Design and Implementation of Lattice-Based Cryptography

Tancrède Lepoint

École Normale Supérieure & Université du Luxembourg
Thèse CIFRE effectuée au sein de CryptoExperts

Soutenance de thèse de doctorat – 30 juin 2014

Outline

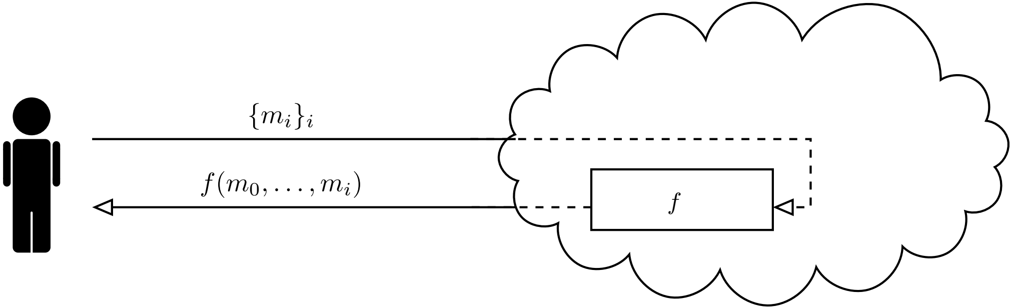
1. Introduction
2. Fully Homomorphic Encryption
3. Cryptographic Multilinear Maps
4. Conclusion

Outline

Cloud Computing

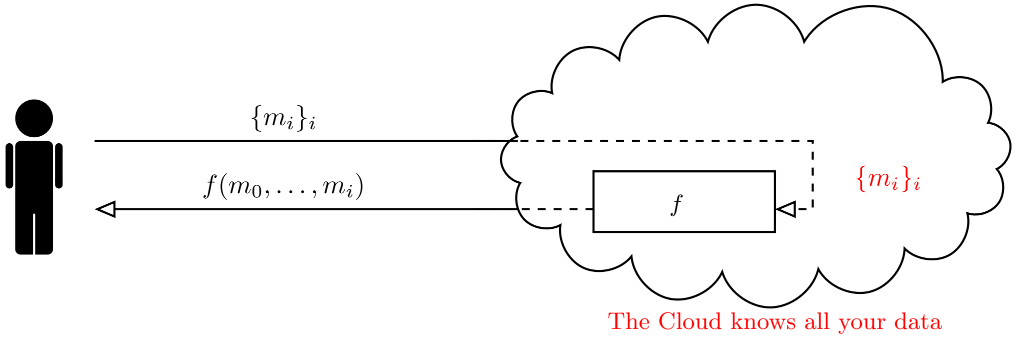
Program or application on connected server(s) rather than locally

Modelization



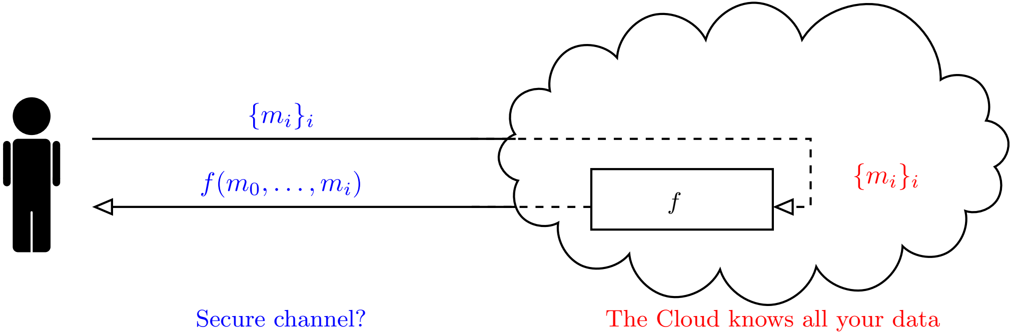
f is the service provided by the Cloud on your data m_i

Confidentiality of Your Data



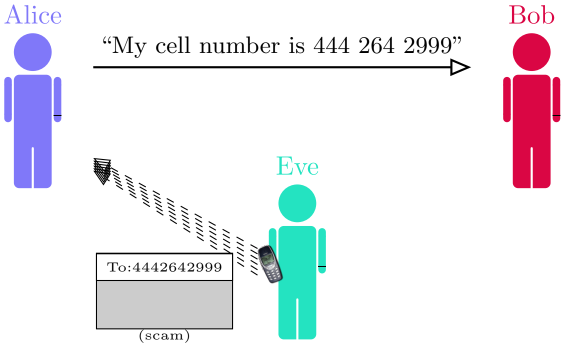
1. Confidentiality of your data in the Cloud?

Confidentiality of Your Data



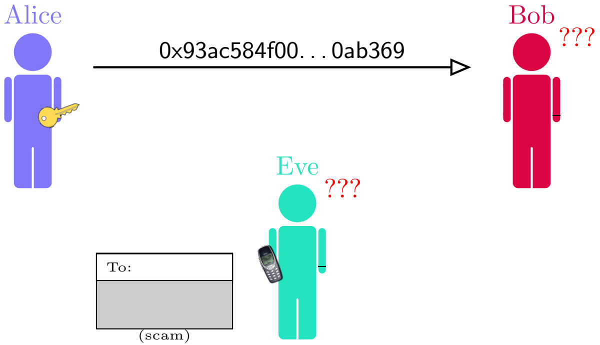
1. Confidentiality of your data in the Cloud?
2. Confidentiality of the channel?

Encryption

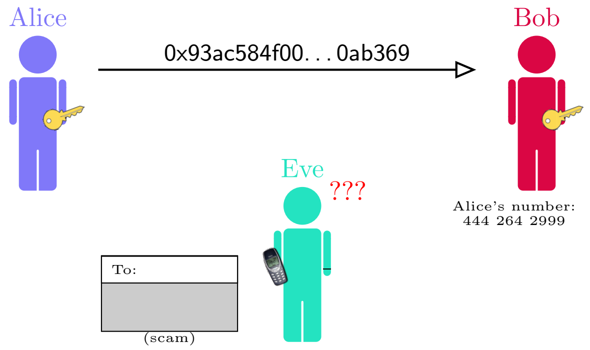


Alice
“My cell number is 444 264 2999”
Bob
Eve
To:4442642999
(scam)

Encryption

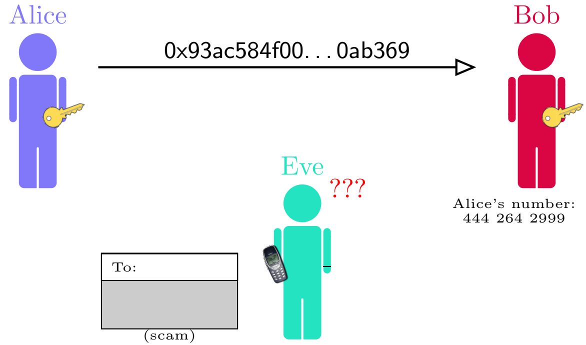


Encryption



Alice
0x93ac584f00. . . 0ab369
Bob
Eve
???
Alice's number:
444 264 2999
To:
(scam)

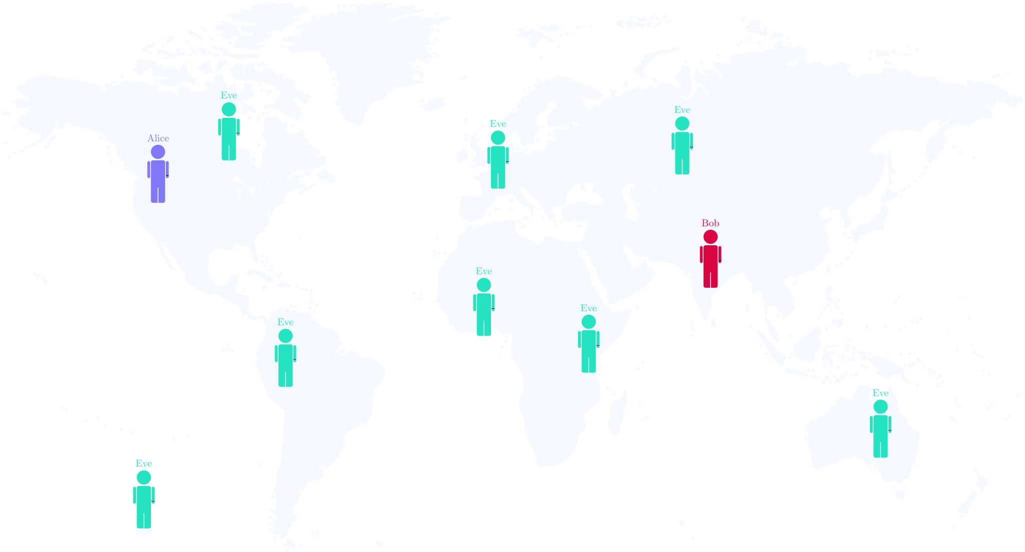
Encryption



But...

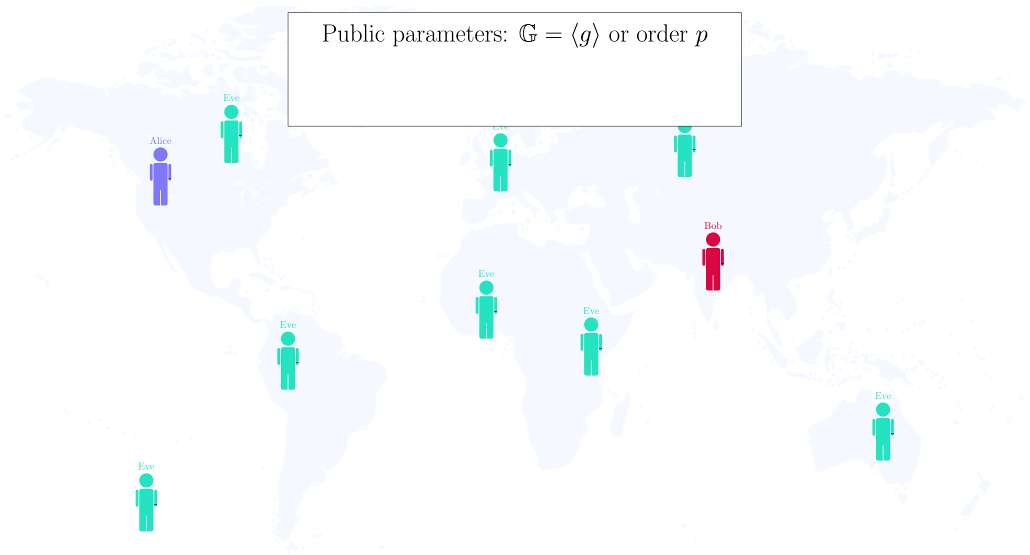
They need to share a secret key !

