New Security Definitions, Constructions and Applications of Proxy Re-Encryption

David Nuñez

Advisors:
Isaac Agudo and **Javier Lopez**

Department of Computer Science
Universidad de Málaga, Spain
Email: dnunez@lcc.uma.es

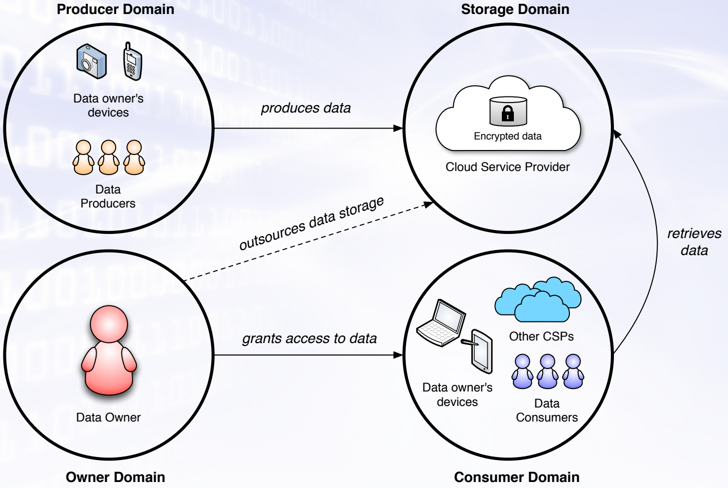
STM 2017 – Oslo, Norway

Outline

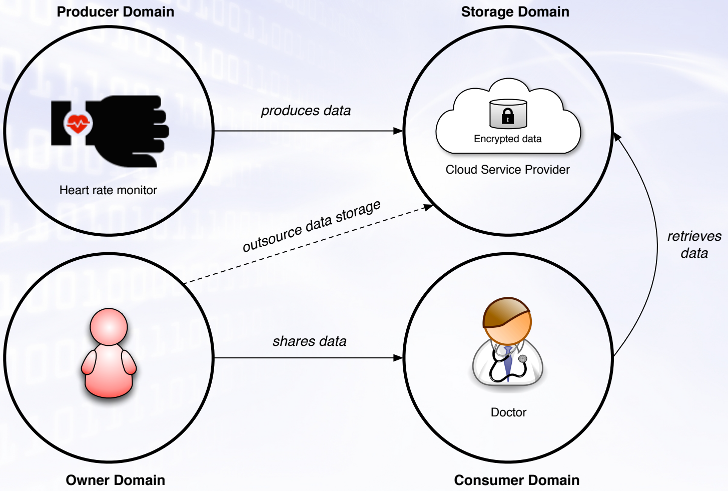
Motivation: Cloud computing

- Great expectations: better performance, cost reduction, etc.
- Great concerns: security and privacy risks
- Conventional security premise:
  ⇒*attackers should not get inside the security domain*
  - Goal: Keep the attacker away from the protected assets
  - Measures: access control systems, firewalls, etc.
  - Cloud provider must be **fully trusted** to not bypass these measures
- A more realistic premise:
  ⇒*attackers have potential access to users' data*
  - Implication: Data must be stored in encrypted form
  - Trust in the cloud provider can be reduced

Secure Data Sharing Scenario



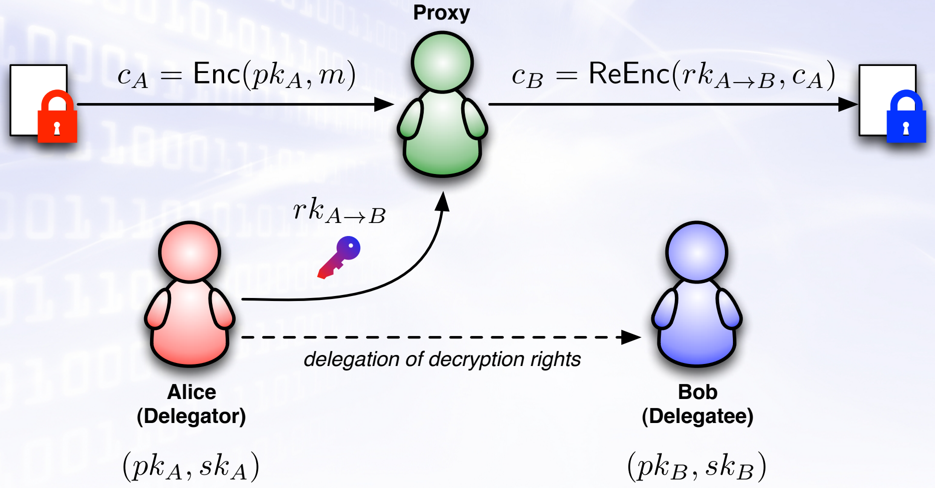
Secure Data Sharing Scenario



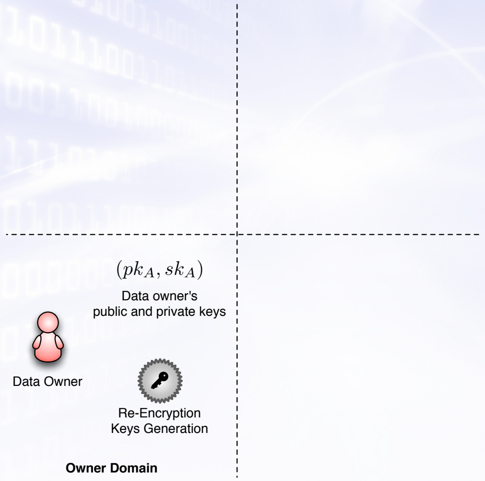
Encrypted data in the cloud

- Critical requirement: the provider should not have access to the decryption keys
- Not an easy task:
  - Symmetric encryption cannot be used alone, since it implies that the same key is shared or agreed
  - Public-key encryption implies fixing a recipient in advance
  - Increasingly complex problem: multiple pieces of data, diverse producers and consumers
- Need for non-traditional cryptosystems that provide advanced functionalities

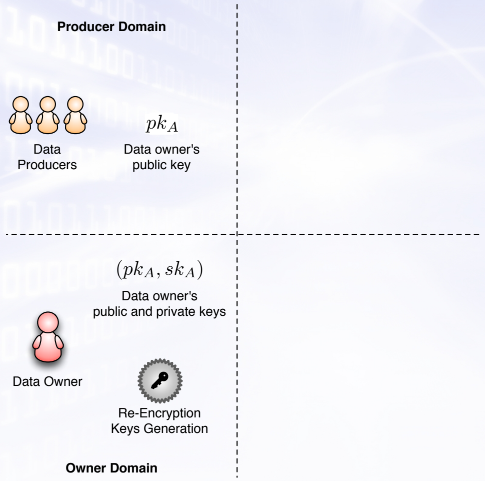
Proxy Re-Encryption


Proxy
$c_A = \mathsf{Enc}(pk_A, m)$
$c_B = \mathsf{ReEnc}(rk_{A\to B}, c_A)$
$rk_{A\to B}$
delegation of decryption rights
Alice
(Delegator)
(pk_A, sk_A)
Bob
(Delegatee)
(pk_B, sk_B)

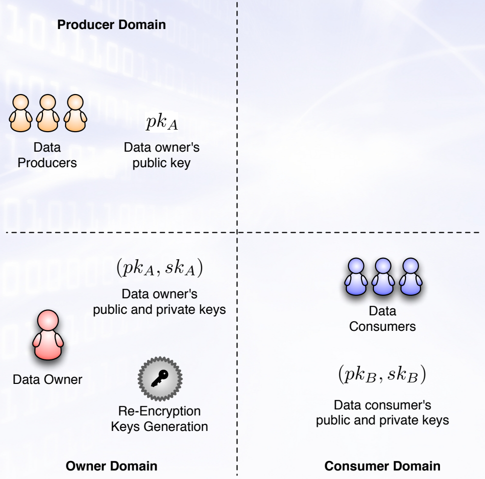
Secure Data Sharing Scenario and PRE


(pk_A, sk_A)
Data owner's public and private keys
Data Owner
Re-Encryption Keys Generation
Owner Domain

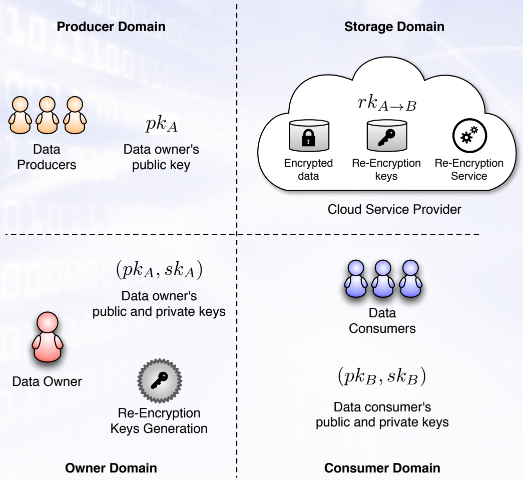
Secure Data Sharing Scenario and PRE



Secure Data Sharing Scenario and PRE



Secure Data Sharing Scenario and PRE



Bibliometric Analysis of PRE Applications

Criteria	Classification	Coverage
Functionality	Access Control	67%
	Key Management	24%
	Communication	4%
Objective	Confidentiality	80%
	Privacy	10%
	Authentication	8%
	Accountability	2%
Scenarios	Cloud	53%
	Wireless Network	8%
	Others	33%

Application to the Real World

Growing interest:

- In 2013, Toshiba deployed in Japan a cloud storage service with PRE
- Increasing number of patents
  $\Rightarrow$ Toshiba, Huawei, Mitsubishi, Nokia, Gemalto, ...
- Some start-ups have appeared in the past months

Goals of this Thesis

1. Better understanding:
   - Great variety of PRE schemes and applications
     $\Rightarrow$ Review and analyze the state of the art
   - Lack of unified definitions of the very idea of security in PRE
     $\Rightarrow$ Achieve definitional unity
   - Reuse of concepts and techniques from PKE
     $\Rightarrow$ Analyze the implications in the PRE context
2. More security:
   - Most PRE schemes achieve weak security notions
     $\Rightarrow$ Define techniques for strengthening security

Goals of this Thesis

3. Better performance:
   - PRE is ideally suited to the secure data sharing scenario
     $\Rightarrow$ Efficiency is a core aspect

4. Bringing theory and practice together:
   - Proposals often lack a practical treatment
     $\Rightarrow$ Integration of PRE within real systems

Syntax of a Proxy Re-Encryption Scheme

A PRE scheme is composed of functions KeyGen, Enc, Dec, ReKeyGen, and ReEnc:

- KeyGen$(n) \rightarrow (pk_i, sk_i)$. On input security parameter n, the key generation algorithm KeyGen outputs a pair of public and secret keys (pk_i, sk_i) for user i.
- Enc$(pk_i, m) \rightarrow c_i$. On input the public key pk_i and a message $m \in \mathcal{M}$, the encryption algorithm Enc outputs a ciphertext $c_i \in \mathcal{C}$.
- Dec$(sk_i, c_i) \rightarrow m$. On input the secret key sk_i and a ciphertext $c_i \in \mathcal{C}$, the decryption algorithm Dec outputs a message $m \in \mathcal{M}$ or the symbol $\perp$ indicating c_i is invalid.

Syntax of a Proxy Re-Encryption Scheme

- ReKeyGen$(pk_i, sk_i, pk_j, sk_j) \to rk_{i\to j}$. On input the pair of public and secret keys (pk_i, sk_i) for user i and the pair of public and secret keys (pk_j, sk_j) for user j, the re-encryption key generation algorithm ReKeyGen outputs a re-encryption key $rk_{i\to j}$.

- ReEnc$(rk_{i\to j}, c_i) \to c_j$. On input a re-encryption key $rk_{i\to j}$ and a ciphertext $c_i \in \mathcal{C}$, the re-encryption algorithm ReEnc outputs a second ciphertext $c_j \in \mathcal{C}$ or the symbol $\perp$ indicating c_i is invalid.

