that have TEE for computations are different. Despite TEEs are interesting to achieve confidential computing with more efficient running times compared to fully-cryptography-based solutions, they also present some drawbacks. First, the need of specialized hardware, which always makes it more difficult to adopt the technology and incorporate updates. Second, TEEs may suffer from side-channels attacks. Since every computational system may somehow be exposed to this, a side-channel in TEE can directly expose private data, while in MPC a single host only reveals its view of the computation, which is not enough

UNIVERSIDAD DE MÁLAGA

for the adversary to learn the secret. Third, key management is a difficult process when working with TEEs, because it is not clear which is the best strategy to store the keys and how to use them. In the end, all data travel encrypted and different objects may be encrypted with different keys, which must reach the TEEs in order to allow decryption. In fact, some works rely on the existence of an MPC committee that securely stores the keys in a distributed manner, e.g., by using threshold cryptography. Finally, TEEs also present transparency issues compared to solutions fully-based on cryptography, where the latter enable different ranges of verifications on public computations more easily. TEEs, on their part, offer attestation services which are centralized and do not enable public verifiability of the smart contracts functionalities; only the confirmation that the encrypted results come from a TEE.

Because of the caveats that arise with TEEs, other works have proposed solutions fully-based on cryptography, mainly techniques related with MPC because it is the technology that offers the highest flexibility regarding the amount of computation to be done on confidential data. An example is the solution offered by Zama [62], which provides mechanisms to deploy a blockchain where confidential computations are entirely done inside the encrypted space by using FHE. Zama's solution is very interesting in the sense that it is EVM-compatible and that specific instructions inside a smart contract can be determined to be computed confidentially, instead of always protecting the whole smart contract from the public view. This solution is entirely on-chain, i.e., every node can compute on encrypted data by using FHE and obtain the same result. While this is an advantage for public verifiability and follows the blockchain philosophy, it also presents a major problem: FHE is very expensive in terms of computation, which implies large monetary costs when computing data on-chain. In addition, Zama's work presents another caveat that is inherent to FHE when working with multiple input providers: it must rely on other MPC techniques to handle decryption. The authors rely on threshold-FHE and they assume a large blockchain committee that holds a global decryption key, i.e., a sufficiently large threshold of nodes must contribute to decrypt or re-encrypt data that is encrypted on-chain.

Finally, there are industry works that propose solutions which are entirely based on MPC, such as Partisia[4] or Nillion[5]. However, they imply completely new blockchains, designed specifically to handle MPC. While this may not be a problem, these solutions lack the already formed user base that exist, e.g., in Ethereum, as it is targeted by Zama's work with an EVM-compatible solution. In addition, having new solutions hinders public verifiability capabilities, overall if blockchains are not permissionless.

UNIVERSIDAD DE MÁLAGA

[4] https://partisiablockchain.com/
[5] https://nillion.com/

Chapter 3

Analysis of MPC instances and protocols

In this chapter, we provide a high level approach to MPC from usability and implementation points of view, which are key and not trivial enough when theoretical designs are to be implemented in real settings. We first cover details about the trust relations involved in the process of deploying and using MPC instances. Then, we introduce some of the main specific purpose MPC protocols and discuss about their implications in trust. Finally, we provide a Systematic Literature Review about Private Set Intersection, which is one of the most widely researched protocols in the MPC ecosystem. We provide a general taxonomy, specific settings, use cases, and an analysis about the usability of the main PSI works.

3.1 Abstract models and trust relations

The paradigm of MPC is very wide and presents different flavors regarding topologies for interaction between the involved parties, the building blocks used, and the security assumptions. Theoretical works normally state a set of assumptions under which a protocol is secure, but relying on those assumptions in a real implementation may not always be as straightforward as it seems. While these assumptions are a mathematical measure, in real-life deployments there are other aspects that interfere, such as personal or political relations, economic interests, or even law enforcement. For that reason, additional work must be done when a theoretical MPC protocol is to be implemented in a real system, regarding how its building blocks, the roles of the involved participants and the trust relations between them relate to the original theoretical design. This analysis to be done prior to implementation is a key factor to minimize malfunctions and vulnerabilities that may lead to attacks not considered in the original cryptographic analysis.

It is not difficult to find research papers for MPC solutions that start with the phrase "*let's consider a set of n distrustful parties where no more than t can be corrupted by the adversary*". This is based on the concept of threshold access structures, introduced in Section 2.1, and is an approach very common for general purpose protocols. However, in real deployments, it may also happen that a specific subset of parties $B \subset P$ with $|B| \leq t$ is required to have access rights, because there exist

UNIVERSIDAD DE MÁLAGA

a more powerful trust relation in such a subset than in other subsets of P with a larger number of parties. This may happen, for example, because parties in B are big competitors in a consortium or because they have management power for exceptional circumstances. Therefore, this would require to adopt more general adversarial access structures. In addition, topologies are also related with this concept of access rights and trust. With topology we refer to the graph of interactions that enables the execution of the protocol in a secure way. Notice that this depends on the design used to solve the MPC problem and is very related to the trade-offs involved in terms of the amount of work to be done. Imagine, for example, a sequential topology where P_{i+1} has to do some work that relies on that done by P_i. It is clear that, even if the protocol is designed to avoid a trust relation between these parties, they still must rely on the others to execute their part. Hence, this is also a problem related with availability and the guarantees for the computation to be executed correctly, even when DoS attacks may be present. Other topologies imply asymmetric workloads and communication, where some parties may be in charge of connecting the others and therefore become interesting parties for targeted DoS. We will cover a few specific topologies more deeply in Section 3.3 when facing the problem of PSI.

To understand better how the previous notions can be analyzed, it is important to understand the different settings that can exist in MPC instances, regarding the roles that the different parties execute. Traditionally, MPC has been approached in a homogeneous manner, where all n parties are assumed to be equal, i.e., they all provide private inputs, are involved in the computation process, and consume the output. However, this notion of joint computation can be generalized regarding the roles, which may not be symmetric [63, 64]. This is not only useful for the process of modeling protocols from an abstract point of view, but also allows analyzing protocol instances in different settings which target different trade-offs. These models typically split the parties involved into three sets, which are the following: the parties that provide inputs $I = \{I_1, ..., I_\alpha\}$, the parties that obtain the output $O = \{O_1, ..., O_\beta\}$, and the parties that perform the computation $C = \{C_1, ..., C_\gamma\}$. Depending on the actual implementation, the intersection between these sets can vary, where the traditional case of a joint homogeneous computation is $I = O = C$. Another option is to deploy an *outsourced* model, as if it was a cloud computing engine, where $I = O$ and $I \cap C = O \cap C = \emptyset$. Finally, if all sets are disjoint, that means that the output consumers are different from the input providers and from the computation parties. Figure 3.1 provides a graphical representation of this, where the elements x_i are inputs and y represents the output of the computation.

In the previous paragraphs, a key aspect in technology and overall in cybersecurity has arisen: *trust*. Trust is particularly present when private or sensitive data are involved. Despite a lot of research in cybersecurity and cryptography try to provide strong guarantees to users that their data is not going to be exposed or used maliciously, achieving a 100% distrustful and secure system is practically impossible. Despite trust assumptions can (and must be) minimized, there is always the need to put some amount of trust in somewhere, which could be the deployment of the system, its administration, or even the assumption that two parties involved in the protocol will not collude between them. Therefore, a protocol can be cryptographically secure under certain circumstances, but achieving those circumstances

UNIVERSIDAD DE MÁLAGA

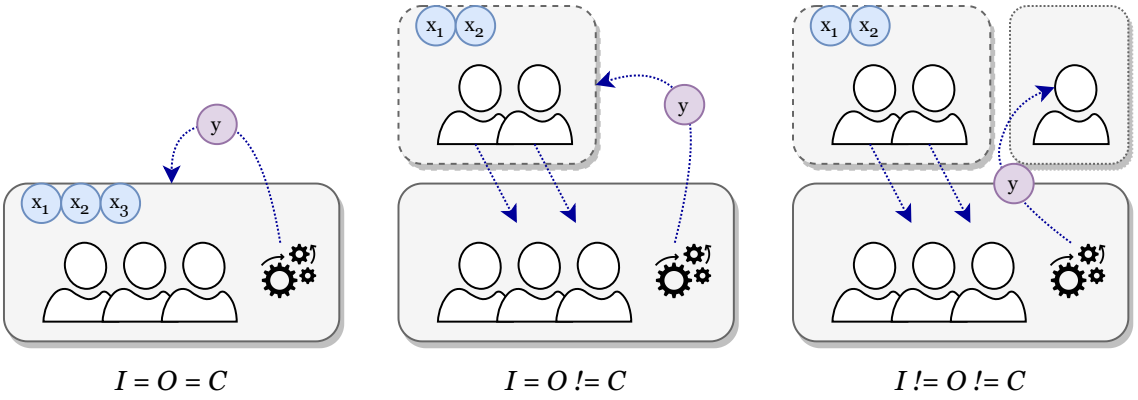


Figure 3.1: Abstract MPC model based on roles.

to happen in the real world is not always trivial. In fact, security tend to be inversely proportional to usability, therefore it is very common in real-world systems to find trade-off solutions that achieve as much security as possible while guaranteeing an acceptable amount of usability for the users.

Cloud computing is the clear example of a trusted setup. It has received great acceptance because of its usability, as it allows users to outsource capabilities for data storage, infrastructure deployment, or computing. Despite this minimizes the amount of work to be carried out by the user, it is evident that trust is fully deposited on the cloud service provider. If the cloud becomes vulnerable or decides to misbehave, users' data can be exposed or computations can be executed wrongly. However, thanks to the development of PETs, processing and storage can be outsourced to an untrusted service with guarantees. The most straightforward approach for the cloud is FHE with one user, where data is encrypted with the user's public key, shared with the cloud, computed inside the encrypted domain, and the result is retrieved such that only the user can decrypt it.

A key concept for trust is that of *trust domains*. A trust domain can be understood as a set of parties and resources that trust each other, typically under the control of the same manager/owner/operator. When analyzing security, two trust domains are assumed not to trust each other, and therefore data belonging to one domain should not be shared directly with the other, unless some mechanism for privacy is involved. In the setting introduced above, a client that wants to outsource data and computation does not trust the cloud, i.e., they belong to different trust domains. Therefore, FHE allows the former to outsource these capabilities with guarantees of non-disclosure of data. On the other hand, if one considers the different physical instances that a cloud provider can offer, e.g., location of data centers, they belong to the same trust domain because a single operator manages all of them.

Regarding MPC, it is highly related with trust domains because its own definition requires for the adversary not being able to control a valid subset of parties within the access structure, i.e., an $A \subset P$ with rights to access or modify secrets involved in the protocol. If each party involved in the protocol deploys its own instance, every honest party is assured that at least its instance is secure. Since there exist MPC approaches that achieve security against $n-1$ corruptions, honest parties can trust that no data will be exposed as long as their instance remain secure. This

UNIVERSIDAD DE MÁLAGA

approach relies on a design notion with data governance, i.e., each party controls at least one part of the data shared. However, this implies that each data owner must deploy its own protocol instance, therefore using own resources for that and minimizing usability.

Lindell et al. wrote an interesting post [65] where the problem of deploying MPC instances is discussed. They state that *designing* and *implementing* an MPC is only the first step of *deploying* an *end-to-end* MPC application. Authors analyze some real deployments of MPC in Signal, Coinbase, the Internet Security Research Group (ISRG), and the UC Berkeley Security Group, and they also gather a few interesting questions that must be answered before deploying MPC in production:

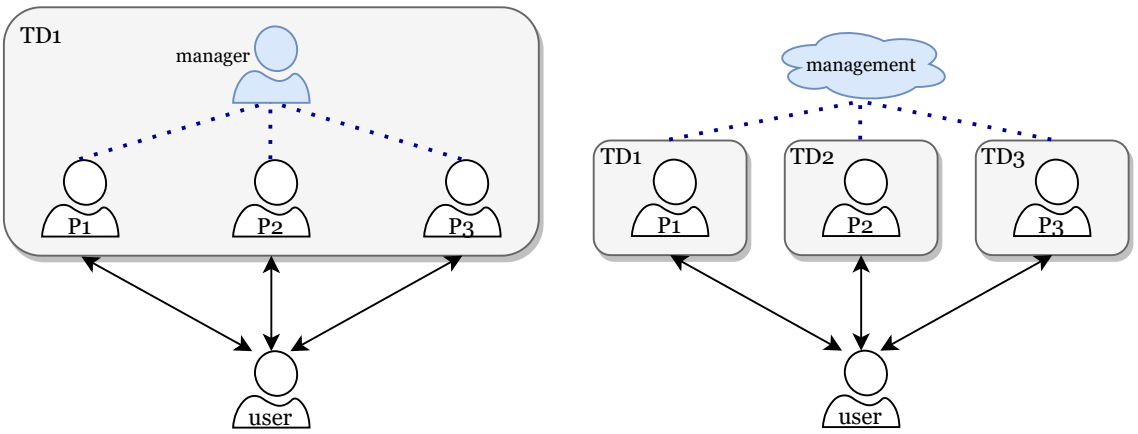
- Who are the parties among which we can run MPC?
- Who assigns session IDs across multi-round protocols? Are we trusting a centralized entity to assign them?
- How is key management done? If parties are storing intermediate MPC secret shares, where are they stored?
- How do the n parties authenticate to one another?
- How does a client authenticate to the n parties?
- How do we ensure that MPC parties have secure communication channels between them[1]?
- How do we resolve the synchronization challenges that manifest when communicating between parties?

As introduced in Figure 3.1, another approach to use MPC can be by outsourcing the computation to an external set of nodes, instead of letting the user to deploy an MPC party himself. This can be deployed, e.g., in a cloud setting. The main problem with that is that trust must be deposited in those nodes, and data owners must be somehow ensured that there will not be enough nodes colluding to reconstruct secret data or manipulate the computation results. In the case of deploying the computation instances in the cloud, one could think of using a different cloud provider for each instance, therefore avoiding a single cloud provider to have access to all private inputs, even if they are secretly shared with the nodes. However, there is still the problem that the application developer has administrative access to the different machines, even if they are in different cloud providers. In this case, the application developer is a single point of failure. This problem is shown in Figure 3.2 and has been addressed by Dauterman et al. [66]. They propose a solution that allows a developer to deploy a distributed-trust system with public auditability by using TEEs, such that anyone can be ensured that the application is correctly deployed with respect to the different trust domains. Basically, each trust domain runs the application (this can be specified by the MPC code) inside a TEE, and the client obtains the hash of the running code, which is ensured to be the actual code running because of the TEE. Then, the client can compare the different hashes obtained, which are supposed to be the same for all honest parties. In addition, if

[1] One example is that of TLS channels, where some mechanism must guarantee trust in public key certificates, such as PKI.

UNIVERSIDAD DE MÁLAGA

the code is shared in an open source way, the client can compare these hashes with the hash of the open source code, to be ensured that the MPC is actually running the computation desired.



(a) Centralized trust domain. (b) Decentralized trust domains.

Figure 3.2: MPC outsourced models regarding trust domains.

Following this line of research, the work in Kaviani et al. [67] proposes a solution to setup on-demand distributed trust by using different cloud providers (each of them in a different trust domain), and a relay. This work is based on a secure protocol that allows the user, i.e., the input provider, to select and setup the cloud instances to perform an outsourced MPC. However, this work is limited to a single input provider, thus the applications supported are limited (secret recovery, signing, decryption, data-sharing, PIR, or data freshness).

Extending *Kaviani et al.* to multiple input providers. Ideally, a system for outsourced computations in settings with distributed trust would allow more than one input provider in order to maximize the use cases. This would require a deeply analysis in the strategies used and the security assumptions involved, however, we provide a brief discussion about how this could be addressed. First, it is interesting to distinguish between two phases: (1) the coordination and contractual terms, i.e., the agreement of the parties about what they want to compute and the security assumptions they want, and (2) the trust domain selection itself, which leads to set up the scenario for the MPC execution. For simplification, we assume a list $L = \{L_1, ..., L_N\}$, where each L_i is a different computation provider. Ideally, each L_i is assumed to be in a different trust domain, but this notion can be very subjective for different clients. A first naive solution would be to apply a secure random selection function to select the subset of providers in L that will execute the MPC. It is key for the random selection process not to be biased (and ideally, publicly verifiable), otherwise malicious users may try to select those trust domains where they have influence. A slight variation takes a first phase where deny-listing is applied by different clients, thus those providers included in the list will never be selected by the secure random selection function. The main problem with this approach is that clients may be such restrictive that they never reach an agreement in a valid subset of L. The last approach would be to use a trust-maximization function, which takes as input a weighted version of L by each client for the trust they have in each provider, and maximizes the trust in the selected subset of L for all the clients involved.

UNIVERSIDAD DE MÁLAGA

Regarding other works, James Reyes gave an invited talk in Real World Crypto 2022 [68] where he introduced the use case of MPC in digital advertising, where lift and attribution metrics can be computed and aggregated on private linked records generated by the advertisers and the platforms where they offer ads. This talk targets some of the difficulties about deploying MPC instances, overall in use cases where the volume of data to be computed is large, and therefore the cost for the MPC is high in monetary terms. He also introduced the Private Computation Framework from Meta, that address the advertising use case in a 2PC setting and tries to facilitate the deployment of MPC to the advertisers, which is known to be a difficult and slow process.

Finally, as it will be covered in Chapter 4, trust and reputation are key concepts in blockchain ecosystems and can be used to set up different trust domains in MPC deployments. The survey of Bellini et al. [69] analyzes the different approaches proposed in the literature for these concepts and provides a taxonomy for distributed trust and reputation management systems. For trust management, different approaches can be used, such as credential-based, reputation-based, or social network-based, regarding if those which are not trusted are the service providers or the service requesters (i.e., clients).

3.2 Relevant protocols and applications in MPC

Despite MPC protocols can be addressed from the general-purpose setting, i.e., protocols that allow to compute (almost) any function on private data by describing the function as a circuit, there also exist a set of problems that can be addressed from other techniques, specifically applied to solve these protocols more easily and/or efficiently. This section is devoted to introduce and discuss some of the main extended problems in MPC which are addressed from special-purpose techniques, which do not only solve their problems for a straightforward application, but that offer interesting building blocks that can be combined with more complex MPC protocols in order to enable additional applications.

3.2.1 Distributed Random Number Generator

Generating truly random numbers is a complex task very related with cryptography. Many protocols require an unpredictable source of randomness in order to remain secure, but software approaches in computers only achieve pseudo-randomness, i.e., values than seem random as long as the *seed* is not revealed. Therefore, this seed is a point of failure, because it can be leaked or worst, modified by an attacker to predict the next random values that will be generated. In fact, bad implementations of pseudo-randomness sources have been exploited by attackers in cryptographic libraries many times. There are approaches that generate truly randomness from hardware devices, by using aspects such as signal noise, that can fit quite well in scenarios with a single computer. However, this section approaches the problem from a decentralized network perspective, where a set of parties want to share a common and trusted source of randomness (typically known as *random beacon*), which is not biased by a malicious party. This problem can be generalized as a trusted functionality, as shown in Functionality 2. Note that, from the functionality

UNIVERSIDAD DE MÁLAGA

perspective, Alice and Bob do not provide inputs (since the functionality is trusted and assumed to use a TRNG), but they will probably provide some randomness as input in a real implementation.

Functionality 2: Distributed Random Number Generator

Alice's private input: *empty.*
Bob's private input: *empty.*
Public common input: sampling range (i, j).
Output: a random k, with $i < k < j$.

A DRNG can be easily implemented with general purpose MPC. From an abstract point of view, it can be generalized with two phases: (1) a *commit* phase, where each party P_i commits to a random value r_i, and a *reveal* phase, where each r_i is revealed and the random beacon r is built from the combination of the set $\{r_i\}_{i=\{1,\dots,n\}}$. It is easy to notice that, given a set of randomly generated r_i values, the combination r will remain unbiased as long as the adversary controls $k \leq t$ parties. A very common way to implement this is by using Publicly Verifiable Secret Sharin (PVSS). In PVSS, the distribution phase implies encryption of the shares (commit phase) and verifiable proofs that they are valid shares of a secret, which prevents a malicious dealer. Then, in the reconstruction phase, proofs of valid decryption avoids malicious parties from providing biased shares.

Some examples are the work of Cascudo et al. [70] or Syta et al. [71], which achieve PVSS-based DRNG with communication complexity $O(n^3)$. Also, the work of Schindler et al. [72] applies PVSS in a Byzantine Failure setting combined with random leader selection in each round, and improves communication complexity, which is $O(n^2)$.

On the other hand, the RandDAO project [73] also uses the commit+reveal phase, but applied on a blockchain setting. It allows parties to contribute by performing some staking and sharing commitments of randomly sampled values r_i. Later, in the reveal phase, those commitments are opened and all the r_i are combined to obtain r. This protocol propose two versions: a hash-based commitment approach, or a BLS signature aggregation method to generate the random numbers. In addition, it reaches $O(n)$ communication, and since it is an incentive-based approach, contributors are rewarded when their openings are correctly verified.

Finally, the approach used by Nguyen-Van et al. [74] relies on Homomorphic Encryption for randomness production and ZKP for the verification phase. This is another way of achieving $O(n)$ communication complexity, but at the expense of more intensive computation.

DRNG presents many use cases, and is typically deployed as a base building block that enables more complex applications. A straightforward example is a lottery application, where MPC ensures (maybe in a transparent and verifiable way) that the result, i.e., the elected winner, is effectively random and unbiased. An interesting aspect of DRNG is that it fits well with threshold assumptions, which allows to deploy a large set of randomness producers where any valid subset is enough to

UNIVERSIDAD DE MÁLAGA

produce an unbiased output. Therefore, this can be used as a building block which is strongly secure against DoS attacks. In addition, notice that DRNG works well with multiple trust domains, but its effectiveness is reduced if a single manager exists. While in such a setting one achieves protection against external adversaries attacks to bias the randomness generated, security cannot be ensured if the attacker aiming to bias the result is the manager or someone inside the unique trust domain.

3.2.2 Threshold cryptography

Threshold cryptography is the family of MPC protocols that imply a secret shared private key $[\![k]\!]$, and where the application of the function $f_k(\cdot)$ over some data, which takes the private key as an input, is done without computing the original value of k.

These protocols are very useful in scenarios where trust has to be shared among a group of parties, and this is one of the reasons because they are widely accepted in blockchain solutions. Having a key k split in a set of servers implies, firstly, that the key is not exposed in a single location, and therefore risk is minimized with respect to external adversaries. Second, it enhances security of delegated processes (or those that imply a third party) because a single key holder cannot take full action on data, e.g., to decrypt or sign. Instead, at least $t + 1$ holders are needed to complete the cryptographic process. However, note that this second implication is directly related with the concept of trust from Section 3.1, where security against an external adversary is always achieved if the key is correctly split, but security against an unilateral decision requires the key holders to be correctly deployed in different trust domains, otherwise the central operator could coordinate them to achieve the desired operation against the will of the user.

Finally, an interesting remark is that, from all MPC protocols, those that fit in threshold cryptography are widely understood as software-based Hardware Security Modules (HSM). The reason is that they achieve the same set of operations: key generation, storage, and protected usage of keys. However, instead of relying on a hardware device for protection of the keys, the assumption relies on the MPC committee of holders, i.e., that enough members will remain honest. Note that key generation process is, in fact, directly related to the concept of DRNG from Section 3.2.1. Each of these approaches has their own advantages and disadvantages, where what MPC can offer better than HSMs is more flexibility on the way that keys are managed, e.g., for backups or changes of location, and the possibility to distribute trust. However, the prize to pay is typically larger computation requirements and latency.

3.2.2.1 Threshold block ciphers

The first MPC evaluation of the AES functionality was presented by Pinkas et al. [75]. This work understands AES as a natural 2-party computation with two private inputs, as shown in Functionality 3: Alice owns the plaintext m and Bob owns the secret key k. At the end of the protocol, Alice obtains $c = Enc_k(m)$. It is interesting to notice that this is the exact definition of an Oblivious Pseudo-Random Function (OPRF), where Alice obtains the evaluation of $PRF_k(\cdot)$ on her private data without revealing her data to the evaluator.

UNIVERSIDAD DE MÁLAGA

Functionality 3: Block Cipher-based OPRF (2-party)

Alice's private input: A secret plaintext m.
Bob's private input: A secret key k.
Output: The ciphertext $c = Enc_k(m)$.

From the work of Pinkas et al., AES has remained as a commonly accepted block to measure general purpose MPC protocols. As stated in Damgard et al.'s work [76], the reasons for this are that AES is a highly non-linear and well understood function, and that its highly mathematical structure allows to investigate various optimization techniques in a single function. In addition, a third aspect is that the oblivious evaluation of AES has its own interest for different applications.

Differently from the initial OPRF focus, subsequent works in MPC-based AES (and other schemes later) moved to a setting where the key is split among the parties involved in the MPC [76, 77, 78], and this is closer to the notion of software-based HSM mentioned in the introduction of this section. The first protocols provide MPC implementations with the Bit Decomposition technique, which takes advantage of the algebraic representation of the AES Sbox. However, the fastest latency is achieved in Keller et al.'s protocol [77] which is based on Lookup tables. Authors report 0.928 ms in the online phase to evaluate one AES block. However, to achieve such fast performance, their protocol requires precomputation in the offline phase, which means additional communication and storage. Regarding this aspect, the work of Durak and Guajardo [79] do not improve latency (1.5 ms per AES block) but achieve a 94.93% of savings in storage.

Table 3.1: Latency comparison (in ms/op) in MPC-based PRFs (from *Grassi et al.*).

Setting	**AES**	**LowMC(M4R)**	**NR(128)(const)**	**Leg(bit)**	**MiMC(cube)**
LAN	7.713	4.148	4.549	0.349	5.889
WAN	2640	659	478	202	3691

Finally, the work of Grassi et al. [80] takes a different direction for MPC-based block ciphers. Since AES is very unnatural for MPC (AES is byte oriented and MPC typically works in $\mathbb{F}_p$ arithmetic) and most symmetric key schemes can be constructed from PRFs, this work targets a set of new MPC constructions based on algebraic PRFs. This allows an easier mapping from the local primitive to its MPC version, therefore obtaining better performance. More precisely, the work of Grassi et al. provide MPC constructions of the following protocols: LowMC, Naor-Reingold, Legendre Symbol, and MiMC. For completeness, we provide in Table 3.1 the comparison provided by the authors between these protocols and the AES version of Damgard et al. [76]. They do not consider the protocol of Keller et al. because of the large preprocessing it requires. One of the motivations to include Table 3.1 is to claim that, while latency comparisons are typically provided in a LAN setting, considering the WAN setting is also required for real implementations, as it happens in Grassi et al.

UNIVERSIDAD DE MÁLAGA

3.2.2.2 Threshold signatures

Functionality 4: Threshold signature

User's public input: A message m.
Functionality private data: The private key sk.
Public common input: The public key pk.
Output: The signature $\sigma = Enc_{sk}(m)$.

A threshold signature scheme, as shown in Functionality 4, is an MPC scheme where the private key is distributed among a set of key holders and they need to collaborate in order to produce a valid signature. There are normally two phases for the signature generation: first, the generation of partial signature shares, one per key holder, and second, the combination of partial signatures to obtain a single signature of the message. While there are different approaches for group production of signatures, a proper (t, n)-threshold signature scheme ensures unforgeability, i.e., that no adversary controlling at most t key holders can generate a valid signature of a new message with non-neglibible probability. Threshold signature schemes has been proposed for most widely adopted signature schemes, either RSA [81, 82], DSA/ECDSA [83, 84, 85, 86], or Schnorr [87, 88, 89].

Some of the early works assumed a dealer to perform the key generation procedure, but some works by Gennaro et al. [88] set the focus on Distributed Key Generation (DKG) to let the set of untrusted key holders jointly generate the shares through an MPC. This is typically achieved by some sort of Verifiable Secret Sharing or Homomorphic Encryption.

While most schemes propose a Non Interactive verification phase, which is interesting because the signature can be verified as if it were a standard single-party generated signature, therefore enabling flexibility and efficiency. However, there are schemes such as BLS [90, 91] that allow to aggregate partial signatures generated with independent key pairs. This allows to avoid the DKG phase at the expense of adding a public key aggregation phase before verifying the signature.

Threshold signatures play a key role in many decentralized settings such as blockchain applications [83, 85, 86, 92]. In this area, they can enhance security of users' wallets by splitting the private key among a couple of different devices (either hardware or online provided storage services). This avoids a single point of failure for non desired transactions to be generated, makes the signature procedure failure tolerant because only $(t + 1)$-out-of-n partial signatures are needed, and also adds the possibility to regenerate the key shares if $n - t + 1$ are lost (in Bitcoin, losing the wallet private key means losing all the Bitcoins owned by that account). Similar to that use case, there are applications in the blockchain ecosystem that decentralize the ability to disclose some data or to start some operation. If the process can be triggered by a signature, e.g., for cryptocurrency locks in blockchain bridges, the private key can be distributed by a set of "guardian" nodes with the assumption that at most $n - t$ nodes will remain honest. In Section 6.3 we take advantage of a decentralized mixer for Bitcoin that relies on threshold signatures to protect users' cryptocurrencies from being stolen.

UNIVERSIDAD DE MÁLAGA

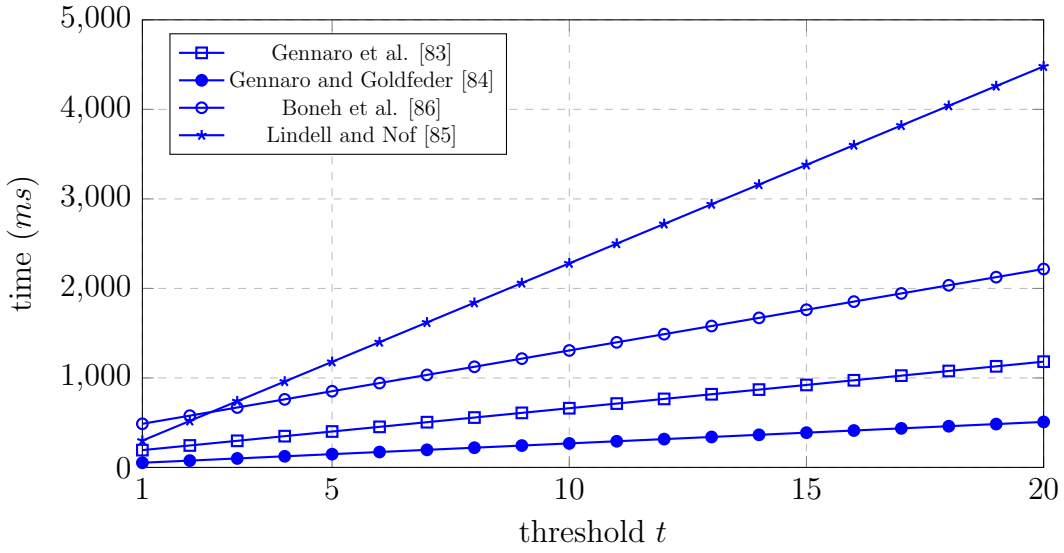


Figure 3.3: Time (ms) for signature generation in threshold signature schemes.

Finally, Figure 3.3 shows a running time comparison for the most recent protocols that provide benchmarking. It is important to note that these times are only for the signature part, but not for the DKG. Also, while the DKG complexity depends on n, the signature generation only takes t into account, since not all the key holders are needed to generate a valid signature but only $t + 1$. In addition, it must be noticed that, from the protocols compared in Figure 3.3, Lindell and Nof [85] is the only one that considers communication times in addition to the local computation time of the different parties.

3.2.2.3 Threshold decryption

Functionality 5: Threshold decryption

User's private input: A ciphertext $c = Enc_{pk}(m)$.
Functionality private data: The private key sk.
Output: $m = Dec_{sk}(c)$.

A threshold decryption or threshold public key encryption scheme, as shown in Functionality 5, is a public key scheme where decryption capabilities are distributed. Use cases range from access control to sensitive information to electronic auctions, where only a valid subset of decryptors can publish the final result. This also works for electronic voting, but to publish the tally.

The standard approach to build a threshold public key encryption scheme is by secret sharing the secret key and producing a single public key for the group of decryptors, such that any ciphertext encrypted with pk can be decrypted with a sufficiently large subset of at least $t + 1$ valid decryptors. This is normally done by a DKG algorithm. Some examples are Canetti et al. [93], which is based on the DDH assumption, or Bendlin and Damgard [94] and Dahl et al. [95], which propose schemes for LWE-based FHE cryptosystems.

UNIVERSIDAD DE MÁLAGA

There is a variation of threshold public key encryption schemes, which is those that are dynamic. This implies that a new party willing to participate as a decryptor can easily join to the system without the need of a resharing of the secret key. In a dynamic scheme, the key generation algorithm executed by P_i produces a pair of keys (pk_i, sk_i). Then, the data owner can decide to which subset of decryptors to encrypt data. The main caveat is that the public key of the target group is not a single key, but the set of all public keys involved in the subgroup. Some schemes that target dynamic schemes are, Ghodosi et al. [96] for ElGamal and RSA schemes, and Delerablée and Pointcheval [97] and Qin et al. [98], which are based on pairings.

3.2.3 Private Set Operations

Private Set Operations (PSO) is an area of MPC that has received many attention from researchers, because these problems present a quite large list of applications and also are more efficient when solved with specific cryptographic techniques than when using general purpose MPC. They basically allow a set of parties to compute a functionality (which is the *operation*) on their private sets, therefore it is a specific problem in MPC. The general functionality of PSO is introduced in Functionality 6.

Functionality 6: Private Set Operation

Alice's private input: set A.
Bob's private input: set B.
Public common input: an operation $OP(\cdot)$.
Output: $OP(A, B)$, and nothing else.

Note that OP can be any function on sets, but two main operations have received the most attention: Private Set Intersection (PSI) and Private Set Union (PSU), which only output the intersection or the union of the private input sets, respectively. From the early beginning, some works have focused only on PSI [99], others only on PSU [100, 101, 102], and others on both of them [103, 104], because there are techniques that can be applied to both problems with only a few modifications. These specific-purpose protocols normally imply some sort of *oblivious mapping* of the set elements, thus they can be compared (for intersection or aggregation) without exposing which particular element is being compared at each step.

There are other variations apart from the basic PSI/PSU that have received great attention. One of the most useful is PSU/PSU-Cardinality (also PSI/PSI-CA) [99, 105], where the protocol does not output the items in the intersection (or union) but the number of items in it. Other variations are threshold-PSI (t-PSI) [99], where the intersection is only returned if its cardinality reaches a certain threshold $t < l$, (with l the length of the smaller set), or Private Subset Inclusion (PSInc), where the output returns whether Alice's input set A is a subset of Bob's input set B or not. The work of Laur et al. [106] uses PSInc as a subroutine to achieve Private Itemset Support Counting (PISC), where $supp_B(A)$ is defined as the subset of all items in A that forms a subset in B, and has applications in data mining.

UNIVERSIDAD DE MÁLAGA

From all the variations in PSO, the intersection has clearly been the most researched one, and because of that, we provide in Section 3.3 an extensive analysis of all its variations, design approaches, and efficiency analysis.

3.3 A review on Private Set Intersection

Private Set Intersection is one of the MPC problems that has received more attention from researchers (see Figure 3.4b below). This is mainly for two reasons: it has many useful applications in different fields, and cryptographic constructions designed specifically to solve PSI show better performance than general-purpose approaches. For those reasons, PSI is normally used as the leading example when introducing special-purpose MPC protocols [1, Section 3.8].

As part of this thesis, a Systematic Literature Review was conducted on PSI, and a summary of the main results is presented in this section. The motivation behind conducting an SLR on PSI was mainly to centralize the different approaches that have been used in the literature to solve it, which are very diverse and present varying trade-offs. At the date of conducting the review, no other work had focused so much on PSI and provided structured insights about all the variations existing. This analysis has not only provided a taxonomy of PSI and a structured analysis of its variations, but also a critical analysis from the point of view of performance, stating the most efficient techniques depending on the scenario to execute the protocol and the implications of using more or less secure variations of PSI. One of the main objectives was also to help in answering the question about if PSI is ready for realistic practical implementations or still remains mainly as a theoretical protocol.

3.3.1 Systematic Literature Review methodology

The purpose behind conducting an SLR is to do the review process using a structured and well-known methodology, which allows replication by others researchers and hence reaching similar conclusions. An SLR is composed by three major phases: (1) an initial planning, (2) conducting the review, and (3) reporting the results. The first thing to do for planning is to establish a set of *research questions*, which are the ones that will guide the whole SLR process. We established four questions for PSI:

(RQ1) What are the most typical building blocks used in PSI protocols?

(RQ2) What are the existing practical implementations of PSI?

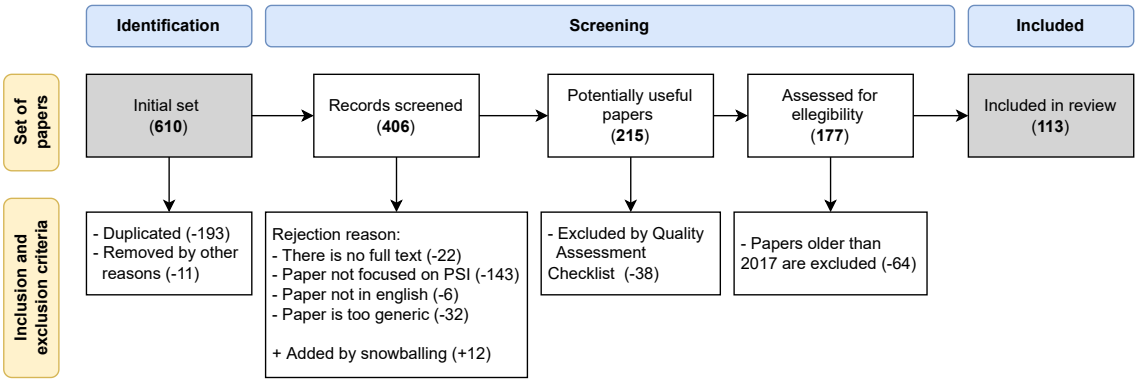
(RQ3) What are the existing use cases for PSI?

(RQ4) What are the major challenges to achieve efficient practical implementations of PSI?

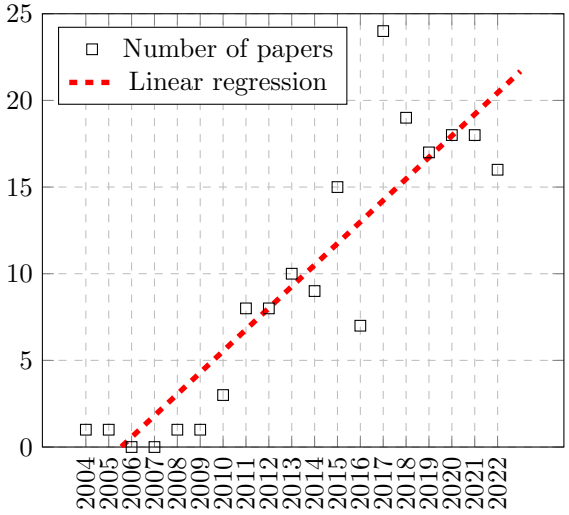
These research questions, clearly targeting the practical aspects of PSI, lead to the construction of a search string to be used in the main data sources for research:

UNIVERSIDAD DE MÁLAGA

((private OR secure) AND set AND (intersection OR computation OR operation)) AND (((protocol OR algorithm OR building block) AND ((implementation OR framework OR development) OR (efficient OR performance))) OR use case OR practical)



(a) PRISMA flow diagram of inclusion/exclusion phases.



(b) Number of accepted papers per year and regression.

Figure 3.4: SLR methodology for screening.

To be precise, the digital libraries used as data sources are the following: *Springer Link*, *Scopus*, *Science Direct*, *ISI Web of Science*, *IEEE Digital Library* and *ACM Digital Library*. After using the search string in these data sources, an initial set of papers is gathered and then it goes through a screening process in order to discard papers based on different criteria. Figure 3.4a shows these phases in a flow diagram according to the PRISMA methodology for reporting SLR results. The exclusion criteria applied during a rough screening lead to a subset of 215 potentially useful papers, and then a Quality Assessment Checklist is applied to each candidate in order to consider only those papers with the best quality, leading to a subset of 177. The final decision to only consider the papers from 2017 has been taken because of three reasons. First, the set of papers with enough quality after the Quality Assessment Checklist was still large enough. Second, the trend of publication increases

UNIVERSIDAD DE MÁLAGA

over the years (as shown in Figure 3.4b), and therefore most of the papers could be gathered in recent years. And third, because it was not expected that the oldest papers could contribute enough to the SLR compared to the newest. Finally, 113 papers were considered in the review, and we refer to the published paper to access the complete list.

3.3.2 Scenarios and applications

Before delving into the technical details that allow to achieve secure protocols for PSI, this section provides a brief gathering of different scenarios and applications where PSI has been proposed. These have been collected through the analysis of the whole literature in the review, and classified on a couple of groups. This is a beneficial work because understanding the specific characteristics of the use case or the environment where PSI is deployed helps in selecting a proper protocol that offers nice trade-offs between security and performance.

Related with security and more precisely with who obtains the output, a first classification is between *one-way* PSI, if only one of the parties obtains the intersection, and *mutual* PSI, if all parties obtain it. It is clear that this is a first discriminatory feature for protocol selection. A second classification is more related with which type of data is being processed and the relation between the participants. Here, *balanced* PSI refers to those protocols where set sizes of all parties are approximately similar, while *unbalanced* PSI refers to those where there is a significant difference. Notice that protocols from any of these two groups can be used for any use case, however, what unbalanced PSI does is offer specific designs that are much more efficient when set sizes are asymmetric. In fact, prior works in PSI did not consider different set sizes, but unbalanced PSI was acquiring more attention due to specific use cases that were identified to be solvable by PSI and present this characteristic.

First, regarding balanced PSI, this normally fits in the "cooperation" area, where a set of parties belonging to enterprises, researches, medical, or financial institutions, want to access to greater analysis or statistics than what they would be able to achieve solely on their own data. This approach is very typical in scenarios where data cannot be shared with others due to being high sensitive, such as data related to clients or enterprise strategic secrets. Some applications that have been proposed are secure computations on genomics or on medical data, data mining on private data, measuring ad conversion rates, security incident information sharing, collaborative botnet detection, aviation safety, satellite collisions matching, or private network interference discovery [107, 108, 109, 110, 111]. Regarding some common aspects, one can assume that datasets in this setting are likely to be large if they hold analytic information from customers, for example. In addition, feeding a PSI with more than one large dataset imply some degree of flexibility regarding the amount of resources available and the delay the computation will take. These use cases may rely on cloud instances to deploy a PSI that takes quite a large time, but that is not an issue for the correct working of other services.

Regarding the unbalanced scenario, it fits with the client-server model, where the server is assumed to have a large database of items and the client has a much more shorter set to feed the PSI with. The borderline case is when the client's dataset has only one item, where the problem will be called Private Membership Test (PMT).

UNIVERSIDAD DE MÁLAGA

In fact, it is usual to build PSI from a PMT solution. Some applications that fit in this scenario are malware detection by signature, checking of compromised credentials, private database querying, biometric authentication, private contact discovery, shoppers' personal preference matching, or COVID contact tracing [112, 113, 114, 115]. Regarding specific characteristics, it may happen that the client's device is limited in the amount of resources it has, and therefore the PSI must be adapted to this. Derived from this, such a constrained device is not supposed to be working on the PSI for long times, therefore delays are a limitation here (imagine an smart phone or an IoT device).

There are other applications where some parties want to perform PSI, but they depend on a central server to perform or coordinate the computation, e.g., private ride-sharing, online matchmaking, social network discovery, or online dating [116, 117, 118].

A final remark is that a balanced PSI scenario is more likely to need a mutual PSI protocol, while the unbalanced scenario is closer to the one-way PSI, where the client is the one obtaining the intersection result.

3.3.3 PSI taxonomy and main works

PSI has many variants, which derive from a common root but offer different properties. The classical definition, as introduced in Section 3.2.3, is to let two parties computing the intersection of their private sets. A straight variant is MP-PSI, which extends the problem to more than two parties. Other variants compute some function related with the intersection itself, such as Threshold PSI (T-PSI), where the intersection is only obtained if it reaches a certain threshold of items, or PSI Cardinality (PSI-CA), where the output is not the intersection but its size. Other variants are Size-Hiding PSI (SH-PSI), which hides the cardinality of the input sets, or Authorized PSI (A-PSI), which requires a previous authorization phase on the items of the set. Finally, Outsourced PSI (O-PSI) or Verifiable-Delegated PSI (VD-PSI) allow the parties to outsource their "encrypted" sets to an external party which is in charge of computing the intersection. All these variants are not only interesting from the point of view of their applications, but also because different building blocks suit better for different PSI variants, therefore incurring in different trade-offs.

The approach for solving PSI is typically to obliviously map the elements inside the sets to a different structure, such that it allows to compare the mapped elements. This way, coinciding elements are known to belong to the intersection, but not coinciding elements do not expose anything about their preimages. PSI solutions typically rely on two phases, solved by different families of building blocks. First, those in charge of encoding data in more efficient structures, to minimize overheads, and second, those in charge of achieving the oblivious map of the set elements. A naive and insecure way of solving PSI would be by using hash functions. Let's assume that Alice, with set $X = \{x_1, x_2, ..., x_n\}$, and Bob, with set $Y = \{y_1, y_2, ..., y_n\}$, want to perform PSI. They each locally compute the hash of each of their elements in their set, i.e., $H(X) = \{H(x_1), H(x_2), ..., H(x_n)\}$ and $H(Y) = \{H(y_1), H(y_2), ..., H(y_n)\}$, and exchange the hashed sets with the other party. Then, the intersection is just given by the common elements in both $H(X)$ and $H(Y)$. Even when this protocol is very efficient, it is insecure because each of them can precompute the hash values

UNIVERSIDAD DE MÁLAGA

if the message space is small, and learn what are the elements of the other party outside the intersection.

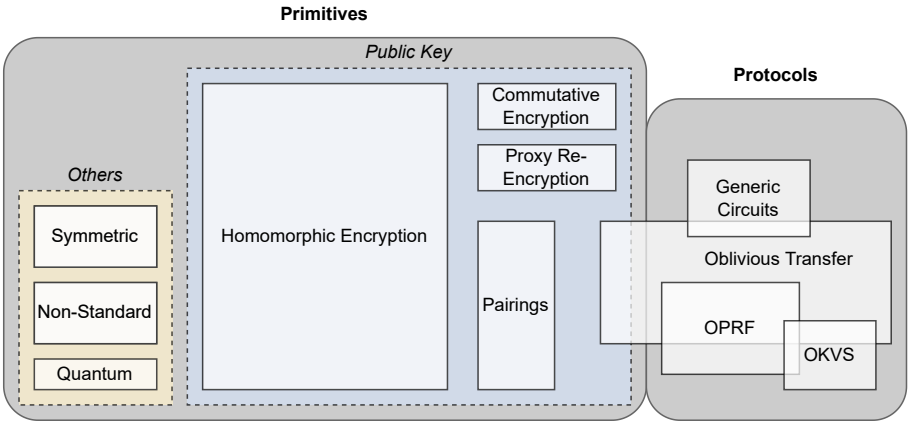


Figure 3.5: Classification of building blocks to solve PSI.

From all the building blocks for the oblivious mapping that we have found in the reviewed set of literature, we conducted a classification, mainly distinguishing between *primitives*, which are mainly computed locally, and *protocols*, which imply some kind of interaction in their own definition. All these building blocks are depicted in Figure 3.5 using a Venn diagram, where the area of each building block is directly related to the number of papers that have been proposed (in our reviewed set) to solve PSI with that building block as base construction. It can be observed that Homomorphic Encryption and Oblivious Transfer are the two most adopted building blocks.

3.3.3.1 Encoding data with hash-based structures

Techniques to encode data are very important in PSI protocols because they allow to reduce the amount of resources needed to compute the intersection, i.e., the number of invocations to the primitive needed to obliviously encode each value, and the amount of communication required. Most techniques rely on hash-based structures, from where we introduce some of them.

One option is to employ a naive hash table, where a hash function $H : S \rightarrow M$ maps elements from the domain of the input data to the domain of hash images M, where $|M| < |S|$. Each hash address points to a bin with a specific maximum capacity $|B| \geq 1$, therefore two or more items can be mapped to the same bin. This allows to reduce the number of comparisons from $O(n^2)$ to $O(n_B|B|^2)$, where $|B|$ is a pre-defined constant and n_B is the number of bins.

Other widely used structures are probabilistic filters, which are also based on hash functions. The two main are Bloom Filter (BF) and Cuckoo Filter (CF). These structures allow queries on elements, and they both allow a rate (typically fixed to be small enough) of false positives, but do not allow false negatives.

A BF insertion maps an element x to k places into a bit array, setting those bits to 1. The positions are determined by a set of hash functions $\{H_1, ..., H_k\}$. To check if y is inside the BF, the values $H_i(y)$ are computed and if $\wedge_i BF[H_i(y)] = 1$, it can

UNIVERSIDAD DE MÁLAGA

be said with high probability that $y \in BF$. It is worth noting that, given BF_X and BF_Y two BFs encoding sets X and Y respectively, $BF_{X \cap Y}[i] = BF_X[i] \wedge BF_Y[i]$.

On the other hand, a CF insertion maps an element x to one of two possible buckets into a hash table, where there are n_B buckets and each can have more than one entry. The buckets for x are determined by Cuckoo Hashing, i.e., $b_1 = H(x)$ and $b_2 = b_1 \oplus H(fingerprint(x))$. This structure allows to relocate elements to make the table more balanced. In addition, since the value stored is the fingerprint of x, elements can be deleted.

3.3.3.2 General PSI

Since most protocols target the general problem of PSI, we will introduce some of them, classified under the main building block that allows the computation of the intersection.

PSI based on HE

Many works rely on HE to solve PSI. From these, many follow the Oblivious Polynomial Evaluation (OPE) approach, which was very common in the oldest works [103]. Essentially, in an OPE protocol, Alice knows a value a and Bob knows a polynomial $P(x)$, and at the end of the protocol Alice learns $P(a)$ and nothing else. To solve PSI with OPE, Bob can compute the polynomial as $P(x) = \prod_i (x - y_i)$, where $y_i \in Y$. This polynomial has the property that each element in Bob's set is a root, therefore $P(y_i) = 0$. Then, it just suffices for Alice to perform n OPEs, one per each $x_i \in X$, and those where $P(x_i) = 0$ would imply that x_i is in the intersection. To achieve privacy for both parties, a common approach is to compute the intersection polynomial as $(\gamma_1 \otimes P_A) \oplus (\gamma_2 \otimes P_B)$, where P_A and P_B are Alice's and Bob's encrypted polynomials, γ_1 and γ_2 are randomly sampled polynomials, $\oplus$ denotes addition of two encrypted polynomials, and $\otimes$ denotes scalar multiplication between a plaintext and an encrypted polynomial. For example, Carpent et al. [119] proposes a PSI-with-Projection (PSI-P) protocol based on OPE using Additive Homomorphic Encryption. However, since it is quadratic in computation, it also proposes an Oblivious Pseudo-Random Function (OPRF) variant which achieves linear computation. We will extend more in OPRF in the section for PSI based on OT.

Other works such as Davidson and Cid [120] or Wang et al. [118] rely on BFs and Additive HE to compute the intersection, taking advantage of the BFs property introduced in Section 3.3.3.1 that allows to compute the intersection easily. Both protocols achieve linear complexities, but require single bit encryptions, which may be large depending on the size of the BFs.

Chen et al. [121] achieves the most efficient PSI protocol based on FHE. Even when FHE is quite expensive in terms of computation, they propose a simple protocol in nature with some implementation improvements that reduce significantly the overload. The core protocol follows the same objective that OPE-based PSI, which is letting Alice obtain 0 when her item is in the intersection. To do that without using polynomials, Alice encrypts her set, obtaining $(c_1, c_2, ..., c_n)$, and sends this to Bob. Then, Bob computes $d_i = r_i \prod_{y \in Y}(c_i - y)$, per each c_i, and where r_i is a random value. Then, Alice receives all the d_i, decrypts them, and if $d_i = 0$, that

UNIVERSIDAD DE MÁLAGA

means that x_i is in the intersection. This protocol is quadratic in computation and linear in communication. To increase the performance, the authors use batching thanks to SIMD, which allow to encrypt more than one item inside one ciphertext, and hashing techniques to reduce the number of comparisons for the PSI. They also employ windowing and partitioning in Bob's set to reduce the depth of the circuit to compute the values d_i. Finally, they rely on modulus switching to reduce the length of the ciphertexts sent through the network.

On the other hand, Jiang et al. [122] extends the work of Chen et al. to a fully-malicious and verifiable setting. They use the FHE construction in Chen et al. for privacy-preservation, but they add an initial phase of OPRF to protect against a malicious receiver, and a verifiable computation phase for inner product based on bilinear pairings to protect against a malicious sender. However, it increases communication complexity by $O(|X|)$ and computation complexity by $O(|Y|+|X|)$ compared to Chen et al.'s work.

PSI based on OT and OPRF

When based on OT, one approach for PSI is the Garbled Bloom Filter (GBF) paradigm [123, 124]. A GBF G is a special type of BF where each cell does not contains a bit, but a λ-bit string. For each element y in Bob's set, he initializes the cells such that $y = \oplus_i G[H_i(y)]$, and fills those that remain empty with random string. On the other hand, Alice initializes a standard BF B with her set X, and engage in a $\binom{2}{1}$-*OT* with Bob per each cell in BF. Alice's choice bit is $B[i]$, where she receives $G[i]$ if and only if $B[i] = 1$, and a random string otherwise. Let G' be the GBF received by Alice after performing the OTs. Finally, Alice can check for each $x \in X$ if $x = \oplus_i G'[H_i(x)]$, which implies that x is in the intersection.

Functionality 7: Oblivious Pseudo-Random Function

Alice's private input: a secret value x.
Bob's private input: a keyed PRF $F_k(\cdot)$, with a secret key k.
Alice's private output: the evaluation $y = F_k(x)$.

Another important approach for PSI is the OPRF paradigm, where the functionality description of OPRF is introduced in Functionality 7. This is very related to pure OT-based PSI for two main reasons: it is very similar in spirit, and OPRF can be instantiated from OT, as it happens in the work of Hemenway Falk et al. [125], where one-time OPRF instantiated from ROT is used as the core building block. OPRF-based PSI can be trivially solved from a smaller problem named Private Set Inclusion, where Alice's set X contains a single element x and she wants to test whether x is in Bob's set $Y = \{y_1, y_2, ..., y_n\}$ or not. After executing the protocol, Alice learns $F_k(x)$ and nothing else. Bob does not learn x, but since he knows k, he can compute $F_k(Y) = \{F_k(y_1), F_k(y_2), ..., F_k(y_n)\}$ locally and send it to Alice. The final step is for Alice verifying if $F_k(x) \in F_k(Y)$, which would imply that $x \in Y$. It is easy to realize that PSI can be achieved by executing this protocol n independent times, each one with a different $x_i \in X$, which is Alice's set. A high-level description of Hemenway Falk et al.'s protocol is shown in Figure 3.6, because

UNIVERSIDAD DE MÁLAGA

it helps to illustrate the approach. In the solution, authors propose for Alice to use a Cuckoo Hash Table with stash, where each bin can only contain one element, and those that cannot be inserted in free bins are inserted in the stash. Figure 3.6 shows that each one-time OPRF is independent from the others and can be easily parallelized.

