Controlled Functional Encryption

Novel models can bring modern cryptographic technology to real applications.

Muhammad Naveed, Shashank Agrawal, Manoj Prabhakaran, XiaoFeng Wang, Erman Ayday, Jean-Pierre Hubaux, Carl A. Gunter

UIUC, IU, EPFL

This research program is supported by a collaborative award from the National Science Foundation (NSF award number CNS-1330491). The views and conclusions in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of NSF.
Problem 1

Patient
or any agent of the patient
Problem 1

Patient

or any agent of the patient

Health Record
Problem 1

Patient
or any agent of the patient

Health Record
Problem 1

Patient or any agent of the patient

Only $f \left( \text{EHR} \right)$
Problem 1

- Type of tests
- Number of tests

Patient

Health Record

Only $f(EHR)$

or any agent of the patient
Problem 1

- Type of tests
- Number of tests

Compute \( f() \) if \( f \) satisfies

Only \( f(\text{EHR}) \)

Patient

or any agent of the patient

Health Record
Health Records Breaches

• HHS Breaches Affecting 500 or More Individuals
  • Short URL: http://goo.gl/3NuWwx (case sensitive)

• Affected Individuals: **39 million**

• Affected Institutions: **1141**
Problem 2

Volunteer
or any agent of the volunteer
Problem 2

Volunteer or any agent of the volunteer

Genome
Problem 2

Volunteer or any agent of the volunteer

Genome
Problem 2

Volunteer or any agent of the volunteer

Only $f(\text{Genome})$
Problem 2

- Type of studies
- Number of studies

Genome

Volunteer or any agent of the volunteer

Only $f(\text{DNA})$
Problem 2

- Type of studies
- Number of studies

Volunteer

or any agent of the volunteer

Compute $f(\text{Genome})$ if $f$ satisfies

Only $f(\text{Genome})$
Havasupai tribe vs. Arizona State University Case

• In 1989, ASU researchers collected Havasupai’s DNA samples to study Type II Diabetes

• Type II Diabetes study was unsuccessful

• Used for anthropological study, results of which conflicted tribe’s beliefs about their origin

• DNA was used to study *taboo topics*
General Problem

Data owner

Clients

Compute $f(x)$ if $f$ satisfies [policy]
Goal

A new cryptographic model that perfectly aligns with real applications, and allows the design of very efficient protocols.
Requirements

• Practically efficient for real applications
• No interaction with the data owner
• Data owner controls access to the data
• Only data owner sees plaintext
Why existing cryptographic models do not work?
Two-party Computation

Data Owner

Client
Two-party Computation

Data Owner

Client
Two-party Computation

Data Owner

Client
Two-party Computation

- Data Owner
- Client

Diagram showing the interaction between the Data Owner and the Client.
Two-party Computation

Data Owner

\[ f \]

Client

Learns \( f(Data) \)
Two-party Computation

Data Owner

Client
Two-party Computation

Data Owner

Client $f'$
Two-party Computation

Data Owner

Client

$f'$
Two-party Computation

Data Owner

Client

\( f' \)

Learns \( f' (\text{Data}) \)
Two-party Computation

Interaction with the data owner

\[ f' \text{ Learns } f'(\text{Data}) \]
Functional Encryption (FE)

Key Authority

Data owner

Client
1. Authority generates MSK, MPK

Key Authority

Data owner

Client
Functional Encryption (FE)

1. Authority generates MSK, MPK
2. Authority sends MPK to the data owner
Functional Encryption (FE)

1. Authority generates MSK, MPK
2. Authority sends MPK to the data owner
3. Data owner encrypts data with MPK

(MSK, MPK)

Key Authority

Data owner

Client
Functional Encryption (FE)

1. Authority generates MSK, MPK
2. Authority sends MPK to the data owner
3. Data owner encrypts data with MPK
4. Data owner sends ciphertext to the client
Functional Encryption (FE)

1. Authority generates MSK, MPK
2. Authority sends MPK to the data owner
3. Data owner encrypts data with MPK
4. Data owner sends ciphertext to the client
5. Client requests key for function f from the authority
1. Authority generates MSK, MPK
2. Authority sends MPK to the data owner
3. Data owner encrypts data with MPK
4. Data owner sends ciphertext to the client
5. Client requests key for function $f$ from the authority
6. Authority sends key $K_f$ to the client
7. Client computes $f$(data)
1. Authority generates MSK, MPK
2. Authority sends MPK to the data owner
3. Data owner encrypts data with MPK
4. Data owner sends ciphertext to the client
5. Client requests key for function \( f \) from the authority
6. Authority sends key \( K_f \) to the client
7. Client computes \( f(\text{data}) \)
Functional Encryption (FE)

