Outsider-Anonymous Broadcast Encryption
with Sublinear Ciphertexts

Nelly Fazio[1,2]   Irippuge Milinda Perera[2]

[1]The City College of CUNY
fazio@cs.ccny.cuny.edu

[2]The Graduate Center of CUNY
{nfazio,iperera}@gc.cuny.edu

NYU Crypto Reading Group

Outline

Outline

Outline

Outline

Outline

Outline

Broadcast Encryption (BE) 101

- Originally proposed by Fiat and Naor (1993)
- Secure broadcast of messages to an arbitrary subset of users
- Settings:
  1. Public-key / Private-key
  2. Stateless / Stateful
     - Are users required to update private keys?
  3. Fully collusion-resistant / t-collusion-resistant
     - What's the upper bound for coalition size?
- Selected work:
  [GSW00,NNL01,HS02,DF02,DF03,DFKY03,
  DFLY04,BGW05,BW06,GW09,BBW06, . . .]

BE – The Setting

- Algorithms:
  - (PK, MSK) $\leftarrow$ Setup$(1^\lambda, N)$
    N – total number of users in the system
  - $sk_i \leftarrow$ KeyGen(PK, MSK, i)
    i – user index in $\mathcal{U} = \{1, \ldots, N\}$
  - $c \leftarrow$ Encrypt(PK, S, m)
    S – set of recipients ($S \subseteq \mathcal{U}$)
  - $m / \perp :=$ Decrypt(PK, S, sk_i, c)
- Correctness:
  - For all $S \subseteq \mathcal{U}$, $i \in S$ and $m \in \mathcal{MSP}$,
    if $sk_i \leftarrow$ KeyGen(PK, MSK, i), then
    Decrypt(PK, S, sk_i, Encrypt(PK, S, m)) $= m$

BE – The Setting

- Algorithms:
  - $(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, N)$
    N – total number of users in the system
  - $sk_i \leftarrow \mathsf{KeyGen}(\mathsf{PK}, \mathsf{MSK}, i)$
    i – user index in $\mathcal{U} = \{1, \ldots, N\}$
  - $c \leftarrow \mathsf{Encrypt}(\mathsf{PK}, S, m)$
    S – set of recipients ($S \subseteq \mathcal{U}$)
  - $m / \perp := \mathsf{Decrypt}(\mathsf{PK}, S, sk_i, c)$
- Correctness:
  - For all $S \subseteq \mathcal{U}$, $i \in S$ and $m \in \mathcal{MSP}$,
    if $sk_i \leftarrow \mathsf{KeyGen}(\mathsf{PK}, \mathsf{MSK}, i)$, then
    $\mathsf{Decrypt}(\mathsf{PK}, S, sk_i, \mathsf{Encrypt}(\mathsf{PK}, S, m)) = m$

BE – The Setting

- Algorithms:
  - $(\text{PK}, \text{MSK}) \leftarrow \text{Setup}(1^\lambda, N)$
    N – total number of users in the system
  - $sk_i \leftarrow \text{KeyGen}(\text{PK}, \text{MSK}, i)$
    i – user index in $\mathcal{U} = \{1, \ldots, N\}$
  - $c \leftarrow \text{Encrypt}(\text{PK}, S, m)$
    S – set of recipients ($S \subseteq \mathcal{U}$)
  - $m/\perp := \text{Decrypt}(\text{PK}, S, sk_i, c)$
- Correctness:
  - For all $S \subseteq \mathcal{U}$, $i \in S$ and $m \in \mathcal{MSP}$,
    if $sk_i \leftarrow \text{KeyGen}(\text{PK}, \text{MSK}, i)$, then
    $\text{Decrypt}(\text{PK}, S, sk_i, \text{Encrypt}(\text{PK}, S, m)) = m$

BE – The Setting

- Algorithms:
  - $(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, N)$
    N – total number of users in the system
  - $sk_i \leftarrow \mathsf{KeyGen}(\mathsf{PK}, \mathsf{MSK}, i)$
    i – user index in $\mathcal{U} = \{1, \ldots, N\}$
  - $c \leftarrow \mathsf{Encrypt}(\mathsf{PK}, S, m)$
    S – set of recipients ($S \subseteq \mathcal{U}$)
  - $m / \perp := \mathsf{Decrypt}(\mathsf{PK}, S, sk_i, c)$
- Correctness:
  - For all $S \subseteq \mathcal{U}$, $i \in S$ and $m \in \mathcal{MSP}$,
    if $sk_i \leftarrow \mathsf{KeyGen}(\mathsf{PK}, \mathsf{MSK}, i)$, then
    $\mathsf{Decrypt}(\mathsf{PK}, S, sk_i, \mathsf{Encrypt}(\mathsf{PK}, S, m)) = m$