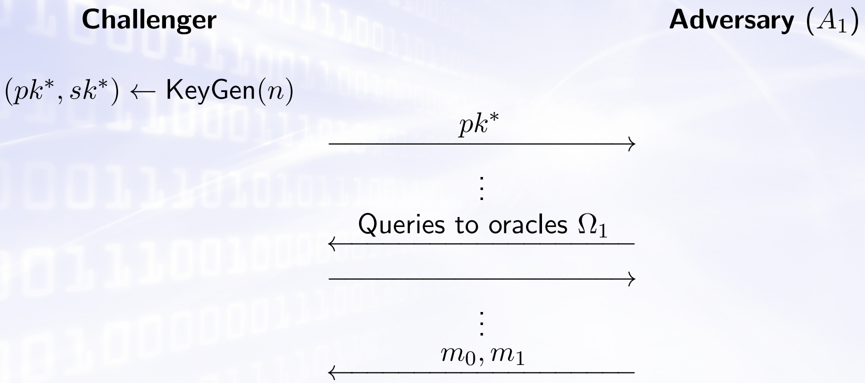
The Indistinguishability game (IND)

It formalizes the inability of an adversary to **distinguish** which message, from two possible options m_0 and m_1, is encrypted under ciphertext c^*.

$$\begin{tikzcd}[column sep=huge]
\textbf{Challenger} \arrow[r, shift right=1, "c^*"] & \textbf{Adversary} \arrow[l, shift right=4, "m_0, m_1"'] \arrow[l, shift left=4, "m?"]
\end{tikzcd}$$

The adversary has some capabilities in the form of **oracles**

The Indistinguishability game (IND): Phase 1


Challenger
Adversary (A_1)
$(pk^*, sk^*) \leftarrow \text{KeyGen}(n)$
pk^*
$\vdots$
Queries to oracles Ω_1
$\vdots$
m_0, m_1

The Indistinguishability game (IND): Phase 2

Challenger **Adversary** (A_2)

$\delta \xleftarrow{R} \{0,1\}$

$c^* \leftarrow \mathsf{Enc}(pk^*, m_\delta)$

$\xrightarrow{\qquad\qquad c^* \qquad\qquad}$

$\vdots$

$\xleftarrow{\text{Queries to oracles } \Omega_2}$

$\xrightarrow{\qquad\qquad\qquad\qquad}$

$\vdots$

$\xleftarrow{\qquad\qquad \delta' \qquad\qquad}$

$\delta \overset{?}{=} \delta'$

Definitions of Security for Public-Key Encryption

- **Decryption oracle**: $\mathcal{O}_{dec}(c) \rightarrow m$
- The IND game admits different **attack models**:
  - CPA $\Rightarrow \Omega_1 = \varnothing \qquad \Omega_2 = \varnothing$
  - CCA1 $\Rightarrow \Omega_1 = \{\mathcal{O}_{dec}\} \qquad \Omega_2 = \varnothing$
  - CCA2 $\Rightarrow \Omega_1 = \{\mathcal{O}_{dec}\} \qquad \Omega_2 = \{\mathcal{O}_{dec}\}$
- Attack models for PKE: $\{\text{CCA}i \mid 0 \leq i \leq 2\}$
  $\Rightarrow$ The index i indicates the last phase of the game where $\mathcal{O}_{dec}$ is available:
  CCA0 ($=$ CPA), CCA1, CCA2

[Bellare et al., CRYPTO 1998]

Definitions of Security for Proxy Re-Encryption

- Reuse of PKE definitions of security:
  - The Indistinguishability (IND) game
  - Decryption oracle: $\mathcal{O}_{dec}(pk, c) \to m$
- In PRE, we also need to add a re-encryption oracle:

$$\mathcal{O}_{reenc}(pk, pk', c) \to c'$$

- This addition makes the definitions of security more complex.

Parametrizing attack models

Attack models for PKE: $\{\mathsf{CCA}i \mid 0 \leq i \leq 2\}$

- The index i indicates the last phase of the security game where $\mathcal{O}_{dec}$ is available

Contribution: Parametric family of attack models

Attack models for PRE: $\{\mathsf{CCA}_{i,j} \mid 0 \leq i, j \leq 2\}$

- The index i indicates the last phase of the security game where $\mathcal{O}_{dec}$ is available
- The index j indicates the last phase of the security game where $\mathcal{O}_{reenc}$ is available

Parametric Family of Attack Models for PRE

Ω_1	Ω_2	Attack model
$\varnothing$	$\varnothing$	$\mathsf{CCA}_{0,0} = \mathsf{CPA}$
$\{\mathcal{O}_{reenc}\}$	$\varnothing$	$\mathsf{CCA}_{0,1}$
$\{\mathcal{O}_{reenc}\}$	$\{\mathcal{O}_{reenc}\}$	$\mathsf{CCA}_{0,2}$
$\{\mathcal{O}_{dec}\}$	$\varnothing$	$\mathsf{CCA}_{1,0}$
$\{\mathcal{O}_{dec}, \mathcal{O}_{reenc}\}$	$\varnothing$	$\mathsf{CCA}_{1,1}$
$\{\mathcal{O}_{dec}, \mathcal{O}_{reenc}\}$	$\{\mathcal{O}_{reenc}\}$	$\mathsf{CCA}_{1,2}$
$\{\mathcal{O}_{dec}\}$	$\{\mathcal{O}_{dec}\}$	$\mathsf{CCA}_{2,0}$
$\{\mathcal{O}_{dec}, \mathcal{O}_{reenc}\}$	$\{\mathcal{O}_{dec}\}$	$\mathsf{CCA}_{2,1}$
$\{\mathcal{O}_{dec}, \mathcal{O}_{reenc}\}$	$\{\mathcal{O}_{dec}, \mathcal{O}_{reenc}\}$	$\mathsf{CCA}_{2,2}$

Trivial implications between PRE security notions

$$\begin{tikzcd}
 & & \text{IND-CCA}_{2,2} \arrow[dl] \arrow[dr] & & \\
 & \text{IND-CCA}_{1,2} \arrow[dl] \arrow[dr] & & \text{IND-CCA}_{2,1} \arrow[dl] \arrow[dr] & \\
\text{IND-CCA}_{0,2} \arrow[dr] & & \text{IND-CCA}_{1,1} \arrow[dl] \arrow[dr] & & \text{IND-CCA}_{2,0} \arrow[dl] \\
 & \text{IND-CCA}_{0,1} \arrow[dr] & & \text{IND-CCA}_{1,0} \arrow[dl] & \\
 & & \text{IND-CCA}_{0,0} & &
\end{tikzcd}$$

Separations between PRE security notions

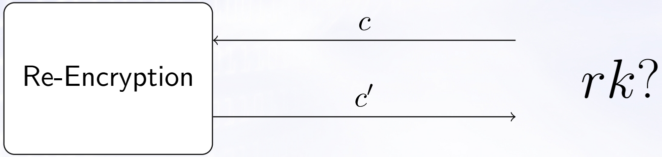
$$\begin{tikzcd}
 & & \text{IND-CCA}_{2,2} \arrow[dl, dashed] \arrow[dr, dashed] & & \\
 & \text{IND-CCA}_{1,2} \arrow[dl, dashed] \arrow[dr, dashed] & & \text{IND-CCA}_{2,1} \arrow[dl, dashed] \arrow[dr, dashed] & \\
\text{IND-CCA}_{0,2} \arrow[dr, dashed] & & \text{IND-CCA}_{1,1} \arrow[dl, dashed] \arrow[dr, dashed] & & \text{IND-CCA}_{2,0} \arrow[dl, dashed] \arrow[ul, red, "/" marking, "\textbf{Thm. 4.7}"'] \arrow[dlll, red, "/" marking, "\textbf{Thm. 4.8}"] \\
 & \text{IND-CCA}_{0,1} \arrow[dr, dashed] & & \text{IND-CCA}_{1,0} \arrow[dl, dashed] & \\
 & & \text{IND-CCA}_{0,0} & &
\end{tikzcd}$$

Separations between PRE security notions

These separations arise from the violation of the following property:

Contribution: Private Re-Encryption Keys property

"The adversary should not be able to **learn the re-encryption key** from a ciphertext and its re-encryption"



If this property is violated, then the scheme is vulnerable to chosen-ciphertext attacks that use the **re-encryption oracle**

Exploiting the re-encryption key leakage

Contribution: Impossibility result

A PRE scheme that violates the private re-encryption keys property can only be IND-CCA$_{i,0}$, for $i \in \{0,1,2\}$ (i.e., a security notion without $\mathcal{O}_{reenc}$)

$$\begin{tikzcd}
 & & \text{IND-CCA}_{2,2} \arrow[dl] \arrow[dr] & & \\
 & \text{IND-CCA}_{1,2} \arrow[dl] \arrow[dr] & & \text{IND-CCA}_{2,1} \arrow[dl] \arrow[dr] & \\
\text{IND-CCA}_{0,2} \arrow[dr] & & \text{IND-CCA}_{1,1} \arrow[dl] \arrow[dr] & & \text{IND-CCA}_{2,0} \arrow[dl] \\
 & \text{IND-CCA}_{0,1} \arrow[dr] & & \text{IND-CCA}_{1,0} \arrow[dl] & \\
 & & \text{IND-CCA}_{0,0} & &
\end{tikzcd}$$

The PRE scheme from Kirshanova (PKC'14)

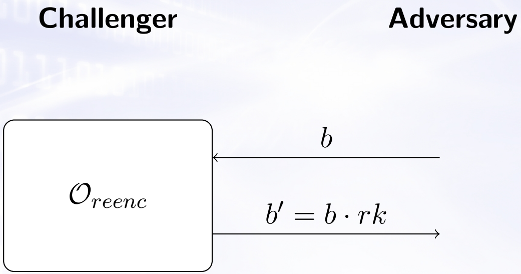
- Lattice-based scheme
- Unidirectional, collusion-resistant, non-interactive
- Allegedly 'CCA1 secure' $\Rightarrow$ We show a IND-CCA$_{1,1}$ attack
- Actually, the scheme is IND-CCA$_{1,0}$ secure because it does not fulfill the private re-encryption keys property

The PRE scheme from Kirshanova (PKC'14)

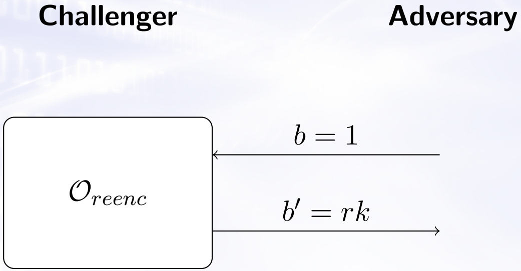
- Ciphertexts are integer **vectors** b
- Re-encryption key is a **matrix** rk
- Re-encryption is simply a vector-matrix multiplication:

$$b' = b \cdot rk$$

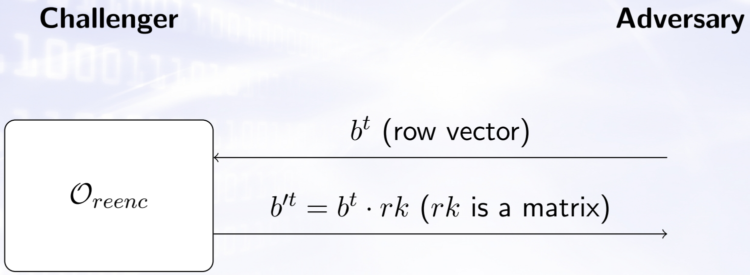
The heart of the attack



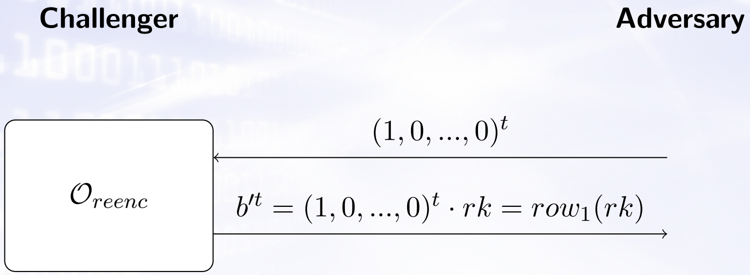
The heart of the attack



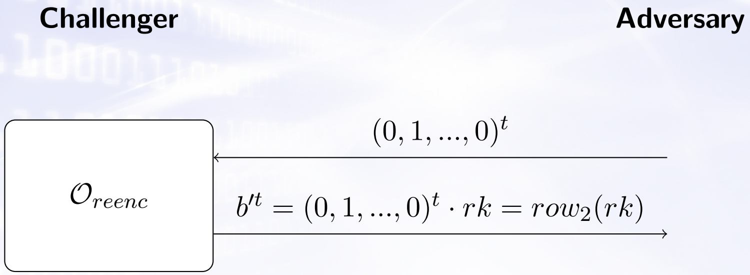
The heart of the attack



The heart of the attack

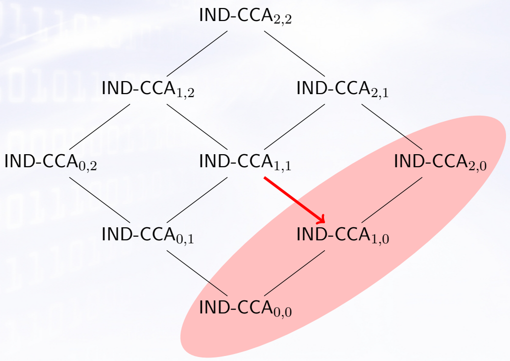


The heart of the attack



Finalizing the attack

- The adversary can extract arbitrary rows of rk
- The scheme **violates the privacy of re-encryption keys**
  $\Rightarrow$ The scheme cannot be IND-CCA$_{1,1}$