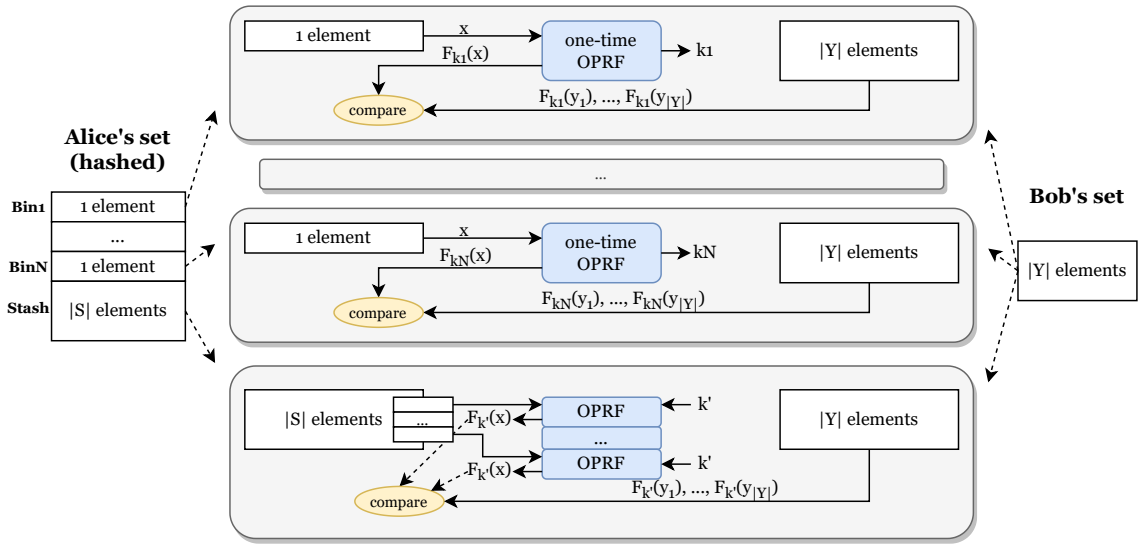


Figure 3.6: OPRF-based PSI from *Hemenway Falk et al.*

Another line of works solve PSI directly from OT, but they are somehow similar in spirit to the OPRF approach, as stated before. One of the core starting works for this line of research is PSZ [126], which was generalized by Rindal and Rosulek [127] using an ideal functionality named *oblivious encoding*. It also solves PSI from Private Set Inclusion, and the latter problem is solved from Private Equality Test, where Alice and Bob own just one element each. We now introduce the general idea using OT for the sake of completeness. Let's assume that Alice and Bob values of $\lambda-$bit lengths are $x = x_1x_2...x_\lambda$ and $y = y_1y_2...y_\lambda$ respectively. Then, Bob builds a random matrix M of λ rows, where row i contains the random elements $m_{i,0}$ and $m_{i,1}$. Then, per each row, Alice and Bob engage in an OT, where Alice's choice is x_i and Bobs inputs are $(m_{i,0}, m_{i,1})$, from where Alice receives m_{i,x_i}. After completing the λ OTs, Bob locally computes $\Delta_B = \oplus_{i=1}^{\lambda} m_{i,y_i}$ and sends it to Alice. It just remain for Alice to locally compute $\Delta_A = \oplus_{i=1}^{\lambda} m_{i,x_i}$ and check whether $\Delta_A = \Delta_B$ or not. Note that they will be equal only if Alice's and Bob's values are exactly the same. As it happened in the OPRF approach, n Private Equality Tests imply Private Set Membership (PSM), and n of the latter imply PSI.

This paradigm to solve PSI has $O(n^2)$ communication complexity. Some works introduced hashing techniques to reduce the numbers of comparison to be done, thus reducing communication to $O(nlogn)$. This has become an standard phase in OT-based PSI, and requires a preliminary phase where elements in the sets are mapped to bins inside hash tables.

Finally, remark that there have been some works improving the approach in PSZ by introducing more efficient techniques for precomputed OT and hashing to bins [128, 129, 130].

UNIVERSIDAD DE MÁLAGA

The OKVS generalization

Most recent works in PSI have proposed a new approach based on Oblivious Key-Value Store (OKVS) and OT-extension, where the former is a generalization of some of the previous approaches to obtain random mappings of the set elements. Following the definition from Garimella et al. [131], a Key Value Store (KVS) is composed of a set of keys K, a set of values V, and a set of functions H, and two algorithms $(\mathsf{Encode}, \mathsf{Decode})$, such that:

$$\begin{aligned} \mathsf{Encode}_H(\{k_i, v_i\}) &\rightarrow S \\ \mathsf{Decode}_H(S, k) &\rightarrow v \end{aligned} \tag{3.1}$$

Correctness in a KVS follows when for all $A \subseteq K \times V$, and for all $(k, v) \in A$, $\mathsf{Encode}_H(A) \rightarrow S \implies \mathsf{Decode}_H(S, k) = v$.

An OKVS is a KVS that must fulfill that, for any two sets $A_0 = \{(k_1^0, v_1^0), ..., (k_n^0, v_n^0)\}$ and $A_1 = \{(k_1^1, v_1^1), ..., (k_n^1, v_n^1)\}$ with randomly sampled values v_i, the encodings $S_0 \leftarrow \mathsf{Encode}(A_0)$ and $S_1 \leftarrow \mathsf{Encode}(A_1)$ are computationally indistinguishable.

There exist different types of OKVS, but those with linear decoding algorithms are the interesting ones for cryptographic applications. In fact, polynonimals can be used as a special type of linear OKVS, where the algorithm for decoding is interpolation. In addition, GBFs are binary OKVS, which are a specific type of linear OKVS. Some linear OKVS such as polynomials or random matrices are quadratic or cubic in the size of the input sets. On their part, new constructions such as PaXoS [132], or the 3 Hash-Garbled Cuckoo Table proposed in Garimella et al. [131], achieve linear time.

The work in Rindal and Schoppmann [133] follows the approach in PaXoS to build an OPRF protocol from Vector Oblivious Linear Evaluation (VOLE), and then uses the OPRF strategy to build PSI as it was previously introduced. Roughly speaking, a VOLE protocol allows Alice to learn Δ, B and Bob to learn A, C, where $C = A\Delta + B$. This work sacrifices computation performance compared to PaXoS and Garimella et al.'s works, but achieves better performance in communication. Finally, Raghuraman and Rindal [134] proposes optimizations for VOLE and OKVS, achieving the most efficient PSI protocol to date.

Other approaches

Finally, for the sake of completeness, we introduce other strategies that, while not as adopted as the previous ones, have served as the basis for some PSI proposals.

First, some protocols achieve PSI using Commutative Encryption (CE) as a building block, as in Cao et al. [135]. In summary, a CE scheme has the following property:

$$Enc_{k_A}(Enc_{k_B}(m)) = Enc_{k_B}(Enc_{k_A}(m)) \tag{3.2}$$

The naive idea for PSI is that Alice can learn the set $\{Enc_{k_B}(x_i)\}$, with $x_i \in X$, thanks to sending her encrypted set $\{Enc_{k_A}(x_i)\}$ to Bob. Then, Bob computes two sets, one with Alice's encrypted set, i.e., $\{Enc_{k_B}(Enc_{k_A}(x_i))\}$, and the other with

UNIVERSIDAD DE MÁLAGA

Bob's set Y, i.e., $\{Enc_{k_B}(y_i)\}$, and sends both to Alice. Since Alice can delete her encryption layer from $Enc_{k_B}(Enc_{k_A}(m))$, she ends with both X and Y encrypted with Bob's key, therefore obtaining the elements inside the intersection.

Other works are based on pairings to achieve PSI. The work of Wen et al. [136] proposes a protocol for Authorized-PSI using Identity-Based Encryption built with pairings. On the other hand, the work of Cerulli et al. [137] achieves Reactive-PSI (the output of PSI depends on previous instances) using bilinear accumulators.

3.3.3.3 PSI-CA and C-PSI

Apart from the standard PSI, its variant PSI-CA is probably the one that has attracted more attention of researchers, mainly because it is useful in some use cases and also because it is quite easy to achieve it from general constructions for PSI. PSI-CA outputs the cardinality of the intersection, but not the intersection itself. A general approach to achieve it is by applying a one-way mapping to the elements of the sets such that, when receiving that mapping in a permuted order, one of the parties can count how many items exist in both sets, without learning the actual items they are. A straightforward solution for this strategy is presented in Lv et al. [138] by using commutative encryption. First, the receiver encrypts his set under his public key and sends $Enc_{pk_R}(Y)$ to the sender, who applies his own key to the encrypted set and sends to the receiver a shuffled version of it using a random permutation π, i.e., $\pi(Enc_{pk_S}(Enc_{pk_R}(Y)))$. From this, the receiver can obtain $\pi(Enc_{pk_S}(Y))$. In the final step, the sender sends his own encrypted set with a different permutation ρ, i.e., $\rho(Enc_{pk_S}(X))$, which allows the receiver to count the number of coincidences and output the cardinality of the intersection. PSI-CA is somehow old, but a generalization of it appeared later, which is C-PSI. This captures that not only the cardinality, but any functionality, can be applied on the intersection result, and the result provided as the output of the protocol. C-PSI is not as straightforward as other variants, but can be trivially achieved by general purpose MPC.

First, we focus on the works that target PSI-CA. In standard PSI, there was somehow a balance between the works that solved PSI with PK/HE and with OT. But for PSI-CA, HE is the main selection. We only found one work for PSI-CA based on OT [139], and more specifically on $\binom{n}{1}$-OT maskings. The rest of the works rely on some sort of HE, where some of them select PHE [140, 141] and other FHE [142, 143]. Regarding the encoding of the data sets, BF is the main tool, where bits can be encrypted and permuted in a way that enables to count the number of coincidences. These works not only provide solutions to PSI-CA, but some variants. For example, Ion et al. [141] solves PSI-Sum-CA, where each item in one of the sets has an associated integer, and the protocol output the sum of these integers for the items in the intersection. On the other hand, Kacsmar et al. [142] proposes PSI-CA with differential privacy in the output. This basically minimizes the amount of information that the party receiving the intersection (or cardinality) gets from the other party set. Finally, Debnath et al. [143] proposes a solution for Size-Hiding PSI-CA, where the additional property is that no party learn the size of the other party's data set.

Second, regarding C-PSI, it is normally composed of two phases: one to compute

UNIVERSIDAD DE MÁLAGA

the intersection, but in a somewhat encrypted format that does not allow to learn its actual values, and another phase to compute the circuit on it. For example, the work of Ciampi and Orlandi [144] proposes an OT-based PSI, where set elements are encoded in an undirected graph that can be obliviously navigated to execute instances of PSM. The produced output can be in one of three special "encrypted" formats, each of them can be consumed by a generic purpose MPC approach: either GC, SS, or HE. On the other hand, the work of Pinkas et al. [145] follows the PSZ approach [126] to encode data sets in hash tables, but then rely on generic circuits for bins comparisons, such that additional operations can be computed on the results.

3.3.3.4 MP-PSI

MP-PSI is the extension of PSI to more than two parties. This variant has not received so much attention than the others, mainly because of two facts: a lot of PSI use cases work in the 2-party setting (mainly client-server model), and MP-PSI can be solved sometimes from 2P-PSI. The main limitation for MP-PSI is that communication rounds can lead to prohibitively costs and delays if there are many parties or the sets are quite large.

A key aspect in MP-PSI is the topology used by the parties. In standard general purpose MPC is quite common to design protocols in a balanced way, where parties communicate through a fully-meshed network topology and spend the same amount of work. However, MP-PSI solutions are mainly based on two different topologies: the chain/ring network topology, where parties perform some work (equal amount) sequentially, and the star network topology, where there is one party that assumes the role of leader, communicates with the other parties, and perform more work. It is important to note that in both of them, since there is no threshold concept, taking down one party means performing an effective DoS to the whole computation.

Regarding the chain topology, the work of Ramezanian et al. [146] relies on it. It uses a vector set encoding approach, where each bit of the vector represents an element of the universe set $\mathbb{U}$. It relies on PHE and threshold decryption to obtain the intersection, and it achieves an $O(\mathbb{U})$ number of encryptions, which is independent of the number of parties N. On the other hand, the work of Debnath et al. [147] uses a BF encoding and moves to a star network topology, which allow better efficiency.

Fitting within the star network group, there is a common approach for MP-PSI, which is to extend a 2P-PSI protocol by executing it between pairs (P_{leader}, P_i), as it happens in [148, 149, 150]. This approach normally implies a global cost which is an $O(N)$ multiple of the base 2P-PSI cost. In addition, these works rely on OT for an offline phase, and then on GBF for an online phase that allow to compute the intersection.

Finally, another extended approach is the Zero Sharing technique. It basically leads parties to share some data that sum up to zero if one item is in the intersection, and something different from zero if it is not. The base solution is proposed in Kolesnikov et al. [151], which works in two phases: (1) a conditional zero sharing phase, where P_i gets a share s_j^i of zero for the element x_j, and (2) a conditional reconstruction phase, where an Oblivious Programmable Pseudo-Random Function (OPPRF) is evaluated

UNIVERSIDAD DE MÁLAGA

on x_j and programmed to output shares s_j^i. Then, if all P_i programmed the OPPRF with x_j, the sum outputs zero, which means that x_j is in the intersection. Based on this work, Chandran et al. [152] improves communication, Wei et al. [153] is more efficient for the small set setting, and Nevo et al. [154] achieves more efficiency thanks to the use of OKVS.

3.3.3.5 O-PSI

Finally, the last group of PSI protocols is O-PSI, where encrypted versions of the data sets are sent to an external party, e.g. a cloud service, which is in charge of computing the intersection without gaining any knowledge about it or the data sets of the different clients. A straightforward implication of sending all data to a single server is that all the approaches to compute the intersection rely on some sort of encryption. This happens in contrast to other protocols where the intersection is computed by the data owners themselves (as introduced in the previous sections), where interactive approaches such as OT can be used in a more natural way.

One of the most straightforward solutions is to encode datasets by OPE, and then using HE to compute the intersection [155, 108, 156]. In the end, once the server has computed the intersection inside the encrypted space, one of the clients receives the encrypted intersection and decrypts it. While Yang et al. [155] proposes O-PSI, Li et al.'s work [108] proposes an O-PSI-CA solution, and Ruan et al.'s [156] an O-PSI that is extensible to MP-PSI.

Another straightforward solution for the outsourced setting is using BF encoding with bit-by-bit encryptions, which allows the server to compute the intersection using AHE. Following this strategy, Wang et al. [118] proposes an O-PSI for two parties, Bay et al. [157] an O-MP-PSI, and Adavoudi Jolfaei et al. [158] an O-MP-PSI-CA.

There are other works that assume a malicious server, and therefore introduce some sort of verifiability. This is called Verifiable Delegated-PSI (VD-PSI) [159, 160]. The main limitation of Abadi et al. [159] is that it needs the clients to agree on a secret β which is sent to the server in a way that this value must appear in the intersection result, otherwise cheating is detected. However, Terada et al. [160] avoids the need of secure channels and a prior negotiation between the clients.

Other protocols such as Oliaiy et al. [161] uses standard DLP asymmetric encryption where pairs $(Enc(x_i), g^{x_i})$ are uploaded to the server. The exponential encoding allows comparisons to obtain the intersection, and the result is retrieved encrypted. Finally, some works rely on pairings, such as Wang et al. [162], which relies on re-encryption to transform encryptions of different sets to the same key, thus comparisons can be done, and Ali et al. [163], which proposes an Attribute-Based PSI protocol with fine-grained access control.

3.3.4 Discussion and conclusions

The previous sections have shown the wide range of PSI solution that exist, and that each of them presents characteristics that suit better one use case or another. Among the aspects to consider when selecting a PSI solution, one can precise the amount of resources available in the devices that will execute PSI, the interface that

UNIVERSIDAD DE MÁLAGA

the implementation exposes to the users when interacting with the PSI instance, or the ability of the protocol to be optimized in a way that benefits the specific use case. Prior to concluding this whole section dedicated to PSI with the answers to the research questions presented at the beginning, we slightly discuss about how the different protocols require more or less amount of resources, such as computation, communication, or memory, and how they support different techniques to improve performance.

First, regarding computation, public key encryption schemes can be considerably heavy if the number of items in the intersection is large, overall in constrained devices such as IoT or smart phones. This can be even heaver when FHE is used instead of PHE schemes. While encoding to hash structures first can reduce the number of comparisons (i.e., communication), the length of the structure can be somewhat large if the error probability is set to be small, e.g., when using BF or CF. More specifically, this length will be $c \cdot n$ bits, where $c > 1$ is a real value obtained from the target overload and error probability, and n is the number of items to be inserted in the filter, which is the set length. This implies that, despite linear with n, the number of encryptions to be done is larger than n. Regarding pairings, they are also heavy in computation, where the cost of computing one pairing can be approximated to be between 4 and 8 exponentiations in the group. The benefit of using pairings is that they allow more complex variations of PSI such as Authorized-PSI. Finally, for OT, OPRF, and OKVS-based PSI, computation is not a real problem, since OTe allows to perform a large set of OT instances mainly with symmetric key cryptography, which is much more efficient.

Regarding communication, the intuition says that the scenario works in a reverse way. Indeed, communication in the OPRF setting is linear with the larger set. However, this is very dependent on the specific scheme used. For example, if public key schemes are instantiated with EC and each point has a 510-bit length, it takes 63.75 MiB to send 2^{20} elements. However, for OKVS-based PSI such as Raghuraman and Rindal [134], communication in the unbalanced scenario for two sets of 2^{10} and 2^{20} elements, respectively, takes just around 16.37 MiB. Even for the GBF approach, which is not as efficient as the OPRF one, it takes 64 MiB for 2^{20} elements when using a security parameter $\lambda = 128$. Where public key encryption schemes beat OT-based schemes is in the number of rounds.

Another aspect where public key schemes beat OT-based PSI is in memory usage. For encryptions, it is not needed that the set is loaded entirely in memory, and elements can be taken one by one. However, OT-based schemes are more interactive and require to load the whole OT matrix, or filter structure for arbitrary checks.

On the other hand, regarding improvements, a strategy used by some protocols is parallelization. This is key for public key schemes, where a great example is using SIMD techniques in FHE. Pure OTe and OPRF computations can be parallelized too, but the main issue is when encoding in dynamic allocation structures such as CF, which cannot be parallelized if it needs to perform arbitrary checks. Another strategy to improve performance is precomputation, which helps schemes based on OTe or general purpose MPC. However, precomputation has more sense when the inputs for the computation are not present yet, while in PSI the parties are normally expected to have their sets ready when they engage in the PSI instance. Finally,

UNIVERSIDAD DE MÁLAGA

Table 3.2: Running Time (in seconds) for different PSI protocols and set sizes.

Ref.	Type	Setting	Asymm. Set Size	Set Size						
				2^{10}	2^{12}	2^{14}	2^{16}	2^{18}	2^{20}	2^{24}
[120]	(PSI) HE, BF	n.s.	-	44.68	175.79	702.39	2834.68	11327.78	-	-
[121]	(PSI) FHE	WAN-1th	2^{12}	-	-	-	2.1	-	9.3	106.2
		WAN-64th		-	-	-	1	-	3.1	15.30
[142]	(DiPSI-CA) LHE	n.s.	2^{12}	-	89.50	-	530.70	-	11751.90	-
[141]	(PSI-Sum-CA) DH+Paill.	LAN	-	-	3.17	-	48.93	-	776.46	-
[164]	(PSI) OPRF	LAN	-	-	0.21	-	0.39	-	3.78	58.57
		WAN	-	-	0.59	-	1.26	-	7.46	106.83
[165]	(PSI) ROT, Hash	LAN	2^{12}	-	-	-	0.80	-	2.96	33.94
		WAN		-	-	-	2.28	-	5.43	53.33
[132]	(PSI) OKVS	LAN	-	-	0.12	-	0.25	-	5.6	-
		WAN	-	-	0.64	-	1.8	-	18.62	-
[134]	(PSI) OKVS	LAN	-	-	-	-	0.062	-	0.439	8.055
[148]	(MP-PSI) GBF	4P	-	-	2.95	-	40.91	-	513.02	-
[152]	(MP-PSI) OPPRF	4P	-	-	1.9	-	7	-	69.6	-
		10P	-	-	3	-	10.4	-	153.9	-
[154]	(MP-PSI) OKVS	4P	-	-	0.36	-	1.83	-	18.48	-
		10P	-	-	1.78	-	8.42	-	124.26	-
[163]	(O-PSI)* Pairings	n.s.	2^{10}	10.28	-	-	-	-	20.72	-
[166]	(O-PSI)* PRF	n.s.	-	0.2	0.8	3.3	13.5	57.2	226.7	-

the last strategy for improvement is outsourcing the computation, i.e., using O-PSI. However, while this avoid the clients to perform the computation themselves, the datasets have to be encrypted and sent to the server computing the intersection, which can be quite consuming with respect to communication if the set is large, as introduced above.

Finally, Table 3.2 shows some running times for protocols that rely on different assumptions. Regarding notation, *n.s.* means not specified, *xth* means that x threads have been used, and xP means that x parties are involved in the protocol execution. Also, if the asymmetric set size is not specified, that means that the protocol works in the balanced setting. It can be easily seen that public key-based protocols are much slower than OT/OPRF/OKVS ones, with the exception of Chen et al. [121] that benefits a lot from parallelization. However, it should be noticed that the setting it uses for simulations has a very powerful equipment (18-core processor and 256GB of RAM), and therefore the running times obtained should be considered only for outsourced scenarios, but not for PSI with relatively constrained devices. On the other hand, Raghuraman and Rindal [134] is the fastest protocol, but its running times are in the LAN setting. However, despite being a bit slower, other OT/OPRF/OKVS-based PSI protocols still provide acceptable running times even in the WAN setting. Regarding MP-PSI, it can be easily seen that increments in the number of parties really affect running times. Finally, for O-PSI, if the application is not very time dependent, slower protocols can be tolerated because the computation does not need users to be involved. A remark is that for the two O-PSI protocols

UNIVERSIDAD DE MÁLAGA

shown in Table 3.2 only the amount of time needed for the intersection computation phase in the server side is provided, but the overall protocols involve more phases related to sharing the obfuscated datasets, authorizing the intersection computation and retrieving the results.

A remark that must be done is that we have omitted great part of the analysis with respect to malicious settings. Since some protocols are semi-honest, i.e., they are functional but not that much realistic for real implementations, their running times may be expected to be slightly slower when implemented with maliciously secure mechanisms. We refer to the full publication for more details about what mechanisms allow malicious security in PSI and how the affect performance.

From all this analysis, we can conclude that, in general, OKVS-based PSI are the most efficient ones. However, they require a somewhat stable network connection because they need more rounds of communication. If this is not the case, or the users want to avoid the cost of computing the intersection themselves, public key based PSI and mostly O-PSI is a better approach. One could think in O-PSI that the client can keep his set encrypted in the cloud all time, and just delegate the computation of the intersection when required. Finally, we have strong evidence to answer the research questions introduced at the beginning of this section, where the answers are the following:

> **(RQ1).** The most used building blocks are public key encryption, mostly HE, and OT-based approaches, such as pure OT, OPRF, or OKVS. Others such as commutative encryption or pairings are also used, but present more limitations.
>
> **(RQ2).** PSI has been mostly covered from a theoretical point of view. Despite many use cases have been proposed, there are not many implementations or systems that offer PSI services available.
>
> **(RQ3).** There are many use cases that have been proposed for PSI. These have been covered in Section 3.3.2.
>
> **(RQ4).** The major challenge is reducing the effort needed for PSI with large set sizes. This can be achieved with highly detailed engineering tricks and powerful equipment, as shown by Chen et al. [121] with respect to running time, but is not realistic for somewhat constrained devices. Also, new approaches for instancing PSI more easily for the users is key, since PSI instances executed entirely by clients with large set sizes normally imply prohibitive costs.

As a final conclusion, we can affirm that PSI has been vastly improved since its conception, mainly in terms of efficiency. After the analysis, we have certain evidence that PSI is ready for real implementations and applications where set sizes are not extremely large, with a practical limit around 2^{20} set lengths. Even when this is acceptable enough for many applications, others that imply larger databases may require a larger effort for PSI. In addition, this is much more remarkable when doing MP-PSI, where efficient running times require to limit the number of parties to no more than three or four. Apart from improving PSI from the cryptographic side, where the most recent approaches based on OKVS have clearly set the line to follow, we envision that engineering techniques for a better performance of implementations, such as parallelization and batching techniques, must play a central role in achieving

UNIVERSIDAD DE MÁLAGA

more efficient PSI in settings where databases are large enough.

3.4 Summary

This chapter has been devoted to analyze different characteristics about MPC that are not always considered in theoretical works. We have first discussed about the assumptions that are sometimes made when new protocols are proposed, and how they do not capture many details related with real deployments. We have analyzed the different ways that exist to instantiate MPC protocols, mainly regarding the roles that the different parties assume and the implications in the concept of trust. A clear conclusion is that achieving real deployments that fulfill some of the assumptions made regarding the trust models, mainly about decentralization, is not an easy task. In addition, we have taken into account a set of the main MPC protocols that present particularities with respect to the general purpose cases: Decentralized Random Number Generators, Threshold Cryptography, and Private Set Operations. We have roughly introduced different solutions to solve these problems, and we have discussed about the importance of these protocols as basic building blocks for larger systems. Finally, we have presented a detailed review about Private Set Intersection, which is one of the most researched special-purpose MPC protocols. The strategy used has been a Systematic Literature Review with the focus very close to the practicality of the existing solutions in real world applications.

UNIVERSIDAD DE MÁLAGA

Chapter 4

Approaches for MPC in the blockchain setting

This chapter is devoted to the relations between MPC and blockchain, technologies that present similarities in some aspects and differences in others. First, we analyze which are the relations that exist between them, taking into account their similarities, but most of all the benefits that each of them can offer to the other when combined in the same system. Next, we delve into the approaches or strategies that exist to combine both technologies, considering that the spectrum of building blocks is wide and it leads to different trade-offs. We also emphasize the challenges that arise when designing a concrete system that puts them together. Finally, we propose two solutions to implement interactive MPC protocols inside blockchain ecosystems, such that users can benefit from confidential computing on their data without making them public. The first solution provides an extension to Hyperledger Besu, which is an Ethereum client, that embeds specific MPC protocols and let a subset of clients to jointly execute the MPC themselves. Besu is selected because it already supports a rough notion of private transactions. The second solution provides a framework for delegated and publicly verifiable confidential computations in smart contract-based blockchains, where functionalities are flexibly specified in smart contracts by users and transactions are executed by MPC committees.

4.1 Relations between MPC and blockchain

Despite MPC and blockchain could be seen as technologies that share a common setting, they pursue different (and even opposite) objectives, at least in general. The common part is that they both work on distributed scenarios with (partially or totally) distrustful parties, i.e., they target decentralization, and rely on cryptography to address the lack of trust. However, while blockchain targets traceability and immutability by having all data publicly available, MPC targets private computations that do not reveal anything about the inputs. These two notions are clearly opposite in their basic forms, but in technology, as it happens in society, middle-ground approaches can boost sometimes the adoption of new solutions that allow new use cases which are more realistic. As well as their decentralized nature, there is another similarity between blockchain and (some variations of) MPC, which is

UNIVERSIDAD DE MÁLAGA

verifiability. Blockchain achieves public verifiability by default, by letting everyone see all data in the ledger and re-compute whichever change a transaction has done in the state. On its part, most modern MPC approaches target verifiability such that parties can be ensured at the end of the protocol that it has been correctly computed with respect to the protocol specification and that the inputs that have been used are those of the involved input providers. In addition, this can be even enhanced to achieve not only verification by the MPC parties, but public verifiability, where anyone can verify correctness of the computation without learning anything about the inputs provided.

Several posts have covered the synergies between MPC and blockchain [167, 168, 169, 170]. Following the approach in the series of posts by Jesper Buus Nielsen [167], we gather the benefits of MPC for blockchain and vice-versa in two lists:

Benefits of using MPC for blockchain

- Protecting signing keys using threshold cryptography (see Section 3.2.2) such that they never reside in a single computer and thus are more difficult to be stolen by an attacker. MPC-based signatures are a good alternative to multi-signature protocols, which are already used in several blockchains, mainly because of three aspects. First, they are protocol agnostic, because the signatures generated are identically to those generated if the key would not be distributed. Second, they are more efficient because they produce a single signature, instead of a set of signatures. Third, they offer more privacy because the public key of the signing group may not expose each individual in it.
- Achieving full privacy/confidentiality in blockchain by computing the state of the smart contracts inside the MPC such that nothing computed by them is revealed unless it is desired. Inputs can be provided in plaintext or encrypted, and outputs can be revealed and posted in the public ledger if desired, while each computation associated to a transaction processing may be required to post only the receipt on-chain. Private or confidential smart contracts can lead to different applications [170] such as preventing front-runnings, performing fraud analysis without revealing individual transaction details, enabling private and publicly verifiable voting systems, etc.
- Securing some specific blockchain components, e.g., by providing MPC-based randomness generators (see Section 3.2.1) or by enabling atomic operations between different blockchains, which has been mainly achieved by using threshold cryptography for asset locking.

Benefits of using blockchain for MPC

- Selecting independent MPC nodes such that the threshold assumptions for security are easier to remain, instead of letting the MPC service provider to maintain all the nodes itself. This is directly related with the notion of trust introduced in Section 3.1.
- Orchestrating several aspects of MPC execution that are not trivial such as nodes connections establishments, setting logic for handling inputs, handling crashes, etc. One highlighted aspect is that blockchain solves the totally-ordered broadcast problem, which also appears inside MPC protocols. This

UNIVERSIDAD DE MÁLAGA

also relates to the fairness problem in MPC output, where an outcome published in the blockchain allows all (authorized) receivers to obtain it simultaneously. In addition, blockchain can provide the environment for recorded and timestamped traces of MPC, enabling the transparency of confidential computations executions.

- Making easier to transfer the ownership of computation outcomes and tracking the requirements for this transfer of ownership, e.g., by specific assets required in dispute resolution smart contracts.
- Providing an ecosystem for MPC solutions, e.g., by letting different MPC instances be related, handling confidential states inside smart contracts or maintaining a reputation system for MPC nodes.

4.2 Approaches and challenges

Despite it has been shown that MPC and blockchain present some synergies and can benefit from each other, the reality is that combining them is not trivial at all. Both of them are complex solutions, including advanced cryptography and different levels of technological problems, from peer identification to communication through the network or synchronization problems. In this section, we clarify some of the main challenges that arise when trying to combine these technologies, and the diversity of trade-offs that may arise depending on which specific solutions are used.

First, computations in blockchain are on-chain, which means that every node (with access to the involved data) can verify the result by computing the result again and comparing it with the result provided on-chain. In fact, this work is done by block proposers (miners, verifiers...) before proposing a block. The reason is simple: proposing a block with an invalid transaction means loosing time (PoW), loosing the stake (PoS), or being banned from the network (PoA). This is the core of an incentive-based solution such as blockchain. However, in MPC, there does not exist this approach of computing again to verify. Consider a blockchain transaction that contains the result of an MPC instance. This can only be proposed by an MPC committee after executing the protocol with the private inputs and verifying it is correct. However, the verifiers of the network (which must be different from block proposers) will not be able to compute the result again because they have no access to the private inputs. Therefore, MPC cannot be processed entirely on-chain.

However, this is a half-truth. Section 2.1 shows that the MPC ecosystem is large, and that there are different technologies that enable solutions to compute on private inputs. It is very common to relate MPC with interactive protocols such as those based on GCs or SS, but this is understanding MPC as a concrete technology. If we understand MPC as a general problem, FHE solves it too (with a few limitations). And FHE is, in fact, the technology that enables MPC to be done (almost) entirely on-chain. The main example is the work of Zama [62], which employs FHE to let any node in the blockchain executing transactions without revealing their contents. However, FHE has a major problem with this model of general computation because it does not work well with multiple receivers. Since the computation must be done inside the ciphertext space generated by a key pair (pk, sk), the one that owns the secret key can reveal anything (inputs, computation, and outputs). This is the

UNIVERSIDAD DE MÁLAGA

reason that they let the key pair to be owned by the network, using Threshold-FHE as a solution. That means that a special committee in the network hold shares of the secret key and uses threshold decryption to provide decryption of the results only to those users allowed. However, this implies a security reduction from FHE to SS-based MPC. In addition, it must be noticed that FHE is quite costly in terms of computation, which may lead to expensive monetary costs for confidential on-chain computations.

There is an approach that allows interactive MPC to be computed entirely on-chain, which is computing the operations of the shares inside the encrypted space using FHE. However, this approach does not have much sense since FHE can compute the circuit entirely by itself. For that reason, interactive MPC cannot be used on-chain. However, despite this can be seen as a problem, it can also present some benefits. First, from a general point of view, it implies a reduction in the computational cost. On-chain operations are executed multiple times, whenever a node wants to verify the result. On the other hand, an MPC operation will only be executed once by the corresponding committee. What this implies is that the cost to compute the operations will be independent from the network costs, which may be prohibitive overall in Layer 1 blockchains. Another benefit can be related to performance, but this is very dependent on the scenario and the capabilities of the peers involved, since FHE and interactive MPC present very different requirements: in general, FHE is computationally intensive, and MPC is network intensive. There are works, such as those introduced in Section 2.1.3.4, that propose new models that split the computation in slices and where each slice can be computed by a different committee. Some of them point out to straightforward applications to blockchain. Following the approach introduced in Benhamouda et al. [50] for SS, if the result of each computation slice is moved from one committee to another by using the blockchain as a Public Bulletin Board, then the MPC can be seen closer to the on-chain scenario, at least by having some notion of verifiable check-points.

Despite all these benefits that interactive MPC can introduce with respect to the FHE approach, the main challenge that remains is how can the network verify the results. A naive way is to rely on the committee, i.e., use a threshold such big that verification is not needed at all. However, this can be a very relaxed assumption. In the approach for on-chain checkpoints introduced before, even if the protocol is maliciously secure and each handover presents verifiable material, this is done with respect to the committees involved, and therefore would also imply trust in the threshold assumption. However, there are a few works that target this notion of publicly verifiable MPC, and we have found solutions that fit well with the proposals of this thesis, mainly based on ZKPs. We will elaborate more on this in Section 4.4.

Another challenge for MPC in blockchain comes from the atomic nature of transactions, which implies implementation difficulties for interactive solutions. Transactions are atomic because they are executed at once, as a single batch of operations. When a transaction is included inside a new block, the result is already known. However, if a transaction implies an MPC process, it requires a set of parties to interact between them to obtain the result. This leads to two design approaches: (1) propose a transaction only after the MPC result is computed by a committee, and (2) propose a transaction and compute the result later. The first approach may work in solutions where all related with smart contracts is assumed to be computed

UNIVERSIDAD DE MÁLAGA

using MPC, but it presents problems when public and private operations coexist. The reason for this is that MPC usually takes much longer than local computation on public values, and therefore it may happen that it takes quite large delays for MPC transactions to be included. In fact, they can be denied at all, because updates in the state must be sequential, and therefore public transactions can block the option for a private transaction to be included in the state. Another limitation is that the committee is computing a transaction before knowing if the transaction is going to be included or not. If the user must pay in advance, the fee can be lost without the transaction being included. If the fee is only payed after inclusion, then the MPC committee can loose execution time without any reward. In Section 4.4 we will elaborate more on the "add first and compute later" approach.

Finally, there must be noticed that private inputs for an MPC are not held by all the nodes of the blockchain. Since each share is only held by a single member of the MPC committee, data availability gets reduced. It may happen that the share is lost, or that the node is not available and the MPC cannot be computed at all. In fact, the risk to suffer a DoS is stronger here than in a public blockchain setting, where making DoS to a blockchain node provider can be solved by the user requesting to another provider. There may be different approaches to mitigate the DoS surface in MPC, from using redundant committees, encryption of shares stored on-chain, or using (k, n)-SS strategies for MPC, where the committee is formed by a sufficiently large set of n members and any subset of $k < n$ members is allowed to compute the function. However, all these solutions imply additional resources.

4.3 MPC in Hyperledger Besu

In this section, we present a new design that allows to embed MPC capabilities inside a Hyperledger Besu[1] blockchain. The main objective is to address the lack of privacy problem in blockchain, as introduced in Section 2.2.2, and more specifically to allow applications selecting in a fine grained way which data to consume in a privacy-preserving manner. There are various reasons to select Besu as a target solution to embed MPC. First, it belongs to the Hyperledger Foundation, which is a widely known and active ecosystem for decentralized solutions. Second, Hyperledger Besu is an Ethereum client, and this is important for programmability, thanks to smart contracts. In addition, Besu can be connected to the public Ethereum blockchain or it can be deployed within a private blockchain. And third, it already includes some capabilities for privacy, even if it does not achieves a very fine grained approach. Because of all these, we consider Besu as an interesting solution where to embed MPC following the design goal of minimizing the amount of modifications made to the standard way in which it works and how users interact with private applications. This is interesting because it implies an easier adoption of our solution, which can be done by a relatively simple update of the client's software. A remark is that, while Besu's solution for privacy achieves protection with respect to which users perform which actions, our target is confidentiality of private data, as it is addressed by MPC.

This design is based on a *governed data model*, which means that private data

[1] `https://besu.hyperledger.org/`

UNIVERSIDAD DE MÁLAGA

resides at the blockchain node's level. Using the general model for MPC introduced in Section 3.1, the design in this section assumes that the blockchain nodes are both the input parties I and the computing parties C. However, a blockchain node can handle inputs from multiple users. Since it sees the inputs as plaintexts, the node becomes a trusted third party for the users. Since this may seem problematic from a trust point of view, is fits with the philosophy of a private/consortium blockchain like Besu, where each node belongs to a different organization. All users interacting with the blockchain through their organization node are assumed to trust it, and the access control for the different private inputs are assumed to be handled by the node itself. This is the unique way to solve the MPC problem in blockchain without requiring the users to become computing parties themselves, or without delegating the whole input to external parties.

4.3.1 Private transactions in Besu

Since one of the main reasons to select Besu's client to embed MPC is that it already supports some notion of privacy (although it does so very coarse-grained), this section is devoted to explain how this privacy works.

Roughly speaking, privacy in Besu means that a transaction is only visible to the involved participants, therefore it is kept private with respect to the rest of the blockchain nodes, which cannot access the transaction's contents nor the list of participants. When a private transaction is sent to the blockchain, its execution (in terms of contents) is similar to if it was a standard public transaction, with the exception that the public view does not learn anything about the transaction. This allows to create subgroups in the network with access to contents that nobody else can access. It is important to notice that this affects the way that the blockchain state is managed, because private contents cannot be included in the global public state. In this case, the set of nodes involved in a private transaction (or a set of private transactions) must keep a separated state which works in the same way as the public one, but that is only accessible by them. We formalize this notion in Definition 7.

Definition 7. *First consider the blockchain network* $\mathbf{B}$ *as a set of nodes, a node* N *and a subgroup of nodes* $\mathbf{P}$*, called* ***privacy group****, with* $N \in \mathbf{P}$ *and* $\mathbf{P} \subseteq \mathbf{B}$*. Then,* $S_N^{\mathbf{P}}$ *denotes the* ***private state****, or sub-state, of* N*. A* ***private transaction*** $tx_{\mathbf{P}}$ *for a privacy group* $\mathbf{P}$ *is a transaction that does not change the global state of the blockchain, but the private state* $S_N^{\mathbf{P}}$ *for all* $N \in \mathbf{P}$*. Finally, the* ***extended state*** *of a node* N *is defined as* $S_{ext} = S_{pub} \cup (\bigcup_{N \in \mathbf{P}} S_N^{\mathbf{P}})$*, where* S_{pub} *is the public global state.*

As it can be understood from Definition 7, this model of private transactions implies that there is not an unified view of the blockchain, beyond that the public state. Each node locally maintains separated states, one for the public view and one for each private state associated with a private group where this node belongs, which can be more than one. This is depicted in Figure 4.1 for more clarification, where transactions that contribute to state XY are only accessible to nodes X and Y.

Regarding how the different states can or cannot interact between them, there are three simple rules in Besu. The following list specifies what actions are allowed for

UNIVERSIDAD DE MÁLAGA

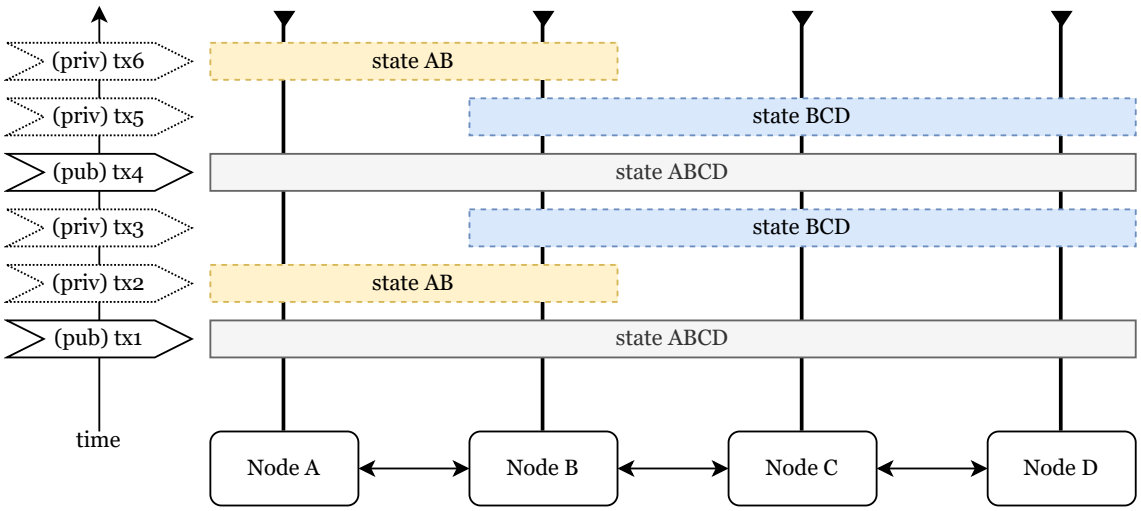


Figure 4.1: Representation of public and private states in a Besu blockchain.

a smart contract that belongs to a privacy group:

1. It can read or write to a contract in the same privacy group.
2. It can read from the public state (accounts' data and contracts).
3. It cannot access contracts from a different privacy group.

From an implementation point of view, Besu can handle these subgroups in two ways. First, assume that each node has a public address $addr_N$ that identifies it uniquely in the blockchain. The first approach to specify the privacy group of a transaction is to use two new fields (with respect to a standard public transaction in Ethereum): *privateFrom*, which includes the address of the sender $addr_S$, and *privateFor*, which includes the list of addresses of the receivers $\{addr_{R_1}, ..., addr_{R_k}\}$. Then, the privacy group is defined as the subset of nodes with addresses $\{addr_S, addr_{R_1}, ..., addr_{R_k}\} \subseteq \mathbf{B}$. The second approach allows the involved participants to create firstly the privacy group on-chain, and obtain a unique identifier $id_{\mathbf{P}}$ for the specified subgroup of nodes. Then, the private transaction will carry a field named *privacyGroupId* with the id provided by the blockchain.

The following paragraphs are to explain how Besu manages private transaction delivery. For that, each Besu node relies on a special *private transaction manager*, called Tessera. When a private transaction reaches a Besu node, it is sent to the associated Tessera node, which encrypts the transaction and delivers it to each of the Tessera nodes marked as destinations in the private transaction (specified by the privacy group), i.e., the delivery process uses point-to-point encryption. That means that the blockchain ends with two layers from a topological point of view: the blockchain mesh, formed by Besu nodes, and the privacy mesh, formed by Tessera nodes, with a one-to-one correspondence between all Tessera nodes and all Besu nodes that support privacy (refer to Figure 4.2 for a graphical visualization where a private transaction is sent from node A to node B). Note that this correspondence is logical and that Tessera can be run on the same physical machine as Besu. The communication between Besu and Tessera uses a REST interface that works in a pull mode, i.e., Besu requests data or actions to Tessera, but Tessera cannot push

UNIVERSIDAD DE MÁLAGA

data to Besu without receiving a request first. This is key to understand the protocol to notify new incoming private transactions to the rest of the network, but prior to that, we introduce a new concept in Definition 8 which is a specification of Definition 7.

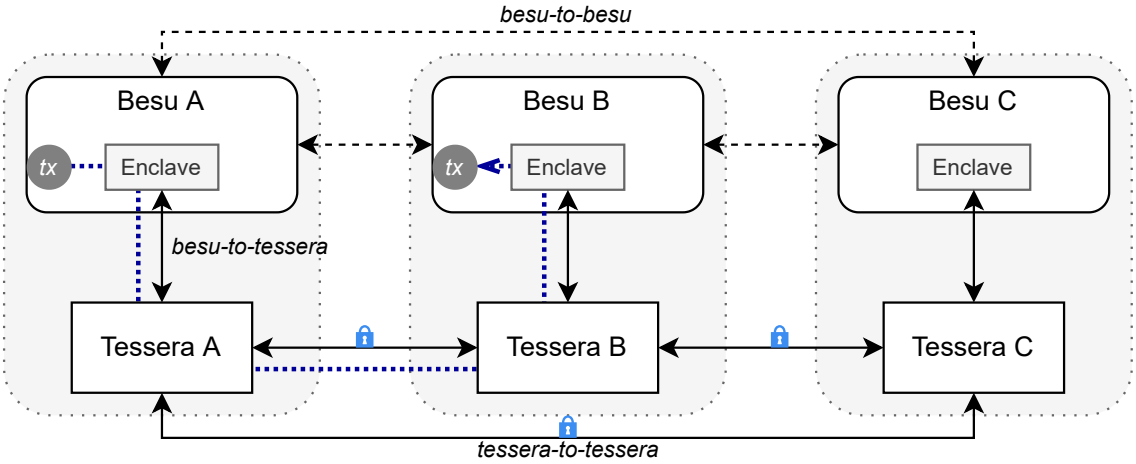


Figure 4.2: Network topology of Besu and Tessera nodes.

Definition 8. *A private transaction with **public receipt** is one where its execution leads to a change in the public state, specifically with a binding ID that does not expose anything about the actual data in the private transaction. Ideally, this ID has the committing property, i.e., the owner of the private transaction can reveal its data at some point in the future to verify that the binding ID actually belongs to that transaction.*

The change in the public state is done in Besu by the creation of a public transaction, named *Privacy Marker Transaction* (PMT), that serves as a reference point to check the reception of new private transactions. More specifically, when a new private transaction reaches Besu, one of its components called *privacy controller* sends to Tessera an encrypted version of the transaction using the *enclave*, which is an isolated component that manages cryptographic functionalities. On reception, Tessera replies with the hash of the encrypted private transaction, which is added to the body of the new PMT and publicly broadcasted through the network (see Figure 4.3). Notice that, without knowledge of the actual ciphertext containing the private transaction, the hash does not expose anything about its content. The broadcast allows the PMT to reach all Besu nodes in the network, and whenever one node receives it, it checks if its associated Tessera node has received a private transaction with this hash, in order to include it to the privacy group state or just to ignore such PMT (see Figure 4.4). As stated before, the API between Besu and Tessera is pull, i.e., that Tessera does not notify Besu from receiving a new private transaction, but all Besu nodes query their Tessera nodes using the anchor point (PMT) from the public state.

The last concept related with how Besu handles private transactions is the *privacy precompiled contract*, which is related with the execution part. A precompiled contract is a set of instructions that are locally executed in the node and represents a frequently called functionality which is heavy to be written in opcode and executed on-chain. Since private transactions do not follow the same flow of execution than

UNIVERSIDAD DE MÁLAGA

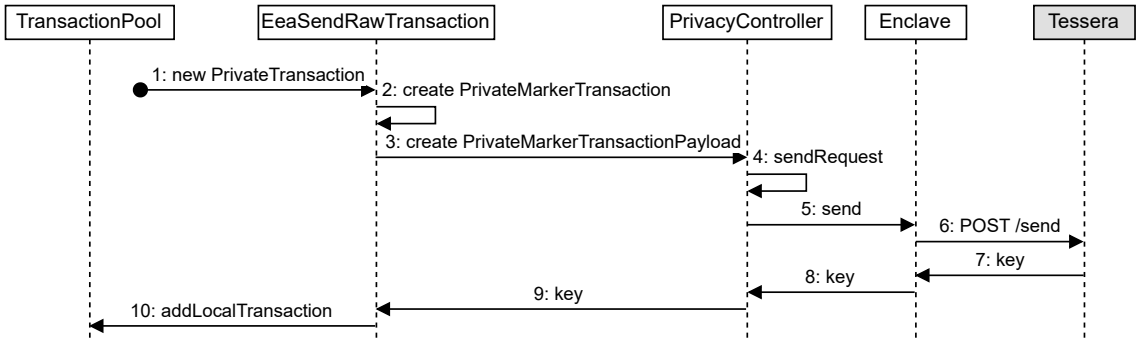


Figure 4.3: Sequence diagram: create PMT.

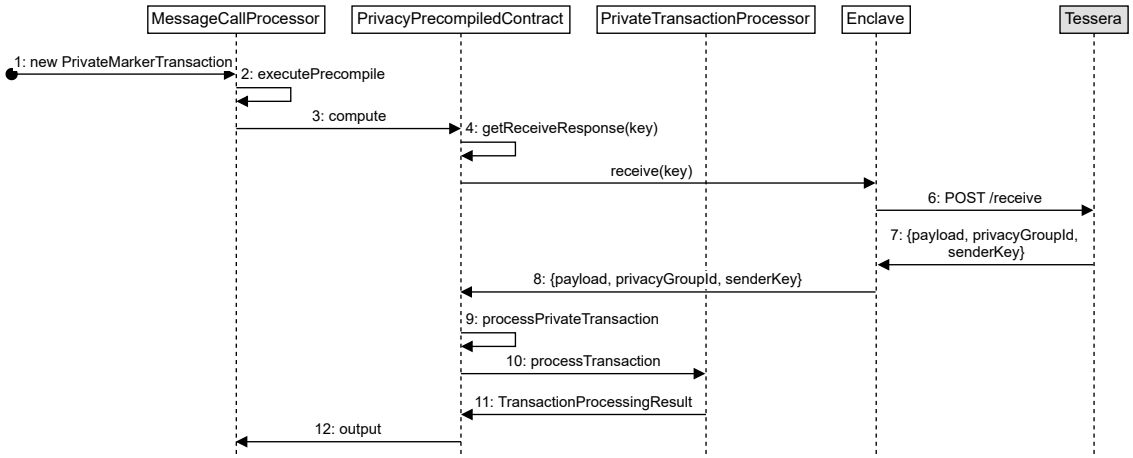


Figure 4.4: Sequence diagram: retrieve private transaction.

public transactions, when the payload is retrieved by the enclave and decrypted, it is provided to the privacy precompiled contract, which starts the execution logic. However, the actual execution, i.e., the emulation of the Ethereum Virtual Machine (EVM) is done by the private transaction processor, as shown in Figure 4.4. Finally, the result of the transaction is returned and stored in the private state that corresponds to the privacy group where the transaction belongs.

4.3.2 MPC extension in Besu

In this section, we introduce a design solution to let Besu clients handling MPC executions natively from their code and the smart contract specification, without the need of external libraries. As previously introduced, the main purpose of this work is to reuse a blockchain solution that already contained some sort of privacy and enhance it to allow more advanced privacy-preserving computations. However, one of the main design principles is to minimize the impact of the solution in the code, in a way that it can be viewed as an easy to implement software update that does not have a major impact in the standard way of interacting with applications deployed in Besu. Notice that this notion of interaction has two main surfaces: the interaction between the user/application with the blockchain node, which is done by using a web3 library, and the interaction between the different blockchain nodes, which follow the specified protocols for handling and executing public or private transactions.

UNIVERSIDAD DE MÁLAGA

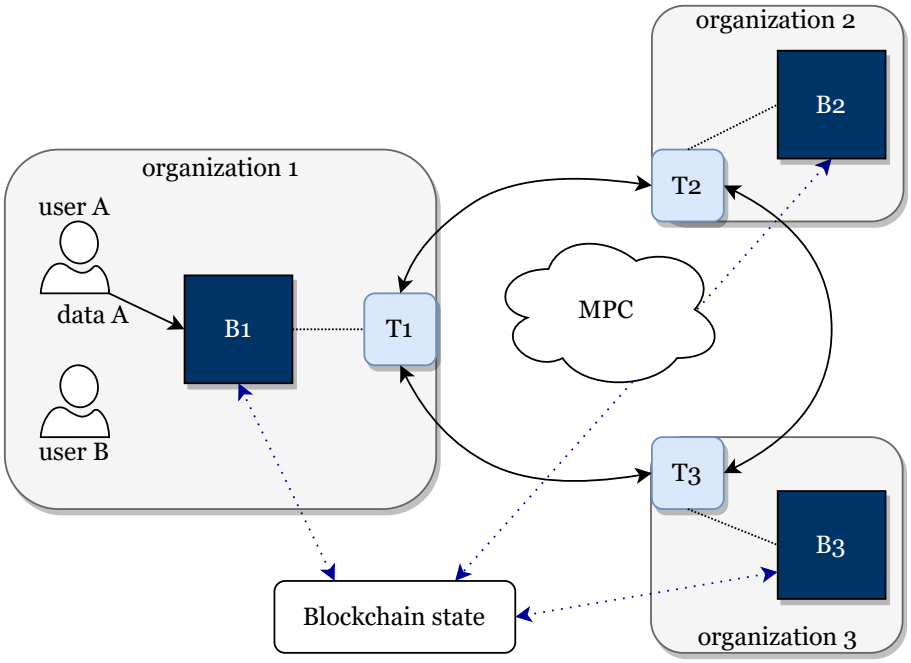


Figure 4.5: Overview of the design for MPC in blockchain using Besu clients.