1. Authority generates MSK, MPK
2. Authority sends MPK to the data owner
3. Data owner encrypts data with MPK
4. Data owner sends ciphertext to the client
5. Client requests key for function $f$ from the authority
6. Authority sends key $K_f$ to the client
7. Client computes $f(data)$

\[
\text{Dec}(\text{data}, K_f) = f(data)
\]

Key Authority

Data owner

Client
Functional Encryption (FE)

1. Authority generates MSK, MPK
2. Authority sends MPK to the data owner
3. Data owner encrypts data with MPK
4. Data owner sends ciphertext to the client
5. Client requests key for function $f$ from the authority
6. Authority sends key $K_f$ to the client
7. Client computes $f(data)$

Data owner

Client

Key Authority

$(MSK, MPK)$

Key Request for function $f$

$Dec(MSK, MPK) = f(data)$
Functional Encryption (FE)

1. Authority generates MSK, MPK
2. Authority sends MPK to the data owner
3. Data owner encrypts data with MPK
4. Data owner sends ciphertext to the client
5. Client requests key for function f from the authority
6. Authority sends key $K_f$ to the client
7. Client computes $f(data)$

Key Authority

Data owner

Key Request for function f

Client

Dec($key$, DATA) = $f(DATA)$
Functional Encryption (FE)

1. Authority generates MSK, MPK
2. Authority sends MPK to the data owner
3. Data owner encrypts data with MPK
4. Data owner sends ciphertext to the client
5. Client requests key for function f from the authority
6. Authority sends key $K_f$ to the client
7. Client computes $f$\(\text{(data)}\)

\[
\text{Dec}(\text{MPK}, \text{DATA}) = f(\text{DATA})
\]
Functional Encryption (FE)

1. Authority generates MSK, MPK
2. Authority sends MPK to the data owner
3. Data owner encrypts data with MPK
4. Data owner sends ciphertext to the client
5. Client requests key for function \( f \) from the authority
6. Authority sends key \( K_f \) to the client
7. Client computes \( f(data) \)

\[
\text{Dec}(\text{MSK}, \text{MPK}) = f(data)
\]
Functional Encryption (FE)

1. Authority generates MSK, MPK
2. Authority sends MPK to the data owner
3. Data owner encrypts data with MPK
4. Data owner sends ciphertext to the client
5. Client requests key for function $f$ from the authority
6. Authority sends key $K_f$ to the client
7. Client computes $f(data)$

**Key Authority**

Extremely inefficient
New untested assumptions
Our model:
Controlled Functional Encryption (CFE)
Controlled Functional Encryption (CFE)

Key Authority

Data owner

Client
Controlled Functional Encryption (CFE)

1. Authority generates MSK, MPK

Key Authority

Data owner

Client
Controlled Functional Encryption (CFE)

1. Authority generates MSK, MPK
2. Authority sends MPK to the data owner
Controlled Functional Encryption (CFE)

1. Authority generates MSK, MPK
2. Authority sends MPK to the data owner
3. Data owner encrypts data with MPK
Controlled Functional Encryption (CFE)

1. Authority generates MSK, MPK
2. Authority sends MPK to the data owner
3. Data owner encrypts data with MPK

Key Authority

Data owner

Client

(MSK, MPK)
Controlled Functional Encryption (CFE)

1. Authority generates MSK, MPK
2. Authority sends MPK to the data owner
3. Data owner encrypts data with MPK
4. Data owner sends ciphertext to the client
Controlled Functional Encryption (CFE)

1. Authority generates MSK, MPK
2. Authority sends MPK to the data owner
3. Data owner encrypts data with MPK
4. Data owner sends ciphertext to the client
5. Client requests ciphertext specific key for function f from the authority

Key Authority

Data owner

Client

(MSK, MPK)
Controlled Functional Encryption (CFE)

1. Authority generates MSK, MPK
2. Authority sends MPK to the data owner
3. Data owner encrypts data with MPK
4. Data owner sends ciphertext to the client
5. Client requests ciphertext specific key for function f from the authority
6. Authority sends key $K_{f,r}$ to the client

Data owner

Key Authority

Client

Key Request for function f

$K_{f,r}$
Controlled Functional Encryption (CFE)