Summary

- We propose a **parametric family of attack models for PRE**
- We formalize the **private re-encryption keys** property and show what happens when it is violated
  $\Rightarrow$ Attacks can be based exclusively on $\mathcal{O}_{reenc}$
- We exemplify this by showing an **attack to a PRE scheme**

New Proxy Re-Encryption Constructions

- Part of the motivation of this thesis is to investigate more concrete aspects of PRE schemes, such as those related to **performance** and **security constructions**.
    - Design faster PRE schemes
    - Increase security of PRE schemes

- Two separate contributions:
    - We explore the use of lattice-based crypto to construct more efficient PRE schemes
    - We study the application of generic transformations to increase the security of PRE schemes

NTRUReEncrypt: Efficient PRE scheme based on NTRU

- We propose new PRE schemes based on NTRU, a widely known lattice-based cryptosystem.
- Lattice-based cryptography is a promising field:
  - Post-quantum security
  - Efficiency through parallelization
- We provide two different schemes:
  - The first is based on the conventional NTRU cryptosystem
  - The second is based on an NTRU variant that is CPA-secure under the Ring-LWE assumption.
- Our experimental results show that the first scheme outperforms previous proposals by an order of magnitude.

NTRU: Overview

- One of the first PKE schemes based on lattices
- Proposed by Hoffstein, Pipher and Silverman in 1996
- NTRUEncrypt is very efficient, orders of magnitude faster than other PKE schemes
- IEEE Standard 1363.1-2008 and ANSI X9.98-2010
- It lacks a formal proof of security

NTRU

- Based on **integer polynomials**: keys, messages, ciphertexts
- Operations are **additions** and **multiplications** modulo p and q
- Private key: $sk = f$ Public key: $pk = h$
- Encryption:
  - Plaintext M Public key h Random noise s
  - Ciphertext $C = h \cdot s + M \mod q$
- Decryption:
  - Secret key f Ciphertext C
  - Compute $C' = f \cdot C \mod q$
  - Output $m = C' \mod p$

NTRUReEncrypt

Contribution: NTRUReEncrypt

We extended NTRU to support re-encryption of ciphertexts

- Private key: $sk_A = f_A$ Public key: $pk_A = h_A$
- Re-Encryption Key Generation:
  - Secret keys $sk_A = f_A$ and $sk_B = f_B$
  - Re-encryption key

$$rk_{A \to B} = sk_A \cdot sk_B^{-1} = f_A \cdot f_B^{-1}$$

- Re-Encryption:
  - Re-encryption key $rk_{A \to B}$ Ciphertext C_A Random noise e
  - Re-encrypted ciphertext

$$C_B = C_A \cdot rk_{A \to B} + p \cdot e$$

NTRUReEncrypt: Re-Encryption

Limited Multihop:

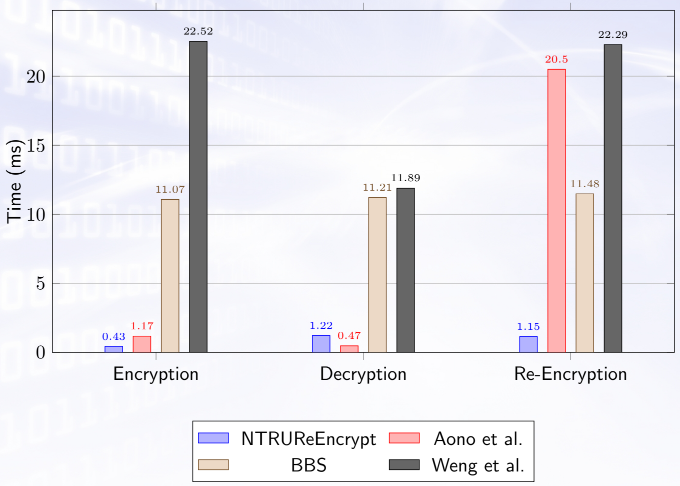
- The scheme does not support unlimited re-encryptions
- The noise e added during the re-encryption accumulates on each hop, until eventually, decryption fails
- This depends heavily on the choice of parameters

NTRUReEncrypt: Analysis

Computational costs:

- The core operation is the **multiplication of polynomials**
- Permits parallelization $\Rightarrow$ Multicore architectures and GPUs
- Encryption, decryption and re-encryption only need a **single multiplication** $\Rightarrow$ The scheme has great performance

Comparison of NTRUReEncrypt to other schemes



NTRUReEncrypt: Analysis

Space costs:

- Keys and ciphertexts are polynomials of size $O(n \cdot \log_2 q)$ bits
- Other lattice-based schemes have ciphertexts of size $O(n^2)$

Table: Comparison of space costs (in KB)

Size	Aono et al.	NTRUReEncrypt
Public keys	60.00	1.57
Secret key	60.00	1.57
Re-Encryption key	2520.00	1.57
Ciphertext	0.66	1.57

PS-NTRUReEncrypt

- NTRUReEncrypt does not have a proof of security

Contribution: PS-NTRUReEncrypt

Provable secure version of NTRUReEncrypt

- **IND-CPA secure** under the **Ring-LWE assumption**
- Extension of the NTRU variant proposed by Stehlé and Steinfeld [Eurocrypt'11]
- More of theoretical interest
  $\Rightarrow$ Not very efficient with current parameters

Motivation

- We just described a PRE scheme that is proven CPA-secure.
- An immediate objective would be to improve its security notion, hopefully achieving full CCA-security
- Two possible strategies can be applied:
  - Redesign, from scratch, a new scheme based on the original
  - Bootstrap the achieved security notion into a stronger one by means of a generic method.
- We focus on the second strategy

Motivation

- Several generic methods exist for achieving CCA-security PKE schemes from weakly secure cryptosystems
  $\Rightarrow$ E.g., Fujisaki-Okamoto, REACT, GEM
- This is not the case of proxy re-encryption
- Several flawed attempts to reuse these transformations
- Goal $\Rightarrow$ To explore the adaptation of these methods to PRE

Fujisaki-Okamoto Transformation (FOT) [J. Crypto, 2013]

- Let PKE be a public-key encryption scheme, Sym a symmetric encryption scheme, and H and G hash functions
- Encryption:
  1. Samples a random σ
  2. $c \leftarrow$ Sym.Enc$(G(\sigma), m)$
  3. $e \leftarrow$ PKE.Enc$(pk, \sigma; H(\sigma, c))$
  4. Output (e, c)
- Decryption:
  1. $\sigma \leftarrow$ PKE.Dec(sk, e)
  2. $m \leftarrow$ Sym.Dec$(G(\sigma), c)$
  3. Check: $e \stackrel{?}{=}$ PKE.Enc$(pk, \sigma; H(\sigma, c))$
  4. Output m if check holds;
     otherwise output $\perp$

Fujisaki-Okamoto Transformation (FOT) [J. Crypto, 2013]

- Let PKE be a public-key encryption scheme, Sym a symmetric encryption scheme, and H and G hash functions
- Encryption:
  1. Samples a random σ
  2. $c \leftarrow \text{Sym.Enc}(G(\sigma), m)$
  3. $e \leftarrow \text{PKE.Enc}(pk, \sigma; H(\sigma, c))$
  4. Output (e, c)
- Decryption:
  1. $\sigma \leftarrow \text{PKE.Dec}(sk, e)$
  2. $m \leftarrow \text{Sym.Dec}(G(\sigma), c)$
  3. Check: $e \stackrel{?}{=} \text{PKE.Enc}(pk, \sigma; H(\sigma, c))$
  4. Output m if check holds; otherwise output $\perp$

FOT produces a **CCA-secure** scheme in the Random Oracle Model if PKE is OW-CPA secure

Applying FOT to Proxy Re-Encryption

- It is tempting to directly use FOT in PRE
- Re-Encryption: input (e, c)
  1. $e' \leftarrow \mathsf{PRE.ReEnc}(rk, e)$
  2. Output (e, c)
- Recall that the check during decryption involves reconstructing the ciphertext
- If the re-encryption alters the randomness of the ciphertext, **the check will fail**
- The "CCA-secure" PRE scheme by Aono et al. [IndoCrypt 2013] suffers from this flaw

Perfect Key-Switching

- A solution could be to require that the PRE scheme does not alter the randomness during re-encryption

Contribution: Perfect Key-Switching property

Re-encryption simply "switches" one public key for another, without altering the original randomness

Perfect Key-Switching

- A solution could be to require that the PRE scheme does not alter the randomness during re-encryption

Contribution: Perfect Key-Switching property

Re-encryption simply "switches" one public key for another, without altering the original randomness

$$\mathsf{Enc}(pk_i, m; r)$$

Perfect Key-Switching

- A solution could be to require that the PRE scheme does not alter the randomness during re-encryption

Contribution: Perfect Key-Switching property

Re-encryption simply "switches" one public key for another, without altering the original randomness

$$\mathsf{ReEnc}(rk_{i\to j}, \mathsf{Enc}(pk_i, m; r))$$

Perfect Key-Switching

- A solution could be to require that the PRE scheme does not alter the randomness during re-encryption

Contribution: Perfect Key-Switching property

Re-encryption simply "switches" one public key for another, without altering the original randomness

$$\mathsf{ReEnc}(rk_{i\to j}, \mathsf{Enc}(pk_i, m; r)) = \mathsf{Enc}(pk_j, m; r)$$

Perfect Key-Switching

- A solution could be to require that the PRE scheme does not alter the randomness during re-encryption

Contribution: Perfect Key-Switching property

Re-encryption simply "switches" one public key for another, without altering the original randomness

$$\mathsf{ReEnc}(rk_{i\to j}, \mathsf{Enc}(pk_i, m; r)) = \mathsf{Enc}(pk_j, m; r)$$

- E.g.: PRE scheme from Blaze, Bleumer, and Strauss

$$\left(\underbrace{(g^a)}_{pk_A}{}^r, g^r \cdot m \right) \xrightarrow{\text{Re-encryption}} \left(\underbrace{(g^b)}_{pk_B}{}^r, g^r \cdot m \right)$$

Applying FOT to Proxy Re-Encryption

- Encryption: input pk, m
    1. Samples a random σ
    2. $c \leftarrow$ Sym.Enc$(G(\sigma), m)$
    3. $e \leftarrow$ PRE.Enc$(pk, \sigma; H(\sigma, c))$
    4. Output (e, c)
- Decryption: input $sk, (e, c)$
    1. $\sigma \leftarrow$ PRE.Dec(sk, e)
    2. $m \leftarrow$ Sym.Dec$(G(\sigma), c)$
    3. Check: $e \overset{?}{=}$ PRE.Enc$(pk, \sigma; H(\sigma, c))$
    4. Output m if check holds; otherwise output $\perp$
- Re-Encryption: input $rk, (e, c)$
    1. $e' \leftarrow$ PRE.ReEnc(rk, e)
    2. Output (e', c)

