Simple Functional Encryption Schemes for Inner Products

Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval

École normale supérieure, CNRS, INRIA, PSL, Paris, France

PKC 2015 — Maryland, USA
Wednesday, April 1
1. Overview of the results
   - What is Functional Encryption?
   - Inner Product functionality
   - What does simple mean? What do we achieve?

2. The Framework
   - Overview of the framework
   - Example
   - Proof of security
   - Generalization

3. Work in progress
   - What is there left to do?
   - Thank you!
What is Functional Encryption?

Introduced by Dan Boneh, Amit Sahai and Brent Waters [BSW10]

Generalizes multiple concepts:
- Identity-Based Encryption
- Fuzzy Identity-Based Encryption
- Attribute-Based Encryption
- Predicate Encryption, etc.

Enables keys that give partial information.
What is Functional Encryption?
Introduced by Dan Boneh, Amit Sahai and Brent Waters [BSW10]
What is Functional Encryption?

Introduced by Dan Boneh, Amit Sahai and Brent Waters [BSW10]

Generalizes multiple concepts:
Brief history

*What is Functional Encryption?*

Introduced by Dan Boneh, Amit Sahai and Brent Waters [BSW10]

Generalizes multiple concepts:

- Identity-Based Encryption
What is Functional Encryption?

Introduced by Dan Boneh, Amit Sahai and Brent Waters [BSW10]

Generalizes multiple concepts:

- Identity-Based Encryption
- Fuzzy Identity-Based Encryption
What is Functional Encryption?

Introduced by Dan Boneh, Amit Sahai and Brent Waters [BSW10]

Generalizes multiple concepts:

- Identity-Based Encryption
- Fuzzy Identity-Based Encryption
- Attribute-Based Encryption
What is Functional Encryption?
Introduced by Dan Boneh, Amit Sahai and Brent Waters [BSW10]
Generalizes multiple concepts:

- Identity-Based Encryption
- Fuzzy Identity-Based Encryption
- Attribute-Based Encryption
- Predicate Encryption, etc.
What is Functional Encryption?

Introduced by Dan Boneh, Amit Sahai and Brent Waters [BSW10]

Generalizes multiple concepts:

- Identity-Based Encryption
- Fuzzy Identity-Based Encryption
- Attribute-Based Encryption
- Predicate Encryption, etc.

Enables keys that give partial information.
Motivation

What is Functional Encryption?
- Inner Product functionality
- What does simple mean? What do we achieve?

Overview of the results
- The Framework
- Work in progress

Simple Functional Encryption Schemes for Inner Products

Alice → The Cloud
Motivation

Alice ➔ Pictures of Bob? ➔ ???

The Cloud
Formal definition

Functionality $\mathcal{F} : \mathcal{K} \times \mathcal{X} \rightarrow \mathcal{M}$

$$(k, x) \mapsto \mathcal{F}(k, x)$$

Secret key for $k : \text{sk}_k \leftarrow msk$

Ciphertext for $x : \text{ct}_x \leftarrow pk$
Formal definition

Functionality $F : \mathcal{K} \times \mathcal{X} \rightarrow \mathcal{M}$

$$(k, x) \mapsto F(k, x)$$

Secret key for $k : \text{sk}_k \leftarrow \text{msk}$

Ciphertext for $x : \text{ct}_x \leftarrow \text{pk}$

Correctness

$\text{Decrypt}(\text{sk}_k, \text{ct}_x) = F(k, x)$
Formal definition

Functionality $\mathcal{F} : \mathcal{K} \times \mathcal{X} \rightarrow \mathcal{M}$

$(k, x) \mapsto \mathcal{F}(k, x)$

($(\text{Picture}, \text{Bob}), \text{data}) \mapsto \text{Pictures of Bob}$

Secret key for $k : \text{sk}_k \leftarrow \text{msk}$

Ciphertext for $x : \text{ct}_x \leftarrow \text{pk}$

Correctness

$\text{Decrypt}(\text{sk}_k, \text{ct}_x) = \mathcal{F}(k, x)$
Formal definition

Functionality $\mathcal{F} : \mathcal{K} \times \mathcal{X} \rightarrow \mathcal{M}$

$$(k, x) \mapsto \mathcal{F}(k, x)$$

$((\text{Picture,Bob}), \text{data}) \mapsto \text{Pictures of Bob}$

Secret key for $k : sk_k \leftarrow msk$

Ciphertext for $x : ct_x \leftarrow pk$

Correctness

$\text{Decrypt}(sk_k, ct_x) = \mathcal{F}(k, x)$

Alice gets Bob’s pictures in her data.
Security

Intuitively:
$s_{\mathcal{K}}$ doesn’t leak any more information than $F(k, x)$

Even if there are collusions:
$s_{\mathcal{K}}$ and $s_{\mathcal{K}}'$ don’t leak more information than $F(k, x)$ and $F(k', x)$
Intuitively:
\( sk_k \) doesn’t leak any more information than \( F(k, x) \)

The server doesn’t access Alice’s private data other than needed.