Key Exchange (Diffie-Hellman)

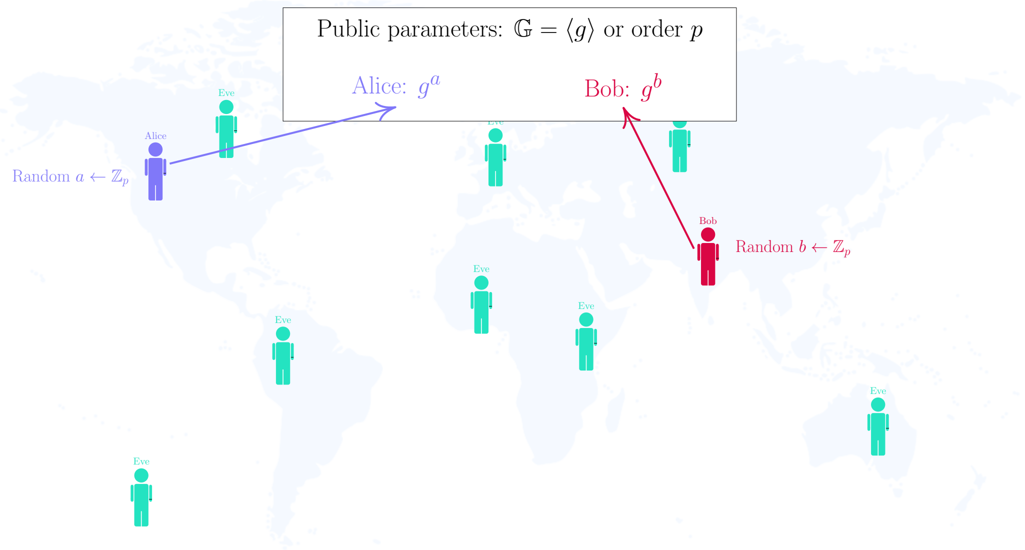


Key Exchange (Diffie-Hellman)

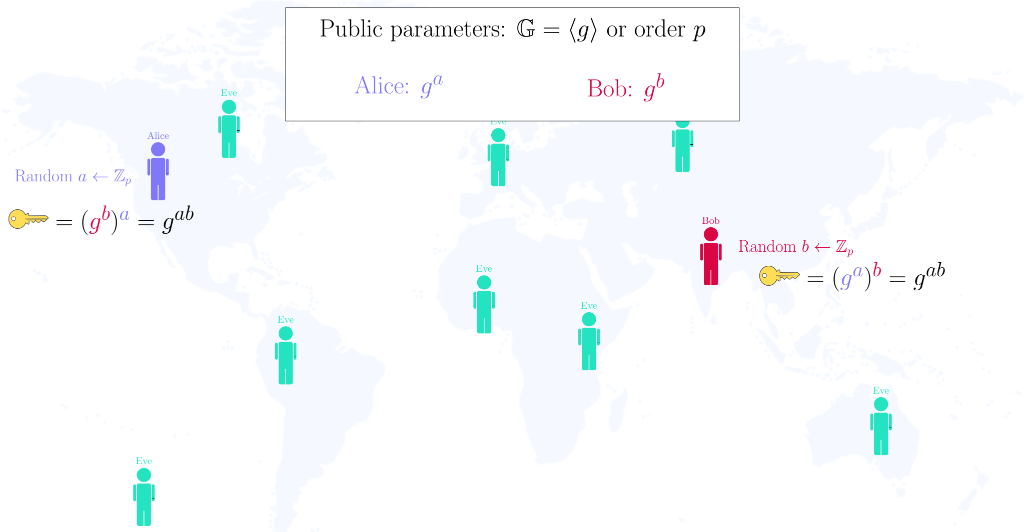
Public parameters: $\mathbb{G} = \langle g \rangle$ or order p



Key Exchange (Diffie-Hellman)



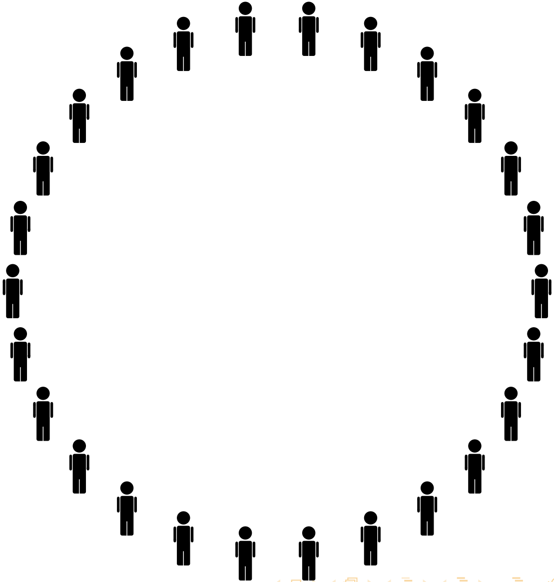
Key Exchange (Diffie-Hellman)



Contribution #1: [CLT-C13]

- New construction of MULTILINEAR MAPS
  - Extension of Bilinear Maps

- First implementations of:
  - Multilinear Maps
  - A 26-parties one-round key exchange



Contribution #1: [CLT-C13]

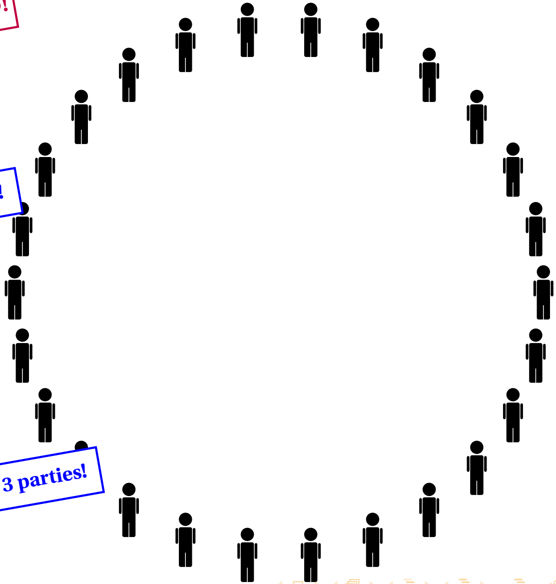
Only one other construction ☺!

- New construction of MULTILINEAR MAPS
  - Extension of Bilinear Maps

Lots of exciting applications!!

- First implementations of:
  - Multilinear Maps
  - A 26-parties one-round key exchange

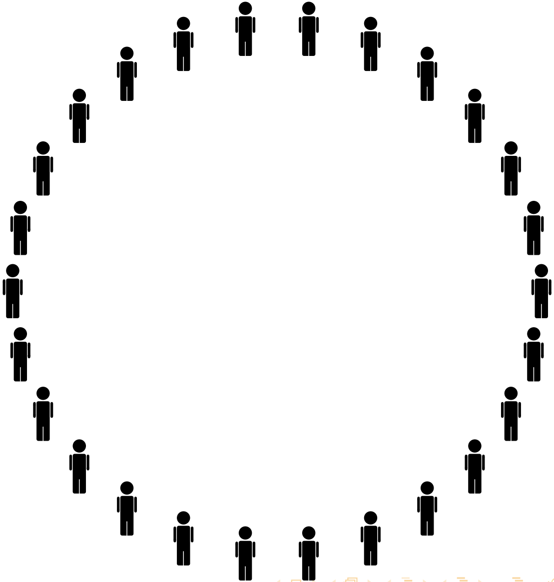
Only implemented for 2 and 3 parties!



Contribution #1: [CLT-C13]

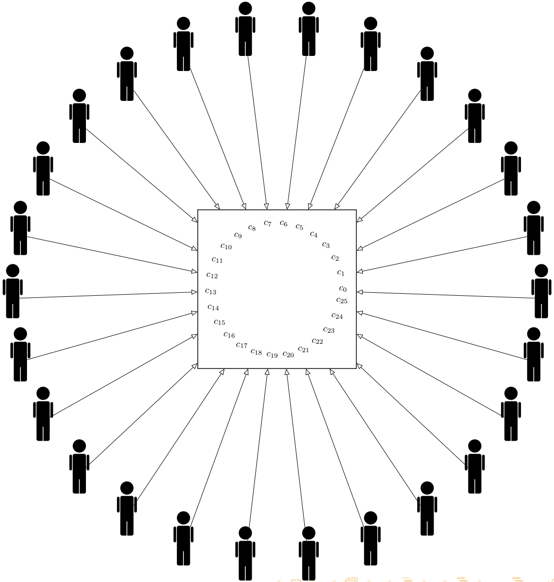
- New construction of MULTILINEAR MAPS
  - Extension of Bilinear Maps

- First implementations of:
  - Multilinear Maps
  - A 26-parties one-round key exchange



Contribution #1: [CLT-C13]

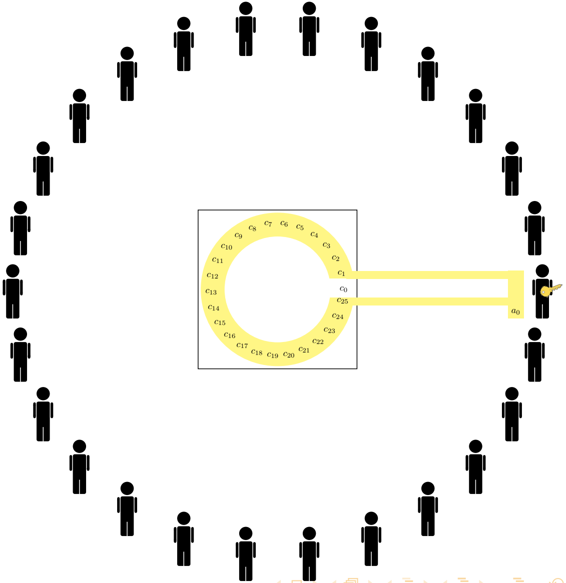
- New construction of MULTILINEAR MAPS
  - Extension of Bilinear Maps
- First implementations of:
  - Multilinear Maps
  - A 26-parties one-round key exchange



Contribution #1: [CLT-C13]