Applying FOT to Proxy Re-Encryption

- Encryption: input pk, m
  1. Samples a random σ
  2. $c \leftarrow \mathsf{Sym.Enc}(G(\sigma), m)$
  3. $e \leftarrow \mathsf{PRE.Enc}(pk, \sigma; H(\sigma, c))$
  4. Output (e, c)
- Decryption: input $sk, (e, c)$
  1. $\sigma \leftarrow \mathsf{PRE.Dec}(sk, e)$
  2. $m \leftarrow \mathsf{Sym.Dec}(G(\sigma), c)$
  3. Check: $e \stackrel{?}{=} \mathsf{PRE.Enc}(pk, \sigma; H(\sigma, c))$
  4. Output m if check holds; otherwise output $\perp$
- Re-Encryption: input $rk, (e, c)$
  1. $e' \leftarrow \mathsf{PRE.ReEnc}(rk, e)$
  2. Output (e', c)

Contribution: Correctness

This extension satisfies PRE correctness if the original PRE scheme satisfies the Perfect Key-Switching property

Applying FOT to Proxy Re-Encryption: Security

- In the original security proof of FOT, $\mathcal{O}_{dec}$ is constructed using only the random oracle table $\mathcal{L}_H$:

```
Algorithm $\mathcal{O}_{dec}(pk_i, (e, c))$
    Search $(\sigma, c, h) \in \mathcal{L}_H$, such that $e = \mathsf{PKE.Enc}(pk_i, \sigma; h)$
    If such tuple does not exist, then return $\perp$
    Return $\mathsf{Sym.Dec}(G(\sigma), c)$
```

- Searches in $\mathcal{L}_H$ for the σ encrypted in e, and uses it to decrypt c

- It is tempting to reuse this idea for $\mathcal{O}_{reenc}$:

```
Algorithm $\mathcal{O}_{reenc}(pk_i, pk_j, (e, c))$
    Search $(\sigma, c, h) \in \mathcal{L}_H$, such that $e = \mathsf{PRE.Enc}(pk_i, \sigma; h)$
    If such tuple does not exist, then return $\perp$
    Return $(\mathsf{PRE.Enc}(pk_j, \sigma; h), c)$
```

Applying FOT to Proxy Re-Encryption: Security

- However, there is a flaw...
  - Suppose the adversary creates an ill-formed ciphertext (e, c) where the randomness in e does not come from H:

$$e = \text{PRE.Enc}(pk, \sigma; r), \text{ for random } r$$

  - If (e, c) is inputed to the re-encryption oracle, it will get rejected
  - This behavior does not match the real execution
- This strategy results in **invalid proofs**...although it is used by >10 PRE schemes
- Conclusion $\Rightarrow$ The security proof cannot rely on the random oracle tables for constructing the re-encryption oracle

Applying FOT to Proxy Re-Encryption: Security

- Alternative: strengthen the requirements on the underlying PRE scheme
  $\Rightarrow$ We require **IND-CCA**$_{0,1}$ security
- Security proof:
  $\Rightarrow$ Reduction to the IND-CCA$_{0,1}$ security of the PRE scheme
  $\Rightarrow$ Without using the random oracle tables for $\mathcal{O}_{reenc}$
- **IND-CCA**$_{2,1}$ security in the ROM

$$\begin{tikzcd}
 & & \text{IND-CCA}_{2,2} \arrow[dl, dash] \arrow[dr, dash] & & \\
 & \text{IND-CCA}_{1,2} \arrow[dl, dash] \arrow[dr, dash] & & \text{IND-CCA}_{2,1} \arrow[dl, dash] \arrow[dr, dash] & \\
\text{IND-CCA}_{0,2} \arrow[dr, dash] & & \text{IND-CCA}_{1,1} \arrow[dl, dash] \arrow[dr, dash] & & \text{IND-CCA}_{2,0} \arrow[dl, dash] \\
 & \text{IND-CCA}_{0,1} \arrow[dr, dash] \arrow[uurr, bend right, red] & & \text{IND-CCA}_{1,0} \arrow[dl, dash] & \\
 & & \text{IND-CCA}_{0,0} & &
\end{tikzcd}$$

Contribution: Extending FOT to PRE

We provide a security proof for FOT in PRE

Summary

- **NTRUReEncrypt** is a highly-efficient proxy re-encryption scheme based on the NTRU cryptosystem
- The main strength of this scheme is its performance
  $\Rightarrow$ Outperforms other schemes by an order of magnitude
- We also propose **PS-NTRUReEncrypt**, a provably-secure variant that is CPA-secure under the Ring-LWE assumption

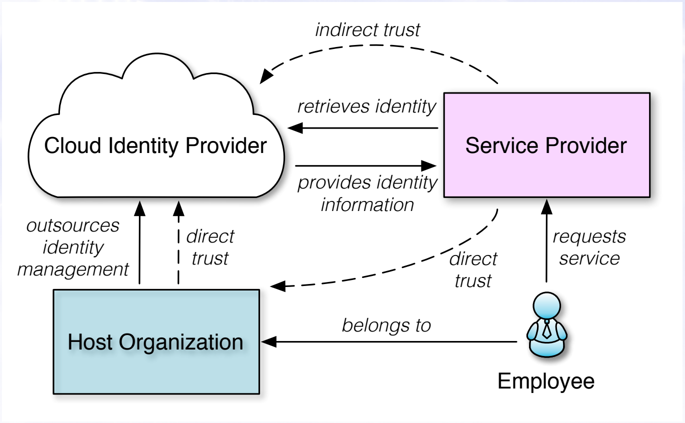
Summary

- We analyze the integration of generic transformations to PRE
- Negative results:
  - It is not possible to apply known transformations directly
  - $>$10 PRE schemes are flawed
- Positive results:
  - FOT can be applied if the PRE scheme satisfies **Perfect Key-Switching** and is IND-CCA$_{0,1}$ secure
  - It achieves IND-CCA$_{2,1}$ security in the random oracle model
- These results can be extended to other generic transformations (e.g., REACT, GEM)

BlindIdM: Motivation

- Identity Management is a ubiquitous service
- Costly $\Rightarrow$ specific applications and personnel
- **Identity Management as a Service (IDaaS)**

BlindIdM: Identity Management as a Service



BlindIdM: Motivation

- Classic problem of cloud computing
  $\Rightarrow$ Organizations lose control of their data
- Now we are talking about **identity** data...
  $\Rightarrow$ Data protection laws and regulations
- Trust problem $\Rightarrow$ Organizations have to trust the provider

Goal: Cryptographic safeguards that support IdM service without compromising users' data

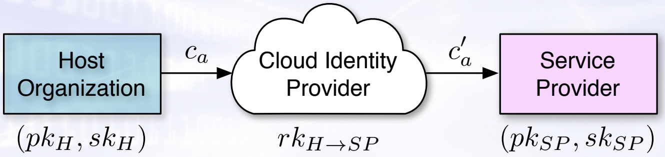
BlindIdM: Proposal

Contribution: BlindIdM

Privacy-preserving IDaaS system based on PRE

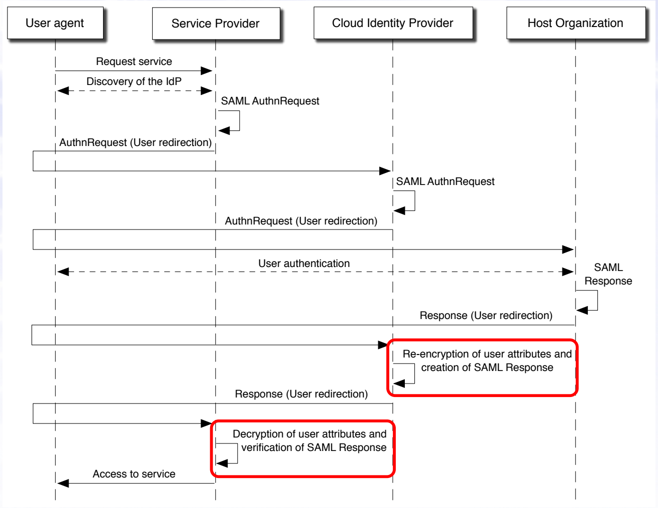
- Identity attributes are encrypted by the user and decrypted by the requester
- The Identity Provider (IdP) stores encrypted attributes
  $\Rightarrow$ Still capable of offering an identity service
- Integrated with SAML 2.0
  $\Rightarrow$ IdM standard for the description and exchange of identity information (e.g., attributes)
- First proposal that tackles this problem

BlindIdM: Our idea



- **Honest-but-curious** provider: The cloud provider will respect protocol fulfillment, but will try to read users' data

BlindIdM: Integration with SAML


User agent
Service Provider
Cloud Identity Provider
Host Organization
Request service
Discovery of the IdP
SAML AuthnRequest
AuthnRequest (User redirection)
SAML AuthnRequest
AuthnRequest (User redirection)
User authentication
SAML Response
Response (User redirection)
Re-encryption of user attributes and creation of SAML Response
Response (User redirection)
Decryption of user attributes and verification of SAML Response
Access to service

Big Data: Motivation

- **Big Data** $\Rightarrow$ use of vast amounts of data that makes processing and maintenance virtually impossible from the traditional perspective of information management

- SMEs are not capable of acquiring and maintaining the infrastructure for running Big Data Analytics on-premise

Big Data in the Cloud

- The Cloud is a natural solution to this problem
  $\Rightarrow$ On-demand high-end clusters

Big Data: Security and privacy challenges

- Multi-tenant environment $\Rightarrow$ Jobs and data from different customers are kept together under the same cluster
- Data may be sensitive or personal
- Malicious agents (insiders and outsiders) can make a profit by selling or exploiting this data
- Security is usually delegated to access control layers

Big Data: Our proposal

Contribution

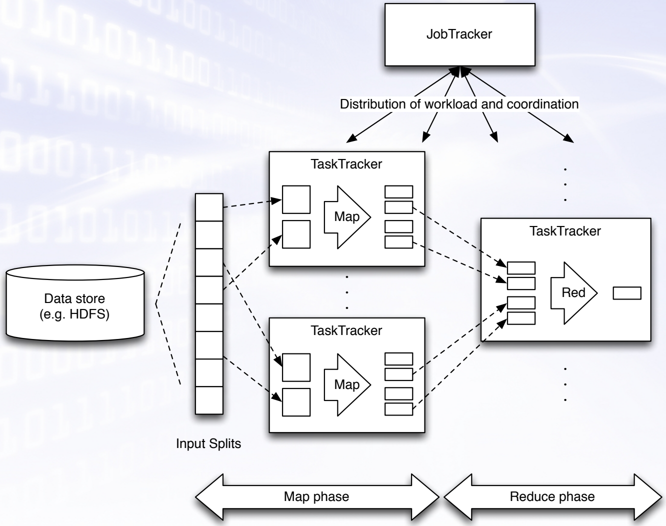
Delegated Access System for Hadoop based on PRE

- Cryptographically-enforced access control system
- Data remains encrypted in the filesystem until it is needed for processing
- Experimental results show that the overhead is reasonable

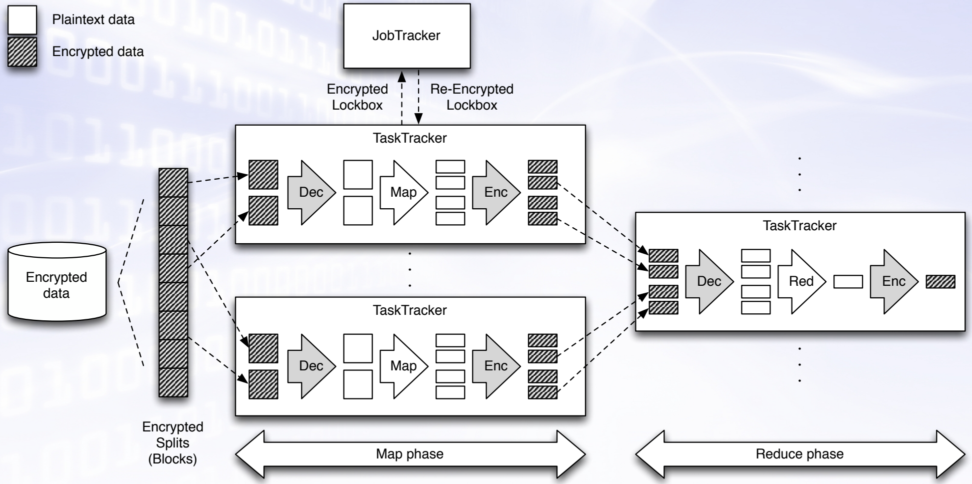
Big Data: Apache Hadoop

- The most prominent framework for processing big datasets.
- Storing and processing of datasets by clusters of machines.
- The workload is divided into parts and distributed throughout the cluster.
- Hadoop was not designed with security in mind
- However, it is widely used by organizations that have strong security requirements regarding data protection.