1. Authority generates MSK, MPK
2. Authority sends MPK to the data owner
3. Data owner encrypts data with MPK
4. Data owner sends ciphertext to the client
5. Client requests **ciphertext specific key** for function f from the authority
6. Authority sends key $K_{f,r}$ to the client
7. Client computes $f(\text{data})$

**Key Authority**

Data owner

Client

$\text{Dec}(\text{key}, \text{data}) = f(\text{data})$
Controlled Functional Encryption (CFE)

1. Authority generates MSK, MPK
2. Authority sends MPK to the data owner
3. Data owner encrypts data with MPK
4. Data owner sends ciphertext to the client
5. Client requests ciphertext specific key for function f from the authority
6. Authority sends key $K_{i,r}$ to the client
7. Client computes $f$ (data)

Decryption: $\text{Dec}(\text{key}, \text{ciphertext}) = f(\text{data})$
Controlled Functional Encryption (CFE)

1. Authority generates MSK, MPK
2. Authority sends MPK to the data owner
3. Data owner encrypts data with MPK
4. Data owner sends ciphertext to the client
5. Client requests **ciphertext specific key** for function f from the authority
6. Authority sends key $K_{f,r}$ to the client
7. Client computes $f(data)$

```
Dec( , ) = f(data)
```

Data owner

Client

Key Authority

(MSK, MPK)
Controlled Functional Encryption (CFE)

1. Authority generates MSK, MPK
2. Authority sends MPK to the data owner
3. Data owner encrypts data with MPK
4. Data owner sends ciphertext to the client
5. Client requests ciphertext specific key for function f from the authority
6. Authority sends key $K_{i,r}$ to the client
7. Client computes $f(data)$

Dec($\text{key}_{i,r}, \text{DATA}_{i,r}$) = $f(\text{DATA}_{i})$
Controlled Functional Encryption (CFE)

1. Authority generates MSK, MPK
2. Authority sends MPK to the data owner
3. Data owner encrypts data with MPK
4. Data owner sends ciphertext to the client
5. Client requests ciphertext specific key for function $f$ from the authority
6. Authority sends key $K_{i,r}$ to the client
7. Client computes $f(data)$

Data owner

Key Authority

Client

$\text{Dec}(\text{Key}, \text{DATA}) = f(\text{DATA})$
Controlled Functional Encryption (CFE)

1. Authority generates MSK, MPK
2. Authority sends MPK to the data owner
3. Data owner encrypts data with MPK
4. Data owner sends ciphertext to the client
5. Client requests ciphertext specific key for function f from the authority
6. Authority sends key $K_{i,r}$ to the client
7. Client computes $f(data)$

Dec($K_{i,r}$, $C$) = $f(DATA)$
Controlled Functional Encryption (CFE)

1. Authority generates MSK, MPK
2. Authority sends MPK to the data owner
3. Data owner encrypts data with MPK
4. Data owner sends ciphertext to the client
5. Client requests ciphertext specific key for function f from the authority
6. Authority sends key $K_{f,r}$ to the client
7. Client computes $f(data)$

$\text{Dec}(\text{key}, \text{ciphertext}) = f(\text{data})$
Controlled Functional Encryption (CFE)

1. Authority generates MSK, MPK
2. Authority sends MPK to the data owner
3. Data owner encrypts data with MPK
4. Data owner sends ciphertext to the client
5. Client requests **ciphertext specific key** for function \( f \) from the authority
6. Authority sends key \( K_{f,r} \) to the client
7. Client computes \( f(\text{data}) \)

\[
\text{Dec}(\text{key}, \text{ciphertext}) = f(\text{data})
\]
Controlled Functional Encryption (CFE)

1. Authority generates MSK, MPK
2. Authority sends MPK to the data owner
3. Data owner encrypts data with MPK
4. Data owner sends ciphertext to the client
5. Client requests ciphertext specific key for function f from the authority
6. Authority sends key $K_{f,r}$ to the client
7. Client computes $f(data)$
CFE vs. FE

FE

Dec(\text{key}, \text{DATA}) = f(\text{DATA})

Dec(\text{key}, \text{DATA}) = f(\text{DATA})

CFE

Dec(\text{key}, \text{DATA}) = f(\text{DATA})

Dec(\text{key}, \text{DATA}) = f(\text{DATA})
Small change makes HUGE difference

- Makes problem more tractable
- Enables fine-grained access control to data
Small change makes HUGE difference

