Unbounded Inner Product Functional Encryption, with Succinct Keys

Edouard Dufour Sans and David Pointcheval

École Normale Supérieure
INRIA

June 6, 2019

Table of Contents

Functional Encryption

Traditional PKE: all or nothing.

Functional Encryption

Traditional PKE: all or nothing.

- Have the key?
  Get the plaintext.
- Don't have the key?
  Get nothing.

Functional Encryption

Traditional PKE: all or nothing.

- Have the key?
  Get the plaintext.
- Don't have the key?
  Get nothing.

Functional Encryption: **A new paradigm**.

Functional Encryption

Traditional PKE: all or nothing.

- Have the key?
  Get the plaintext.
- Don't have the key?
  Get nothing.

Functional Encryption: **A new paradigm**.
Get a *function* of the cleartext.

Functional Encryption

Traditional PKE: all or nothing.

- Have the key?
  Get the plaintext.
- Don't have the key?
  Get nothing.

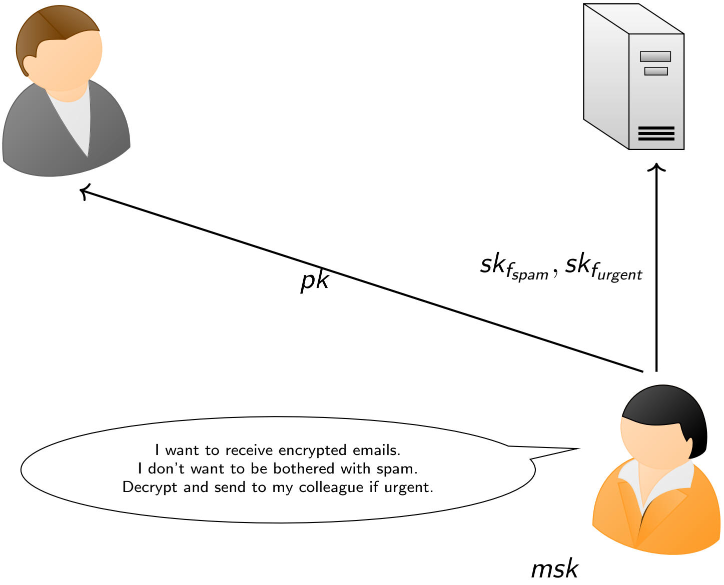
Functional Encryption: **A new paradigm**.
Get a *function* of the cleartext.
Function depends on the key.

Functional Encryption: Formal definition

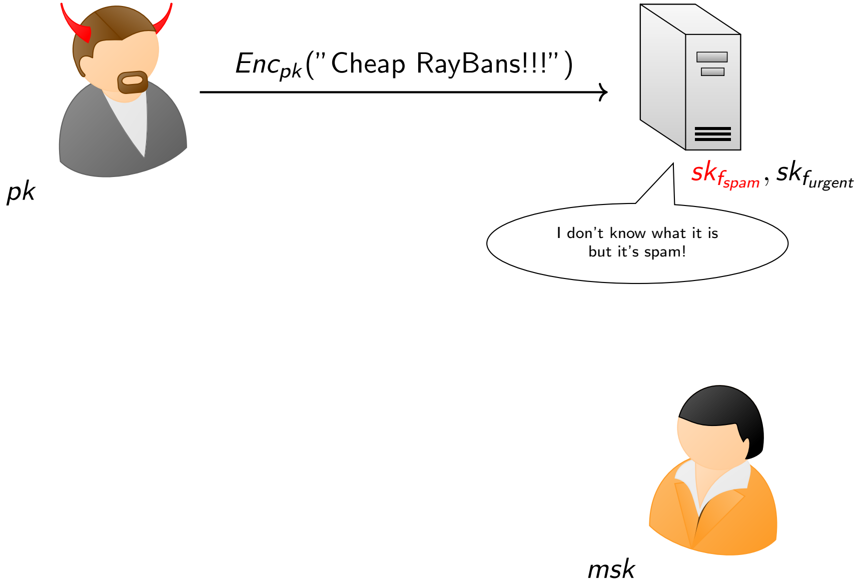
Four algorithms:

- Setup(λ): Returns (pk, msk).
- Encrypt(pk,x): Returns c.
- KeyGen(msk,f): Returns sk_f.
- Decrypt(sk_f,c): Returns $f(x)$.