Big Data: Hadoop operation



Big Data: Our proposal



Big Data: Experiment

- Execution of the WordCount benchmark, a simple application that counts the occurrence of words over a set of files.
- The job input was a set of 1800 encrypted files of 64 MB each
- Almost 30 billions of words, approximately 112.5 GB.
- Execution time:

Clean version:	1932.09 seconds
Prototype:	1960.74 seconds
Difference:	**28.74** seconds → Overhead: **1.49%**

Escrowed Decryption: Motivation

- Dilemma between data confidentiality and law enforcement investigations
- Perception of “impunity” derived from the use of confidential communications.
- Growing concern coming from governments and law enforcement agencies (LEAs)
- Demand for mechanisms that break confidentiality of communications
- Lack of accountability from the government and LEAs

Escrowed Decryption: Original idea

- **Accountable Escrowed Encryption**: Proposed by Liu, Ryan and Chen in CSF 2013
- PKE scheme with **Escrowed Decryption**
- This solution does not involve **key escrow**
- Authorities can request decryption of ciphertexts to a coalition of third-party entities called **custodians**
- Custodians log all requests in a public log ➔ **Accountability**
- Drawbacks:
  - The custodians can decrypt any message if they collude
  - The protocol for escrow decryption is composed of 2 synchronous rounds involving all custodians

Escrowed Decryption: Our idea

Contribution

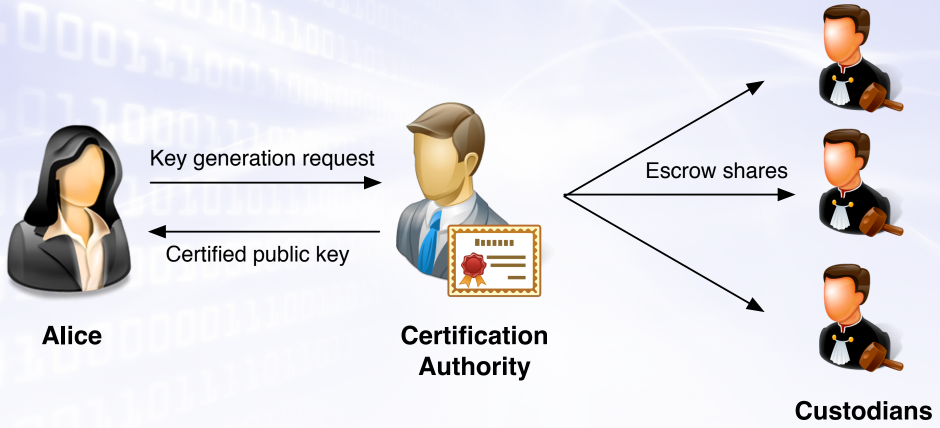
Escrow decryption system based on PRE

- Authorities can request the re-encryption of a ciphertext to custodians and decrypt the result
- Re-encryption should be "shared" by the custodians
  $\Rightarrow$ Re-encryption key is split in **escrow shares**
- Our proposal uses a multiplicative homomorphism:

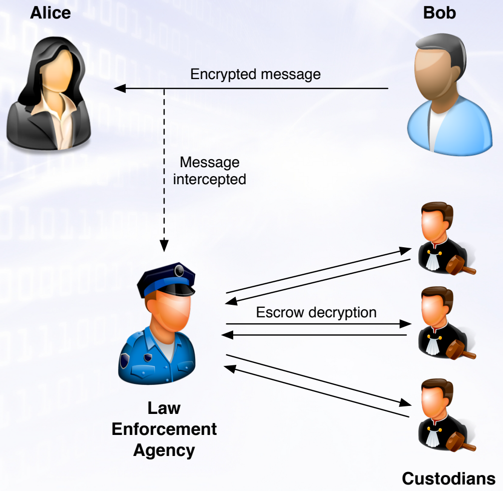
$$rk = \prod_{i=1}^{n} rk_i$$

$$\mathsf{ReEnc}(rk, CT) = \prod_{i=1}^{n} \mathsf{ReEnc}(rk_i, CT)$$

Escrowed Decryption: Our idea



Escrowed Decryption: Our idea



Escrowed Decryption: Analysis

- **IND-CPA secure** under the 1-weak Decisional Bilinear Diffie-Hellman Inversion assumption (1-wDBDHI)
- Our solution only requires **one round of communications** per escrow decryption

Summary: BlindIdM

- We propose an IDaaS system that handles encrypted attributes and still provides an identity service
- Our system is based on SAML and Proxy Re-Encryption
- The cloud identity provider transforms encrypted attributes from the original users to ciphertexts for the requesters using re-encryption

Summary: Big Data

- We propose a cryptographically-enforced access control system for Hadoop, based on PRE
- Stored data is always encrypted and encryption keys do not need to be shared between different data sources.
- Experimental results show that the overhead produced by the encryption and decryption operations is reasonable

Summary: Escrowed Decryption

- We propose an escrowed decryption system based on PRE
- Inspired by the Accountable Escrowed Encryption scheme from Liu, Ryan and Chen
- Escrowed decryption is based on a “shared” re-encryption process
- Authorities can request the escrow decryption to custodians and decrypt the response themselves

Contributions

- We reviewed the basic concepts of PRE:
  - Definitions
  - Security models
  - Properties.
- We analyzed the state of the art:
  - Review of the main PRE schemes
  - Applications of PRE, with a focus secure data sharing problem.

Contributions

- We examined the notions of security for PRE:
  - We propose a parametric family of attack models
  - Fine-grained security notions, whose relations we also analyze.
  - We define the Private Re-Encryption Keys property and show why it is relevant
- We present new proxy re-encryption schemes:
  - NTRUReEncrypt, based on the NTRU cryptosystem and extremely efficient
  - PS-NTRUReEncrypt, a provably-secure version that is CPA-secure under lattice-based assumptions.

Contributions

- We study the application of generic CCA-secure transformations to PRE:
  - We focus on the Fujisaki-Okamoto transformation and formulate sufficient conditions that allow to use it directly in PRE.
  - These conditions include a new property called perfect key-switching
  - We detect flaws in 12 PRE schemes that are allegedly "CCA-secure".
- We propose several applications of PRE:
  - A model for privacy-preserving Identity Management as a Service
  - A system for delegating access to encrypted information in Big Data
  - An escrowed decryption system

Contributions

Security Definitions

Constructions

Applications

- Parametric Family of Attack Models for Proxy Re-Encryption
  ↳**IEEE CSF 2015**
- Application of Generic CCA-Secure Transformations
  ↳**Security and Communication Networks**
- NTRUReEncrypt
  ↳**ACM AsiaCCS 2015**
- Survey of Proxy Re-Encryption Constructions and Applications
  ↳**Journal of Network and Computer Applications**
- Escrowed Decryption System
  ↳**Manuscript**
- Blind Identity Management as a Service
  ↳**International Journal of Information Security**
  ↳**IEEE CloudCom 2012**
- Data Confidentiality in Big Data
  ↳**IEEE CloudCom 2014**
  ↳**Commercial applications and patents**

Publications

Articles in ISI-JCR Journals

- D. Nuñez, I. Agudo, and J. Lopez.
  Proxy Re-Encryption: Analysis of Constructions and its Application to Secure Access Delegation
  Journal of Network and Computer Applications, 87:193-209, 2017.
- D. Nuñez, I. Agudo, and J. Lopez.
  On the Application of Generic CCA-Secure Transformations to Proxy Re-Encryption.
  Security and Communication Networks, 9(12):1769-1785, 2016.
- D. Nuñez, and I. Agudo.
  BlindIdM: A Privacy-Preserving Approach for Identity Management as a Service.
  International Journal of Information Security, 13(2):199-215, 2014.

International conference papers

- D. Nuñez, I. Agudo, and J. Lopez.
  A Parametric Family of Attack Models for Proxy Re-Encryption.
  IEEE CSF 2015, pp. 290-301.
- D. Nuñez, I. Agudo, and J. Lopez.
  NTRUReEncrypt: An Efficient Proxy Re-Encryption Scheme Based on NTRU.
  ACM AsiaCCS 2015, pp. 179-189.
- D. Nuñez, I. Agudo, and J. Lopez.
  Delegated Access for Hadoop Clusters in the Cloud.
  IEEE CloudCom 2014, pp. 374-379.
- D. Nuñez, I. Agudo, and J. Lopez.
  Integrating OpenID with Proxy Re-Encryption to enhance privacy in cloud-based identity services.
  IEEE CloudCom 2012, pp. 241-248.

Open Issues and Future Work

- Family of attack models and security notions for PRE
  - We studied some of the relations between these notions
  - We do not rule out the possibility of additional separations and implications
- NTRUReEncrypt:
  - Achieving CCA-security

Open Issues and Future Work

- Generic transformations:
  - Concrete estimations of the obtained security level
    $\Rightarrow$ Permit to define sets of parameters and to perform meaningful comparison
  - Transformations that are not defined in the random oracle model
- Applications
  - BlindIdM $\Rightarrow$ Integration with other IdM standards
  - Big Data $\Rightarrow$ Improve the integration with the Hadoop
  - Escrow $\Rightarrow$ Reduce trust in the Certification Authority

New Security Definitions, Constructions and Applications of Proxy Re-Encryption

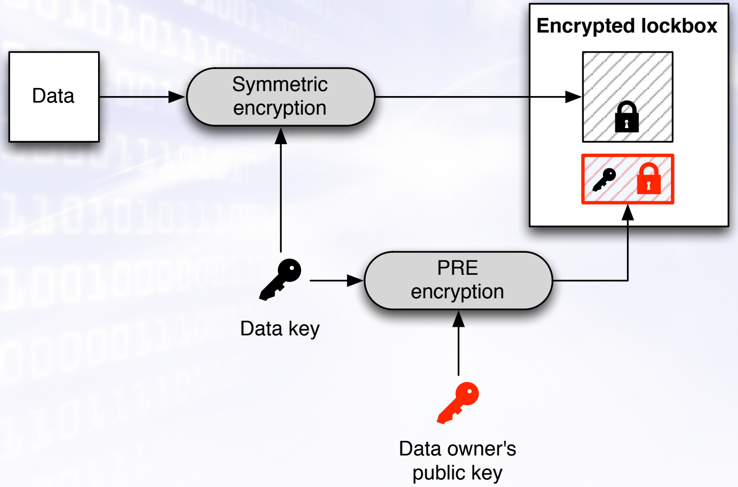
David Nuñez

Advisors:
Isaac Agudo and **Javier Lopez**

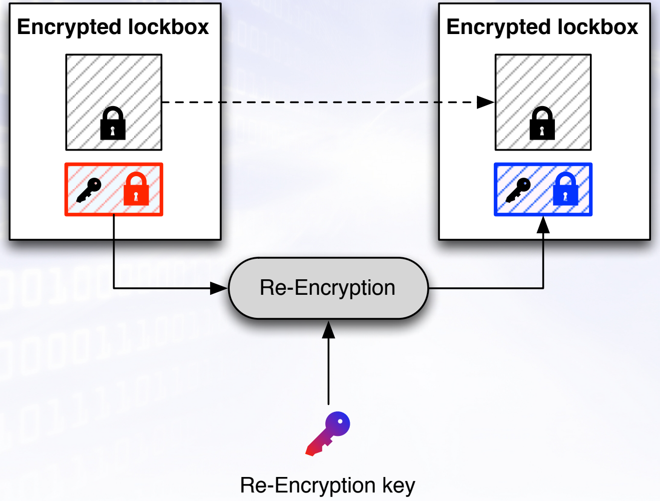
Department of Computer Science
Universidad de Málaga, Spain
Email: dnunez@lcc.uma.es

