Inner Product Functional Encryption

Edouard Dufour Sans

January 25, 2018

Table of Contents

Functional Encryption

Traditional PKE: all or nothing.

Functional Encryption

Traditional PKE: all or nothing.

- Have the key? Get the plaintext.

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:

Functional Encryption: Formal definition

Four algorithms:

- Setup
- Encrypt
- KeyGen
- Decrypt

Functional Encryption: Formal definition

Four algorithms:

- Setup(λ): Returns (ek, msk).
- Encrypt
- KeyGen
- Decrypt

Functional Encryption: Formal definition

Four algorithms:

- Setup(λ): Returns (ek, msk).
- Encrypt(ek,x): Returns c.
- KeyGen
- Decrypt

Functional Encryption: Formal definition

Four algorithms:

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

Functional Encryption: Formal definition

Four algorithms:

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

Functional Encryption: Formal definition

Four algorithms:

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

Function hiding.

Functional Encryption: Formal definition

Four algorithms:

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

Function hiding (or *not*).

Functional Encryption: Formal definition

Four algorithms:

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

Function hiding (or *not*).
$f \in \mathcal{F}$: the *functionality*.

Security definitions

Can we simply re-use the definitions of standard SE or PKE?

Security definitions

Can we simply re-use the definitions of standard SE or PKE?
No.

Security definitions

Can we simply re-use the definitions of standard SE or PKE?
No.
For any non-trivial $f \implies$ distinguish by submitting x_0, x_1 with $f(x_0) \neq f(x_1)$.

Security definitions

Can we simply re-use the definitions of standard SE or PKE?
No.
For any non-trivial $f \implies$ distinguish by submitting x_0, x_1 with $f(x_0) \neq f(x_1)$.
Would not be a useful definition.

Security definitions

Indistinguishability-Based Game

Security definitions

Indistinguishability-Based Game

Polynomial number of queries to the following oracles:

Security definitions

Indistinguishability-Based Game

Polynomial number of queries to the following oracles:

- Initialize: Run the setup and send the public key.

Security definitions

Indistinguishibility-Based Game

Polynomial number of queries to the following oracles:

- Initialize: Run the setup and send the public key.
- KeyDer: Run KeyGen and give the decryption key.

Security definitions

Indistinguishability-Based Game

Polynomial number of queries to the following oracles:

- Initialize: Run the setup and send the public key.
- KeyDer: Run KeyGen and give the decryption key.
- LeftOrRight: Receive (x_0, x_1), return Encrypt(ek, x_b).

Security definitions

Indistinguishibility-Based Game

Polynomial number of queries to the following oracles:

- Initialize: Run the setup and send the public key.
- KeyDer: Run KeyGen and give the decryption key.
- LeftOrRight: Receive (x_0, x_1), return Encrypt(ek, x_b).
- Finalize: If key requests were legitimate, check validity of guess.

Security definitions

Indistinguishibility-Based Game

Polynomial number of queries to the following oracles:

- Initialize: Run the setup and send the public key.
- KeyDer: Run KeyGen and give the decryption key.
- LeftOrRight: Receive (x_0, x_1), return Encrypt(ek, x_b).
- Finalize: If key requests were legitimate, check validity of guess.

One query to LeftOrRight is enough.

Security definitions

Indistinguishibility-Based Game

Polynomial number of queries to the following oracles:

- Initialize: Run the setup and send the public key.
- KeyDer: Run KeyGen and give the decryption key.
- LeftOrRight: Receive (x_0, x_1), return Encrypt(ek, x_b).
- Finalize: If key requests were legitimate, check validity of guess.

One query to LeftOrRight is enough.
Requests were illegitimate if for some f queries to KeyDer, $f(x_0) \neq f(x_1)$.