- New construction of MULTILINEAR MAPS
  - Extension of Bilinear Maps

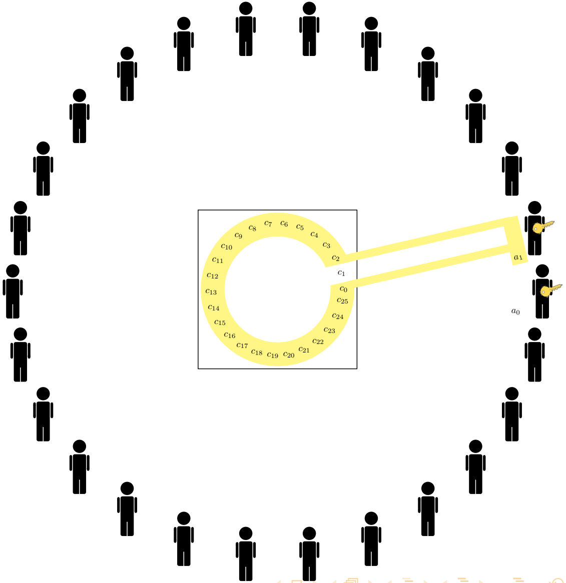
- First implementations of:
  - Multilinear Maps
  - A 26-parties one-round key exchange



Contribution #1: [CLT-C13]

- New construction of MULTILINEAR MAPS
  - Extension of Bilinear Maps

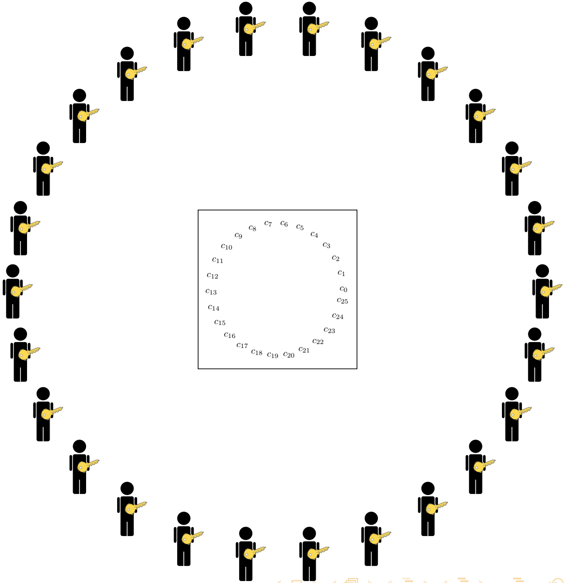
- First implementations of:
  - Multilinear Maps
  - A 26-parties one-round key exchange



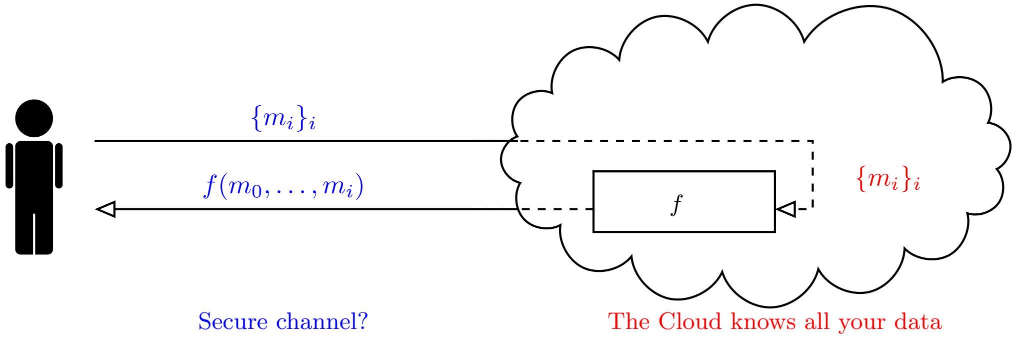
Contribution #1: [CLT-C13]

- New construction of MULTILINEAR MAPS
  - Extension of Bilinear Maps

- First implementations of:
  - Multilinear Maps
  - A 26-parties one-round key exchange

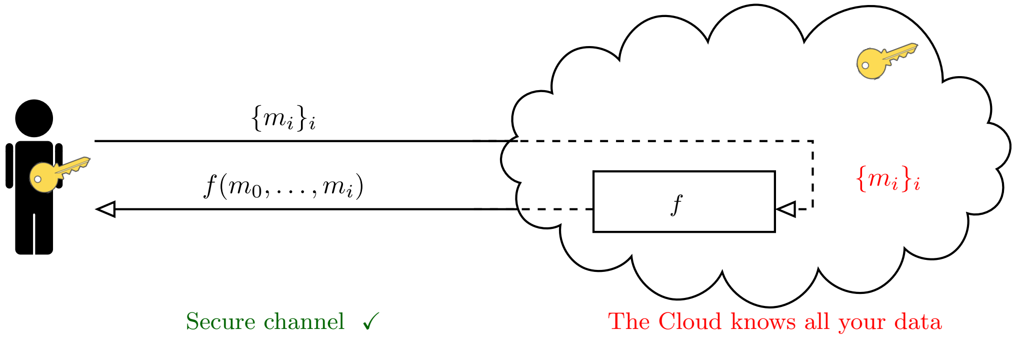


Confidentiality of Your Data



- We assume communication with the Cloud is secure ✓

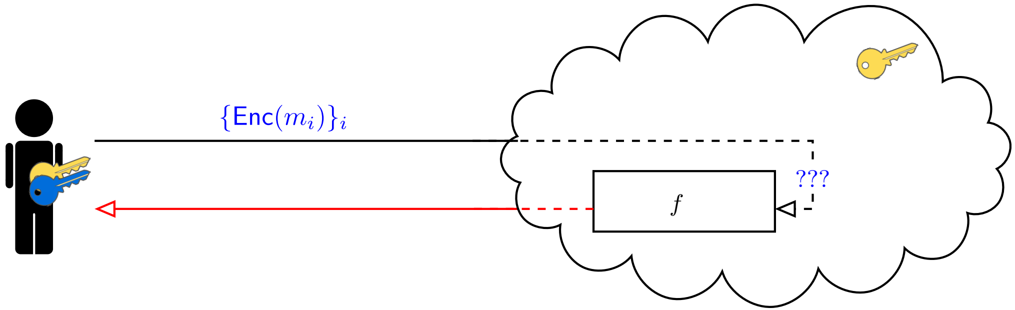
Confidentiality of Your Data



- We assume communication with the Cloud is secure ✓

This is the current situation

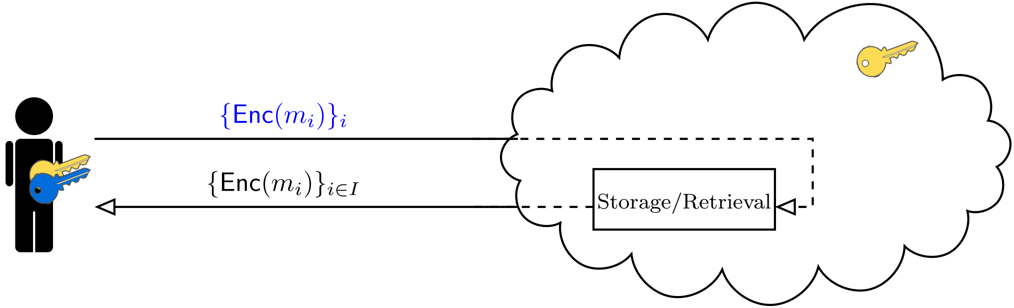
Confidentiality w.r.t. The Cloud



The Cloud knows nothing about your data

- For confidentiality, we use encryption

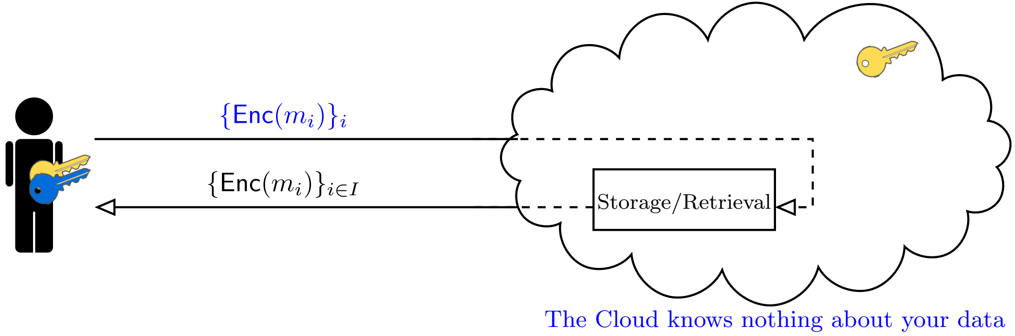
Confidentiality w.r.t. The Cloud



The Cloud knows nothing about your data

- For confidentiality, we use encryption
  - Now... limited to storage/retrieval

Confidentiality w.r.t. The Cloud



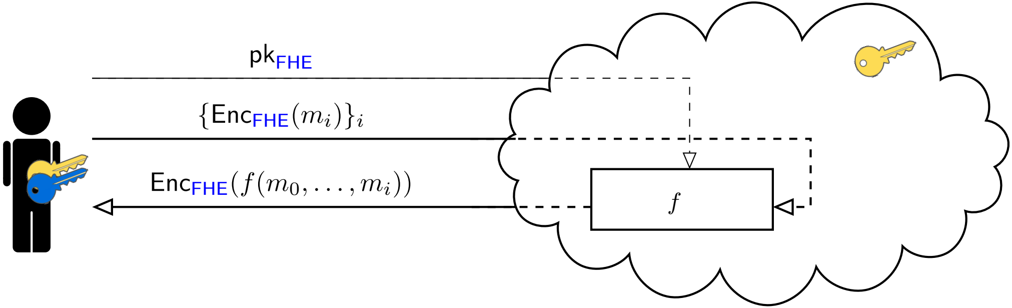
- For confidentiality, we use encryption
  - Now... limited to storage/retrieval
  - This is not even what Dropbox/Google Drive/Microsoft OneDrive/Amazon S2/iCloud Drive/etc. are doing
    - Allow access control and sharing, interaction with whole app universe, etc.

Fully Homomorphic Encryption

[RivestAdlemanDertouzos78]

Going beyond the storage/retrieval of encrypted data by permitting encrypted data to be operated on for interesting operations, in a public fashion?

- Enable **unlimited computation on encrypted data**
  (w.l.o.g. m_i's are bits and f Boolean circuit)



(*public* homomorphic computations)

Contribution #2

- Theoretical improvements of the DGHV scheme
  - Packing several plaintexts in one ciphertext [CCKLLTY-EC13]
  - Adaptation of a technique to manage noise growth [CLT-PKC14]
    - Exponential improvement!