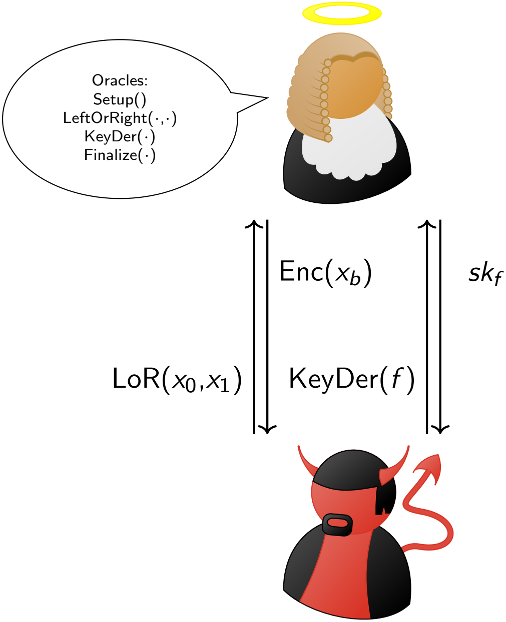
FE example


$sk_{f_{spam}}, sk_{f_{urgent}}$
pk
I want to receive encrypted emails.
I don't want to be bothered with spam.
Decrypt and send to my colleague if urgent.
msk

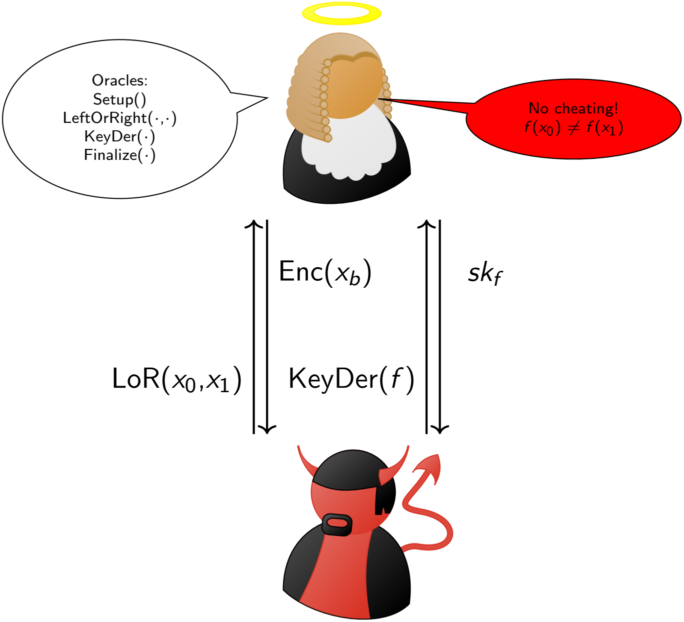
FE example


Enc_{pk}(" Cheap RayBans!!!")
pk
$sk_{f_{spam}}$, $sk_{f_{urgent}}$
I don't know what it is
but it's spam!
msk

Security definitions



Security definitions



The First Inner Product Functional Encryption

ABDP15

Fixed n. $\mathcal{F} \approx \mathbb{Z}_p^n$, $f_{\vec{y}} \approx \vec{y}$.

