Computations on Encrypted Data and Privacy

11th International Conference on Provable Security
Xi'an, China - October 23rd, 2017

David Pointcheval
CNRS - ENS - INRIA

The Cloud

Dropbox
iCloud
Drive

Security Requirements

As from a local hard drive/server, one expects
- **Storage guarantees**
- **Privacy guarantees**
  - confidentiality of the data
  - anonymity of the users
  - obliviousness of the queries/processing

How to proceed?

Anything from Anywhere

One can store
- Documents to share
- Pictures to edit
- Databases to query

and access from everywhere

David Pointcheval

Introduction
Confidentiality vs Sharing & Computations

**Classical Encryption** allows to protect data
- The provider stores them without knowing them
- Nobody can access them either, except the owner/target receiver

**How to share the data?**
**How to compute on the data?**

Broadcast Encryption

The sender chooses a target set
Users get all-or-nothing about the data

Sharing to a Target Set but **No Computations!**

Fully Homomorphic Encryption

FHE allows any computations on encrypted data
But the result is encrypted as the inputs!

Computations But **No Controlled Sharing!**
Functional Encryption

The authority generates functional decryption keys $DK_f$ according to functions $f$ from $C = \text{Encrypt}(x)$, $\text{Decrypt}(DK_f, C)$ outputs $f(x)$. This allows controlled sharing of data.

Result in clear for a Specific Function for Specific Users

Functional Encryption is Powerful

Functional Encryption allows access control:
- with $f_{\text{id}}(x | y) = (\text{if } y = \text{id}, \text{ then } x, \text{ else } \perp)$: identity-based encryption
- with $f_G(x | y) = (\text{if } y \in G, \text{ then } x, \text{ else } \perp)$: broadcast encryption

Functional Encryption allows computations:
- any function $f$: in theory, with $iO$ (Indistinguishable Obfuscation)
- concrete functions: inner product

FE: Concrete Case

Cells of derived tables are linear combinations of the grades $\overrightarrow{a}_i$ of the grades $\overrightarrow{b}$ from the main table:

$$c_i = \sum_j a_{i,j} b_j = \overrightarrow{a}_i \cdot \overrightarrow{b}$$

$\overrightarrow{b}$: vector of the private grades, encrypted in the main table
$\overrightarrow{a}_i$: vector of the public coefficients for the cell $c_i$, defines $f_i$

With ElGamal encryption:
- computations modulo $p$
- if grades, coefficients, and classes small enough: DLog computation

FE: Inner Product

Cells of derived tables are linear combinations of the grades $\overrightarrow{a}_i$ of the grades $\overrightarrow{b}$ from the main table:

$$c_i = \sum_j a_{i,j} b_j = \overrightarrow{a}_i \cdot \overrightarrow{b}$$

$\overrightarrow{b}$: vector of the private grades, encrypted in the main table
$\overrightarrow{a}_i$: vector of the public coefficients for the cell $c_i$, defines $f_i$

With ElGamal encryption:
- computations modulo $p$
- if grades, coefficients, and classes small enough: DLog computation
FE: Limitations

Initial result: selective security
But improved to adaptive security
Anyway:
- one key limits to one function on any vector
- a malicious player could ask many functional keys
- too many keys might reveal the plaintexts
- a unique sender only can encrypt all the inputs
- Multi-Input Functional Encryption (MIFE)

IP-FE: Concrete Security?

IP-FE: from $c = E(x)$ and $dk_y$, for $n$-vectors $x$ and $y$, one gets $x \cdot y$
- $n$ different keys reveal $x$
- for the indistinguishability between two sets of vectors, the adversary is not allowed to ask keys that trivially tell them appart
  $\Rightarrow$ if $n$ vectors in the sets, the adversary cannot ask any key!

IP-FE: Too Many Messages/Keys?

IP-FE with Helper:
from $c = E(x)$ and $dk_y$, for $n$-vectors $x$ and $y$, one must ask an helper
- the helper
  - learns as few as possible about the input
    - (which ciphertext, which function, which user, etc)
  - limits the number of answers (according to a bound on the inputs)
  - learns nothing about the output
- whereas there are additional interactions
  - no much leakage of information to the helper
  - more reasonable security model

IP-MIFE: Concrete Security?

IP-MIFE: from $c_1 = E(x_1)$, $\ldots$, $c_n = E(x_n)$ and $dk_y$, one gets $x \cdot y$
- if no ordering: one immediately gets $n!$ linear relations on $x$
- even with ordering, $c_1 = E(1, x_1)$, $\ldots$, $c_n = E(n, x_n)$
  - if public encryption: only constant-functional keys allowed!
  - if private encryption: mix-and-match attacks
Multi-Client Functional Encryption

- In addition to the ordering, there is a label (or a time period)
  - Client \( C_i \) generates \( c_i = E(i, \lambda, x_i) \) for a label \( \lambda \)
  - \( \Rightarrow \) only one ciphertext for each index \( i \) and each label \( \lambda \)

- Multi-User Inputs
- Mix-and-match attacks avoided by private encryption
- More reasonable security model 😞
- But still a unique authority for the functional key generation 😞


Independent and Untrusted Clients

- Senders \((S_i)_i\) provide sensitive inputs \( x_i \) (e.g., financial data) in an encrypted way under secret encryption keys \( e_k \)
  - \( c_i = E(e_k, \lambda, x_i) \) for a label \( \lambda \) (or every time period)
- For some functions \( f \), an aggregator proposes, as a service, to communicate the aggregation \( f(x) \) for every label \( \lambda \), thanks to a functional decryption key \( d_k_f \)
- The senders want to keep control on \( f \)
  - \( d_k_f \) is generated by the senders