Before introducing the specific details of the design, an overview of the solution is presented, together with the MPC model and trust relations it follows. Figure 4.5 shows the general architecture, where B_i and T_i are Besu and Tessera nodes, respectively. First, regarding how to include MPC in the workflow of Besu, it is clear that the easiest way is using private transactions as the wagon for the contents, i.e., function selection and data used. The reason is because there is already a full logic for handling computations that only involves a subset of nodes in the blockchain. Therefore, this can be reused for an MPC instance, where the private group is the set of the peers that interact in the MPC. Even when an MPC allows the description of the functionality to be public, implementing that with public transactions would require larger modifications in the code and the way of processing the transactions, since they do not present interaction with Tessera or the privacy precompiled contract. Second, this feature about the way of solving the MPC problem implies that the blockchain node is the one in charge of handling the transaction, and therefore the associated data (either private or public) and the instructions specified by the contracts. This is also reasonable because leading the clients to carry out the computation would imply a large overload for them, with no difference as if they were using an external and independent application for the MPC part. Then, the model used according to that introduced in Section 3.1 is $I = C = O$, which means that the blockchain node is either the input provider, the computing party, and the output receiver, with respect to the MPC. This model presents some implications in trust relations between the different parties:

1. Blockchain nodes do not trust each other and rely of MPC assumptions when computing the functionality. Each node handles its own private data and do not share it in plaintext form with the rest of the nodes in the blockchain.

2. Users that interact with the blockchain to execute an MPC trust the Besu node they are interacting with, because they must share their private data in

UNIVERSIDAD DE MÁLAGA

plaintext form with the node. This model follows the consortium philosophy where an organization with several users deploy one node (or more) in the blockchain, and they trust their own node. The node is in charge of providing additional mechanisms of protection inside the organization, such as access control, to avoid different users accessing other users' private data.

Finally, regarding the way that MPC is embedded and interacted, this design does not allow the flexibility to let each user specifying its own MPC functionality, because the smart contracts do not present fine grained operations that can be mapped at deployment time to operations in the cryptographic library. Instead, this design allows the blockchain to expose a set of useful (and already specified) protocols, which can be seen as different classes of a cryptographic library. This allow the users to call these utilities directly from the blockchain application.

4.3.2.1 Asymmetric state

The main problem to embed MPC in a blockchain application is how to relate the state with the private inputs. In Besu's standard private transactions, all the privacy group sees the same private state, but if an MPC is related to a privacy group, i.e., all the nodes in the privacy group are participants involved in an MPC, the view of the protocol for each node will be different, thus they cannot reach a common state. Conceptually, what this solution needs to handle is an asymmetry in the private state stored by a privacy group, which is introduced in Definition 9.

Definition 9. *A privacy group* $\mathbf{P}$ *is* ***symmetric*** *if for all nodes* $N, N' \in \mathbf{P}$, $S^{\mathbf{P}}_{N'} = S^{\mathbf{P}}_{N}$. *Otherwise, we say that the privacy group is* ***asymmetric****.*

What is happening is a split of the state in three levels of granularity: (1) what is accessible by every node in the blockchain, (2) what is accessible only by a subgroup of the blockchain, and (3) what is accessible only by a node in the blockchain. To solve the asymmetry problem, data which is accessible only by a node (e.g., MPC inputs) cannot be included in the state. The dummy solution is to handle private data in another component in the node and replace private values in the transaction with dummy data, e.g., a set of zeroes. Another option could be to include commitments of the private data, such that it enables a future verification of the data used in an MPC, as it happens with the hash value inside a PMT for standard private transactions.

Then, a Besu node will maintain three different state storage: one for the public state S_{pub}, another one for each symmetric private state $S^{\mathbf{P}}_{N}$, and a third one for the private values owned by N which are included in a state $S^{\mathbf{P}}_{N}$ through an MPC instance. Notice that this will require some logic inside the Besu node to modify the transaction from the first version sent by the application through the web3 library, which includes the private inputs, to a second version that does not include them but can be delivered to every node involved in the MPC.

4.3.2.2 Extended private transactions

To handle MPC, we follow the same approach used by Besu to handle private transactions, which is adding new fields to standard Ethereum transactions. Besu adds three fields, which are:

UNIVERSIDAD DE MÁLAGA

- *privateFrom*: the Tessera public key of the transaction sender.
- *privacyGroupId*: the id of the privacy group to receive the transaction.
- *restriction*: to determine which nodes in the network receive the encrypted payload.

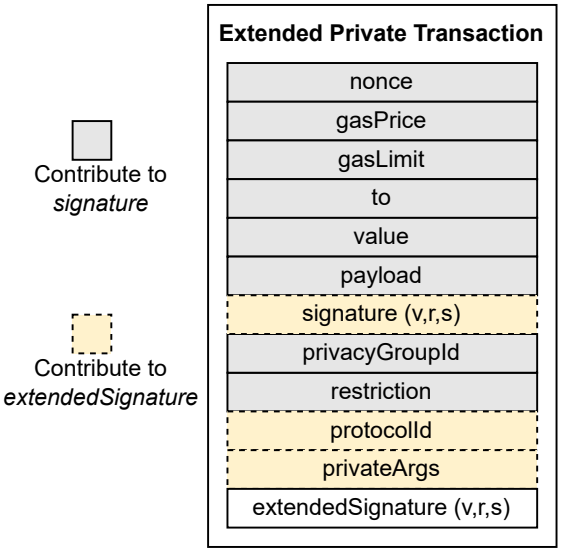


Figure 4.6: Fields of an extended private transaction.

The transactions used in our extension for MPC are called *extended private transactions*, and we also add three fields (see Figure 4.6). First, a field *protocolId*, which identifies the specific type (or family) of MPC protocol to be executed. This value allows all the Besu nodes involved to select the same MPC protocol description and follow the corresponding steps, either if all the parties involved perform the same steps, or if the protocol is somewhat asymmetric and different parties play different roles in the whole process. Next, as previously explained, the private inputs cannot reach every Besu node involved in the MPC, but they must reach the Besu node of the owner from the application side. For that reason, we add a field *privateArgs*, which carries the private inputs provided by the application. Note that this field will carry the actual inputs when the transaction is sent by the application to Besu node, but there, the private inputs will be stored locally and removed from the *privateArgs* field. Before delivering the transaction to the other nodes using Tessera, this field can be set to an empty value, to zero, or can carry commitments of the private inputs. Finally, the third field added is *extendedSignature*, which ensures integrity of private data at the owner's node side. Since private data cannot reach other nodes in the blockchain, the standard signature cannot include these values. For that reason, the *extendedSignature* gathers the private inputs with the standard signature to avoid modification attacks.

4.3.2.3 Smart contract interface

Another important part of the design is related to how the user (the application) interacts with the MPC functionality. Since our contribution targets a solution with the less impact as possible to the standard way of interacting with an application in Besu, we conclude that the MPC functionality must be executed through a smart

UNIVERSIDAD DE MÁLAGA

```
contract OPRF {
    constructor(address alice, address bob) public;
    function loadKey(uint256 key) public onlyBob();
    function evalData(uint256 value) public onlyAlice()
        returns(uint256);
}
```

Listing 4.1: Solidity interface for OPRF.

```
contract BftRng {
    constructor(address[] participants) public;
    function commit(uint256 random) public;
    function getRandom() public returns(uint256);
}
```

Listing 4.2: Solidity interface for BFT-RNG.

contract. Differently from a standard smart contract which is executed inside the EVM, this is called *Interface Smart Contract* (ISC), and its purpose is to expose the MPC methods to the blockchain node in a way that the application can call them using a web3 library.

Since the ISC is an interface-description of the MPC protocol, the contract and the protocol execution (which is not done by the contract but the cryptographic module of Besu) must be intertwined at some level. There may be different approaches to design ISCs, mainly regarding on how many details about the cryptographic protocol are desired to be selected from the application. As introduced in Section 3.2, MPC has several families of specific protocols which share common characteristics, but inside each family there are variations regarding the specific instance. Therefore, the ISC can capture a family of protocols, e.g., OPRF, and then let the cryptographic module to select the specific version of OPRF to execute (this can even be selected from the application using a specific *protocolType* parameter). One can think of this as it happens in cryptographic libraries, where an encryptor object interface is common and the specific protocol is selected by using a parameter. Another approach is to implement an independent ISC for each protocol version, but this can lower usability in the application side.

We propose two examples of ISC, one for OPRF (see Listing 4.1) and one for BFT-RNG (see Listing 4.2), where each of them shows different technicalities regarding on the actual execution of the protocol. First, the ISC for the OPRF is sequential, i.e., each of its methods is executed only once: first, it is created; next, Bob loads his private key; then, Alice evaluates the OPRF with her data, producing the output. On the other hand, the ISC for BFT-RNG is an n-party protocol. The method *commit* must be called once by each of the n parties, therefore, an internal state machine must keep track of who has called that method and if the following method, i.e., *getRandom*, is already accessible or not. In this case, the final random result will only be produced once enough parties have contributed first with their independent random values.

UNIVERSIDAD DE MÁLAGA

4.3.2.4 MPC execution and message delivery

Finally, this section covers the aspects related with transaction management at the blockchain node side and execution. To facilitate understanding, Figure 4.7 shows the different functional blocks (from a high-level point of view) that are involved in the process. More specifically, the blocks filled in gray are those that have been modified to make possible the inclusion of MPC.

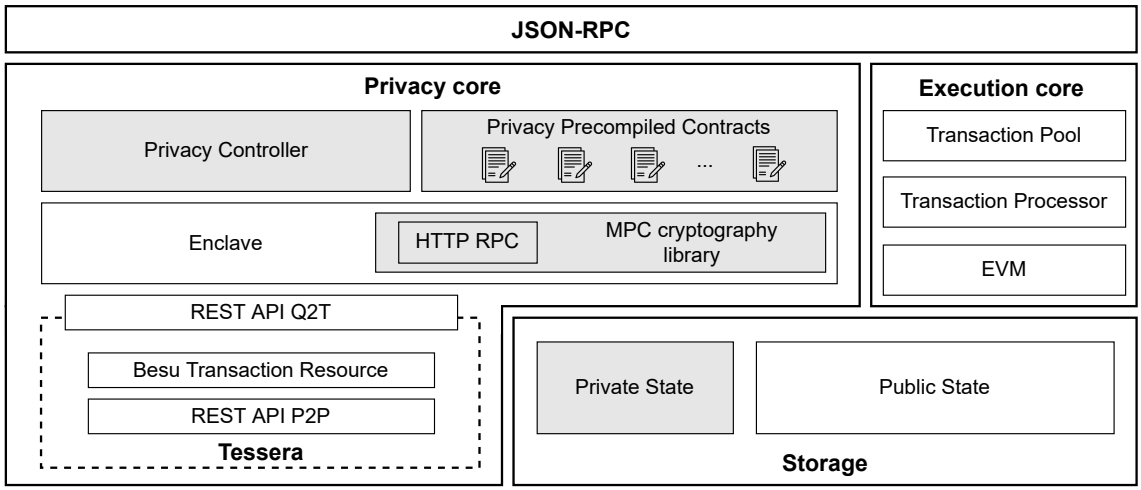


Figure 4.7: Functional blocks for Besu with MPC.

In Besu's standard private transactions, there are two main components involved in the execution process: the *privacy controller* and the *privacy precompiled contract*.

The privacy controller is the one in charge of handling new incoming transactions that arrive through the JSON-RPC interface, and querying Tessera for the generation of the PMT. In our design, we have modified the privacy controller such that it understands the private data carried by an MPC transaction based on the field *protocolID*. It generates a new transaction identical to the one received but without the private values, and this version is the one sent to Tessera for the PMT generation and also the one which is sent to the other nodes involved in the MPC. Since execution is not done yet, but it is done by a different component after the PMT has been flooded through the network, the privacy controller saves the private values of the transaction in a (key, value) private database, where the key used is the hash of the PMT.

After that, the PMT reaches all nodes in the blockchain, and those included in the privacy group retrieve the transaction, included the node that distributed it. It is then the privacy precompiled contract which is in charge of execution. In standard Besu, it delegates the execution to the execution core where the EVM is. However, in our model, the privacy precompiled contract is modified to handle each logic specific to different MPC protocols. First, the private data for the MPC execution can be retrieved from the private database using the PMT hash as the lookup value. Then, execution is coordinated inside the privacy precompiled contract, which calls the specific functionalities required in the cryptographic library, which resides inside the enclave.

Figure 4.8 shows a simplified sequence diagram of the main phases to handle an

UNIVERSIDAD DE MÁLAGA

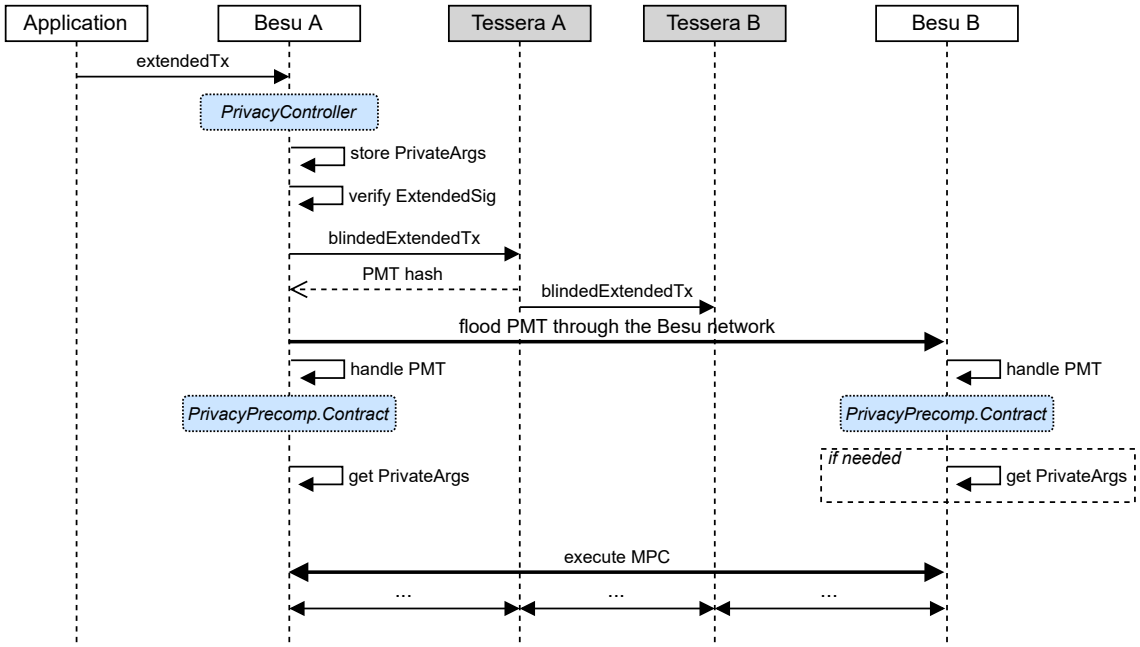


Figure 4.8: Sequence diagram: handling a extended private transaction in Besu

extended private transaction as it has been described above. First, the privacy controller handles the private data, verifies the signature and delivers through Tessera a blinded version of the private transaction without the private data. Second, the privacy precompiled contract uses the protocolId in the transaction to know how the MPC should be executed, retrieve the corresponding private data and executes it with the other nodes involved. In this figure, the MPC part has been left open because it can be implemented in different ways, mainly because of a caveat that arises in Besu when exchanging messages in an MPC execution.

An MPC execution implies a set of messages exchanged between the different parties through point-to-point secured channels. The ideal case in our design is to let Tessera be the manager of MPC messages delivery, not only transactions, i.e., each time the MPC library uses *send* or *receive* methods, an HTTP message is sent from Besu to Tessera, which delivers it to the Tessera instance in the destination. However, the way that Besu-Tessera communication is designed introduces a technical challenge which is not trivial to solve. Let's first generalize an MPC execution as a tuple of messages $((h_1, m_1), ..., (h_k, m_k))$, where each message of data m_i has an associated header h_i that identifies the message inside the protocol instance (headers are protocol specific and known to each party). As it was introduced in Section 4.3.1, when Tessera sends a transaction, it computes its hash which is included in the PMT and publicly shared through the blockchain. Later, in order to retrieve a transaction from Tessera, a POST /receive method must be sent, containing the hash as the transaction identifier. In an MPC protocol, the retrieving value for each message is the header; however Tessera automatically assigns the hash of the message m_i as the identifier, value that is unknown to the other parties (differently from h_i). To solve this, we introduce three different strategies.

a) Modify Tessera.

A special endpoint is added to Tessera such that it does not store the message hash

UNIVERSIDAD DE MÁLAGA

as the identifier of a transaction, but the header h_i, sent together with the message m_i through the HTTP API. To retrieve the message, the receiver periodically queries Tessera with the identifier h_i until it arrives. This periodic query is required because of the pull approach of communication. While this solution is somehow simple, modifying Tessera is not desired because we want the protocol to be agnostic to the specific private transaction manager used, which purpose is only the encrypted delivery of transactions.

b) Whisper protocol in Besu.

A whisper protocol in Besu would be one that allows Besu nodes to send off-chain secure messages between them. With such a protocol, it would be easy for Besu to notify through whisper the hash of the message that is sent through Tessera, therefore letting the destination to learn which transaction has to be queried to its Tessera node. Whisper protocol has never been supported by Besu, and even if it was supported by other client such as Geth, it is no longer. In addition, adding new specific communication channels between blockchain nodes may lead to new vector attacks.

c) Message Delivery Smart Contract.

Another solution is to provide the blockchain with an additional precompiled contract that supports a message delivery ISC. Each time a message m_i is sent by the MPC library (carried by a private transaction), it also sends a new private transaction using the JSON-RPC interface of the blockchain node, which deploys the message delivery ISC instance with the message hash h_i as a parameter. This transaction is a standard private transaction that is handled by the sender's Besu node, sent through Tessera, and delivered to the receiver's Besu node. When the contract is executed, it emits an event containing the message hash h_i, which is listened and consumed by the MPC privacy precompiled contract, enabling the message retrieval. Despite this solution is more expensive than the other two, it leaves a public trace (the PMT) of each message sent by the MPC protocol, improving accountability. In addition, this alternative avoids modifying Tessera, which is desirable.

4.3.3 Threat analysis and discussion

To finalize this proposal, we provide a brief analysis and discussion.

First, regarding the security threats, it should be noticed that the point of failure is the blockchain node. Since we target a private/consortium blockchain, the node is assumed to be trusted by the users belonging to that organization. Anything private held by a user that is used as an input to an MPC will reside in the blockchain node. As previously introduced, the node is assumed to provide access control for the different users in the organization. Therefore, if confidential data uploaded by Alice to be used as an input to the MPC should not be accessed by Bob (for example, if they belong to different departments), the node should restrict access to Bob when querying such data. On the other hand, regarding the MPC procedure itself, notice that the design is agnostic to which protocol is implemented. Protocols with different security levels could be integrated such that users can instantiate smart contracts with the desired security, e.g., targeting a semi-honest secure implementation or a

UNIVERSIDAD DE MÁLAGA

malicious one, depending on the type of data being computed or on which parties are involved in the protocol execution.

Second, we provide a brief discussion about the design itself. It should be noticed that embedding new features like MPC without altering the way that Besu works is highly difficult. In this design, easy integration is prioritized over efficiency. In fact, Besu's way of achieving privacy is not the most efficient one in the blockchain ecosystem. Splitting the MPC protocol in phases may introduce delays, despite it produces a higher degree of linkage to the on-chain state, thanks to the PMT. In addition, the communication mechanism for the nodes involved in the MPC is somewhat inefficient, overall when using the messages through Tessera, which would be the more native solution. Despite all these caveats, the design is functional and ready for use, and it allows users to call MPC functionalities within the same application, without external procedures. In addition, the node controls data, which is beneficial if it is really trusted by users. The reason for this is that the organization blockchain node is a party involved in the MPC, and therefore other parties (organization nodes) will not be able to reveal data or computation results as long as the user's node is needed in the access structure to reveal data.

4.4 Framework for delegated MPC in blockchain

This section is also devoted to achieve fine-grained confidentiality in blockchain, but it takes a different approach than Section 4.3. While the previous solution sets the target in Besu and provides a solution specifically designed to be embedded within its functionality, the solution in this section targets confidential computations in blockchain from a wider scope. Despite it is very inspired by how Ethereum works, it is not intended for a specific blockchain software; it just assumes the existence of smart contracts and PoS. To be more precise, what we introduce in this section is a framework to compute MPC functionalities in a public blockchain in a highly programmable and fine-grained way. The main objectives to be achieved by such a framework are the following:

1. It is cost-effective.
2. MPC functionalities are dynamically specified by users.
3. Public and private operations can coexist in a natural way inside the decentralized application.
4. There is public verification made on-chain for the MPC results.

Despite the framework is intended to be wide-scoped, in order to achieve the desired objectives in a concrete way, it is based on a set of design principles. Notice that, despite that, it is very agnostic to the specific set of protocols it can support when instantiated:

- The computation to be performed, which can include both public and private operations, is specified in smart contracts that are deployed on the network and interacted with by users.
- Both public and private computations are outsourced to the blockchain, therefore saving computational resources to users and avoiding the need to use

UNIVERSIDAD DE MÁLAGA

external off-chain applications specifically designed for the private ones.

- Interactive MPC is the tool selected for confidential computations, where execution is performed off-chain by a committee of parties and verification of the output is performed on-chain. This model enables independence between the cost of the MPC and the cost of on-chain operations. However, as we will specify later, the framework is agnostic to the MPC protocol and can be instantiated with any of them.
- MPC committees are incentivized to behave honestly, through on-chain mechanisms. Committees can take various forms, either static or dynamic, randomly selected or preselected, etc.
- The solution fits with the atomic nature of blockchain transactions, without incurring in unfairness, or wasting fees and resources.

From the prior list, it is key to emphasize on the aspect that the computation is entirely outsourced to the blockchain. This is a relevant aspect that modifies the architecture and the trust model. In the model of Section 4.3, despite the computation with private inputs is also outsourced to the blockchain, it is not outsourced with respect to the trust model, because the blockchain node is assumed to be trusted. From the point of view of trust, the whole organization fits within the same trust domain, therefore, each domain handles its own private data. However, in this model, a consortium blockchain is not assumed, but a public one. In such a blockchain, nodes are not trusted by users, and despite this, the solution presented in this section assumes the entire delegation of the computation to a subset of nodes. Notice that this implies $I \neq C$ from the point of view of the MPC model of Section 3.1.

4.4.1 Architecture and workflow

In this section, we introduce the framework proposed to achieve delegated MPC-based confidential computations in public blockchains. The framework is first introduced from a high-level point of view, with a high degree of generalization, together with the workflow between all its components. After that, the following sections will introduce specific aspects more related to implementation details which are of paramount importance. Figure 4.9 depicts the framework proposed. We now describe each of the main components it has.

Users. Individuals that want to delegate some computation to the blockchain or consume data that has been generated from previous computations. These computations may contain public data, confidential data, or both. Users must pay fees for usage, which will vary depending on the length of the computation requested and the type of data involved.

MPC nodes. A pool of nodes from where committees are formed to compute functionalities with confidential inputs. Each committee must hold the confidential data owned by users in order to perform the computation, but without accessing the plaintexts. Committees receive fees for computing the functionality. It should be noticed that larger committees may imply additional security, but also larger fees. For public verification of the result, they must provide a valid proof of correctness (see Section 4.4.1.3). If the proof does not verify correctly,

UNIVERSIDAD DE MÁLAGA

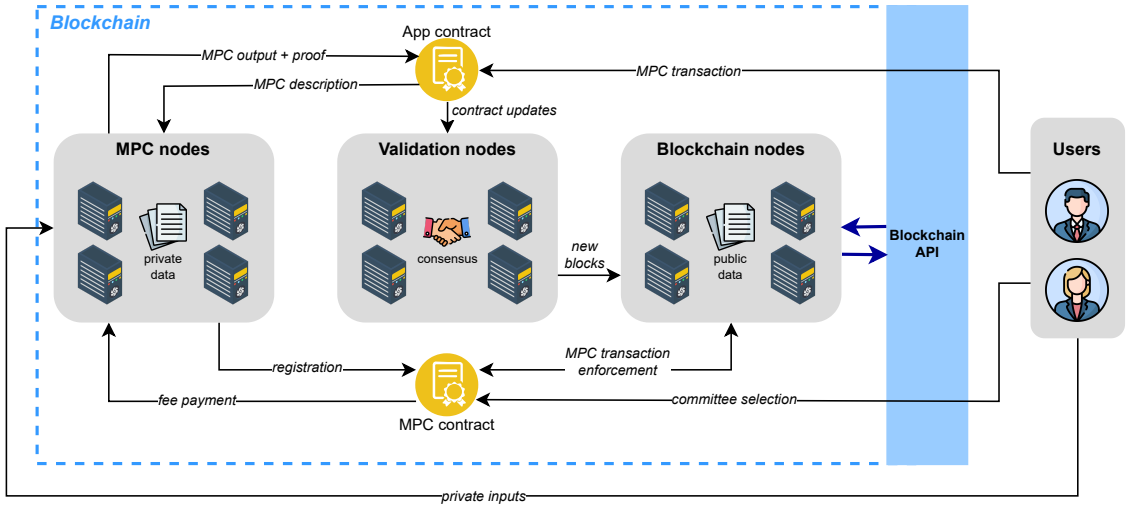


Figure 4.9: MPC-based confidential computing framework in blockchain.

the computation result will not be included on-chain and nodes will not be paid with the fee.

Validation nodes. Nodes that generate and propose new blocks. The procedure depends on the consensus mechanism, such as PoW, PoS, or PoA.

Blockchain nodes. Nodes that perform the rest of the blockchain functionalities, such as storing data and state information, distributing pending transactions and new updates to the rest of the nodes, or verifying the validity of transactions. To validate correctness of transaction results, they need to recompute the function if all data involved are public values, or to verify the proof generated by the MPC committee if confidential data are involved. A certain number of blockchain nodes identifying incorrect transactions may lead the node that proposed the block to loose its stake.

MPC SC. An SC governed by the blockchain that serves as a mechanism to keep track of the available MPC nodes and performs the locking of their stakes. It also pays the reward fees provided by the users to the MPC nodes once the computation result has been correctly stored on-chain and verified.

App SC. An SC that describes the functionalities that can be consumed by the users through contract calls, either public or confidential. Users can deploy new application contracts and the blockchain ensures the governance of data and functionalities (access control, rights delegation...).

We now introduce the workflow for the framework, which roughly describes each of the phases involved in computing MPC and the interactions between the different components of the blockchain. A sequence diagram of the workflow is presented in Figure 4.10.

1. **Registration of MPC nodes.** This phase registers the nodes willing to compute MPC in the pool of available nodes. For that, they must stake a certain amount of cryptocurrency in the MPC SC *(1a)*. Depending on how the committees are formed, nodes are directly assigned to specific committees which are publicly available *(1b)*, or they just remain in the pool for future

UNIVERSIDAD DE MÁLAGA

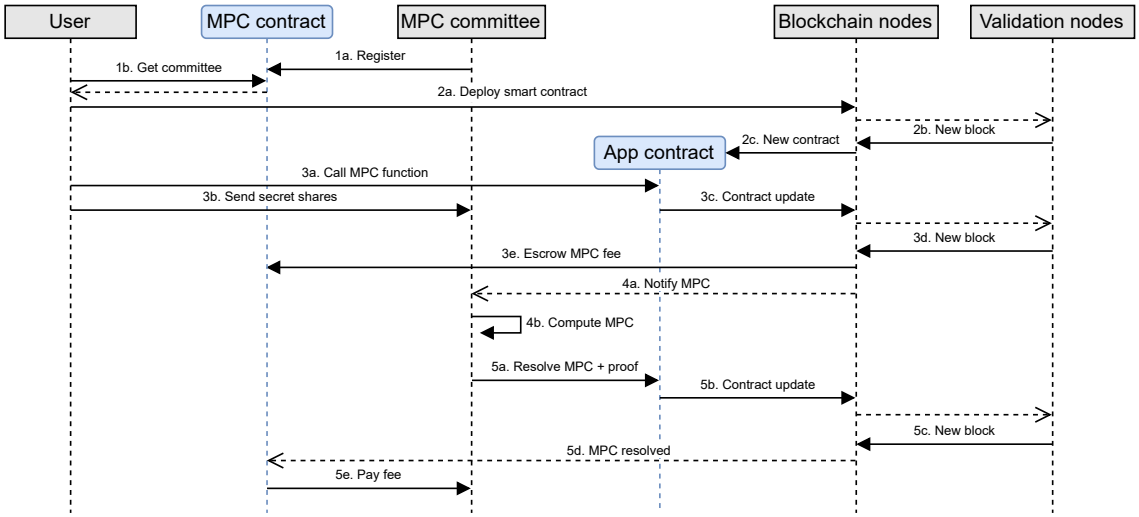


Figure 4.10: Sequence diagram of the framework's workflow.

procedures of committee selection. For more details into this aspect, we refer to Section 4.4.1.2.

2. **Deploy the App SC.** The App SC is deployed to the blockchain by a user *(2a)*. This contract works in the same way than a standard public contract, but it can include MPC functionalities. Once it is deployed *(2b-2c)*, users can interact with it.

3. **Call an MPC function.** If the SC provides MPC functions, users can call them as if they were standard transactions *(3a)*. However, there are three main differences. First, the committee to compute the MPC can be selected by users (specified in the transaction), or selected by the network. Second, the private inputs are sent to the MPC committee through off-chain channels *(3b)*. Finally, users must include additional fees in the transaction to pay the MPC nodes for the computation. Regarding the off-chain channels, they can be managed entirely off-chain (using TLS, for example), or the blockchain transaction can be used as a delivery mechanism, using public key cryptography. Regarding this problem of delivering inputs to an MPC committee, we provide in Chapter 5 a new cryptographic protocol that can facilitate data sharing in a more efficient way in different settings. Once the transaction is accepted, the App SC state is updated *(3c-3d)*, but the MPC result is not yet available. In addition, the fee for the MPC committee is locked in escrow within the MPC SC *(3e)*.

4. **MPC execution.** Once notified *(4a)*, the MPC nodes jointly compute the function described in the App SC *(4b)* if the confidential inputs have been correctly sent by users. The MPC nodes generate the output and compute a verifiable proof of correctness.

5. **MPC resolution and fee payment.** When the MPC output is generated, the committee members update the App SC state with the result *(5a-5c)*. Once the network verifies its correctness *(5d)*, the MPC committee can retrieve the fee from the MPC SC *(5e)*.

UNIVERSIDAD DE MÁLAGA

It should be remarked that the computation implies two on-chain transactions: one submitted by the user requesting the execution of the SC method that contains the confidential computation, and the other submitted by the MPC committee executing the computation. The purpose to split the execution chain in two phases with on-chain links is motivated by the atomic nature of transaction execution in blockchain. This approach for modeling the computation also allows us to bring a natural solution regarding how the application is interacted with. We will elaborate more on this in the next section.

Another detail to be noticed is that this framework, as it is specified, captures confidential computation with a single input provider very naturally. However, if more than one input provider would exist, i.e., a set of inputs $(x_1, ..., x_k)$ with $k \geq 2$ belonging to different users to be executed within the same SC method, additional capabilities should be specified. Here capabilities means some kind of coordinated storage for the private inputs, which can be the MPC committee itself. However, the execution logic must consider that the SC method cannot be executed until enough inputs have been provided to the specific committee. Despite this can be implemented somewhat easily, the main difficulty would be to manage the fees from the different users, overall if some of them do not engage in the computation and make other users to waste their fees when submitting private inputs.

4.4.1.1 Atomic embedding of asynchronous operations

As it was previously introduced in Section 4.2, interactive MPC presents implementation difficulties because of the atomic nature presented by blockchain transactions. In order to emphasize the type of problematic situations that can arise when executing MPC in blockchain, we first introduce a specific instance of the problem that leads to a race condition situation.

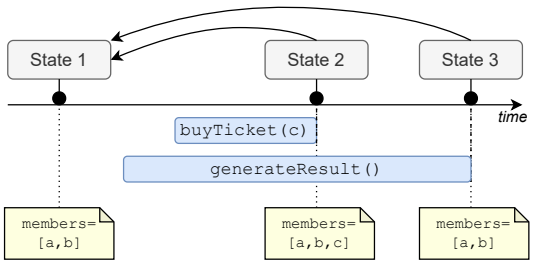
```
Contract:

  public members[];

  function public buyTicket(public address){
      members.add(address);
  }

  function MPC generateResult() {
      MPC res = lottery();
      return res.open();
  }
```

(a) An SC supporting both public and confidential data.



(b) The state fork problem presented by the SC.

Figure 4.11: Race condition issue of MPC in blockchain.

Figure 4.11 presents two figures that help understanding how the race condition is generated. First, Figure 4.11a shows a small SC for a lottery game. This contract contains an array of *members*, to keep track of the users who have bought a lottery ticket, and two functions, *buyTicket*, to register participants, and *generateResult*, to select a random winner. Notice that the second function if of MPC type, since it involves an MPC committee to generate a truly secure random number, using an approach similar that those introduced in Section 3.2.1. This is, in fact, one problem very present in public blockchains, since they do not have a truly secure randomness

UNIVERSIDAD DE MÁLAGA

generator because attackers can try to modify the state of the blockchain to bias the result of the randomness generator. In this particular example, the race condition is produced because the public function is likely to be executed faster than the MPC function, therefore leading to a fork in the state. Figure 4.11b shows how the array of members already contains two users at the beginning, *a* and *b*. The function *generateResult* is called, but while it is being processed, an unaware user *c* may call *buyTicket* to enter the lottery. The latter function will most likely finish before *generateResult* (thus updating the state) because MPC introduces larger latencies. However, user *c* will never be included in the result because the MPC started its execution with the value in state 1, unaware of the update of state 2, hence leading to an unfair scenario. This issue can even be used by malicious actors to perform DoS attacks, where they monitor the blockchain queue of pending transactions, and then call public functions in those SCs.

We notice that the race condition in Figure 4.11 could be fixed by an extra public function *lockTickets*, which freezes the array, and only after that lock is appended in the state, *generateResult* could be called. However, we remark that this framework aims a general and flexible environment where users design their specific SCs, and imposing this restrictions when programming may not be acceptable and may lead to additional vulnerabilities in SCs. For that reason, we adopt a specific solution for our framework that avoids these type of failures. Section 4.2 briefly advanced two approaches that solve this problem, from where we adopt the one for "add first and compute later" (we refer to that section for the reasons).

In order to implement this approach, we have translated to the blockchain workflow an asynchronous programming pattern which is widely used in web applications, the *promise* pattern. A promise is defined as the proxy of a value that is not necessarily known when the promise is created. It allows the system to instantiate a variable, continue the function's workflow, and return to that variable when its value is resolved. Thus, a promise is tagged with one of three possible states during its life-cycle: pending, resolved, or rejected. In our solution, a promise identifies an MPC output that has not yet been resolved. However, when the promise is instantiated, there is on-chain evidence that the MPC is being processed. For it to work inside a blockchain, two on-chain transactions are needed: the one sent by the user and the one sent by the MPC committee. In addition, a final resolution must be taken into account by the network, e.g., through the MPC SC. To be more precise, the sequence of steps would be the following:

the first transaction is sent by the user and instantiates the promise, while the second transaction is sent by the MPC committee to resolve the value obtained in the MPC. We note that a rejected state could be provided if the committee cannot resolve the promise within a certain amount of time.

1. The user calls a SC method that contains some confidential operations. This call is made through an on-chain transaction that locks the user fee inside the MPC SC. In addition, the private inputs must be sent to the MPC committee through an off-chain channel. Once the transaction is accepted on-chain, the output variable that is of type MPC is treated as a promise, and therefore it is updated to *pending*.

2. The MPC committee receives the private inputs from the user and is notified

UNIVERSIDAD DE MÁLAGA

to execute the specific method. When the output is generated, it is provided (together with a verifiable proof) to the application SC and the promise is updated to *resolved*. Note that this implies the second on-chain transaction. This change in its state can be notified to the user such that he can consume the result from the blockchain. However, it may also happen that the MPC committee does not reach to compute the result. If there is some kind of timer for this, the blockchain itself can update the promise to the state *rejected*. Then, the MPC SC can be in charge of sending back the fee to the user by taking it from the stake of the MPC committee.

4.4.1.2 Committee selection and interaction

Determining which parties belong to an MPC committee is not a trivial task because it is directly related to the notion of trust and how easy can be for an adversary to corrupt enough members from the committee. To formalize it, we introduce the notion of two groups: the *available pool of nodes* $\mathbf{P}$, which is the set of all nodes eligible for MPC tasks, and the *computing committee* $\mathbf{C} \subseteq \mathbf{P}$, which is the specific set of parties that carry on an MPC for a specific functionality. Considering that the pool of nodes exist to compute confidential transactions in order to obtain some benefit, regarding on the amount of resources available on the nodes, it should not be a requirement that $\mathbf{C_i} \cap \mathbf{C_j} = \varnothing$, i.e., one node could belong to more than one committee if desired. These committees can present different characteristics from different aspects of analysis, where normally more dynamism or security introduces more complexity on how to orchestrate this from the system, and also more delays and costs involved in obtaining the output of the computation. We identify two aspects for classification. First, regarding the node selection mechanism for committees:

Predefined committees. Committees are formed in advance and published somewhere publicly accessible, thus allowing users to check the existing committees and selecting the one that fits better with their requirements.

Randomly chosen committees. Committees are randomly chosen from the pool of available nodes at the time of starting the computation. Note that this does not imply that users cannot use some sort of deny list, where they can include nodes which are not trusted at all by them (as introduced in Section 3.1).

Note that if nodes are predefined, it would be easier for the adversary to corrupt enough nodes from a committee in order to access all confidential data processed by that committee. Random selection makes targeted attacks more difficult, since the adversary does not know which specific nodes will handle a computation. However, randomly selecting committees in a secure and non biased way is not trivial, and some solutions [50] rely on selection committees that perform some DRNG-like procedure (see Section 3.2.1) to select committees for computation. Therefore, the adversary could prefer to attack the selection committee in order to bias the result.

The second aspect to classify committees strategies is whether they are static or dynamic:

Static committee. A computation f is assigned to a committee $\mathbf{C}$, which is in charge of processing f entirely and providing the result.

UNIVERSIDAD DE MÁLAGA

Dynamic committee. A computation f is split in parts $\{f_1, ..., f_k\}$, where each f_i is assigned to a different committee $\mathbf{C}_i$ from a set $\{\mathbf{C}_1, ..., \mathbf{C}_m\}$. Note that k and m may differ or not, depending on whether the same $\mathbf{C}_i$ is allowed to compute more than one f_i or not.

Dynamic committees, such as those introduced in Section 2.1.3.4, have the advantage that the computation "changes its location" from time to time, and hence the adversary is not bounded only in corruption power, but also in the time available to corrupt enough nodes before the committee changes. These solutions are typically based on proactive secret sharing schemes. However, handover mechanisms must be provided such that $\mathbf{C}_i$ can provide its partial output y_i to $\mathbf{C}_{i+1}$ without revealing the actual result.

4.4.1.3 Public verifiability

As previously introduced, one of the main benefits of blockchain is achieving public verifiability of computations, which allows users to trust the system. However, achieving public verifiability when the computation must be done confidentially is not trivial. Achieving some publicly verifiable proof than an MPC result is correct would introduce two main benefits: (1) it allows any node in the blockchain to verify correctness of the result, even when it has not access to the inputs, and (2) it impedes a malicious committee to output an incorrect result such that the blockchain keeps it as a correct one.

This notion of public verifiability was introduced by Baum et al. [171] by the name of *publicly auditable MPC*. Their definition of *auditable correctness* (see Definition 10) captures the general idea very succinctly.

Definition 10 (Auditable correctness)**.** *Let C be a circuit with inputs $(x_1, ..., x_n)$, y a potential output of C, and τ a protocol transcript for the evaluation of C. An MPC protocol is said to satisfy auditable correctness if it holds that an external party with the role of auditor and provided the transcript τ outputs the following: (1) accept y with overwhelming probability if $C(x_1, ..., x_n) = y$, and (2) reject y (except with negligible probability) if τ is not a transcript of an evaluation of C or $C(x_1, ..., x_n) \neq y$.*

The notion of publicly auditable MPC is very strong because it allows to prove correctness for public verifiers even in the scenario where all the computing parties are malicious. Other aspects related with the deployment of the MPC such as the threshold selected, the type of adversary against the protocol remains secure, and the incentives given to the computing parties to avoid misbehavior, directly target prevention. However, auditable correctness targets the identification of cheating after it has been produced.

An important remark is that publicly auditable MPC needs commitments of the inputs. Notice that if commitments are not provided by input parties, a malicious committee can modify the input, produce an output, and generate the proof of correctness on the modified input. Therefore the auditor cannot be ensured that they used the valid inputs or not. However, when the input parties commit to their inputs, a valid proof on modified inputs cannot be produced by a malicious committee, otherwise it would imply breaking the commitment scheme. Protocol

UNIVERSIDAD DE MÁLAGA

2 abstracts these steps as they would proceed in a blockchain (or any other public bulletin board system).

Protocol 2: Committed MPC with public audit.

1. The user sends to the blockchain the function $f(\cdot)$ to be computed and a commitment on its confidential input $c = Com(x)$.
2. The user sends to the MPC committee valid secret shares of the input $[\![x]\!]$.
3. The MPC reads $f(\cdot)$ and c from the blockchain.
4. The MPC computes $y = f(x)$ using $[\![x]\!]$.
5. The MPC computes $\pi = Proof(f(\cdot), [\![x]\!], c)$.
6. Any public verifier can check whether $Verify(y, f(\cdot), c, \pi) = 1$ or not.

The protocol of Baum et al. works on SPDZ-like MPC settings and relies on SWHE and ZKP. However, it achieves proofs which have linear size, which may not be suitable for blockchain environments, since storage is expensive. However, there is an interesting paper proposed by Ozdemir et al. [172] that deals with publicly auditable MPC taking a slightly different direction. They take zk-SNARKs (see Section 2.1.2.4) as the base construction and generalize it to *collaborative zk-SNARKs*, where the witness is distributed across a set of parties such that $\mathbf{w} = (w_1, ..., w_n)$, where w_i is the secret witness held by party P_i. Then, the idea is to run the proof generation algorithm of a zk-SNARK as if it was an MPC, which takes $(w_1, ..., w_n)$ as secret inputs instead of a single w located in a single machine. It is important to notice that this produces the same output as if it was a standard zk-SNARK procedure, therefore the size of the proof π does not depend on the number of parties involved, and it can be non-interactively verified by public auditors. In addition, most zk-SNARKs produce constant size proofs, which is a great advantage for proofs that are being submitted to the blockchain.

Regarding the type of zk-SNARKs used, the authors identify that not all of them adapt so easily to the distributed setting when generating the proof. More precisely, they adapt four schemes: Groth16, Plonk, Marlin, and Fractal. From these, they claim that Groth16 and Marlin are the most efficient, Plonk implies larger communication between the provers, and Fractal increases the proof size and verification time by a factor of n when deployed in the MPC setting. In general, for Groth16, Marlin, and Plonk implementations, they state that the runtime is approximately the same in the single prover setting and when the distributed generation is secure against a malicious minority of the provers. On the other hand, for security against $n-1$ malicious provers, the slowdown is around $2\times$. This performance level is quite promising for these proofs to be implemented in real scenarios.

An interesting remark is that this model of proof generation requires that all provers participate, i.e., the whole MPC committee, otherwise the proof is not generated. Note that in our framework, an invalid generation of the proof, or not being generated at all, means tagging the promise as rejected and the MPC committee members loose their stake. Then, it is important to guarantee a dispute resolution procedure that allows honest members in the committee to point out those that prevented the proof of being generated. Despite this issue, Ozdemir et al. claims it as a bene-

UNIVERSIDAD DE MÁLAGA

fit to use MPC-with-aborts protocols, which are somehow simpler than those with guaranteed output delivery.

4.4.1.4 Implementation details

This section addresses some details about the procedure to instantiate the framework in a blockchain. For that, we identify two main characteristics to consider, which are the coordination layer and the computation layer.

With coordination layer we refer to the procedure that allows identifying, registering and/or selecting the MPC nodes, i.e., those that can participate in the computation. Our framework assumes that this is done though the MPC SC, which can be ideally embedded in the blockchain protocol to work in a similar way than mining rewards. However, we note that the MPC SC can also be deployed at the application layer such that it instantiates a *Layer 2* solution, similar to scaling blockchains. The main difference is that in this framework the Layer 2 computes on confidential data instead of public data, but verifiable proofs can be provided by ZKP as it happens in zk-Rollups[2]. The most difficult part to implement is the escrow mechanism for the MPC fees, which are locked inside the MPC SC and payed to the MPC committee only once it has finished the computation correctly. In a setting where the MPC SC manager is trusted, this is trivially implemented by the smart contract, but to minimize the trust assumptions, a typical mechanism would imply a set of non-colluding managers that employ threshold signatures (refer to Section 3.2.2.2).

Regarding the computation layer, i.e., the MPC committees and the actual MPC protocol, we remark again that the framework is agnostic to the MPC protocol selected, despite the protocol selection can condition the way that the committees are selected and how the MPC nodes interact between them. Therefore, the framework can be instantiated with any of the protocols specified in Section 2.1.3, including those which are dynamic. However, selecting between protocols with static or dynamic committees is a key aspect that relates to the coordination layer, regarding if committees can be selected by the users or if there must exist some on-chain mechanism to randomly select MPC nodes per computation. Another aspect that is affected by the protocol selection is the type of inputs that the user must provide to the MPC nodes, which can be standard secret shares for some of them, or maybe the keys to a Garbled Circuit instance. Finally, when we refer to a framework which is agnostic to the specific MPC protocol, we also consider FHE-based MPC protocols. As it was introduced in Section 2.1.2.5, MPC for multiple input providers can be achieved by Threshold or Multikey FHE, and therefore in this framework the MPC committee could comprise an FHE evaluation and an interactive MPC protocol for decryption using either a Threshold or Multikey instance. In such an implementation, the users would just need to provide encrypted inputs with the MPC committee public key.

In the next section, a brief evaluation will be provided by assuming MPC committees instantiated with the SS-based general purpose MPC protocol of SPDZ.

[2] `https://ethereum.org/es/developers/docs/scaling/zk-rollups/`

UNIVERSIDAD DE MÁLAGA

4.4.2 Theoretical evaluation

To evaluate the proposed solution, we provide a cost comparison between the standard public case (STD) where the blockchain computes on public data and two confidential approaches: our solution, which uses SS, and Zama's work called fhEVM [62], which is based on FHE (see Section 2.2.2 for more details). For a fair relative comparison, we analyze the cost of storing a single UINT32 value and computing an addition and a multiplication.

Regarding the scheme instantiation, fhEVM uses TFHE (based on the LWE problem) and authors provide a cost analysis. Regarding SS-based MPC, we choose SPDZ, which is somewhat standard for maliciously secure MPC with authenticated shares (we refer to Section 2.1.3.3 for a short description). To compare these schemes on an equal basis, we choose the parameters for the schemes such that they have 128 bits of security. Increasing the bits would imply increasing the parameters used and the cost.

First, we briefly describe some details about TFHE and SPDZ. In TFHE, a ciphertext is an $(n+1)$-tuple of elements $(a_1, ..., a_n, b)$. To achieve 128-bit security, the authors of fhEVM choose $n = 2048$, and each element in the tuple is 64 bits long. Adding two ciphertexts is done component-wise, so it takes $n+1$ additions. As for multiplication, one of the ciphertexts must be encoded in a special ciphertext, which is a matrix of $(n+1) \times l$ LWE ciphertexts. Multiplication using such a ciphertext requires $l(n+1)^2$ products, where l is an instance-specific small integer.

On the other hand, in SPDZ, an authenticated share of the secret s is a triple of shares $(\llbracket s \rrbracket, \llbracket \alpha \rrbracket, \llbracket \alpha \cdot s \rrbracket)$, where α is the authentication parameter. To achieve 128-bit security, each share of $\llbracket x \rrbracket$ is 128 bits long. Addition between two secret shares requires each MPC node to add them locally, component-wise, so it takes 3 local additions per node. As for multiplication, it requires additional pre-computation to support the local operations required. Specifically, it requires $2N + 5$ additions and 2 multiplications in each MPC node, where N indicates the number of parties involved.

Table 4.1: On-chain storage cost for a UINT32 value.

Approach	Data stored	Mandatory	Algorithm	Cost (B)
STD	Plaintext	Yes	-	4
SS (ours)	Data pointer	Yes	Keccak256	32
	ZKP for computation correctness	No	Plonk ($\pi + vk$)	1,184
	Commitment	No	EC-Pedersen	64
FHE	ZKP for input correctness	Yes	Plonk ($\pi + vk$)	1,184
	Ciphertext	Yes	TFHE	(packed) 16,600 (unpacked) 263,000

Table 4.1 compares the on-chain amount of data required to store a single UINT32. The FHE approach requires a ZKP to verify the data ownership of each ciphertext; otherwise, malicious users could steal unowned ciphertexts from the network and upload them again to request decryption. In our SS-based solution, the ZKP is optional, used only for public verification of the computation performed by the MPC committee (using the multi-prover solution of Ozdemir et al. [172]). To

UNIVERSIDAD DE MÁLAGA

instantiate the ZKP, we assume the Plonk scheme, which supports proofs on variable functionalities (this is needed to accept arbitrary functions described in the SC). Note that both the proof π and the verification key vk need to be uploaded to the blockchain. In addition, SS also requires a Pedersen commitment of the user's data to verify the correctness of the inputs in the proof of computation. Table 4.1 shows that FHE has a much more significant storage requirement than SS. In addition, in SS, the main bottleneck is the ZKP (optional), while the data pointer is small and constant (independent of the actual length of the shares).

Table 4.2: Computational cost to add and multiply UINT32.

Operation	**Scheme**	**Modulus**	**Cost (general)**	**Cost (specific)**
Add	STD	32 bits	$Add32$	$Add32$
	SS	128 bits	$3 \cdot Add128$	$12 \cdot Add32$
	FHE	64 bits	$(n+1) \cdot Add64$	$4,096 \cdot Add32$
Mul	STD	32 bits	$Mul32$	$16 \cdot Add32$
	SS	128 bits	$(2N+5) \cdot Add128 + 2 \cdot Mul128$	$(8N+532) \cdot Add32$
	FHE	64 bits	$l(n+1)^2 \cdot Mul64$	$268,697,664l \cdot Add32$

After analyzing storage, we also consider the computational cost. Table 4.2 compares the number of operations required to perform additions and multiplications. We use the addition of two 32-bit elements $Add32$ as the base unit for comparison. Then, we use some rough generalizations: an $Add64$ operation is assumed to cost (on average) the same as $2 \cdot Add32$. On the other hand, one $Mul32$ will cost approximately $\frac{32}{2} \cdot Add32$, using a binary "shift-and-multiply" algorithm. Table 4.2 shows that FHE is much more computationally expensive than SS. In fact, TFHE in Zama's paper [62] is instantiated with a plaintext space of 2 bits, which means that additional operations must be taken into account to add two 32-bit integers. A multiplication in FHE (assuming $l = 1$) costs the same as a multiplication in SS with 33,587,142 nodes, but for SS setups, it is usually sufficient to have a few dozen nodes. Also note that the operations in SS are off-chain, so they can be cheaper.

From the analysis provided above, it can be seen that on-chain confidential data implies a price, therefore it is not intended for large-scale computations. While FHE introduces unrealistic and prohibitive costs, MPC is much more manageable. However, there is a clear storage overload, which is the ZKP.

Using Ethereum as a reference, with average values for gas and ETH costs incurred over the last year, storing a single UINT256 (the smallest possible storage slot) costs 1.13 USD. The SS solution without ZKP requires the same amount of storage, hence the same price. However, adding the ZKP increases the cost by 40× for SS and 555× for FHE. Therefore, while SS with ZKP may be acceptable in cheaper scenarios such as layer 2 blockchains, the FHE approach may still be unacceptable.

Regarding computation, our SS-based solution computes off-chain, therefore it could fit in Ethereum-based solutions with MPC fees assigned independently from on-chain costs. On the contrary, FHE computation is on-chain (and much more expensive); therefore, it is not realistic for public blockchains with the existing prices.

UNIVERSIDAD DE MÁLAGA

4.4.3 Threat analysis and discussion

We finally discuss about the main threats that can arise when relying on this framework for confidential computations on a public blockchain, and the approaches to mitigate them.

Impersonation attack

In such attack, the adversary pretends to impersonate either the MPC nodes or the user. In the first case, the adversary can deploy a set of malicious instances which are offered to the user as if they were the real committee, thus receiving all the secret shared data from the user and gaining the power of opening them. The straightforward solution goes through a strong authentication of the nodes, which must be done in the MPC SC preferably. There, identities and public key certificates of the MPC nodes can be placed, such that the user is guaranteed to be communicating with a registered MPC node (which must, in fact, perform some staking prior to belong to the pool of nodes). A different aspect would be for the adversary registering more than t malicious nodes, but this situation will be covered in the next threat. On the other hand, regarding the adversary trying to impersonate the user, that would imply access to his confidential data and capability of determining what to compute on it. The security for this is not different than the security of any SC: the user is the only one that can determine operations based on signatures with his private key. If a user has not decided to reveal a secret shared data from a committee, it will remain undisclosed as long as $n - t$ nodes remain honest.

MPC attack

This attack implies the adversary controlling at least $t + 1$ nodes from an MPC committee, and there are different ways of achieving it. First, it may try to "honestly" register enough malicious nodes inside the MPC SC. This can be mitigated by obtaining some sort of prior knowledge of the MPC node providers, such that none of them can deploy enough nodes to gain power on a committee. In addition, this is more dangerous if the node providers determine the MPC committee themselves, but other mechanisms such that random selection, voting committees, or proactive MPC techniques can mitigate the power of the adversary, who may end with $t + 1$ corrupted nodes but sparse in different committees. The second approach to break MPC security would be for an external adversary to corrupt honest nodes already registered. However, proactive techniques also work in this situation. In addition, setting a large parameter t can be enough to raise the difficulty of corrupting enough nodes. These two approaches are from the point of view of an external adversary, and mainly target to break privacy. With respect to correctness, it is more difficult, mainly because of on-chain verification of the results thanks to NI-ZKP. There can also be dispute resolution mechanisms inside the MPC committees, such that honest nodes point out to others which are misbehaving. Based on these mechanisms to detect misbehavior, owners of the nodes are not expected to misbehave consciously because they can loose their stake if detected.

DoS attack

UNIVERSIDAD DE MÁLAGA

Finally, it may happen that a user pays for a transaction that implies an MPC and never gets the result back. This may happen if the committee of nodes does not reach to obtain the output in time. As introduced in Section 4.4.1.1, a promise can be tagged as rejected if not resolved within a certain amount of time. This would imply the fee of the transaction being refunded to the user from the stake of the MPC committee. On the other hand, DoS attacks can be generated by an external adversary trying to collapse enough nodes from a committee. For this, MPC protocols where the threshold is $t < n$ are a better choice than those with full threshold, where just breaking one node is enough for the committee not to be able to finish the work.