Security definitions

Indistinguishibility-Based Game

Polynomial number of queries to the following oracles:

- Initialize: Run the setup and send the public key.
- KeyDer: Run KeyGen and give the decryption key.
- LeftOrRight: Receive (x_0, x_1), return Encrypt(ek, x_b).
- Finalize: If key requests were legitimate, check validity of guess.

One query to LeftOrRight is enough.
Requests were illegitimate if for some f queries to KeyDer, $f(x_0) \neq f(x_1)$.
Selective game: Adversary must query LeftOrRight first.
Adaptive game: No such constraint.

Notations

- Brackets: $[x] = g^x$.
- Matrices and brackets:

$$\left[\begin{pmatrix} x_{11} & \dots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{d1} & \dots & x_{dn} \end{pmatrix}\right] = \begin{pmatrix} [x_{11}] & \dots & [x_{1n}] \\ \vdots & \ddots & \vdots \\ [x_{d1}] & \dots & [x_{dn}] \end{pmatrix}$$

- We encrypt vectors $\mathbf{x}$, and give keys for vectors $\mathbf{y}$. We conflate $f_{\mathbf{y}} : \mathbf{x} \to \sum_{i=1}^{n} x_i y_i$ and $\mathbf{y}$.
- Scalar x, vector $\mathbf{x}$ and matrix $\mathbf{X}$.

Table of Contents

The Power of Inner Products

We will work towards constructing schemes for the inner product functionality.

The Power of Inner Products

We will work towards constructing schemes for the inner product functionality.

Is this a useful primitive?

Descriptive statistics

- Averages.

Descriptive statistics

- Averages.
- Weighted averages.

Descriptive statistics

- Averages.
- Weighted averages.
- Standard deviation.

Descriptive statistics

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

Machine learning: linear regression

Predict t (e.g. income) from $\mathbf{x}$ (e.g. housing data about the family).

Machine learning: linear regression

Predict t (e.g. income) from **x** (e.g. housing data about the family).
A somewhat naive model:

$$
\begin{aligned}
t &\approx \sum_{i=1}^{n} x_i y_i \\
&\approx \langle \mathbf{x}, \mathbf{y} \rangle
\end{aligned}
$$

Machine learning: linear regression

Predict t (e.g. income) from $\mathbf{x}$ (e.g. housing data about the family).
A somewhat naive model:

$$\begin{aligned} t &\approx \sum_{i=1}^{n} x_i y_i \\ &\approx \langle \mathbf{x}, \mathbf{y} \rangle \end{aligned}$$

Works very well for some (basic) problems!

Machine learning: linear classification

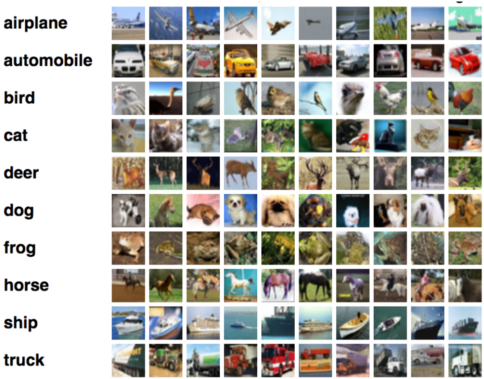

Figure: The CIFAR10 dataset.
Source: https://www.cs.toronto.edu/∼kriz/cifar.html

Machine learning: linear classification

Figure: CIFAR10 linear classifiers as images.
Source: http://cs231n.github.io/linear-classify/

Leakage

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

Leakage

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

Leakage

The key for 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

The key for 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

Presentation

ABDP15

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

Presentation

ABDP15

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

- Setup
- Encrypt
- KeyGen
- Decrypt

Presentation

ABDP15

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

- Setup(λ): Pick $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_p^n$. Return $[\mathbf{s}], \mathbf{s}$.
- Encrypt
- KeyGen
- Decrypt