BE – The Setting

- Algorithms:
  - $(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, N)$
    N – total number of users in the system
  - $sk_i \leftarrow \mathsf{KeyGen}(\mathsf{PK}, \mathsf{MSK}, i)$
    i – user index in $\mathcal{U} = \{1, \ldots, N\}$
  - $c \leftarrow \mathsf{Encrypt}(\mathsf{PK}, S, m)$
    S – set of recipients ($S \subseteq \mathcal{U}$)
  - $m / \perp := \mathsf{Decrypt}(\mathsf{PK}, S, sk_i, c)$
- Correctness:
  - For all $S \subseteq \mathcal{U}$, $i \in S$ and $m \in \mathcal{MSP}$,
    if $sk_i \leftarrow \mathsf{KeyGen}(\mathsf{PK}, \mathsf{MSK}, i)$, then
    $\mathsf{Decrypt}(\mathsf{PK}, S, sk_i, \mathsf{Encrypt}(\mathsf{PK}, S, m)) = m$

BE – The Setting

- Algorithms:
  - $(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, N)$
    N – total number of users in the system
  - $sk_i \leftarrow \mathsf{KeyGen}(\mathsf{PK}, \mathsf{MSK}, i)$
    i – user index in $\mathcal{U} = \{1, \ldots, N\}$
  - $c \leftarrow \mathsf{Encrypt}(\mathsf{PK}, S, m)$
    S – set of recipients ($S \subseteq \mathcal{U}$)
  - $m / \perp := \mathsf{Decrypt}(\mathsf{PK}, S, sk_i, c)$
- Correctness:
  - For all $S \subseteq \mathcal{U}$, $i \in S$ and $m \in \mathcal{MSP}$,
    if $sk_i \leftarrow \mathsf{KeyGen}(\mathsf{PK}, \mathsf{MSK}, i)$, then
    $\mathsf{Decrypt}(\mathsf{PK}, S, sk_i, \mathsf{Encrypt}(\mathsf{PK}, S, m)) = m$

BE – Drawbacks

- Previous work mostly aimed at ever more efficient solutions
  - Ciphertext length
  - Public/Private key length
  - Encryption/Decryption running time
- Privacy concerns of the recipients largely overlooked
  - Set of recipients transmitted as part of the ciphertext

BE – Drawbacks

- Previous work mostly aimed at ever more efficient solutions
  - Ciphertext length
  - Public/Private key length
  - Encryption/Decryption running time
- Privacy concerns of the recipients largely overlooked
  - Set of recipients transmitted as part of the ciphertext
  - What if identities of recipients also sensitive?

Outline

Private Broadcast Encryption

- Originally proposed by Barth *et al.* (2006)
- Goal: completely hide the identities of the recipients
- Idea: have a ciphertext component for each recipient

Simplified CPA Construction (Sketch) from BBW06

Given a Robust and Anonymous PKE $\Pi' = (\text{Setup}', \text{KeyGen}', \text{Enc}', \text{Dec}')$,

$\text{Setup}(1^\lambda, N)$:
1. For each $i \in \{1, \ldots, N\}$:
   Generate (pk_i, sk_i)
2. Save pk_i's in PK, sk_i's in MSK

$\text{KeyGen}(\text{PK}, \text{MSK}, i)$:
1. Get sk_i from MSK and output sk_i

$\text{Encrypt}(\text{PK}, S, m)$:
1. For each $i \in S$:
   Compute $c_i \leftarrow \text{Enc}'(pk_i, m)$
2. Output $c = \left(c_{\pi(1)}, \ldots, c_{\pi(l)}\right)$

$\text{Decrypt}(\text{PK}, sk_i, c)$:
1. Parse c as $(c_1, \ldots, c_l)$
2. For each $j \in \{1, \ldots, l\}$:
   Compute $m = \text{Dec}'(sk_i, c_j)$
   If $m \neq \perp$, return m
3. Return $\perp$

Simplified CPA Construction (Sketch) from BBW06

Given a Robust and Anonymous PKE $\Pi' = (\text{Setup}', \text{KeyGen}', \text{Enc}', \text{Dec}')$,

Setup$(1^\lambda, N)$:
1. For each $i \in \{1, \ldots, N\}$: Generate (pk_i, sk_i)
2. Save pk_i's in PK, sk_i's in MSK

KeyGen(PK, MSK, i):
1. Get sk_i from MSK and output sk_i

Encrypt(PK, S, m):
1. For each $i \in S$: Compute $c_i \leftarrow \text{Enc}'(pk_i, m)$
2. Output $c = \left(c_{\pi(1)}, \ldots, c_{\pi(l)}\right)$

Decrypt(PK, sk_i, c):
1. Parse c as $(c_1, \ldots, c_l)$
2. For each $j \in \{1, \ldots, l\}$: Compute $m = \text{Dec}'(sk_i, c_j)$; If $m \neq \bot$, return m
3. Return $\bot$

Simplified CPA Construction (Sketch) from BBW06

Given a Robust and Anonymous PKE $\Pi' = (\text{Setup}', \text{KeyGen}', \text{Enc}', \text{Dec}')$,

Setup$(1^\lambda, N)$:
1. For each $i \in \{1, \ldots, N\}$:
   Generate (pk_i, sk_i)
2. Save pk_i's in PK, sk_i's in MSK