- Fine analysis of the constraints to select concrete parameters

- Implementations of the schemes and benchmark on $f = \mathrm{AES}$

Outline

1. Introduction

2. Fully Homomorphic Encryption

3. Cryptographic Multilinear Maps

4. Conclusion

DGHV Scheme [vDGHV10]

- Public error-free element: $x_0 = q_0 \cdot p$
- Secret key $\mathsf{sk} = p$

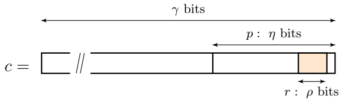
DGHV Scheme [vDGHV10]

- Public error-free element: $x_0 = q_0 \cdot p$
- Secret key $\mathsf{sk} = p$

- Ciphertext for $m \in \{0, 1\}$:

$$\boxed{c = q \cdot p + 2 \cdot r + m}$$

where q large random, r small random



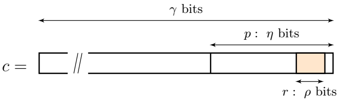
DGHV Scheme [vDGHV10]

- Public error-free element: $x_0 = q_0 \cdot p$
- Secret key $\mathsf{sk} = p$

- Ciphertext for $m \in \{0, 1\}$:

$$\boxed{c = q \cdot p + 2 \cdot r + m}$$

where q large random, r small random



- Decryption of c:

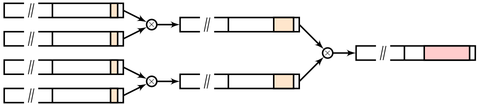
$$\boxed{m = (c \bmod p) \bmod 2}$$

Homomorphic Properties

- How to Add and Multiply Encrypted Bits:
  - Add/Mult two near-multiples of p gives a near-multiple of p
  - $c_1 = q_1 \cdot p + 2 \cdot r_1 + m_1, \qquad c_2 = q_2 \cdot p + 2 \cdot r_2 + m_2$
  - $c_1 + c_2 = p \cdot (q_1 + q_2) + \underbrace{2 \cdot (r_1 + r_2) + m_1 + m_2}_{\bmod 2 \rightarrow m_1 \text{XOR} m_2}$
  - $c_1 \cdot c_2 = p \cdot (c_2 q_1 + c_1 q_2 - q_1 q_2) + \underbrace{2 \cdot (2 r_1 r_2 + r_2 m_1 + r_1 m_2) + m_1 \cdot m_2}_{\bmod 2 \rightarrow m_1 \text{AND} m_2}$

Homomorphic Properties

- How to Add and Multiply Encrypted Bits:
  - Add/Mult two near-multiples of p gives a near-multiple of p
  - $c_1 = q_1 \cdot p + 2 \cdot r_1 + m_1, \qquad c_2 = q_2 \cdot p + 2 \cdot r_2 + m_2$
  - $c_1 + c_2 = p \cdot (q_1 + q_2) + \underbrace{2 \cdot (r_1 + r_2) + m_1 + m_2}_{\bmod 2 \rightarrow m_1 \,\mathsf{XOR}\, m_2}$
  - $c_1 \cdot c_2 = p \cdot (c_2 q_1 + c_1 q_2 - q_1 q_2) + \underbrace{2 \cdot (2 r_1 r_2 + r_2 m_1 + r_1 m_2) + m_1 \cdot m_2}_{\bmod 2 \rightarrow m_1 \,\mathsf{AND}\, m_2}$



Correctness for multiplicative depth of L: $\log_2 p = \eta \approx 2^L \cdot (\rho + 1)$

Our Contributions

1. New problem: Decisional Approximate-GCD problem [CCKLLTY-EC13]
   - Proved equivalent to the computational AGCD problem of [vDGHV10] in [CLT-PKC14]
   - Proofs are simpler!

Our Contributions

1. New problem: Decisional Approximate-GCD problem [CCKLLTY-EC13]
   - Proved equivalent to the computational AGCD problem of [vDGHV10] in [CLT-PKC14]
   - Proofs are simpler!
2. Batching: encrypt **vectors of bits** instead of single bits [CCKLLTY-EC13]
   - Reduce asymptotic overhead per gate
   - Useful for parallelization

Our Contributions

1. New problem: Decisional Approximate-GCD problem [CCKLLTY-EC13]
    - Proved equivalent to the computational AGCD problem of [vDGHV10] in [CLT-PKC14]
    - Proofs are simpler!
2. Batching: encrypt **vectors of bits** instead of single bits [CCKLLTY-EC13]
    - Reduce asymptotic overhead per gate
    - Useful for parallelization
3. Management of the noise growth
    - Heuristic method modeling noise growth [LP13]
    - Exponential improvement with scale-invariance technique [CLT-PKC14]

Our Contributions

1. New problem: Decisional Approximate-GCD problem [CCKLLTY-EC13]
   - Proved equivalent to the computational AGCD problem of [vDGHV10] in [CLT-PKC14]
   - Proofs are simpler!
2. Batching: encrypt **vectors of bits** instead of single bits [CCKLLTY-EC13]
   - Reduce asymptotic overhead per gate
   - Useful for parallelization
3. Management of the noise growth
   - Heuristic method modeling noise growth [LP13]
   - Exponential improvement with scale-invariance technique [CLT-PKC14]
4. Implementations
   - Benchmark on AES circuit [CCKLLTY-EC13,CLT-PKC14]

Semantic Security of the Scheme

Consider

$$D = \{q \cdot p + r : q \leftarrow [0, q_0), r \leftarrow [0, 2^\rho)\}$$

Security of the scheme based on:

(Error-Free) Decisional Approximate-GCD

Given $x_0 = q_0 \cdot p$ and polynomially many $x_i \in D$, decide whether z is uniformly generated in $[0, x_0)$ or in D

Semantic Security of the Scheme

Consider

$$D = \{q \cdot p + r : q \leftarrow [0, q_0), r \leftarrow [0, 2^{\rho})\}$$

Security of the scheme based on:

(Error-Free) Decisional Approximate-GCD

Given $x_0 = q_0 \cdot p$ and polynomially many $x_i \in D$, decide whether z is uniformly generated in $[0, x_0)$ or in D

Semantic security of the scheme:

- Recall that $c = q \cdot p + 2r + m$
  - Since $\gcd(2, q_0) = 1$, $c = 2 \cdot \Big(\underbrace{(q/2 \bmod q_0) \cdot p + r}_{\text{indistinguishable from uniform mod } x_0} \Big) + m \bmod (q_0 \cdot p)$

Semantic Security of the Scheme

Consider

$$D = \{q \cdot p + r : q \leftarrow [0, q_0), r \leftarrow [0, 2^\rho)\}$$

Security of the scheme based on:

(Error-Free) Decisional Approximate-GCD

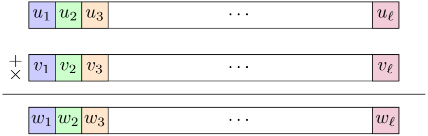
Given $x_0 = q_0 \cdot p$ and polynomially many $x_i \in D$, decide whether z is uniformly generated in $[0, x_0)$ or in D

Semantic security of the scheme:

- Recall that $c = q \cdot p + 2r + m$
  - Since $\gcd(2, q_0) = 1$, $c = 2 \cdot \Big(\underbrace{(q/2 \bmod q_0) \cdot p + r}_{\text{indistinguishable from uniform mod } x_0} \Big) + m \bmod (q_0 \cdot p)$
- Therefore ciphertext of m indistinguishable from uniform

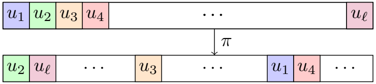
Batching (1)

- In one ciphertext, encode ℓ plaintexts
- Addition and Multiplication: in parallel over the ℓ slots



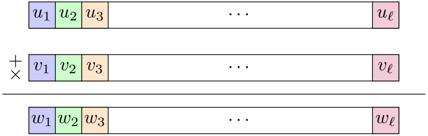
Batching (1)

- In one ciphertext, encode ℓ plaintexts
- Addition and Multiplication: in parallel over the ℓ slots
- Permutations between the slots (algebraic structure)



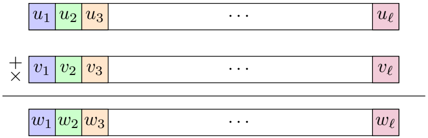
Batching (1)

- In one ciphertext, encode ℓ plaintexts
- Addition and Multiplication: in parallel over the ℓ slots
- Permutations between the slots (algebraic structure)



Batching (1)

- In <u>one</u> ciphertext, encode <u>ℓ</u> plaintexts
- Addition and Multiplication: <u>in parallel</u> over the ℓ slots
- Permutations between the slots (algebraic structure)

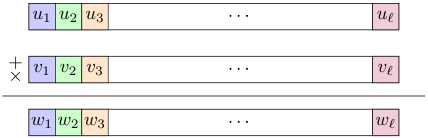


- Public element $x_0 = q_0 \cdot p$
- Ciphertext of $m \in \{0,1\}$:

$$c = q \cdot p + 2r + m$$

Batching (1)

- In one ciphertext, encode ℓ plaintexts
- Addition and Multiplication: in parallel over the ℓ slots
- Permutations between the slots (algebraic structure)



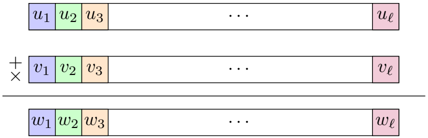
- Public element $x_0 = q_0 \cdot p$
- Ciphertext of $m \in \{0,1\}$:

$$c = q \cdot p + 2r + m$$

- $c \bmod p = 2r + m$; $c \bmod q_0 = \underbrace{q}_{\text{uniform in } [0, q_0)} \cdot p + 2r + m \bmod q_0$

Batching (1)

- In one ciphertext, encode ℓ plaintexts
- Addition and Multiplication: in parallel over the ℓ slots
- Permutations between the slots (algebraic structure)



- Public element $x_0 = q_0 \cdot p$
- Ciphertext of $m \in \{0, 1\}$:

$$c = q \cdot p + 2r + m$$

  - $c \bmod p = 2r + m$; $c \bmod q_0 = \underbrace{q}_{\text{uniform in } [0, q_0)} \cdot p + 2r + m \bmod q_0$

- We can write

$$c = \mathsf{CRT}_{q_0, p}\left(q', 2r + m\right)$$

Batching (2): Extend the Chinese Remainder Theorem

$$c = \mathsf{CRT}_{q_0, p}\left(\cancel{q},\ 2r + m\right)$$

- Generalization to several slots is easy!
- Ciphertext of $\vec{m} = (m_1, \ldots, m_\ell) \in \{0,1\}^\ell$:

$$c = \mathsf{CRT}_{q_0, p_1, \ldots, p_\ell}\left(\cancel{q},\ 2r_1 + m_1,\ \ldots,\ 2r_\ell + m_\ell\right)$$

Batching (2): Extend the Chinese Remainder Theorem

$$c = \mathsf{CRT}_{q_0,p}\left(q', \, 2r + m\right)$$

- Generalization to several slots is easy!
- Ciphertext of $\vec{m} = (m_1, \ldots, m_\ell) \in \{0,1\}^\ell$:

$$c = \mathsf{CRT}_{q_0,p_1,\ldots,p_\ell}\left(q', \, 2r_1 + m_1, \, \ldots, \, 2r_\ell + m_\ell\right)$$