- Setup(λ): Pick $\vec{s} \xleftarrow{\$} \mathbb{Z}_p^n$. Return $g^{\vec{s}}, \vec{s}$.
- Encrypt($g^{\vec{s}}$, $\vec{x}$): Pick $r \xleftarrow{\$} Z_p$. Return $g^r, g^{\vec{x}} \cdot \left(g^{\vec{s}}\right)^r = g^r, g^{\vec{x}+r\cdot\vec{s}}$.
- KeyGen($\vec{s}$, $\vec{y}$): Return $\langle \vec{s}, \vec{y} \rangle$.
- Decrypt($\langle \vec{s}, \vec{y} \rangle$, $(g^r, g^{\vec{x}+r\cdot\vec{s}})$): Compute

$$g^{\gamma} = \langle g^{\vec{x}+r\cdot\vec{s}}, \vec{y} \rangle / \left(g^r\right)^{\langle \vec{s}, \vec{y} \rangle}$$

and solve the discrete logarithm to return γ.

Application: Descriptive statistics

- Averages.
- Weighted averages.

Application: Descriptive statistics

- Averages.
- Weighted averages.
- Standard deviation.

Application: Descriptive statistics

- Averages.
- Weighted averages.
- Standard deviation (if we encrypt the squares).

Application: Descriptive statistics

- Averages.
- Weighted averages.
- Standard deviation (if we encrypt the squares).
- Machine Learning Inference via Linear Regression.

Leakage

Say you have a ciphertext for vector $\mathbf{x}$.
The key for $\mathbf{y}$ lets you compute $\langle \mathbf{x}, \mathbf{y} \rangle \implies$ one projection.

Leakage

Say you have a ciphertext for vector $\mathbf{x}$.
The key for $\mathbf{y}$ lets you compute $\langle \mathbf{x}, \mathbf{y} \rangle \implies$ one projection.
m independent keys $\implies$ m projections.

Leakage

Say you have a ciphertext for vector $\mathbf{x}$.
The key for $\mathbf{y}$ lets you compute $\langle \mathbf{x}, \mathbf{y} \rangle \implies$ one projection.
m independent keys $\implies$ m projections.
Actual number of keys you can give?

Leakage

Say you have a ciphertext for vector $\mathbf{x}$.
The key for $\mathbf{y}$ lets you compute $\langle \mathbf{x}, \mathbf{y} \rangle \implies$ one projection.
m independent keys $\implies$ m projections.
Actual number of keys you can give depends on plaintext distribution.

Table of Contents

Limitations of Inner Product Functional Encryption

What if you want to receive vectors of various lengths?

Limitations of Inner Product Functional Encryption

What if you want to receive vectors of various lengths?
You need multiple public keys.

Limitations of Inner Product Functional Encryption

What if you want to receive vectors of various lengths?
You need multiple public keys.
What if you want to create subcategories between vectors?

Limitations of Inner Product Functional Encryption

What if you want to receive vectors of various lengths?
You need multiple public keys.
What if you want to create subcategories between vectors?
More keys.

Limitations of Inner Product Functional Encryption

What if you want to receive vectors of various lengths?
You need multiple public keys.
What if you want to create subcategories between vectors?
More keys.
What if you don't know the size of the vector ahead of time?

Limitations of Inner Product Functional Encryption

What if you want to receive vectors of various lengths?
You need multiple public keys.
What if you want to create subcategories between vectors?
More keys.
What if you don't know the size of the vector ahead of time?
No great solutions.

Solution: Unbounded Inner Product Functional Encryption

- No fixed size for vectors (ciphertexts or keys).
- One constant-size public-key.
- Vectors are maps from indices to scalars.
- Identity-based version allows for categorization.

UIPFE Variants

We introduce two unbounded functionalities:

UIPFE Variants

We introduce two unbounded functionalities:

- Strict UIPFE: Indices of ciphertext must match those of key.

UIPFE Variants

We introduce two unbounded functionalities:

- Strict UIPFE: Indices of ciphertext must match those of key.
- Permissive UIPFE: Indices of ciphertext must contain those of key.

Technical overview

ABDP builds on El Gamal.
Want n coordinates? Instantiate n El Gamal schemes you control.

