Removing Trust Assumptions from Functional Encryption

David Wu
December 2023

based on joint works with Cody Freitag, Rachit Garg, Susan Hohenberger, George Lu, and Brent Waters
Functional Encryption (FE)

Cipher text encrypting $x$

 learns $f_1(x)$

 learns $f_2(x)$

 learns $f_3(x)$

master secret key

$[SS10, O’N10, BSW11]$
Functional Encryption (FE)

A ciphertext encrypting $x$ should not learn more than $f_1(x)$ and $f_2(x)$.

- $f_1$ learns $f_1(x)$
- $f_2$ learns $f_2(x)$
- $f_3$ learns $f_3(x)$

Master secret key
Functional Encryption (FE)

[SS10, O’N10, BSW11]

ciphertext encrypting $x$

What if the key-issuer is compromised?
Functional Encryption (FE)

Key issuer can decrypt all ciphertexts

**Central** point of failure

Users do **not** have control over keys

What if the key-issuer is compromised?

Key issuer

- Decryption of all ciphertexts
- **Central** point of failure
- Users do not have control over keys

- Master secret key

- $f_1$ learns $f_1(x)$
- $f_2$ learns $f_2(x)$
- $f_3$ learns $f_3(x)$

[SS10, O’N10, BSW11]
Public-key encryption is decentralized

Every user generates their own key (no coordination or trust needed)

Does not support fine-grained decryption

Functional encryption is centralized

Central (trusted) authority generates individual keys

Supports fine-grained decryption capabilities

Can we get the best of both worlds?
Registration-Based Encryption (RBE)

Key issuer replaced with key curator

(Alice, pk_1)  (Bob, pk_2)  (Carol, pk_3)

sk_1  sk_2  sk_3

[GHMR18]

Special case of identity-based encryption (IBE)

Decryption keys are associated with identities

Users choose their own public/secret key and register their public key with the curator
Registration-Based Encryption (RBE) 

Users choose their own public/secret key and register their public key with the curator.

Key issuer replaced with key curator

Key curator is deterministic and transparent (no secrets)

Aggregate public keys together

Aggregated key is short: for $L$ users, $|mpk| = \text{poly}(\lambda, \log L)$

Users chooses their own public/secret key and register their public key with the curator

[GHMR18]
Registration-Based Encryption (RBE)

Key issuer replaced with key curator

Aggregate public keys together

Encrypt(mpk, Carol, message)

Master public key functions as the public key for an identity-based encryption scheme

[GHMR18]
**Registration-Based Encryption (RBE)**

Key issuer replaced with **key curator**

Aggregate public keys together

(Alice, pk)

To decrypt, users periodically retrieve a helper decryption key hsk (function of mpk and user’s public key pk₁)

\[ |hsk| = \text{poly}(\lambda, \log L) \]

**Note:** As users join, the master public key is updated, so users occasionally need to retrieve a new helper decryption key

\[ \# \text{ key updates per user} = \text{poly}(\lambda, \log L) \]
Registration-Based Encryption (RBE)

Key issuer replaced with key curator

Aggregate public keys together

- Initial constructions based on indistinguishability obfuscation or hash garbling (based on CDH, QR, LWE) – all require non-black-box use of cryptography
- High concrete efficiency costs: ciphertext is 4.5 TB for supporting 2 billion users [CES21]

*Can we construct RBE schemes that only need black-box use of cryptography?*

*Can we construct support more general policies (beyond identity-based encryption)?*
Removing Trust from Functional Encryption

Key issuer replaced with key curator

Aggregate public keys together

Users chooses their own key and register the public key (together with function $f$) with the curator

Note: $f$ could also be chosen by the key curator

$mpk = \text{poly}(\lambda, \log L)$
Removing Trust from Functional Encryption

Encrypt(mpk, x) → x

mpk is essentially a key for a functional encryption scheme

Aggregate public keys together

mpk

|mpk| = poly(\(\lambda, \log L\))
Registered Functional Encryption

Can we construct RBE schemes that only need black-box use of cryptography?  Yes!

Can we construct support more general policies (beyond identity-based encryption)?  Yes!