4.5 Summary

In this chapter, we have analyzed the relations that exist between MPC and blockchain technologies. First, this analysis has been done from a wide point of view by detecting both the similarities and the incompatible characteristics between both technologies. It is clear that, despite both of them work in settings with multiple distrustful parties involved, their main objectives are not equal and even somewhat conflicting. However, some proposals in research and industry have identified MPC as an interesting technology to help in solving the lack of privacy problem in blockchain technologies. Following this line of research, we have argued about the general strategies that could be adopted for that and present two specific works that rely on MPC as a core technology for confidential computations in blockchain. First, we have introduced a specific design to embed MPC capabilities into Hyperledger Besu, an Ethereum client that already considers some notion of privacy. We have enhanced its model to be able to use MPC protocols in a consortium-like setting where the node of each organization serves as a participant in the MPC instances. Second, we have proposed a general framework to achieve outsourced MPC in public blockchains with public verifiability of the results. The main purpose of the framework has been to capture a natural way to embed interactive MPC protocols into the atomic nature of blockchain, while maintaining the solution cost-effective. For that, our solutions relies on a pool of MPC nodes that compute on demand, and also on a integration of the promise pattern for asynchronous programming inside the workflow of smart contracts computations, in order to capture MPC in a fine-grained way.

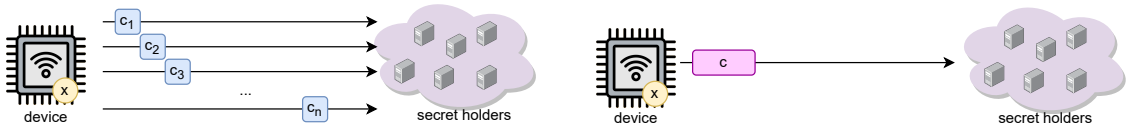
UNIVERSIDAD DE MÁLAGA

Chapter 5

Dynamic and compact sharing of secrets

5.1 Introduction

In this chapter we initiate a new line of study and propose a new family of protocols that are related with an efficient delivery of confidential data to MPC committees. More specifically, the focus is on MPC schemes based on SS, which are most of the n-party protocols for general purpose computations (refer to Section 2.1.3.3 for more details).



(a) Standard secret sharing with n independent ciphertexts.

(b) New secret sharing with a single ciphertext.

Figure 5.1: Two settings for secret sharing data.

As previously introduced, a SS scheme has a dealer, who owns the secret x, to split the secret into n shares $\{x_1, ..., x_n\}$, such that a subset of $t+1 \leq n$ shares are enough to reconstruct the secret x. There is a fundamental assumption in SS schemes to remain secure, which is that each share x_i must reach the holder P_i through a confidential channel, otherwise the secret could be exposed by a gathering of enough shares. In practice, such a confidential channel is instantiated with some kind of encryption, e.g., using TLS if the dealer can establish independent connections with each of the holders, or using public/hybrid encryption if the shares are delivered through an intermediate platform which should not see the shares themselves (cloud storage, blockchain...). Note that these channels do not only imply the cost of encrypting and sending data, but the cost of establishing the channels themselves, in terms of memory (maintaining sockets open), data delivered (authentication certificates), and parallel computation management, if available. For a large n, this may incur in a waste of resources and can lead to an unmanageable setting for some sort of constrained devices, e.g., an IoT device. This model is briefly conceptualized in

UNIVERSIDAD DE MÁLAGA

Figure 5.1a. We now introduce a new research direction, as depicted in Figure 5.1b, that tries to answer the following question:

> *Can the dealer of a secret sharing scheme send a single ciphertext that allows each holder to recover only her own secret share of the original secret, and nothing else?*

5.1.1 Approaches to efficient communication in SS

This chapter introduces a new way of distributing secret shared data that, as far as we are concerned, has never been proposed before in the literature.

First, recall to Beimel's survey [13] for some lower bounds of SS communication cost. Roughly speaking, the size of the share to be sent to each holder must be at least the size of the original secret. Therefore, using standard SS and assuming n holders, $n \cdot len(m)$ is the minimum amount of the overall communication cost for the dealer. We identify two approaches to bypass this lower bound and achieve better overall communication performance.

The first approach has been the standard for SS efficiency and can be generalized as the family of Multi-Secret Sharing (MSS) schemes [173, 174]. In an MSS scheme, a set $\{m_1, ..., m_l\}$ of secret messages is shared with a set of holders $\{H_i\}$, such that each H_i holds l shares of $\{[\![m_1]\!], ..., [\![m_l]\!]\}$. The main goal of MSSS is to reduce the share size by taking advantage of the simultaneous sharing of l secrets. For unconditionally secure schemes, there exists a lower bound $O(l)$ on the share size. This can be achieved by simply running l independent instances of Shamir SS, for example, and therefore achieving the same information ratio than the Shamir scheme. On the other hand, for computationally secure schemes, there exist some that achieve $O(1)$ size. We note that even though these schemes save communication resources for sharing multiple secrets, they still require the device to send at least one share (when $l = 1$) to each secret holder, implying n independent confidential channels.

The second approach is the main contribution of this chapter, which does not focus on amortizing several secrets into a single sharing, but in allowing the dealer to send a single ciphertext from where each holder can retrieve its own share of the secret.

5.1.2 General conceptualization

Prior to introducing the different specific approaches that provide ways of solving the research question introduced before, we try to provide some intuition from an abstract point of view. The purpose is to minimize the work of the dealer such that it only needs to encrypt data and send it. It is straightforward that using a standard encryption scheme, being symmetric or asymmetric, does not provide such property. Some work must be done somehow in relation with the encryption scheme, and for that work, we identify two directions (which can also be combined). The first approach implies a setup or precomputation phase, prior to encryption. This setup allows to generate "special" encryption keys that allow to recover the shares of the plaintext from the ciphertext, instead of the plaintext itself. The second approach implies some sort of computation on encrypted data, such that the shares can be

UNIVERSIDAD DE MÁLAGA

generated "in the ciphertext space", but without revealing the plaintext. Note that this implies that the ciphertext has some sort of *malleability*, otherwise nothing but the plaintext can be retrieved from it. There is an additional property that can be achieved, independently from the approach, which is *post-encryption share delivery*, that allows to select the specific committee of holders that is going to be able to retrieve the shares from the ciphertext after the ciphertext itself has been generated. Note that this property allows to encrypt a set of data first, store it somewhere, and deciding to feed an MPC instance with them later in time. In addition, this can lead to interoperable technologies, such that data can be encrypted and consumed in different ways (either standard decryption or using SS) which are selected later in time.

Sections 5.2 and 5.3 introduce specific technologies to solve this problem with both approaches. We first discuss about a protocol called Homomorphic Encryption with Decryption to Shares (HEDS), by Chillotti et al. [175], which solves this straightforwardly, and then propose new approaches that lead to Oblivious Sharing Re-Encryption (OSRE), a new protocol methodology that employs the second approach to solve the delivery problem.

5.1.3 Use cases

Prior to delving into the details of the different constructions that solve the research question presented above, we introduce a couple of use cases to emphasize the motivation behind this new approach.

5.1.3.1 IoT setting

It is clear that a great effort must be put into an efficient use of resources when working with IoT devices. It is very normal that such devices are deployed in scenarios where their main functionalities are somehow limited, i.e., they carry on very specific tasks. This normally goes by gathering measurements from the environment and sending data to central processing equipment, with a larger amount of resources available, or by executing a specific task that can interact with the analog world when a specific signal reaches the device. The problem with the processing phase is that it normally implies some centralized architecture where a server (or a cloud or edge nodes) receives data from different devices and processes the aggregated of the data. It is easy to notice that this setting can present several threats when the gathered data is somehow sensitive, and that even if the system is secure it implies the exposure of that data to the service provider.

MPC and others privacy-enhancing technologies have been proposed as solutions to enable computations on the aggregated data without exposing anything else but the result. A specific example where this approach has been proposed in the literature is in smart grids [176, 177, 178], where Smart Meters secret share measured data and send it to a set of Computing Parties that will process the aggregate to compute specific functions, and where results can be consumed by other parties. While this adds a protection layer on measurements of specific housings and avoids the exposure of consumption habits of individuals, letting the Smart Meters to send secret shared data to the Computing Parties periodically can imply an impractical cost.

UNIVERSIDAD DE MÁLAGA

We identify our approach as a good solution for this use case because it not only allows the Smart Meters to send a single ciphertext per measurement, but it also allows them to store encrypted data somewhere (in the cloud, for example), and let the MPC committee to be formed later in time, as long as *post-encryption share delivery* is supported by the scheme. Even if some additional work must be done by devices to complete a set up with the MPC committee, besides the delivery of the encrypted data, we notice that it is a process executed only once. After that, the only amount of work done by devices will be to store a single encryption of each measurement somewhere accessible.

We identify another use case in the constrained device scenario for protection of location in Location Based Services, when reports are produced by a smartphone with tagged location data. This use case will be covered more deeply in Section 6.2, where an application for crowd counting in pandemic scenarios is introduced.

5.1.3.2 Blockchain setting

We also identify a blockchain scenario as one where this approach for efficient secret sharing can fit well, and more precisely in a system that instantiates the framework proposed in Section 4.4. When a user inputs some confidential data to an MPC committee that process a functionality described in a smart contract, the user must establish an independent confidential channel with each member of the committee. Despite blockchain interaction is normally based on a single read or write operation through a node provider (to read the state or to send some transaction), this setting implies an additional mechanism, totally off-chain, to establish the communication channels and deliver the shares. A solution which is sometimes used in the context of secret sharing in blockchain, such as in some of the protocols in Section 2.1.3.4, is to use the blockchain as a public bulletin board. This leads the user to encrypt each share with the receiver's public key and upload that ciphertext to the blockchain, from where each holder can retrieve it. This adds an additional interesting property, which is that the blockchain serves as an accountability mechanism that the ciphertexts have been delivered by a specific user, but it also presents a problem: it increases the on-chain cost. For a large committee with n holders, storing n ciphertexts on-chain for each piece of data desired to be shared can be not acceptable because on-chain storage is costly. Therefore, achieving a unique and succinct ciphertext that can be delivered as data inside one transaction and from where each holder can retrieve its share seems a desirable property.

5.2 Homomorphic Encryption with Decryption to Shares

An HEDS scheme, introduced in Chillotti et al. [175], is a special encryption scheme that involves users that can encrypt data, and a committee of secret share holders H. HEDS is formally introduced in Definition 11.

Definition 11 (HEDS scheme)**.** *An HEDS scheme is composed of a tuple of algorithms* (Setup, Enc, Dec, Eval)*.*

- $\mathsf{Setup}(1^\lambda, n) \rightarrow (pk, sk_1, ..., sk_n)$*. On input a security parameter* λ *and a num-*

UNIVERSIDAD DE MÁLAGA

ber of parties n, it outputs a tuple $(pk, sk_1, ..., sk_n)$, with pk a public key and sk_i the private key of party P_i.

- $\mathsf{Enc}(pk; m) \rightarrow c$. *On input the public key and a message m, it outputs a ciphertext c.*
- $\mathsf{Dec}(sk_i; c) \rightarrow y_i$. *Given a ciphertext c encrypted under the public key, this algorithm outputs a share y_i when the secret key share sk_i is used, for each $i \in [n]$.*
- $\mathsf{Eval}(pk; C, (c_1, ..., c_l)) \rightarrow c$. *On input the public key, a circuit description C, and a set of ciphertexts $(c_1, ..., c_l)$ encrypted under the same public key, the algorithm outputs a new ciphertext c that contains the homomorphic evaluation of C using the ciphertexts $(c_1, ..., c_l)$.*

HEDS implies a standard FHE algorithm, but introduces a special decryption algorithm which allows the holders of $\{sk_i\}$ to obtain (non-interactively) a share of the secret m encrypted inside the ciphertext c. More specifically, given that $c \leftarrow \mathsf{Enc}(pk; m)$ and $y_i \leftarrow \mathsf{Dec}(sk_i; c)$ for all $i \in [n]$, we have that $\sum_{i=1}^{n} y_i = m$. A remark is that the decryption to shares can be done on a newly generated ciphertext encrypting a message, or on a ciphertext obtained as the result of the homomorphic evaluation using the algorithm Eval. Also notice that authors implicitly propose in the definition that shares are additive, but it would also be interesting to analyze if other type of shares (e.g., Shamir shares) can be obtained from HEDS. Finally, despite the Setup algorithm is generated by a TTP, it would be easy to implement it by a secure distributed protocol between the holders H using MPC.

It can be noticed that this encryption scheme automatically fulfills our desired properties, since it just takes one ciphertext for Alice to send shares of her secret message to the committee H. In fact, the fully-homomorphism property of the scheme is an addition not needed in our scenario, so the notion of this scheme could be generalized to Encryption with Decryption to Shares (EDS), where $HEDS \subseteq EDS$. At this point, we do not know if $HEDS \subset EDS$ or $HEDS = EDS$, since there is no known scheme other than Chillotti et al. that has the decryption to shares property. In addition, their solution requires FHE to enable setups with more than 2 parties. Therefore, we leave the work of finding an EDS scheme as future work, since taking away the homomorphic property could lead to a scheme more efficient in terms of ciphertext space.

Another important aspect to consider is that HEDS is a proper scheme of the first general approach introduced before, i.e., it takes a setup phase, then encryption, and finally decryption. This implies that the committee must exist prior to encryption, because the public key is generated by the committee itself (if using DKG to avoid a TTP). A straightforward trick to solve this when using HEDS would be a *Proxy-HEDS* solution, where the encryption scheme is also a proper Proxy Re-Encryption (PRE) scheme. This would allow a user, owning a key pair (pk_U, sk_U), to encrypt some message under his own public key $c \leftarrow \mathsf{Enc}(pk_U; m)$ and post it somewhere accessible by a proxy. Later in time, when the committee of holders is formed and Setup executed, producing $(pk_H, (sk_{H_1}, ..., sk_{H_n}))$, the user can generate a re-encryption key $rk_{U \rightarrow H} \leftarrow \mathsf{ReKeyGen}(sk_U, pk_H)$ which allows the proxy to compute $c' \leftarrow \mathsf{ReEnc}(rk_{U \rightarrow H}; c)$ such that c' is a valid HEDS ciphertext for the committee H.

UNIVERSIDAD DE MÁLAGA

Then, from c', each member H_i can retrieve its own share of the secret by invoking $\mathsf{Dec}(sk_{H_i}; c')$. An interesting remark is that a single-hop PRE scheme is sufficient because a single re-encryption is needed, and bidirectionality is not an issue, as long as the committee does not use pk_H to generate encryptions.

The main problem with this PRE solution is that most FHE schemes do not have proper PRE schemes. For HEDS to work, the encryption must be a linear-decryption based FHE scheme, and to the date, the only form to do PRE is by using the general approach proposed in Gentry's thesis [31], which implies circular security. This approach benefits from the Recrypt algorithm used in FHE for bootstrapping, where an old ciphertext with a high level of noise is recrypted to a new ciphertext of the same message under a different key that has a low level of noise (see Definition 12).

Definition 12 (Recrypt algorithm in FHE)**.** *A* Recrypt *algorithm is defined as follows:*

- $\mathsf{Recrypt}(pk_2; C_{\mathsf{Dec}}, c_k, c_1) \rightarrow c_2$. *On input the target public key pk_2, the circuit description of the decryption algorithm of the scheme C_{Dec}, c_k the encryption of sk_1 under pk_2, and c_1 the encryption of the message under pk_1, this algorithm invokes $c_1' \leftarrow \mathsf{Enc}(pk_2; c_1)$ to produce a new encryption under pk_2 of the old ciphertext, and then outputs $c_2 \leftarrow \mathsf{Eval}(pk_2; C_{\mathsf{Dec}}, (c_k, c_1'))$, which is the evaluation of the decryption circuit inside the ciphertext space on inputs c_k and c_1'.*

5.3 Oblivious Sharing Re-Encryption

In this section, we introduce OSRE, a new general procedure for EDS by following the second approach introduced in this chapter, i.e., encryption of data first, computation of the shares by using encrypted data, and finally decryption. As it was explained in Section 5.2, HEDS schemes require FHE with linear decryption, and it is not dynamic by default. The definition of OSRE is intended to be generically enough to capture a wide set of protocols for this compact way of sharing secrets while also enabling protocols which are dynamic and do not require FHE. Note that this definition assumes a semi-honest proxy that is in charge of computing the shares of the message without being able to learn them, i.e., inside the encrypted space. From an abstract point of view, this can be understood as a secure delegation of the process to generate shares of the original plaintext. To be more precise, OSRE is composed of three general phases:

1. **Dealer's data delivery.** The dealer[1] encrypts m using its own long-term public key pk_D and stores $c \rightarrow Enc_{pk_D}(m)$ somewhere public.
2. **Oblivious sharing.** A proxy retrieves c and generates a set of new ciphertexts $\{c_i^* = Enc_{pk_{H_i}}(m_i)\}$, where pk_{H_i} is a key held by the share holder H_i.
3. **Share decryption.** Each holder decrypts c_i^* by using its secret key sk_{H_i}, obtaining m_i.

[1]We understand the dealer and the data owner as the same party.

UNIVERSIDAD DE MÁLAGA

Following this general description for OSRE, we identify different approaches to solve it, which are introduced in the following sections. First, some "intermediate" solutions are introduced in Section 5.3.1, which are solutions that are very close to be functional or that are functional but with somewhat relaxed security models. They are included because they serve as good intuitions to OSRE that can help to develop additional protocols, hence delving into them is left as future work. After that, in Section 5.3.2 we propose a concrete OSRE protocol which is functional and secure, based on NTRU encryption scheme which fulfills both PHE and PRE. We also provide a security proof, an implementation, and an evaluation for this protocol.

5.3.1 Intermediate solutions

This section introduces three "intermediate" solutions which are not entirely functional or practical, but present interesting intuitions to understand OSRE as a general procedure. We believe that additional research on these approaches could lead to more practical and secure OSRE schemes.

5.3.1.1 Approach 1 - Interactive one-time pad

This intermediate solution is not a proper OSRE scheme because it does not fit straightforwardly into the definition provided before. Instead of assuming that the data owner encrypts it with his own public key, this solution works in a more general setting with secret shares. It is composed of both a pre-computation phase and a computation phase on encrypted data, however its assumptions are very relaxed.

Protocol 3: OSRE using secret sharing OTP

Remark: All computations are assumed to be within $\mathbb{F}_p$, where p is a large prime.

1. The committee of holders generate a random blinding $[\![r]\!]$ using a DRNG protocol and without opening r, each committee member sends r_i to the proxy.
2. The proxy reconstructs r from the shares $\{r_1, ..., r_n\}$ and sends r to the user.
3. The user "encrypts" a piece of data m using the blinding factor, i.e., $c = r \cdot m$, and sends c to the committee.
4. The committee generate a set of valid shares for $[\![m]\!]$.
   (a) Sample $n-1$ random $[\![k_i]\!]$.
   (b) Compute $[\![k_n]\!] = r \cdot m - [\![\sum_i k_i]\!]$.
   (c) For each $[\![k_i]\!]$, compute $[\![r^{-1}]\!] \cdot [\![k_i]\!] = [\![m_i]\!]$.
   (d) H_i opens $[\![m_i]\!]$.

Protocol 3 describes the steps, which are mainly based on linear SS operations. The intuition behind it is letting the committee to pre-compute a blinding value that allows the dealer to encrypt data, such that the committee can remove the blinding value and obtain shares of the plaintext m. Since it employs general purpose SS-based MPC, the security assumption is that no more than t parties in the committee

UNIVERSIDAD DE MÁLAGA

will misbehave. In order to base security on that assumption, the committee must not learn the blinding value in clear, otherwise it would be trivial for any share holder to remove it from the ciphertext and learn m. That is the reason for the committee to generate the blinding value r in a decentralized manner, i.e., $[\![r]\!]$ is the secret shared version of r. Note that a naive way to send r to the dealer would be letting each H_i to send r_i to D, and then letting D to reconstruct r and encrypt m. However, this would imply n independent confidential channels, one per each (D, H_i) pair, which is the situation we are trying to avoid in this chapter (even when the dealer now receives data instead of sending it). That is the reason of the proxy P in this protocol, which reconstructs r and sends it to D. Note that it is mandatory that the proxy does not learn c, otherwise it could trivially decrypt it. The last step of the protocol is just a small MPC that allows the committee to obtain valid shares of m without learning m, r, or the shares of the other holders.

We now show in Equation 5.1 that correctness holds in Protocol 3. Note that the protocol uses double sharings, i.e., it generates shares in the form $[\![m_i]\!]$, which means that every holder H_j holds a secret share $m_{i,j}$ of the share m_i. Finally, with n shares $m_{i,j}$, H_i can reconstruct m_i, and with n shares m_i the committee can reconstruct m.

$$\sum_i m_i = \sum_i \sum_j m_{i,j} = \sum_i \sum_j (r^{-1} \cdot k_{i,j}) = \\ = \sum_i r^{-1} \sum_j k_{i,j} = r^{-1} \sum_i k_i = r^{-1} \cdot r \cdot m = m \quad (5.1)$$

5.3.1.2 Approach 2 - Ciphertext space sampling

Ciphertext space sampling is a method for sampling valid ciphertexts without knowledge of their plaintexts, but where ciphertexts fulfill a specific condition. We first note that, for the oblivious sharing phase, AHE allows to obtain additive secret shares, while MHE allows to obtain multiplicative secret shares (see Equations 5.2 and 5.3 respectively).

$$c = \sum_i c_i = \sum_i Enc_{pk}(m_i) = Enc_{pk}(\sum_i m_i) = Enc_{pk}(m) \quad (5.2)$$

$$c = \prod_i c_i = \prod_i Enc_{pk}(m_i) = Enc_{pk}(\prod_i m_i) = Enc_{pk}(m) \quad (5.3)$$

The general idea for the oblivious sharing phase is to obtain one independent ciphertext per each share of the message. For ciphertext space sampling, there is a requirement for the encryption scheme: one can sample random ciphertexts such that they will always be valid under a desired decryption key. We call this notion as *density* in the ciphertext space. Note that this allows to sample valid ciphertexts even when the content of the ciphertexts are not known. Then, thanks to the homomorphism of the encryption scheme, and given that the ciphertext c contains the encryption of the plaintext m, a semi-honest proxy could sample a valid set of

UNIVERSIDAD DE MÁLAGA

ciphertexts such that c_i contains a share m_i of the message m (either additive or multiplicative shares). The steps are the following:

1. Sample $n-1$ random ciphertexts $\{c_1, ..., c_{n-1}\}$.
2. Compute the n^{th} share as
   - $c_n = c - \sum_{i=1}^{n-1} c_i$, if using AHE.
   - $c_n = c / \prod_{i=1}^{n-1} c_i$, if using MHE.

Notice that the set of ciphertexts encrypt shares under the same key used for c. Then, this approach would require an additional step to re-encrypt each c_i to the key of H_i, as specified in the general description of OSRE, hence relying in PRE. Protocol 4 shows the full sequence of steps to achieve OSRE using AHE and PRE. We remark that it can be achieved indistinctly by using MHE, but this imply obtaining multiplicative shares of the secret in the end. Since standard MPC protocols normally use additive shares instead multiplicative shares (such as SPDZ [47, 48, 49]), we also consider a "trick" to achieve additive shares if the scheme used is MHE, which is executing a Multiplicative-to-Additive shares protocol [179]. Such a protocol allows the committee of parties, that holds shares $\{m_1, ..., m_n\}$ with $m = \prod_i m_i$, to perform a secure re-sharing and thus obtaining $\{m'_1, ..., m'_n\}$ wiht $m = \sum_i m'_i$.

Protocol 4: OSRE using ciphertext sampling

1. **Initial data delivery**
   (a) $D \to P$: $c \leftarrow Enc_{pk_D}(m)$
2. **Oblivious sharing**
   (a) P: sample $n-1$ random c_i from the ciphertext space, and compute $c_n = c - \sum_{i=1}^{n-1} c_i$
3. **Share re-encryption**
   (a) $D \to P$: $\{rk_{D \to H_i} = ReKeyGen(sk_D, pk_{H_i})\}_{i=1}^{n}$
   (b) $P \to H_i$: $c'_{H_i} \leftarrow ReEnc_{rk_{D \to H_i}}(c_i)$
4. **Share decryption**
   (a) H_i: $m_i \leftarrow Dec_{sk_{H_i}}(c'_{H_i})$

The main problem of Protocol 4 is that there is no encryption scheme that fulfills all required properties, i.e., density in the ciphertext space, supporting a partial homomorphism, and that also is PRE. For example, Paillier encryption is AHE, but has no PRE, and NTRU has PRE, but it is not dense. The closest one is ElGamal, which is dense and MHE, but its PRE scheme requires a slightly different encryption algorithm that eliminates the homomorphism. We refer to the scheme of Blaze et al. for more details [180]. Despite that, for the sake of completeness, we prove in Theorem 1 that ElGamal ciphertexts are dense when working in a multiplicative group $\mathbb{Z}_q^*$, and if a PRE scheme which does not take away the homomorphic property would be discovered, ElGamal would fulfill Protocol 4. We first introduce ElGamal encryption scheme in Definition 13.

Definition 13 (ElGamal encryption scheme)**.** *An ElGamal encryption scheme is composed of the tuple of algorithms* (KeyGen, Enc, Dec)*, where:*

UNIVERSIDAD DE MÁLAGA

- $\mathsf{KeyGen}(\mathbb{G}, g) \rightarrow (pk, sk)$. *Let* $\mathbb{G}$ *be a cyclic group of order* q *with generator* g. *The private key is a randomly sampled integer* $sk := x \in_R \{1, ..., q-1\}$. *The public key is computed as* $pk := h = g^x$.
- $\mathsf{Enc}(pk; M) \rightarrow (c_1, c_2)$. *To encrypt a message* $M \in \{0,1\}^*$, *first map* M *to an element* $m \in \mathbb{G}$ *using a reversible mapping function. Then choose a random* $r \in_R \{1, ..., q-1\}$, *and output the ciphertext* (c_1, c_2), *with* $c_1 = g^r$ *and* $c_2 = mh^r$.
- $\mathsf{Dec}(sk; (c_1, c_2)) \rightarrow M$. *To decrypt a ciphertext* (c_1, c_2), *compute* $s = c_1^x = h^r$ *and then* $m = c_2 s^{-1}$. *Map* m *back to* M.

Theorem 1. *Any random pair* (e_1, e_2) *with* e_1, $e_2 \in \mathbb{Z}_q^*$ *is a valid ElGamal ciphertext under some private key* $x \in \mathbb{Z}_q$.

Proof. Since $e_1, e_2 \in \mathbb{Z}_q^*$, they can be expressed as the pair (g^a, g^b), being g a valid generator of $\mathbb{Z}_q^*$. Any valid ElGamal ciphertext has the form $(g^r, g^m(g^x)^r)$, with $r, x \in \mathbb{Z}_q$ and $m \in \mathbb{Z}_q^*$, which generates the restrictions $a = r$ and $b = m + rx$. Then, $m = b - ax$, so any valid ciphertext can be expressed as $(g^a, g^b) = (g^a, g^{(b-ax)}(g^{ax}))$. Since the group is modular, any g^{d_1} with $d_1 \in \mathbb{Z}_q$ can be expressed as $g^{d_2}g^{d_3} \equiv g^{d_2+d_3}(mod\ q)$. Hence, for all (g^a, g^b), there always exist an $x \in \mathbb{Z}_q$ s.t. $g^b = g^b(g^{ax})(g^{ax})^{-1}$. □

Remark. *Sampling random ElGamal ciphertexts reduces the plaintext space from* $\mathbb{F}_p$ *to* $\mathbb{Z}_q^*$.

Remark. *The proof uses* g^m *instead of* m, *but correctness holds because ElGamal assumes that* $m \in \mathbb{Z}_q^*$, *where* $\forall a \in \mathbb{Z}_q^* \; \exists \; b$ *such that* $g^b \equiv a \; (mod\ q)$.

Note from the first remark that to achieve the same level of security, q must be larger. To reduce communication costs, ElGamal typically works on elliptic curves. However, despite Theorem 1 can be easily adapted to elliptic curves under the group law, and EC ElGamal is AHE, addition is performed on the curve, i.e., there is no direct mapping to $\mathbb{F}_p$, needed in additive secret sharing. Therefore, this technique cannot be used with EC ElGamal.

5.3.1.3 Approach 3 - Plaintext space sampling

Plaintext space sampling is similar in concept to ciphertext space sampling, but the shares are sampled directly from the plaintext space, which is easier. The hard part is how to achieve this without exposing the secret to the proxy. In Protocol 5, we present a protocol that achieves OSRE using this technique but with a much more relaxed security assumption, which is that the proxy only needs to collude with one share holder in order to retrieve the original secret.

It can be easily noticed that the proxy is the one that generates the $n-1$ random shares by sampling them from $\mathbb{F}_p$. However, the last share is computed inside the ciphertext space thanks to the homomorphism. The proxy would need to collude with H_n to learn m_n and therefore be able to compute $m = Open(m_1, ..., m_n)$, since in additive secret sharing the threshold is the whole set of shares, i.e., $t = n$. Note that even when this security relaxation may not be very realistic, it is not a problem if the proxy is assumed to be semi-honest and never collude.

UNIVERSIDAD DE MÁLAGA

Protocol 5: OSRE using plaintext sampling

1. **Initial data delivery**
   (a) $D \rightarrow P$: $pk_D, c \leftarrow Enc_{pk_D}(m)$
2. **Oblivious sharing**
   (a) P: sample $\{m_i\}_{i=1}^{n-1}$ at random from $\mathbb{F}_p$, computes $c_{add} \leftarrow Enc_{pk_D}(\sum_{i=1}^{n-1} m_i)$, and then $c_n = c - c_{add}$
3. **Share (re)-encryption**
   (a) $D \rightarrow P$: $\{pk_{H_i}\}_i^{n-1}, rk_{D \rightarrow H_n} = ReKeyGen(sk_D, pk_{H_n})$
   (b) P: $c'_{H_i} \leftarrow Enc_{pk_{H_i}}(m_i)$, $c'_{H_n} \leftarrow ReEnc_{rk_{D \rightarrow H_n}}(c_n)$
   (c) $P \rightarrow H_i$: c'_{H_i}
4. **Share decryption**
   (a) H_i: $m_i \leftarrow Dec_{sk_{H_i}}(c'_{H_i})$

5.3.2 Secure OSRE based on NTRU for Shamir SS

In this section we present an OSRE protocol to deliver Shamir shares which follows the plaintext sampling approach from Section 5.3.1.3 but has better security guarantees. The high-level description is introduced in Protocol 6.

Protocol 6: OSRE for Shamir SS

1. **Initial data delivery**
   (a) D: $a_0 = m$
   (b) $D \rightarrow P$: $pk_D, c_0 \leftarrow Enc_{pk_D}(a_0)$
2. **Oblivious sharing**
   (a) P: sample random $\{a_1, ..., a_k\} \in \mathbb{F}_p$. Then, for each holder H_i, compute $Q(i) = \sum_{j=1}^{k} a_j i^j$, $c_{H_i}^{(Q)} \leftarrow Enc_{pk_D}(Q(i))$, and $c_{H_i}^{(W)} = c_0 + c_{H_i}^{(Q)} = Enc_{pk_D}(W(i))$, where $W(i)$ is H_i's share
3. **Share (re)-encryption**
   (a) $D \rightarrow P$: $\{rk_{D \rightarrow H_i} = ReKeyGen(sk_D, pk_{H_i})\}_{i=1}^{n}$
   (b) $P \rightarrow H_i$: $c'_{H_i} \leftarrow ReEnc_{rk_{D \rightarrow H_i}}(c_{H_i}^{(W)})$
4. **Share decryption**
   (a) H_i: $m_i \leftarrow Dec_{sk_{H_i}}(c'_{H_i})$

The basic idea for the protocol is that the proxy is the one that samples the polynomial, i.e., the coefficients $\{a_1, ..., a_k\}$, except the coefficient a_0, which is the secret message m. If the polynomial would be represented in the cartesian plane, it could be noticed that the proxy knows the exact polynomial that encodes the secret m, except the shift it has in the y axis. Therefore, without knowledge of a_0, there are still $|\mathbb{F}_p|$ different polynomials that can be valid candidates, and therefore the secret is kept securely hidden from the proxy view. The next step for the oblivious sharing phase is to achieve a valid evaluation of the polynomial $W(x) = \sum_{i=0}^{k} a_i x^i$ using each holder's index j, i.e., $W(j)$ is P_j's share. To achieve that, the partial polynomial $Q(x) = \sum_{i=1}^{k} a_i x^i$ is evaluated in the plaintext space, and finally, the coefficient a_0

UNIVERSIDAD DE MÁLAGA

is added by using homomorphic encryption. The last steps (re-encryption and decryption) are common to those in Protocol 4, where each P_j's share is independently re-encrypted.

It can be noticed that, differently from protocols that use the ciphertext sampling approach (Section 5.3.1.2), those that use plaintext sampling do not require the encryption scheme to be dense, just to have homomorphic properties (additive, in the case of Protocols 5 and 6) and PRE. There is a scheme that fulfills those properties, which is NTRU. For the sake of completeness, we now introduce NTRU-RLWE [181, 182] and its PRE version [183] in Definition 14.

Definition 14 (NTRU encryption scheme with PRE)**.** *An NTRU encryption scheme with PRE is composed by a tuple of algorithms* (KeyGen, Enc, Dec, ReEncKeyGen, ReEnc) *such that:*

- $\mathsf{KeyGen}(1^\lambda) \rightarrow (pk, sk)$. *Select the following parameters: security parameter λ, ciphertext modulus q, ring dimension N, gaussian key distribution $\mathbf{X}_k$ over the polynomial ring $R = \mathbb{Z}[N]/\langle x^N+1\rangle$ with distribution parameter σ_e, and an empirically selected assurance measure α to minimize the number of bits needed to represent q. The plaintext space is $M = \{0, 1, ..., p-1\}^N$, where $p \geq 2$ is the plaintext modulus. Then, sample polynomials $\mathbf{f}', \mathbf{g} \leftarrow \mathbf{X}_k$ and set $\mathbf{f} = p\mathbf{f}' + 1$ to satisfy $\mathbf{f} \equiv 1 (mod\ p)$. Set $sk := \mathbf{f} \in R$ and $pk := \mathbf{h} = p\mathbf{g}\mathbf{f}^{-1} \in R_q$.*
- $\mathsf{Enc}(pk; m) \rightarrow c$. *Sample random polynomials $\mathbf{s}, \mathbf{e} \leftarrow \mathbf{X}_e$, and compute $\mathbf{c} = \mathbf{h}\mathbf{s} + p\mathbf{e} + \mathbf{m} \in R_q$.*
- $\mathsf{Dec}(sk; c) \rightarrow m$. *Compute $\mathbf{b} = \mathbf{f}\mathbf{c} \in R_q$ and output $\mathbf{m}' = \mathbf{b} (mod\ p)$. Correctness holds as long as there is no wrap-around mod q, i.e., when $||\mathbf{b}||_\infty \leq q/2$. Then, it holds that $\mathbf{m}' = \mathbf{f}\mathbf{c} = p\mathbf{g}\mathbf{s} + p\mathbf{f}\mathbf{e} + \mathbf{f}\mathbf{m} = \mathbf{f}\mathbf{m} = \mathbf{m} (mod\ p)$.*
- $\mathsf{ReEncKeyGen}(sk_A, sk_B) \rightarrow rk_{A\rightarrow B}$. *On input the secret keys $sk_A = \mathbf{f}_A$ and $sk_B = \mathbf{f}_B$, compute the re-encryption key between A and B as $rk_{A\rightarrow B} = sk_A \cdot sk_B^{-1} = \mathbf{f}_A \cdot \mathbf{f}_B^{-1}$.*
- $\mathsf{ReEnc}(rk_{A\rightarrow B}; c_A) \rightarrow c_B$. *On input c_A a ciphertext encrypted with pk_A and $sk_{A\rightarrow B}$ a re-encryption key from A to B, sample a noise polynomial $\mathbf{e}'$ from a distribution $\mathbf{X}_e$ and outputs the ciphertext $\mathbf{c}_B = \mathbf{c}_A \cdot rk_{A\rightarrow B} + p\mathbf{e}' \in R_q$. Note that the re-encryption process increases the amount of error in the ciphertext, which must be within a specific bound in order to fulfill correctness for decryption.*

We now show that NTRU is also additively homomorphic. Given a set of ciphertexts $\{c_i\}_{i=1}^l$ where all are encrypted under the same public key, i.e., $c_i = Enc_{pk}(m_i)$, correctness for homomorphic addition is shown in Equation 5.4. Note that, as it happens with re-encryption, the error is increased with each addition and it must not reach a threshold that produces a wrap-around *mod* q.

$$c_{add} = \sum_i c_i = \sum_i (hs_i + pe_i + m_i) = \sum_i hs_i + \sum_i pe_i + \sum_i m_i = \\ = h\sum_i s_i + p\sum_i e_i + \sum_i m_i = hs' + pe' + \sum_i m_i \quad (5.4)$$

UNIVERSIDAD DE MÁLAGA

5.3.2.1 Making dealer's communication constant

One of the main disadvantages of Protocol 6 is that communication is $O(n)$ in the dealer side, because of the re-encryption keys. However, it is not as costly as in the standard setting with n independent confidential channels, because the latter needs to send n ciphertexts per each piece of data sent to the holders, while in OSRE the re-encryptions are only sent once (as long as the committee of holders does not change). Therefore, the amortized cost in OSRE is still better than in the standard setting when the number of ciphertexts sent is increased. However, it would be desirable to reduce the linearity in the generation of the re-encryption keys.

We now propose a slight variation of Protocol 6 that achieves $O(1)$ communication in the dealer's side, but that requires an extra property in the encryption scheme: to allow threshold decryption. The idea is that the committee of holders, instead of generating individual key pairs (pk_{H_i}, sk_{H_i}), performs a DKG procedure that outputs $(pk_H, \{sk_{H_1}, ..., sk_{H_n}\})$, where sk_{H_i} is a share of the secret key held by H_i. Then, to share a ciphertext with the committee, a single re-encryption key $rk_{pk_D \to pk_H}$ is required, therefore achieving $O(1)$ for the dealer. However, instead of re-encrypting the original ciphertext c directly to pk_H, as it would happen in HEDS, our solution relies on OSRE to generate a set of ciphertexts $\{c_{H_i}\}$, with the peculiarity that all of them are re-encrypted to the same key (as opposed to Protocol 6). The final step implies a joint decryption procedure using threshold decryption where all the partial decryption shares $d_{i,j}$ of c_{H_i} are sent from all H_j to H_i, such that the latter is the only one that can effectively retrieve the plaintext from c_{H_i}, i.e., $m_i = Open(\{d_{i,j}\})$. Equation 5.5 shows that NTRU allows threshold decryption when partial secret keys are additive, i.e., $\mathbf{f} = \sum_i \mathbf{f}_i$, and where $\mathbf{d}_i$ means the partial decryption of a ciphertext locally obtained by H_i using its partial key $\mathbf{f}_i$. However, linear decryption schemes leak information about the partial decryption keys [184], so we leave as future work finding a proper decryption algorithm based on NTRU.

$$\sum_i \mathbf{d}_i (mod\ p) = \sum_i \mathbf{f}_i (p\mathbf{g}\mathbf{s}\mathbf{f}^{-1} + p\mathbf{e} + \mathbf{m}) (mod\ p) = p\mathbf{g}\mathbf{s} + p\mathbf{f}\mathbf{e} + \mathbf{f}\mathbf{m} = \mathbf{m} \quad (5.5)$$

A key remark is that, despite HEDS with re-encryption needs less steps that this version of OSRE with threshold decryption, HEDS needs that the encryption scheme implicitly supports decryption to shares (which needs some sort of FHE), while OSRE does not. However, the caveat is that it needs AHE, PRE, and threshold decryption.

5.3.2.2 Security proof

This section provides a security analysis of Protocol 6. We remark that the property to be guaranteed is privacy of the secret m with respect to a semi-honest proxy.

As introduced before, the protocol needs NTRU because it is additively homomorphic and has PRE. More specifically, NTRU-RLWE has been proven to be IND-CPA both in the original paper [181, 182] and in its PRE version [183]. We remark that IND-CPA in such two cases implies the standard encryption IND-CPA game, but

UNIVERSIDAD DE MÁLAGA

with the addition that the adversary can perform homomorphic operations or re-encryptions on the ciphertext m_b, respectively, which still does not allow to distinguish if the encrypted message was m_0 or m_1. In addition, IND-CCA is not needed because our scheme assumes that the proxy has no oracle access to decryption, i.e., decryption keys are kept by the dealer (for m) and the share holders (for m_i).

For the proof, we follow an **inductive approach** for the oblivious sharing phase. We consider two parameters, which are the threshold t and the number of parties n, and use the notation $OSRE(t, n)$ for an instance of OSRE where the dealer delivers a message m using a Shamir scheme with threshold t and n holders. To ease notation, we consider in this proof that t is the minimum number of shares needed to reconstruct the secret m.

First, we define $OSRE(1, 1)$ as Protocol 6's base case with $t = 1$ and a single holder H_1, where the ciphertext is $c_{H_1}^{(1)} = Enc_{pk_D}(a_0)$, with $a_0 = m$. The case $OSRE(1, 2)$ implies sending the same $c^{(1)}$ to both holders, where any of them reveals m. The following case, $OSRE(2, 2)$, needs to increase the polynomial degree from 0 to 1, since the degree must always be $t - 1$. Therefore, we can define $c_{H_1}^{(2)} = c_{H_1}^{(1)} + Enc_{pk_D}(a_1 \cdot 1)$, i.e., the evaluation of the polynomial $W(i) = a_0 + a_1 \cdot i$, with a_1 a randomly sampled coefficient. Next, $OSRE(2, 3)$ computes an additional share for H_3, but security is the same, i.e., two shares are enough to open the secret. The general case is presented in Equation 5.6 to define the ciphertext of a holder H_i in $OSRE(k, n)$.

$$c_{H_i}^{(k)} = c_{H_i}^{(k-1)} + Enc_{pk_D}(a_r \cdot i^{k-1}) \tag{5.6}$$

Correctness holds because $\sum_i Enc(a_i \cdot x_i) = Enc(\sum_i a_i \cdot x_i)$ is guaranteed by the additive homomorphism of the encryption scheme. Regarding security, it can be noticed that $OSRE(k+1, n)$ can be reduced to $OSRE(k, n)$. In addition, $OSRE(k, n)$ presents the same security as $OSRE(k, k)$, as introduced above. Therefore, any step can be reduced by exposing the content in $OSRE(1, 1)$, i.e., breaking the IND-CPA assumption of the base encryption scheme.

Finally, regarding re-encryption, we recall the sequential composition property in Lindell's tutorial [3]. Since both AHE and PRE are IND-CPA, they are easily simulatable in the stand-alone model. This imply that they can be sequentially composed while maintaining the security properties. Even when Protocol 6 performs n re-encryptions, one per share holder, security under sequential composition still holds because each re-encryption is independent of the others (each one uses a different re-encryption key $rk_{D \to H_i}$).

5.3.2.3 Implementation

We have implemented Protocol 6, relying on two available libraries for NTRU and its PRE scheme, in Java[2]. However, there are two important aspects to consider.

First, the *ReKeyGen* algorithm of the NTRU PRE scheme of Nuñez et al. [183] needs both private keys of the origin and the destination, which is unacceptable

[2] `https://github.com/nicslabdev/ntrureencrypt`

UNIVERSIDAD DE MÁLAGA

```
for(int j = 0; j < nC; j++) {
    OSRE osre = new OSRE(encryptedMessages.get(j), N, N-1, prime,
        paramSpecs);
    List<BigInteger> coefficients = osre.sampleCoefficients();
    for(int i = 1; i <= N; i++) {
        IntegerPolynomial encryptedPartialShare = osre.
            encryptPartialShare(i, coefficients, devicePublicKey);
        IntegerPolynomial reEncryptedPartialShare = ntruReEncrypt.
            reEncrypt(reEncryptionKeys.get(i-1), encryptedPartialShare,
             SecureRandom.getSeed(64));
        // Send share to holder i...
    }
}
```

Listing 5.1: Java code to execute OSRE in the proxy for nC ciphertexts.

in our scenario because it would imply the dealer learning the private keys of the holders. However, there is an small interactive algorithm that allows to derive the re-encryption keys without exposing the private keys, but at the expense of extra communication. The protocol works as follows:

1. The dealer selects a random $\mathbf{r}$ and sends $\mathbf{r} \cdot \mathbf{f}_D \ (mod\ q)$ to the holder and $\mathbf{r}$ to the proxy.
2. The holder sends $\mathbf{b} = \mathbf{r} \cdot \mathbf{f}_D \cdot \mathbf{f}_{H_i}^{-1} \ (mod\ q)$ to the proxy.
3. The proxy computes $rk_{D \to H_i} = \mathbf{b} \cdot \mathbf{r}^{-1} \ (mod\ q) = \mathbf{f}_D \cdot \mathbf{f}_{H_i}^{-1}$.

There is a second aspect, which is that NTRU supports homomorphic addition of plaintexts, but these are mapped to elements inside the NTRU ring, i.e., addition is done between polynomials. However, OSRE needs addition inside $\mathbb{F}_p$, which is the field used by Shamir SS. Therefore, we have implemented a special codification method that allows to perform a single addition of two ciphertexts s.t. they can be decoded back to the sum of integers. For that, first notice that an NTRU plaintext M has ternary coefficients, i.e., from $\{-1, 0, 1\}$. Then, given m_b the binary representation of $m \in \mathbb{F}_p$, we let the coefficient of M with degree k to encode the digit $m_b[k]$. NTRU must guarantee a minimum number of coefficients with values of either -1, 0, or 1, therefore we adjust this once m_b is set. What we do next is to perform addition between the polynomials M_1 and M_2 inside the ciphertext space. This emulates binary addition, but leaving the carry to the final step, i.e., if the result $(M_1 + M_2)[k]$ is -1, that means that there is a carry in that digit. Finally, once the result is decrypted, the carry is applied from the Least Significant Bit (LSB) to the Most Significant Bit (MSB). We note that, if the polynomial M to encode the message m_b is large enough, different strategies can be applied to decide which cells to use for the message bits, therefore adding variations on how to sparse the digits to avoid attacks based on plaintext knowledge.

Listing 5.1 shows the prototype interface for this implementation in Java. It can be seen how the OSRE object provides methods for the three main steps in Protocol 6: it first samples the random coefficients, it then obtains a ciphertext of the share to each holder inside the ciphertext space, and it finally performs the re-encryption of each of the ciphertexts to the public key of each holder.

UNIVERSIDAD DE MÁLAGA

5.4 Cost analysis and comparison

5.4.1 Theoretical analysis

This section provides a detailed analysis of the different techniques introduced in the chapter. The main objective is to identify under which circumstances these new approaches are more efficient for a dealer that wants to share a message with a committee instead of using the standard setting of n independent confidential channels. What is evident is that these approaches' overall cost is higher than the standard case for dealing a secret, but the benefit is reducing the cost in the dealer's side, which may be of particular interest in specific settings. But before diving into the analysis of costs, we show in Table 5.1 a comparison between features or requirements of the different protocols, which may condition the applicability they have in different scenarios. As it can be seen, OSRE is the only one that achieves 2/3, but at the expense of always requiring a proxy. Not needing FHE is somehow relative because it brings the possibility to use schemes which are lighter, but this is not implicit.

Table 5.1: Comparison of the required features for the different solutions.

Requirement	**HEDS**	**ProxyHEDS**	**Protocol 3**	**OSRE Protocols 4, 5, 6**
Does not need proxy	✓	✗	✗	✗
Committee selected after encryption	✗	✓	✗	✓
Does not need FHE	✗	✗	✓	✓

Regarding the asymptotic costs, these are provided in Table 5.2, where STD is the standard dealing phase with n independent confidential channels. It can be observed that OSRE is more intensive than HEDS in the sides of the proxy and the holders, which is the prize to pay for dynamism. In addition, we notice that Protocol 6 with threshold decryption moves the $O(n)$ term from the dealer side to the share holder side, which can be desirable if the holders have more resource power.

Table 5.2: Asymptotic cost comparison

Role	**STD**	**HEDS**	**ProxyHEDS**	**Prot. 6**	**Prot. 6 (threshold decryption)**
Dealer (ciphertext)	$O(n)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$
Dealer (re-enc keys)	n.a.	n.a.	$O(1)$	$O(n)$	$O(1)$
Proxy	n.a.	n.a.	$O(1)$	$O(n)$	$O(n)$
Share holder	$O(1)$	$O(n)$	$O(n)$	$O(1)$	$O(n)$

The rest of this section is intended to provide some specific communication costs for the different encryption schemes used for HEDS and OSRE. We focus on the length of the ciphertexts and the re-encryption keys. For simplicity, the energy savings achieved by using one connection instead of n are omitted. First, we briefly introduce how lengths are computed for each scheme. To ensure a fair comparison, we set the security parameter $\lambda = 128$, and obtain the size of the parameters.

ECIES. This hybrid encryption scheme [185] is used to model n confidential channels, i.e., the standard setting STD. The channel establishment is based on

UNIVERSIDAD DE MÁLAGA

Diffie-Hellman, hence the device sends 510 bits (a point in the elliptic curve). For message encryption, the recommendation is AES-128-CBC for the ciphertext and HMAC-SHA-256 for the authentication tag (in constrained devices), therefore 384 bits.

NTRU-SVES. The parameters are from Hoffstein et al. [186]. A ciphertext in NTRU is a polynomial with N coefficients *mod* q. Therefore, one polynomial can be encoded with $N\lceil log_2(q)\rceil$ bits. As for the re-encryption keys, they are computed as $\mathbf{rk}_{A\to B} = \mathbf{f}_A\mathbf{f}_B^{-1}$. A secret key in NTRU has N coefficients in ternary form, i.e., from $\{-1, 0, 1\}$. Therefore, the length of $rk_{A\to B}$ is $2N$ (2 bits per coefficient).

ElGamal. For DDH/CDH problems, the fastest attack known is GNFS [187]. This attack has a run-time of $L_n(1/3, 2)$, where $L_n(\alpha, c) := e^{(c+o_n(1))n^\alpha ln^{1-\alpha}(n)}$. To achieve 128-bit security, we let $2^{128} \approx L_n(1/3, 2)$. An ElGamal ciphertext has 2 group elements, therefore its length is $2\lceil log_2(q)\rceil$. Regarding re-encryption, algorithms for BBS PRE (similar to ElGamal) compute the re-encryption key as $rk_{A\to B} = g^{sk_B/sk_A}$ (one group element), hence its length is $\lceil log_2(q)\rceil$.

TFHE. A TLWE sample [188] is determined by the dimension d. Then, a ciphertext is defined as $(a_1, ..., a_d, b) \in LWE_{\mathbf{s},\sigma}((q/p)m) \subseteq R_q^{d+1}$. Each element is typically encoded with 32 bits, therefore the length of a TFHE ciphertext is $32(d+1)$ bits. Regarding re-encryption, using the FHE implicit approach proposed in Gentry's thesis [31] we have that $rk_{A\to B} = Enc_{pk_B}(sk_A)$. Therefore, it has the length of one ciphertext.

Table 5.3: Length of one ciphertext and re-encryption key with $\lambda = 128$.

Algorithm	Params	Ciphertext (bits)	Re-Encryption key (bits)
ECIES	n.a.	510 + 384	n.a.
TFHE	$d = 630$	20192	20192
ElGamal	$len(q) = 4096$ bits	8192	4096
NTRU-SVES	$N = 439, q = 2048$	4829	878

Table 5.3 provides the specific lengths for the schemes proposed. It shows that TFHE presents the largest ciphertexts. Regarding ElGamal (for Protocol 4), it is quite large compared to NTRU, mainly because of $\mathbb{Z}_q^*$ instead of elliptic curves. NTRU is the most efficient and flexible, because it works for both Additive SS (Protocol 5) and Shamir SS (Protocol 6).

To compare the different schemes, notice that the cost of sharing one message in HEDS/OSRE is $O(1)$ at the device's side, but $O(n)$ with STD. Therefore, there is a certain n from where HEDS/OSRE are lighter. However, at the dealer's side, there is also the cost of sending the re-encryption keys, i.e., $O(1)$ in ProxyHEDS and OSRE with threshold decryption, and $O(n)$ in OSRE without threshold decryption. The intuition for the analysis is that, even if the cost of sending $\{rk_i\}_{i\in[n]}$ grows faster in n than the cost of sending STD ciphertexts, when the device shares $N_c > 1$ messages, $\{rk_i\}_{i\in[n]}$ are only sent once, but n STD ciphertexts are sent per message. Therefore, for certain n and N_c, HEDS/OSRE are more efficient than STD. Equation 5.7 models this, where the first term represents the cost of sharing with STD channels, and

UNIVERSIDAD DE MÁLAGA

the second term stands for HEDS/OSRE. Notice that n^* fixes the amount of re-encryption keys that are sent in each scheme, which is n in OSRE, 1 in OSRE with threshold decryption, 0 in HEDS, and 1 in ProxyHEDS.

$$n \cdot PK + n \cdot c_{len}^{symm} \cdot N_c > n^* \cdot rk_{len} + c_{len}^{asymm} \cdot N_c \tag{5.7}$$

Equation 5.8 represents the minimum value of n, as a function of N_c, that fulfills the restriction for HEDS and OSRE, i.e., from where they are more efficient than STD.

$$n > f(N_c) = \frac{c_{len}^{asymm} \cdot N_c}{PK + c_{len}^{symm} \cdot N_c - n^* \cdot rk_{len}} \tag{5.8}$$

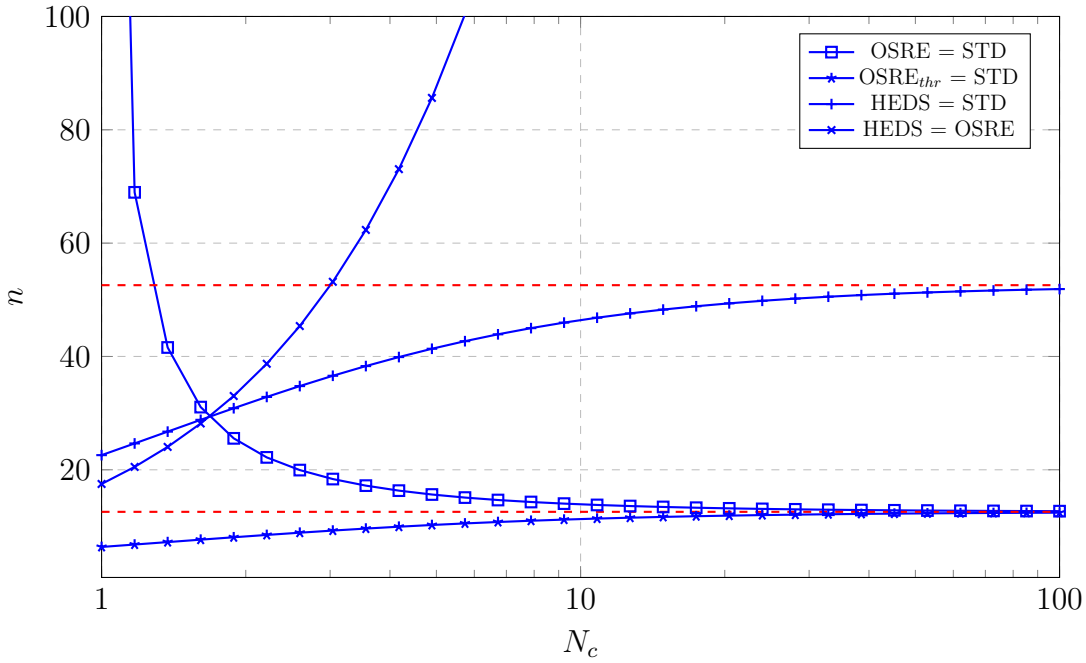


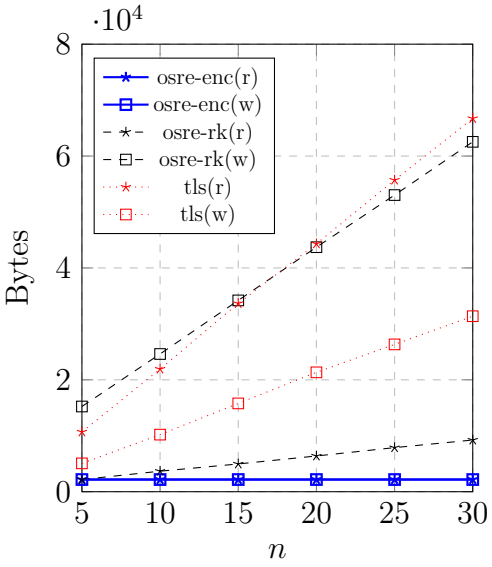
Figure 5.2: Comparison of the minimum n needed for scheme A to outperform scheme B ($A = B$), as a function of N_c.