Technical overview

ABDP builds on El Gamal.
Want n coordinates? Instantiate n El Gamal schemes you control.
How do we go to Unbounded?

Technical overview

ABDP builds on El Gamal.
Want n coordinates? Instantiate n El Gamal schemes you control.
How do we go to Unbounded?
Boneh-Franklin Identity-Based Encryption is ElGamal-like.

Our construction

Permissive UIPFE: Setup

Choose a pairing group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ and a hash function $\mathcal{H}$ into $\mathbb{G}_2$.
Pick a single scalar $s \xleftarrow{\$} \mathbb{Z}_p$.
Return g_1^s, s.

Our construction

Permissive UIPFE: Encrypt

- Setup(λ): Pick $s \xleftarrow{\$} \mathbb{Z}_p$. Return g_1^s, s.

You have an unbounded vector $(x_i)_{i \in \mathcal{D}}$ and $pk = g_1^s$.
Pick $r \xleftarrow{\$} \mathbb{Z}_p$. Return $(g_1^r, (c_i)_{i \in \mathcal{D}})$ where

$$c_i = g_T^{x_i} \cdot e(g_1^s, \mathcal{H}(i)^r) \approx g_T^{x_i + rs_i}$$

Our construction

Permissive UIPFE: KeyGen

- Setup(λ): Pick $s \xleftarrow{\$} \mathbb{Z}_p$. Return g_1^s, s.
- Encrypt(g^s, $(x_i)_{i \in \mathcal{D}}$): Pick $r \xleftarrow{\$} \mathbb{Z}_p$. Return $(g_1^r, (c_i)_{i \in \mathcal{D}})$ where
$$c_i = g_T^{x_i} \cdot e(g_1^s, \mathcal{H}(i)^r) \approx g_T^{x_i + rs_i}$$

You have an unbounded vector $(y_i)_{i \in \mathcal{D}'}$ and $sk = s$.
Return
$$\prod_{i \in \mathcal{D}'} \mathcal{H}(i)^{-sy_i} \approx g_2^{-\langle \vec{s}, \vec{y} \rangle}$$

Our construction

Permissive UIPFE: Decrypt

- Setup(λ): Pick $s \xleftarrow{\$} \mathbb{Z}_p$. Return g_1^s, s.
- Encrypt(g^s, $(x_i)_{i\in\mathcal{D}}$): Pick $r \xleftarrow{\$} \mathbb{Z}_p$. Return $(g_1^r, (c_i)_{i\in\mathcal{D}})$ where
$$c_i = g_T^{x_i} \cdot e(g_1^s, \mathcal{H}(i)^r) \approx g_T^{x_i + rs_i}$$
- KeyGen(s, $(y_i)_{i\in\mathcal{D}'}$): Return
$$\prod_{i\in\mathcal{D}'} \mathcal{H}(i)^{-sy_i} \approx g_2^{-\langle\vec{s},\vec{y}\rangle}$$

You have a ciphertext $(g_1^r, (c_i)_{i\in\mathcal{D}})$ and a key $\prod_{i\in\mathcal{D}'} \mathcal{H}(i)^{-sy_i}$
Compute
$$g_T^{\gamma} = e\left(g_1^r, \prod_{i\in\mathcal{D}'} \mathcal{H}(i)^{-sy_i}\right) \cdot \prod_{i\in\mathcal{D}'} c_i^{y_i}$$

and recover γ.