STM 2017 – Oslo, Norway

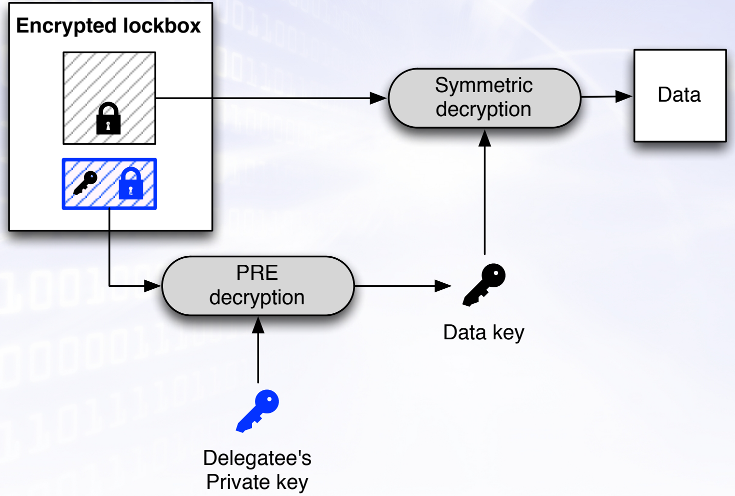
Creating an Encrypted Lockbox



Delegating an Encrypted Lockbox



Opening an Encrypted Lockbox



Definitions of Security for PKE

Definition (IND-atk game (Bellare *et al.*, CRYPTO'98))

Let Π=(KeyGen, Enc, Dec) be a public-key encryption scheme, $A = (A_1, A_2)$ a polynomial-time adversary, and Ω_1 and Ω_2 the set of available oracles for A_1 and A_2, respectively. For $atk \in \{$CPA, CCA1, CCA2$\}$, $n \in \mathbb{N}$, and $\delta \in \{0,1\}$, the **indistinguishability of encryptions** game is defined by the experiment

```
Experiment $\mathbf{Exp}^{\text{IND-}atk}_{\Pi,A,\delta}(n)$
    $(pk^*, sk^*) \overset{R}{\leftarrow} \text{KeyGen}(n);$        $(m_0, m_1, s) \leftarrow A_1(pk^*);$
    $c^* \leftarrow \text{Enc}(pk^*, m_\delta);$                    $d \leftarrow A_2(m_0, m_1, s, c^*);$
    return $d$
```

where

If $atk = \text{CPA}(= \text{CCA0})$	then $\Omega_1 = \varnothing$	and $\Omega_2 = \varnothing$
If $atk = \text{CCA1}$	then $\Omega_1 = \{\mathcal{O}_{dec}\}$	and $\Omega_2 = \varnothing$
If $atk = \text{CCA2}$	then $\Omega_1 = \{\mathcal{O}_{dec}\}$	and $\Omega_2 = \{\mathcal{O}_{dec}\}$

PRE Oracles

- Honest key generation $\mathcal{O}_{honest}$: The oracle obtains a new keypair $(pk_i, sk_i) \leftarrow \mathsf{KeyGen}(n)$, and returns the public key pk_i.
- Corrupt key generation $\mathcal{O}_{corrupt}$: The oracle obtains a new keypair $(pk_i, sk_i) \leftarrow \mathsf{KeyGen}(n)$, and returns the pair (pk_i, sk_i).
- Re-encryption key generation $\mathcal{O}_{rkgen}$: On input a pair of public keys (pk_i, pk_j), the oracle returns the re-encryption key $rk_{i \to j} \leftarrow \mathsf{ReKeyGen}(pk_i, sk_i, pk_j, sk_j)$.
  - The adversary is only allowed to make queries where $i \neq j$, and i and j are both either honest or corrupt.

PRE Oracles

- Re-encryption $\mathcal{O}_{reenc}$: On input (pk_i, pk_j, c), where $i \neq j$ the oracle returns the re-encrypted ciphertext $c' \leftarrow \mathsf{ReEnc}(rk_{i \to j}, c)$.
    - The adversary is not allowed to make queries where j is a corrupt user and (pk_i, c) is a derivative of (pk^*, c^*).
- Decryption $\mathcal{O}_{dec}$: On input (pk_i, c), the oracle returns $m \leftarrow \mathsf{Dec}(sk_i, c)$.
    - The adversary is not allowed to make queries where (pk_i, c) is a derivative of (pk^*, c^*).

Derivatives of the challenge

Definition (Derivatives of the challenge (Canetti and Hohenberger, CCS'07))

The set of derivatives of (pk^*, c^*) is defined inductively, as follows:

- (pk^*, c^*) is a derivative of itself.
- If (pk_j, c_j) is a derivative of (pk_i, c_i) and (pk_i, c_i) is a derivative of (pk^*, c^*), then (pk_j, c_j) is a derivative of (pk^*, c^*).
- If the adversary has issued a $\mathcal{O}_{reenc}$ query (pk_i, pk_j, c_i) and obtained a ciphertext c_j as response, then (pk_j, c_j) is a derivative of (pk_i, c_i).
- If the adversary has issued a $\mathcal{O}_{rkg}$ query (pk_i, pk_j), and $\mathsf{Dec}(pk_j, c_j) \in \{m_0, m_1\}$, then (pk_j, c_j) is a derivative of all pairs (pk_i, c).

Private Re-Encryption Keys

Definition (Private Re-Encryption Keys)

Let Π be a proxy re-encryption scheme and A a polynomial-time adversary. Let $\Omega = \{\mathcal{O}_{honest}, \mathcal{O}_{corrupt}, \mathcal{O}_{reenc}\}$ be the set of available oracles for A, and pk the public key of a honest user. The scheme Π satisfies the *private re-encryption keys* property if the probability that A computes a valid $rk_{pk \rightarrow pk'}$, for some public key pk' of her choice, is negligible.

Separation strategies

We show two separation strategies:

1. Leaking re-encryption keys from the target user to a **honest** user
   - The adversary does not know the secret key of honest users
2. Leaking re-encryption keys from the target user to a **corrupt** user
   - The adversary knows the secret key of corrupt users

Exploiting the target-to-honest re-encryption key leakage

- Suppose that a PRE scheme leaks $rk_{pk^* \to pk_h}$ through $\mathcal{O}_{reenc}$
- The attack strategy works as follows:
  1. Extract $rk_{pk^* \to pk_h}$ from $\mathcal{O}_{reenc}$ queries
  2. Obtain the challenge ciphertext c^*
  3. Re-encrypt the challenge ciphertext locally: $c' = \mathsf{ReEnc}(rk_{pk^* \to pk_h}, c^*)$
  4. Call the decryption oracle $\mathcal{O}_{dec}$ with (pk_h, c') to obtain m_δ
- We need $\mathcal{O}_{reenc}$ at some phase and $\mathcal{O}_{dec}$ in phase 2
- Separation IND-CCA$_{2,0} \not\Rightarrow$ IND-CCA$_{2,1}$ (Theorem 4.7)

Exploiting the target-to-corrupt re-encryption key leakage

- Suppose that a PRE scheme leaks $rk_{pk^* \to pk_x}$ through $\mathcal{O}_{reenc}$
- The attack strategy works as follows:
  1. Extract $rk_{pk^* \to pk_x}$ from $\mathcal{O}_{reenc}$ queries
  2. Obtain the challenge ciphertext c^*
  3. Re-encrypt the challenge ciphertext locally: $c' = \mathsf{ReEnc}(rk_{pk^* \to pk_x}, c^*)$
  4. Decrypt c' locally: $m_\delta = \mathsf{Dec}(sk_x, c')$
- We only need $\mathcal{O}_{reenc}$ at some phase
- Separation IND-CCA$_{2,0} \not\Rightarrow$ IND-CCA$_{0,1}$ (Theorem 4.8)

Impossibility result

A PRE scheme that violates the private re-encryption keys property can only be IND-CCA$_{i,0}$, for $i \in \{0, 1, 2\}$ (i.e., a security notion without $\mathcal{O}_{reenc}$)

The PRE scheme from Kirshanova (PKC'14)

KeyGen(n):

1. Choose $A_0 \leftarrow \mathbb{Z}_q^{n\times\bar{m}}$, $R_1, R_2 \leftarrow D_R$ and an invertible matrix $H \leftarrow \mathbb{Z}_q^{nk\times nk}$
2. Define $A_1 = -A_0 R_1 \in \mathbb{Z}_q^{n\times nk}$ and $A_2 = -A_0 R_2 \in \mathbb{Z}_q^{n\times nk}$
3. Compose the matrix $A = [A_0|A_1|A_2] \in \mathbb{Z}_q^{n\times m}$
4. The public key is the pair $pk = (A, H)$. The secret key is matrix $sk = [R_1|R_2] \in \mathbb{Z}^{\bar{m}\times 2nk}$

The PRE scheme from Kirshanova (PKC'14)

Enc$(pk = ([A_0|A_1|A_2], H), m \in \{0,1\}^{nk})$:

1. Choose a non-zero invertible matrix H_u, and a vector $s \leftarrow \mathbb{Z}_q^n$
2. Set $A_u = [A_0|A_1 + HG|A_2 + H_uG]$
3. Sample error vector $e \leftarrow D_e$
4. Compute
$$b^t = 2(s^t A_u \mod q) + e^t + (0, 0, enc(m))^t \mod 2q$$
where the first zero vector has dimension $\bar{m}$, the second has dimension nk and enc is an encoding function
5. Output the ciphertext $c = (H_u, b) \in \mathbb{Z}_q^{n\times n} \times \mathbb{Z}_{2q}^m$

➔ The main part of the ciphertext is **vector** b

The PRE scheme from Kirshanova (PKC'14)

Dec$(pk = ([A_0|A_1|A_2], H), sk = [R_1|R_2], c = (H_u, b))$:

1. Using matrix H_u, compute $A_u = [A_0|A_1 + HG|A_2 + H_uG]$
2. With the secret key call algorithm Invert$_O([R_1|R_2], A_u, b \mod q, H_u)$. As output we receive two vectors $z \in \mathbb{Z}_q^n$ and $e \in \mathbb{Z}_q^m$ that satisfy $b^t = z^t A + e^t \mod q$
3. Let $v = b - e \mod 2q$
4. Compute

$$v^t \begin{bmatrix} R_1 & R_2 \\ I & 0 \\ 0 & I \end{bmatrix} \mod 2q$$

and apply enc^{-1} to the last nk coordinates

The PRE scheme from Kirshanova (PKC'14)

ReKeyGen$(pk = ([A_0|A_1|A_2], H), sk = [R_1|R_2], pk' = ([A'_0|A'_1|A'_2], H'))$:

1. Let $Y = [A'_0|A'_1 + H'G|A'_2 - A_2]$ and y_i be the i-th column of Y
2. Execute Sample$_O(y_i, [A_0|A_1], R_1, H)$ for each column vector y_i and concatenate the column vector outputs to form matrix X. This matrix satisfies that $[A_0|A_1]X = Y$
3. Parse matrix X as $[X_0|X_1|X_2]$, where the block $X_0 \in \mathbb{Z}^{(\bar{m}+nk)\times\bar{m}}$ is the output corresponding to the first part of Y, $X_1 \in \mathbb{Z}^{(\bar{m}+nk)\times nk}$ to the second one, and $X_2 \in \mathbb{Z}^{(\bar{m}+nk)\times nk}$ to the last part
4. Finally, output the re-encryption key $rk_{pk\to pk'}$:

$$rk_{pk\to pk'} = \begin{bmatrix} X_0 & X_1 & X_2 \\ 0 & 0 & I \end{bmatrix}$$

→ The re-encryption key is a matrix

The PRE scheme from Kirshanova (PKC'14)

ReEnc$(rk_{pk\to pk'}, c = (H_u, b))$: to change the underlying public key in the ciphertext component b, compute

$$b'^t = b^t \cdot rk_{pk\to pk'}$$

Finally, output $c' = (H_u, b')$.