Registration-based encryption [GHMR18, GHMMRS19, GV20, CES21, DKLLMR23, GMRK23, ZZGQ23, FKP23]

Registered attribute-based encryption (ABE)
- Monotone Boolean formulas [HLWW23, ZZGQ23]
- Inner products [FFMMRV23, ZZGQ23]
- Arithmetic branching program [ZZGQ23]
- Boolean circuits [HLWW23, FW23]

Lots of progress in this past year!

Distributed/flexible broadcast [BZ14, KMW23, FW23, GLW23]

Registered functional encryption
- Linear functions [DPY23]
- Boolean circuits [FFMMRV23, DPY23]

Underlined schemes only need black-box use of cryptography
Registered Functional Encryption

*Can we construct RBE schemes that only need black-box use of cryptography?*  
Yes!

*Can we construct support more general policies (beyond identity-based encryption)?*  
Yes!

Registration-based encryption [GHMR18, GHMMRS19, GV20, CES21, DKLLMR23, GKM23, ZZGQ23, FKP23]

Registered attribute-based encryption (ABE)
- Monotone Boolean formulas [HLW23, ZZGQ23]
- Inner products [FFMMRV23, ZZGQ23]
- Arithmetic branching program [ZZGQ23]
- Boolean circuits [HLW23, FW23]

Lots of progress in this past year!

Distributed/flexible broadcast [BZ14, KMW23, FW23, GLW23]

Registered functional encryption
- Linear functions [DPY23]
- Boolean circuits [FFMMRV23, DPY23]

Underlined schemes only need black-box use of cryptography

This talk
Attribute-Based Encryption

Policy: CS and faculty

Message

Master secret key

"faculty" "CS"

"faculty" "math"

"student" "CS"
Attribute-Based Encryption

message

policy: CS and faculty

"faculty" "CS"
"faculty" "math"
"student" "CS"

Can decrypt

master secret key

[SW05, GPSW06]
Attribute-Based Encryption

**message**

**policy:** CS and faculty

- "faculty"
- "CS"

Can decrypt

- "faculty"
- "math"

Cannot decrypt

- "student"
- "CS"

Cannot decrypt

[SW05, GPSW06]
Attribute-Based Encryption

**message**

**policy:** CS and faculty

Users cannot collude to decrypt
Users choose their own public/secret key

message associated with policy

policy: CS and faculty

ciphertexts associated with policy

"faculty" and "CS"

"student" and "CS"

Users join the system by registering their public key along with a set of attributes

Users choose their own public/secret key
**A Template for Building Registered ABE**

**Simplification:** assume that all of the users register at the same time (rather than in an online fashion)

**Slotted registered ABE:**

Let $L$ be the number of users

$$
\begin{array}{cccc}
\text{pk}_1, S_1 & \text{pk}_2, S_2 & \text{pk}_3, S_3 & \text{pk}_4, S_4 & \ldots & \text{pk}_L, S_L \\
\end{array}
$$

Each slot associated with a public key $\text{pk}$ and a set of attributes $S$

$$
|\text{mpk}| = \text{poly}(\lambda, |\mathcal{U}|, \log L) \quad \lambda: \text{security parameter}
$$

$$
|hsk_i| = \text{poly}(\lambda, |\mathcal{U}|, \log L) \quad \mathcal{U}: \text{universe of attributes}
$$

Aggregate

$\text{mpk}$

$hsk_1, \ldots, hsk_L$
A Template for Building Registered ABE

**Simplification**: assume that all of the users register at the **same** time (rather than in an online fashion)

**Slotted registered ABE:**
Let $L$ be the number of users

| $pk_1, S_1$ | $pk_2, S_2$ | $pk_3, S_3$ | $pk_4, S_4$ | ... | $pk_L, S_L$ |

Each slot associated with a **public key** $pk$ and a set of attributes $S$

Encrypt($mpk, P, m$) $\rightarrow$ ct

Encryption takes master public key and policy $P$ (**no slot**)

Decrypt($sk_i, hsk_i, ct$) $\rightarrow$ $m$

Decryption takes secret key $sk_i$ for some slot and the helper key $hsk_i$ for that slot
**Simplification:** assume that all of the users register at the *same* time (rather than in an online fashion)

**Slotted registered ABE:**

Let \( L \) be the number of users

\[
\begin{array}{cccccc}
\text{pk}_1, S_1 & \text{pk}_2, S_2 & \text{pk}_3, S_3 & \text{pk}_4, S_4 & \cdots & \text{pk}_L, S_L
\end{array}
\]