- Makes problem more tractable
- Enables fine-grained access control to data

Practically efficient for real applications
Small change makes HUGE difference

- Makes problem more tractable
- Enables fine-grained access control to data

Practically efficient for real applications

Data owner controls access to the data
Small change makes HUGE difference

- Makes problem more tractable
- Enables fine-grained access control to data

Practically efficient for real applications

Data owner controls access to the data

Only data owner sees plaintext
Small change makes HUGE difference

- Makes problem more tractable
- Enables fine-grained access control to data

- Practically efficient for real applications
- Data owner controls access to the data
- Only data owner sees plaintext
- No interaction with the data owner
Threat Model

- Honest-but curious data owner
- Honest-but curious Key Authority
- Malicious client
Constructions
Superfast CFE Construction
(for inner-product functionality)
Superfast Construction

- Uses Additive Secret Sharing and Public Key Encryption

- As efficient as public key encryption is

- Computes inner product (actual value) of two vectors
Inner-product function family

• Covers a large class of functions

• Genomic Applications
  • Personalized Medicine (weighted average)
  • Patient similarity (hamming distance)

• Set intersection cardinality

• And many others
Key Authority
Data

Key Authority
Data $x$

Key Authority

Function $f_y$
Goal: Compute $f_y(x) = y \cdot x$
Goal: Compute $f_y(x) = y \cdot x$
Goal: Compute $f_y(x) = y \cdot x$
Goal: Compute $f_y(x) = y \cdot x$
Goal: Compute $f_y(x) = y \cdot x$
**Encryption**

MPK

Data \( x \)

Function \( f_y \)

Goal: Compute \( f_y(x) = y \cdot x \)
Encryption

Key Authority

Data $x$  

$(r + x)$

Function $f_y$

Goal: Compute $f_y(x) = y \cdot x$
Encryption

Key Authority

Data $x$

$E_{MPK}(r) (r + x)$

MSK

Function $f_y$

Goal: Compute $f_y(x) = y \cdot x$
Encryption

MPK

Data $x$

Function $f_y(x) = y \cdot x$

Goal: Compute $f_y(x) = y \cdot x$

Key Authority

MSK
Goal: Compute $f_y(x) = y . x$
Goal: Compute $f_y(x) = y \cdot x$
Key Request

Goal: Compute $f_y(x) = y \cdot x$

$E_{MPK}(r)$

Function $f_y$

Key Authority

MSK
Key Request

$E_{MPK}(r)$  Key Authority

Goal: Compute $f_y(x) = y \cdot x$
Goal: Compute $f_y(x) = y \cdot x$
Key Gen

\[ r = D_{MSK} \left( E_{MPK} (r) \right) \]

Function \( f_y \)

Goal: Compute \( f_y(x) = y \cdot x \)
Key Gen

\[ r = D_{MSK}(E_{MPK}(r)) \]

Key Authority

Key \( y \cdot r \)

Goal: Compute \( f_y(x) = y \cdot x \)
Key Gen

\[ r = D_{MSK}(E_{MPK}(r)) \]

MSK

Key Authority

Goal: Compute \( f_y(x) = y \cdot x \)
Decryption

\[ r = D_{MSK}(E_{MPK}(r)) \]

Key Authority

Key \( y \cdot r \)

Function \( f_y \)

Goal: Compute \( f_y(x) = y \cdot x \)
Decryption

\[
\begin{align*}
    \mathbf{r} &= D_{\text{MSK}} \left( E_{\text{MPK}} (\mathbf{r}) \right) \\
    y \cdot (\mathbf{r} + \mathbf{x}) - y \cdot \mathbf{r} &= \text{Goal: Compute } f_y(x) = y \cdot x
\end{align*}
\]
Decryption

\[ r = D_{MSK} \left( E_{MPK}(r) \right) \]

Key Authority

Goal: Compute \( f_y(x) = y \cdot x \)
Decryption

\[ r = D_{\text{MSK}}(E_{\text{MPK}}(r)) \]

Key Authority

Goal: Compute \( f_y(x) = y \cdot x \)

Key \( y \cdot r \)

Function \( f_y \)

\[
\begin{align*}
y \cdot (r+x) - y \cdot r \\
y \cdot x + y \cdot r - y \cdot r \\
y \cdot x
\end{align*}
\]
Goal: Compute $f_y(x) = y \cdot x$