- Decryption:

$$m_i = (c \bmod p_i) \bmod 2$$

Batching (2): Extend the Chinese Remainder Theorem

$$c = \mathsf{CRT}_{q_0, p}(q', 2r + m)$$

- Generalization to several slots is easy!
- Ciphertext of $\vec{m} = (m_1, \ldots, m_\ell) \in \{0,1\}^\ell$:

$$c = \mathsf{CRT}_{q_0, p_1, \ldots, p_\ell}(q', 2r_1 + m_1, \ldots, 2r_\ell + m_\ell)$$

- Decryption:

$$m_i = (c \bmod p_i) \bmod 2$$

- Thanks to the structure of the CRT:
  - **Addition**: the addition is performed modulo each p_i similarly to DGHV
  - **Multiplication**: the multiplication is performed modulo each p_i similarly to DGHV

Security of the Batch Scheme BDGHV

(Error-Free) Decisional Approximate-GCD

Given $x_0 = q_0 \cdot p$ and polynomially many $x_i \in D = \{q \cdot p + r : q \leftarrow [0, q_0), r \leftarrow [0, 2^\rho)\}$, decide whether z is uniformly generated in $[0, x_0)$ or in D

Security of the Batch Scheme BDGHV

(Error-Free) Decisional Approximate-GCD

Given $x_0 = q_0 \cdot p$ and polynomially many $x_i \in D = \{q \cdot p + r : q \leftarrow [0, q_0), r \leftarrow [0, 2^\rho)\}$, decide whether z is uniformly generated in $[0, x_0)$ or in D

Sketch:

(Error-Free) ℓ-Decisional Approximate-GCD

Given $x_0 = q_0 \cdot p_1 \cdots p_\ell$ and polynomially many $x_i \in D_\ell = \{\mathsf{CRT}_{q_0, p_i}(q, \ldots, r_i, \ldots) : q \leftarrow [0, q_0), r_i \leftarrow [0, 2^\rho)\}$, decide whether z is uniformly generated in $[0, x_0)$ or in D_ℓ

Security of the Batch Scheme BDGHV

(Error-Free) Decisional Approximate-GCD

Given $x_0 = q_0 \cdot p$ and polynomially many $x_i \in D = \{q \cdot p + r : q \leftarrow [0, q_0), r \leftarrow [0, 2^\rho)\}$, decide whether z is uniformly generated in $[0, x_0)$ or in D

Sketch:

(Error-Free) ℓ-Decisional Approximate-GCD

Given $x_0 = q_0 \cdot p_1 \cdots p_\ell$ and polynomially many $x_i \in D_\ell = \{\mathsf{CRT}_{q_0, p_i}(q, \ldots, r_i, \ldots) : q \leftarrow [0, q_0), r_i \leftarrow [0, 2^\rho)\}$, decide whether z is uniformly generated in $[0, x_0)$ or in D_ℓ

- For $\ell = 1$, the above problem is the (Error-Free) Decisional Approximate-GCD

Security of the Batch Scheme BDGHV

(Error-Free) Decisional Approximate-GCD

Given $x_0 = q_0 \cdot p$ and polynomially many $x_i \in D = \{q \cdot p + r : q \leftarrow [0, q_0), r \leftarrow [0, 2^\rho)\}$, decide whether z is uniformly generated in $[0, x_0)$ or in D

Sketch:

(Error-Free) ℓ-Decisional Approximate-GCD

Given $x_0 = q_0 \cdot p_1 \cdots p_\ell$ and polynomially many $x_i \in D_\ell = \{\mathsf{CRT}_{q_0, p_i}(q, \ldots, r_i, \ldots) : q \leftarrow [0, q_0), r_i \leftarrow [0, 2^\rho)\}$, decide whether z is uniformly generated in $[0, x_0)$ or in D_ℓ

- For $\ell = 1$, the above problem is the (Error-Free) Decisional Approximate-GCD
- Let A be an adversary having adv. ϵ to solve this latter problem

Security of the Batch Scheme BDGHV

(Error-Free) Decisional Approximate-GCD

Given $x_0 = q_0 \cdot p$ and polynomially many $x_i \in D = \{q \cdot p + r : q \leftarrow [0, q_0), r \leftarrow [0, 2^\rho)\}$, decide whether z is uniformly generated in $[0, x_0)$ or in D

Sketch:

(Error-Free) ℓ-Decisional Approximate-GCD

Given $x_0 = q_0 \cdot p_1 \cdots p_\ell$ and polynomially many $x_i \in D_\ell = \{\mathsf{CRT}_{q_0, p_i}(q, \ldots, r_i, \ldots) : q \leftarrow [0, q_0), r_i \leftarrow [0, 2^\rho)\}$, decide whether z is uniformly generated in $[0, x_0)$ or in D_ℓ

- For $\ell = 1$, the above problem is the (Error-Free) Decisional Approximate-GCD
- Let A be an adversary having adv. ϵ to solve this latter problem
- Denote D_i the distribution of elements of the form

$$\mathsf{CRT}_{q_0, p_1, \ldots, p_\ell}(q, \underbrace{*, \ldots, *}_{\ell - i \text{ random}}, r_i, \ldots, r_\ell)$$

Security of the Batch Scheme BDGHV

(Error-Free) Decisional Approximate-GCD

Given $x_0 = q_0 \cdot p$ and polynomially many $x_i \in D = \{q \cdot p + r : q \leftarrow [0, q_0), r \leftarrow [0, 2^\rho)\}$, decide whether z is uniformly generated in $[0, x_0)$ or in D

Sketch:

(Error-Free) ℓ-Decisional Approximate-GCD

Given $x_0 = q_0 \cdot p_1 \cdots p_\ell$ and polynomially many $x_i \in D_\ell = \{\mathsf{CRT}_{q_0, p_i}(q, \dots, r_i, \dots) : q \leftarrow [0, q_0), r_i \leftarrow [0, 2^\rho)\}$, decide whether z is uniformly generated in $[0, x_0)$ or in D_ℓ

- For $\ell = 1$, the above problem is the (Error-Free) Decisional Approximate-GCD
- Let A be an adversary having adv. ϵ to solve this latter problem
- Denote D_i the distribution of elements of the form

$$\mathsf{CRT}_{q_0, p_1, \dots, p_\ell}(q, \underbrace{*, \dots, *}_{\ell - i \text{ random}}, r_i, \dots, r_\ell)$$

- $\exists j_0$ s.t. A has advantage $\geq \epsilon/\ell$ to distinguish $D_{j_0 - 1}$ and D_{j_0}

Security of the Batch Scheme BDGHV

(Error-Free) Decisional Approximate-GCD

Given $x_0 = q_0 \cdot p$ and polynomially many $x_i \in D = \{q \cdot p + r : q \leftarrow [0, q_0), r \leftarrow [0, 2^\rho)\}$, decide whether z is uniformly generated in $[0, x_0)$ or in D

Sketch:

(Error-Free) ℓ-Decisional Approximate-GCD

Given $x_0 = q_0 \cdot p_1 \cdots p_\ell$ and polynomially many $x_i \in D_\ell = \{\mathsf{CRT}_{q_0, p_i}(q, \ldots, r_i, \ldots) : q \leftarrow [0, q_0), r_i \leftarrow [0, 2^\rho)\}$, decide whether z is uniformly generated in $[0, x_0)$ or in D_ℓ

- For $\ell = 1$, the above problem is the (Error-Free) Decisional Approximate-GCD
- Let A be an adversary having adv. ϵ to solve this latter problem
- Denote D_i the distribution of elements of the form

$$\mathsf{CRT}_{q_0, p_1, \ldots, p_\ell}(q, \underbrace{*, \ldots, *}_{\ell - i \text{ random}}, r_i, \ldots, r_\ell)$$

- $\exists j_0$ s.t. A has advantage $\geq \epsilon/\ell$ to distinguish $D_{j_0 - 1}$ and D_{j_0}
- With proba $1/\ell$, you can place p at the position j_0 (generate the $\ell - 1$ other p_i's yourself), and you use the challenge z for this slot

Security of the Batch Scheme BDGHV

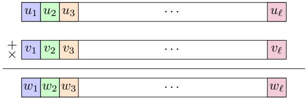
(Error-Free) Decisional Approximate-GCD

Given $x_0 = q_0 \cdot p$ and polynomially many $x_i \in D = \{q \cdot p + r : q \leftarrow [0, q_0), r \leftarrow [0, 2^\rho)\}$, decide whether z is uniformly generated in $[0, x_0)$ or in D

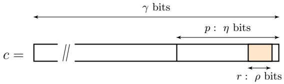
Security based on same problem as before!

Advantages of the Batch Variant

- Parallelization:



- Use the fact that $q \gg p$ to pack elements



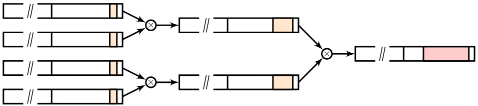
- (Also asymptotic reduction of overhead per gate with permutations)

[CCKLLTY13]

With **essentially same complexity costs** and **same security**, operations over $\ell \geq 1$ bits!

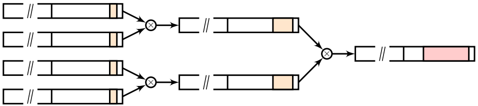
Mitigating Noise Growth: Scale-Invariance

- Even with batch variant, exponential growth of the noise

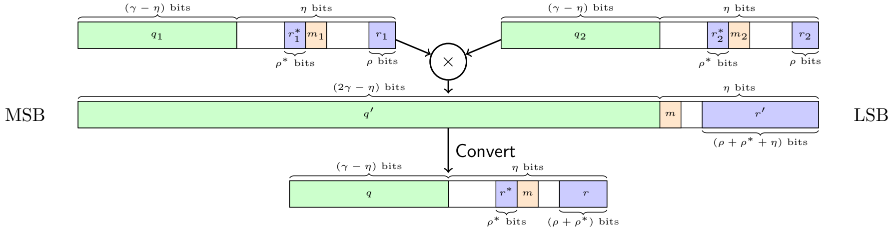


Mitigating Noise Growth: Scale-Invariance

- Even with batch variant, exponential growth of the noise



- New technique introduced by Brakerski: **scale-invariance**
  - Instead of encrypting in the LSB of $c \bmod p$, encrypt in the MSB
  - Adapted for DGHV [CLT-PKC14]



Contributions to Scale-Invariance

- Design of a new scheme based on Brakerski's idea
- Quantification of the noise growth:

Lemma (simplified) [CLT-PKC14]

Let c_1 and c_2 be ciphertexts of m_1 and m_2 with noises $\leq 2^{\rho}$. Then

$$c_3 = \mathsf{Convert}(c_1 \cdot c_2)$$

is a ciphertext of m_1 AND m_2 with noise $\leq 2^{\rho+\theta}$ for a fixed $\theta = \mathcal{O}(\log_2 \lambda)$

Contributions to Scale-Invariance

- Design of a new scheme based on Brakerski's idea
- Quantification of the noise growth:

Lemma (simplified) [CLT-PKC14]

Let c_1 and c_2 be ciphertexts of m_1 and m_2 with noises $\leq 2^\rho$. Then

$$c_3 = \mathsf{Convert}(c_1 \cdot c_2)$$

is a ciphertext of m_1 AND m_2 with noise $\leq 2^{\rho+\theta}$ for a fixed $\theta = \mathcal{O}(\log_2 \lambda)$

- Noise growth is **linear in multiplicative depth**
  - Correctness for multiplicative depth of L:

$$\log_2 p = \eta \approx \rho + \theta \cdot L$$

instead of $\approx 2^L \cdot \rho$ of the previous scheme

Exponential improvement!