Each slot associated with a public key \( \text{pk} \) and a set of attributes \( S \)

Encrypt(\( \text{mpk}, P, m \)) → \( \text{ct} \)

Decrypt(\( \text{sk}_i, \text{hsk}_i, \text{ct} \)) → \( m \)

Main difference with registered ABE:
Aggregate takes all \( L \) keys **simultaneously**
Let $L$ be the number of users

$pk_1, S_1 \quad pk_2, S_2 \quad pk_3, S_3 \quad pk_4, S_4 \quad \ldots \quad pk_L, S_L$

Aggregated

$mpk$

$hsk_1, \ldots, hsk_L$

Slotted scheme does *not* support online registration

**Solution:** use "powers-of-two" approach (like [GHMR18])

[HLWW23]
Slotted Registered ABE to Registered ABE

Solution: use “powers-of-two” approach (like [GHMR18])

To support $L = 2^\ell$ users: maintain $\ell$ slotted schemes

$$
2^0 = 1
\quad \square
$$

$$
2^1 = 2
\quad \square \quad \square
$$

$$
2^2 = 4
\quad \square \quad \square \quad \square \quad \square
$$

$$
\vdots
$$

$$
2^\ell = L
\quad \square \quad \square \quad \square \quad \square \quad \square \quad \square \quad \square \quad \square \quad \square \quad \square \quad \square \quad \square
$$

Initially: all slots are empty

$mpk = \bot$
Slotted Registered ABE to Registered ABE

**Solution:** use "powers-of-two" approach (like [GHMR18])

To support $L = 2^\ell$ users: maintain $\ell$ slotted schemes

- $2^0 = 1$ \[ pk_1, S_1 \]
- $2^1 = 2$ \[ pk_1, S_1 \]
- $2^2 = 4$ \[ pk_1, S_1 \]
- \[ \vdots \]
- $2^\ell = L$ \[ pk_1, S_1 \]

**Initially:** all slots are empty \[ \text{mpk} = \bot \]

Add key to each scheme with available slot

HLW23
**Solution:** use “powers-of-two” approach (like [GHMR18])

To support $L = 2^\ell$ users: maintain $\ell$ slotted schemes

- $2^0 = 1$ all slots are full
- $2^1 = 2$
- $2^2 = 4$
- \[ \vdots \]
- $2^\ell = L$

**Initially:** all slots are empty

- $\text{mpk} = \bot$

- $\text{pk}_1, S_1$

- $\text{pk}_1, S_1$

- $\text{pk}_1, S_1$

- $\text{pk}_1, S_1$

- $\text{pk}_1, S_1$
Initially: all slots are empty

\( mpk = (mpk_1) \)

Solution: use “powers-of-two” approach (like [GHMR18])

To support \( L = 2^\ell \) users: maintain \( \ell \) slotted schemes

\[
\begin{align*}
2^0 = 1 & \quad \text{all slots are full} & \quad mpk_1 \\
2^1 = 2 & \\
2^2 = 4 & \\
\vdots & \\
2^\ell = L & \\
\end{align*}
\]
Slotted Registered ABE to Registered ABE

Solution: use “powers-of-two” approach (like [GHMR18])

To support $L = 2^\ell$ users: maintain $\ell$ slotted schemes

Initially: all slots are empty

\[
\text{mpk} = (\text{mpk}_1)
\]

Initially: all slots are empty

\[
\text{mpk} = (\text{mpk}_1)
\]

Add key to each scheme with available slot

\[
\text{pk}_2, S_2
\]

\[
\text{pk}_2, S_2
\]

\[
\text{pk}_2, S_2
\]
Slotted Registered ABE to Registered ABE

**Solution:** use “powers-of-two” approach (like [GHMR18])

To support $L = 2^\ell$ users: maintain $\ell$ slotted schemes

Initially: all slots are empty

$\text{mpk} = (\text{mpk}_1)$

Initially: all slots are empty

<table>
<thead>
<tr>
<th>$2^0 = 1$</th>
<th>all slots are full</th>
<th>$\text{mpk}_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{pk}_1, S_1$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$2^1 = 2$</th>
<th>all slots are full</th>
<th>$\text{mpk}_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{pk}_1, S_1$</td>
<td>$\text{pk}_2, S_2$</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$2^2 = 4$</th>
<th>$\text{pk}_1, S_1$</th>
<th>$\text{pk}_2, S_2$</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>$\vdots$</th>
<th>$\vdots$</th>
<th>$\vdots$</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>$2^\ell = L$</th>
<th>$\text{pk}_1, S_1$</th>
<th>$\text{pk}_2, S_2$</th>
</tr>
</thead>
</table>