Using the result of Equation 5.8, we can finally depict specific values of $n = f(N_c)$ in Figure 5.2, for different comparisons between OSRE (NTRU), OSRE$_{thr}$ (NTRU), HEDS (TFHE) and STD (ECIES). Regarding notation of the figure, a curve $A = B$ means that these are the minimum values of n, as a function of N_c, such that scheme A outperforms scheme B.

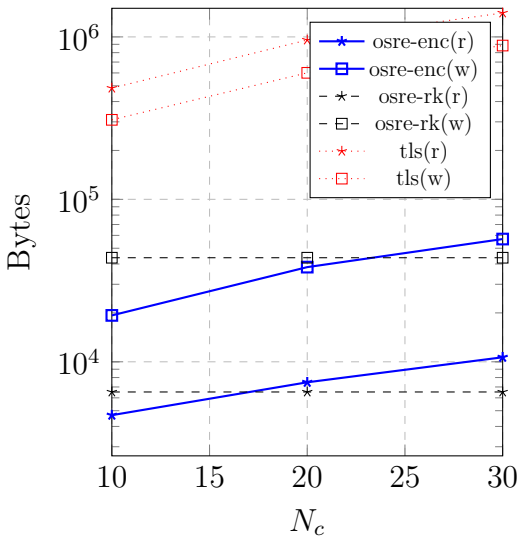
First, $N_c = 1$ is a special case because OSRE cannot outperform STD, due to the n re-encryption keys. It also happens that HEDS outperforms OSRE, but only for $n \geq 18$, otherwise STD is better. It is clear that the most efficient option is OSRE$_{thr}$, but only for $n \geq 7$, otherwise STD is still the best. Second, we analyze the amortized case, i.e., when the same re-encryption key is reused for $N_c \geq 1$ ciphertexts. Note that the area above the curve $A = B$ means that scheme A is more efficient than scheme B. The first conclusion is that OSRE outperforms HEDS in most cases, but those in the area above the curve HEDS=OSRE.

UNIVERSIDAD DE MÁLAGA

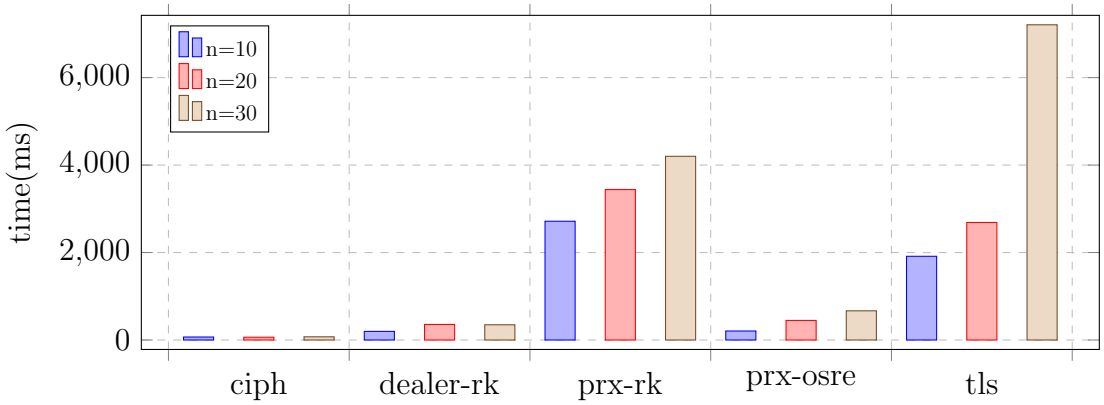
Next, since n varies depending of N_c, we perform an asymptotic analysis to achieve the bounds of the different schemes. First, OSRE=STD presents a lower bound $\lim_{N_c \to \infty} f(N_c) = 12.58$, i.e., OSRE cannot outperform STD with $n < 13$. The bound for OSRE$_{thr}$ =STD is the same, but an upper bound instead of a lower one, which means that OSRE$_{thr}$ always outperforms STD with $n > 13$. It can be concluded then that the benefit of OSRE$_{thr}$ against OSRE is presented for small N_c values. Finally, HEDS=STD presents an upper bound $\lim_{N_c \to \infty} f(N_c) = 52.58$, which means that HEDS always outperforms STD with $n > 52$.



(a) Client side communication cost in Bytes (read/write) for different holder committee sizes n and fixed $N_c = 1$.



(b) Client side communication cost in Bytes (read/write) to send N_c ciphertexts with a fixed $n = 20$.



(c) Running time for different steps and different values of n.

Figure 5.3: Performance evaluation based on simulations.

5.4.2 Experimental analysis

For the experimental analysis, we evaluate our implementation of OSRE compared to an implementation of STD by simulating a network scenario using docker inside a Ubuntu 22.04.1 LTS virtual machine, operating on an Intel ESXi server with an 8-core CPU (2.2 MHz) and 32 GB of RAM. Note that the intention of the simulation is not showing realistic performance values, but a fair comparison between OSRE and

UNIVERSIDAD DE MÁLAGA

the STD case. While OSRE has been implemented using Protocol 6, STD has been implemented with TLS channels. All the simulations have been run with 8 threads. The running times in Figure 5.3c are computed as the average of 5 independent runs of the same step.

Figure 5.3 shows the results obtained. First, Figure 5.3a, for a fixed $N_c = 1$, shows the amount of read/write bytes in the dealer side for both the ciphertext and the re-encryption keys delivery in OSRE, and also in the dealer in STD (using TLS for secure channels). It is clear that the major cost for OSRE is for the dealer sending the re-encryption keys. However, in certain settings (such as IoT), the re-encryption keys can be generated in a different space than the actual sender of the encrypted message, which may be a small sensor. Second, Figure 5.3b shows that the cost for TLS grows much faster with N_c than the OSRE case, where the re-encryption keys are only sent once. This graph has been generated for $n = 20$, but the difference is expected to be greater for larger n values. Finally, Figure 5.3c presents some running times with different n values. It can be seen how the major latency is introduced by the proxy, overall to derive the re-encryption keys due to the interactivity of the ReKeyGen step. If a non-interactive PRE were used instead, this step would be much more efficient. However, despite this phase is not efficient, for a large n TLS is even slower, due to the parallel management of connections. Then, we can conclude that there is a certain point from where OSRE is not only more efficient in the dealer's side than STD, but also that OSRE is more fast in its whole than the time that takes the dealer to securely communicate with each of the secret share holders.

5.5 Summary

This chapter has been devoted to introduce a new family of cryptographic protocols named Oblivious Sharing Re-Encryption, that targets a more efficient and dynamic approach to feed private inputs to secret sharing-based MPC engines. For that, we have first introduced the general problem to solve, which is roughly speaking to allow a dealer to secret share a private message by sending a single ciphertext from where the share holders can retrieve their shares. We have first introduced the use cases where this solution is more natural, together with other schemes with are close to this. Then, we have provided intermediate solutions to OSRE, that captures better the nature of this approach and establish starting points to develop new protocols that fit within this group. Finally, we have provided a concrete solution to OSRE which is based on the NTRU encryption system and its characteristics of being AHE and PRE. Together with the concrete protocol, we have provided a security proof, an implementation, and an evaluation, both from the theoretical side and by performing experiments with the implementation. We have concluded that OSRE achieves its purpose in some settings, allowing data owners to share data with MPC committees in a more efficient and dynamic way.

UNIVERSIDAD DE MÁLAGA

Chapter 6

Applications of MPC

This chapter addresses mainly MPC, but also ZKP, from a practical point of view, and this is done by proposing and discussing about different applications. The main objective is to detect scenarios where MPC by its own, and MPC combined with blockchain technologies can be beneficial. We do so by introducing three different applications that fit within three areas that imply the use of sensitive data, and where protecting them with guarantees makes sense enough to introduce MPC and publicly verifiable mechanisms. These areas are, in order of appearance in this chapter: (1) sensitive data marketplaces and healthcare applications, (2) location-based services, and (3) digital payments and cryptocurrency. In the next sections, we provide concrete designs for the applications, together with performance metrics and discussion that will help to understand how MPC and ZKP can help to these areas of application.

6.1 Private Set Intersection in blockchain

As analyzed in Section 3.3, PSI is one of the most widely researched problems within MPC, mainly because of the wide set of use cases it presents. In this section, we present PSI as an interesting problem to be solved in applications running on blockchain systems where transparency and traceability is a key aspect.

6.1.1 PSI in consortium blockchains

In Section 4.3, we presented a design to enable MPC instances in a private Besu blockchain, where a set of distrustful nodes collaboratively run the MPC on their private inputs. In addition, we introduced in Section 2.2 the concept of consortium blockchain, where different organizations that do not fully trust each other (and therefore are modeled as different trusted domains) share a decentralized and transparent ledger. Consortium blockchains have the benefits of only enabling access to authorized parties while maintaining a sufficient amount of decentralization, contrary to purely private blockchains orchestrated by a single manager. The consortium approach is interesting mainly to some industries where collaboration leads to a better understanding of data, performance of procedures, and Return Of Investment (ROI), such as the financial sector, healthcare, or supply chain management.

UNIVERSIDAD DE MÁLAGA

In addition, MPC is also very strong in these collaborative scenarios, where there exist data related to business insights that cannot be shared with other organizations, but that could lead to great results for analysis if computed privately.

We now present two examples of consortium blockchain in industry where having MPC capabilities could be desirable:

- **Synaptic Health Alliance**[1]**.** It is a consortium blockchain that provides a healthcare data exchange, where members create a complete, shared, and trusted source of data across healthcare. It uses blockchain to improve provider directories and defends that a decentralized approach to exchanging information allows companies to avoid building duplicate point-to-point connections.
- **RiskStream Collaborative**[2]**.** It provides an insurance-specific enterprise blockchain platform, where users can access to an insurance data exchange. They also build industrywide working groups to bring the industry together in collaboration to build applications using emerging technology that solve multiparty business process challenges.

We will now set the focus on PSI and the healthcare area, even when MPC in general, and PSI in particular, could have applications in other use cases such as collaborative malware analysis or bot detection, where malware analysis enterprises or Internet Service Providers (ISPs) could fit in a consortium environment where encrypted data and analytic procedures are handled with transparency and traceability. Regarding the healthcare area, there is a particular use case where PSI is very natural and straightforward, which is privacy-preserving genome analysis. The reason is twofold: genome is analyzed as a sequence of data, which could be encoded as an ordered set, and it contains highly sensitive data for users, and therefore should not be exposed.

There exist works in the literature that have pointed out PSI as an interesting tool for genome analysis. First, Baldi et al. [189] proposes different PSI-based solutions for different forms of privacy-preserving genome analysis. More precisely, authors identify PSI-CA as a tool for Genetic Paternity Tests, A-PSI for Personalized Medicine, and PSI for privacy-preserving Genetic Compatibility Testing. It is interesting to briefly analyze how performance in PSI has evolved during the years, because genome analysis is very expensive in computational terms. The number of nucleotide in human genome is around $3 \cdot 10^9$, and according to Baldi et al., computing a PSI on a fully sequenced genome is prohibitively expensive, taking between 4 and 5 days of computation. However, this paper was published in 2011, and since then, PSI has evolved greatly. Let's consider the most efficient PSI protocol from Section 3.3, i.e., Raghuraman and Rindal [134], which achieves 8.055 seconds for a 2P-PSI where sets have 2^{24} items. We notice that $3 \cdot 10^9$ is around 178×2^{24}, and therefore, computing PSI on the full genome sequence could be approximated to take around 24 minutes. However, Baldi et al. already provided a solution to perform tests on pre-processed genome data where the sets are much smaller, and therefore this use case will become much more efficient with actual PSI solutions. Another work is that of Mma et al. [190], which proposes a secure approximation of the edit distance method to be applied on genome data. For this, it relies on

[1] `https://www.synaptichealthalliance.com/project`

[2] `https://www.riskstream.org/`

UNIVERSIDAD DE MÁLAGA

PSI over shingled data, and then on Garbled Circuits for banded alignment. These procedures allows researchers to query for similarities between genomes of different patients, find specific patterns, and provide more accurate diagnosis.

We notice that genome, healthcare, and other types of data previously introduced, can fit in a collaborative marketplace based on blockchain technologies. An interesting feature of blockchain, apart from transparency and accountability, is that of programmability thanks to smart contracts. These are a good tool for policy enforcement, which is a feature very present in a consortium where sensitive data is used. Following this concept, we could envision a blockchain deployment where contractual agreements between organizations are enforced by the blockchain in a decentralized manner, and where sensitive data is consumed by MPC automatically, only when it is enabled by the accepted policies. For example, if the marketplace considers pay-per-use data, an automatic trigger could be the on-chain purchase of a private dataset to be consumed through MPC. This could lead to the automatic execution of a smart contract supporting MPC, which consumes the datasets provided by the seller and the purchaser, being the latter the one that will receive the outcome of the computation.

This scenario with automatic triggers introduces a way of working with data somewhat different of our implementation for PSI in Besu, which is introduced in the next section. Since we considered a design where datasets are uploaded to the blockchain nodes when instantiating the PSI, we notice that marketplace-like scenarios are likely to have datasets already uploaded somewhere public, probably in encrypted form. To avoid users having to upload their datasets to the MPC functionality each time they want to execute it (this is even harder if the functionality is executed programmatically by a trigger), we identify our construction from Chapter 5, OSRE, as a useful tool. This will be more elaborated in Section 6.1.3.

6.1.2 Implementations of PSI in blockchain

6.1.2.1 Implementation in Besu

For all the reasons introduced above, that point out to PSI as an interesting MPC protocol, we selected it as the first one to be implemented inside our design for MPC in Besu and built a functional prototype. For that, we designed an ISC that captures the phases of a 2P-PSI generically, such that it can support different implementations of PSI. Since it is a two party protocol, the interface is quite simple, as shown in Listing 6.1: the contract constructor is used by Alice to generate the PSI instance and upload her set to her Besu node, and the *computePSI* method is used by Bob to upload his set to his node and let both nodes to execute the PSI.

Thanks to this ISC, Alice and Bob can execute a PSI through a standard interaction with their respective Besu nodes, which is by using a web3js interface. Both Listings 6.2 and 6.3 show the code executed by Alice's and Bob's applications to send transactions to their nodes that trigger the execution of PSI. It can be seen how they both use the public keys of Tessera nodes to fulfill the *privateFrom* and *privateFor* fields, in order to select the delivery path of the private transaction. In addition, the inputs loaded by Alice and Bob follow the specifications of the methods in the ISC, i.e., Alice specifies who are the sender and receiver of the PSI, the

UNIVERSIDAD DE MÁLAGA

```
contract PSI {
    constructor(
        address _alice, address _bob, uint256 _maxLength,
        uint256[] memory _aliceSet) public;
    function computePSI(
        uint256[] memory _bobSet) public bobOnly()
        returns(uint256[] memory);
}
```

Listing 6.1: Solidity interface for 2P-PSI.

```
// Imports...
async function main() {
    // Init web3NodeAlice, binary, abi...
    const inputs = [alice.address, bob.address, 5, [1,2]];
    const contract = new web3NodeAlice.eth.Contract(JSON.parse(abi)
        );
    let transaction = contract.deploy({data: `0x${binary}`,
        arguments: inputs});
    let contractOptions = {
        data: transaction.encodeABI(),
        privateKey: alicePrivKey,
        privateFrom: alicePubKeyTessera,
        privateFor: [bobPubKeyTessera],
        extendedPrivacy: "2P-PSI",
    };
    let txHash = web3NodeAlice.priv.generateAndSendRawTransaction(
        contractOptions);
    const contractAddr = await getPrivateContractAddress(txHash);
    const tx = await web3Node1.priv.getPrivateTransaction(txHash);
}
```

Listing 6.2: JavaScript code to call *PSI load* method.

maximum set length, and uploads her private set, while Bob only has to upload his set if he is allowed to consume the PSI as a receiver, which is specified by the modifier *bobOnly*. Finally, the field extendedPrivacy specifies which type of MPC protocol is being instantiated, to let the Besu nodes knowing which functionalities from the MPC library have to be executed. Notice that if there were several types of 2P-PSI in our implementation, this could be specified as *2P-PSI-X*, with X a selector for the specific PSI protocol.

Since this design is functional, it has space left to improvements. We left as future work to provide a more robust implementation with a set of PSI protocols (both two-party and multi-party) that allow us to run a benchmark and analyze the efficiency of our design.

6.1.2.2 Design for public blockchain with MPC framework

Apart from the design to run MPC protocols in a private Besu blockchain, we also provided in Chapter 4 a framework to run outsourced MPC protocols in a public blockchain by a pool of MPC committees. Despite the framework does not present a proper implementation yet, we provide a brief discussion in this section about using

UNIVERSIDAD DE MÁLAGA

```
// Imports...
async function main() {
    const inputs = [2,3,4];
    const contract = new web3NodeBob.eth.Contract(JSON.parse(abi));
    let functionAbi = contract._jsonInterface.find((e) => {
        return e.name === method;
    });
    let functionArgs = web3NodeBob.eth.abi.encodeParameters(
       functionAbi.inputs, inputs).slice(2);
    let functionCall = {
        to: contractAddr,
        data: functionArgs !== null ? functionAbi.signature +
           functionArgs : functionAbi.signature,
        privateKey: bobPrivKey,
        privateFrom: bobPubKeyTessera,
        privateFor: [alicePubKeyTessera],
        extendedPrivacy: "2P-PSI",
    };
    let result = await web3NodeBob.priv.
       generateAndSendRawTransaction(functionCall)
    .then((txHash) => {
        return web3NodeBob.priv.waitForTransactionReceipt(txHash);
    }).then((result) => {
        return result;
    });
}
```

Listing 6.3: JavaScript code to call *PSI consume* method.

PSI in such a framework from a design point of view.

The first aspect to notice is that, since the framework targets outsourced MPC, it is a straightforward solution to instantiate O-PSI or VD-PSI, which are special types of PSI where an external service provider computes the intersection without learning the input sets, as introduced in Section 3.3. However, another aspect is that PSI implies a multi-input provider setting, which is not directly captured by the naive design of the framework. This naive design captures the scenario where an MPC functionality described in a smart contract is self-contained, i.e., the user that calls the execution of the function provides the inputs to the committee, which computes on these inputs and provide a result. However, if a smart contract requires the inputs from different users to exist at once in order to execute the MPC, some kind of confidential storage must be implemented. One option can be to deploy storage committees (with a large set of parties), that always handles inputs that escape the context of the function, and later move confidential data to selected committees, if possible. Another option is to delivery data to its target committee from the first step, but there must be some agreement in which committee will hold data from all users within the same smart contract context, otherwise the computation cannot be completed. This can even be automatically handled by the blockchain system itself, without requiring the users to specify which committee they want to compute their data. Another valid option is to negotiate this on contract deployment. One possibility can be using a security indicator in the contract constructor that selects a committee with more or less number of parties (and security capabilities) depending

UNIVERSIDAD DE MÁLAGA

```
contract PSI {
    MPC inputSets[][];
    public address inputTrack[][];

    function MPC uploadSet(MPC set[], public uint id) {
        inputSet[id] = set;
        inputTrack[id] = msg.sender;
    }

    function MPC computePSI() {
        MPC intersection = PSI();
        return intersection.open();
    }
}
```

Listing 6.4: Solidity smart contract to compute PSI in public blockchain.

on the security indicator. All these are specific implementation decisions which are left as future work to consider when actually instantiating the framework. However, we provide in Listing 6.4 a smart contract design that captures the functionality of a multi-party PSI. Notice that there is an *inputTrack* variable that controls which users have uploaded their sets as MPC inputs by calling *uploadSet*, and the function *computePSI* finalizes the protocol by running the PSI on the inputs provided.

6.1.3 OSRE for input delivery

It can be noticed from the sections above that providing the inputs to an MPC instance is not as straightforward as it may seem, overall if the input set is large, it changes from time to time, or if the MPC instance is outsourced. For all of these, we identify OSRE as an interesting construction to minimize the impact of providing the inputs to MPC.

The main benefit of using OSRE is that data must only be encrypted once by the owner and using its own public key. Then, data can be uploaded to any public storage, either the on-chain storage or somewhere off-chain accessible by the nodes, such as the Inter Planetary File System (IPFS). Then, at computation time, the only work to be done by the data owner is to generate the re-encryption keys to let the proxy re-encrypting data to the destination MPC committee. Notice that this approach introduces the proxy as a role that also belongs to the blockchain ecosystem, in addition to the pool of MPC nodes, and therefore it can also be instantiated with incentives notions and publicly verifiable mechanisms, such as NI-ZKPs. Even beyond that, the proxy can be designed to be a decentralized service, i.e., an MPC by itself, where the re-encryption generation process requires more than one party to be involved[3], and therefore minimizing the impact of an adversary in corrupting the proxy.

In Listing 6.5 we provide a modification of the *uploadSet* function, from Listing 6.4, that considers OSRE for data sharing. Notice that the input set is uploaded to the blockchain in encrypted form, and the proxy to perform re-encryption is also selected in the function. Then, the off-chain mechanism will be for the user to

[3]See Umbral, from NuCypher: `https://github.com/nucypher/pyUmbral`

UNIVERSIDAD DE MÁLAGA

```
function MPC uploadSet(Encrypted set[], address _proxyId, public uint
    id) {
    inputSet[id] = set;
    proxyId = _proxyId;
    inputTrack[id] = msg.sender;
}
```

Listing 6.5: Function for uploading the set to the smart contract using OSRE.

share the re-encryption keys with the proxy, instead of sending the inputs to the MPC committee. This not only implies a saving in resources if the input data set is large, but also facilitates dynamism in the MPC instance, e.g., if some kind of proactive MPC is used by the blockchain to compute MPC. Finally, Figure 6.1 shows a high-level interaction diagram between the entities involved in the process. Notice that authorize re-encryption means that the user sends the re-encryption keys to the proxy, which can even be a single key if OSRE is instantiated with the scheme proposed in Section 5.3.2 that employs threshold decryption.

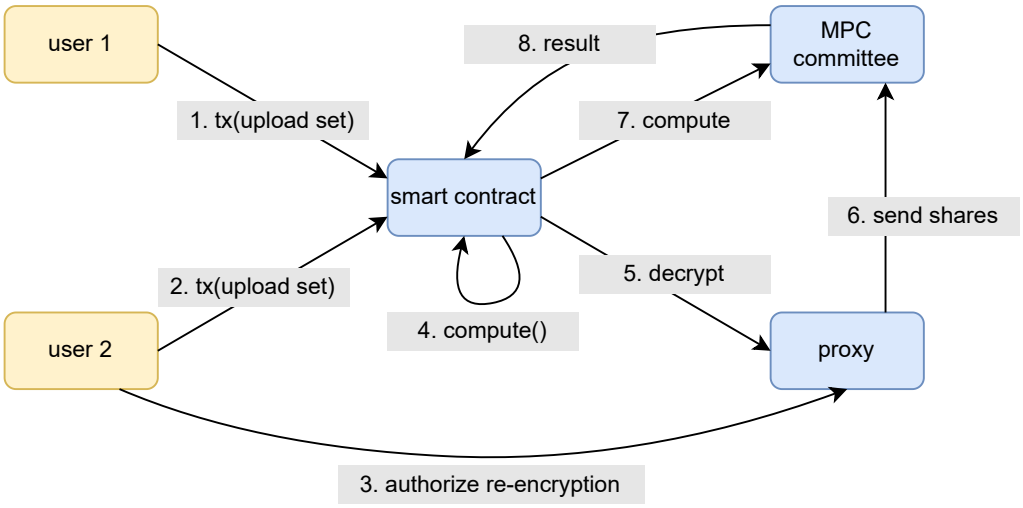


Figure 6.1: Uploading confidential data to a SC using OSRE.

6.2 Location based services with privacy

In this section, we introduce an application that can benefit of using MPC to protect user's sensitive data. As it will be explained below, this application takes the functionality of a contact tracing application in the context of a pandemic scenario to achieve crowd counting. However, our solution for crowd counting fits in a wider category of applications which are *Location Based Services* (LBS), therefore our solution for privacy can also be considered for other LBS applications. LBS are, essentially, services that rely on user's location to provide specific content related with that location.

LBS is quite an old research field and has received much attention through the years [191, 192]. D'Roza and Bilchev [191] identify two basic broad categories of LBS. The first gathers *user-requested* scenarios, where the user requests the location and then perform queries based on it. The most extended applications for this category

UNIVERSIDAD DE MÁLAGA

are navigation-like applications, to allow guidance, discover the shortest paths or retrieving information about commerce close to a specific location. On the other hand, the second category gathers *triggered* scenarios, where the occurrence of an event or condition triggers the request about the location. A great example of this is calls to emergency services, where the location can be provided by the mobile network once the call is established. There exist many others applications that fit in LBS, such as fine grained advertising, vehicle-to-vehicle applications, games that provide events on a real map, etc.

Regarding our solution, we first provide in Section 6.2.1 a design for a crowd counting application that reuses one of the most accepted designs for a contact tracing application during the COVID-19 pandemic. Then, we introduce in Section 6.2.2 how MPC can be used to protect location data with respect the service provider while ensuring its functionality.

6.2.1 Contact tracing and crowd counting

Contact tracing is a problem from the public health area, which purpose is to identify people who might be exposed to other people infected by some pathogen, i.e., the contacts. Gathering data of those contacts allows to take actions in order to avoid the spread of the pathogen and test contacts for infection, also allowing a faster detection.

The contact tracing problem was very relevant a few years ago, because of the impact of the COVID-19 pandemic, a great disease generated by the virus SARS-CoV-2, which has lead to enormous problems around the whole world. The virus spreaded quite fast and many people dead; others lost their jobs and many businesses had to close. COVID-19 lead the world to large controversy regarding the approaches that had to be used to avoid the spread of the virus. Some countries obliged people to stay at home during periods with large infected rates, and during periods with lower rates, different preventive measures were used. In these periods, contact tracing solutions (many of them relying on technology) were identified as useful tools for a fast detection of infection, therefore lowering the exposure of the virus to non-infected people.

On the other hand, crowd counting or crowd density estimation is the problem of counting how much people are there in a specific area. This problem has acquired great importance overall after the widespread growth of the urban scenario, where highly crowded places may lead to problems with implications on social, politics, or economics areas (traffic jams, risk of attacks, general insecurity, blocking of access to stores, etc.). Another case where crowd counting has had a lot of relevance is in the estimation of participants in demonstrations, because the estimation value can involve changes on perception and alignment of people with respect to political groups or social movements. However, in the context of this work, we identified crowd counting as a useful tool for pandemic situations. As introduced above, contact tracing is a widespread solution for a fast handling of infections after they has produced. We identify crowd counting as a prevention tool for avoiding crowded places in advance, therefore reducing the exposure of citizens to the disease.

The following sections introduce the most accepted approaches to solve contact

UNIVERSIDAD DE MÁLAGA

tracing by using technology, and how we have used a contact tracing application to build a privacy-preserving crowd counting application on top of that.

6.2.1.1 Contact tracing approaches

The paper of Chowdhury et al. [193] provides an extensive analysis of the different contact tracing solution proposals during the COVID-19 pandemic.

There are many technologies that have been proposed in the literature for proximity measurement: GPS, Bluetooth, WiFi, cellular networks, RFID, and NFC. However, most solutions followed a very common line of designed based on Bluetooth, which allow users to exchange data when they are close enough. Other solutions such as WiFi restrict the application to a fixed location, e.g., an enterprise building, and RFID or NFC reduces usability. On their part, GPS or cellular networks may expose sensitive data if not secured correctly. An example of GPS-based contact tracing is Reichert et al.'s work [194], where users periodically store pairs $(location, timestamp)$ in their devices. If a user Alice is tested positive, she shares her data with the Health Authority server. If another user Bob queries the server to check infected contacts, Bob and the server engage in an GC-based MPC where Bob locally evaluates the GC with his data, and the GC encodes Alice's data, in order to find the intersection of their location data. It can be seen that GC evaluations on the users' side may be highly intensive in terms of resources, and in addition, the server must be trusted because it learns location data of infected patients.

The rest of the works proposed for contact tracing follow a similar design, but with differences mainly in the type of cryptography that protects data that can be linked with users and the approach they use to compute the contacts. Figure 6.2 shows this high-level design. The main idea is that each user is identified with a unique ID (seed, session key...) from which they can generate a list of pseudo-random IDs. Then, Bluetooth Low Energy (BLE) is used to exchange the pseudo-random IDs, i.e., whenever two users are close enough, their devices will detect the broadcast emission of the others, which contain the IDs.

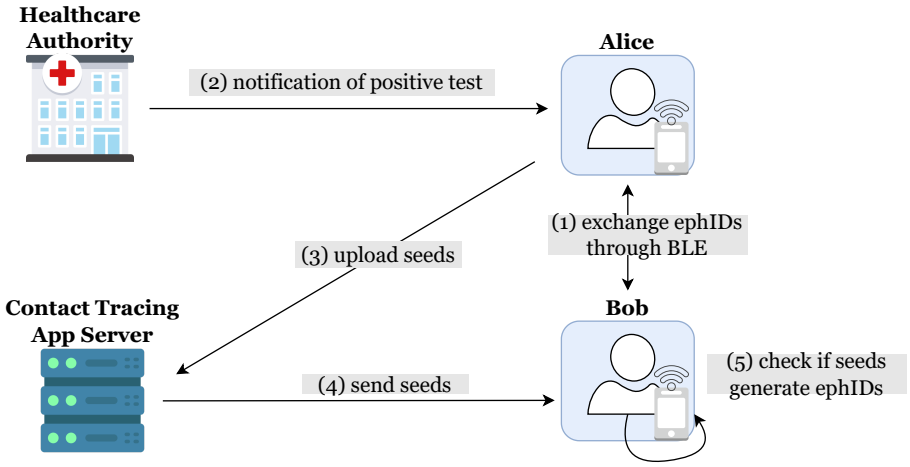


Figure 6.2: High-level overview of a contact tracing app.

The next phase allows to detect the contacts, and it is dependent on how the users' IDs are generated. The problem can be seen as computing the intersection of two

UNIVERSIDAD DE MÁLAGA

sets: if Alice has a set of received IDs $R_A = \{a_1, ..., a_n\}$ and Bob's set of broadcasted pseudo-random IDs is $S_B = \{b_1, ..., b_m\}$, it can be learned that Alice is a contact of Bob if there is some $a_i \in R_A$ such that $a_i \in S_B$ too. It can be noticed that the protocol of Reichert et al. [194] already does something similar, but the intersection is computed on location data and using general purpose MPC evaluated on the client side.

There is another contact tracing application, Epione [115], which in fact applies a PSI-CA protocol between the application server and the requester for exposure verification (see Section 3.3 for a wider explanation of PSI). The PSI-CA proposed is based on the DH assumption and somehow follows the OPRF strategy, where the server maps the client set S_c to another space using a key k without learning S_c, and the client does not learn k. To achieve PSI-CA instead of PSI, the server sends the client's mapped set in a randomly permuted order. Then, the mapped server set is also sent to the client, who counts the number of coincidences, therefore obtaining a risk index based on the cardinality. However, this protocol is $O(n)$ in communication and computation for the client, which can be costly when instantiated with DH for mobile phones, overall when exposure checks must be done periodically.

On the other hand, the most accepted protocols such as DP3T [195] or BlueTrace [196], employ more efficient methods which may sacrifice a bit of privacy but lead to better usability. DP3T also has an intermediate solution which is based on the efficient method of computation, but makes it unlinkable thanks to the use of Cuckoo Filters, at the expense of sacrificing a bit of performance (however, not that much as in MPC/PSI solutions). We specify all the steps of this protocol in Protocol 7. In more efficient settings [195, Section 2.1] the backend distributes the list of seeds send by positive tested patients, from where everyone can recompute $EphID = H(seed)$.

6.2.1.2 Crowd counting

There is quite a large literature for crowd counting, where computer vision technologies have been the most accepted ones. The approach is to process images taken from the crowded places to estimate the number of people in the scene. For that, three main techniques are used: (1) detection-based approaches, using feature extraction [197]; (2) regression-based approaches, which construct density maps [198]; and (3) Convolutional Neural Network (CNN)-based approaches, that achieve the best accuracy [199]. Other works reuse this ideas in novel scenarios, such as the edge [200] or inside embedded terminals [201]. Since computer vision present some problems, such as the need of placing specialized devices in specific locations, or the privacy concerns of people when recorded permanently, other strategies are based on sensors [202] or device signal processing techniques [203, 204].

Contrary to these approaches for crowd counting, our solution reuses the ephemeral ID-based approach in P2P contact tracing as a basic building block. In fact, contact tracing and crowd counting following this approach can be seen as parallel features of a single large application. The rest of the section elaborates on how the crowd counting problem is solved for pandemic scenarios.

First, from a conceptual point of view, there can be two strategies for achieving crowd counting based on IDs. The first strategy assumes one or more anchor points, which are base stations that periodically broadcast IDs through a radio network

UNIVERSIDAD DE MÁLAGA

Protocol 7: DP3T unlinkable contact tracing

1. **Seed generation and broadcasting**
   (a) For each epoch i of fixed duration L, each user generates a seed $seed_i$ and an ephemeral ID $EphID_i = H_{128}(seed_i)$.
   (b) Users periodically broadcast $EphID_i$ using BLE beacons during epoch i.
   (c) Users keep their seeds in storage during 14 days.
2. **Beacon reception**
   (a) When a user receives a BLE beacon, he stores:
   - $H(EphID||i)$.
   - A exposure measurement, such as signal strength.
   - A timestamp (the date).
3. **Positive diagnosis**
   (a) User tested positive uploads the set $\{(i, seed_i)\}$ for all epochs in a fixed amount of days.
4. **Decentralized exposure verification**
   (a) Every 2 hours, the backend creates a new Cuckoo Filter F and for all $(i, seed_i)$ received from positive tested users inserts $H(H_{128}(seed_i)||i)$, which is equal to $H(EphID||i)$.
   (b) Users periodically download F generated by the backend and check if the $EphIDs$ they have received through BLE beacons are included in F.

interface. All users' equipment (such as mobile phones) which are in range will receive the IDs and will notify a central server. Then, based on the IDs, the central server can aggregate users based on their "location", i.e., the area of coverage of the anchor point. The second strategy is to invert the roles with respect to the IDs. Users' equipment are now the ID broadcasters and the anchor points will receive the IDs. Therefore, the latter will be in charge of aggregating the IDs and reporting the central server to estimate the amount of people inside its area of coverage.

In this work, we follow the second approach but without the restrictions of an enclosed area. Following the solution for contact tracing, we assume the users exchange IDs between them in a peer to peer scenario, and they send to the central server the information needed (through periodical reports) to estimate the amount of people in different areas. The rough idea is just letting the server to count in real-time the number of different IDs detected in a specific location and output that value such that it can be consulted by users.

Spatial and temporal management

The first detail that must be considered is that, in order to avoid anchor points with fixed location, users must provide their location in their reports themselves. This allows the server to compute the aggregation of users at the specific location queried by some user and with a certain granularity. The most accurate scenario would imply each user notifying the central server each time they receive a new ID through the BLE interface. However, this may imply a large number of connections

UNIVERSIDAD DE MÁLAGA

between users' devices and the server, overall when they are in crowded places. To reduce resource consumption, we propose a solution based on batching, where the user notifies the server at time $t + \Delta$ with all the IDs received in the interval time $(t, t + \Delta)$. Note that this, however, reduces accuracy due to temporal and spatial errors. We provide a model description and parameter-based analysis for accuracy below. Algorithm 1 provides the pseudo-code for the sampling algorithm performed by users, where we select the mean of the location and the final timestamp of the batch when notifying the server.

Algorithm 1 Batched sampling algorithm

```
while true do
    Initialization : timer, location, batch
    if read(ID_x) then
        batch.insert(ID_x)
        timer.init(getTimeEnd(location.getSpeed))
        locationInit ← location.getPosition
    end if
    while timer.notEnd do
        if read(ID_y) then
            batch.insert(ID_y)
        end if
    end while
    locationEnd ← location.getPosition
    locationMean ← mean(locationInit, locationEnd)
    batch.save(locationMean)
end while
```

ID aggregation

Algorithm 1 uses an abstract object named *batch* to keep track of received IDs. In a naive implementation, this could be instantiated with a simple list of IDs. However, when aggregating batches in the server side, it would imply building a single list of non-repeated objects. For ordering and inclusion tasks like that, it is very common to use hash-based objects for implementation (see Section 3.3.3.1). For this application, we rely on Bloom Filters to store all the IDs received in a single batch. The advantages are threefold: (1) the size of the batch is $O(1)$, for specific BF parameters set by the application, (2) the aggregation is reduced to bitwise OR operations, as shown in Algorithm 2, and (3) it hides the received IDs from the server, which must use brute-force to query inclusion for specific IDs. Equation 6.1 shows the error formula for BFs, which is normally set to be very small, and another important feature is shown in Equation 6.2, which is that BFs can easily compute their cardinality (the number of items they contain) by using a simple formula, where z represents the number of 0s in the BF.

$$P_\varepsilon \approx (1 - e^{-kn/m})^k \tag{6.1}$$

$$|S| = \frac{ln(z/m)}{k\ ln(1 - 1/m)} \tag{6.2}$$

UNIVERSIDAD DE MÁLAGA

Algorithm 2 IDs aggregation with Bloom Filters

$P_i \rightarrow Agg$: $BF(\{ID\}^i)$
if new query **then**
  Agg: $count = \bigoplus_j BF_j$
end if

Model description and simulations

We now introduce the theoretical model description and some simulations that allow to evaluate that the system is accurate. For the rest of the section, all assumptions are done within a specific location area. First, Equation 6.3 describes the probability that a device's ID is included in the count for that location. For that, it must be noticed that not only a device notifies its own ID, but other devices can notify it too if they have been in its area of coverage (this is captured by p_{other}). The term Δ_s represents the period of time that is taken into account to aggregate BFs when a query arrives, i.e., for a query arriving at time t, the interval is $(t - \Delta_s, t)$. On the other hand, T_r is the reporting period of devices. The function $min(\cdot, 1)$ allows to limit the theoretical analysis to the real case, i.e., the same ID will not appear more than once because of BF aggregation.

$$p_{dev} = min\left(\frac{\Delta_s}{T_r} + p_{other}, 1\right) \tag{6.3}$$

Another consideration is that IDs change from time to time, which can affect the accuracy of the count. Equation 6.4 shows the probability that an ID has changed, where T_{ID} represents the period of time where the same ID is sent by a device, before changing it to a different one.

$$p_{ID} = \frac{\Delta_s}{T_{ID}} \tag{6.4}$$

Finally, Equation 6.5 describes the estimated number of devices (therefore, people) that are simultaneously in a specific area. The term N is the real number of devices that are in that area, ε is a negligible error rate introduced by the cardinality estimation of BFs, and E_{out} is an increment of the estimated number of devices due to devices that are on surrounding areas close to those in the edge of the scoped area. Note that $E_{inf} \geq 1$.

$$|Dev| = (1 \pm \varepsilon)(N \cdot p_{dev}(1 + p_{ID}))E_{out} \tag{6.5}$$

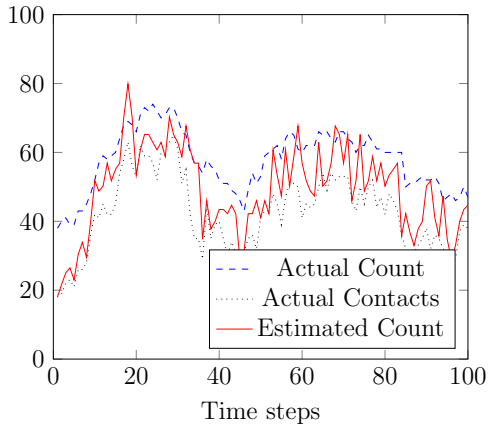
In addition to the system characterization, we have implemented a prototype that allow to obtain some insights about performance and accuracy. Even when it has some limitations and must be considered as approximate results, it is still good enough to understand the effects of the different parameters.

The prototype is based on a mobility scenario generator, and the implementation of the central server. Since testing data related to users in movement is quite difficult, we have used a mobility trace generator [205] which generates mobility data based on a mobility model and a simulation map. On the other hand, the server has been

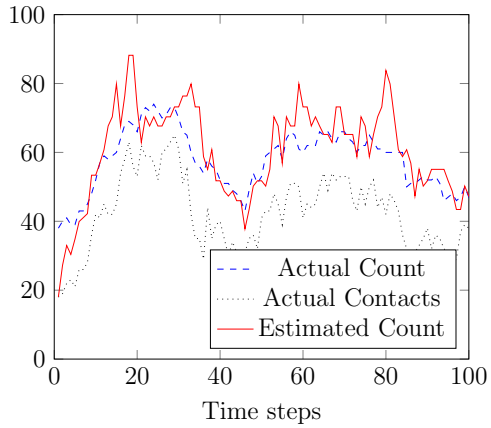
UNIVERSIDAD DE MÁLAGA

implemented in Python using Flask. It saves users' reports on a SqLite database and and retrieves them on demand to handle queries. To test the real-time behavior, we have developed a simulator in Python that takes as input the mobility traces generated previously generated and sends the BFs to the server in real-time as if it were the users sending them.

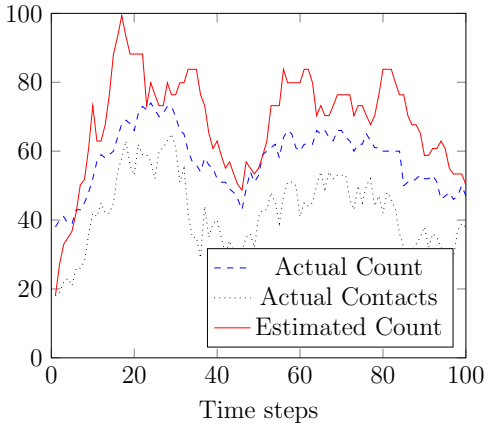
Regarding the limitations of the simulation, one is that it is based on mobility models, which are an approximation to real cases. The results introduced below are obtained with traces generated by a Random Way Point model. Second, in real scenarios, sampling rates and timers for users are independent of each others. However, the simulator employs discrete time steps to generate and send data, which can make data converge slower than in a real execution. Finally, for BLE, we have just considered a 1 meter range around each user where if others come inside is considered a contact.



(a) $\Delta_s = 2s$



(b) $\Delta_s = 5s$

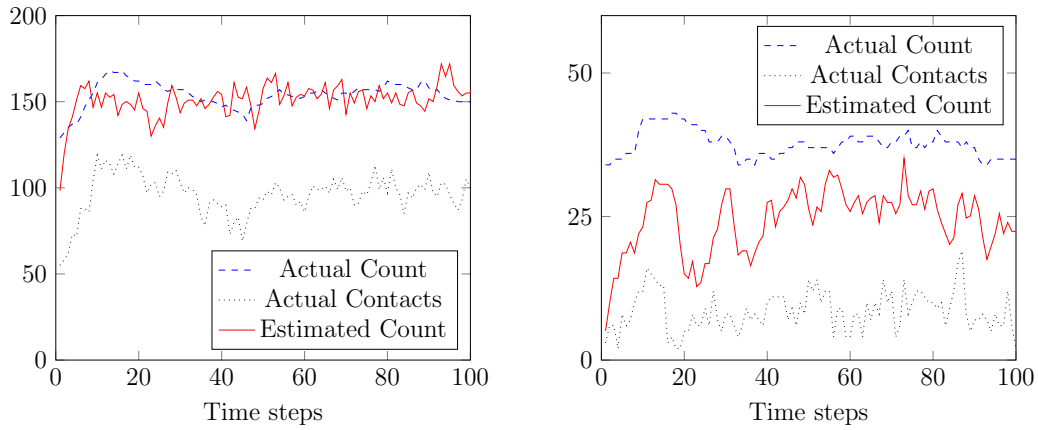


(c) $\Delta_s = 10s$

Figure 6.3: Simulation with 200 people in a 50x50 map with different Δ_s values, speed 1 m/s and 20x20 area queried.

There are two parameters that greatly affect the results obtained. First, temporal granularity Δ_s, which is the duration of a batch before sending it to the server (see Figure 6.3). Second, spatial device density, modeled with $d = devices/m^2$ (see Figure 6.4). Regarding the legends of the figures, *Actual Count* is the total number of devices in the queried area, *Actual Contacts* is the number of devices that are

UNIVERSIDAD DE MÁLAGA



(a) 200 devices ($d = 0.08\ dev/m^2$) (b) 50 devices ($d = 0.02\ dev/m^2$)

Figure 6.4: Simulation in a 50x50 map and 40x40 area queried, with different values for device density d.

close enough to others, and *Estimated Count* is the estimation made on the server. To measure the results, we have used the Relative Mean Error described in Equation 6.6, and the results obtained for each simulation are shown in Table 6.1.

$$RME = \frac{\sum_{i=1}^{n} \frac{|ActualCount_i - EstimatedCount_i|}{ActualCount_i}}{n} \tag{6.6}$$

Table 6.1: RME values obtained from the simulations

Simulation	Figure 6.3a	Figure 6.3b	Figure 6.3c	Figure 6.4a	Figure 6.4b
RME	0.1308	0.0620	0.1613	0.0422	0.3391

From Figure 6.3, it can be noticed that Figure 6.3b is the more accurate one with a lower RME, i.e., the estimated count closer to the actual count. The inaccuracy of the others cases is that, in the case of Figure 6.3a, the interval is so fast that devices that do not produce contacts are not considered. With respect to Figure 6.3c, the interval is so large that it takes into account devices that are not anymore in the scoped area.

On the other hand, Figure 6.4a shows that a scenario with high density of people lead to accurate results, while a decrement in density implies an underestimation of the count. This happens because not enough IDs are exchanged and the system is P2P. However, since this solution targets pandemic scenarios, the underestimation is not problematic if the density of people is low, because it implies that people will be able to move with a lower risk of exposure.

6.2.2 Privacy preserving space transformation of location data

We have introduced our design to achieve crowd counting using ephemeral IDs exchanges in a P2P scenario, which provides privacy regarding user identification,

UNIVERSIDAD DE MÁLAGA

thanks to generating the IDs pseudo-randomly and encapsulating them in BFs. However, it rests to protect users' location, because the application server receives the exact location of users periodically, which straightforwardly leads to tracking users' path of movement.

There already exists many works in the literature that target the problem of location privacy in LBS [206]. Even when there is not a clear consensus on a best strategy, mainly because of the different trade-offs involved, one of the most extended is K-Anonymity, usually implemented using cloaking techniques. The objective is to generalize the area where the user is located until there are at least other K users in the same area, which allows this set of $K+1$ users to blur between them. K-Anonymity has been proposed for different architectures, such as using a single trusted server [207], multi-server setting [208], or even P2P scenarios [209]. However, K-Anonymity (and other obfuscation schemes such as Differential Privacy) does not fit well with our crowd counting solution, because privacy is directly related to location generalization. Therefore, increasing privacy implies a reduction in accuracy.

Our solution for privacy relies on a Space Transformation problem instantiated with trapdoor functions, i.e., there exists a function $f_\tau : L \to M$, and its inverse $f_\tau^{-1} : M \to L$, where L is the space of locations, and M is the space of pseudo-randomly mapped values. The trapdoor τ restricts the usage of the function to only those in possession of τ.

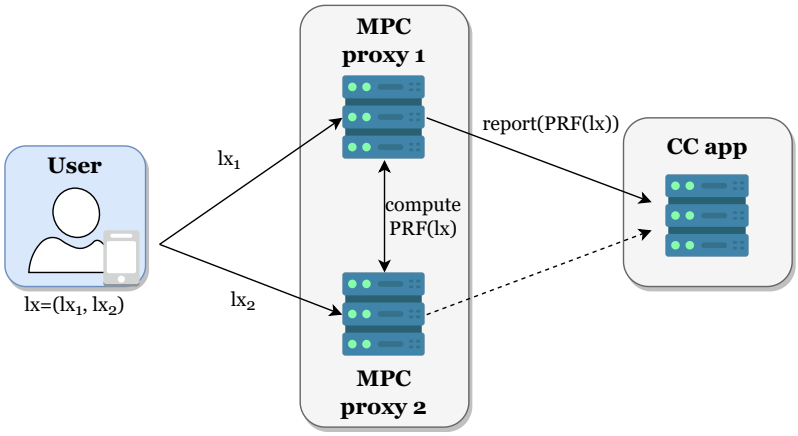


Figure 6.5: Notification of location data with MPC proxy.

A naive way of implementing this is by using a trusted proxy, which is a resource sometimes proposed in the literature, e.g., for Oblivious DNS queries [210]. In our application, the trusted proxy (the only one who knows τ) applies the mapping function to location data, before delivering the mapped reports to the application server. This can be easily instantiated using a symmetric encryption scheme where τ is the key. However, it can be noticed that if f_τ is not a structure-preserving function, closely located reports cannot be aggregated because a two pairs (l_x, l_y), $(l_x+\Delta_x, l_y+\Delta_y)$ which are close will be mapped to two different values (l_{x_a}, l_{y_a}), (l_{x_b}, l_{y_b}) where the distance between l_{x_a} and l_{x_b} cannot be computed inside the mapped space. The solution is to apply a space generalization technique based on a grid with cells of a specific dimension, where all reports in the same grid are notified with the same location, e.g., the center of the cell. However, despite this reduces accuracy, note

UNIVERSIDAD DE MÁLAGA

that it is independent of privacy (on the contrary to K-Anonymity), and will be parametrized with a cell area value that depends on the computational power of the service.

Despite this solution is functional, it just translates the problem of trusting the application data to trusting the proxy. For that reason, we propose a final solution based on MPC, which distributes the functionality f_τ to a committee of parties that apply threshold cryptography (see Figure 6.5). Thanks to this, no single party can see the plaintexts of location data, but instead, the distributed proxy receives pairs of shares $([\![l_x]\!], [\![l_y]\!])$, and applies the trapdoor function through an MPC protocol such as the distributed block ciphers introduced in Section 3.2.2.1.

Regarding trust, as discussed in Chapter 3, if the MPC committee is instantiated by the application provider itself, privacy is guaranteed against external adversaries, but not against the application provider. In order to avoid trust in the application provider, the distributed trapdoor function can be externalized to an independent provider of MPC services such as the framework for blockchain proposed in Section 4.4. The nature of blockchain allows to randomize the selection of the nodes that perform the MPC, which would receive location data through a smart contract and will output the result to the application server.

It also can be noticed that, since the encryption process is distributed and location has to be protected, the users must send shares of the location values instead of plaintexts, which implies additional costs on constrained devices. In addition, reports are sent periodically. We notice that, in order to mitigate the increment in communication resources and independent connections, we could use our solution for OSRE, proposed in Chapter 5. This allows the device to send a single ciphertext per location data, and the MPC committee will be able to decrypt to shares directly, thus allowing to compute the trapdoor function on secret shared data.

Finally, we put the focus on specific performance of the distributed block cipher. First note that location data can be easily represented by decimal degrees, where the latitude takes values in the interval $(-90, +90)$, and the longitude in $(-180, +180)$. Regarding decimals, they affect the precision of the measures. For LBS that need to identify individual streets or large buildings, using only 4 decimals may be enough. Notice that this is true since our application uses a map generalization technique, and identifying crowded streets or buildings can be enough for our purposes. Therefore, to achieve a compact representation of location in computers, 4 bytes are enough per measure, 2 for latitude and 2 for longitude (1 bit for the $\pm$ sign, 1 byte of the integer part, and 14 bits for integer representation of the decimals, which take values between 0 and 9999). This representation fits well with the MPC block ciphers proposed in the most efficient works of Section 3.2.2.1, e.g., those based on SPDZ [77, 79] require a GF(2^{40}) field, and those from algebraic approaches [80] are typically instantiated with 128/256-bit prime field $\mathbb{F}_p$. In fact, both the latitude and the longitude can be encoded in a single element and therefore encrypted with a single operation of the MPC block cipher. Regarding performance, the fastest protocol [79] achieves 1.5 ms per block processed, which would lead to an upper bound of 666.6 location queries processed per second. However, the benchmark is proposed on a LAN, which is not realistic for most real deployments, and overall if we use the blockchain as the trustless environment to instantiate the MPC with

UNIVERSIDAD DE MÁLAGA

independent parties. Then, taking values from the WAN setting in Grassi et al. [80], the processing capability of the system would be between 2 and 5 queries processed per second. This is a key aspect to consider when deploying the application, because it directly affects the reporting period of the devices.

If we consider that these numbers for the WAN setting are not very realistic for a deployment with a high number of reports, we identify three possible ways of solving that drawback:

1. Deploy the MPC protocol in a LAN setting, at the expense of relaxing the trust model. This is mainly because deploying all the nodes involved in the MPC in the same network requires a single coordinator that deploys them, and therefore the model achieves security against external adversaries but not against a malicious coordinator.