Presentation

ABDP15

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

- Setup(λ): Pick $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_p^n$. Return $[\mathbf{s}], \mathbf{s}$.
- Encrypt($[\mathbf{s}]$, $\mathbf{x}$): Pick $r \xleftarrow{\$} Z_p$. Return $[r], [\mathbf{x}] \cdot [\mathbf{s}]^r = [r], [\mathbf{x} + r\mathbf{s}]$.
- KeyGen
- Decrypt

Presentation

ABDP15

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

- Setup(λ): Pick $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_p^n$. Return $[\mathbf{s}], \mathbf{s}$.
- Encrypt($[\mathbf{s}]$, $\mathbf{x}$): Pick $r \xleftarrow{\$} Z_p$. Return $[r], [\mathbf{x} + r\mathbf{s}]$.
- KeyGen($\mathbf{s}$, $\mathbf{y}$): Return $\langle \mathbf{s}, \mathbf{y} \rangle$.
- Decrypt

Presentation

ABDP15

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

- Setup(λ): Pick $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_p^n$. Return $[\mathbf{s}], \mathbf{s}$.
- Encrypt($[\mathbf{s}]$, $\mathbf{x}$): Pick $r \xleftarrow{\$} Z_p$. Return $[r], [\mathbf{x} + r\mathbf{s}]$.
- KeyGen($\mathbf{s}$, $\mathbf{y}$): Return $\langle \mathbf{s}, \mathbf{y} \rangle$.
- Decrypt($\langle \mathbf{s}, \mathbf{y} \rangle$, $([r], [\mathbf{x} + r\mathbf{s}])$): Compute

$$[\gamma] = \quad [\mathbf{x} + r\mathbf{s}]^\intercal \cdot \mathbf{y} / [r]^{\langle \mathbf{s}, \mathbf{y} \rangle}$$

and solve the discrete logarithm to return γ.

Correctness

- Decrypt($\langle \mathbf{s}, \mathbf{y} \rangle$, $([r], [\mathbf{x} + r\mathbf{s}])$): Compute

$$[\gamma] = \quad [\mathbf{x} + r\mathbf{s}]^{\intercal} \cdot \mathbf{y} / [r]^{\langle \mathbf{s}, \mathbf{y} \rangle}$$

  and solve the discrete logarithm to return γ.

Proof.
On the black board, or check the paper. □

Table of Contents

Presentation

ALS16

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

Presentation

ALS16

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

- Setup
- Encrypt
- KeyGen
- Decrypt

Presentation

ALS16

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

- Setup(λ): Pick $\mathbf{a} \xleftarrow{\$} \mathbb{Z}_p^2$, $\mathbf{S} \xleftarrow{\$} \mathbb{Z}_p^{n \times 2}$. Return $([\mathbf{a}], [\mathbf{Sa}]), (\mathbf{a}, \mathbf{S})$.
- Encrypt
- KeyGen
- Decrypt

Presentation

ALS16

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

- Setup(λ): Pick $\mathbf{a} \xleftarrow{\$} \mathbb{Z}_p^2$, $\mathbf{S} \xleftarrow{\$} \mathbb{Z}_p^{n\times 2}$. Return $([\mathbf{a}], [\mathbf{Sa}]), (\mathbf{a}, \mathbf{S})$.
- Encrypt(($[\mathbf{a}], [\mathbf{Sa}]$), $\mathbf{x}$): Pick $r \xleftarrow{\$} Z_p$. Return $[\mathbf{a}r], [\mathbf{x} + \mathbf{Sa}r]$.
- KeyGen
- Decrypt

Presentation

ALS16

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

- Setup(λ): Pick $\mathbf{a} \xleftarrow{\$} \mathbb{Z}_p^2$, $\mathbf{S} \xleftarrow{\$} \mathbb{Z}_p^{n\times 2}$. Return $([\mathbf{a}], [\mathbf{Sa}]), (\mathbf{a}, \mathbf{S})$.
- Encrypt(($[\mathbf{a}], [\mathbf{Sa}]$), $\mathbf{x}$): Pick $r \xleftarrow{\$} Z_p$. Return $[\mathbf{a}r], [\mathbf{x} + \mathbf{Sa}r]$.
- KeyGen($\mathbf{S}$, $\mathbf{y}$): Return $\mathbf{S}^\intercal\mathbf{y}$.
- Decrypt