Our construction

Permissive UIPFE

- Setup(λ): Pick $s \xleftarrow{\$} \mathbb{Z}_p$. Return g_1^s, s.
- Encrypt(g^s, $(x_i)_{i\in\mathcal{D}}$): Pick $r \xleftarrow{\$} \mathbb{Z}_p$. Return $(g_1^r, (c_i)_{i\in\mathcal{D}})$ where

$$c_i = g_T^{x_i} \cdot e(g_1^s, \mathcal{H}(i)^r) \approx g_T^{x_i + rs_i}$$

- KeyGen(s, $(y_i)_{i\in\mathcal{D}'}$): Return

$$\prod_{i\in\mathcal{D}'} \mathcal{H}(i)^{-sy_i} \approx g_2^{-\langle \vec{s},\vec{y}\rangle}$$

- Decrypt($\prod_{i\in\mathcal{D}'} \mathcal{H}(i)^{-sy_i} \approx g_2^{-\langle \vec{s},\vec{y}\rangle}$, $(g_1^r, (c_i)_{i\in\mathcal{D}})$): Compute

$$g_T^{\gamma} = e\left(g_1^r, \prod_{i\in\mathcal{D}'} \mathcal{H}(i)^{-sy_i}\right) \cdot \prod_{i\in\mathcal{D}'} c_i^{y_i}$$

and recover γ.

Technical Difficulties: Norms

$$||x_0 - x_1|| = 0 \mod p \not\Rightarrow x_0 = x_1 \mod p$$

Other UIPFE works bypass this by bounding individual components.

This doesn't work here.

We define a pseudonorm and impose an upper bound on it.

Technical Difficulties: Key Homomorphism

In most (all?) IPFE schemes, keys are homomorphic:

$$f(\alpha, sk_y, \beta, sk_{y'}) = sk_{\alpha y + \beta y'}$$

This is typically fine by functionality.

Technical Difficulties: Key Homomorphism

In most (all?) IPFE schemes, keys are homomorphic:

$$f(\alpha, sk_y, \beta, sk_{y'}) = sk_{\alpha y + \beta y'}$$

This is typically fine by functionality.
But it becomes an issue in permissive UIPFE.
Need to adjust security definitions.

Concurrent and Independent Work

Tomida and Takashima proposed UIPFE at ASIACRYPT18.

Concurrent and Independent Work

Tomida and Takashima proposed UIPFE at ASIACRYPT18.

- No Random Oracles.
- Adaptive security.
- Only standard assumptions.

Concurrent and Independent Work

Tomida and Takashima proposed UIPFE at ASIACRYPT18.

- No Random Oracles.
- Adaptive security.
- Only standard assumptions.

- Requires contiguous indices.
- No access control.
- Bigger keys, slower operations.

	Public Key	Ciphertext	Functional Key
TT18	$28\lvert\mathbb{G}_1\rvert$	$7n\lvert\mathbb{G}_1\rvert$	$7n\lvert\mathbb{G}_2\rvert + \alpha$
Ours	$\lvert\mathbb{G}_1\rvert$	$\lvert\mathbb{G}_1\rvert + n\lvert\mathbb{G}_T\rvert$	$\lvert\mathbb{G}_2\rvert$

Open problems

- Better security with efficiency.

Open problems

- Better security with efficiency.
- Different UIPFE functionalities.

Open problems

- Better security with efficiency.
- Different UIPFE functionalities.
- More functionalities.

Open problems

- Better security with efficiency.
- Different UIPFE functionalities.
- More functionalities.

References

1. Abdalla, Michel, et al. "Simple functional encryption schemes for inner products." IACR International Workshop on Public Key Cryptography. Springer, Berlin, Heidelberg, 2015.
2. Boneh, Dan, and Matt Franklin. "Identity-based encryption from the Weil pairing." Annual international cryptology conference. Springer, Berlin, Heidelberg, 2001.
3. Boneh, Dan, Amit Sahai, and Brent Waters. "Functional encryption: Definitions and challenges." Theory of Cryptography Conference. Springer, Berlin, Heidelberg, 2011.
4. O'Neill, Adam. "Definitional Issues in Functional Encryption." IACR Cryptology ePrint Archive 2010 (2010): 556.
5. Tomida, Junichi, and Katsuyuki Takashima. "Unbounded Inner Product Functional Encryption from Bilinear Maps." International Conference on the Theory and Application of Cryptology and Information Security. Springer, Cham, 2018.