2. Use a replicated MPC instance for load balancing, where instead of having a single MPC committee, there are N committees that distribute all the users' requests. If we assume that committee C_i has a request handling rate r_i, the overall rate of the system will be $R = \sum_{i=1}^{N} r_i$. The major difficulty here is that the secret key k to perform the space mapping must be the same in all the committees and this implies some sort of coordination in the generation phase of the key and in the refreshing phases, where key k is replaced by key k'. There can be different strategies for that, from having one committee generating the key and distributing it with the others (e.g., in a secret shared form) to deploying an MPC protocol that implies all the N committees C_i in the generation phase.

3. Finally, the last solution targets MPC from the theoretical approach. The problem of having a low request handling rate is mainly due to the high latency introduced by the WAN setting. Since generic-purpose MPC protocols are quite intensive in communication resources (either the overall amount of data to be sent or the number of communication rounds), a WAN setting has a high impact. Because of the lower bounds that exist in the literature of MPC, efforts are being put in finding new assumptions for MPC such that they allow to reduce the impact of the communication cost (but at the expense of increasing the computational cost) as in the work of Dao et al. [211]. Therefore, for WAN settings, the request handling rate could be improved with this new type of MPC protocols, as long as the nodes in the committee are able to handle the computational cost.

6.3 Private and accountable transactions in cryptocurrency

In this section, we introduce an application that relies on blockchain and threshold cryptography to achieve private payments with selective disclosure. This enables users to control fine grained privacy levels and enhances usability aspects.

Blockchain technologies have evolved a lot from their conception and many use cases such as NFTs, e-voting, or traceable supply chain have been proposed. However,

UNIVERSIDAD DE MÁLAGA

digital payments by cryptocurrency, which is the original use case for which Bitcoin was designed, still remains as the most adopted use case.

Despite the new range of use cases, blockchain presents a huge limitation, as mentioned in Section 2.2, which is the need of having all information publicly available in order to enable public verifiability. As introduced in Section 4.1, the benefits between MPC and blockchain are bidirectional, since both technologies can offer advantages to the other in some of their limitations. In this section, we focus on solutions that enable private payments in blockchain, not only built on MPC, but other privacy-enhancing technologies. The literature has provided two main approaches for this: blockchain mixers and private currencies. While mixers are based on an intermediary service that provides privacy, private coins typically imply a redesign of the blockchain protocol or the way that cryptocurrencies are transacted.

Regarding the concept of privacy, it is understood by most people as a right, and there are regulations that enforces that (an example is the GDPR data minimization principle[4]). However, there are applications where full privacy can make it much easier for criminal behaviors, such as money laundering [212, Section 1] in the case of cryptocurrencies. The discussion about privacy and control does not have an unanimous position, and finding the proper level of data disclosure is not easy. What is somehow clear is that full transparency by default (as it happens in standard and widely adopted blockchains such as Bitcoin or Ethereum) is not realistic or acceptable, and not only because it goes against the right to privacy, but also because it makes it almost impossible to adopt some use cases in the blockchain ecosystem. A clear example is blind auctions, where providing a public record of bids clearly biases the outcome. In the payment use case, public transactions include the pair of addresses involved and the amount sent. Even if the pseudonymous nature of identities in blockchain adds some relative level of protection, there are use cases where this protection is automatically removed. For example, when a customer pays for goods or services with cryptocurrency, the seller can map the customer's real identity with his wallet address, and therefore learn his entire transaction history. The same happens in the other direction with respect to the seller's wallet address. While a typical recommendation to protect the identity in blockchain payments is not to reuse the same wallet address, ephemeral addresses can be correlated if they have a common source of funds, in addition to the lack of usability that this implies. We notice that blockchain mixers help in avoiding this exposure of data.

However, there is a general perception that the blockchain ecosystem is not secure enough for payments. Some works such as Feng et al. [213] highlights the need for future research to focus on the legal traceability and accountability of transactions. Some efforts have already been made to enable compliant payments in cryptocurrency environments, such as the use of deny lists and Know Your Customer (KYC) procedures. For example, the Treasury's Office of Foreign Assets Control (OFAC) maintains a list of sanctioned addresses for various assets belonging to entities on economic/trade embargo lists[5]. However, these procedures for compliance are more difficult to achieve when it comes to privacy coins, and this has even lead to the sanction of privacy services such as Tornado Cash by the OFAC for being used for

[4] https://gdpr-info.eu/art-5-gdpr/

[5] https://github.com/0xB10C/ofac-sanctioned-digital-currency-addresses

UNIVERSIDAD DE MÁLAGA

money laundering[6].

In view of the foregoing, what is clear is that, in scenarios with sensitive data, privacy must be ensured by default, and additional techniques must be used to achieve some level of transparency when necessary. In fact, it is key to notice that this is a one-way problem, because a system that sets full privacy by default can allow different levels of data disclosure, reaching full transparency in the most extreme case. However, once full transparency is set by default, it is impossible to move towards privacy, as graphically shown in Figure 6.6.

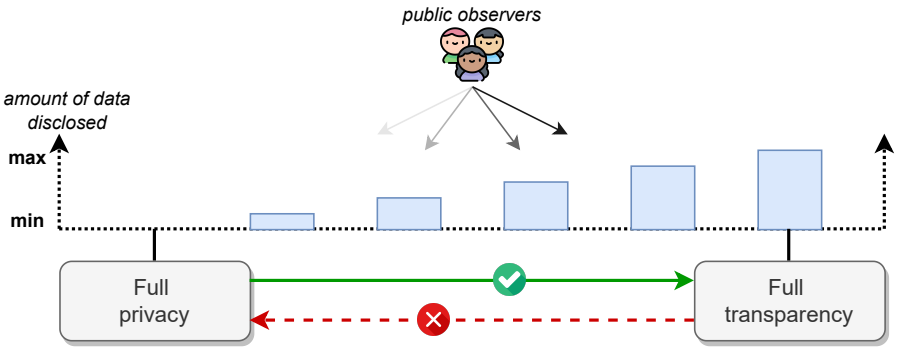


Figure 6.6: The relation between privacy and transparency.

In this section, we provide a solution for a compliant Bitcoin mixer, where private-by-default payment information can be selectively disclosed thanks to reversible accountability. We propose a general design and specific constructions based on trapdoor functions and NI-ZKP.

6.3.1 Technical background

6.3.1.1 Blockchain mixers

Blockchain mixers allow payers to send payments to recipients without exposing the link, i.e., with privacy, and they achieve that by anonymity and unlinkability. *Anonymity* guarantees that a subject is unidentifiable within a set of peers in a system, referred to as the *anonymity set*. On the other hand, *unlinkability* ensures that multiple items within an anonymous system are equally unrelated based on prior knowledge. Based on these notions, a formal definition for blockchain mixers is introduced in Definition 15.

Definition 15. *A **blockchain mixer** is a service that enables anonymous payments by unlinking transactions. More precisely, the anonymity sets for the payer and the recipient are the set of input and output transactions respectively, i.e.,* $\{tx_{in_1}, ..., tx_{in_n}\}$ *and* $\{tx_{out_1}, ..., tx_{out_n}\}$*. The mixer ensures unlinkability by making sure that any input transaction* tx_{IN_i} *appears to be equally related to any* tx_{OUT_j}*.*

A centralized mixer owns a wallet with address m. It accepts payments from input addresses until the anonymity set reaches a predefined threshold, i.e., $I = \{in_1, ..., in_n\}$. Additionally, it receives the output set $O = \{out_1, ..., out_n\}$ through

[6] https://home.treasury.gov/news/press-releases/jy0916

UNIVERSIDAD DE MÁLAGA

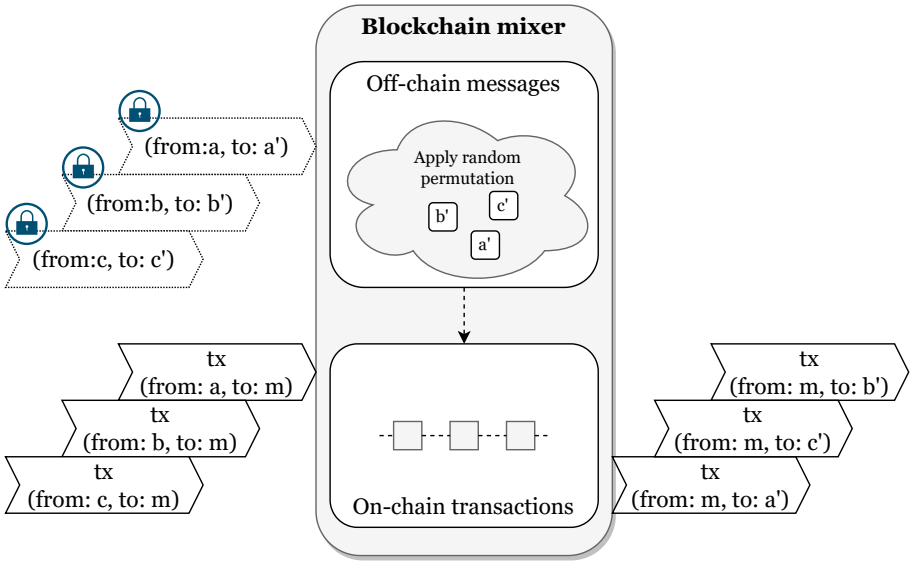


Figure 6.7: Graphical description of a centralized blockchain mixer.

confidential off-chain channels, which is mixed by a private permutation π. Each incoming payment is sent from m to each address in $\pi(O)$, as shown in Figure 6.7. However, to avoid over-reliance in the mixer, some works propose decentralized mixers, as introduced in Section 6.3.1.3.

6.3.1.2 Mixnets

A mixing network (mixnet) [214] is composed of a set of nodes $\{M_1, ..., M_N\}$ that, given a set of ordered input messages $\{\langle x_1 \rangle, ..., \langle x_n \rangle\}$, where $\langle \cdot \rangle$ denotes some sort of confidentiality protection such as encryption, outputs the set of decrypted and permuted messages $\{x_{\pi(1)}, ..., x_{\pi(n)}\}$, where π is a private permutation. The mixing process is performed such that no single M_i can correlate a message from the output set with one from the input set, and can therefore be generalized as an MPC functionality.

Decryption-based mixnet

A decryption-based mixnet is constructed upon the concept of *onion encryption* [215], where each server in the mixnet, denoted as M_i, owns an asymmetric key pair (sk_{M_i}, pk_{M_i}). Moreover, each sender S_j encrypts its message sequentially using the public keys of all servers (Equation 6.7) before sending $\langle x_j \rangle$ to M_1.

$$\langle x_j \rangle = Enc_{pk_{M_1}}(Enc_{pk_{M_2}}(...Enc_{pk_{M_N}}(x_j))) \tag{6.7}$$

When the mixnet receives n encrypted messages, each server M_i sequentially decrypts its layer, applies a private permutation π_{M_i} to the message list, and sends the outcome to M_{i+1}. At the end, M_N outputs the permuted plaintexts, i.e., $\pi_{M_1}(\pi_{M_2}(...\pi_{M_N}(\{x_1, ..., x_n\})))$, where no server can link x_i to the original sender.

Secret shared-based mixnet

A secret-shared-based mixnet is built on the concept of *secret-shared shuffle*, introduced in Chase et al. [216] as a 2-party protocol and extended in Eskandarian and

UNIVERSIDAD DE MÁLAGA

Boneh [217] for the n-party case. Each sender shares its input x in secret-shared form $[\![x]\!]$, i.e., each server receives a share of x. When the servers receive n messages, the input to the mixer has the form $[\![\mathbf{x}]\!] = ([\![x_1]\!], [\![x_2]\!], ..., [\![x_n]\!])$. In addition, each server M_i holds a private permutation π_i. In the end, after applying the permutations using an MPC protocol, each server holds a secret-shared vector $[\![\mathbf{y}]\!]$ that satisfies Equation 6.8, where $x_{\pi_{[1:N]}(i)}$ is the position occupied by x determined by the sequential application of the permutations $\pi_1, \pi_2, ..., \pi_N$.

$$\mathbf{y} = \pi_1(\pi_2(...\pi_N(\mathbf{x}))) = (x_{\pi_{[1:N]}(1)}, x_{\pi_{[1:N]}(2)}, ..., x_{\pi_{[1:N]}(n)}) \tag{6.8}$$

The core block for secret-shared shuffle is a 2-party *share translation protocol*, where P_1 and P_2 take no inputs, but at the end of the protocol P_1 holds random vectors $\mathbf{a}, \mathbf{b}$, and P_2 holds a random permutation π and $\mathbf{\Delta} = \pi(\mathbf{a}) - \mathbf{b}$. We refer to the paper of Chase et al. for more details.

6.3.1.3 Decentralized mixers

Table 6.2: Related work comparison. * Unless the whole mixnet cheats. ** As long as 2/3 of the nodes are honest. † If *pk* of a Zether account can be linked to the ETH address, then anonymity is lost. ‡ Only briefly mentioned. **n.a.** Not applies.

	Anonymity (with respect to)					Theft prevention	Accountability	Availability		Compliance	# Txs	Bitcoin
	Payer	Recipient	Outsider	Users	Mixer			Payer	Mixer			
Mixcoin	✗	✓	✓	n.a.	✗	✗	✓	✓	✗	✗	2	✓
Blindcoin	✗	✓	✓	n.a.	✓	✗	✓	✓	✗	✗	2	✓
TumbleBit	✓	✓	✓	n.a.	✓	✓	✗	✓	✗	✗	4	✓
CoinJoin	✗	✓	✓	✗	n.a.	✓	✗	✗	n.a.	✗	1	✓
CoinShuffle	✗	✓	✓	✓*	n.a.	✓	✗	✗	n.a.	✗	1	✓
CoinParty	✗	✓	✓	n.a.	✓*	✓**	✗	✓	✓**	✗	2	✓
Möbius	✓	✗	✓	n.a.	✓	✓	✗	✓	✓	✗	2	✗
Tornado C.	✓	✓	✓	n.a.	✓	✓	✗	✓	✓	✓	2	✗
Zether	✓†	✓†	✓	n.a.	✓	✓	✗	✓	✓	✗	1	✗
Haze	✓	✓	✓	n.a.	✓	✓	✗	✓	✓	✓‡	2	✗
Ours	✗	✓	✓	n.a.	✓*	✓**	✗	✓	✓**	✓	2	✓

This section discusses the primary research in the field of blockchain mixers, which is summarized in Table 6.2 together with our solution, introduced in Section 6.3.2. We notice that, while the older solutions worked in Bitcoin, most newer approaches do not because they rely on smart contracts. However, since Bitcoin remains as the most important blockchain for cryptocurrency transactions, our solution focuses on Bitcoin to enhance the options for private transactions on it.

Some works, such as Mixcoin [218] and Blindcoin [219], proposed centralized solutions whithout theft prevention, i.e., malicious mixers can decide not sending the funds to their recipients. In addition, Mixcoin's payments are not anonymous with respect to the mixer. TumbleBit [220] (also centralized) uses multisig addresses to escrow deposits and provide theft prevention, where more than one signature are needed to complete a transaction.

A different line of research studied decentralized mixers, which assumes a set of distrustful parties to perform the mix of transactions with theft prevention. In CoinJoin [221] and CoinShuffle [222], the payers partake in a distributed protocol that uses a permutation mixnet, therefore ensuring anonymous payments with respect to the mixing parties. To tackle the overhead on the payers, CoinParty [223]

UNIVERSIDAD DE MÁLAGA

proposes a decentralized outsourced mixer, where the payers send their funds to an address generated from a secret-shared key. Since threshold signatures are needed to send transactions from the mixer, thefts are prevented as long as enough mixer servers remain honest. CoinParty also relies on a permutation mixnet to perform the mix.

Most recent works moved from the Bitcoin world and have benefited from smart contract capabilities to achieve decentralized secure mixers. Möbius [224] is based on stealth addresses and ring signatures to ensure anonymity, which is achieved with respect to the payer instead of the recipient, differently from the rest of solutions. Regarding Tornado Cash [59], it relies on NI-ZKPs to maintain on-chain privacy and enable correctness verification of blinded notes and nullifiers. On the other hand, Zether [60] enables confidential balances and transactions using Additive Homomorphic Encryption and NI-ZKPs for encryption correctness verification. It also facilitates anonymous transfers by using anonymity sets.

We note that, despite of the different aspects addressed in blockchain mixers, compliance has only been addressed very recently in Tornado Cash and Haze [225]. However, Haze mostly prioritizes compliance before withdraws and Tornado Cash assumes the payer to be the one deciding to generate the compliance proof and keeping it. In addition, almost all solutions focus on providing a full-private setting, but do not facilitate a selective disclosure of data.

6.3.2 A decentralized Bitcoin mixer with selective transparency

This section introduces a solution to achieve a Bitcoin payment system with full privacy by default, which can be configured by the users to achieve tailored transparency. As discussed in the introduction of this section, the relation between privacy and transparency is one-way, i.e., different levels of transparency can be achieved from full privacy by default, but no privacy can be achieved from full transparency by default. To the date, public blockchains cannot regulate these levels of transparency, except by requiring node providers not to reveal certain subsets of data. However, this is typically related to law enforcement and blocking malicious services, and has nothing to do with user privacy.

In the rest of this section, we introduce a CoinParty-based mixer [223] that is Bitcoin-compatible, together with an additional layer of transparency disclosure control that provides selective disclosure capabilities of payment data and accountability of queries. Such a layer can be considered similar to a Policy Enforcement Point (PEP), which ensures access control (to payment data) only to those authorized by the payer. There is an additional particularity with respect to the standard use of CoinParty, which is that the mixer must somehow verify that privacy can be selectively revoked, but without being able to learn the actual payment information. To achieve this, our solution relies on ZKPs.

6.3.2.1 Roles and protocol design

The different roles involved in the protocol and the interactions between them are presented in Figure 6.8, where *dashed* arrows represent on-chain transactions and

UNIVERSIDAD DE MÁLAGA

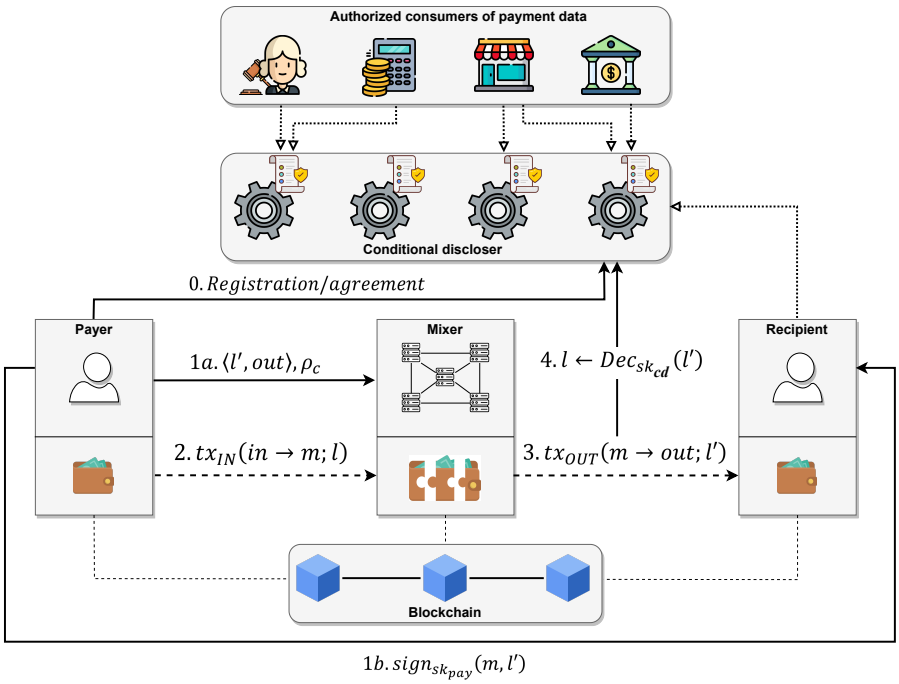


Figure 6.8: Roles and interactions for a mixer with private and relinkable payments.

solid arrows represent off-chain communication. In addition, the wallets interact with the blockchain and the mixer wallet employs a threshold mechanism. From all the roles, the **payer**, the **recipient**, and the **mixer** are those directly related to the payment. The payer sends some amount of BTC to the recipient through the mixer, which adds the privacy layer. These are the roles that already existed in CoinParty [223]. However, we also add the **conditional disclosers** and the **authorized consumers** of payment data. The former are those in charge of providing selective disclosure of payment data, therefore they are trusted to the payer. The latter, on the other hand, are those willing to learn payment data from a particular payer or recipient. In this work we focus on providing an agnostic technical tool that enables selective disclosure, but it can be applied differently regarding the use case. For example, enabling an accountant to relink the payments is decided solely by the payer, therefore the accountant is provided access by the allowlist of the conditional discloser trusted by the payer. Another example is to let the mixer having a list of accepted conditional disclosers, therefore refusing to process all payments which are not verifiable to be relinkable by at least one of those disclosers, which may be deployed by regulatory entities.

Regarding the technical aspects, as in CoinParty, we assume a decentralized mixer (a set of parties $\mathbf{M} = \{M_1, ..., M_N\}$) with a wallet address m and a mixnet to shuffle outgoing addresses. For simplicity, we will assume for the rest of the paper that mixer and mixnet parties are the same, i.e., $\mathbf{M}$, but they can be decoupled. An incoming transaction tx_{IN} from a payer has input address in and output address m. In addition, the address of the recipient out is sent in encrypted form to the mixnet. The mixer wallet private key is secret shared with all the servers, thus avoiding to spend funds unless the majority agrees. Then, after permuting the output addresses sent by the payers to the mixnet, one outgoing transaction tx_{OUT} is sent by the mixer from address m to each address out. For the selective disclosure of payment

UNIVERSIDAD DE MÁLAGA

information, this solution relies on reversible unlinkability, which is based on the concept of trapdoor-linkable transactions, introduced in Definition 16.

Definition 16 (Tapdoor-linkable transactions)**.** *Two transactions tx_{IN} and tx_{OUT} are linked if tx_{IN} includes a parameter l and tx_{OUT} includes a parameter l' such that $l' = f(l)$, where f is a deterministic function. We say two trapdoor-linkable transactions are related by parameters l, l' generated by a trapdoor function f_τ, i.e., a one way function. Without knowledge of the trapdoor τ, the view of (l, l') is indistinguishable from (l, r), where r is a randomly sampled value.*

The linkability parameter l is included in tx_{IN} by the payer (Figure 6.8, step 2) and l' is included in tx_{OUT} by the mixer (step 3). Since we target trapdoor-linkable transactions, we define $l' = Enc_{pk_{cd}}(l)$, where (pk_{cd}, sk_{cd}) is the asymmetric key pair generated by the conditional discloser and the private key works as the trapdoor. It is important that the pair (l, l') does not expose any additional information than (l, r) without knowledge of sk_{cd}, being r a random value.

To let the mixer learning l' without knowing which payer it belongs to, it is included into the ciphertext that inputs the mixnet, i.e., $\langle l', out \rangle$ (step 1a). In addition, the payer can send an off-chain signed message to the recipient (step 1b) to confirm the address from where the mixer is supposed to send tx_{OUT}.

We note that trust in the conditional disclosers can be easily minimized by accountable decryption based on threshold cryptography (see Section 3.2.2) or escrowed decryption [226]. These solutions allow to split private keys into shares, each one held by a different discloser. This implies that more than one discloser must agree in retrieving the link, thus minimizing the probability that a malicious discloser exposes data when a non-authorized consumer queries for it.

In addition to relinkability of transactions, the conditional disclosers provide accountability of the queries received to obtain payment data. In a standard public blockchain, all the history of transactions can be queried through a node provider, and the latter may not even request the identity of the requester. However, the conditional disclosers can be equipped with an authentication mechanism such that they log the identity of each requester for a payment and provide such log to the payer.

It remains to introduce how the mixer verifies compliance without learning the payment link, otherwise the payer could just provide dummy (l, l') values such that no transparency can be achieved at all. For that, we introduce the concept of proof of compliance in Definition 17.

Definition 17 (Proof of compliance)**.** *A proof of compliance ρ_c ensures a verifier, in zero-knowledge, that given a first parameter l, there exist a second parameter l' generated with a correct trapdoor τ, such that only another designated verifier (who knows τ) can relink both parameters.*

While the conditional discloser is the verifier of the relation (l, l'), the mixer is the verifier of the proof ρ_c, which is computed by the payer and included in the message in step 1a. More specifically, the proof ρ_c is a ZKP that states the following: (1) the private value l' is equal to $Enc_{pk_{cd}}(l)$ and (2) l' is included inside $\langle l', out \rangle$. To verify the ZKP, the mixer uses $\langle l', out \rangle$ and pk_{cd} as public inputs, thus being ensured that

UNIVERSIDAD DE MÁLAGA

the owner of pk_{cd} has the ability to link l and an unknown l', which has not been mixed by the mixnet yet. Only when a sufficiently large set of incoming transactions are ensured to be compliant, their off-chain messages $\langle l', out \rangle$ are processed by the mixnet, thus returning unlinkable pairs (l', out) to build the outgoing transactions.

Finally, once a round of mixing has been completed and outgoing payments are sent, to relink a transaction (after receiving a query from an authorized consumer), the conditional discloser just needs to decrypt l' (step 4) and look for the incoming transactions of that round to see which one holds l. However, there is an important remark in this final step related to how the parameters l, l' are implemented. Notice that, since $l' \leftarrow Enc_{pk_{cd}}(l)$, learning l from l' is as straightforward as decrypting l'. Then, when there are two lists of transactions $\{tx_{IN}(l_1), ..., tx_{IN}(l_m)\}$ and $\{tx_{OUT}(l'_1), ..., tx_{OUT}(l'_m)\}$, given a query on any $tx_{OUT}(l'_i)$ directly leads to its pair $tx_{IN}(l_i)$. However, if the query is done on an input transaction, reaching l' from l implies opening $m/2$ outgoing transactions in average to find which l' is decrypted to l. This is not only an efficiency problem, but one that exposes the relation of other pairs of transactions in that mixing round. Therefore, the naive form of this design suits well for one-way queries only.

6.3.2.2 Security requirements

An important aspect must be considered with respect the mixer model. As it has been discussed through the rest of this thesis, just assuming that there is a set of distrustful parties is half of the way. In the case of CoinParty, threshold signatures makes it more difficult for an external adversary to manipulate signing keys' usage and steal funds. However, if a single service provider deploys the mixing nodes by its own, the trust domain remains the same for all nodes. As previously discussed, this characteristic can or cannot be an issue for users depending on their relation with the service provider and the amount of trust they want to deposit on it. For example, trust can be minimized if the code used by the different parties is correctly audited or enforced through verifiable mechanisms such as Trusted Execution Environments. However, if a truly decentralized environment is desired, where even the service provider does not have power to steal funds, additional (more complex) mechanisms must be used. One approach for this would be to rely on blockchain capabilities, as it happens in our proposed framework in Section 4.4, where there exist a pool of nodes for MPC from where subsets can be selected for different tasks based on different strategies, such as random selection. This aspect of secure and not biased selection of nodes is an interesting and hot topic in the field of decentralized environments, which is addressed in some of the most recent proposals for MPC models, as introduced in Section 2.1.3.4. However, we notice that our design targets Bitcoin, which does not facilitate these complex decentralized procedures. Therefore, implementing such a solution would need some cross-chain capabilities where nodes selection for certain mixing rounds is done, for example in Ethereum (through a process of node registration in a pool through staking, etc.), and the mixer's incoming and outgoing transactions work in Bitcoin.

Another important characteristic that this decentralized mixer must fulfill in our proposal, as long as compliance is desired to be ensured for all transactions, is that it verifies each incoming transaction without learning the payment link, i.e., it checks if the transaction is correctly linked with at least one of the conditional disclosers

UNIVERSIDAD DE MÁLAGA

available by verifying the ZKP of compliance. Then, the mixer is assumed to be semi-honest or malicious. Notice that if verification does not happen (the lazy mixer case), then users could link transactions with a key pair that does not relate with any of the disclosers. Switching on or off the possibility for a user to use the mixer without compliance proofs is an implementation decision for the service provider. Also remark that, if compliance is mandatory, then the client is assumed to be malicious because he may try to cheat on the mixer with the proof.

On the other hand, regarding the encryption mechanism, it is mandatory to be IND-CPA with randomness hiding, such as ElGamal, thus protecting against anyone computing l' from l and pk_{cd}, which are public values. Since authorized consumers cannot verify the relation (l, l'), conditional disclosers may be requested to provide a ZKP that ensures l' decrypts to l when using sk_{cd}, therefore avoiding malicious disclosers.

Finally, a conditional discloser is assumed to be trusted for privacy because it can relink all the payments linked with its public key. In addition, authorized consumers trust the conditional discloser to answer their queries, if they are allowed. Once again, as it happens with the decentralized mixer, threshold cryptography (threshold decryption in this case) can help to reduce trust. Therefore, it may be required for some conditional disclosers that they are a set of nodes that must agree in opening a payment when some consumer queries for it. This would make it more difficult for a non-authorized consumer to access payment information, and in addition, makes it more easy to provide accountability of opening requests.

6.3.3 Implementation of proofs and evaluation

6.3.3.1 Design of proofs

This section provides the specific design for the NI-ZKPs used by the payers to guarantee compliance of transactions sent to the mixer. As introduced in Section 6.3.1.2, we rely on two main types of mixnets, decryption-based and secret sharing-based, and therefore we propose proofs for both of them. Notice that the proof is directly related with the mixnet because the payer must relate information that is public in the incoming transaction with information that must be mixed first, and that will appear in the outgoing transaction. Therefore, the mixer must be able to verify the proof without learning anything else than what is supposed to learn by the decentralized mixing process.

Decryption-based mixnet

In this setting, the first server of the mixnet is in charge of verifying the proof ρ_c, which must confirm that both (l, l') are linked and that l' is included in the ciphertext x. The computation that must be verified inside the proof is presented in ZKP 1, where private inputs are only known by the payer, i.e., the prover.

The reason for the first server to be the verifier is because it is the only one that sees the whole chain of encryptions of the off-chain message sent by the payer, i.e., $x = Enc_{[pk_{M_1}:pk_{M_N}]}(x_0)$. Note that x_0 is the plaintext version of x and does not only contains the recipient address *out*, but the linkage parameter l' that must be included in that outgoing transaction. On the other hand, r_i is the randomness for the i^{th}

UNIVERSIDAD DE MÁLAGA

ZKP 1: Decryption mixnet proof

Public input: l, x, pk_{cd}, $\{pk_{M_i}\}_{i\in\{1,...,N\}}$
Private input: l', out, $\{r_i\}_{i\in\{1,...,N\}}$, r_l
Computation:
1. $l' = Enc_{pk_{cd}}(l; r_l)$
2. $x_0 = (l', out)$
3. For all $i \in \{1, ..., N\}$: $x_i = Enc_{pk_{M_i}}(x_{i-1}; r_{i-1})$
4. $x == x_N$

encryption layer, needed for IND-CPA. The proof computes (in zero-knowledge) the encryption of l' and the sequence of encryptions to achieve x. The mixer verifies the correctness of the public keys used for the encryption, i.e., pk_{cd} and $\{pk_{M_i}\}$ are public inputs. Consequently, the mixer is guaranteed that the parameters have not only been produced correctly, but can also be decrypted by the designated party using pk_{cd}. It is noteworthy that this solution involves computing $n + 1$ public key encryptions within the NI-ZKP, which can be quite costly for a large n.

Avoiding a single verifier. As previously stated, the proof is verified by the first server, whose ciphertext includes the encryption with all the layers. This server holds the potential to collude with the payer and assert to the other servers that ρ_c has been successfully verified, even if it has not. While it may be tempting to have all servers verify ρ_c using the public parameter x, this approach incurs the risk of a linkability attack. Specifically, each server M_i could use the public keys from the previous servers to sequentially re-encrypt the ciphertext obtained from M_{i-1}, thereby obtaining x and bypassing the permutations.

We slightly sketch a solution based on recursive NI-ZKP, where each mixnet server M_{i-1} proves to M_i that ρ_c has been properly verified. It would be composed of two main verifications. First, the proof must ensure that $x_{i-1} = Enc_{pk_{M_{i-1}}}(x_i; r_{i-1})$, i.e., the ciphertext x_i sent from M_{i-1} to M_i follows the proper layered encryption. Second, $true \leftarrow Verify(\rho_{i-1}, x_{i-1})$ ensures that the proof received by M_{i-1} on ciphertext x_{i-1} verified correctly, and here is where recursion happens. Note that in proof ρ_i, which is sent from M_{i-1} to M_i, the value x_i is public, but x_{i-1}, r_{i-1}, and ρ_{i-1} are not. In the side of the last server, the proof will capture the whole verification process, i.e., $\rho_i(\rho_{i-1}(...\rho_1))$.

Secret Sharing-based mixnet

In this approach, the verification is not performed against a ciphertext, but on a secret shared value, i.e., opening $[\![x]\!]$ allows the retrieval of (l', out). This method introduces a problem that does not exist in the decryption-based scenario: even if the payer proves the validity of a share to the corresponding mixnet server, the payer can cheat by sending shares of different secrets to the other servers, rendering the proof useless and altering the real value of x. A simple but inefficient solution would imply sending independent proofs ${\rho_c}^{(i)}$ to the servers, where ${\rho_c}^{(i)}$ verifies the share $[\![x]\!]_i$ for server M_i. However, this increases computation and communication overheads $\times N$. Therefore, we propose a solution requiring a single proof ρ_c which is

UNIVERSIDAD DE MÁLAGA

ZKP 2: Additive SS mixnet proof

Public inputs: $MAC, (g, h), l, pk_{cd}$
Private inputs: $\{[\![x]\!]_i\}_{i\in\{1,...,N\}}$, $\{[\![r]\!]_i\}_{i\in\{1,...,N\}}$, l', out, r_l
Computation:
1. $x = \sum_{i=1}^{N} [\![x]\!]_i$
2. $r = \sum_{i=1}^{N} [\![r]\!]_i$
3. $MAC == g^x h^r$
4. $l' = Enc_{pk_{cd}}(l; r_l)$
5. $x == (l', out)$

verifiable by all servers in **M**. The verification process consists of two phases: (1) a check for share consistency, which requires a light online interaction by the mixnet parties, and (2) the verification of the proof, which is done locally. We now provide the solution for both Additive and Shamir SS schemes.

Additive SS. Details are shown in ZKP 2. The shares are computed as $x = \sum_{i=1}^{N} x_i$, where $N-1$ shares do not reveal the secret. Initially, the payer sends (ρ_c, MAC) to each M_i, where MAC is a Message Authentication Code for the secret instantiated with a Pedersen commitment scheme as defined in Equation 6.9 (we note that the commitment randomness is also secret shared). We benefit from the additive homomorphic property of Pedersen commitments to enable every M_i to recompute the MAC using their secret shares. To achieve this, each M_i broadcasts $(MAC, g^{x_i} h^{r_i})$ to the other servers, thus each server can recompute the MAC using the partial commitments received from the other servers according to Equation 6.10.

$$MAC = g^x h^r \tag{6.9}$$

$$\prod_{i=1}^{N} g^{x_i} h^{r_i} = g^{\sum_{i=1}^{N} x_i} h^{\sum_{i=1}^{N} r_i} = g^x h^r = MAC \tag{6.10}$$

After the authentication of the shares, the proof ρ_c (presented in ZKP 2) can be locally verified using the MAC as public input. We should note that the public inputs are common to all M_i and, even if the shares of the secret are not verified with respect to l' inside the proof, the MAC is, leading to a secure verification because the MAC implies the verification of the shares in the online phase.

Shamir SS. Details are shown in ZKP 3. The shares are based on a secret polynomial of degree t such that $x = P(0)$. Each secret holder is identified with an integer id $j \in \{1, ..., N\}$, and the share for M_j is computed as $P(j)$. Given a sufficiently large number of shares $t+1$, the polynomial can be reconstructed using interpolation, and the secret can be recovered. First, for the online verification, the payer sends $(\rho_c, \{c_0, c_1, ..., c_t\})$ to every M_j, where c_i are Feldman's polynomial commitments [227] such that $c_0 = g^x$, and $g_i = g^{a_i}$, being a_i the i^{th} coefficient of the secret polynomial $P(X)$. The share consistency check leads every M_i to locally verify its share using Equation 6.11 and broadcast $\{c_0, c_1, ..., c_t\}$ to the rest of M_j.

UNIVERSIDAD DE MÁLAGA

ZKP 3: Shamir SS mixnet proof

Public inputs: $\{c_0, c_1, ..., c_t\}$, g, l, pk_{cd}
Private inputs: $\{[\![x]\!]_i\}_{i\in\{1,...,N\}}$, $\{a_1, ..., a_t\}$, l', out, r_l
Computation:

1. $x = Open([\![x]\!]_1, [\![x]\!]_2, ..., [\![x]\!]_N)$
2. Commitment verification:
   (a) $c_0 = g^x$
   (b) For all $i \in \{1, ..., t\}$: $c_i = g^{a_i}$
3. $l' = Enc_{pk_{cd}}(l; r_l)$
4. $x = (l', out)$

The slight difference with respect to the additive version is that in the former, the consistency check is jointly performed using the MAC, i.e., it checks that the shares reconstruct x. On the other hand, in Shamir's version, each server locally verifies that its share is consistent with respect to a secret polynomial, i.e., they do not verify the other servers shares, and for that reason they must agree that the polynomial commitments sent by the payer are the same to each server.

$$g^{[\![x]\!]_i} = \prod_{j=0}^{t} {c_j}^{i^j} = g^{\sum_{j=0}^{t} {a_j}^{i^j}} = g^{P(i)} \tag{6.11}$$

After the authentication of the shares, the proof ρ_c can be locally verified using the polynomial commitments, similar to the scheme with Additive SS.

6.3.3.2 Proof of concept implementation

This section presents a proof of concept implementation[7] of the proof of compliance generated by the payer, that we employ to run a benchmark to test its practical feasibility in Section 6.3.3.4.

This implementation is assumed to work in tandem with the Bitcoin wallet application and is comprised of two main modules: (1) a cryptographic library, responsible for generating the linkage parameters (l, l'), and (2) a compliance proof generator, which produces the compliance proof ρ_c. Moreover, we assume that the application sends the off-chain messages to the mixer, while the wallet signs and sends the on-chain transactions. The cryptographic library is implemented in Python and Sagemath (the latter for an easier handling of elliptic curve operations) and it supports the two schemes presented in the section above, i.e., proofs based on decryption and proofs based on secret sharing.

Regarding the proof generator, we opted for zk-SNARKs due to their zero-knowledge, succinctness, and non-interactivity. Nevertheless, there are several protocols available, each with distinct characteristics, as introduced in Section 2.1.2.4. The main

[7] It can be found in `https://github.com/damesca/zk-mixer`

UNIVERSIDAD DE MÁLAGA

decision choice is typically based on the need for universal circuits, and then performance can be sought. We notice that our application does not need universal circuits because the mixer verifies proofs which are always built on the same circuit. Therefore, needing a setup is not an issue, because it can be done once (when deploying the mixing service for the first time), and then the proving key can be publicly shared somewhere. Since this is the case, we can perfectly choose Groth16 algorithm because it provides the fastest proving time and the shortest proof size from the set of zk-SNARKs options.

Additionally, Groth16 has been extensively researched and there are quite good developments for its usage. This implementation relies on Circom[8] [228] and SnarkJS[9] toolkit. Circom enables writting ZKP programs in a high-level language and compiling them to constraints. On the other hand, SnarkJS facilitates the computation of the proofs. Moreover, Circomlib[10] is a Circom library that includes diverse functionalities such as ECC operations and hash functions. We have implemented new circuits to compute ρ_c, including public key encryption and secret sharing.

There are additional details to consider. This implementation uses altBn128 curve as Groth16 building block. It natively supports operations modulo p, i.e., the curve's field modulo. In this case, field elements are limited to 253 bits. Although this modulo can be expanded, it comes at the cost of introducing bit-wise circuits, hence additional constraints. Since the bottleneck for constraints are SHA2 hash functions, we also consider circuits based on MiMC [229] and Poseidon [230], which are hash functions designed to be efficient in arithmetic circuits.

Decryption-based mixnet proof. Its main block is public key encryption. We have implemented a hybrid encryption scheme, based on EC-Elgamal for key encryption and AES[11] for message encryption. Hybrid encryption is selected as a solution to layered encryption, where ElGamal's output is a pair of points and the input message is a single input.

Secret-sharing-based mixnet proof. This scenario requires computation for only one encryption, thus computational workload is expected to be light. The additional overhead comes from the additive secret shares and the commitments. The secret sharing version of the proof has a challenge: x must be shared modulo p. We note that $x = (l', out)$, where l' is a hybrid encryption ciphertext and the address *out* may have, e.g., 20 bytes in Bitcoin. Having field elements limited to 253 bits, concatenation cannot be used to encode these values in x. To solve this issue, we have devised two possible solutions, the first of which has been implemented:

1. To encode $x = (l, out)$ and include l' inside tx_{IN} instead of tx_{OUT}, because l is shorter than l'. However, assuming a P2PKH Bitcoin address of 20 bytes, l is limited to be sampled with 11 bytes maximum.

2. To send l and *out* as different secret shared values, which expands to 253 bits the length of l. However, both secret shared lists must be ensured to be permuted with the same π. Below, we sketch a short protocol to verify that

[8] `https://docs.circom.io/`
[9] `https://github.com/iden3/snarkjs`
[10] `https://github.com/iden3/circomlib`
[11] `https://github.com/Electron-Labs/aes-circom`

UNIVERSIDAD DE MÁLAGA

two lists are permuted with the same π.

Secure Permute Twice Protocol

This protocol allows to verify that, given two instances of the *secret share shuffle* protocol from Chase et al. [216], they have been executed with the same permutation. More precisely, it is applied on the values obtained in the core functionality, i.e., the *share translation* protocol, which relies on a *Oblivious Punctured Vector* protocol. OPV can be seen as a $\binom{n}{n-1}$-OT because it allows Alice to learn $n-1$ values from a vector $\mathbf{v}$ of length n known by Bob. A share translation instance of length N implies the execution of N OPVs, where in OPV i, the position of the vector which remains unknown to Alice is determined by $\pi(i)$, i.e., the value of the permutation in index i.

Protocol 8: Permute Twice

1. Execute OPV twice, with $\mathbf{M}^{(1)}$ and $\mathbf{M}^{(2)}$ (N independent executions):
   (a) Alice inputs $\pi^{(1)}(i), \pi^{(2)}(i)$.
   (b) Bob inputs $\mathbf{v}_i^{(1)}[j], \mathbf{v}_i^{(2)}[j]$.
   (c) Alice outputs $\{\mathbf{v}_i^{(1)}[j], \mathbf{v}_i^{(2)}[j]\}_{j \neq \pi(i)}$ and computes $\mathbf{\Delta}^{(1)}$ and $\mathbf{\Delta}^{(2)}$.
2. Index linking phase:
   (a) Alice computes $\mathbf{c}_A$, with $\mathbf{c}_A[j] = \mathbf{\Delta}^{(1)}[j] \oplus \mathbf{\Delta}^{(2)}[j]$ or nothing if $j = \pi(i)$.
   (b) Bob computes $\mathbf{c}_B$, with $\mathbf{c}_B[j] = \mathbf{\Delta}^{(1)}[j] \oplus \mathbf{\Delta}^{(2)}[j]$.
3. PSI-CA phase:
   (a) Alice inputs $\mathbf{c}_A$.
   (b) Bob inputs $\mathbf{c}_B$.
   (c) If $|\cap| \neq |\mathbf{c}_B| - 1$, abort.

Protocol 8 shows the complete verification protocol to be ensured the application of the same permutation π in both independent instances of the share translation. First, phase 1 depicts the execution of the OPV protocol twice, independently and in parallel. This allows Alice to learn both matrices $\mathbf{M}^{(1)}$ and $\mathbf{M}^{(2)}$, where row i is a vector $\mathbf{v}_i$ from where Alice learns all its cells but $\mathbf{v}_i[\pi(i)]$, i.e., the puncture. Note that to ensure that Alice uses the same π in both executions, it just suffices to check whether Alice does not know the values in the same position in $\mathbf{v}^{(1)}$ and $\mathbf{v}^{(2)}$. To achieve that, in phase 2, both Alice and Bob compute vectors $\mathbf{c}_A$ and $\mathbf{c}_B$, which should be the same but exactly in one position: the one unknown to Alice. If Alice cheats, she ends with two unknown cells in different positions, and therefore $\mathbf{c}_A$ and $\mathbf{c}_B$ will differ in more than one cell. Finally, the verification phase can be achieved in a privacy preserving way by using a PSI-CA protocol (see Section 3.3 for more details). Thanks to this, by using $\mathbf{c}_A$ and $\mathbf{c}_B$ as inputs for Alice and Bob respectively, they both can learn the number of coincidences of their vectors. Note that the verification must be done in a privacy-preserving way to avoid Alice actually learning the values she does not known, which would imply an insecure instance of the share translation protocol. Therefore, Bob will be ensured that Alice applied the same permutation π twice (in both independent executions of the OPV) only if the result of the PSI-CA is $|\mathbf{c}_B - 1|$.

UNIVERSIDAD DE MÁLAGA

6.3.3.3 Integration in Bitcoin

Arbitrary data can be added to Bitcoin transactions inside the PubKey locking script, with a maximum allowed length of 10,000 bytes. More specifically, the DROP opcode allows to remove the previously stacked item so it does not affect the unlocking logic. Therefore, it allows to include l and l' inside the transactions without interfering them, but enabling an off-chain verifier to read both parameters.

We remark that l is a short and randomly sampled value of 32 bytes ideally (but only 11 bytes in the secret sharing-based approach). On the other hand, l' is the ciphertext of a hybrid encryption scheme, therefore it takes 32 bytes for the symmetric ciphertext when using AES and 128 bytes for the key encapsulation when using EC ElGamal (it takes two curve points, hence 4 scalars of 32 bytes), i.e., a total of 160 additional bytes.

As an example, a P2PKH locking script with the linkage parameter would be as follows:

$\langle linkParam \rangle$ OP_DROP OP_DUP OP_HASH160 $\langle pubKeyHash \rangle$ OP_EQUALVERIFY OP_CHECKSIG.

Finally, we notice that the cost to include l is cheaper than l', and assumed by the payer in the transaction that funds the mixer. Regarding the larger cost assumed by the mixer for l', despite it may lead to higher fees requested from the mixer to the payers, the former can decrease the payment fees by sending one round of mixed UTXOs inside a single transaction including all the payments to the recipients.

6.3.3.4 Evaluation

This section presents the results acquired through the execution of the proof of concept under diverse settings. The programs were tested using an Ubuntu 22.04.1 LTS virtual machine, operating on an Intel ESXi server with an 8-core CPU (2.2 MHz) and 32 GB of RAM.

The efficiency of the proof varies according to the particular hash function used. Our design employs hashing inside ElGamal encryption (hashed version) and authenticated AES. The number of constraints for the ElGamal block with respect to the hashing algorithm are the following: 62,906 with sha256, 4,097 with mimc7, and 3,609 with poseidon.

To evaluate the implementation, we conducted different executions of ZKP generation on both the decryption-based and the secret sharing-based mixnet version. In each setting, we chose varying numbers of mixing nodes, with fewer nodes for the first setting compared to the second. There are two reasons for this: firstly, the decryption-based solution allows a higher corruption threshold than the secret sharing-based option, and secondly, the decryption-based setting becomes inefficient before the latter. Table 6.3 shows the results from several executions, including different hash functions. It is clear that sha256 instances introduce more constraints than mimc7 and poseidon ones. This has a direct impact on the execution time, i.e., generating the witness and the proof. We notice that despite mimc7 has a slightly

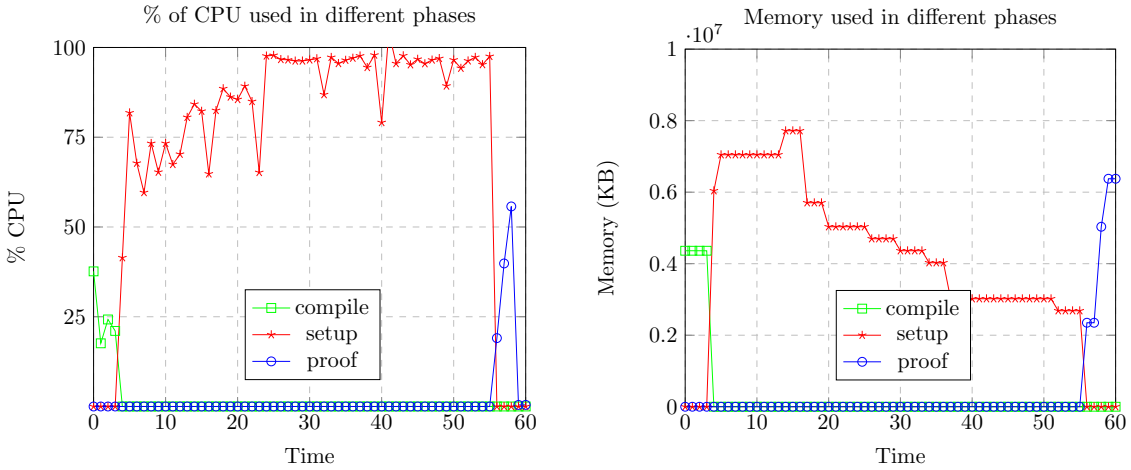
UNIVERSIDAD DE MÁLAGA

Table 6.3: Performance of decryption-based and secret-sharing-based ZKPs for transaction compliance. Terminology: *constr.:* number of constraints in the circuit, τ_{min}*:* minimum size of the tau ceremony (setup) needed to compute the circuit, *wt:* witness generation time, and *pt:* proof generation time. All times are indicated in seconds.

Prot.	N	sha256				mimc7				poseidon			
		constr.	τ_{min}	wt	pt	constr.	τ_{min}	wt	pt	constr.	τ_{min}	wt	pt
Dec.	3	875,504	2^{20}	3.513	30.405	168,964	2^{18}	0.854	8.111	163,108	2^{18}	5.051	8.090
	4	1,101,350	2^{21}	4.379	47.630	218,053	2^{18}	1.036	8.767	210,733	2^{18}	5.219	8.590
	5	1,327,196	2^{21}	5.227	51.622	267,142	2^{19}	1.223	13.905	258,358	2^{18}	5.408	9.594
SS	4	202,092	2^{18}	1.102	8.613	25,823	2^{15}	0.419	2.318	24,359	2^{15}	4.582	2.269
	8	204,142	2^{18}	1.098	8.725	27,873	2^{15}	0.443	2.383	26,409	2^{15}	4.646	2.295
	12	206,640	2^{18}	1.106	8.607	30,371	2^{15}	0.405	2.434	28,907	2^{15}	4.579	2.377

larger number of constraints, it is slightly faster to compute because it presents fewer layers.

All the implementations add a constant number of constraints per round in the protocol, rendering them linear. The base cost for the encryption of l with pk_{cd} is fixed for both the decryption-based and the secret sharing-based schemes (although dependent of the hash scheme). Then, even when both scenarios are linear, the decryption-based mixnet grows at a much faster rate than the secret sharing scheme because each round of hybrid encryption imposes significantly more constraints than each new secret holder added in the secret sharing method. Ultimately, the additive secret sharing approach appears almost constant compared to the decryption-based alternative due to the hybrid encryption block's overwhelming cost.



(a) Percentage of CPU used

(b) Amount of memory used

Figure 6.9: Performance metrics to compute a proof of compliance

We also gathered data on energy consumption during the proof generation. Figure 6.9a displays the percentage of CPU used in proof computation, using the decryption-based proof with three layers of encryption and sha256 as the hash function. The figure presents the consumption of each of the three phases: circuit compilation, circuit-specific setup, and witness and proof generation. We note how the setup phase consumes significantly more CPU than other phases, often running

UNIVERSIDAD DE MÁLAGA

at almost full capacity. Nonetheless, it is worth noting that circuit compilation and setup only need to occur once, since our system design does not require circuit modification. In practice, the payer only needs to execute the witness and proof generation functionalities. A similar result is shown in Figure 6.9b regarding memory consumption, where setup consumes the most.

6.4 Summary

In this chapter, we have introduced different applications that benefit from the use of PETs, in general, and MPC, in particular. The first application is the use of PSI in consortium blockchain scenarios for different use cases, but mainly for medical analysis such as genomic. We have shown how our designs to embed MPC in the blockchain allows easy-to-use applications where the users can interact with PSI in a natural way by calling smart contracts methods. Second, we have introduced an application that targets crowd counting for pandemic scenarios based on the solutions provided for contact tracing and adding geolocation capabilities. The application proposed has been formalized and evaluated by simulations and it is concluded that it can help citizens to avoid excessively crowded places. In addition, we have proposed a solution to protect location data by using a privacy-preserving space mapping thanks to threshold cryptography. Finally, our last application introduces a solution for private payments with cryptocurrency based on a decentralized mixer and a fine-grained disclosure mechanism. We adopt a mixer based on threshold signatures that is compatible with Bitcoin, and propose a reversible unlinkability solution that allows the users to reveal payment data only to authorized parties. In addition, compliance is ensured by NI-ZKPs which are verified by the mixer without learning anything about the private payments.

UNIVERSIDAD DE MÁLAGA

Chapter 7

Conclusions

This final chapter provides a brief summary of the work done in this thesis. We first summarize the details found about the main problems of MPC deployments in real scenarios and the difficulties of blockchain to adopt solutions with privacy. After that, we present a gathering of the main contributions that have been proposed in this thesis. Finally, we conclude with a discussion about the future lines of work that remain from the open problems identified throughout this work.

7.1 MPC and real deployments

As described in the introduction, protection of data in usage is a novel approach that enhances the security of data while still enabling their use in different applications, even in those that are outsourced from the domain of the data owner. For this paradigm, Privacy Enhancing Technologies gather a range of cryptographic tools that provide interesting capabilities for different ways of protection. From these, Secure Multi-Party Computation is the one that captures the most general and adaptive setting, despite it is probably the most difficult one. The reason is mainly because MPC is a wide paradigm with a large range of solutions, and many different issues arise when it is instantiated in real systems. These vary from protocols and parameters selection, to coordination between parties across the network. In addition, MPC typically works on decentralized models of trust, where the different parties that collaborate to carry on the protocol do not trust each other. This not only implies several difficulties for the protocol design aspect, but also to execute the protocol in a concrete instance that guarantees the security assumptions.

7.2 Blockchain and privacy

In this thesis, we have discussed about blockchain as an interesting technology that provides a layer of security, mainly by decentralization and immutability of data. This is directly related with the notion of trust, where a decentralized deployment with distrustful parties that have the proper incentives leads to that security layer. Blockchain is not only interesting because of the additional security it provides, but also because it enables new use cases that are based on that notion of distrust.

UNIVERSIDAD DE MÁLAGA

However, it presents a major barrier which is the lack of privacy. This is a consequence of its design, which is based on transparency and public disclosure of data in order to achieve public verifiability. However, it prevents many use cases from being adopted in a decentralized setting, even when they could benefit from such an additional layer of security. We have introduced some works that already have targeted this problem and we have discussed about their limitations. These vary from hardware-based TEEs, which are not very flexible and also introduce additional security issues inherent to hardware, to FHE-based solutions which can be very heavy for real deployments.