[Chotard-Dufour Sans-Phan-P. - EPrint 2017/989]

Decentralized MCFE

- Setup() \( \rightarrow \) secret key \( s_k_i \) and encryption key \( e_k_i \) for each sender \( S_i \) and \( m_p_k \), the master public key
- Encrypt(\( e_k, \lambda, x_i \)) \( \rightarrow c_i = E(e_k, \lambda, x_i) \) for the label \( \lambda \)
- DKeyGen((\( s_k_i \), \( f \)) \( \rightarrow d_k_f \)
- Decrypt(\( d_k_f, \lambda, C \)) \( \rightarrow f(x) \) if \( C = (c_i = E(e_k, \lambda, x_i)) \):

[Chotard-Dufour Sans-Phan-P. - EPrint 2017/989]

- Encrypt/Decrypt are non-interactive algorithms
- Setup/DKeyGen are interactive protocols between the senders
- DKeyGen should be a one-round protocol only
ElGamal Encryption

ElGamal Encryption on $\mathbb{G} = \langle g \rangle$:
- Secret key: $s \in \mathbb{Z}_p$
- Public key: $h = g^s$
- Encryption: $c = (c_0 = g^r, c_1 = h^r \cdot m)$
- Decryption: $m = c_1/c_0^s$
- Semantically secure under DDH in $\mathbb{G} = \langle g \rangle$
- Multiplicatively homomorphic
- Additive variant: $m$ is replaced by $g^m$ but requires discrete logarithm computation
- Encryption of vectors:
  - with many $h_i$ and the same randomness

FE: IP with ElGamal

Parameters:
- a group $\mathbb{G} = \langle g \rangle$ of prime order $p$
- Secret key: $\tilde{s} = (s_i)_i$ for random scalars in $\mathbb{Z}_p$
- Public key: $\overline{h} = (h_j = g^{s_j})_j$
- Encryption:
  - $c = g^r$ and $\overline{C} = (C_j = h_j^r \cdot g^{r_j})_j$
  - $D = \overline{f} \cdot \overline{C} = \prod_j C_j^f_j = g^{\sum_j f_j s_j \cdot 1} = g^{\sum_j f_j \tilde{s}} g^{\overline{f} \cdot \overline{x}}$
- Functional key: $dk_f = \sum_j f_j s_j = \overline{f} \cdot \overline{s}$
- Decryption:
  - $D = e^{dk_f} \cdot g^m \rightarrow m = \log_g (\overline{f} \cdot \overline{C} / e^{dk_f}) = \overline{f} \cdot \overline{x}$

Because of the common $r$ in the ciphertext, a unique sender must encrypt the full vector

MCFE: IP with ElGamal

Parameters:
- $\mathbb{G} = \langle g \rangle$ of prime order $p$, hash function $\mathcal{H}$
- Encryption/Secret key: $e k_i = s k_i = s_i$, for random scalar in $\mathbb{Z}_p$
- Encryption:
  - $C_i = \mathcal{H}(\lambda)^{s_i} \cdot g^{r_i}$
  - $D = \overline{f} \cdot \overline{C} = \prod_i C_i^{f_i} = \mathcal{H}(\lambda)^{\sum_i f_i s_i} g^{\sum_i f_i r_i} = \mathcal{H}(\lambda)^{\overline{f} \cdot \overline{s}} g^{\overline{f} \cdot \overline{x}}$
- Functional key: $dk_f = \sum_i f_i s_i = \overline{f} \cdot \overline{s}$
- Decryption:
  - $D = \mathcal{H}(\lambda)^{dk_f} \cdot g^m \rightarrow m = \log_g (\overline{f} \cdot \overline{C} / \mathcal{H}(\lambda)^{dk_f}) = \overline{f} \cdot \overline{x}$

Encryption can be performed by independent senders

DMCFE: IP with ElGamal

Functional key: $dk_f = \sum_i f_i s_i = \overline{f} \cdot \overline{s}$ where $\overline{X} = (X_i = f_i s_i)_i$

- The senders can encrypt $(X_i = f_i s_i)$ under another IP-MCFE and the label $f$
- The aggregator knows the functional key for $(1,\ldots,1)$
- From the ciphertext of $(X_i = f_i s_i)_i$, it can extract $dk_f$
- This would work with a perfect IP-MCFE: any plaintext can be decrypted 😨
- Here, only small plaintexts can be decrypted: $dk_f$ is large! 😭
DMCFE: IP with Pairings

- Two IP-MCFE: $E_1$ in $G_1$ and $E_2$ in $G_2$
- The senders encrypt the messages $x_i$ with $E_1$
- The senders encrypt the functional key shares $X_i$ with $E_2$
- The aggregator knows the functional key for $(1,\ldots,1)$ in $E_2 \rightarrow$ it gets $g_2^{E_{f,f}}$
- From $g_2^{E_{f,f}}$ and ciphertexts of $x_i$ with $E_1$ in $G_1 \rightarrow$ one gets $g_1^{E_{f,x}}$

The discrete logarithm is small: can be extracted!

DMCFE: IP with Pairings

- Our Decentralised Multi-Client Functional Encryption:
  - Selective Security
  - Even with Adaptive Corruptions of the Clients/Senders
  - Under the classical SXDH assumption
  - Efficient Setup: generation of the functional key for $(1,\ldots,1)$
  - Efficient $DKeyGen$ protocol: just one ciphertext sent by each sender

Conclusion

- Functional Encryption
  - Ideal functionalities on encrypted data
  - Authority-based functionality
  - Inputs from a unique sender
- DMCFE
  - Aggregation of multi-source inputs
  - Functionality under control of the senders