Clear out previous schemes

$\text{pk}_2, S_2$
Solution: use “powers-of-two” approach (like [GHMR18])

To support $L = 2^\ell$ users: maintain $\ell$ slotted schemes

Initially: all slots are empty

\[
\text{mpk} = (\text{mpk}_1)
\]

Initially:

all slots are full

\[
\text{clear out previous schemes}
\]

Initially:

all slots are empty

\[
\text{mpk} = (\text{mpk}_1)
\]
Slotted Registered ABE to Registered ABE

Solution: use “powers-of-two” approach (like [GHMR18])

To support $L = 2^\ell$ users: maintain $\ell$ slotted schemes

Initially: all slots are empty

$\text{mpk} = (\text{mpk}_2)$

Initially: all slots are empty

$\text{clear out previous schemes}$

$2^0 = 1$

$2^1 = 2$

$\text{all slots are full}$

$mpk_2$

$2^2 = 4$

$pk_1, S_1 \quad pk_2, S_2$

$\vdots$

$2^\ell = L$

$pk_1, S_1 \quad pk_2, S_2$
Solution: use “powers-of-two” approach (like [GHMR18])

To support $L = 2^\ell$ users: maintain $\ell$ slotted schemes

Initially: all slots are empty

$\text{mpk} = (\text{mpk}_2)$

\[ 2^0 = 1 \quad \text{pk}_3, S_3 \]

\[ 2^1 = 2 \quad \text{pk}_1, S_1 \quad \text{pk}_2, S_2 \]

all slots are full

\[ 2^2 = 4 \quad \text{pk}_1, S_1 \quad \text{pk}_2, S_2 \quad \text{pk}_3, S_3 \]

Add key to each scheme with available slot

\[ 2^\ell = L \quad \text{pk}_1, S_1 \quad \text{pk}_2, S_2 \quad \text{pk}_3, S_3 \]
Slotted Registered ABE to Registered ABE

**Solution:** use “powers-of-two” approach (like [GHMR18])

To support $L = 2^\ell$ users: maintain $\ell$ slotted schemes

- $2^0 = 1$ all slots are full
  - $\text{pk}_3, S_3$ \rightarrow mpk_1$
- $2^1 = 2$ all slots are full
  - $\text{pk}_1, S_1$ \rightarrow mpk_2$
  - $\text{pk}_2, S_2$
- $2^2 = 4$
  - $\text{pk}_1, S_1$ \rightarrow mpk_2$
  - $\text{pk}_2, S_2$
  - $\text{pk}_3, S_3$

$\vdots$

- $2^\ell = L$
  - $\text{pk}_1, S_1$ \rightarrow mpk_2$
  - $\text{pk}_2, S_2$
  - $\text{pk}_3, S_3$
  - $\text{pk}_4, S_4$
  - $\text{pk}_5, S_5$

**Initially:** all slots are empty
- mpk = (mpk_2)

Add key to each scheme with available slot
- $\text{pk}_3, S_3$
- $\text{pk}_4, S_4$
- $\text{pk}_5, S_5$

[HLW23]
Solution: use “powers-of-two” approach (like [GHMR18])

To support $L = 2^\ell$ users: maintain $\ell$ slotted schemes

Initially: all slots are empty

$mpk = (mpk_1, mpk_2)$

Initially:

- $2^0 = 1$: all slots are full
  - $pk_3, S_3$ → $mpk_1$

- $2^1 = 2$: all slots are full
  - $pk_1, S_1$ → $mpk_2$

- $2^2 = 4$: available slots
  - $pk_1, S_1$ → $mpk_2$
  - $pk_2, S_2$ → $mpk_2$
  - $pk_3, S_3$ → $mpk_2$

- $\vdots$

- $2^\ell = L$: available slots
  - $pk_1, S_1$ → $mpk_2$
  - $pk_2, S_2$ → $mpk_2$
  - $pk_3, S_3$ → $mpk_2$
  - $\cdots$
Solution: use “powers-of-two” approach (like [GHMR18])