Fully Homomorphic Encryption Scheme

- Only way to get fully homomorphic encryption: select parameters to evaluate decryption circuit

Bootstrapping

  - If $c = \mathsf{Enc}(m)$, run homomorphically Dec:

$$c_{\text{result}} = \mathsf{Enc}\Big(\mathsf{Dec}(c)\Big) = \mathsf{Enc}\Big(\mathsf{Dec}(\mathsf{Enc}(m))\Big) = \mathsf{Enc}\Big(m\Big)$$

  - select parameters s.t. one can do additional homomorphic operation(s)

Fully Homomorphic Encryption Scheme

- Only way to get fully homomorphic encryption: select parameters to evaluate decryption circuit

  Bootstrapping

  - If $c = \mathsf{Enc}(m)$, run homomorphically Dec:

$$c_{\text{result}} = \mathsf{Enc}\Big(\mathsf{Dec}(c)\Big) = \mathsf{Enc}\Big(\mathsf{Dec}(\mathsf{Enc}(m))\Big) = \mathsf{Enc}\Big(m\Big)$$

  - select parameters s.t. one can do additional homomorphic operation(s)

- Adaptation to batch scheme BDGHV in [CCKLLTY-EC13] and to scale-invariant scheme in [CLT-PKC14]

Fully Homomorphic Encryption Scheme

- Only way to get fully homomorphic encryption: select parameters to evaluate decryption circuit

Bootstrapping

  - If $c = \mathsf{Enc}(m)$, run homomorphically Dec:

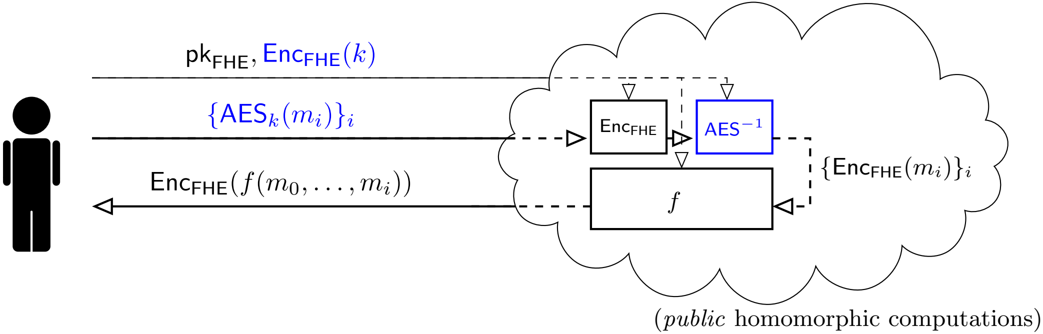
$$c_{\mathsf{result}} = \mathsf{Enc}\Big(\mathsf{Dec}(c)\Big) = \mathsf{Enc}\Big(\mathsf{Dec}(\mathsf{Enc}(m))\Big) = \mathsf{Enc}\Big(m\Big)$$

  - select parameters s.t. one can do additional homomorphic operation(s)

- Adaptation to batch scheme BDGHV in [CCKLLTY-EC13] and to **scale-invariant** scheme in [CLT-PKC14]
  - for scale-invariant scheme:
    linear noise growth $\Rightarrow$ bootstrapping **not** required for many levels

Implementations

- Benchmark on a nontrivial, not astronomical circuit: AES



Implementations

- Benchmark on a nontrivial, not astronomical circuit: AES
- Batch DGHV (with bootstrapping) [CCKLLTY-EC13]

λ	γ	ℓ	Mult	Bootstrapping	AES	Relative time
72	2.9MB	544	0.68 s	225 s	113 h	768 s
80	–	–	–	–	–	–

Implementations

- Benchmark on a nontrivial, not astronomical circuit: AES
- Batch DGHV (with bootstrapping) [CCKLLTY-EC13]

λ	γ	ℓ	Mult	Bootstrapping	AES	Relative time
72	2.9MB	544	0.68 s	225 s	113 h	768 s
80	–	–	–	–	–	–

- Scale-Invariant DGHV (without bootstrapping) [CLT-PKC14]

λ	γ	ℓ	Mult	Convert	AES	Relative time
72	2MB	569	0.1 s	33 s	3.6 h	23 s
80	4.5MB	1875	0.3 s	277 s	102 h	195 s

Implementations

- Benchmark on a nontrivial, not astronomical circuit: AES
- Batch DGHV (with bootstrapping) [CCKLLTY-EC13]

λ	γ	ℓ	Mult	Bootstrapping	AES	Relative time
72	2.9MB	544	0.68 s	225 s	113 h	768 s
80	–	–	–	–	–	–

- Scale-Invariant DGHV (without bootstrapping) [CLT-PKC14]

λ	γ	ℓ	Mult	Convert	AES	Relative time
72	2MB	569	0.1 s	33 s	3.6 h	23 s
80	4.5MB	1875	0.3 s	277 s	102 h	195 s

- Lattice-Based Scheme [GHS12]

λ	Ciphertext size	ℓ	AES	Relative time
80	0.3 MB	720	65 h	300 s

Future Work

- Assessment of advantages/disadvantages of existing schemes
- Optimizing cloud communications
- Prototypes of real-world applications?
- FHE outside “noisy” framework?

Outline

Starting Point: DDH and Bilinear Maps

- “The **DDH** assumption is a gold mine” (Boneh, 98)
  - Given (g^a, g^b, z) hard to decide if $z = g^{ab}$ or random
  - We “hide” values a_i’s in g^{a_i}
    - Easy to compute linear/affine functions + check if $a_i = 0$ (and constants)
    - Hard to compute/check quadratic functions

Starting Point: DDH and Bilinear Maps

- “The **DDH** assumption is a gold mine” (Boneh, 98)
  - Given (g^a, g^b, z) hard to decide if $z = g^{ab}$ or random
  - We “hide” values a_i’s in g^{a_i}
    - Easy to compute linear/affine functions + check if $a_i = 0$ (and constants)
    - Hard to compute/check quadratic functions
- Beyond DDH: **Bilinear Maps**
  - Give possibility to compute quadratic functions in the exponent
    - but computing cubic is hard…
  - Lots of new capabilities

Starting Point: DDH and Bilinear Maps

- "The **DDH** assumption is a gold mine" (Boneh, 98)
  - Given (g^a, g^b, z) hard to decide if $z = g^{ab}$ or random
  - We "hide" values a_i's in g^{a_i}
    - Easy to compute linear/affine functions + check if $a_i = 0$ (and constants)
    - Hard to compute/check quadratic functions
- Beyond DDH: **Bilinear Maps**
  - Give possibility to compute quadratic functions in the exponent
    - but computing cubic is hard...
  - Lots of new capabilities
- Can we do better **multilinear maps**?
  - i.e. give possibility to compute polynomials up to degree k in the exponents, but no more?
  - Considered by [BS03]: very fruitful, but unlikely to be constructed similarly to bilinear maps

MMaps vs. HE

- Wanted: add and multiply (bounded # times) encodings... ⇒ looks like HE

Multilinear Maps	**Homomorphic Encryption**
Encoding $e_a = g^a$	Encrypting $c_a = \mathsf{Enc}(a)$
Computing low-degree polynomials of the e_a's is easy	Computing low-degree polynomials of the c_a's is easy
Can test if encoding of 0	Cannot test anything... ...unless you know the secret key sk

MMaps vs. HE

- Wanted: add and multiply (bounded # times) encodings... $\Rightarrow$ looks like HE

Multilinear Maps	**Homomorphic Encryption**
Encoding $e_a = g^a$	Encrypting $c_a = \mathsf{Enc}(a)$
Computing low-degree polynomials of the e_a's is easy	Computing low-degree polynomials of the c_a's is easy
Can test if encoding of 0	Cannot test anything... ...unless you know the secret key sk

Can we modify the existing HE schemes to get MMaps?

MMaps vs. HE

- Wanted: add and multiply (bounded # times) encodings... ⇒ looks like HE

Multilinear Maps	**Homomorphic Encryption**
Encoding $e_a = g^a$	Encrypting $c_a = \mathsf{Enc}(a)$
Computing low-degree polynomials of the e_a's is easy	Computing low-degree polynomials of the c_a's is easy
Can test if encoding of 0	Cannot test anything... ...unless you know the secret key sk

Can we modify the existing HE schemes to get MMaps?

- First construction of approximate MMaps: Garg, Gentry, Halevi in 2013

Our Contributions [CLT-C13]

1. Start from (B)DGHV and transform it into approximate MMaps!
   - Only 1 other known construction of MMaps: the initial one
   - All $(\kappa + 1)$-degree functions seem hard
     - Some attacks in the original scheme have no equivalent here
2. Optimizations and (first!) implementation
   - Open-Source implementation of multilinear maps (Github)
   - Implementation of a 26-partite Diffie-Hellman Key Exchange

MMaps from DGHV?

Ciphertext of $m \in \{0, \ldots, g-1\}$ using DGHV:

$$c = \mathsf{CRT}_{q_0,p}(q,\ g \cdot r + m)$$

- **Problem**: q was used as a **mask** to hide everything
  - But we need a deterministic extraction procedure to construct protocols
    - seems hard to cancel a large random
  - If we remove it, no more encryption... $c = g \cdot r + m \in \mathbf{Z}$!

MMaps from DGHV?