Key Authority

Key $y \cdot r$

Function $f_y$

$y \cdot (r+x) - y \cdot r$

$= y \cdot x + y \cdot r - y \cdot r$

$= y \cdot x$

Decryption

$r = D_{MSK}(E_{MPK}(r))$

$y$

MSK
Future Work
Superfast schemes for other specialized function families?
General CFE Construction
(for any functionality)
General Construction

- Uses Yao’s Protocol and Hybrid Encryption
- As efficient as Yao’s protocol implementation is
- Computes any function
High-level Idea

Key Authority (MSK, MPK)

MPK

Data owner

Client
High-level Idea

Key Authority

Data owner

Client

(MSK, MPK)
High-level Idea

Key Authority

MSK, MPK

Key Request for function f

Data owner

Client
High-level Idea

Key Authority

Data owner

Client

(MSK, MPK)

MPK

Key Request for function f

Key f

DATA
High-level Idea

Data owner

Key Authority

Client

(MSK, MPK)

Secure two party computation

Key Request for function f
High-level Idea

Data owner

Client

Key Authority

Secure two party computation

Key Request for function $f$:

$$\text{Dec}(\text{MPK}, \text{MSK}) = f(\text{DATA})$$
Yao’s protocol

\[ f(x,y) = \text{AND}(x,y) \]

\( x = 0 \)

\( y = 1 \)

Alice

Bob
Yao’s protocol

\[ f(x, y) = \text{AND}(x, y) \]

Alice

Bob

\[ x = 0 \]

\[ y = 1 \]
Yao’s protocol

\[ f(x, y) = \text{AND}(x, y) \]

Alice

\( x = 0 \)

Bob

\( y = 1 \)
Yao’s protocol

\[ f(x,y) = \text{AND}(x,y) \]

Alice

Bob

\( x = 0 \)

\( y = 1 \)

\( k_{0x}, k_{1x} \)

\( k_{0y}, k_{1y} \)

\( k_{0z}, k_{1z} \)
Yao’s protocol

f(x,y) = \text{AND}(x,y)

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Yao's protocol

\[ f(x,y) = \text{AND}(x,y) \]

\begin{align*}
\text{Alice} & : \quad x = 0 \\
& \quad k_{0x}, k_{1x} \\
& \quad k_{0y}, k_{1y} \\
\text{Bob} & : \quad y = 1 \\
& \quad z
\end{align*}

\begin{align*}
\text{Table:} & \quad x & y & z \\
0 & 0 & 0 & E_{k_{0x}}(E_{k_{0y}}(0)) \\
0 & 1 & 0 & E_{k_{0x}}(E_{k_{1y}}(0)) \\
1 & 0 & 0 & E_{k_{1x}}(E_{k_{0y}}(0)) \\
1 & 1 & 1 & E_{k_{1x}}(E_{k_{1y}}(1))
\end{align*}
Yao’s protocol

\[ f(x,y) = \text{AND}(x,y) \]

Permute Rows

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ E_{k_0x}(E_{k_0y}(0)) \]
\[ E_{k_1x}(E_{k_1y}(1)) \]
\[ E_{k_1x}(E_{k_0y}(0)) \]
\[ E_{k_0x}(E_{k_1y}(0)) \]
Yao’s protocol

\[ f(x, y) = \text{AND}(x, y) \]

<table>
<thead>
<tr>
<th>( x )</th>
<th>( y )</th>
<th>( z )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Alice

Bob

\[ E_{k_0x}(E_{k_0y}(0)) \]
\[ E_{k_1x}(E_{k_1y}(1)) \]
\[ E_{k_1x}(E_{k_0y}(0)) \]
\[ E_{k_0x}(E_{k_1y}(0)) \]
Yao’s protocol

\[ f(x,y) = \text{AND}(x,y) \]

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Alice

Bob

\[ E_{k_{0x}}(E_{k_{0y}}(0)) \]
\[ E_{k_{1x}}(E_{k_{1y}}(1)) \]
\[ E_{k_{1x}}(E_{k_{0y}}(0)) \]
\[ E_{k_{0x}}(E_{k_{1y}}(0)) \]
Yao’s protocol

\[ f(x, y) = \text{AND}(x, y) \]

Bob runs OT to learn \( k_{1y} \)

\[
\begin{array}{c|c|c}
  x & y & z \\
  \\
  0 & 0 & 0 \\
  0 & 1 & 0 \\
  1 & 0 & 0 \\
  1 & 1 & 1 \\
\end{array}
\]

\[
E_{k_{0x}}(E_{k_{0y}}(0))
\]
\[
E_{k_{1x}}(E_{k_{1y}}(1))
\]
\[
E_{k_{0x}}(E_{k_{0y}}(0))
\]
\[
E_{k_{0x}}(E_{k_{1y}}(0))
\]
Yao's protocol

\[ f(x,y) = \text{AND}(x,y) \]