Even if there are collusions!

\( sk_k \) and \( sk'_k \) don’t leak more information than \( F(k, x) \) and \( F(k', x) \)

Pictures of Jean and pictures of Jacques don’t make pictures of Jean-Jacques.
current lines of work

- Designing efficient functional encryption for access control...
current lines of work

- Designing efficient functional encryption for access control... nothing about partial information
current lines of work

- Designing efficient functional encryption for access control... nothing about partial information
- Obtain functional encryption for all circuits...
current lines of work

- Designing efficient functional encryption for access control... nothing about partial information
- Obtain functional encryption for all circuits... construction from inefficient primitives
current lines of work

- Designing efficient functional encryption for access control... nothing about partial information
- Obtain functional encryption for all circuits... construction from inefficient primitives
- This work: figuring out what we can do with simple assumption
### Inner Product functionality

Functionality $\mathcal{F} : \mathbb{Z}_p^\ell \times \mathbb{Z}_p^\ell \rightarrow \mathbb{Z}_p$

$$(y, x) \mapsto \langle x, y \rangle$$

Secret key for $y$ : $sk_y$

Ciphertext for $x$ : $ct_x$
Inner Product functionality

Functionality $F : \mathbb{Z}_p^l \times \mathbb{Z}_p^l \rightarrow \mathbb{Z}_p$

$(y, x) \rightarrow < x, y >$

Secret key for $y : sk_y$

Ciphertext for $x : ct_x$

Correctness

Decrypt$(y, ct_x) = < x, y >$
Motivation example: Online dating system

- Kindness
- Intelligence
- Beauty
Motivation example: Online dating system
Motivation example: Online dating system
Motivation example: Online dating system
Properties

Inner product is very interesting:
- lots of applications
- easy to compute - only need additions if one vector is known
- still non-trivial: $|\mathcal{K}|$ is exponential in $\ell$
- theoretically interesting problem - enables any computation in $NC^0$
Inherent security limitation

\[ \langle x, y \rangle \] gives a lot of information about \( x \)

\( \ell \) well chosen secret keys reveals everything
Basic primitive: PKE with some additional structural properties

Our framework can be instantiated with different well known Public Key Encryption schemes.
Basic primitive: PKE with some additional structural properties

Our framework can be instantiated with different well known Public Key Encryption schemes. Additive ElGamal, based on Decisional Diffie-Hellman (DDH) assumption.
Basic primitive: PKE with some additional structural properties

Our framework can be instantiated with different well known Public Key Encryption schemes:
- Additive ElGamal, based on Decisional Diffie-Hellman (DDH) assumption
- Lattice based Public Key Encryption scheme, based on the Learning With Errors (LWE) assumption
Ciphertext size is $\ell + 1$ elements
Key size is 1 element
Efficient

Ciphertext size is $\ell + 1$ elements
Key size is 1 element
This is really close to information theoretical optimal for correctness
The resulting scheme is secure under selective chosen plaintext attacks.

**Security game:**
- \( \mathcal{A} \) submits \( x_0, x_1 \)
Selective IND-CPA security

The resulting scheme is secure under selective chosen plaintext attacks

Security game:

- $\mathcal{A}$ submits $x_0, x_1$
- $\mathcal{A}$ receives $pk, ct_{x_0}$
Selective IND-CPA security

The resulting scheme is secure under selective chosen plaintext attacks.

**Security game:**

- $\mathcal{A}$ submits $x_0, x_1$
- $\mathcal{A}$ receives $pk, ct_{x_b}$
- $\mathcal{A}$ sends some set of queries $\{y\}$, such that
  \[
  \langle x_0, y \rangle = \langle x_1, y \rangle
  \]
The resulting scheme is secure under selective chosen plaintext attacks.

**Security game:**
- $A$ submits $x_0, x_1$
- $A$ receives $pk, ct_{x_b}$
- $A$ sends some set of queries $\{y\}$, such that $<x_0, y> = <x_1, y>$
- $A$ receives $\{sk_y\}$
Selective IND-CPA security

The resulting scheme is secure under selective chosen plaintext attacks

**Security game:**

- $A$ submits $x_0, x_1$
- $A$ receives $pk, ct_{xb}$
- $A$ sends some set of queries $\{y\}$, such that $\langle x_0, y \rangle = \langle x_1, y \rangle$
- $A$ receives $\{sk_y\}$
- $A$ guesses $b'$
How to apply our framework?