Ciphertext of $m \in \{0, \ldots, g-1\}$ using DGHV:

$$c = \mathsf{CRT}_{q_0, p}(q,\ g \cdot r + m)$$

- **Problem**: q was used as a **mask** to hide everything
  - But we need a deterministic extraction procedure to construct protocols
    - seems hard to cancel a large random
  - If we remove it, no more encryption... $c = g \cdot r + m \in \mathbf{Z}$!
- Let us consider Batch DGHV instead!

From Batch DGHV to MMaps (1)

Ciphertext of $\vec{m} \in \{0, \ldots, g-1\}^\ell$ using BDGHV:

$$c = \mathsf{CRT}_{q_0, p_1, \ldots, p_\ell}(q,\ g \cdot r_1 + m_1,\ \ldots,\ g \cdot r_\ell + m_\ell)$$

- **Problem #1**: (Again) q was used as a **mask** to hide everything

From Batch DGHV to MMaps (1)

Ciphertext of $\vec{m} \in \{0,\ldots,g-1\}^\ell$ using BDGHV without mask:

$$c = \mathsf{CRT}_{p_1,\ldots,p_\ell}(g \cdot r_1 + m_1,\ \ldots,\ g \cdot r_\ell + m_\ell)$$

- **Problem #1**: (Again) q was used as a **mask** to hide everything
  - Let us remove it!
    - Seems less secure (does not rely on Approximate-GCD anymore)? How can we exploit that?

From Batch DGHV to MMaps (1)

Ciphertext of $\vec{m} \in \{0, \ldots, g-1\}^\ell$ using BDGHV without mask:

$$c = \mathsf{CRT}_{p_1,\ldots,p_\ell}(g \cdot r_1 + m_1, \ \ldots, \ g \cdot r_\ell + m_\ell)$$

- **Problem #1**: (Again) q was used as a **mask** to hide everything
  - Let us remove it!
    - Seems less secure (does not rely on Approximate-GCD anymore)? How can we exploit that?
- **Problem #2**: We don't know the p_i's, how can we sample?

From Batch DGHV to MMaps (1)

Encoding of a random $\vec{m} \in \{0, \ldots, g-1\}^\ell$:

$$c = \mathsf{CRT}_{p_1,\ldots,p_\ell}(g \cdot r_1 + m_1, \ \ldots, \ g \cdot r_\ell + m_\ell) = \sum_{i \in S} x_i$$

- **Problem #1**: (Again) q was used as a **mask** to hide everything
  - Let us remove it!
    - Seems less secure (does not rely on Approximate-GCD anymore)? How can we exploit that?
- **Problem #2**: We don't know the p_i's, how can we sample?
  - Define random encodings x_i's, and compute a subset sum of them
    - We don't know anymore what is the value of $\vec{m}$, but we don't often need it in protocols

From Batch DGHV to MMaps (1)

Encoding of a random $\vec{m} \in \{0, \ldots, g-1\}^\ell$:

$$c = \mathsf{CRT}_{p_1,\ldots,p_\ell}(g \cdot r_1 + m_1, \ \ldots, \ g \cdot r_\ell + m_\ell) = \sum_{i \in S} x_i$$

- **Problem #1**: (Again) q was used as a **mask** to hide everything
  - Let us remove it!
    - Seems less secure (does not rely on Approximate-GCD anymore)? How can we exploit that?
- **Problem #2**: We don't know the p_i's, how can we sample?
  - Define random encodings x_i's, and compute a subset sum of them
    - We don't know anymore what is the value of $\vec{m}$, but we don't often need it in protocols
- **Problem #3**: Fuzzy threshold for easy vs. hard?
  - Because we don't know exactly how the noise increases

From Batch DGHV to MMaps (1)

Encoding of a random $\vec{m} \in \{0, \ldots, g-1\}^\ell$:

$$c = \frac{\mathsf{CRT}_{p_1,\ldots,p_\ell}(g \cdot r_1 + m_1, \ \ldots, \ g \cdot r_\ell + m_\ell)}{z} = \sum_{i \in S} x'_i$$

- **Problem #1**: (Again) q was used as a **mask** to hide everything
  - Let us remove it!
    - Seems less secure (does not rely on Approximate-GCD anymore)? How can we exploit that?
- **Problem #2**: We don't know the p_i's, how can we sample?
  - Define random encodings x_i's, and compute a subset sum of them
    - We don't know anymore what is the value of $\vec{m}$, but we don't often need it in protocols
- **Problem #3**: Fuzzy threshold for easy vs. hard?
  - Because we don't know exactly how the noise increases
  - Use a secret mask z with $x'_i = x_i / z$!

From Batch DGHV to MMaps (2)

$$c = \frac{\mathsf{CRT}_{p_1,\ldots,p_\ell}(g \cdot r_1 + m_1, \ldots, g \cdot r_\ell + m_\ell)}{z} = \sum_{i \in S} x'_i$$

- **Multiplication** of encodings with masks z^i (i.e. level-i) and z^j (i.e. level-j) $\Rightarrow$ encoding with mask z^{i+j} (i.e. level-$(i+j)$)

From Batch DGHV to MMaps (2)

$$c = \frac{\mathsf{CRT}_{p_1,\ldots,p_\ell}(g \cdot r_1 + m_1, \ldots, g \cdot r_\ell + m_\ell)}{z} = \sum_{i \in S} x'_i$$

- **Multiplication** of encodings with masks z^i (i.e. level-i) and z^j (i.e. level-j) $\Rightarrow$ encoding with mask z^{i+j} (i.e. level-$(i+j)$)
- **Zero-test procedure**: does a level-κ encoding encodes $\vec{0}$?

From Batch DGHV to MMaps (2)

$$c = \frac{\mathsf{CRT}_{p_1,\ldots,p_\ell}(g \cdot r_1 + m_1, \ldots, g \cdot r_\ell + m_\ell)}{z} = \sum_{i \in S} x'_i$$

- **Multiplication** of encodings with masks z^i (i.e. level-i) and z^j (i.e. level-j) $\Rightarrow$ encoding with mask z^{i+j} (i.e. level-$(i+j)$)
- **Zero-test procedure**: does a level-κ encoding encodes $\vec{0}$?
  - Need to cancel z^κ but cannot reveal z!
  - Define

$$p_{zt} = \sum_{i=1}^{\ell} h_i \cdot (z^\kappa \cdot g^{-1} \bmod p_i) \cdot \prod_{j \neq i} p_j$$

From Batch DGHV to MMaps (2)

$$c = \frac{\mathsf{CRT}_{p_1,\ldots,p_\ell}(g\cdot r_1 + m_1,\ \ldots,\ g\cdot r_\ell + m_\ell)}{z} = \sum_{i\in S} x'_i$$

- **Multiplication** of encodings with masks z^i (i.e. level-i) and z^j (i.e. level-j) $\Rightarrow$ encoding with mask z^{i+j} (i.e. level-$(i+j)$)
- **Zero-test procedure**: does a level-κ encoding encodes $\vec{0}$?
  - Need to cancel z^κ but cannot reveal z!
  - Define

$$p_{zt} = \sum_{i=1}^{\ell} h_i \cdot (z^\kappa \cdot g^{-1} \bmod p_i) \cdot \prod_{j\neq i} p_j$$

  - Compute $\omega = c \cdot p_{zt} \bmod x_0$

$$\mathsf{isZero}(\omega) = \begin{cases} 1 & \text{if } \omega \ll x_0 \\ 0 & \text{otherwise} \end{cases}$$

Zero Test

$$c = \frac{\mathsf{CRT}_{p_1,\ldots,p_\ell}(g \cdot r_1 + m_1, \ldots, g \cdot r_\ell + m_\ell)}{z} = \sum_{i \in S} x'_i$$

and

$$p_{zt} = \sum_{i=1}^{\ell} h_i \cdot (z^\kappa \cdot g^{-1} \bmod p_i) \cdot \prod_{j \neq i} p_j$$

- If c encodes $\vec{0}$, we have

$$c \cdot p_{zt} \bmod x_0 = \sum_{i=1}^{\ell} h_i r_i \cdot \prod_{j \neq i} p_j \ll x_0 = \prod_{i=1,\ldots,n} p_i$$

- If c encodes $\vec{m} \neq \vec{0}$, we have

$$c \cdot p_{zt} \bmod x_0 = \sum_{i=1}^{\ell} h_i (r_i + m_i \cdot g^{-1} \bmod p_i) \cdot \prod_{j \neq i} p_j \approx x_0$$

Zero Test

$$c = \frac{\mathsf{CRT}_{p_1,\ldots,p_\ell}(g_1 \cdot r_1 + m_1, \ \ldots, \ g_\ell \cdot r_\ell + m_\ell)}{z} = \sum_{i \in S} x_i'$$

and

$$p_{zt} = \sum_{i=1}^{\ell} h_i \cdot (z^\kappa \cdot g_i^{-1} \bmod p_i) \cdot \prod_{j \neq i} p_j$$

- If c encodes $\vec{0}$, we have

$$c \cdot p_{zt} \bmod x_0 = \sum_{i=1}^{\ell} h_i r_i \cdot \prod_{j \neq i} p_j \ll x_0 = \prod_{i=1,\ldots,n} p_i$$

- If c encodes $\vec{m} \neq \vec{0}$, we have

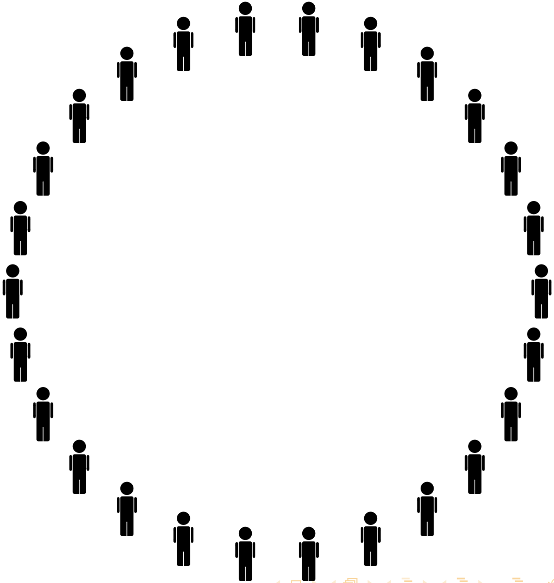
Actually we need distinct g_i's to avoid another attack

$$c \cdot p_{zt} \bmod x_0 = \sum_{i=1}^{\ell} h_i (r_i + m_i \cdot g_i^{-1} \bmod p_i) \cdot \prod_{j \neq i} p_j \approx x_0$$