→ The only operation during re-encryption is the **multiplication** of a vector b (the ciphertext) by a matrix (the re-encryption key), obtaining another vector b' (the re-encrypted ciphertext)

Finalizing the attack

- Using the unit vectors as input ciphertexts to the re-encryption oracle, one can extract arbitrary rows of the re-encryption key matrix.
- The adversary queries the re-encryption oracle from the target user to a corrupt one.
- After $\bar{m} + nk$ queries the adversary can reconstruct the re-encryption key $rk_{pk^* \to pk_x}$
- The adversary continues the generic attack strategy when target-to-corrupt re-encryption keys are leaked:
    1. Extract $rk_{pk^* \to pk_x}$ from $\mathcal{O}_{reenc}$ queries
    2. Obtain the challenge ciphertext c^*
    3. Re-encrypt the challenge ciphertext locally: $c' = \mathsf{ReEnc}(rk_{pk^* \to pk_x}, c^*)$
    4. Decrypt c' locally: $m_\delta = \mathsf{Dec}(sk_x, c')$

Refining the attack

- One could argue that unit vectors don't look as ciphertexts...
- In fact, the probability that a unit vector is output from the encryption function is negligible.
- The oracle could take the risk and reject them or return garbage.
- The adversary doesn't know whether these ciphertexts were valid encryptions under pk^*

Refining the attack

- Instead using unit vectors as ciphertexts, the adversary uses the encryption of random messages.
- She constructs a square matrix P out of these ciphertexts.
- The LWE assumption ensures that the distribution of ciphertexts is indistinguishable from the uniform distribution.
- The matrix P will be invertible with overwhelming probability. Otherwise, resample and produce a new P.
- After m queries to $\mathcal{O}_{reenc}$, the adversary obtains $P \cdot rk$
- She only has to compute $P^{-1} \cdot P \cdot rk = rk$
- She continues with the attack as before.

NTRUEncrypt: Setup and Key Generation

Setup:

- Quotient ring $\mathcal{R}_{NTRU} = \mathbb{Z}[x]/(x^n - 1)$, where n is a prime parameter
- Other parameters: Integer q, small polynomial $p \in \mathcal{R}_{NTRU}$
- Operations are performed in $\mathcal{R}_{NTRU}/q$ or $\mathcal{R}_{NTRU}/p$

Private key: $sk = f$

- f is chosen at random $\in \mathcal{R}_{NTRU}$
- f must be invertible in $\mathcal{R}_{NTRU}/q \Rightarrow f^{-1}$
- f must be congruent to $1 \mod p$

Public key: $pk = h = p \cdot g \cdot f^{-1} \mod q$

- $g \in \mathcal{R}_{NTRU}$ is chosen at random

NTRUEncrypt: Encryption and Decryption

Encryption:

- plaintext M from message space $\mathcal{R}_{NTRU}/p$
- public key $h = p \cdot g \cdot f^{-1} \mod q$
- noise term s is a small random polynomial in $\mathcal{R}_{NTRU}$
- ciphertext $C = h \cdot s + M \mod q = p \cdot g \cdot s \cdot f^{-1} + M \mod q$

Decryption:

- Secret key f
- Compute $C' = f \cdot C \mod q$
- Output $m = C' \mod p$

Why does it work?

- $C' = f \cdot (p \cdot g \cdot f^{-1} \cdot s + M) \mod q = p \cdot g \cdot s + f \cdot M \mod q$
- $\cancel{p \cdot g \cdot s} + f \cdot M \mod p = f \cdot M \mod p = M$
- Decryption is consistent if $f \cdot C$ is "*small enough*"

NTRUReEncrypt

- Setup and Key Generation are identical to NTRUEncrypt
- Private key: $sk_A = f_A \in \mathcal{R}_{NTRU}$
- Public key: $pk_A = h_A = p \cdot g_A \cdot f_A^{-1} \mod q$

NTRUReEncrypt: Encryption and Decryption

Encryption:

- plaintext M from message space $\mathcal{R}_{NTRU}/p$
- public key h_A
- ciphertext $C_A = h_A \cdot s + M \mod q$
- noise term s is a small random polynomial in $\mathcal{R}_{NTRU}$

Decryption:

- Compute $C'_A = f \cdot C_A \mod q$
- Compute $m = C'_A \mod p$

NTRUReEncrypt: Re-Encryption Key Generation

Re-Encryption Key Generation:

- Input: secret keys $sk_A = f_A$ and $sk_B = f_B$
- The re-encryption key between users A and B is

$$rk_{A \to B} = sk_A \cdot sk_B^{-1} = f_A \cdot f_B^{-1}$$

- Three-party protocol, so neither A, B nor the proxy learns any secret key.
  - A selects a random $r \in \mathcal{R}_{NTRU}/q$
  - A sends $r \cdot f_A \mod q$ to B and r to the proxy
  - B sends $r \cdot f_A \cdot f_B^{-1} \mod q$ to the proxy
  - The proxy computes $rk_{A \to B} = f_A \cdot f_B^{-1} \mod q$

NTRUReEncrypt: Re-Encryption

Re-Encryption

- Input: a re-encryption key $rk_{A\rightarrow B}$ and a ciphertext C_A
- Samples a random polynomial $e \in \mathcal{R}_{NTRU}$
- Output re-encrypted ciphertext

$$C_B = C_A \cdot rk_{A\rightarrow B} + p \cdot e$$

- The noise e prevents B from extracting A's private key

NTRUReEncrypt: Re-Encryption

Why does it work?

- Re-encrypted ciphertext:

$$\begin{aligned} C_B &= C_A \cdot rk_{A\to B} + p \cdot e \mod q \\ &= (p \cdot g \cdot f_A^{-1} \cdot s + M) \cdot f_A \cdot f_B^{-1} + p \cdot e \mod q \\ &= p \cdot g \cdot f_B^{-1} \cdot s + f_A \cdot f_B^{-1} \cdot M + p \cdot e \mod q \end{aligned}$$

- Decrypting a re-encrypted ciphertext:

$$\begin{aligned} f_B \cdot C_B \mod p &= \cancel{(p \cdot g \cdot s + p \cdot e)} + f_A \cdot M \mod p \\ &= f_A \cdot M \mod p \\ &= M \end{aligned}$$

NTRUReEncrypt: Analysis

- Bidirectional: Given $rk_{A\to B} = f_A f_B^{-1}$, one can easily compute

$$rk_{B\to A} = (rk_{A\to B})^{-1} = f_B f_A^{-1}$$

- Limited multihop
- Not collusion-safe: Secret keys can be extracted from the re-encryption key if the proxy colludes with a user involved

$$f_A = rk_{B\to A} \cdot f_B$$

- This is common in interactive bidirectional PRE schemes

Preliminaries

- $\Phi(x)$ is the cyclotomic polynomial $x^n + 1$, with n a power of 2
- q is a prime integer such that $q = 1 \mod 2n$
- $\mathcal{R}$ is the ring $\mathbb{Z}[x]/\Phi(x)$
- $\mathcal{R}_q = \mathcal{R}/q = \mathbb{Z}_q[x]/\Phi(x)$
- $\mathcal{R}_q^\times$ is the set of invertible elements of $\mathcal{R}_q$

The Ring-LWE problem

- The **Ring Learning With Errors** (Ring-LWE) problem is a hard decisional problem based on lattices
- We use a variant of this problem proposed by Stehlé and Steinfeld.
- $s \in \mathcal{R}_q$ and ψ a distribution over $\mathcal{R}_q^{\times}$
- $A_{s,\psi}^{\times}$ is the distribution that samples pairs of the form (a, b)
  - a is chosen uniformly from $\mathcal{R}_q^{\times}$
  - $b = a \cdot s + e$, for some e sampled from ψ
- The Ring-LWE problem is to distinguish distribution $A_{s,\psi}^{\times}$ from a uniform distribution over $\mathcal{R}_q^{\times} \times \mathcal{R}_q$
- The Ring-LWE assumption is that this problem is computationally infeasible

PS-NTRUReEncrypt: Setup and Key Generation

Setup:

- Global parameters: $(n, q, p, \alpha, \sigma)$

Key Generation:

- $D_{\mathbb{Z}^n,\sigma}$ is a Gaussian distribution over $\mathbb{Z}^n$ with standard deviation σ
- The keys are computed as follows:
  1. Sample f' from $D_{\mathbb{Z}^n,\sigma}$
     Let $f_A = 1 + p \cdot f'$; if $(f_A \mod q) \notin \mathcal{R}_q^\times$, resample
  2. Sample g_A from $D_{\mathbb{Z}^n,\sigma}$; if $(g_A \mod q) \notin \mathcal{R}_q^\times$, resample
  3. Compute $h_A = p \cdot g_A \cdot f_A^{-1}$
  4. Return secret key $sk_A = f_A$ and $pk_A = h_A$

PS-NTRUReEncrypt: Encryption and Decryption

Encryption:

- Input: public key pk_A and message $M \in \mathcal{M}$
- Sample noise polynomials s, e from a distribution Ψ_α
- Output ciphertext:

$$C_A = h_A s + pe + M \in \mathcal{R}_q$$

Decryption:

- Input: secret key $sk_A = f_A$ and ciphertext C_A
- Compute $C'_A = C_A \cdot f_A$
- Output the message $M = (C'_A \mod p) \in \mathcal{M}$

PS-NTRUReEncrypt: Re-Encryption Key Generation and Re-Encryption

Re-Encryption Key Generation:

- Input: secret keys $sk_A = f_A$ and $sk_B = f_B$
- The re-encryption key between users A and B is

$$rk_{A \to B} = sk_A \cdot sk_B^{-1} = f_A \cdot f_B^{-1}$$

Re-Encryption:

- Input: a re-encryption key $rk_{A \to B}$ and a ciphertext C_A
- Samples a random polynomial e' from a distribution Ψ_α
- Output re-encrypted ciphertext

$$C_B = C_A \cdot rk_{A \to B} + pe'$$

Multihop Correctness

Ciphertext re-encrypted N times:

$$\begin{aligned} C_N &= pg_0 f_N^{-1} s + pe_0 f_0 f_N^{-1} + pe_1 f_1 f_N^{-1} + \ldots \\ &\qquad + pe_{N-1} f_{N-1} f_N^{-1} + pe_N + M f_0 f_N^{-1} \\ &= pg_0 f_N^{-1} s + \left[\sum_{i=0}^{N-1} pe_i f_i f_N^{-1} \right] + pe_N + M f_0 f_N^{-1} \end{aligned}$$

When decrypting C_N (*assuming no decryption failures*):

$$C'_N = C_N \cdot f_N = pg_0 s + \left[\sum_{i=0}^{N} pe_i f_i \right] + M f_0$$

Since, $f_0 = 1 \mod p$ and $pg_0 s = pe_i f_i = 0 \mod p$, then:

$$C'_N \mod p = M$$

Experimental setting

- Implementation of our proposals:
  - NTRUReEncrypt is implemented on top of an available open-source Java implementation of NTRU
  - PS-NTRUReEncrypt was coded from scratch, using the Java Lattice-Based Cryptography (jLBC) library
- Execution enviroment: Intel Core 2 Duo @ 2.66 GHz

Performance of NTRUReEncrypt

Table: Computation time (in ms) and number of hops of NTRUReEncrypt for different parameters

Parameters	Enc.	Dec.	Re-Enc.	# Hops
(439, no, 128)	0.64	0.30	0.24	5
(439, yes, 128)	0.16	0.30	0.23	5
(1087, no, 256)	1.39	1.25	1.05	21
(1087, yes, 256)	0.48	1.26	1.07	15
(1171, no, 256)	0.80	1.12	1.14	21
(1171, yes, 256)	**0.43**	**1.22**	**1.15**	**14**
(1499, no, 256)	0.74	1.78	1.73	50
(1499, yes, 256)	0.32	1.67	1.66	42

