Functional Encryption

Allison Lewko, Microsoft Research
### The Cast of Characters

This talk will feature work by:

<table>
<thead>
<tr>
<th>Brent Waters</th>
<th>Shweta Agrawal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jon Katz</td>
<td>Sergey Gorbunov</td>
</tr>
<tr>
<td>Amit Sahai</td>
<td>Vinod Vaikuntanathan</td>
</tr>
<tr>
<td>Dan Boneh</td>
<td></td>
</tr>
<tr>
<td>Spot guest appearances by:</td>
<td></td>
</tr>
<tr>
<td>Adam O’Neill, Yael Kalai, Shafi Goldwasser, Raluca Ada Popa, Nickolai Zeldovich</td>
<td></td>
</tr>
</tbody>
</table>
Motivation

Who should have access to my data? What should they see?
A Broad Vision

Data D

$f_3(D)$

$f_1(D)$

$f_4(D)$

$f_2(D)$

$f_3$ and $f_4$ have keys to access $f_3(D)$ and $f_4(D)$, respectively.

$f_1$ and $f_2$ have keys to access $f_1(D)$ and $f_2(D)$, respectively.
A First Step

How might we hide the access policy itself?

Access Denied!

Data D

Why?
Inner Product Encryption

\[ \lambda = \text{security parameter} \quad \text{and} \quad n = \text{vector length} \]

**Setup(\( \lambda, n \)):**
- generate public parameters \( PP \) and master key \( MSK \)

**KeyGen(\( \mathbf{u}, MSK \)):**
- generate a user key for a given vector of length \( n \)

**Encrypt(PP, M, \( \mathbf{x} \)):**
- encrypt message \( M \) under a vector of length \( n \)

**Decrypt(CT, SK):**
- decrypt ciphertext using a key: successful iff \( \mathbf{x} \cdot \mathbf{u} \equiv 0 \)
IND-CPA game:

- **PP, MSK**
- **PP**
- **vector v**
- **SK_v**
- **M_0, M_1**
- **Encrypt(PP, M_b, x_b)**
- **vector v**
- **SK_v**
- **Repeat**

**Attacker**

- Required that $v \cdot x_1 = v \cdot x_2$

**Challenger**

- If $v \cdot x_1 = 0$ for some $v$, required that $M_0 = M_1$
We want to compute $\vec{x} \cdot \vec{v}$ while hiding $\vec{x}$:

Basic idea: compute $\vec{x} \cdot \vec{v}$ in the exponent

CT: $g^{x_1} g^{x_2} g^{x_3} \ldots g^{x_n}$

SK: $g^{v_1} g^{v_2} g^{v_3} \ldots g^{v_n}$

$e(g, g)^{x_1 v_1} e(g, g)^{x_2 v_2} e(g, g)$

Some remaining problems:

Can tell if $x_i = 0$ or not

Can permute the computation
Subgroup Roles

- $G_{p_1}$: Encode vectors $\tilde{x}, \tilde{y}$
- $G_{p_2}$: Enforce computation done properly
- $G_{p_3}$: Extra randomness
IPE Construction (Predicate Only) [KSW08]

Setup \((\lambda, n)\): generate \(G\) of order \(N = p_1p_2p_3\)

\[ PP = g, g, gR, \{h_i^V_i, k_i^W_i\}_{i=1}^n \]

Encrypt \((\vec{x} = (x_1, x_2, \ldots, x_n))\):

choose \(s, \alpha, \beta \in \mathbb{Z}_N\)

\[ CT = g^s, \{(h_i^V_i)^s(gV_i)^{\alpha x_i}U_i, (k_i^W_i)^s(gV_i)^{\beta x_i}Z_i\}_{i=1}^n \]

KeyGen \((\vec{\nu} = (\nu_1, \nu_2, \ldots, \nu_n))\):

\[ SK = AB \prod_{i=1}^n h_i^{r_i} k_i^{-t_i}, \{g^{r_i}g^{\nu_i}, g^{t_i}g^{\sigma_i}\}_{i=1}^n \]

2 parallel systems, stayed tuned for why we want this
Decryption

CT: \[ g^s \quad (h_i V_i)^s (g V_i)^{\alpha x_i} U_i \quad (k_i W_i)^s (g V_i)^{\beta x_i} Z_i \]

Take product for \( i = 1 \) to \( n \)

SK: \[ AB \prod_{i=1}^n h_i^{-r_i} k_i^{-t_i} \quad g^{r_i} g^{\delta v_i} \quad g^{t_i} g^{\sigma v_i} \]

\[ \prod_{i=1}^n e(g, h_i)^{-s r_i} e(g, k_i)^{-s t_i} \quad e(g, h_i)^{s r_i} e(g, g)^{\alpha \delta x_i v_i} \quad e(g, k_i)^{s t_i} e(g, g)^{\beta \sigma x_i v_i} \]

\[ e(g, g)^{(\alpha \delta + \beta \sigma) \bar{x} \cdot \bar{v}} \]

\[ = 1 \text{ if and only if } \bar{x} \cdot \bar{v} \equiv 0 \mod p_1 \]
Proof Intuition

New challenge:

Adversary attempting to distinguish CT under $\vec{x}$ from CT under $\vec{y}$ requests key for $\vec{v}$ such that $\vec{v} \cdot \vec{x} = \vec{v} \cdot \vec{y} = 0$

Natural approach would be a hybrid changing $\vec{x}$ to $\vec{y}$ one coordinate at a time:

$$(x_1, \ldots, x_n) \Rightarrow (x_1, \ldots, x_i, y_{i+1}, \ldots, y_n) \Rightarrow (y_1, \ldots, y_n)$$

may not be orthogonal to $\vec{v}$!
Proof Intuition

Idea: use two parallel systems, change one half at a time

Hybrid structure: CT exponent vectors change as

$$(\vec{x}, \vec{x}) \Rightarrow (\vec{x}, \vec{0}) \Rightarrow (\vec{x}, \vec{y}) \Rightarrow (\vec{0}, \vec{y}) \Rightarrow (\vec{y}, \vec{y})$$

Why go through $\vec{0}$?