7.3 Contributions

The work in this thesis has targeted different layers where MPC and blockchain interact, and that range from the applications to the specific cryptographic constructions. In this section we provide a summary of the main contributions provided.

7.3.1 Analysis of real world drawbacks

MPC has been analyzed mainly from the theoretical point of view since it was conceived. Despite it has evolved a lot and many different protocols have been proposed in order to solve a wide range of problems in efficient ways, additional details concerning how MPC is deployed or interacted with are omitted many times. We have addressed this issue by analyzing the different settings in which MPC protocols can be instantiated and how the notion of trust presents a central role in the whole process. We have identified two main settings for MPC instances, which are those governed by the parties that provide the inputs and those where the computation is outsourced to a committee of parties. While in the first setting the main difficulty is to coordinate the parties to execute the MPC, the second setting implies additional difficulties such as the process of selecting the committee of computing parties such that trust relations remain in a secure and decentralized way. This last point is very related to one of the frameworks we have proposed to provide a confidentiality layer in public blockchains, where the blockchain itself can serve as a useful layer to setup secure MPC committees.

In addition, we have gathered some of the most useful and extended special-purpose protocols in the MPC ecosystem and we have analyzed many of their details from a perspective of usability and the benefits introduced by the different approaches to solve them. Specifically, we have provided more precise details in the case of PSI, because we identified it as an interesting MPC protocol with many different applications. By following a Systematic Literature Review approach, we have provided a deep analysis of this protocol from a point of view of practicality and usability, because we identified that there were many research proposals for PSI but such a critical compilation has never been done properly. We have provided a taxonomy, a deeper understanding of the different building blocks used to solve PSI, and useful notions to select PSI protocols regarding the properties of the specific use case to solve with it.

UNIVERSIDAD DE MÁLAGA

7.3.2 Confidential computations in blockchain

As previously stated, one of the main difficulties for a wider adoption of blockchain technology is its lack of privacy. From the range of PETs proposed to solve this drawback, we identified that, despite some recent works have aimed at MPC, such solutions have not been researched with the proper detail. Hence, we have discussed about the main relations between MPC and blockchain, and we have used that knowledge to propose two specific solutions to use MPC as a core block for confidential computations in blockchain. Despite both are decentralized, linking them in the same system arise a set of difficulties that are not trivial to solve, mainly because of the atomic nature of public transactions in blockchain.

In our first model, we have proposed a solution to embed MPC protocols inside Hyperledger Besu, an Ethereum client that already presents some sort of privacy in private/consortium blockchains. With the target of minimizing the impact of adopting our solution, we have provided specific components that, once embedded in the blockchain node, allows it to execute MPC instances from the application side by calling smart contract methods. A direct implication is that, in order to minimize the amount of modifications made to the software of the blockchain client, we have not addressed the most efficient solution. However, our design allows to embed specific families of useful MPC protocols directly in the client's software, such that the application can lead to a set of blockchain nodes jointly computing some function on private data.

In our second model, we have targeted an outsourced MPC model, which facilitates the adoption for more users despite the amount of resources they have available. The framework proposed is very interesting because it presents a great balance between what MPC offers to the blockchain, i.e., a fine-grained confidentiality layer, and what blockchain offer to MPC, i.e., a decentralized ecosystem based on incentives where committees of computing parties can be selected in different ways that maximize the guarantees on the desired trust notions. For the fine-grained aspect, we have developed a solution that adopts the promise pattern from client-server settings with asynchronous computation to blockchain, such that smart contracts can be executed naturally while containing specific variables that are private and resolved by MPC committees. In addition, we have proven that this solution is more cost-effective compared to other solutions such as those that rely on FHE, mainly because our framework leads the cost of the MPC execution to be independent of the cost for public computations in the specific blockchain where it is instantiated.

7.3.3 Efficient and dynamic delivery of private data

Another contribution of this thesis has been the introduction of a new approach to design protocols that allow an efficient and dynamic way to secret share data. With efficient, we refer to protocols that allow the data owner, i.e., the dealer, to send a single ciphertext containing the message desired to be shared. With dynamic, we refer to the possibility of encrypting data first, storing it somewhere public, and deciding later the committee of holders with which data will be shared. Our solution targets committees that hold secret shared data because the most extended general-purpose MPC protocols are based on secret sharing. Thanks to this, we are able to avoid the assumption in outsourced MPC that the dealer securely delivers each

UNIVERSIDAD DE MÁLAGA

share to each holder, which implies setting up n independent confidential channels, e.g., by using TLS.

Our proposal, named OSRE, captures this notion from a wide perspective. We notice that other solutions, such as HEDS, fit within this concept of compactly sharing data with a single ciphertext, but it lacks dynamism and also imply FHE, which can be more expensive when sharing. We have contributed to OSRE with some intuitions and intermediate results, which have set the base for future lines of research. In addition, we have provided a concrete protocol instance that achieve both efficiency and dynamism, and which is based on the NTRU encryption scheme for being AHE and PRE. This concrete protocol has been proven secure, and we have also provided an implementation and an evaluation that shows that it is more efficient that the standard solution with independent confidential channels in specific scenarios.

Regarding the use cases that are mostly benefited from this solution, we have identified two main. The first one comprises IoT scenarios where devices share private data with MPC committees in periodic intervals, which benefits a lot from the amortized cost. The second use case is in blockchain deployments with MPC-based confidentiality, like the framework we have proposed in this thesis. In that case, including a single ciphertext in a transaction is more efficient and easy to use from an application in order to secret share private data with an MPC committee.

7.3.4 New applications

In addition to the analysis, discussion, and new proposals for MPC and blockchain, from the architecture and the cryptographic points of view, we have also contributed in this thesis with new applications that solve specific problems and that rely in PETs and MPC to protect data in use.

In the first application, we have provided an instance of our model for MPC in Hyperledger Besu by integrating a PSI protocol inside the node. We selected PSI because some of its use cases fit well with the consortium blockchain approach, for example privacy-preserving computations on genomic data. Thanks to our model, PSI can be executed directly from the blockchain application and the computation is carried on by the Besu nodes. We have also discussed about usability of PSI on both the Besu model and the framework for general-purpose MPC in public blockchains, mainly regarding how the protocol is called from smart contracts methods. In addition, we have noticed that our new method for efficient delivery of secret shared data, namely OSRE, can benefit the PSI use case when being computed in the outsourced MPC model.

The second application targets crowd counting in a pandemic scenario, where users can query different locations in order to learn if they are overcrowded and avoid them. We have based our design on a contact tracing application that uses ephemeral IDs to protect the users' identities, and also on geolocation data to allow location-based queries. The design has been proven to be functional and accurate when many users share the same location, therefore achieving its main purpose. In addition, since geolocation data is very sensitive, we have provided a privacy-preserving mechanism that employs a pseudo-random domain mapping such that the appli-

UNIVERSIDAD DE MÁLAGA

cation server cannot learn the real location of users. We have also analyzed how MPC can help in securing the pseudo-random domain mapping process, which in its core employs a PRF, by distributing the key in a set of non-colluding parties and therefore avoiding a TTP for the mapping. However, it can be observed that the performance of the MPC procedure still has room for improvement, especially when working on a WAN with high latency.

Finally, our last application provides a mechanism that allows an MPC-based decentralized mixnet in Bitcoin, that provides private cryptocurrency payments, to disclose payment information in a fine-grained way. For this, our solution relies on reversible unlinkability by trapdoor functions, which are instantiated with public key cryptography. In addition, we have provided a mechanism based on NI-ZKP that enables compliance by letting the users that send transactions to the mixer proving it that the transaction's privacy can be reverted by a specific conditional discloser, while remaining the content of the transaction hidden to the mixer.

7.4 Future work

To conclude this thesis, this last section is devoted to the open work that has been identified throughout the rest of the thesis, which has interest enough to keep researching on it as future work.

First, regarding the proposals provided in Chapter 4, there is plenty space left for improvements, mainly with respect to implementation and evaluation. Regarding the design to embed MPC in Besu, we notice that a deeper work can be done to simplify some functional blocks and improve performance. In addition, it would be interesting to include other protocols by using ISCs, and perform a benchmarking with different instances of MPC solutions, in order to provide a proper benchmarking in the blockchain setting. On the other hand, regarding the framework for MPC in public blockchains, despite we have provided a theoretical evaluation that proves its feasibility, it would be of paramount importance to provide a proof of concept implementation that allows to test performance in a scenario close to reality. This will also allow to detect functional caveats and bottlenecks, in order to improve the design of these parts. Apart from the implementation, we notice that the mechanisms for committee member selection and the multi-input setting have been discussed somewhat generically, and therefore the implementation would help in testing how the different approaches for these aspects behave in real settings.

Second, regarding Chapter 5 and our new proposal named OSRE, we identify many research directions from where additional work would be useful. This is mainly because it is a novel approach that has not been considered before in the literature. First, as we noticed in its corresponding section, we identified HEDS as a solution for OSRE without dynamism, but also noticed that the homomorphism is not needed in our scenario. For that, an EDS scheme would be enough, but we left as future work to research if EDS is possible, i.e., whether $EDS \subset HEDS$ or, on the contrary, $EDS = HEDS$. Next, regarding the specific constructions provided for OSRE, some of them are intuitions or intermediate solutions that present some security inconveniences or do not have a proper scheme that can instantiate them. We identify this line of research of great interest because finding additional schemes that are more efficient

UNIVERSIDAD DE MÁLAGA

can help in lowering the communication bounds where OSRE is better than the standard setting, and therefore improving its usability. In addition, our concrete scheme, despite functional, has also space left for improvements. First, it would be interesting to find other encryption schemes that presents both characteristics needed, i.e., AHE and PRE, while presenting more succinct ciphertexts than NTRU. Second, we introduced a variation in which threshold decryption is used to achieve an $O(1)$ communication cost in the dealer side, however, it remains to find a proper instance of threshold decryption for NTRU that does not expose anything about the partial decryption keys.

Another interesting line of research can be the intersection between MSS schemes and those that enable OSRE. Achieving a solution that combines both approaches could lead to a single ciphertext sent by the dealer that allows each holder from a set of n to retrieve its own share from the ciphertext and where each share compacts $l > 1$ messages, therefore improving communication efficiency even more.

Finally, regarding the three applications provided in this thesis, they have space left for improving the designs in terms of efficiency and usability, mainly by selecting more efficient cryptographic mechanisms, improving the designs from the engineering perspective, and also by providing more realistic implementations that allow testing them under better conditions. MPC is quite expensive in terms of communication and latency when deployed in WAN settings, but new works in the literature are trying to find new approaches and assumptions such that the main lower bounds in MPC can be beaten. Since our constructions and applications are not built on a specific assumption for MPC and treat that building block somehow like a black box, performance improvements in the MPC protocol would lead to better performance in our proposals.

UNIVERSIDAD DE MÁLAGA

Appendix A

Resumen

A.1 Motivación, retos y contribuciones

Proteger los datos de los usuarios con técnicas de grano fino es una cuestión cada vez más aceptada. Actualmente el enfoque en la ciberseguridad dista mucho de la mera seguridad perimetral de años atrás. Por el contrario, y motivado por la gran cantidad de fugas de datos y abuso de los usuarios que se están produciendo los últimos años, el foco se sitúa ahora mucho más cercano a los propios datos, independientemente de donde estén. Dentro de este prisma amplio de la seguridad, una paradigma reciente es aquel que proporciona "protección de los datos mientras se usan", ya que este es el punto máximo de exposición, principalmente por la necesidad de eliminar las protecciones habituales como el cifrado para poder utilizar la información.

Dentro de este paradigma se recogen las llamadas *tecnologías para mejorar la privacidad* (PETs), las cuales ofrecen un amplio abanico de opciones (fundamentalmente criptográficas) para proporcionar capacidades extra en materia de privacidad y protección de los datos. En particular, esta tesis se centra en un grupo de tecnologías muy particular, recogidas bajo el nombre de Computación Segura Multiparte (MPC).

MPC captura el problema abstracto en el que un grupo de usuarios con desconfianza mutua quieren evaluar algún tipo de función sobre sus datos privados de forma que estos no se expongan en ningún momento a nadie y que el único resultado compartido al final de la evaluación de la función sea su resultado. MPC incluye un conjunto diverso de tecnologías y aplicaciones, las cuales son analizadas en esta tesis desde un foco crítico y analítico respecto a los problemas que surgen al implementarla en sistemas reales. La motivación tras esto es el hecho de que son tecnologías muy complejas las cuales se han tratado principalmente desde un foco teórico en la literatura.

Además, otro eje fundamental de esta tesis son las tecnologías de registro distribuido y más concretamente las tecnologías blockchain. Estas tecnologías han sido adoptadas con gran interés durante los últimos años ya que ofrecen un paradigma de descentralización y seguridad muy interesante, el cual permite que muchos casos de uso se puedan llevar a cabo bajo requisitos de confianza más relajados. Sin embargo, las tecnologías blockchain presentan un problema fundamental: su falta de

UNIVERSIDAD DE MÁLAGA

privacidad. Esto se debe a que para obtener sistemas seguros y altamente verificables por cualquier participante de la red, toda la información utilizada debe estar almacenada de forma pública.

De todas las soluciones posibles para contribuir en materia de privacidad en entornos blockchain, en esta tesis identificamos MPC como una de gran valor y que aporta altas capacidades para procesar datos manteniendo su protección durante toda la cadena de ejecución. Además, no solo MPC es útil para aportar capacidades de cómputo confidencial a la blockchain, sino que ésta última también es interesante para garantizar despliegues de nodos MPC de forma más segura y verificable. Por tanto, concluimos que el beneficio de unir estas tecnologías se produce en ambos sentidos.

Además de soluciones en materia de integración a nivel de sistema y componentes de estas dos tecnologías, el resto de la tesis las aborda desde diferente capas de concreción. Por ejemplo, respecto a la construcción de nuevos protocolos criptográficos, aportamos una nueva familia de protocolos que permiten a los dueños de los datos el compartirlos con motores de cómputo MPC de una forma más eficiente y dinámica. Esto contrasta con la forma clásica en la que un usuario tiene que romper un dato secreto en fragmentos y enviar cada fragmento de forma individual, a través de un canal confidencial, a cada uno de los miembros del comité MPC.

Finalmente, desde el punto de vista de nuevas aplicaciones, ofrecemos tres en esta tesis. En primer lugar, una aplicación para ejecutar protocolos de Intersección Privada de Conjuntos (PSI) en blockchains de consorcio, lo cual tiene gran utilidad para el procesamiento de datos en determinados ecosistemas colaborativos, como el sector médico y los datos genéticos. En segundo lugar, una aplicación que ofrece a los usuarios información sobre espacios con grandes multitudes en situaciones de pandemia, de forma que puedan evitarlos y prevenir posibles contagios. Esta solución se basa en tecnologías de trazabilidad de contactos y geolocalización, y utiliza MPC como solución para la protección de los datos de geolocalización de los usuarios. Finalmente, la tercera aplicación se centra en los mecanismos de pago privados en blockchain mediante el uso de mezcladores. Concretamente, nuestra solución se centra en un mezclador descentralizado en Bitcoin que utiliza MPC como base y ofrece un mecanismo de privacidad reversible, de forma que los datos de pagos privados se pueden compartir bajo demanda y, además, con garantías de cumplimiento de normativa gracias al uso de pruebas de conocimiento cero.

A.2 Análisis de arquitecturas y protocolos en el paradigma de la Computación Segura Multiparte

Como primera contribución de esta tesis, se analiza el paradigma de las soluciones MPC desde un prisma de enfoque práctico. Concretamente, identificamos algunas generalizaciones que suelen hacerse en las propuestas teóricas de nuevos protocolos, las cuales pueden ser problemáticas al instanciarlos dentro de soluciones reales. Concretamente, la falta de confianza entre los nodos implicados es un gran problema a la hora de coordinar el despliegue, principalmente cuando los dueños de los datos

UNIVERSIDAD DE MÁLAGA

no son participantes del proceso de computación MPC, sino que lo delegan a un comité externo.

Además de esto, analizamos algunos de los principales protocolos de propósito específico de MPC y discutimos sobre sus principales características y cuestiones de aplicabilidad. Dentro de este conjuntos de protocolos se encuentran los de generación distribuida de números pseudo-aleatorios, los de criptografía de umbral, y los de operaciones privadas sobre conjuntos de datos. Dentro de este grupo, se identifica a PSI como uno de los protocolos más investigados y con mayor número de casos de usos del paradigma MPC, por lo que proporcionamos un análisis extenso sobre dicho protocolo. De forma más específica, se realiza un análisis sistemático de la literatura, desde un punto de vista de aplicabilidad, el cual busca recoger las principales soluciones de la literatura para resolver PSI y analizar si están listas para ser integradas en sistemas reales.

A.3 Computación Segura Multiparte en entornos blockchain

La siguiente contribución de esta tesis se enfoca en el uso de MPC para resolver el problema de falta de privacidad en las tecnologías blockchain. De forma más específica, se propone usar MPC como motor de computación sobre datos confidenciales, de forma que los usuarios que interactúan con aplicaciones descentralizadas en la blockchain, puedan proporcionar también datos confidenciales los cuales no sean expuestos durante todo el proceso, más allá del resultado de la computación. Además de analizar las principales relaciones e incompatibilidades que existen entre MPC y blockchain, ofrecemos dos soluciones concretas para integrar MPC en sistemas blockchain. En primer lugar, una solución que permite a los nodos de una red blockchain basada en el cliente Hyperledger Besu ejecutar instancias MPC mediante llamadas de los usuarios a interfaces de contratos inteligentes. Esta solución aborda un escenario de consortio en el que un nodo blockchain se asume que pertenece a una organización y sus usuarios delegan en él la participación en el proceso MPC como un miembro más. En segundo lugar, ofrecemos un marco de trabajo para implementar soluciones que permitan ejecutar instancias MPC de forma delegada en blockchains públicas, también mediante llamadas a contratos inteligentes. Esta solución está basada en la existencia de un conjunto de nodos especiales para MPC los cuales funcionan bajo mecanismos de incentivos. Los usuarios pueden llamar a funciones con privacidad de una forma muy granular en los contratos inteligentes, de forma que los datos confidenciales son manipulados por comités de nodos MPC para obtener el resultado. Además, este marco de trabajo ofrece una solución en la que el resultado de la computación es públicamente verificable, de forma que se minimiza la posibilidad de que los comités MPC trabajen de forma incorrecta o maliciosa.

A.4 Compartición dinámica y compacta de secretos

Otro de los enfoques dentro de esta tesis ha sido el analizar los detalles criptográficos de los protocolos MPC. Concretamente, se ha contribuido con una nueva familia de

UNIVERSIDAD DE MÁLAGA

protocolos llamada OSRE, la cual captura (de forma general) una sequencia de operaciones que permiten enviar datos confidenciales a comités de nodos MPC de una forma más eficiente y dinámica. Concretamente, los protocolos OSRE permiten a los dueños de los datos, conocidos como *dealer* en los esquemas de secreto compartido, el enviar un solo cifrado del mensaje original a través del cual cada miembro del comité MPC puede extraer el fragmento del secreto que le corresponde. Esto permite que el coste de compartir datos usando técnicas de secreto compartido pase de un coste lineal con el número de miembros del comité a un coste constante, dependiente únicamente del esquema de cifrado. Además, el hecho de que sean protocolos dinámicos implica que el dueño del secreto puede, inicialmente, cifrarlo bajo su propia clave y almacenarlo en algún lugar, y decidir a posteriori qué comité concreto será el receptor del secreto. Esto implica una facilidad mayor a la hora de interconectar capacidades criptográficas y formas de consumir los datos, además de evitar el tener que volver a cifrar todos los datos de nuevo cada vez que se selecciona otro comité.

Además de proporcionar soluciones intermedias las cuales capturan bien las nociones de OSRE pero no ofrecen esquemas concretos que puedan ser instanciados, también proporcionamos un esquema concreto basado en el esquema de cifrado NTRU. Este esquema presenta propiedades de cifrado homomórfico aditivo, además de re-cifrado proxy. Basándonos en estas características, proporcionamos un esquema OSRE con el que un proxy semi-honesto es capaz de generar, a partir de un único cifrado original, un conjunto de nuevos cifrados en el que cada uno contiene el fragmento correspondiente del secreto para cada miembro del comité. Además, gracias al re-cifrado proxy, cada uno de estos cifrados puede ser descifrado únicamente por el miembro correspondiente, utilizando su par de claves criptográficas. Además del esquema concreto, se proporciona una prueba formal de seguridad, una implementación en Java y una evaluación de la eficiencia tanto de forma teórica como mediante experimentos realizados con la implementación.

A.5 Aplicaciones de la Computación Segura Multiparte

El último capítulo de contribuciones de esta tesis se corresponde con tres aplicaciones en las cuales las tecnologías PETs en general y MPC en concreto, aportan ventajas para poder llevar a cabo estas aplicaciones con garantías de privacidad y protección de los datos.

La primera aplicación está enfocada a ejecutar el protocolo PSI en entornos blockchain de consorcio, mediante la solución proporcionada previamente para el cliente de Ethereum Hyperledger Besu. En primer lugar, identificamos que muchos de los casos de uso de PSI tienen sentido en un marco colaborativo de consorcio, donde las tecnologías blockchain ofrecen capacidades interesantes para garantizar un ecosistema seguro y con trazabilidad de los datos y las operaciones. Dentro de los casos de uso, uno muy relevante es el que aplica PSI a operaciones con genomas, tanto para test de paternidad como para la realización de tratamientos médicos especializados. Dentro de este prisma, facilitamos una instancia concreta de PSI en Besu y analizamos las facilidades de uso con respecto a la interación de la aplicación.

UNIVERSIDAD DE MÁLAGA

Además, identificamos también que la solución de OSRE para compartir datos de forma eficiente y dinámica presentaría grandes ventajas en este caso concreto.

La segunda aplicación resuelve el problema de recuento de multitudes, orientado a escenarios de pandemia en los que los usuarios quieren tratar de evitar las multitudes para no contraer posibles contagios. Concretamente, durante la pandemia COVID-19, aparecieron muchas soluciones de trazabilidad de contactos, con el objetivo de identificar de forma precoz a aquellos ciudadanos con alto riesgo de contagio. Nuestra solucion para recuento de multitudes se basa en un protocolo de trazabilidad de contactos basado en identificadores pseudo-aleatorios y también en datos de geolocalización. Para proteger estos últimos, se propone una solución criptográfica de mapeo del espacio de datos de geolocalización a otro espacio pseudo-aleatorio, de forma que la aplicación no es consciente de en qué lugares reales se encuentran los usuarios. Este mapeo se realiza mediante un intermediario que aplica una función pseudo-aleatoria a los datos de geolocalización, por lo que también proponemos una alternativa en la que esta función se aplica de forma distribuida con MPC, minimizando los riesgos de filtración de los datos de geolocalización reales.

Por último, se propone una solución para realizar pagos anónimos con criptomonedas, concretamente en Bitcoin. Para ello, nuestra solución se basa en un mezclador descentralizado el cual utiliza criptografía de umbral para la firma de las transacciones de salida, y en una mixnet para mezclar las distintas transacciones de entrada entre sí, de forma que se garantiza la privacidad de los usuarios. Sobre esta solución, aportamos un mecanismo de desvinculabilidad reversible, mediante el cual los pagos son anónimos por defecto pero pueden ser revelados de forma selectiva mediante unos intermediarios que siguen las políticas marcadas por los usuarios. Este mecanismo se basa en criptografía de clave pública y, además, proporcionamos una solución con pruebas de conocimiento cero para garantizar el cumplimiento de la normativa, de forma que el mezclador puede estar seguro de que el anonimato de los pagos puede ser revertido en algún momento, pero sin aprender realmente nada sobre el vínculo entre los emisores y los receptores de los pagos.

A.6 Conclusiones y trabajo futuro

Como conclusiones, se enfatiza la complejidad de los protocolos MPC y el amplio abanico de opciones y características diferentes que existe. El análisis muestra que, a pesar de la gran evolución que han recibido estos protocolos a lo largo de los años, aún resta trabajo por hacer en materia de aplicabilidad y un acercamiento a los despliegues en entornos más realistas. Además, se muestra cómo las relaciones entre las tecnologías MPC y blockchain son fuertes, donde ambas pueden aportar beneficios a la otra. Nuestras soluciones para integrar MPC en blockchain permiten mostrar su viabilidad para solucionar el problema de la falta de privacidad, además de incluso aportar mecanismos más baratos (en términos monetarios) que con otras tecnologías. Respecto a la solución propuesta para OSRE, se permite ver cómo nuevas construcciones con un mayor foco en los aspectos prácticos y de eficiencia pueden contribuir a la adopción de MPC en casos de uso para los que inicialmente se pudiera considerar como inviable o demasiado costoso. Finalmente, las tres aplicaciones propuestas muestran como las tecnologías PETs y MPC ofrecen capacidades muy versátiles e interesante para instanciar casos de uso con altas garantías de

UNIVERSIDAD DE MÁLAGA

privacidad.

Como trabajo futuro, resta perfeccionar las soluciones propuestas para integrar MPC en los despliegues blockchain, principalmente en materia de eficiencia y una mayor facilidad de uso. Concretamente, se pretende proporcionar una prueba de concepto para la solución que utiliza comités MPC en la blockchain pública, la cual permita, además de realizar pruebas de rendimiento, el identificar las mejores estrategias para seleccionar los comités MPC y para proporcionar datos privados en los casos de uso con múltiples proveedores de datos. Por otro lado, respecto a OSRE, identificamos mucha investigación por delante para encontrar nuevos esquemas concretos que permitan instanciar OSRE de formas más eficientes y con mayores garantías de seguridad. Finalmente, respecto a las aplicaciones propuestas, también tiene cabida el análisis de propuestas de mejora en materia de eficiencia y facilidad de uso, además de perfeccionar la forma en que las instancias MPC se despliegan y los nodos interactúan entre sí.

UNIVERSIDAD DE MÁLAGA

Bibliography

[1] David Evans, Vladimir Kolesnikov, and Mike Rosulek. *A Pragmatic Introduction to Secure Multi-Party Computation*. en.

[2] Daniel Escudero. *An Introduction to Secret-Sharing-Based Secure Multiparty Computation*. Publication info: Preprint. 2022. URL: https://eprint.iacr.org/2022/062 (visited on 11/28/2023).

[3] Yehuda Lindell. *How To Simulate It - A Tutorial on the Simulation Proof Technique*. Publication info: Published elsewhere. Minor revision. This paper appeared in the book "Tutorials on the Foundations of Cryptography", published in honor of Oded Goldreich's 60th birthday. 2016. URL: https://eprint.iacr.org/2016/046 (visited on 05/07/2024).

[4] Shimon Even, Oded Goldreich, and Abraham Lempel. "A randomized protocol for signing contracts". In: *Commun. ACM* 28.6 (June 1985), pp. 637–647. ISSN: 0001-0782. DOI: 10.1145/3812.3818. URL: https://dl.acm.org/doi/10.1145/3812.3818 (visited on 07/30/2024).

[5] Michael O. Rabin. *How To Exchange Secrets with Oblivious Transfer*. Publication info: Published elsewhere. Harvard University Technical Report 81. 2005. URL: https://eprint.iacr.org/2005/187 (visited on 07/30/2024).

[6] Gilles Brassard, Claude Crepeau, and Jean-Marc Robert. "All-or-Nothing Disclosure of Secrets". en. In: *Advances in Cryptology — CRYPTO' 86*. Ed. by Andrew M. Odlyzko. Berlin, Heidelberg: Springer, 1987, pp. 234–238. ISBN: 978-3-540-47721-1. DOI: 10.1007/3-540-47721-7_17.

[7] Moni Naor and Benny Pinkas. "Computationally Secure Oblivious Transfer". en. In: *Journal of Cryptology* 18.1 (Jan. 2005), pp. 1–35. ISSN: 1432-1378. DOI: 10.1007/s00145-004-0102-6. URL: https://doi.org/10.1007/s00145-004-0102-6 (visited on 07/30/2024).

[8] Tung Chou and Claudio Orlandi. "The Simplest Protocol for Oblivious Transfer". en. In: *Progress in Cryptology – LATINCRYPT 2015*. Ed. by Kristin Lauter and Francisco Rodríguez-Henríquez. Cham: Springer International Publishing, 2015, pp. 40–58. ISBN: 978-3-319-22174-8. DOI: 10.1007/978-3-319-22174-8_3.

[9] Moni Naor and Benny Pinkas. "Efficient oblivious transfer protocols". In: *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*. SODA '01. USA: Society for Industrial and Applied Mathematics, Jan. 2001, pp. 448–457. ISBN: 978-0-89871-490-6. (Visited on 07/30/2024).

[10] Yuval Ishai et al. "Extending Oblivious Transfers Efficiently". en. In: *Advances in Cryptology - CRYPTO 2003*. Ed. by Dan Boneh. Berlin, Heidelberg: Springer, 2003, pp. 145–161. ISBN: 978-3-540-45146-4. DOI: 10.1007/978-3-540-45146-4_9.

UNIVERSIDAD DE MÁLAGA

[11] Vladimir Kolesnikov and Ranjit Kumaresan. "Improved OT Extension for Transferring Short Secrets". en. In: *Advances in Cryptology – CRYPTO 2013*. Ed. by Ran Canetti and Juan A. Garay. Berlin, Heidelberg: Springer, 2013, pp. 54–70. ISBN: 978-3-642-40084-1. DOI: 10.1007/978-3-642-40084-1_4.

[12] O. Goldreich, S. Micali, and A. Wigderson. "How to play ANY mental game". In: *Proceedings of the nineteenth annual ACM symposium on Theory of computing*. STOC '87. New York, NY, USA: Association for Computing Machinery, Jan. 1987, pp. 218–229. ISBN: 978-0-89791-221-1. DOI: 10.1145/28395.28420. URL: https://doi.org/10.1145/28395.28420 (visited on 07/05/2024).

[13] Amos Beimel. "Secret-Sharing Schemes: A Survey". en. In: *Coding and Cryptology*. Ed. by Yeow Meng Chee et al. Berlin, Heidelberg: Springer, 2011, pp. 11–46. ISBN: 978-3-642-20901-7. DOI: 10.1007/978-3-642-20901-7_2.

[14] Adi Shamir. "How to share a secret". In: *Commun. ACM* 22.11 (Nov. 1979), pp. 612–613. ISSN: 0001-0782. DOI: 10.1145/359168.359176. URL: https://dl.acm.org/doi/10.1145/359168.359176 (visited on 07/30/2024).

[15] Shai Halevi and Silvio Micali. "Practical and Provably-Secure Commitment Schemes from Collision-Free Hashing". en. In: *Advances in Cryptology — CRYPTO '96*. Ed. by Neal Koblitz. Berlin, Heidelberg: Springer, 1996, pp. 201–215. ISBN: 978-3-540-68697-2. DOI: 10.1007/3-540-68697-5_16.

[16] Marc Fischlin and Roger Fischlin. "Efficient Non-malleable Commitment Schemes". en. In: *Advances in Cryptology — CRYPTO 2000*. Ed. by Mihir Bellare. Berlin, Heidelberg: Springer, 2000, pp. 413–431. ISBN: 978-3-540-44598-2. DOI: 10.1007/3-540-44598-6_26.

[17] Ivan Damgard and Jens Groth. "Non-interactive and reusable non-malleable commitment schemes". In: *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*. STOC '03. New York, NY, USA: Association for Computing Machinery, June 2003, pp. 426–437. ISBN: 978-1-58113-674-6. DOI: 10.1145/780542.780605. URL: https://dl.acm.org/doi/10.1145/780542.780605 (visited on 09/23/2024).

[18] Torben Pryds Pedersen. "Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing". en. In: *Advances in Cryptology — CRYPTO '91*. Ed. by Joan Feigenbaum. Berlin, Heidelberg: Springer, 1992, pp. 129–140. ISBN: 978-3-540-46766-3. DOI: 10.1007/3-540-46766-1_9.

[19] Jens Groth. "On the Size of Pairing-Based Non-interactive Arguments". In: *Advances in Cryptology – EUROCRYPT 2016*. Ed. by Marc Fischlin and Jean-Sébastien Coron. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 305–326. ISBN: 978-3-662-49896-5.

[20] Mary Maller et al. "Sonic: Zero-Knowledge SNARKs from Linear-Size Universal and Updatable Structured Reference Strings". In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. CCS '19. London, United Kingdom: Association for Computing Machinery, 2019, pp. 2111–2128. ISBN: 9781450367479.

[21] Alessandro Chiesa et al. "Marlin: Preprocessing zkSNARKs with Universal and Updatable SRS". In: *Advances in Cryptology – EUROCRYPT 2020*. Ed. by Anne Canteaut and Yuval Ishai. Cham: Springer International Publishing, 2020, pp. 738–768. ISBN: 978-3-030-45721-1.

UNIVERSIDAD DE MÁLAGA

[22] Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. *PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge*. Cryptology ePrint Archive, Paper 2019/953. 2019.

[23] Alessandro Chiesa, Dev Ojha, and Nicholas Spooner. "Fractal: Post-quantum and Transparent Recursive Proofs from Holography". In: *Advances in Cryptology – EUROCRYPT 2020*. Ed. by Anne Canteaut and Yuval Ishai. Cham: Springer International Publishing, 2020, pp. 769–793.

[24] Benedikt Bünz et al. "Bulletproofs: Short Proofs for Confidential Transactions and More". In: *2018 IEEE Symposium on Security and Privacy (SP)*. 2018, pp. 315–334.

[25] Frederik Armknecht et al. *A Guide to Fully Homomorphic Encryption*. Publication info: Preprint. MINOR revision. 2015. URL: https://eprint.iacr.org/2015/1192 (visited on 09/17/2024).

[26] Abbas Acar et al. "A Survey on Homomorphic Encryption Schemes: Theory and Implementation". In: *ACM Comput. Surv.* 51.4 (July 2018), 79:1–79:35. ISSN: 0360-0300. DOI: 10.1145/3214303. URL: https://dl.acm.org/doi/10.1145/3214303 (visited on 09/17/2024).

[27] Pascal Paillier. "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes". en. In: *Advances in Cryptology — EUROCRYPT '99*. Ed. by Jacques Stern. Berlin, Heidelberg: Springer, 1999, pp. 223–238. ISBN: 978-3-540-48910-8. DOI: 10.1007/3-540-48910-X_16.

[28] T. Elgamal. "A public key cryptosystem and a signature scheme based on discrete logarithms". In: *IEEE Transactions on Information Theory* 31.4 (July 1985). Conference Name: IEEE Transactions on Information Theory, pp. 469–472. ISSN: 1557-9654. DOI: 10.1109/TIT.1985.1057074. URL: https://ieeexplore.ieee.org/document/1057074/authors#authors (visited on 09/17/2024).

[29] Yuval Ishai and Anat Paskin. "Evaluating Branching Programs on Encrypted Data". en. In: *Theory of Cryptography*. Ed. by Salil P. Vadhan. Berlin, Heidelberg: Springer, 2007, pp. 575–594. ISBN: 978-3-540-70936-7. DOI: 10.1007/978-3-540-70936-7_31.

[30] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. "Evaluating 2-DNF Formulas on Ciphertexts". en. In: *Theory of Cryptography*. Ed. by Joe Kilian. Berlin, Heidelberg: Springer, 2005, pp. 325–341. ISBN: 978-3-540-30576-7. DOI: 10.1007/978-3-540-30576-7_18.

[31] Craig Gentry. "A fully homomorphic encryption scheme". crypto.stanford.edu/craig. PhD thesis. Stanford University, 2009.

[32] Marten van Dijk et al. "Fully Homomorphic Encryption over the Integers". en. In: *Advances in Cryptology – EUROCRYPT 2010*. Ed. by Henri Gilbert. Berlin, Heidelberg: Springer, 2010, pp. 24–43. ISBN: 978-3-642-13190-5. DOI: 10.1007/978-3-642-13190-5_2.

[33] Jung Hee Cheon et al. "Homomorphic Encryption for Arithmetic of Approximate Numbers". en. In: *Advances in Cryptology – ASIACRYPT 2017*. Ed. by Tsuyoshi Takagi and Thomas Peyrin. Cham: Springer International Publishing, 2017, pp. 409–437. ISBN: 978-3-319-70694-8. DOI: 10.1007/978-3-319-70694-8_15.

[34] Joppe W. Bos et al. "Improved Security for a Ring-Based Fully Homomorphic Encryption Scheme". en. In: *Cryptography and Coding*. Ed. by Martijn Stam.

UNIVERSIDAD DE MÁLAGA

Berlin, Heidelberg: Springer, 2013, pp. 45–64. ISBN: 978-3-642-45239-0. DOI: 10.1007/978-3-642-45239-0_4.

[35] Ilaria Chillotti et al. *TFHE: Fast Fully Homomorphic Encryption over the Torus*. Publication info: Published elsewhere. Journal of Cryptology 2019. 2018. URL: https://eprint.iacr.org/2018/421 (visited on 09/17/2024).

[36] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. "On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption". English. In: *STOC '12 - Proceedings of the 2012 ACM Symposium on Theory of Computing*. 2012, pp. 1219–1234. DOI: 10.1145/2213977.2214086. URL: https://cris.tau.ac.il/en/publications/on-the-fly-multiparty-computation-on-the-cloud-via-multikey-fully (visited on 09/17/2024).

[37] Aayush Jain, Peter M. R. Rasmussen, and Amit Sahai. *Threshold Fully Homomorphic Encryption*. Publication info: Preprint. MINOR revision. 2017. URL: https://eprint.iacr.org/2017/257 (visited on 09/18/2024).

[38] Andrew C. Yao. "Protocols for secure computations". In: *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*. ISSN: 0272-5428. Nov. 1982, pp. 160–164. DOI: 10.1109/SFCS.1982.38. URL: https://ieeexplore.ieee.org/document/4568388 (visited on 07/05/2024).

[39] Andrew Chi-Chih Yao. "How to generate and exchange secrets". In: *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*. ISSN: 0272-5428. Oct. 1986, pp. 162–167. DOI: 10.1109/SFCS.1986.25. URL: https://ieeexplore.ieee.org/document/4568207 (visited on 07/14/2024).

[40] Joe Kilian. "Founding crytpography on oblivious transfer". In: *Proceedings of the twentieth annual ACM symposium on Theory of computing*. STOC '88. New York, NY, USA: Association for Computing Machinery, Jan. 1988, pp. 20–31. ISBN: 978-0-89791-264-8. DOI: 10.1145/62212.62215. URL: https://doi.org/10.1145/62212.62215 (visited on 07/14/2024).

[41] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. "Completeness theorems for non-cryptographic fault-tolerant distributed computation". In: *Proceedings of the twentieth annual ACM symposium on Theory of computing*. STOC '88. New York, NY, USA: Association for Computing Machinery, Jan. 1988, pp. 1–10. ISBN: 978-0-89791-264-8. DOI: 10.1145/62212.62213. URL: https://dl.acm.org/doi/10.1145/62212.62213 (visited on 09/19/2024).

[42] Donald Beaver. "Efficient Multiparty Protocols Using Circuit Randomization". en. In: *Advances in Cryptology — CRYPTO '91*. Ed. by Joan Feigenbaum. Berlin, Heidelberg: Springer, 1992, pp. 420–432. ISBN: 978-3-540-46766-3. DOI: 10.1007/3-540-46766-1_34.

[43] Zuzana Beerliová-Trubíniová and Martin Hirt. "Perfectly-Secure MPC with Linear Communication Complexity". en. In: *Theory of Cryptography*. Ed. by Ran Canetti. Berlin, Heidelberg: Springer, 2008, pp. 213–230. ISBN: 978-3-540-78524-8. DOI: 10.1007/978-3-540-78524-8_13.

[44] Marcel Keller, Emmanuela Orsini, and Peter Scholl. "MASCOT: Faster Malicious Arithmetic Secure Computation with Oblivious Transfer". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. CCS '16. New York, NY, USA: Association for Computing Machinery, Oct. 2016, pp. 830–842. ISBN: 978-1-4503-4139-4. DOI: 10.1145/

UNIVERSIDAD DE MÁLAGA

2976749.2978357. URL: https://doi.org/10.1145/2976749.2978357 (visited on 09/19/2024).

[45] D. Beaver, S. Micali, and P. Rogaway. "The round complexity of secure protocols". In: *Proceedings of the twenty-second annual ACM symposium on Theory of Computing*. STOC '90. New York, NY, USA: Association for Computing Machinery, Apr. 1990, pp. 503–513. ISBN: 978-0-89791-361-4. DOI: 10.1145/100216.100287. URL: https://dl.acm.org/doi/10.1145/100216.100287 (visited on 09/19/2024).

[46] Vipul Goyal, Yifan Song, and Chenzhi Zhu. "Guaranteed Output Delivery Comes Free in Honest Majority MPC". en. In: *Advances in Cryptology – CRYPTO 2020*. Ed. by Daniele Micciancio and Thomas Ristenpart. Cham: Springer International Publishing, 2020, pp. 618–646. ISBN: 978-3-030-56880-1. DOI: 10.1007/978-3-030-56880-1_22.

[47] Ivan Damgård et al. "Multiparty Computation from Somewhat Homomorphic Encryption". en. In: *Advances in Cryptology – CRYPTO 2012*. Ed. by Reihaneh Safavi-Naini and Ran Canetti. Berlin, Heidelberg: Springer, 2012, pp. 643–662. ISBN: 978-3-642-32009-5. DOI: 10.1007/978-3-642-32009-5_38.

[48] Ivan Damgård et al. "Practical Covertly Secure MPC for Dishonest Majority – Or: Breaking the SPDZ Limits". en. In: *Computer Security – ESORICS 2013*. Ed. by Jason Crampton, Sushil Jajodia, and Keith Mayes. Berlin, Heidelberg: Springer, 2013, pp. 1–18. ISBN: 978-3-642-40203-6. DOI: 10.1007/978-3-642-40203-6_1.

[49] Marcel Keller, Valerio Pastro, and Dragos Rotaru. "Overdrive: Making SPDZ Great Again". en. In: *Advances in Cryptology – EUROCRYPT 2018*. Ed. by Jesper Buus Nielsen and Vincent Rijmen. Cham: Springer International Publishing, 2018, pp. 158–189. ISBN: 978-3-319-78372-7. DOI: 10.1007/978-3-319-78372-7_6.

[50] Fabrice Benhamouda et al. "Can a Public Blockchain Keep a Secret?" en. In: *Theory of Cryptography*. Ed. by Rafael Pass and Krzysztof Pietrzak. Cham: Springer International Publishing, 2020, pp. 260–290. ISBN: 978-3-030-64375-1. DOI: 10.1007/978-3-030-64375-1_10.

[51] Vipul Goyal et al. "Storing and Retrieving Secrets on a Blockchain". en. In: *Public-Key Cryptography – PKC 2022*. Ed. by Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe. Cham: Springer International Publishing, 2022, pp. 252–282. ISBN: 978-3-030-97121-2. DOI: 10.1007/978-3-030-97121-2_10.

[52] Arka Rai Choudhuri et al. "Fluid MPC: Secure Multiparty Computation with Dynamic Participants". en. In: *Advances in Cryptology – CRYPTO 2021*. Ed. by Tal Malkin and Chris Peikert. Cham: Springer International Publishing, 2021, pp. 94–123. ISBN: 978-3-030-84245-1. DOI: 10.1007/978-3-030-84245-1_4.

[53] Craig Gentry et al. "YOSO: You Only Speak Once". en. In: *Advances in Cryptology – CRYPTO 2021*. Ed. by Tal Malkin and Chris Peikert. Cham: Springer International Publishing, 2021, pp. 64–93. ISBN: 978-3-030-84245-1. DOI: 10.1007/978-3-030-84245-1_3.

[54] Anasuya Acharya et al. "SCALES - MPC with Small Clients and Larger Ephemeral Servers". en. In: *Theory of Cryptography*. Ed. by Eike Kiltz and

UNIVERSIDAD DE MÁLAGA

Vinod Vaikuntanathan. Cham: Springer Nature Switzerland, 2022, pp. 502–531. ISBN: 978-3-031-22365-5. DOI: 10.1007/978-3-031-22365-5_18.

[55] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. *The Bitcoin Backbone Protocol: Analysis and Applications*. Report Number: 765. 2014. URL: https://eprint.iacr.org/2014/765 (visited on 07/07/2023).

[56] Leslie Lamport, Robert Shostak, and Marshall Pease. "The Byzantine Generals Problem". In: *ACM Trans. Program. Lang. Syst.* 4.3 (July 1982), pp. 382–401. ISSN: 0164-0925. DOI: 10.1145/357172.357176. URL: https://doi.org/10.1145/357172.357176 (visited on 07/11/2024).

[57] Vitalik Buterin. *Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform*. en. 2014.

[58] Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. en. 2008. URL: https://bitcoin.org/en/bitcoin-paper (visited on 07/11/2024).

[59] Alexey Pertsev, Roman Semenov, and Roman Storm. "Tornado Cash Privacy Solution Version 1.4". en. In: (2019).

[60] Benedikt Bünz et al. "Zether: Towards Privacy in a Smart Contract World". en. In: *Financial Cryptography and Data Security*. Ed. by Joseph Bonneau and Nadia Heninger. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 423–443. ISBN: 978-3-030-51280-4. DOI: 10.1007/978-3-030-51280-4_23.

[61] Rujia Li et al. "SoK: TEE-Assisted Confidential Smart Contract". en. In: *Proceedings on Privacy Enhancing Technologies* 2022.3 (July 2022), pp. 711–731. ISSN: 2299-0984. DOI: 10.56553/popets-2022-0093. URL: https://petsymposium.org/popets/2022/popets-2022-0093.php (visited on 07/26/2023).

[62] Morten Dahl et al. *fhEVM: Confidential EVM Smart Contracts using Fully Homomorphic Encryption*. 2023. (Visited on 11/24/2023).

[63] Assaf Ben-David, Noam Nisan, and Benny Pinkas. "FairplayMP: a system for secure multi-party computation". en. In: *Proceedings of the 15th ACM conference on Computer and communications security*. Alexandria Virginia USA: ACM, Oct. 2008, pp. 257–266. ISBN: 978-1-59593-810-7. DOI: 10.1145/1455770.1455804. URL: https://dl.acm.org/doi/10.1145/1455770.1455804 (visited on 07/23/2024).

[64] Nigel P. Smart. "Practical and Efficient FHE-Based MPC". en. In: *Cryptography and Coding*. Ed. by Elizabeth A. Quaglia. Cham: Springer Nature Switzerland, 2024, pp. 263–283. ISBN: 978-3-031-47818-5. DOI: 10.1007/978-3-031-47818-5_14.

[65] Yehuda Lindell et al. *The Deployment Dilemma: Merits & Challenges of Deploying MPC*. en. Sept. 2023. URL: https://mpc.cs.berkeley.edu/blog/deployment-dilemma.html (visited on 08/11/2024).

[66] Emma Dauterman et al. "Reflections on trusting distributed trust". In: *Proceedings of the 21st ACM Workshop on Hot Topics in Networks*. HotNets '22. New York, NY, USA: Association for Computing Machinery, Nov. 2022, pp. 38–45. ISBN: 978-1-4503-9899-2. DOI: 10.1145/3563766.3564089. URL: https://doi.org/10.1145/3563766.3564089 (visited on 07/11/2024).

[67] Darya Kaviani et al. *Flock: A Framework for Deploying On-Demand Distributed Trust*. Publication info: Published elsewhere. OSDI 2024. 2024. URL: https://eprint.iacr.org/2024/892 (visited on 07/11/2024).

UNIVERSIDAD DE MÁLAGA

[68] James Reyes. *Building the next generation of digital advertising with MPC*. en. 2022.

[69] Emanuele Bellini, Youssef Iraqi, and Ernesto Damiani. "Blockchain-Based Distributed Trust and Reputation Management Systems: A Survey". In: *IEEE Access* 8 (2020). Conference Name: IEEE Access, pp. 21127–21151. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.2969820. URL: https://ieeexplore.ieee.org/document/8970496/?arnumber=8970496&tag=1 (visited on 08/12/2024).

[70] Ignacio Cascudo and Bernardo David. "SCRAPE: Scalable Randomness Attested by Public Entities". en. In: *Applied Cryptography and Network Security*. Ed. by Dieter Gollmann, Atsuko Miyaji, and Hiroaki Kikuchi. Vol. 10355. Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, 2017, pp. 537–556. ISBN: 978-3-319-61203-4 978-3-319-61204-1. DOI: 10.1007/978-3-319-61204-1_27. URL: https://link.springer.com/10.1007/978-3-319-61204-1_27 (visited on 07/29/2024).

[71] Ewa Syta et al. *Scalable Bias-Resistant Distributed Randomness*. Publication info: Preprint. MINOR revision. 2016. URL: https://eprint.iacr.org/2016/1067 (visited on 07/29/2024).

[72] Philipp Schindler et al. "HydRand: Efficient Continuous Distributed Randomness". In: *2020 IEEE Symposium on Security and Privacy (SP)*. ISSN: 2375-1207. May 2020, pp. 73–89. DOI: 10.1109/SP40000.2020.00003. URL: https://ieeexplore.ieee.org/document/9152802 (visited on 07/29/2024).

[73] randao.org. *RANDAO*. Tech. rep. randao.org, 2017.

[74] Thanh Nguyen-Van et al. "Scalable Distributed Random Number Generation Based on Homomorphic Encryption". In: *2019 IEEE International Conference on Blockchain (Blockchain)*. July 2019, pp. 572–579. DOI: 10.1109/Blockchain.2019.00083. URL: https://ieeexplore.ieee.org/document/8946121/?arnumber=8946121 (visited on 07/29/2024).

[75] Benny Pinkas et al. "Secure Two-Party Computation Is Practical". en. In: *Advances in Cryptology – ASIACRYPT 2009*. Ed. by Mitsuru Matsui. Berlin, Heidelberg: Springer, 2009, pp. 250–267. ISBN: 978-3-642-10366-7. DOI: 10.1007/978-3-642-10366-7_15.

[76] Ivan Damgård et al. "Implementing AES via an Actively/Covertly Secure Dishonest-Majority MPC Protocol". en. In: *Security and Cryptography for Networks*. Ed. by David Hutchison et al. Vol. 7485. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 241–263. ISBN: 978-3-642-32927-2 978-3-642-32928-9. DOI: 10.1007/978-3-642-32928-9_14. URL: http://link.springer.com/10.1007/978-3-642-32928-9_14 (visited on 07/29/2024).

[77] Marcel Keller et al. "Faster Secure Multi-party Computation of AES and DES Using Lookup Tables". en. In: *Applied Cryptography and Network Security*. Ed. by Dieter Gollmann, Atsuko Miyaji, and Hiroaki Kikuchi. Cham: Springer International Publishing, 2017, pp. 229–249. ISBN: 978-3-319-61204-1. DOI: 10.1007/978-3-319-61204-1_12.

[78] F. Betül Durak and Jorge Guajardo. "Improving the Efficiency of AES Protocols in Multi-Party Computation". en. In: *Financial Cryptography and Data Security*. Ed. by Nikita Borisov and Claudia Diaz. Berlin, Heidelberg:

UNIVERSIDAD DE MÁLAGA

Springer, 2021, pp. 229–248. ISBN: 978-3-662-64322-8. DOI: 10.1007/978-3-662-64322-8_11.

[79] F. Betül Durak and Jorge Guajardo. "Improving the Efficiency of AES Protocols in Multi-Party Computation". en. In: *Financial Cryptography and Data Security*. Ed. by Nikita Borisov and Claudia Diaz. Berlin, Heidelberg: Springer, 2021, pp. 229–248. ISBN: 978-3-662-64322-8. DOI: 10.1007/978-3-662-64322-8_11.

[80] Lorenzo Grassi et al. "MPC-Friendly Symmetric Key Primitives". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. CCS '16. New York, NY, USA: Association for Computing Machinery, Oct. 2016, pp. 430–443. ISBN: 978-1-4503-4139-4. DOI: 10.1145/2976749.2978332. URL: https://doi.org/10.1145/2976749.2978332 (visited on 07/09/2024).

[81] Victor Shoup. "Practical Threshold Signatures". en. In: *Advances in Cryptology — EUROCRYPT 2000*. Ed. by Gerhard Goos et al. Vol. 1807. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 207–220. ISBN: 978-3-540-67517-4 978-3-540-45539-4. DOI: 10.1007/3-540-45539-6_15. URL: http://link.springer.com/10.1007/3-540-45539-6_15 (visited on 07/28/2024).

[82] Rosario Gennaro et al. "Robust and Efficient Sharing of RSA Functions". en. In: *Journal of Cryptology* 13.2 (Mar. 2000), pp. 273–300. ISSN: 1432-1378. DOI: 10.1007/s001459910011. URL: https://doi.org/10.1007/s001459910011 (visited on 07/28/2024).

[83] Rosario Gennaro, Steven Goldfeder, and Arvind Narayanan. "Threshold-Optimal DSA/ECDSA Signatures and an Application to Bitcoin Wallet Security". en. In: *Applied Cryptography and Network Security*. Ed. by Mark Manulis, Ahmad-Reza Sadeghi, and Steve Schneider. Vol. 9696. Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016, pp. 156–174. ISBN: 978-3-319-39554-8 978-3-319-39555-5. DOI: 10.1007/978-3-319-39555-5_9. URL: https://link.springer.com/10.1007/978-3-319-39555-5_9 (visited on 07/28/2024).

[84] Rosario Gennaro and Steven Goldfeder. "Fast Multiparty Threshold ECDSA with Fast Trustless Setup". In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. CCS '18. New York, NY, USA: Association for Computing Machinery, Oct. 2018, pp. 1179–1194. ISBN: 978-1-4503-5693-0. DOI: 10.1145/3243734.3243859. URL: https://doi.org/10.1145/3243734.3243859 (visited on 07/28/2024).

[85] Yehuda Lindell and Ariel Nof. "Fast Secure Multiparty ECDSA with Practical Distributed Key Generation and Applications to Cryptocurrency Custody". en. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. Toronto Canada: ACM, Oct. 2018, pp. 1837–1854. ISBN: 978-1-4503-5693-0. DOI: 10.1145/3243734.3243788. URL: https://dl.acm.org/doi/10.1145/3243734.3243788 (visited on 07/28/2024).

[86] Dan Boneh, Rosario Gennaro, and Steven Goldfeder. "Using Level-1 Homomorphic Encryption to Improve Threshold DSA Signatures for Bitcoin Wallet Security". en. In: *Progress in Cryptology – LATINCRYPT 2017*. Ed. by Tanja Lange and Orr Dunkelman. Cham: Springer International Publishing, 2019,

UNIVERSIDAD DE MÁLAGA

pp. 352–377. ISBN: 978-3-030-25283-0. DOI: 10.1007/978-3-030-25283-0_19.