To support $L = 2^\ell$ users: maintain $\ell$ slotted schemes

Initially: all slots are empty

$\text{mpk} = (\text{mpk}_1, \text{mpk}_2)$

- $2^0 = 1$
  - $\text{pk}_3, S_3$
  - all slots are full
  - $\text{mpk}_1$

- $2^1 = 2$
  - $\text{pk}_1, S_1$
  - $\text{pk}_2, S_2$
  - all slots are full
  - $\text{mpk}_2$

- $2^2 = 4$
  - $\text{pk}_1, S_1$
  - $\text{pk}_2, S_2$
  - $\text{pk}_3, S_3$
  - $\text{pk}_4, S_4$

- $2^\ell = L$
  - $\text{pk}_1, S_1$
  - $\text{pk}_2, S_2$
  - $\text{pk}_3, S_3$
  - $\text{pk}_4, S_4$

Add key to each scheme with available slot.
Slotted Registered ABE to Registered ABE

**Solution:** use “powers-of-two” approach (like [GHMR18])

To support $L = 2^\ell$ users: maintain $\ell$ slotted schemes

Initially: all slots are empty

$\text{mpk} = (\text{mpk}_1, \text{mpk}_2)$

Initially: all slots are empty

$\text{mpk} = (\text{mpk}_1, \text{mpk}_2)$

<table>
<thead>
<tr>
<th>$2^0 = 1$</th>
<th>pk$_3$, $S_3$</th>
<th>all slots are full</th>
<th>mpk$_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^1 = 2$</td>
<td>pk$_1$, $S_1$</td>
<td>pk$_2$, $S_2$</td>
<td>all slots are full</td>
</tr>
<tr>
<td>$2^2 = 4$</td>
<td>pk$_1$, $S_1$</td>
<td>pk$_2$, $S_2$</td>
<td>pk$_3$, $S_3$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$2^\ell = L$</td>
<td>pk$_1$, $S_1$</td>
<td>pk$_2$, $S_2$</td>
<td>pk$_3$, $S_3$</td>
</tr>
</tbody>
</table>
**Solution:** use “powers-of-two” approach (like [GHMR18])

To support $L = 2^\ell$ users: maintain $\ell$ slotted schemes

\[
\begin{align*}
2^0 &= 1 & \text{all slots are empty} \\
2^1 &= 2 & \text{clear out previous schemes} \\
2^2 &= 4 & \text{all slots are full} \\
&\vdots \\
2^\ell &= L & \text{clear out previous schemes}
\end{align*}
\]

Initially: all slots are empty
\[mpk = (mpk_1, mpk_2)\]
Solution: use “powers-of-two” approach (like [GHMR18])

To support $L = 2^\ell$ users: maintain $\ell$ slotted schemes

Initially: all slots are empty

Initially: all slots are empty

$$\text{mpk} = (\text{mpk}_3)$$

Initially: all slots are empty

Initially: all slots are empty
**Solution:** use “powers-of-two” approach (like [GHMR18])

To support $L = 2^\ell$ users: maintain $\ell$ slotted schemes

$2^0 = 1$

$2^1 = 2$

$2^2 = 4$

$\vdots$

$2^\ell = L$

Initially: all slots are empty

$\text{mpk} = (\text{mpk}_3)$

Ciphertext is an encryption to each public key

$log L$ overhead
**Slotted Registered ABE to Registered ABE**

**Solution:** use “powers-of-two” approach (like [GHMR18])

To support $L = 2^{\ell}$ users: maintain $\ell$ slotted schemes

<table>
<thead>
<tr>
<th>$2^0 = 1$</th>
<th>$2^1 = 2$</th>
<th>$2^2 = 4$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$\text{pk}_1, S_1$ $\text{pk}_2, S_2$ $\text{pk}_3, S_3$ $\text{pk}_4, S_4$</td>
</tr>
</tbody>
</table>

Initially: all slots are empty

$\text{mpk} = (\text{mpk}_3)$

Update needed whenever user’s key moves from scheme $i$ to scheme $j > i$

At most $\ell = \log L$ updates
Constructing Slotted Registered ABE

Construction will rely on (composite-order) pairing groups \((\mathbb{G}, \mathbb{G}_T)\)