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Alice

Bob runs OT to learn \( k_{1y} \)

\[ k_{0y}, k_{1y} \]

Bob

\[ k_{0x} \]

\[ E_{k_{0x}}(E_{k_{0y}}(0)) \]

\[ E_{k_{1x}}(E_{k_{1y}}(1)) \]

\[ E_{k_{0x}}(E_{k_{0y}}(0)) \]

\[ E_{k_{0x}}(E_{k_{1y}}(0)) \]
Yao's protocol

\[ f(x, y) = \text{AND}(x, y) \]

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Alice

\[ k_{0x}, k_{1x} \]

Bob runs OT to learn \( k_{1y} \)

\[ k_{0y}, k_{1y} \text{ OT 1} \]

Bob

\[ E_{k_{0x}}(E_{k_{0y}}(0)) \]
\[ E_{k_{1x}}(E_{k_{1y}}(1)) \]
\[ E_{k_{0x}}(E_{k_{1y}}(0)) \]
\[ E_{k_{0y}}(E_{k_{0y}}(0)) \]
Yao’s protocol

\[ f(x, y) = \text{AND}(x, y) \]

Bob runs OT to learn \( k_{1y} \)

\[
\begin{array}{c|c|c}
 x & y & z \\
--- & --- & --- \\
 0 & 0 & 0 \\
 0 & 1 & 0 \\
 1 & 0 & 0 \\
 1 & 1 & 1 \\
\end{array}
\]
Yao’s protocol

\[ f(x, y) = \text{AND}(x, y) \]

Alice

\[ x = 0 \]

\[ k_{0x}, k_{1x} \]

\[ k_{0y}, k_{1y} \]

Bob

\[ y = 1 \]

\[ k_{0x}k_{1y} \]

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Yao’s protocol

\[ f(x, y) = \text{AND}(x, y) \]

<table>
<thead>
<tr>
<th>( x )</th>
<th>( y )</th>
<th>( z )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Alice

Bob

\( k_{0x}, k_{1x} \)

\( k_{0y}, k_{1y} \)

\( E_{k_{0x}}(E_{k_{0y}}(0)) \)

\( E_{k_{1x}}(E_{k_{1y}}(1)) \)

\( E_{k_{0x}}(E_{k_{1y}}(0)) \)
Yao’s protocol

\[ f(x, y) = \text{AND}(x, y) \]

\[
\begin{array}{c|c|c|c|}
 x & y & z \\
 0 & 0 & 0 \\
 0 & 1 & 0 \\
 1 & 0 & 0 \\
 1 & 1 & 1 \\
\end{array}
\]
Yao’s protocol

\[ f(x, y) = \text{AND}(x, y) \]

\[ x = 0 \]

Alice

\[ y = 1 \]

Bob

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ k_0x, k_1x \]

\[ k_0y, k_1y \]

\[ k_0z, k_1z \]

\[ E_{k_0x}(E_{k_0y}(0)) \]

\[ E_{k_1x}(E_{k_1y}(1)) \]

\[ E_{k_0x}(E_{k_1y}(0)) \]
Key Authority

y=1
Key Authority

x=0

y=1
Goal: Compute $\text{AND}(x,y)$
Goal: Compute AND(x, y)

x = 0

y = 1
Goal: Compute $\text{AND}(x, y)$
**Goal:** Compute \( \text{AND}(x,y) \)
Goal: Compute $\text{AND}(x,y)$
Encryption

MPK

x = 0

Key Authority

MSK

MPK

y = 1

Goal: Compute AND(x, y)
Goal: Compute AND(x, y)
Encryption

**Goal:** Compute $\text{AND}(x, y)$

- $x = 0$
- $y = 1$
- $r^0, r^1$

Key Authority
Encryption

$\text{MPK}$

$x = 0$

$\text{Enc}_{\text{MPK}}(r^0, r^1)$

$r^0$

$y = 1$

Goal: Compute $\text{AND}(x, y)$
Encryption

Goal: Compute \( \text{AND}(x, y) \)

\[ \text{Enc}_{\text{MPK}}(r^0, r^1) \]

\( x = 0 \)