$\vec{0}$ is orthogonal to everything, and we can go from $\vec{x}$ to $\vec{0}$ with Subgroup Decision Assumptions
A general definition [BSW11]:

Def. 1.
A functionality $F$ is a function $F : K \times X \rightarrow \{0, 1\}^*$ whose domain is the product of a key space $K$ and a plaintext space $X$. It is required that $K$ include an “empty key” $\epsilon$. 

\[ k \in K \]
\[ x \in X \]
Formal Specification

Setup(\(\lambda\)):

- generate public parameters PP and master key MSK

KeyGen(\(k, MSK\)):

- generate a user key for a given \(k \in K\)

Encrypt(PP, \(x \in X\)):

- encrypt message \(x\)

Decrypt(CT, SK):

- decrypt ciphertext using a key to obtain \(F(k,x)\)
IND Game-Based Security Definition

IND-CPA game:

- \( k_i \in K \)
- \( m_0, m_1 \)
- \( \text{Encrypt}(PP, m_b) \)
- \( \text{SK for } k_i \)
- \( \text{PP, MSK} \)
- \( \text{Challenger} \)
- \( \text{Attacker} \)

Required that \( F(k_i, m_0) = F(k_i, m_1) \) \( \forall k_i \) requested
When IND Security May Be Insufficient

It does not capture computational properties of the functionality $F$:

Example:
Consider $K = \{\epsilon\}$, $X = \{0, 1\}^n$
$F(\epsilon, x) := \pi(x)$

one-way permutation

Proposed Construction:
$Encrypt(x) = x$

$$F(\epsilon, x_1) = F(\epsilon, x_2) \iff x_1 = x_2$$

So this is “secure” under game-based definition!
A Further Example

from [O10]: Let $\mathcal{F} = \{f_1, \ldots, f_n\}$ be set of functions associated with keys
Let $g$ be a function so that given $f_1(x) || \ldots || f_n(x)$,
it is hard to guess $g(x)$
And $g(x) = g(y) \iff f_1(x) || \ldots || f_n(x) = f_1(y) || \ldots || f_n(y)$

Let (Setup, Encrypt, Decrypt) be a Public Key Encryption scheme
Let (Setup*, KeyGen*, Encrypt*, Decrypt*) be a FE scheme

New FE scheme:

**Setup:** run Setup and Setup* to get $(pk, sk), (pk^*, sk^*)$
secret share $sk$ as $\omega_1, \ldots, \omega_n$
set $pk := pk || pk^*, sk := \omega_1 || \ldots || \omega_n || sk^*$

**KeyGen($f_i$):** run KeyGen*($f_i$) to get $sk_i$, set $sk_i := \omega_i || sk_i$

**Encrypt($m$):** run Encrypt*($m$) and Encrypt($g(m)$), concat results
A Simulation-Based Security Definition

Real World:
- Challenger
  - PP
  - $k_i$
  - $SK_{k_i}$
  - $m \leftarrow \mathcal{M}$
  - $\mathcal{M}$
  - CT
  - $k_j$
  - $SK_{k_j}$

Ideal World:
- Simulator
  - PP
  - $k_i$
  - $SK_{k_i}$
  - $m \leftarrow \mathcal{M}$
  - $\mathcal{M}$
  - CT
  - $k_j$
  - $SK_{k_j}$

\[ \Rightarrow F(k_i, m) \forall i \]
\[ \Rightarrow F(k_j, m) \]
Impossibility Results for Sim-Based Security

$m_1, m_\cdot \leftarrow m \mathcal{M} \leftarrow \mathcal{M}$

Ideal World:

Simulator $\rightarrow$ PP $\rightarrow$ Attacker

$\leftarrow k_i$

$\rightleftharpoons SK_{k_i}$

$\mathcal{M} \leftarrow \mathcal{M}$

$\Rightarrow \{F(k_i, m_\ell) \ \forall i, \ell\}$

$\Rightarrow F(k_j, m_1), \ldots, F(k_j, m_N)$

$CT_1, \ldots, CT_N$

$\leftarrow k_j$

$SK_{k_j}$

needs to encode too much!
Impossibility Results for Sim-Based Security

$\mathbf{Ideal\ World:}$

Simulator $\xrightarrow{\text{PP}}$ Attacker

$m \leftarrow \mathcal{M}$

$k_i$

$SK_{k_i}$

$\mathcal{M}$

$\Rightarrow \{F(k_i, m) \forall i\}$

$\text{CT}$

*Can avoid this by bounding the queries

[AGVW12]

for some $F$, needs to encode too much!
Positive Result for Bounded Collusion [GVW12]

- Impose bound of $q$ on key queries
- Can build a scheme for general circuit functionalities
- Techniques include: secret sharing, garbled circuits
Result for general functionalities with succinct CT, Bounded collusion [GKPVZ13]:

- draws upon ABE and FHE constructions
- can be instantiated from LWE
- applications to garbled circuits, obfuscation, delegation