Pairing is an **efficiently-computable** bilinear map \(e: \mathbb{G} \rightarrow \mathbb{G}_T\) from \(\mathbb{G}\) to \(\mathbb{G}_T\):

\[
e(g^x, g^y) = e(g, g)^{xy}
\]

*Multiplies exponents in the **target group***
Outline of Slotted Registered ABE

Scheme will rely on a **structured** common reference string (CRS)

**Slot components:** each slot $i \in [L]$ will have a set of associated group elements (denoted $A_i$)

\[
\begin{array}{cccccc}
A_1 & A_2 & A_3 & A_4 & \cdots & A_L \\
\end{array}
\]

**Attribute components:** each attribute $w \in \mathcal{U}$ will have a group element $U_w$

User’s individual public/secret key is an ElGamal key-pair

\[
\text{sk} = r, \quad \text{pk} = g^r
\]

Aggregated public key is just the product of every user’s public key:

\[
\text{mpk} = \prod_{i \in [L]} g^{r_i}
\]
Outline of Slotted Registered ABE

Scheme will rely on a **structured** common reference string (CRS)

**Slot components:** each slot \( i \in [L] \) will have a set of associated group elements (denoted \( A_i \))

\[
\begin{array}{cccccc}
A_1 & A_2 & A_3 & A_4 & \cdots & A_L
\end{array}
\]

**Attribute components:** each attribute \( w \in \mathcal{U} \) will have a group element \( U_w \)

Decryption enforces the following two requirements:

**Slot requirement:** Decrypter know a secret key associated with the public key for some slot \( i^* \)

**Attribute requirement:** Attributes associated with slot \( i^* \) satisfy the decryption policy

In the construction, message is “blinded” by \( v_1 v_2 \), where \( v_1 \) can be computed with knowledge of a secret key associated with a slot \( i^* \) and \( v_2 \) can be computed if the attributes for slot \( i^* \) satisfy the policy
Outline of Slotted Registered ABE

Scheme will rely on a **structured** common reference string (CRS)

**Slot components:** each slot $i \in [L]$ will have a set of associated group elements (denoted $A_i$)

| $A_1$ | $A_2$ | $A_3$ | $A_4$ | $\cdots$ | $A_L$ |

**Attribute components:** each attribute $w \in \mathcal{U}$ will have a group element $U_w$

Having two requirements:

1. Decrypter knows a secret key associated with the public key for some slot $i^*$
2. Attributes associated with slot $i^*$ satisfy the decryption policy

Need to be careful to defend against collusions

[see paper for details]

In the construction, message is “blinded” by $v_1 v_2$, where $v_1$ can be computed with knowledge of a secret key associated with a slot $i^*$ and $v_2$ can be computed if the attributes for slot $i^*$ satisfy the policy
Registered ABE Summary

Key issuer replaced with key curator

Users chooses their own public/secret key

Aggregated key

mpk_{L}

"faculty" "CS"

pk_1

hsk_1

sk_1

• New approach to constructing RBE-type of primitives
• Registered ABE scheme (for Boolean formulas) only makes black-box use of cryptography
• Construction will need a (trusted) common reference string (CRS) and supports bounded number of users
Registered ABE is a useful building block for other trustless cryptographic systems.

Suppose we want to encrypt a message to \{pk_1, pk_3, pk_4\}.

**Public-key encryption:** ciphertext size grows with the size of the set.

**Broadcast encryption:** achieve sublinear ciphertext size, but requires central authority.
An Application to Broadcast Encryption

Distributed broadcast encryption [BZ14]

Each user chooses its own public key, and each key has a unique index

Encrypt\( (pp, \{pk_i\}_{i \in S}, m) \rightarrow ct \)

Can encrypt a message \( m \) to any set of public keys

Efficiency: \( |ct| = |m| + \text{poly}(\lambda, \log|S|) \)

Decrypt\( (pp, \{pk_i\}_{i \in S}, sk, ct) \rightarrow m \)

Any secret key associated with broadcast set can decrypt

Decryption does require knowledge of public keys in broadcast set
Consider a registered ABE scheme with a single dummy attribute $x$

Public key for an index $i$ is a key for slot $i$ with attribute $x$.

Suppose we want to encrypt to a set $S = \{2, 3, 5\}$.

Aggregate public keys using slotted registered ABE scheme.