Implementation: 26-partite Key Exchange

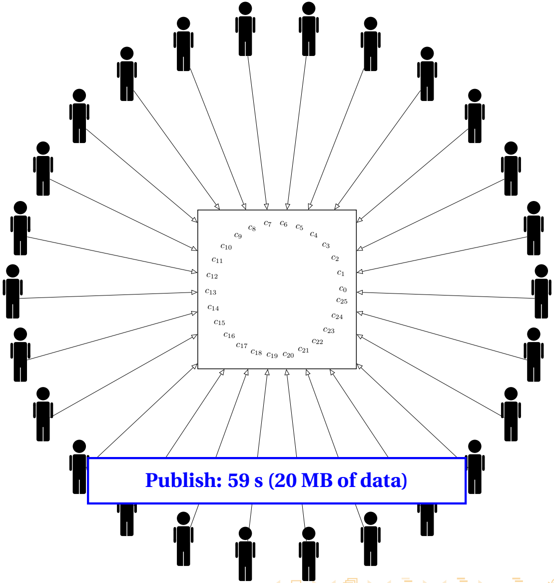
- Implementation of a 26-partite one-round Diffie-Hellman key exchange

- Public parameters of multilinear maps for $\kappa = 25$ levels



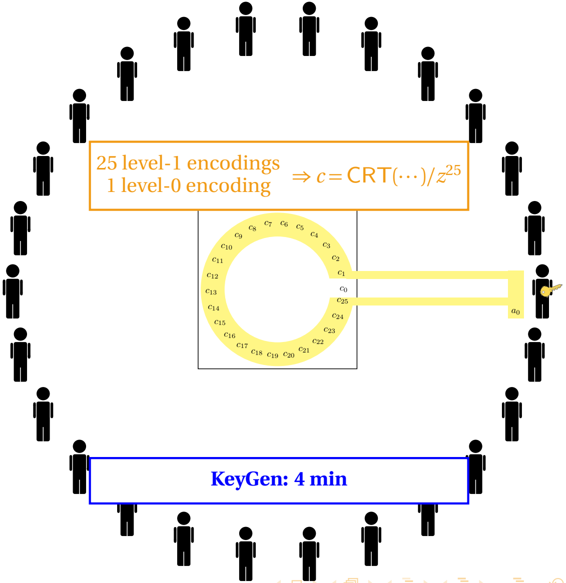
Implementation: 26-partite Key Exchange

- Implementation of a 26-partite one-round Diffie-Hellman key exchange
- Public parameters of multilinear maps for $\kappa = 25$ levels



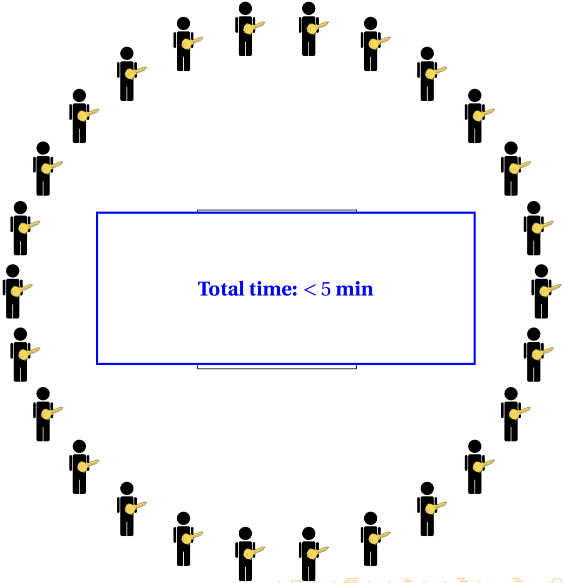
Implementation: 26-partite Key Exchange

- Implementation of a 26-partite one-round Diffie-Hellman key exchange
- Public parameters of multilinear maps for $\kappa = 25$ levels



Implementation: 26-partite Key Exchange

- Implementation of a 26-partite one-round Diffie-Hellman key exchange
- Public parameters of multilinear maps for $\kappa = 25$ levels



Future Work

- Explosion of multilinear maps in cryptography (and of obfuscation, built on multilinear maps)
- Improve the practicality of multilinear maps
  - akin to what has been done for FHE, and beyond
- Applications with reasonable number of multilinearity level
- Cryptanalysis to build confidence in the multilinear maps proposals

Outline

1. Introduction
2. Fully Homomorphic Encryption
3. Cryptographic Multilinear Maps
4. Conclusion

Contributions to Fully Homomorphic Encryption

On the Minimal Number of Bootstrappings in Homomorphic Circuits.
L., Paillier [WAHC 2013]

Batch Fully Homomorphic Encryption over the Integers.
Cheon, Coron, Kim, Lee, L., Tibouchi, Yun [EUROCRYPT 2013]

Scale-Invariant Fully Homomorphic Encryption over the Integers.
Coron, L., Tibouchi [PKC 2014]

A Comparison of the Homomorphic Encryption Schemes FV and YASHE.
L., Naehrig [AFRICACRYPT 2014]

Implementation: `https://github.com/tlepoint/homomorphic-simon`

Contributions to Multilinear Maps

Practical Multilinear Maps over the Integers.
Coron, L., Tibouchi [CRYPTO 2013]

Implementation: `https://github.com/tlepoint/multimap`

Other Areas

Most efficient existing lattice-based signature scheme!

- Lattice-Based Signature

  Lattice Signatures and Bimodal Gaussians.
  Ducas, Durmus, L., Lyubashevsky [CRYPTO 2013]

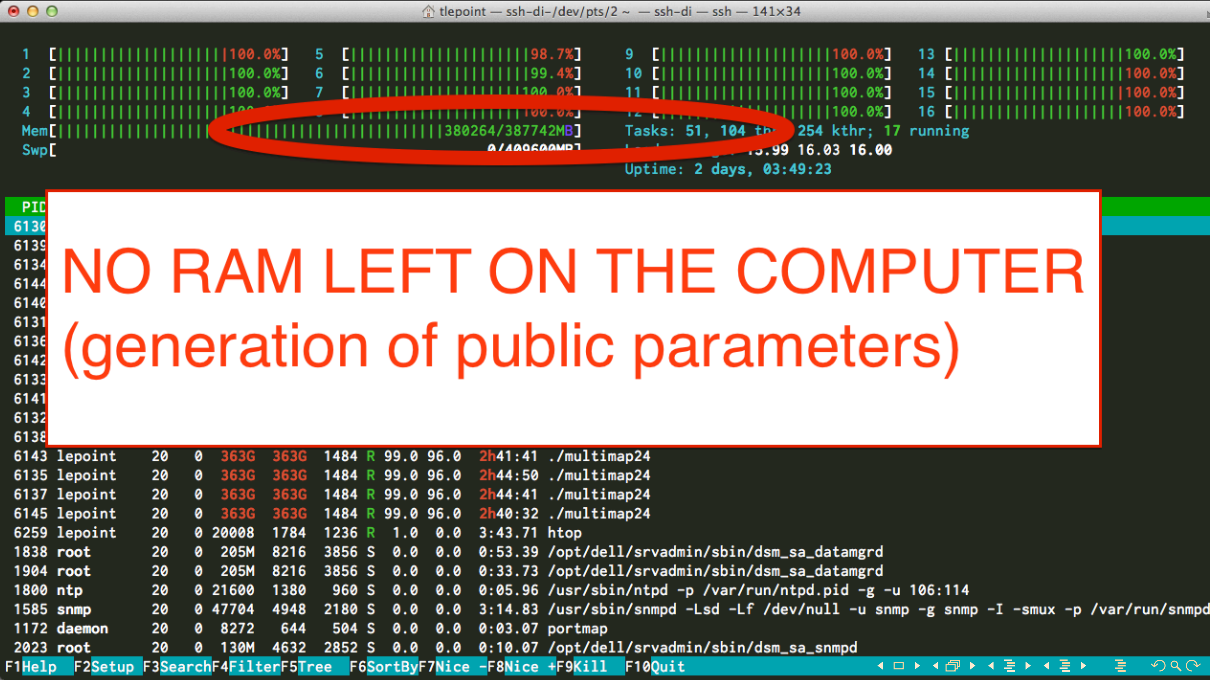
  Implementation: `http://bliss.di.ens.fr`

- White-Box Cryptography

  Two Attacks on a White-Box AES Implementation.
  L., Rivain, De Mulder, Roelse, Preneel [SAC 2013]

  White-Box Security Notions for Symmetric Encryption Schemes.
  Delerablée, L., Paillier, Rivain [SAC 2013]


THANK YOU


tlepoint — ssh-di-/dev/pts/2 ~ — ssh-di — ssh — 141×34
1 [|||||||||||||||||||100.0%]
2 [|||||||||||||||||||100.0%]
3 [|||||||||||||||||||100.0%]
5 [||||||||||||||||||||98.7%]
6 [||||||||||||||||||||99.4%]
9 [|||||||||||||||||||100.0%]
10 [|||||||||||||||||||100.0%]
11 [|||||||||||||||||||100.0%]
13 [|||||||||||||||||||100.0%]
14 [|||||||||||||||||||100.0%]
15 [|||||||||||||||||||100.0%]
16 [|||||||||||||||||||100.0%]
Mem[380264/387742MB]
254 kthr; 17 running
Uptime: 2 days, 03:49:23
NO RAM LEFT ON THE COMPUTER
(generation of public parameters)
6143 lepoint 20 0 363G 363G 1484 R 99.0 96.0 2h41:41 ./multimap24
6135 lepoint 20 0 363G 363G 1484 R 99.0 96.0 2h44:50 ./multimap24
6137 lepoint 20 0 363G 363G 1484 R 99.0 96.0 2h44:41 ./multimap24
6145 lepoint 20 0 363G 363G 1484 R 99.0 96.0 2h40:32 ./multimap24
6259 lepoint 20 0 20008 1784 1236 R 1.0 0.0 3:43.71 htop
1838 root 20 0 205M 8216 3856 S 0.0 0.0 0:53.39 /opt/dell/srvadmin/sbin/dsm_sa_datamgrd
1904 root 20 0 205M 8216 3856 S 0.0 0.0 0:33.73 /opt/dell/srvadmin/sbin/dsm_sa_datamgrd
1800 ntp 20 0 21600 1380 960 S 0.0 0.0 0:05.96 /usr/sbin/ntpd -p /var/run/ntpd.pid -g -u 106:114
1585 snmp 20 0 47704 4948 2180 S 0.0 0.0 3:14.83 /usr/sbin/snmpd -Lsd -Lf /dev/null -u snmp -g snmp -I -smux -p /var/run/snmpd
1172 daemon 20 0 8272 644 504 S 0.0 0.0 0:03.07 portmap
2023 root 20 0 130M 4632 2852 S 0.0 0.0 0:10.07 /opt/dell/srvadmin/sbin/dsm_sa_snmpd
F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice - F8Nice + F9Kill F10Quit