Our framework is easy to instantiate:

**Pick a good Public Key Encryption scheme**

requires structural properties stated later
How to apply our framework?

Our framework is easy to instantiate:

Pick a good Public Key Encryption scheme
requires structural properties stated later

Reuse Randomness to encrypt a vector
How to apply our framework?

Our framework is easy to instantiate:

**Pick a good Public Key Encryption scheme**
requires structural properties stated later

**Reuse Randomness** to encrypt a vector

**Use additive homomorphism** to decrypt the correct value
How to apply our framework?

Our framework is easy to instantiate:

Pick a good Public Key Encryption scheme
requires structural properties stated later

Reuse Randomness to encrypt a vector

Use additive homomorphism to decrypt the correct value

And it’s done!
How to apply our framework?

Our framework is easy to instantiate:

Pick a good Public Key Encryption scheme
  requires structural properties stated later

Reuse Randomness to encrypt a vector

Use additive homomorphism to decrypt the correct value

And it’s done! (and safe!)
The additively homomorphic ElGamal public key encryption scheme

Public parameters: \( p, G, g \)

Secret key: \( s \)

Public key: \( g^s \)

Ciphertext for \( m \): \( (g^r, g^{rs} g^m) \)
The additively homomorphic ElGamal public key encryption scheme

Public parameters: \( p, G, g \)

Secret key: \( s \)

Public key: \( g^s \)

Ciphertext for \( m \): \( (g^r, g^{rs}g^m) \)

Correctness

\[
\frac{g^{rs}g^m}{(g^r)^s} = g^m
\]
Reusing randomness

Public parameters: $p, G, g$

Secret key: $s$

Public key: $g^s$

Ciphertext for $m$: $(g^r, g^{rs} g^m)$
Reusing randomness

Public parameters: $p, G, g, \ell$

Secret key: $\vec{s} = s_1 \ldots s_\ell$

Public key: $g^{\vec{s}} = g^{s_1} \ldots g^{s_\ell}$

Ciphertext for $\vec{x}$: $(g^r, g^{r \vec{s}} g^{\vec{x}} = g^{rs_1} g^{x_1} \ldots g^{rs_\ell} g^{x_\ell})$
Reusing randomness

Public parameters: \( p, G, g, \ell \)

Secret key: \( \vec{s} = s_1 \ldots s_\ell \)

Public key: \( g^{\vec{s}} = g^{s_1} \ldots g^{s_\ell} \)

Ciphertext for \( \vec{x} \): \( (g^r, g^{rs\vec{s}} g^{\vec{x}} = g^{rs_1} g^{x_1} \ldots g^{rs_\ell} g^{x_\ell}) \)

Now onto correctness...
Using homomorphism to decrypt the inner product

Secret key: \( \vec{s} = s_1 \ldots s_\ell \)

Public key: \( g^{\vec{s}} = g^{s_1} \ldots g^{s_\ell} \)

Ciphertext for \( \vec{x} \): \( (g^r, g^{rs_1} g^{x_1} \ldots g^{rs_\ell} g^{x_\ell}) \)

Correctness

\[ g^{rs_1} g^{x_1} g^{rs_2} g^{x_2} = g^{r(s_1 + s_2)} g^{x_1 + x_2} \]
Overview of the results

The Framework

Work in progress

Overview of the framework

Example

Proof of security

Generalization

Using homomorphism to decrypt the inner product

Secret key: \( \vec{s} = s_1 \ldots s_\ell \)

Public key: \( g^{\vec{s}} = g^{s_1} \ldots g^{s_\ell} \)

Ciphertext for \( \vec{x} \): \( (g^r, g^{rs_1}g^{x_1} \ldots g^{rs_\ell}g^{x_\ell}) \)

Correctness

\[
g^{rs_1}g^{x_1}g^{rs_2}g^{x_2} = g^{r(s_1 + s_2)}g^{x_1 + x_2}
\]

\[
\prod_{i}(g^{rs_i}g^{x_i})^{y_i} = (g^r)\sum_{i}y_is_ig^{\sum_{i}x_iy_i}
\]
First trick

You can change easily the basis used in the whole scheme
You can change easily the basis used in the whole scheme

Given a matrix $P$, a ciphertext $ct_{\vec{x}}$, and the master secret key $\vec{s}$

You can generate a new ciphertext $ct_{P\vec{x}}$ using the homomorphism, and a new master secret key $P\vec{s}$
In the security game, there exists a basis in which the adversary cannot find the first coordinate.
In the security game, there exists a basis in which the adversary cannot find the first coordinate
Indeed, $\mathcal{A}$ can only ask secret keys for $\vec{y}$ such that
$\langle \vec{y}, \vec{x}_1 - \vec{x}_0 \rangle \geq 0$
So a basis having $\vec{x}_1 - \vec{x}_0$ as first vector verifies this
Here is a simulator $S$ using both tricks to solve a challenge given an adversary breaking the scheme:
Putting it together