Encrypt with respect to $mpk$ to policy $P$ that accepts $x$.

Encrypt($mpk, x, P$)
Consider a registered ABE scheme with a single dummy attribute $x$

Public key for an index $i$ is a key for slot $i$ with attribute $x$

Suppose we want to encrypt to a set $S = \{2, 3, 5\}$

Correctness: If $i \in S$, then $(i, \text{pk}_i, x)$ was aggregated in mpk so decryption is possible using $\text{sk}_i$

Security: If $i \notin S$, then $(i, \text{pk}_i, x)$ was not aggregated in mpk so we can appeal to security of registered ABE
Distributed Broadcast from Slotted Registered ABE

Consider a registered ABE scheme with a single dummy attribute $x$

Public key for an index $i$ is a key for slot $i$ with attribute $x$

Suppose we want to encrypt to a set $S = \{2, 3, 5\}$

Registered ABE [HLW23] + compiler $\Rightarrow$ distributed broadcast encryption from composite-order pairing groups

Concurrent work [KMW23]: show how to adapt a centralized broadcast encryption scheme into a distributed one from prime-order pairing groups
Flexible Broadcast Encryption

Distributed broadcast encryption still requires some coordination.

Users have to generate public keys for distinct slots (for correctness), so public-key directory needs to be centralized.

- $(1, \text{pk}_1)$
- $(2, \text{pk}_2)$
- $(3, \text{pk}_3)$
- $(4, \text{pk}_4)$
- $(5, \text{pk}_5)$
Flexible Broadcast Encryption

Distributed broadcast encryption still requires *some* coordination.

Users have to generate public keys for **distinct** slots (for correctness), so public-key directory needs to be **centralized**.

**Flexible broadcast encryption:** no notion of slots, can encrypt to an *arbitrary* set of public keys.
Flexible Broadcast Encryption

Distributed broadcast encryption still requires *some* coordination

Encrypt(pp, \{pk_i\}_{i \in S}, m) \rightarrow ct

Can encrypt a message \(m\) to any set of public keys

**Efficiency:** \(|ct| = |m| + \text{poly}(\lambda, \log|S|)\)

Decrypt(pp, \{pk_i\}_{i \in S}, sk, ct) \rightarrow m

Any secret key associated with broadcast set can decrypt

Decryption does require knowledge of public keys in broadcast set
Flexible Broadcast Encryption

Distributed broadcast encryption still requires some coordination

Encrypt(pp, \{pk_i\}_{i \in S}, m) \rightarrow ct

Can encrypt a message \(m\) to any set of public keys

**Efficiency:** \(|ct| = |m| + \text{poly}(\lambda, \log|S|)\)

Decrypt(pp, \{pk_i\}_{i \in S}, sk, ct) \rightarrow m

Any secret key associated with broadcast set can decrypt

Decryption does requires knowledge of public keys in broadcast set

[FWW23, GLW23]: **distributed** broadcast encryption \(\Rightarrow\) **flexible** broadcast encryption
Removing Trust from Functional Encryption

Encrypt(mpk, x) → x

mpk is essentially a key for a functional encryption scheme

Aggregate public keys together

(Alice, \( f_1 \))

(Bob, \( f_2 \))

(Carol, \( f_3 \))

\( f_1(x) \)

\( f_2(x) \)

\( f_3(x) \)

Goal: Support capabilities of functional encryption without a trusted authority
Open Problems

Schemes with short CRS or unstructured CRS without non-black-box use of cryptography

Existing constructions have long structured CRS (typically quadratic in the number of users)

Lattice-based constructions of registered FE (and special cases of FE)

Registration-based encryption known from LWE [DKLLMR23]
Registered ABE for circuits known from evasive LWE (via witness encryption) [FWW23]

Key revocation and verifiability

Defending against possibly malicious adversaries

Improve concrete efficiency for registered FE schemes

Current bottlenecks include large CRS and large public keys

Thank you!
References


References


<table>
<thead>
<tr>
<th>Reference</th>
<th>Author(s)</th>
<th>Title</th>
<th>Conference/Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>[SW05]</td>
<td>Amit Sahai and Brent Waters</td>
<td>Fuzzy Identity-Based Encryption</td>
<td>EUROCRYPT 2005.</td>
</tr>
</tbody>
</table>