\( y = 1 \)
**Goal:** Compute \( \text{AND}(x, y) \)

- **Key Authority**
  - \( r^0 \)
  - \( r^1 \)

- **Encryption**
  - \( \text{Enc}_{\text{MPK}}(r^0, r^1) \)

- **x = 0**

- **y = 1**
Goal: Compute AND(x, y)
Goal: Compute AND(x,y)

Key Request

Key Authority

Enc_{MPK}(r^0, r^1) = 1
Key Request

Key Authority

\[ \text{Enc}_{\text{MPK}}(r^0, r^1) \quad \text{AND} \]

\[ r^0 \quad \text{AND} \quad y=1 \]

Goal: Compute \( \text{AND}(x,y) \)
Goal: Compute $\text{AND}(x, y)$
Goal: Compute AND(x,y)
Key Gen

Goal: Compute AND(x,y)

Key Authority

$\text{Enc}_{\text{MPK}}(r^0, r^1) \quad \text{AND}$

$r^0, r^1$

$y = 1$
Goal: Compute AND(x, y)

Key Gen

x y z

k₀ₓ, k₁ₓ  k₀₂, k₁₂  MSK

k₀ᵧ, k₁ᵧ

MPK

Enc_{MPK}(r⁰, r¹)

AND

Key Authority

r⁰, r¹

y = 1

r⁰
**Goal:** Compute $\text{AND}(x, y)$

- $k_{0x}, k_{1x}$
- $k_{0y}, k_{1y}$

Encryption formulas:
- $\text{Enc}_{\text{MPK}}(r^0, r^1)$
- $\text{Enc}_{\text{MPK}}(r^0, r^1)$
- $\text{Enc}_{\text{MPK}}(r^0, r^1)$
- $\text{Enc}_{\text{MPK}}(r^0, r^1)$
- $\text{Enc}_{\text{MPK}}(r^0, r^1)$
- $\text{Enc}_{\text{MPK}}(r^0, r^1)$

Key Authority

$r^0, r^1$
Goal: Compute AND(x, y)
Goal: Compute $\text{AND}(x, y)$

Key Authority

Key Gen

$E_{k_0x}(E_{k_0y}(0))$
$E_{k_1x}(E_{k_1y}(1))$
$E_{k_1x}(E_{k_0y}(0))$
$E_{k_0x}(E_{k_1y}(0))$

$y = 1$

$E_{k_0x}(E_{k_0y}(0))$
$E_{k_1x}(E_{k_1y}(1))$
$E_{k_1x}(E_{k_0y}(0))$
$E_{k_0x}(E_{k_1y}(0))$

$r^0$

$r^0, r^1$

$\text{Enc}_{\text{MPK}}(r^0, r^1) \text{ AND} \text{ Enc}_{r^0}(k_{0x}), \text{Enc}_{r^1}(k_{1x})$
**Goal:** Compute \( \text{AND}(x,y) \)

\[
\begin{align*}
E_{k_0x}(E_{k_0y}(0)) & \quad r^0 \\
E_{k_1x}(E_{k_1y}(1)) & \\
E_{k_1x}(E_{k_0y}(0)) & \\
E_{k_0x}(E_{k_1y}(0)) & \\
E_{k_0x}(E_{k_1y}(0)) & \quad y=1
\end{align*}
\]
Goal: Compute AND(x,y)

Key Gen

k₀ₓ, k₁ₓ
k₀ᵧ, k₁ᵧ
k₀z, k₁z

MSK
MPK

Encᵣ₀(k₀ₓ), Encᵣ₁(k₁ₓ)

Eₖ₀ₓ(Eₖ₀ᵧ(0))
Eₖ₁ₓ(Eₖ₁ᵧ(1))
Eₖ₁ₓ(Eₖ₀ᵧ(0))
Eₖ₀ₓ(Eₖ₁ᵧ(0))

y = 1
Goal: Compute $\text{AND}(x,y)$

Key Gen

$Z \leftarrow k_{0z}, k_{1z}$

$X, Y \leftarrow k_{0x}, k_{1x}$

$Y \leftarrow k_{0y}, k_{1y}$

Key Authority

$E_{k_0x}(E_{k_{0y}}(0))$

$E_{k_{1x}}(E_{k_{1y}}(1))$

$E_{k_{1x}}(E_{k_{0y}}(0))$

$E_{k_{0x}}(E_{k_{1y}}(0))$

$r^0\quad y=1$

$\text{Enc}_{r^0}(k_{0x}), \text{Enc}_{r^1}(k_{1x})$
Goal: Compute $\text{AND}(x, y)$