[87] Douglas R. Stinson and Reto Strobl. "Provably Secure Distributed Schnorr Signatures and a (t, n) Threshold Scheme for Implicit Certificates". en. In: *Information Security and Privacy*. Ed. by Vijay Varadharajan and Yi Mu. Berlin, Heidelberg: Springer, 2001, pp. 417–434. ISBN: 978-3-540-47719-8. DOI: 10.1007/3-540-47719-5_33.

[88] Rosario Gennaro et al. "Secure Applications of Pedersen's Distributed Key Generation Protocol". en. In: *Topics in Cryptology — CT-RSA 2003*. Ed. by Marc Joye. Berlin, Heidelberg: Springer, 2003, pp. 373–390. ISBN: 978-3-540-36563-1. DOI: 10.1007/3-540-36563-X_26.

[89] Chelsea Komlo and Ian Goldberg. "FROST: Flexible Round-Optimized Schnorr Threshold Signatures". en. In: *Selected Areas in Cryptography*. Ed. by Orr Dunkelman, Michael J. Jacobson, and Colin O'Flynn. Vol. 12804. Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, 2021, pp. 34–65. ISBN: 978-3-030-81651-3 978-3-030-81652-0. DOI: 10.1007/978-3-030-81652-0_2. URL: https://link.springer.com/10.1007/978-3-030-81652-0_2 (visited on 07/28/2024).

[90] Dan Boneh, Ben Lynn, and Hovav Shacham. "Short Signatures from the Weil Pairing". en. In: *Advances in Cryptology — ASIACRYPT 2001*. Ed. by Colin Boyd. Berlin, Heidelberg: Springer, 2001, pp. 514–532. ISBN: 978-3-540-45682-7. DOI: 10.1007/3-540-45682-1_30.

[91] Alexandra Boldyreva. "Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme". en. In: *Public Key Cryptography — PKC 2003*. Ed. by Yvo G. Desmedt. Berlin, Heidelberg: Springer, 2002, pp. 31–46. ISBN: 978-3-540-36288-3. DOI: 10.1007/3-540-36288-6_3.

[92] Sara Ricci et al. "Threshold Signature for Privacy-Preserving Blockchain". en. In: *Business Process Management: Blockchain, Robotic Process Automation, and Central and Eastern Europe Forum*. Ed. by Andrea Marrella et al. Cham: Springer International Publishing, 2022, pp. 100–115. ISBN: 978-3-031-16168-1. DOI: 10.1007/978-3-031-16168-1_7.

[93] Ran Canetti and Shafi Goldwasser. "An Efficient threshold Public Key Cryptosystem Secure Against Adaptive Chosen Ciphertext Attack (Extended Abstract)". en. In: *Advances in Cryptology — EUROCRYPT '99*. Ed. by Jacques Stern. Berlin, Heidelberg: Springer, 1999, pp. 90–106. ISBN: 978-3-540-48910-8. DOI: 10.1007/3-540-48910-X_7.

[94] Rikke Bendlin and Ivan Damgård. "Threshold Decryption and Zero-Knowledge Proofs for Lattice-Based Cryptosystems". en. In: *Theory of Cryptography*. Ed. by Daniele Micciancio. Berlin, Heidelberg: Springer, 2010, pp. 201–218. ISBN: 978-3-642-11799-2. DOI: 10.1007/978-3-642-11799-2_13.

[95] Morten Dahl et al. *Noah's Ark: Efficient Threshold-FHE Using Noise Flooding*. Publication info: Published elsewhere. Major revision. WAHC 2023. 2023. URL: https://eprint.iacr.org/2023/815 (visited on 09/22/2024).

[96] Hossein Ghodosi, Josef Pieprzyk, and Rei Safavi-Naini. "Dynamic Threshold Cryptosystems (A New Scheme in Group Oriented Cryptography)". en. In: *Proceedings of Pragocrypt 1996*. 1996, pp. 370–379.

UNIVERSIDAD DE MÁLAGA

[97] Cécile Delerablée and David Pointcheval. "Dynamic Threshold Public-Key Encryption". en. In: *Advances in Cryptology – CRYPTO 2008*. Ed. by David Wagner. Vol. 5157. ISSN: 0302-9743, 1611-3349 Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 317–334. ISBN: 978-3-540-85173-8 978-3-540-85174-5. DOI: 10.1007/978-3-540-85174-5_18. URL: http://link.springer.com/10.1007/978-3-540-85174-5_18 (visited on 07/29/2024).

[98] Bo Qin et al. "Threshold Public-Key Encryption with Adaptive Security and Short Ciphertexts". en. In: *Information and Communications Security*. Ed. by Miguel Soriano, Sihan Qing, and Javier López. Vol. 6476. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 62–76. ISBN: 978-3-642-17649-4 978-3-642-17650-0. DOI: 10.1007/978-3-642-17650-0_6. URL: http://link.springer.com/10.1007/978-3-642-17650-0_6 (visited on 07/29/2024).

[99] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. "Efficient Private Matching and Set Intersection". en. In: *Advances in Cryptology - EUROCRYPT 2004*. Ed. by Christian Cachin and Jan L. Camenisch. Berlin, Heidelberg: Springer, 2004, pp. 1–19. ISBN: 978-3-540-24676-3. DOI: 10.1007/978-3-540-24676-3_1.

[100] Justin Brickell and Vitaly Shmatikov. "Privacy-Preserving Graph Algorithms in the Semi-honest Model". en. In: *Advances in Cryptology - ASIACRYPT 2005*. Ed. by Bimal Roy. Berlin, Heidelberg: Springer, 2005, pp. 236–252. ISBN: 978-3-540-32267-2. DOI: 10.1007/11593447_13.

[101] Keith Frikken. "Privacy-Preserving Set Union". en. In: *Applied Cryptography and Network Security*. Ed. by David Hutchison et al. Vol. 4521. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 237–252. ISBN: 978-3-540-72737-8 978-3-540-72738-5. DOI: 10.1007/978-3-540-72738-5_16. URL: http://link.springer.com/10.1007/978-3-540-72738-5_16 (visited on 08/03/2024).

[102] Jae Hong Seo, Jung Hee Cheon, and Jonathan Katz. "Constant-Round Multiparty Private Set Union Using Reversed Laurent Series". en. In: *Public Key Cryptography – PKC 2012*. Ed. by David Hutchison et al. Vol. 7293. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 398–412. ISBN: 978-3-642-30056-1 978-3-642-30057-8. DOI: 10.1007/978-3-642-30057-8_24. URL: http://link.springer.com/10.1007/978-3-642-30057-8_24 (visited on 08/03/2024).

[103] Lea Kissner and Dawn Song. "Privacy-Preserving Set Operations". en. In: *Advances in Cryptology – CRYPTO 2005*. Ed. by Victor Shoup. Berlin, Heidelberg: Springer, 2005, pp. 241–257. ISBN: 978-3-540-31870-5. DOI: 10.1007/11535218_15.

[104] Carmit Hazay and Kobbi Nissim. "Efficient Set Operations in the Presence of Malicious Adversaries". en. In: *Journal of Cryptology* 25.3 (July 2012), pp. 383–433. ISSN: 1432-1378. DOI: 10.1007/s00145-011-9098-x. URL: https://doi.org/10.1007/s00145-011-9098-x (visited on 08/03/2024).

[105] Emiliano De Cristofaro, Paolo Gasti, and Gene Tsudik. "Fast and Private Computation of Cardinality of Set Intersection and Union". en. In: *Cryptology and Network Security*. Ed. by Josef Pieprzyk, Ahmad-Reza Sadeghi, and

UNIVERSIDAD DE MÁLAGA

Mark Manulis. Berlin, Heidelberg: Springer, 2012, pp. 218–231. ISBN: 978-3-642-35404-5. DOI: 10.1007/978-3-642-35404-5_17.

[106] Sven Laur, Helger Lipmaa, and Taneli Mielikäinen. "Private Itemset Support Counting". en. In: *Information and Communications Security*. Ed. by Sihan Qing et al. Berlin, Heidelberg: Springer, 2005, pp. 97–111. ISBN: 978-3-540-32099-9. DOI: 10.1007/11602897_9.

[107] Liyan Shen et al. "Efficient and Private Set Intersection of Human Genomes". In: *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. Madrid, Spain: IEEE, Dec. 2018, pp. 761–764. ISBN: 978-1-5386-5488-0. DOI: 10.1109/BIBM.2018.8621291. URL: https://ieeexplore.ieee.org/document/8621291/ (visited on 09/19/2024).

[108] Ye Li et al. "Outsourced privacy-preserving C4.5 decision tree algorithm over horizontally and vertically partitioned dataset among multiple parties". en. In: *Cluster Computing* 22.S1 (Jan. 2019), pp. 1581–1593. ISSN: 1386-7857, 1573-7543. DOI: 10.1007/s10586-017-1019-9. URL: http://link.springer.com/10.1007/s10586-017-1019-9 (visited on 09/19/2024).

[109] Shishir Nagaraja et al. "BotGrep: finding P2P bots with structured graph analysis". In: *Proceedings of the 19th USENIX conference on Security*. USENIX Security'10. USA: USENIX Association, Aug. 2010, p. 7. (Visited on 09/19/2024).

[110] Liina Kamm and Jan Willemson. "Secure floating point arithmetic and private satellite collision analysis". In: *Int. J. Inf. Secur.* 14.6 (Nov. 2015), pp. 531–548. ISSN: 1615-5262. DOI: 10.1007/s10207-014-0271-8. URL: https://doi.org/10.1007/s10207-014-0271-8 (visited on 09/19/2024).

[111] Dominik Roy George and Savio Sciancalepore. "PRM - Private Interference Discovery for IEEE 802.15. 4 Networks". In: *2022 IEEE Conference on Communications and Network Security (CNS)*. Austin, TX, USA: IEEE, Oct. 2022, pp. 136–144. ISBN: 978-1-66546-255-6. DOI: 10.1109/CNS56114.2022.9947236. URL: https://ieeexplore.ieee.org/document/9947236/ (visited on 09/19/2024).

[112] Lucy Li et al. "Protocols for Checking Compromised Credentials". In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. CCS '19. New York, NY, USA: Association for Computing Machinery, Nov. 2019, pp. 1387–1403. ISBN: 978-1-4503-6747-9. DOI: 10.1145/3319535.3354229. URL: https://dl.acm.org/doi/10.1145/3319535.3354229 (visited on 09/19/2024).

[113] Daniel Kales et al. "Mobile Private Contact Discovery at Scale". en. In: ().

[114] Zakaria Gheid, Yacine Challal, and Lin Chen. "Private and efficient set intersection protocol for RFID-based food adequacy check". In: *2018 IEEE Wireless Communications and Networking Conference (WCNC)*. Barcelona: IEEE, Apr. 2018, pp. 1–6. ISBN: 978-1-5386-1734-2. DOI: 10.1109/WCNC.2018.8377207. URL: https://ieeexplore.ieee.org/document/8377207/ (visited on 09/19/2024).

[115] Ni Trieu et al. *Epione: Lightweight Contact Tracing with Strong Privacy*. arXiv:2004.13293 [cs]. May 2020. DOI: 10.48550/arXiv.2004.13293. URL: http://arxiv.org/abs/2004.13293 (visited on 09/07/2024).

[116] Ulrich Matchi Aïvodji et al. "SRide: A Privacy-Preserving Ridesharing System". In: *Proceedings of the 11th ACM Conference on Security & Privacy in Wireless and Mobile Networks*. WiSec '18. New York, NY, USA: Association

UNIVERSIDAD DE MÁLAGA

for Computing Machinery, June 2018, pp. 40–50. ISBN: 978-1-4503-5731-9. DOI: 10.1145/3212480.3212483. URL: https://dl.acm.org/doi/10.1145/3212480.3212483 (visited on 09/19/2024).

[117] Yongjun Zhao and Sherman S.M. Chow. "Can You Find The One for Me?" In: *Proceedings of the 2018 Workshop on Privacy in the Electronic Society*. WPES'18. New York, NY, USA: Association for Computing Machinery, Jan. 2018, pp. 54–65. ISBN: 978-1-4503-5989-4. DOI: 10.1145/3267323.3268965. URL: https://dl.acm.org/doi/10.1145/3267323.3268965 (visited on 09/19/2024).

[118] Xu An Wang et al. "A privacy-preserving fuzzy interest matching protocol for friends finding in social networks". en. In: *Soft Computing* 22.8 (Apr. 2018), pp. 2517–2526. ISSN: 1433-7479. DOI: 10.1007/s00500-017-2506-x. URL: https://doi.org/10.1007/s00500-017-2506-x (visited on 08/07/2024).

[119] Xavier Carpent et al. "Private Set Projections & Variants". en. In: *Proceedings of the 2017 on Workshop on Privacy in the Electronic Society*. Dallas Texas USA: ACM, Oct. 2017, pp. 87–98. ISBN: 978-1-4503-5175-1. DOI: 10.1145/3139550.3139554. URL: https://dl.acm.org/doi/10.1145/3139550.3139554 (visited on 08/07/2024).

[120] Alex Davidson and Carlos Cid. "An Efficient Toolkit for Computing Private Set Operations". en. In: *Information Security and Privacy*. Ed. by Josef Pieprzyk and Suriadi Suriadi. Cham: Springer International Publishing, 2017, pp. 261–278. ISBN: 978-3-319-59870-3. DOI: 10.1007/978-3-319-59870-3_15.

[121] Hao Chen, Kim Laine, and Peter Rindal. "Fast Private Set Intersection from Homomorphic Encryption". en. In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. Dallas Texas USA: ACM, Oct. 2017, pp. 1243–1255. ISBN: 978-1-4503-4946-8. DOI: 10.1145/3133956.3134061. URL: https://dl.acm.org/doi/10.1145/3133956.3134061 (visited on 08/07/2024).

[122] Yuting Jiang, Jianghong Wei, and Jing Pan. "Publicly Verifiable Private Set Intersection from Homomorphic Encryption". en. In: *Security and Privacy in Social Networks and Big Data*. Ed. by Xiaofeng Chen, Xinyi Huang, and Mirosław Kutyłowski. Singapore: Springer Nature, 2022, pp. 117–137. ISBN: 978-981-19724-2-3. DOI: 10.1007/978-981-19-7242-3_8.

[123] Peter Rindal and Mike Rosulek. "Improved Private Set Intersection Against Malicious Adversaries". en. In: *Advances in Cryptology – EUROCRYPT 2017*. Ed. by Jean-Sébastien Coron and Jesper Buus Nielsen. Cham: Springer International Publishing, 2017, pp. 235–259. ISBN: 978-3-319-56620-7. DOI: 10.1007/978-3-319-56620-7_9.

[124] Tao Jiang and Xu Yuan. "Traceable Private Set Intersection in Cloud Computing". In: *2019 IEEE Conference on Dependable and Secure Computing (DSC)*. Nov. 2019, pp. 1–7. DOI: 10.1109/DSC47296.2019.8937666. URL: https://ieeexplore.ieee.org/document/8937666 (visited on 08/06/2024).

[125] Brett Hemenway Falk, Daniel Noble, and Rafail Ostrovsky. "Private Set Intersection with Linear Communication from General Assumptions". en. In: *Proceedings of the 18th ACM Workshop on Privacy in the Electronic Society*. London United Kingdom: ACM, Nov. 2019, pp. 14–25. ISBN: 978-1-4503-

UNIVERSIDAD DE MÁLAGA

6830-8. DOI: 10.1145/3338498.3358645. URL: https://dl.acm.org/doi/10.1145/3338498.3358645 (visited on 08/06/2024).

[126] Benny Pinkas, Thomas Schneider, and Michael Zohner. "Faster Private Set Intersection Based on OT Extension". en. In: *23rd USENIX Security Symposium (USENIX Security 14)*. 2014, pp. 797–812. ISBN: 978-1-931971-15-7. URL: https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/pinkas (visited on 08/05/2024).

[127] Peter Rindal and Mike Rosulek. "Malicious-Secure Private Set Intersection via Dual Execution". en. In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. Dallas Texas USA: ACM, Oct. 2017, pp. 1229–1242. ISBN: 978-1-4503-4946-8. DOI: 10.1145/3133956.3134044. URL: https://dl.acm.org/doi/10.1145/3133956.3134044 (visited on 08/05/2024).

[128] Daniel Kales et al. "Mobile Private Contact Discovery at Scale". en. In: *Proceedings of the 28th USENIX Security Symposium* (2019).

[129] Agnes Kiss et al. "Private Set Intersection for Unequal Set Sizes with Mobile Applications". en. In: *Proceedings on Privacy Enhancing Technologies* 2017.4 (Oct. 2017), pp. 177–197. ISSN: 2299-0984. DOI: 10.1515/popets-2017-0044. (Visited on 08/05/2024).

[130] Benny Pinkas, Thomas Schneider, and Michael Zohner. "Scalable Private Set Intersection Based on OT Extension". en. In: *ACM Transactions on Privacy and Security* 21.2 (May 2018), pp. 1–35. ISSN: 2471-2566, 2471-2574. DOI: 10.1145/3154794. URL: https://dl.acm.org/doi/10.1145/3154794 (visited on 08/05/2024).

[131] Gayathri Garimella et al. "Oblivious Key-Value Stores and Amplification for Private Set Intersection". en. In: *Advances in Cryptology – CRYPTO 2021*. Ed. by Tal Malkin and Chris Peikert. Vol. 12826. Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, 2021, pp. 395–425. ISBN: 978-3-030-84244-4 978-3-030-84245-1. DOI: 10.1007/978-3-030-84245-1_14. URL: https://link.springer.com/10.1007/978-3-030-84245-1_14 (visited on 08/06/2024).

[132] Benny Pinkas et al. "PSI from PaXoS: Fast, Malicious Private Set Intersection". en. In: *Advances in Cryptology – EUROCRYPT 2020*. Ed. by Anne Canteaut and Yuval Ishai. Cham: Springer International Publishing, 2020, pp. 739–767. ISBN: 978-3-030-45724-2. DOI: 10.1007/978-3-030-45724-2_25.

[133] Peter Rindal and Phillipp Schoppmann. "VOLE-PSI: Fast OPRF and Circuit-PSI from Vector-OLE". en. In: *Advances in Cryptology – EUROCRYPT 2021*. Ed. by Anne Canteaut and François-Xavier Standaert. Vol. 12697. Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, 2021, pp. 901–930. ISBN: 978-3-030-77885-9 978-3-030-77886-6. DOI: 10.1007/978-3-030-77886-6_31. URL: https://link.springer.com/10.1007/978-3-030-77886-6_31 (visited on 08/06/2024).

[134] Srinivasan Raghuraman and Peter Rindal. "Blazing Fast PSI from Improved OKVS and Subfield VOLE". In: *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. CCS '22. New York, NY, USA: Association for Computing Machinery, Nov. 2022, pp. 2505–2517.

UNIVERSIDAD DE MÁLAGA

ISBN: 978-1-4503-9450-5. DOI: 10.1145/3548606.3560658. URL: https://dl.acm.org/doi/10.1145/3548606.3560658 (visited on 08/06/2024).

[135] Xuefei Cao et al. "A two-party privacy preserving set intersection protocol against malicious users in cloud computing". In: *Computer Standards & Interfaces*. SI: CCSP-SR 54 (Nov. 2017), pp. 41–45. ISSN: 0920-5489. DOI: 10.1016/j.csi.2016.08.004. URL: https://www.sciencedirect.com/science/article/pii/S0920548916300617 (visited on 08/19/2024).

[136] Yamin Wen et al. "Intersection-policy private mutual authentication from authorized private set intersection". en. In: *Science China Information Sciences* 63.2 (Jan. 2020), p. 122101. ISSN: 1869-1919. DOI: 10.1007/s11432-019-9907-x. URL: https://doi.org/10.1007/s11432-019-9907-x (visited on 09/20/2024).

[137] Andrea Cerulli, Emiliano De Cristofaro, and Claudio Soriente. "Nothing Refreshes Like a RePSI: Reactive Private Set Intersection". en. In: *Applied Cryptography and Network Security*. Ed. by Bart Preneel and Frederik Vercauteren. Cham: Springer International Publishing, 2018, pp. 280–300. ISBN: 978-3-319-93387-0. DOI: 10.1007/978-3-319-93387-0_15.

[138] Siyi Lv et al. "Unbalanced private set intersection cardinality protocol with low communication cost". In: *Future Generation Computer Systems* 102 (Jan. 2020), pp. 1054–1061. ISSN: 0167-739X. DOI: 10.1016/j.future.2019.09.022. URL: https://www.sciencedirect.com/science/article/pii/S0167739X19316413 (visited on 09/20/2024).

[139] Changyu Dong and Grigorios Loukides. "Approximating Private Set Union/Intersection Cardinality With Logarithmic Complexity". In: *IEEE Transactions on Information Forensics and Security* 12.11 (Nov. 2017). Conference Name: IEEE Transactions on Information Forensics and Security, pp. 2792–2806. ISSN: 1556-6021. DOI: 10.1109/TIFS.2017.2721360. URL: https://ieeexplore.ieee.org/document/7961247 (visited on 09/20/2024).

[140] Sumit Kumar Debnath and Ratna Dutta. "Provably Secure Fair Mutual Private Set Intersection Cardinality Utilizing Bloom Filter". en. In: *Information Security and Cryptology*. Ed. by Kefei Chen, Dongdai Lin, and Moti Yung. Cham: Springer International Publishing, 2017, pp. 505–525. ISBN: 978-3-319-54705-3. DOI: 10.1007/978-3-319-54705-3_31.

[141] Mihaela Ion et al. "On Deploying Secure Computing: Private Intersection-Sum-with-Cardinality". In: *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*. Genoa, Italy: IEEE, Sept. 2020, pp. 370–389. ISBN: 978-1-72815-087-1. DOI: 10.1109/EuroSP48549.2020.00031. URL: https://ieeexplore.ieee.org/document/9230369/ (visited on 09/19/2024).

[142] Bailey Kacsmar et al. "Differentially Private Two-Party Set Operations". In: *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*. Sept. 2020, pp. 390–404. DOI: 10.1109/EuroSP48549.2020.00032. URL: https://ieeexplore.ieee.org/document/9230399 (visited on 09/20/2024).

[143] Sumit Kumar Debnath et al. "Post-quantum protocol for computing set intersection cardinality with linear complexity". en. In: *IET Information Security* 14.6 (2020), pp. 661–669. ISSN: 1751-8717. DOI: 10.1049/iet-ifs.2019.0315. URL: https://onlinelibrary.wiley.com/doi/abs/10.1049/iet-ifs.2019.0315 (visited on 09/20/2024).

UNIVERSIDAD DE MÁLAGA

[144] Michele Ciampi and Claudio Orlandi. "Combining Private Set-Intersection with Secure Two-Party Computation". en. In: *Security and Cryptography for Networks*. Ed. by Dario Catalano and Roberto De Prisco. Cham: Springer International Publishing, 2018, pp. 464–482. ISBN: 978-3-319-98113-0. DOI: 10.1007/978-3-319-98113-0_25.

[145] Benny Pinkas et al. "Efficient Circuit-Based PSI via Cuckoo Hashing". en. In: *Advances in Cryptology – EUROCRYPT 2018*. Ed. by Jesper Buus Nielsen and Vincent Rijmen. Cham: Springer International Publishing, 2018, pp. 125–157. ISBN: 978-3-319-78372-7. DOI: 10.1007/978-3-319-78372-7_5.

[146] Sara Ramezanian, Tommi Meskanen, and Valtteri Niemi. "Multi-party Private Set Operations with an External Decider". en. In: *Data and Applications Security and Privacy XXXV*. Ed. by Ken Barker and Kambiz Ghazinour. Vol. 12840. Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, 2021, pp. 117–135. ISBN: 978-3-030-81241-6 978-3-030-81242-3. DOI: 10.1007/978-3-030-81242-3_7. URL: https://link.springer.com/10.1007/978-3-030-81242-3_7 (visited on 09/20/2024).

[147] Sumit Kumar Debnath et al. "Post-quantum secure multi-party private set-intersection in star network topology". In: *Journal of Information Security and Applications* 58 (May 2021), p. 102731. ISSN: 2214-2126. DOI: 10.1016/j.jisa.2020.102731. URL: https://www.sciencedirect.com/science/article/pii/S2214212620308668 (visited on 09/20/2024).

[148] En Zhang et al. "Efficient Multi-Party Private Set Intersection Against Malicious Adversaries". In: *Proceedings of the 2019 ACM SIGSAC Conference on Cloud Computing Security Workshop*. CCSW'19. New York, NY, USA: Association for Computing Machinery, Nov. 2019, pp. 93–104. ISBN: 978-1-4503-6826-1. DOI: 10.1145/3338466.3358927. URL: https://doi.org/10.1145/3338466.3358927 (visited on 09/20/2024).

[149] Alireza Kavousi, Javad Mohajeri, and Mahmoud Salmasizadeh. "Efficient Scalable Multi-party Private Set Intersection Using Oblivious PRF". In: *Security and Trust Management: 17th International Workshop, STM 2021, Darmstadt, Germany, October 8, 2021, Proceedings*. Berlin, Heidelberg, Oct. 2021, pp. 81–99. ISBN: 978-3-030-91858-3. DOI: 10.1007/978-3-030-91859-0_5. URL: https://doi.org/10.1007/978-3-030-91859-0_5 (visited on 09/20/2024).

[150] Aner Ben-Efraim et al. "PSImple: Practical Multiparty Maliciously-Secure Private Set Intersection". In: *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*. ASIA CCS '22. New York, NY, USA: Association for Computing Machinery, May 2022, pp. 1098–1112. ISBN: 978-1-4503-9140-5. DOI: 10.1145/3488932.3523254. URL: https://doi.org/10.1145/3488932.3523254 (visited on 09/20/2024).

[151] Vladimir Kolesnikov et al. "Practical Multi-party Private Set Intersection from Symmetric-Key Techniques". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. CCS '17. New York, NY, USA: Association for Computing Machinery, Oct. 2017, pp. 1257–1272. ISBN: 978-1-4503-4946-8. DOI: 10.1145/3133956.3134065. URL: https://dl.acm.org/doi/10.1145/3133956.3134065 (visited on 09/20/2024).

UNIVERSIDAD DE MÁLAGA

[152] Nishanth Chandran et al. "Efficient Linear Multiparty PSI and Extensions to Circuit/Quorum PSI". In: *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. CCS '21. New York, NY, USA: Association for Computing Machinery, Nov. 2021, pp. 1182–1204. ISBN: 978-1-4503-8454-4. DOI: 10.1145/3460120.3484591. URL: https://doi.org/10.1145/3460120.3484591 (visited on 09/20/2024).

[153] Lifei Wei et al. "Efficient and Collusion Resistant Multi-party Private Set Intersection Protocols for Large Participants and Small Sets Setting". en. In: ed. by Xiaofeng Chen, Jian Shen, and Willy Susilo. Vol. 13547. Book Title: Cyberspace Safety and Security Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, 2022, pp. 118–132. ISBN: 978-3-031-18066-8 978-3-031-18067-5. DOI: 10.1007/978-3-031-18067-5_9. URL: https://link.springer.com/10.1007/978-3-031-18067-5_9 (visited on 09/20/2024).

[154] Ofri Nevo, Ni Trieu, and Avishay Yanai. "Simple, Fast Malicious Multiparty Private Set Intersection". In: *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. CCS '21. New York, NY, USA: Association for Computing Machinery, Nov. 2021, pp. 1151–1165. ISBN: 978-1-4503-8454-4. DOI: 10.1145/3460120.3484772. URL: https://dl.acm.org/doi/10.1145/3460120.3484772 (visited on 09/20/2024).

[155] Xiaoyuan Yang et al. "Improved outsourced private set intersection protocol based on polynomial interpolation". en. In: *Concurrency and Computation: Practice and Experience* 30.1 (2018), e4329. ISSN: 1532-0634. DOI: 10.1002/cpe.4329. (Visited on 09/21/2024).

[156] Ou Ruan, Xiongbo Huang, and Hao Mao. "An Efficient Private Set Intersection Protocol for the Cloud Computing Environments". In: *2020 IEEE 6th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)*. Baltimore, MD, USA: IEEE, May 2020, pp. 254–259. ISBN: 978-1-72816-873-9. DOI: 10.1109/BigDataSecurity-HPSC-IDS49724.2020.00053. URL: https://ieeexplore.ieee.org/document/9123042/ (visited on 09/21/2024).

[157] Aslı Bay et al. "Practical Multi-Party Private Set Intersection Protocols". In: *Trans. Info. For. Sec.* 17 (Jan. 2022), pp. 1–15. ISSN: 1556-6013. DOI: 10.1109/TIFS.2021.3118879. URL: https://doi.org/10.1109/TIFS.2021.3118879 (visited on 09/21/2024).

[158] Amirhossein Adavoudi Jolfaei, Hamid Mala, and Maryam Zarezadeh. "EO-PSI-CA: Efficient outsourced private set intersection cardinality". In: *Journal of Information Security and Applications* 65 (Mar. 2022), p. 102996. ISSN: 2214-2126. DOI: 10.1016/j.jisa.2021.102996. URL: https://www.sciencedirect.com/science/article/pii/S2214212621002040 (visited on 09/21/2024).

[159] Aydin Abadi, Sotirios Terzis, and Changyu Dong. "VD-PSI: Verifiable Delegated Private Set Intersection on Outsourced Private Datasets". en. In: *Financial Cryptography and Data Security*. Ed. by Jens Grossklags and Bart Preneel. Berlin, Heidelberg: Springer, 2017, pp. 149–168. ISBN: 978-3-662-54970-4. DOI: 10.1007/978-3-662-54970-4_9.

UNIVERSIDAD DE MÁLAGA

[160] Shintaro Terada and Kazuki Yoneyama. "Improved Verifiable Delegated Private Set Intersection". In: *2018 International Symposium on Information Theory and Its Applications (ISITA)*. Singapore: IEEE, Oct. 2018, pp. 520–524. ISBN: 978-4-88552-318-2. DOI: 10.23919/ISITA.2018.8664310. URL: https://ieeexplore.ieee.org/document/8664310/ (visited on 09/21/2024).

[161] Mahdi Mahdavi Oliaiy et al. "A Verifiable Delegated Set Intersection without pairing". In: *2017 Iranian Conference on Electrical Engineering (ICEE)*. Tehran, Iran: IEEE, May 2017, pp. 2047–2051. ISBN: 978-1-5090-5963-8. DOI: 10.1109/IranianCEE.2017.7985395. URL: http://ieeexplore.ieee.org/document/7985395/ (visited on 09/21/2024).

[162] Qiang Wang et al. "Tag-Based Verifiable Delegated Set Intersection Over Outsourced Private Datasets". In: *IEEE Transactions on Cloud Computing* 10.2 (Apr. 2022), pp. 1201–1214. ISSN: 2168-7161, 2372-0018. DOI: 10.1109/TCC.2020.2968320. URL: https://ieeexplore.ieee.org/document/8964424/ (visited on 09/21/2024).

[163] Mohammad Ali et al. "Attribute-based fine-grained access control for outscored private set intersection computation". In: *Information Sciences* 536 (Oct. 2020), pp. 222–243. ISSN: 0020-0255. DOI: 10.1016/j.ins.2020.05.041. URL: https://www.sciencedirect.com/science/article/pii/S0020025520304618 (visited on 09/21/2024).

[164] Vladimir Kolesnikov et al. "Efficient Batched Oblivious PRF with Applications to Private Set Intersection". en. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. Vienna Austria: ACM, Oct. 2016, pp. 818–829. ISBN: 978-1-4503-4139-4. DOI: 10.1145/2976749.2978381. URL: https://dl.acm.org/doi/10.1145/2976749.2978381 (visited on 10/03/2024).

[165] Liyan Shen et al. "A More Efficient Private Set Intersection Protocol Based on Random OT and Balance Hash". In: *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*. ISSN: 1938-1883. May 2019, pp. 1–7. DOI: 10.1109/ICC.2019.8761417. URL: https://ieeexplore.ieee.org/document/8761417 (visited on 10/03/2024).

[166] Aydin Abadi et al. "Efficient Delegated Private Set Intersection on Outsourced Private Datasets". In: *IEEE Transactions on Dependable and Secure Computing* 16.4 (July 2019). Conference Name: IEEE Transactions on Dependable and Secure Computing, pp. 608–624. ISSN: 1941-0018. DOI: 10.1109/TDSC.2017.2708710. URL: https://ieeexplore.ieee.org/document/7934388/?arnumber=7934388&tag=1 (visited on 01/24/2025).

[167] Jesper Buus Nielsen. *MPC and Blockchain: A Match Made In Heaven?* en. Mar. 2021. URL: https://medium.com/partisia-blockchain/mpc-and-blockchain-a-match-made-in-heaven-df4291390b5b (visited on 08/19/2024).

[168] Hitarshi Buch. *Synergizing Blockchain & Multi-Party Computation to Reimagine Transactions*. en. 2020. URL: https://www.wipro.com/blockchain/synergizing-blockchain-and-multi-party-computation-to-reimagine-transactions/ (visited on 08/19/2024).

[169] Rob Behnke. *What Is Multi-Party Computation (MPC) in Blockchain?* en. 2023. URL: https://www.halborn.com/blog/post/what-is-multi-party-computation-mpc-in-blockchain (visited on 08/19/2024).

UNIVERSIDAD DE MÁLAGA

[170] Tiago Serôdio. *Multiparty computation (MPC): Its effects on blockchain innovation*. en. 2023. URL: https://cointelegraph.com/innovation-circle/multiparty-computation-mpc-its-effects-on-blockchain-innovation (visited on 08/19/2024).

[171] Carsten Baum, Ivan Damgård, and Claudio Orlandi. *Publicly Auditable Secure Multi-Party Computation*. Publication info: Published elsewhere. Major revision. 9th Conference on Security and Cryptography for Networks (SCN 2014). 2014. URL: https://eprint.iacr.org/2014/075 (visited on 09/22/2024).

[172] Alex Ozdemir and Dan Boneh. "Experimenting with Collaborative zk-SNARKs: Zero-Knowledge Proofs for Distributed Secrets". en. In: *31st USENIX Security Symposium (USENIX Security 22)*. 2022, pp. 4291–4308. ISBN: 978-1-939133-31-1. URL: https://www.usenix.org/conference/usenixsecurity22/presentation/ozdemir (visited on 07/09/2024).

[173] Carlo Blundo et al. "Multi-Secret Sharing Schemes". en. In: *Advances in Cryptology — CRYPTO '94*. Ed. by Yvo G. Desmedt. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 1994, pp. 150–163. ISBN: 978-3-540-48658-9. DOI: 10.1007/3-540-48658-5_17.

[174] Javier Herranz, Alexandre Ruiz, and Germán Sáez. "New results and applications for multi-secret sharing schemes". en. In: *Designs, Codes and Cryptography* 73.3 (Dec. 2014), pp. 841–864. ISSN: 1573-7586. DOI: 10.1007/s10623-013-9831-6. URL: https://doi.org/10.1007/s10623-013-9831-6 (visited on 02/14/2024).

[175] Ilaria Chillotti et al. "Scooby: Improved Multi-party Homomorphic Secret Sharing Based on FHE". en. In: *Security and Cryptography for Networks*. Ed. by Clemente Galdi and Stanislaw Jarecki. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2022, pp. 540–563. ISBN: 978-3-031-14791-3. DOI: 10.1007/978-3-031-14791-3_24.

[176] Zhusen Liu et al. "A verifiable privacy-preserving data collection scheme supporting multi-party computation in fog-based smart grid". en. In: *Frontiers of Computer Science* 16.1 (Oct. 2021), p. 161810. ISSN: 2095-2236. DOI: 10.1007/s11704-021-0410-0. URL: https://doi.org/10.1007/s11704-021-0410-0 (visited on 09/25/2024).

[177] Mustafa A. Mustafa et al. "A Secure and Privacy-Preserving Protocol for Smart Metering Operational Data Collection". In: *IEEE Transactions on Smart Grid* 10.6 (Nov. 2019). Conference Name: IEEE Transactions on Smart Grid, pp. 6481–6490. ISSN: 1949-3061. DOI: 10.1109/TSG.2019.2906016. URL: https://ieeexplore.ieee.org/document/8669800 (visited on 09/25/2024).

[178] Hayat Mohammad Khan et al. "Fog-enabled secure multiparty computation based aggregation scheme in smart grid". In: *Computers & Electrical Engineering* 94 (Sept. 2021), p. 107358. ISSN: 0045-7906. DOI: 10.1016/j.compeleceng.2021.107358. URL: https://www.sciencedirect.com/science/article/pii/S0045790621003281 (visited on 09/25/2024).

[179] Haiyang Xue et al. *Efficient Multiplicative-to-Additive Function from Joye-Libert Cryptosystem and Its Application to Threshold ECDSA*. Publication info: Published elsewhere. ACM CCS 23. 2023. URL: https://eprint.iacr.org/2023/1312 (visited on 01/22/2024).

UNIVERSIDAD DE MÁLAGA

[180] Matt Blaze, Gerrit Bleumer, and Martin Strauss. "Divertible protocols and atomic proxy cryptography". en. In: *Advances in Cryptology — EUROCRYPT'98*. Ed. by Kaisa Nyberg. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 1998, pp. 127–144. ISBN: 978-3-540-69795-4. DOI: 10.1007/BFb0054122.

[181] Damien Stehlé and Ron Steinfeld. "Making NTRU as Secure as Worst-Case Problems over Ideal Lattices". en. In: *Advances in Cryptology – EUROCRYPT 2011*. Ed. by Kenneth G. Paterson. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2011, pp. 27–47. ISBN: 978-3-642-20465-4. DOI: 10.1007/978-3-642-20465-4_4.

[182] Damien Stehlé and Ron Steinfeld. *Making NTRUEncrypt and NTRUSign as Secure as Standard Worst-Case Problems over Ideal Lattices*. Publication info: Published elsewhere. Submitted. Some of the results in this paper have been presented in preliminary form at Eurocrypt 2011. 2013. URL: https://eprint.iacr.org/2013/004 (visited on 01/15/2024).

[183] David Nuñez, Isaac Agudo, and Javier Lopez. "NTRUReEncrypt: An Efficient Proxy Re-Encryption Scheme Based on NTRU". en. In: *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*. Singapore Republic of Singapore: ACM, Apr. 2015, pp. 179–189. ISBN: 978-1-4503-3245-3. DOI: 10.1145/2714576.2714585. URL: https://dl.acm.org/doi/10.1145/2714576.2714585 (visited on 11/14/2023).

[184] Dan Boneh et al. "Threshold Cryptosystems from Threshold Fully Homomorphic Encryption". en. In: *Advances in Cryptology – CRYPTO 2018*. Ed. by Hovav Shacham and Alexandra Boldyreva. Cham: Springer International Publishing, 2018, pp. 565–596. ISBN: 978-3-319-96884-1. DOI: 10.1007/978-3-319-96884-1_19.

[185] V. Gayoso Martínez, L. Hernández Encinas, and A. Queiruga Dios. "Security and Practical Considerations When Implementing the Elliptic Curve Integrated Encryption Scheme". In: *Cryptologia* 39.3 (July 2015). Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/01611194.2014.988363, pp. 244–269. ISSN: 0161-1194. DOI: 10.1080/01611194.2014.988363. URL: https://doi.org/10.1080/01611194.2014.988363 (visited on 02/13/2024).

[186] Jeff Hoffstein et al. "Choosing Parameters for NTRUEncrypt". en. In: *Topics in Cryptology – CT-RSA 2017*. Ed. by Helena Handschuh. Cham: Springer International Publishing, 2017, pp. 3–18. ISBN: 978-3-319-52153-4. DOI: 10.1007/978-3-319-52153-4_1.

[187] Matthew Edward Briggs. "An Introduction to the General Number Field Sieve". es. PhD thesis. Virginia Tech, Apr. 1998. URL: http://hdl.handle.net/10919/36618 (visited on 02/13/2024).

[188] *TFHE - Security and Parameters*. Online: https://tfhe.github.io/tfhe/security_and_params.html. (Visited on 02/07/2024).

[189] Pierre Baldi et al. "Countering GATTACA: efficient and secure testing of fully-sequenced human genomes". en. In: *Proceedings of the 18th ACM conference on Computer and communications security*. Chicago Illinois USA: ACM, Oct. 2011, pp. 691–702. ISBN: 978-1-4503-0948-6. DOI: 10.1145/2046707.2046785. URL: https://dl.acm.org/doi/10.1145/2046707.2046785 (visited on 10/06/2024).

UNIVERSIDAD DE MÁLAGA

[190] Aziz Mma, Alhadidi D, and Mohammed N. "Secure approximation of edit distance on genomic data". en. In: *BMC medical genomics* 10.Suppl 2 (July 2017). ISSN: 1755-8794. DOI: 10.1186/s12920-017-0279-9. URL: https://pubmed.ncbi.nlm.nih.gov/28786362/ (visited on 10/06/2024).

[191] T. D'Roza and G. Bilchev. "An Overview of Location-Based Services". en. In: *BT Technology Journal* 21.1 (Jan. 2003), pp. 20–27. ISSN: 1573-1995. DOI: 10.1023/A:1022491825047. URL: https://doi.org/10.1023/A:1022491825047 (visited on 09/12/2024).

[192] Iris A. Junglas and Richard T. Watson. "Location-based services". en. In: *Communications of the ACM* 51.3 (Mar. 2008), pp. 65–69. ISSN: 0001-0782, 1557-7317. DOI: 10.1145/1325555.1325568. URL: https://dl.acm.org/doi/10.1145/1325555.1325568 (visited on 09/12/2024).

[193] Mohammad Jabed Morshed Chowdhury et al. "COVID-19 Contact Tracing: Challenges and Future Directions". eng. In: *IEEE access: practical innovations, open solutions* 8 (2020), pp. 225703–225729. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.3036718.

[194] Leonie Reichert, Samuel Brack, and Björn Scheuermann. *Privacy-Preserving Contact Tracing of COVID-19 Patients*. 2020. URL: https://eprint.iacr.org/2020/375 (visited on 09/04/2024).

[195] C. Troncoso et al. "Decentralized Privacy-Preserving Proximity Tracing". In: *IEEE Data Eng. Bull.* (May 2020). URL: https://www.semanticscholar.org/paper/Decentralized-Privacy-Preserving-Proximity-Tracing-Troncoso-Payer/fc4dbff143839daec45c572c46bff76795c5d843 (visited on 09/05/2024).

[196] Jason Bay et al. *BlueTrace: A privacy-preserving protocol for community-driven contact tracing across borders*. en. Online: https://bluetrace.io/static/bluetrace_whitepaper-938063656596c104632def383eb33b3c.pdf. 2020.

[197] Piotr Dollar et al. "Pedestrian Detection: An Evaluation of the State of the Art". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.4 (Apr. 2012). Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 743–761. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2011.155. URL: https://ieeexplore.ieee.org/document/5975165 (visited on 09/07/2024).

[198] Haroon Idrees et al. "Multi-source Multi-scale Counting in Extremely Dense Crowd Images". In: *2013 IEEE Conference on Computer Vision and Pattern Recognition*. ISSN: 1063-6919. June 2013, pp. 2547–2554. DOI: 10.1109/CVPR.2013.329. URL: https://ieeexplore.ieee.org/document/6619173 (visited on 09/07/2024).

[199] Peng Wang et al. "MobileCount: An efficient encoder-decoder framework for real-time crowd counting". In: *Neurocomputing* 407 (Sept. 2020), pp. 292–299. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2020.05.056. URL: https://www.sciencedirect.com/science/article/pii/S0925231220308912 (visited on 09/07/2024).

[200] Camille Bailas et al. "Performance of video processing at the edge for crowd-monitoring applications". In: *2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*. Feb. 2018, pp. 482–487. DOI: 10.1109/WF-IoT.2018.

UNIVERSIDAD DE MÁLAGA

8355170. URL: https://ieeexplore.ieee.org/document/8355170 (visited on 09/07/2024).

[201] Jingyu Chen et al. "Efficient and Switchable CNN for Crowd Counting Based on Embedded Terminal". In: *IEEE Access* 7 (2019). Conference Name: IEEE Access, pp. 51533–51541. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2910458. URL: https://ieeexplore.ieee.org/document/8686066 (visited on 09/07/2024).

[202] F. Wahl, M. Milenkovic, and O. Amft. "A Distributed PIR-based Approach for Estimating People Count in Office Environments". In: *2012 IEEE 15th International Conference on Computational Science and Engineering*. Dec. 2012, pp. 640–647. DOI: 10.1109/ICCSE.2012.92. URL: https://ieeexplore.ieee.org/document/6417352 (visited on 09/07/2024).

[203] Han Zou et al. "Device-free occupancy detection and crowd counting in smart buildings with WiFi-enabled IoT". In: *Energy and Buildings* 174 (Sept. 2018), pp. 309–322. ISSN: 0378-7788. DOI: 10.1016/j.enbuild.2018.06.040. URL: https://www.sciencedirect.com/science/article/pii/S0378778817339336 (visited on 09/07/2024).

[204] Simone Di Domenico et al. "LTE-based passive device-free crowd density estimation". In: *2017 IEEE International Conference on Communications (ICC)*. ISSN: 1938-1883. May 2017, pp. 1–6. DOI: 10.1109/ICC.2017.7997194. URL: https://ieeexplore.ieee.org/document/7997194 (visited on 09/07/2024).

[205] S. M. Mousavi et al. "MobiSim: A Framework for Simulation of Mobility Models in Mobile Ad-Hoc Networks". In: *Third IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob 2007)*. ISSN: 2160-4894. Oct. 2007, pp. 82–82. DOI: 10.1109/WIMOB.2007.4390876. URL: https://ieeexplore.ieee.org/document/4390876 (visited on 09/05/2024).

[206] Hongbo Jiang et al. "Location Privacy-preserving Mechanisms in Location-based Services: A Comprehensive Survey". In: *ACM Comput. Surv.* 54.1 (Jan. 2021), 4:1–4:36. ISSN: 0360-0300. DOI: 10.1145/3423165. URL: https://dl.acm.org/doi/10.1145/3423165 (visited on 09/07/2024).

[207] Marco Gruteser and Dirk Grunwald. "Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking". In: *Proceedings of the 1st international conference on Mobile systems, applications and services*. MobiSys '03. New York, NY, USA: Association for Computing Machinery, May 2003, pp. 31–42. ISBN: 978-1-4503-7797-3. DOI: 10.1145/1066116.1189037. URL: https://dl.acm.org/doi/10.1145/1066116.1189037 (visited on 09/07/2024).

[208] Jin Li et al. "Location-Sharing Systems With Enhanced Privacy in Mobile Online Social Networks". In: *IEEE Systems Journal* 11.2 (June 2017). Conference Name: IEEE Systems Journal, pp. 439–448. ISSN: 1937-9234. DOI: 10.1109/JSYST.2015.2415835. URL: https://ieeexplore.ieee.org/document/7098330 (visited on 09/07/2024).

[209] Chi-Yin Chow, Mohamed F. Mokbel, and Xuan Liu. "Spatial cloaking for anonymous location-based services in mobile peer-to-peer environments". en. In: *GeoInformatica* 15.2 (Apr. 2011), pp. 351–380. ISSN: 1573-7624. DOI: 10.

UNIVERSIDAD DE MÁLAGA

1007/s10707-009-0099-y. URL: https://doi.org/10.1007/s10707-009-0099-y (visited on 09/07/2024).

[210] Sudheesh Singanamalla et al. "Oblivious DNS over HTTPS (ODoH): A Practical Privacy Enhancement to DNS". en. In: *Proceedings on Privacy Enhancing Technologies* 2021.4 (Oct. 2021), pp. 575–592. ISSN: 2299-0984. DOI: 10.2478/popets-2021-0085. URL: https://petsymposium.org/popets/2021/popets-2021-0085.php (visited on 09/07/2024).

[211] Quang Dao et al. "Multi-party Homomorphic Secret Sharing and Sublinear MPC from Sparse LPN". In: *Advances in Cryptology – CRYPTO 2023*. Ed. by Helena Handschuh and Anna Lysyanskaya. Cham: Springer Nature Switzerland, 2023, pp. 315–348. ISBN: 978-3-031-38545-2.

[212] Kartick Kolachala et al. "SoK: Money Laundering in Cryptocurrencies". In: *Proceedings of the 16th International Conference on Availability, Reliability and Security*. ARES '21. New York, NY, USA: Association for Computing Machinery, Aug. 2021, pp. 1–10. ISBN: 978-1-4503-9051-4. DOI: 10.1145/3465481.3465774. URL: https://dl.acm.org/doi/10.1145/3465481.3465774 (visited on 09/24/2024).

[213] Qi Feng et al. "A survey on privacy protection in blockchain system". In: *Journal of Network and Computer Applications* 126 (Jan. 2019), pp. 45–58. ISSN: 1084-8045. DOI: 10.1016/j.jnca.2018.10.020. URL: https://www.sciencedirect.com/science/article/pii/S1084804518303485 (visited on 09/24/2024).

[214] K. Sampigethaya and R. Poovendran. "A Survey on Mix Networks and Their Secure Applications". In: *Proceedings of the IEEE* 94.12 (Dec. 2006), pp. 2142–2181. ISSN: 0018-9219, 1558-2256. DOI: 10.1109/JPROC.2006.889687. URL: http://ieeexplore.ieee.org/document/4077263/ (visited on 09/24/2024).

[215] David L. Chaum. "Untraceable electronic mail, return addresses, and digital pseudonyms". In: *Commun. ACM* 24.2 (Feb. 1981), pp. 84–90. ISSN: 0001-0782. DOI: 10.1145/358549.358563. URL: https://dl.acm.org/doi/10.1145/358549.358563 (visited on 09/24/2024).

[216] Melissa Chase, Esha Ghosh, and Oxana Poburinnaya. *Secret Shared Shuffle*. Publication info: A major revision of an IACR publication in ASIACRYPT 2020. 2019. URL: https://eprint.iacr.org/2019/1340 (visited on 09/24/2024).

[217] Saba Eskandarian and Dan Boneh. "Clarion: Anonymous Communication from Multiparty Shuffling Protocols". en. In: *Proceedings 2022 Network and Distributed System Security Symposium*. San Diego, CA, USA: Internet Society, 2022. ISBN: 978-1-891562-74-7. DOI: 10.14722/ndss.2022.24141. URL: https://www.ndss-symposium.org/wp-content/uploads/2022-141-paper.pdf (visited on 09/24/2024).

[218] Joseph Bonneau et al. *Mixcoin: Anonymity for Bitcoin with accountable mixes*. Report Number: 077. 2014. URL: https://eprint.iacr.org/2014/077 (visited on 08/09/2023).

[219] Luke Valenta and Brendan Rowan. "Blindcoin: Blinded, Accountable Mixes for Bitcoin". en. In: *Financial Cryptography and Data Security*. Ed. by Michael Brenner et al. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer,

UNIVERSIDAD DE MÁLAGA

2015, pp. 112–126. ISBN: 978-3-662-48051-9. DOI: 10.1007/978-3-662-48051-9_9.

[220] Ethan Heilman et al. *TumbleBit: An Untrusted Bitcoin-Compatible Anonymous Payment Hub*. Publication info: Preprint. 2016. URL: https://eprint.iacr.org/2016/575 (visited on 08/30/2023).

[221] Gregory Maxwell. *CoinJoin: Bitcoin privacy for the real world*. 2013. URL: https://bitcointalk.org/index.php?topic=279249.0 (visited on 08/30/2023).

[222] Tim Ruffing, Pedro Moreno-Sanchez, and Aniket Kate. "CoinShuffle: Practical Decentralized Coin Mixing for Bitcoin". en. In: *Computer Security - ESORICS 2014*. Ed. by Mirosław Kutyłowski and Jaideep Vaidya. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2014, pp. 345–364. ISBN: 978-3-319-11212-1. DOI: 10.1007/978-3-319-11212-1_20.

[223] Jan Henrik Ziegeldorf et al. "CoinParty: Secure Multi-Party Mixing of Bitcoins". In: *Proceedings of the 5th ACM Conference on Data and Application Security and Privacy*. CODASPY '15. New York, NY, USA: Association for Computing Machinery, Mar. 2015, pp. 75–86. ISBN: 978-1-4503-3191-3. DOI: 10.1145/2699026.2699100. URL: https://dl.acm.org/doi/10.1145/2699026.2699100 (visited on 09/05/2023).

[224] Sarah Meiklejohn and Rebekah Mercer. "Möbius: Trustless Tumbling for Transaction Privacy". en. In: *Proceedings on Privacy Enhancing Technologies* 2018.2 (Apr. 2018), pp. 105–121. ISSN: 2299-0984. DOI: 10.1515/popets-2018-0015. URL: https://petsymposium.org/popets/2018/popets-2018-0015.php (visited on 08/09/2023).

[225] Maya Dotan, Ayelet Lotem, and Margarita Vald. *Haze: A Compliant Privacy Mixer*. Report Number: 1152. 2023. URL: https://eprint.iacr.org/2023/1152 (visited on 08/08/2023).

[226] David Nuñez, Isaac Agudo, and Javier Lopez. "Escrowed decryption protocols for lawful interception of encrypted data". In: *IET Information Security* 13.5 (2019), pp. 498–507. ISSN: 1751-8717. (Visited on 09/07/2023).

[227] Paul Feldman. "A practical scheme for non-interactive verifiable secret sharing". In: *28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*. Los Angeles, CA, USA: IEEE, Oct. 1987, pp. 427–438. ISBN: 978-0-8186-0807-0. DOI: 10.1109/SFCS.1987.4. URL: http://ieeexplore.ieee.org/document/4568297/ (visited on 09/25/2024).

[228] Marta Bellés-Muñoz et al. "Circom: A Circuit Description Language for Building Zero-Knowledge Applications". In: *IEEE Transactions on Dependable and Secure Computing* 20.6 (Nov. 2023), pp. 4733–4751. ISSN: 1545-5971, 1941-0018, 2160-9209. DOI: 10.1109/TDSC.2022.3232813. URL: https://ieeexplore.ieee.org/document/10002421/ (visited on 09/25/2024).

[229] Martin Albrecht et al. "MiMC: Efficient Encryption and Cryptographic Hashing with Minimal Multiplicative Complexity". en. In: *Advances in Cryptology – ASIACRYPT 2016*. Ed. by Jung Hee Cheon and Tsuyoshi Takagi. Berlin, Heidelberg: Springer, 2016, pp. 191–219. ISBN: 978-3-662-53887-6. DOI: 10.1007/978-3-662-53887-6_7.

[230] Lorenzo Grassi et al. "Poseidon: A New Hash Function for {Zero-Knowledge} Proof Systems". en. In: 2021, pp. 519–535. ISBN: 978-1-939133-24-3. URL:

UNIVERSIDAD DE MÁLAGA

https://www.usenix.org/conference/usenixsecurity21/presentation/grassi (visited on 09/25/2024).

UNIVERSIDAD DE MÁLAGA