Comparison of NTRUReEncrypt to other schemes

Table: Computation time of several proxy re-encryption schemes (in ms)

Scheme	Enc.	Dec.	Re-Enc.
NTRUReEncrypt	0.43	1.22	1.15
Aono et al	1.17	0.47	20.5
BBS	11.07	11.21	11.48
Weng et al	22.52	11.89	22.29
Ateniese et al	22.76	13.76	83.52
Libert and Vergnaud	155.27	443.87	386.93

Performance of PS-NTRUReEncrypt

Table: Computation time (in ms) and size (in KB) of PS-NTRUReEncrypt for different parameters

n	$\log_2 q$	Enc.	Dec.	Re-Enc.	Size
32	23	0.93	0.99	1.05	0.09
64	28	4.53	4.23	4.32	0.22
128	32	17.28	17.32	17.45	0.50
256	37	80.64	81.045	86.56	1.16
512	41	333.75	334.07	359.54	2.56
1024	46	1333.03	1344.10	1461.46	5.75

Perfect Key-Switching: Example

- PRE scheme from Blaze, Bleumer, and Strauss
- Based on ElGamal:
  - Private key: $a \in \mathbb{Z}_q$ Public key: $pk = g^a$
  - Ciphertexts: $((g^a)^r, g^r \cdot m)$, random r
  - Re-Encryption Key: $rk_{A \to B} = b/a$
  - Re-Encryption: $((g^{ar})^{b/a}, g^r \cdot m)$
- Satisfies the Perfect Key-Switching property:

$$\begin{aligned} & \mathsf{ReEnc}(rk_{A \to B}, \mathsf{Enc}(pk_A, m; r)) \\ = {} & \mathsf{ReEnc}(b/a, ((g^a)^r, g^r \cdot m)) \\ = {} & ((g^{ar})^{b/a}, g^r \cdot m) \\ = {} & (g^{br}, g^r \cdot m) \\ = {} & \mathsf{Enc}(pk_B, m; r) \end{aligned}$$

Economic analysis

- Most of proposals do not analyze their economic impact
- Cryptographic operations have an economic cost due to computation, communication, etc.
  $\Rightarrow$ Cloud provider incurs in expenses due to energy consumption, personnel, ...
- Our estimations are based on a research from Chen & Sion
  $\Rightarrow$ They give estimations for computation, storage and communication costs, expressed in *picocents* (1 picocent $= 10E^{-12}$ USD cent)
- We estimate the number of CPU cycles to give an approximation of the costs

Economic analysis: costs

Table: Costs in picocents for the main operations

Operation	Cost per operation	Operations per cent
Encryption	4.34E+08	2304
Re-encryption	4.79E+08	2087
Decryption	5.70E+08	1755

Economic analysis: example scenario

- IDaaS provider that handles 1 million attribute requests per day $\Rightarrow$ 1 million re-encryptions per day
- Approx. 2000 USD per year
- Reasonable cost for an average-sized company, considering that their information is encrypted at the cloud provider

DASHR

- Delegated Access System for Hadoop based on Re-Encryption
- Data is stored encrypted in the cluster and the owner can delegate access rights to the computing cluster for processing.
- The data lifecycle is composed of three phases:
  1. Production phase: data is generated by different data sources, and stored encrypted under the owner's public key for later processing.
  2. Delegation phase: the data owner produces the necessary master re-encryption key for initiating the delegation process.
  3. Consumption phase: This phase occurs each time a user of the Hadoop cluster submits a job; is in this phase where encrypted data is read by the worker nodes of the cluster. At the beginning of this phase, re-encryption keys for each job are generated.

DASHR: Production phase

- Generation of data by different sources
- Data is split into blocks by the filesystem (e.g., HDFS)
- Each block is an encrypted lockbox, which contains encrypted data and an encrypted key, using the public key of the data owner pk_{DO}

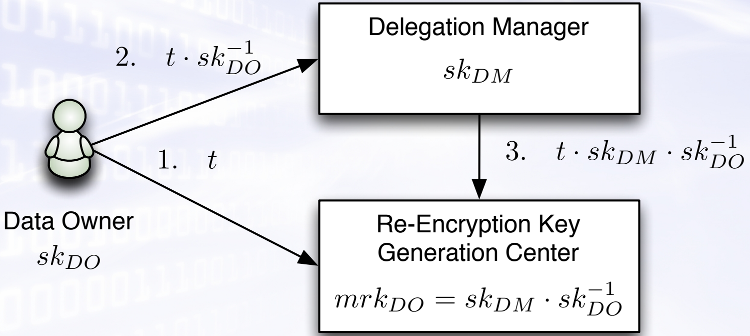
DASHR: Delegation phase

- The dataset owner produces a master re-encryption key mrk_{DO} to allow the delegation of access to the encrypted data
- The master re-encryption key is used to derive re-encryption keys in the next phase.
- The delegation phase is done only once for each computing cluster

DASHR: Delegation phase

- This phase involves the interaction of three entities:
  1. Dataset Owner (DO), with a pair of public and secret keys (pk_{DO}, sk_{DO}), the former used to encrypted generated data for consumption
  2. Delegation Manager (DM), with keys (pk_{DM}, sk_{DM}), and which belongs to the security domain of the data owner, so it is assumed trusted. It can be either local or external to the computing cluster. If it is external, then the data owner can control the issuing of re-encryption keys during the consumption phase. The delegation manager has a pair of public and secret keys, pk_{DM} and sk_{DM}.
  3. Re-Encryption Key Generation Center (RKGC), which is local to the cluster and is responsible for generating all the re-encryption keys needed for access delegation during the consumption phase.

DASHR: Delegation Protocol

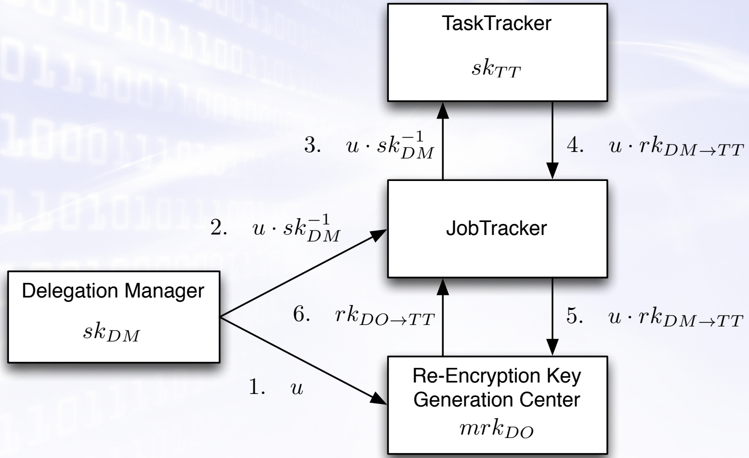


Delegation Manager
sk_{DM}
2. $t \cdot sk_{DO}^{-1}$
1. t
3. $t \cdot sk_{DM} \cdot sk_{DO}^{-1}$
Data Owner
sk_{DO}
Re-Encryption Key Generation Center
$mrk_{DO} = sk_{DM} \cdot sk_{DO}^{-1}$

DASHR: Consumption phase

- This phase is performed each time a user submits a job to the Hadoop cluster
- A pair of public and private keys (pk_{TT}, sk_{TT}) for the TaskTrackers is initialized in this step, which will be used later during encryption and decryption.
- The Delegation Manager, the Re-Encryption Key Generation Center, the JobTracker and the TaskTrackers interact in order to generate the re-encryption key $rk_{DO \to TT}$
- The final re-encryption key $rk_{DO \to TT}$ held by the JobTracker, who will be the one performing re-encryptions. This process could be repeated in case that more TaskTrackers' keys are in place.

DASHR: Re-Encryption Key Generation Protocol


TaskTracker
sk_{TT}
JobTracker
Delegation Manager
sk_{DM}
Re-Encryption Key Generation Center
mrk_{DO}
1. u
2. $u \cdot sk_{DM}^{-1}$
3. $u \cdot sk_{DM}^{-1}$
4. $u \cdot rk_{DM \to TT}$
5. $u \cdot rk_{DM \to TT}$
6. $rk_{DO \to TT}$

Experimental setting

- Focused on the main part of the consumption phase, where the processing of the data occurs.
- From the Hadoop perspective, the other phases are offline processes, since are not related with Hadoop's flow.
- Environment:
    - Virtualized environment on a rack of IBM BladeCenter HS23 servers connected through 10 gigabit Ethernet, running VMware ESXi 5.1.0.
    - Each of the blade servers is equipped with two quad-core Intel(R) Xeon(R) CPU E5-2680 @ 2.70GHz.
    - Cluster of 17 VMs (1 master node and 16 slave nodes)
    - Each of the VMs has two logical cores and 4 GB of RAM, running a modified version of Hadoop 1.2.1.

Experimental setting: Cryptographic details

- Proxy Re-Encryption scheme from Weng et al.
- Implemented using the NIST P-256 curve, which provides 128 bits of security and is therefore appropriate for encapsulating 128 bits symmetric keys.
- AES-128-CBC for symmetric encryption.
- We make use of the built-in support for AES included in some Intel processors through the AES-NI instruction set.

Time costs

Operation	**Time (ms)**
Block Encryption (AES-128-CBC, ∼64 MB)	212.62
Block Decryption (AES-128-CBC, ∼64 MB)	116.81
Lockbox Encryption (PRE scheme)	17.84
Lockbox Re-Encryption (PRE scheme)	17.59
Lockbox Decryption (PRE scheme)	11.66

Our idea

- Setup:
  - $\mathbb{G}$ and $\mathbb{G}_T$ are cyclic groups of order q
  - $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a bilinear pairing.
  - g is a generator of $\mathbb{G}$, and $Z = e(g, g)$
- Actors:
  - Users
  - Escrow Authority EA: $pk_{EA} = g^a, sk_{EA} = a$
  - Custodians
  - Certification Authority (CA)

Key Generation

U | CA

$u, \beta \leftarrow \mathbb{Z}_q^*$

$\widetilde{\kappa}_i \in \mathbb{G}$, for $2 \leq i \leq n$

$$\widetilde{\kappa}_1 = \frac{(pk_{EA})^{\beta/u}}{\prod_{i=2}^{n} \widetilde{\kappa}_i}$$

$$\xrightarrow{g^u, \{\widetilde{\kappa}_i\}_{i=1}^n, g^\beta}$$

Check: $e(\prod_{i=1}^{n} \widetilde{\kappa}_i, g^u) \overset{?}{=} e(pk_{EA}, g^\beta)$

$s, \gamma \leftarrow \mathbb{Z}_q^*$

$pk = ((g^u)^s, e(g^\beta, g)^{s\gamma})$

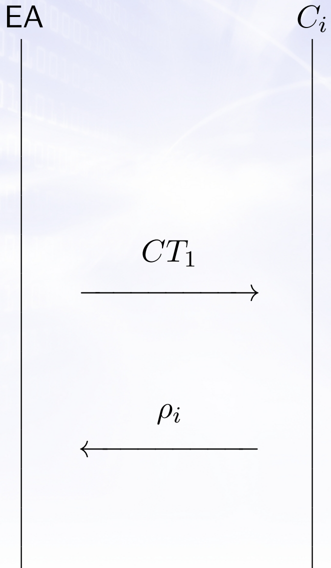
$\kappa_i = (\widetilde{\kappa}_i)^\gamma$

$$\xleftarrow{pk, g^\gamma}$$

Escrow Decryption

$$\begin{aligned} CT &= (CT_1, CT_2) \\ &= (g^{sur}, m \cdot Z^{svr}) \end{aligned}$$

For each custodian C_i:



$$\begin{aligned} \rho_i &= \mathsf{ShareReEnc}(\kappa_i, CT_1) \\ &= e(\kappa_i, g^{sur}) \end{aligned}$$

Collect $\{\rho_i\}_{i=1}^n$

$m = \mathsf{Comb}(sk_{EA}, CT, \{\rho_i\}_{i=1}^n)$

Work in progress

- Translate this solution to asymmetric pairings
- Security can be based on a more common assumption: External Diffie-Hellman (XDH)
- *Threshold Re-Encryption*: integration with Shamir's Secret Sharing