Presentation

ALS16

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

- Setup(λ): Pick $\mathbf{a} \xleftarrow{\$} \mathbb{Z}_p^2$, $\mathbf{S} \xleftarrow{\$} \mathbb{Z}_p^{n\times 2}$. Return $([\mathbf{a}], [\mathbf{Sa}]), (\mathbf{a}, \mathbf{S})$.
- Encrypt(($[\mathbf{a}], [\mathbf{Sa}]$), $\mathbf{x}$): Pick $r \xleftarrow{\$} Z_p$. Return $[\mathbf{a}r], [\mathbf{x} + \mathbf{Sa}r]$.
- KeyGen($\mathbf{S}$, $\mathbf{y}$): Return $\mathbf{S}^\intercal \mathbf{y}$.
- Decrypt($\mathbf{S}^\intercal \mathbf{y}$, ($[\mathbf{a}r], [\mathbf{x} + \mathbf{Sa}r]$)): Compute

$$[\gamma] = [\mathbf{x} + \mathbf{Sa}r]^\intercal \cdot \mathbf{y} - [\mathbf{a}r]^\intercal \cdot \mathbf{S}^\intercal \mathbf{y}$$

and solve the discrete logarithm to return γ.

Correctness

Compute

$$[\gamma] = [(\mathbf{x} + \mathbf{S}\mathbf{a}r)^\intercal \mathbf{y} - (\mathbf{a}r)^\intercal \mathbf{S}^\intercal \mathbf{y}]$$

and solve the discrete logarithm to return γ.

Proof.

On the black board (or check the paper). $\square$

Security

ALS16

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

- Setup(λ): Pick $\mathbf{a} \xleftarrow{\$} \mathbb{Z}_p^2$, $\mathbf{S} \xleftarrow{\$} \mathbb{Z}_p^{n\times 2}$. Return $([\mathbf{a}], [\mathbf{S}\mathbf{a}]), (\mathbf{a}, \mathbf{S})$.
- Encrypt($([\mathbf{a}], [\mathbf{S}\mathbf{a}])$, $\mathbf{x}$): Pick $r \xleftarrow{\$} Z_p$. Return $[\mathbf{a}r], [\mathbf{x} + \mathbf{S}\mathbf{a}r]$.
- KeyGen($\mathbf{S}$, $\mathbf{y}$): Return $\mathbf{S}^\intercal\mathbf{y}$.
- Decrypt($\mathbf{S}^\intercal\mathbf{y}$, $([\mathbf{a}r], [\mathbf{x} + \mathbf{S}\mathbf{a}r])$): Compute

$$[\gamma] = [(\mathbf{x} + \mathbf{S}\mathbf{a}r)^\intercal\mathbf{y} - (\mathbf{a}r)^\intercal\mathbf{S}^\intercal\mathbf{y}]$$

  and solve the discrete logarithm to return γ.

Proof.

On the black board (or check Appendix A in AGR+17). $\square$

References

1. M. Abdalla, F. Bourse, A. De Caro, and D. Pointcheval. Simple functional encryption schemes for inner products. PKC 2015.
2. M. Abdalla, R. Gay, M. Raykova, and H. Wee. Multi-input Inner-Product Functional Encryption from Pairings. EUROCRYPT 2017.
3. S. Agrawal, B. Libert, and D. Stehlé. Fully secure functional encryption for inner products, from standard assumptions. CRYPTO 2016.
4. D. Boneh, A. Sahai, and B. Waters. Functional encryption: Definitions and challenges. TCC 2011.
5. A. O'Neill. Definitional Issues in Functional Encryption. Cryptology ePrint Archive, Report 2010/556.