Here is a simulator $S$ using both tricks to solve a challenge given an adversary breaking the scheme:

- $S$ finds a basis having $\vec{x}_1 - \vec{x}_0$ as first vector
Putting it together

Here is a simulator $S$ using both tricks to solve a challenge given an adversary breaking the scheme:

- $S$ finds a basis having $\vec{x}_1 - \vec{x}_0$ as first vector
- $S$ generates $ct^*$ with its input challenge in the first coordinate
Here is a simulator $S$ using both tricks to solve a challenge given an adversary breaking the scheme:

- $S$ finds a basis having $\vec{x}_1 - \vec{x}_0$ as first vector
- $S$ generates $ct^*$ with its input challenge in the first coordinate
- $S$ moves $ct^*$ in the correct basis
Here is a simulator $S$ using both tricks to solve a challenge given an adversary breaking the scheme:

- $S$ finds a basis having $\vec{x}_1 - \vec{x}_0$ as first vector
- $S$ generates $ct^*$ with its input challenge in the first coordinate
- $S$ moves $ct^*$ in the correct basis
What properties do we need?

2 properties:

**Randomness Reuse** $g^r$, $g^{rs}$ is safe

In this case, it is an instance of ElGamal with secret keys $r$ and randomnesses $s_i$

**Homomorphism of message and key**

$$g^{rs_1 + x_1} g^{rs_2 + x_2} = g^r(s_1 + s_2) + (x_1 + x_2)$$
How to generalize?

To generalize, replace:

- $s \rightarrow sk$
- $g^s \rightarrow pk$
- $g^r \rightarrow C(r)$
- $g^{rs+x} \rightarrow Enc(pk, x; r)$
the LWE assumption

Public parameters: \( q, n, m, A \in \mathbb{Z}^{m \times n} \)

Secret key: \( \vec{s} \in \mathbb{Z}_q^m \)

Public key: \( A\vec{s} + \vec{e} \in \mathbb{Z}_q^m \)

Ciphertext for \( x \): \( (\vec{r}A, \vec{r}(A\vec{s} + \vec{e}) + \lfloor \frac{q}{2} \rfloor x) \) \( \vec{r} \leftarrow \{0, 1\}^{1 \times m} \)
the LWE assumption

Public parameters: \( q, n, m, A \in \mathbb{Z}_q^{m \times n} \)

Secret key: \( \vec{s} \in \mathbb{Z}_q^m \)

Public key: \( A\vec{s} + \vec{e} \in \mathbb{Z}_q^m \)

\( \vec{e} \leftarrow \chi^m \)

Ciphertext for \( x \): \( (\vec{r}A, \vec{r}(A\vec{s} + \vec{e}) + \left\lfloor \frac{q}{2} \right\rfloor x) \)

\( \vec{r} \leftarrow \{0, 1\}^{1 \times m} \)

Advantages

- Avoid small space restriction of additive ElGamal
- Post-quantum
the LWE assumption

Public parameters: \( q, n, m, A \in \mathbb{Z}_q^{m \times n} \)

Secret key: \( \vec{s} \in \mathbb{Z}_q^m \)

Public key: \( A \vec{s} + \vec{e} \in \mathbb{Z}_q^m \) \( \vec{e} \leftarrow \chi^m \)

Ciphertext for \( x \): \( (\vec{r}A, \vec{r}(A\vec{s} + \vec{e}) + \lfloor \frac{q}{2} \rfloor x) \) \( \vec{r} \leftarrow \{0, 1\}^{1 \times m} \)

Advantages
- Avoid small space restriction of additive ElGamal
- Post-quantum

Inconveniences
- Noisy setup - proof is more subtle
Work in progress

What is there left to do?

- Adaptive security
  \( \mathcal{A} \) gets \( pk \) before choosing \( \vec{x}_0 \) and \( \vec{x}_1 \)
Work in progress

Work in progress
What is there left to do?

- Adaptive security
  - $\mathcal{A}$ gets $\textbf{pk}$ before choosing $\vec{x}_0$ and $\vec{x}_1$

- Function privacy
  - In private setting - $\mathcal{A}$ doesn't know what his key compute
Work in progress

What is there left to do?

- Adaptive security
  \( \mathcal{A} \) gets \( \textbf{pk} \) before choosing \( \vec{x}_0 \) and \( \vec{x}_1 \)

- Function privacy
  In private setting - \( \mathcal{A} \) doesn’t know what his key compute

- Find other interesting fitting PKE
  Paillier-like cryptosystem would solve the small space restrictions

- etc.
Thank you for your attention!