Key Gen

$z = \text{AND}(x, y)$

$\text{Enc}_{r_0}(k_{0x}), \text{Enc}_{r_1}(k_{1x})$

$E_{k_0x}(E_{k_{0y}}(0))$

$E_{k_1x}(E_{k_{1y}}(1))$

$E_{k_1x}(E_{k_{0y}}(0))$

$E_{k_{0x}}(E_{k_{1y}}(0))$

$y = 1$
Goal: Compute $\text{AND}(x,y)$
Goal: Compute \( \text{AND}(x, y) \)

Key Gen

\[ k_{0x}, k_{1x}, k_{0y}, k_{1y}, k_0z, k_1z \]

Key Authority

\[ \text{MSK}, \text{MPK} \]

\[ E_{k_0x}(E_{k_{0y}}(0)), E_{k_1x}(E_{k_{1y}}(1)) \]

OT

\[ r^0, y=1 \]

\[ \text{Enc}_{r^0}(k_{0x}), \text{Enc}_{r^1}(k_{1x}) \]
Goal: Compute AND(x, y)

Key Authority

Key Gen

k₀ₓ, k₁ₓ
k₀ᵧ, k₁ᵧ
k₀z, k₁z

MSK

MPK

AND

z

x

y

OT

k₀ₓ

Encᵣ₀(k₀ₓ), Encᵣ¹(k₁ₓ)

Eₖ₀ₓ(Eₖ₀ᵧ(0))
Eₖ₁ₓ(Eₖ₁ᵧ(1))
Eₖ₁ₓ(Eₖ₀ᵧ(0))
Eₖ₀ₓ(Eₖ₁ᵧ(0))

r⁰

y = 1
Goal: Compute AND(x,y)

Key Gen

\[ k_{0x}, k_{1x} \]

\[ k_{0z}, k_{1z} \]

MSK

MPK

Key Authority

\[ x, y \]

\[ Z \]

\[ E_{k_0}(E_{k_0}(0)) \]
\[ E_{k_1}(E_{k_1}(1)) \]
\[ E_{k_0}(E_{k_0}(0)) \]
\[ E_{k_0}(E_{k_1}(0)) \]
\[ E_{k_0}(E_{k_1}(0)) \]

OT

1

\[ k_{0x}, k_{1x} \]

\[ r^0 \]

\[ y = 1 \]
Goal: Compute \( \text{AND}(x, y) \)
Goal: Compute \( \text{AND}(x,y) \)
Goal: Compute $\text{AND}(x,y)$
Goal: Compute AND(x, y)

Decryption

k₀ₓ, k₁ₓ | x \ y | k₀z, k₁z

Key Authority

Encᵣ₀(k₀ₓ), Encᵣ₁(k₁ₓ)

Eₖ₀ₓ(Eₖ₀ᵧ(0))
Eₖ₁ₓ(Eₖ₁ᵧ(0))
Eₖ₀ₓ(Eₖ₁ᵧ(1))
Eₖ₁ₓ(Eₖ₀ᵧ(1))

y = 1

AND(x, y) = 0
Applications

• Controlled Functional Encryption is a general primitive

• We evaluated our scheme with full scale data for following applications:
  • Personalized medicine
  • Genomic Patient similarity
Implementation & Evaluation
Superfast Scheme

- Macbook Pro (with Intel Core i7 and 8GB RAM) used for evaluation

- **Personalized medicine test**
  - Takes 1.28s and consumes 132KB bandwidth for a 1,000* SNP disease marker

- **Patient genomic similarity**
  - Costs $0.014 per 4-million SNP profile comparison [based on Amazon EC2 pricing]
  - Takes 4 minutes and consumes 53.77MB bandwidth per 4-million SNP profile comparison

*Practical disease markers have at most tens of SNPs
General Scheme

• Uses Huang et al. FastGC implementation

• Hamming Distance
  • Problem size: Two strings of size 1,500,000 bits
  • Encryption: 0.4ms
  • Key generation: 8 mins offline, 45s online, 135MB bandwidth
  • Decryption: 7 mins offline, 44s online

• See paper for detailed evaluation
  • Levenshtein distance, SmithWaterman score, AES, Inner product
Superfast Scheme Demo
Superfast Scheme Code

www.naveed.pro/software
Take Home Message

Novel models can bring modern cryptographic technology to real applications.

www.naveed.pro/software