KeyGen$(\text{PK}, \text{MSK}, i)$:
1. Get sk_i from MSK and output sk_i

Encrypt(PK, S, m):
1. For each $i \in S$:
   Compute $c_i \leftarrow \text{Enc}'(pk_i, m)$
2. Output $c = \left(c_{\pi(1)}, \ldots, c_{\pi(l)}\right)$

Decrypt(PK, sk_i, c):
1. Parse c as $(c_1, \ldots, c_l)$
2. For each $j \in \{1, \ldots, l\}$:
   Compute $m = \text{Dec}'(sk_i, c_j)$
   If $m \neq \perp$, return m
3. Return $\perp$

Simplified CPA Construction (Sketch) from BBW06

Given a Robust and Anonymous PKE $\Pi' = (\text{Setup}', \text{KeyGen}', \text{Enc}', \text{Dec}')$,

Setup$(1^\lambda, N)$:
1. For each $i \in \{1, \ldots, N\}$:
   Generate (pk_i, sk_i)
2. Save pk_i's in PK, sk_i's in MSK

KeyGen(PK, MSK, i):
1. Get sk_i from MSK and output sk_i

Encrypt(PK, S, m):
1. For each $i \in S$:
   Compute $c_i \leftarrow \text{Enc}'(pk_i, m)$
2. Output $c = \left(c_{\pi(1)}, \ldots, c_{\pi(l)}\right)$

Decrypt(PK, sk_i, c):
1. Parse c as $(c_1, \ldots, c_l)$
2. For each $j \in \{1, \ldots, l\}$:
   Compute $m = \text{Dec}'(sk_i, c_j)$
   If $m \neq \perp$, return m
3. Return $\perp$

Simplified CPA Construction (Sketch) from BBW06

Given a Robust and Anonymous PKE $\Pi' = (\mathsf{Setup}', \mathsf{KeyGen}', \mathsf{Enc}', \mathsf{Dec}')$,

$\mathsf{Setup}(1^\lambda, N)$:

1. For each $i \in \{1, \ldots, N\}$:
   Generate (pk_i, sk_i)
2. Save pk_i's in PK, sk_i's in MSK

$\mathsf{KeyGen}(\mathsf{PK}, \mathsf{MSK}, i)$:

1. Get sk_i from MSK and output sk_i

$\mathsf{Encrypt}(\mathsf{PK}, S, m)$:

1. For each $i \in S$:
   Compute $c_i \leftarrow \mathsf{Enc}'(pk_i, m)$
2. Output $c = \left(c_{\pi(1)}, \ldots, c_{\pi(l)}\right)$

$\mathsf{Decrypt}(\mathsf{PK}, sk_i, c)$:

1. Parse c as $(c_1, \ldots, c_l)$
2. For each $j \in \{1, \ldots, l\}$:
   Compute $m = \mathsf{Dec}'(sk_i, c_j)$
   If $m \neq \perp$, return m
3. Return $\perp$

Actual Constructions in BBW06

1. CCA Construction
   Add a signature to the ciphertext in CPA cons.
2. Enhanced CCA Construction
   Add tags to ciphertext components in CPA cons.

Drawbacks of BBW06

- Ciphertext length linear in # of recipients
- Security model is against a *static* adversary
- Security of enhanced construction based on the ROM
  - Libert *et al.* (2012) recently removed ROM from [BBW06]

Drawbacks of BBW06

- Ciphertext length linear in # of recipients
- Security model is against a *static* adversary
- Security of enhanced construction based on the ROM
  - Libert *et al.* (2012) recently removed ROM from [BBW06]
- Can we achieve better performance?
  (*i.e.*, sub-linear ciphertext length)

Outline

Outsider-Anonymous Broadcast Encryption (oABE)

- Outsider-Anonymity: A new notion of receiver privacy that enables shorter ciphertext (in the standard model)
  - Recipients' identities hidden from outsiders ...
  - ...but individual recipients might learn who else is getting msg
- Idea: Trade some degree of anonymity for better efficiency

oABE – The Setting

- Algorithms:
  - $(\text{PK}, \text{MSK}) \leftarrow \text{Setup}(1^\lambda, N)$
  - $sk_i \leftarrow \text{KeyGen}(\text{PK}, \text{MSK}, i)$
  - $c \leftarrow \text{Encrypt}(\text{PK}, S, m)$
  - $m/\bot := \text{Decrypt}(\text{PK}, sk_i, c)$
    S no longer provided to Decrypt
- Correctness:
  - For all $S \subseteq \mathcal{U}$, $i \in S$ and $m \in \mathcal{MSP}$,
    if $sk_i \leftarrow \text{KeyGen}(\text{PK}, \text{MSK}, i)$, then
    $\text{Decrypt}(\text{PK}, sk_i, \text{Encrypt}(\text{PK}, S, m)) = m$

oABE – The Setting

- Algorithms:
  - $(\text{PK}, \text{MSK}) \leftarrow \text{Setup}(1^\lambda, N)$
  - $sk_i \leftarrow \text{KeyGen}(\text{PK}, \text{MSK}, i)$
  - $c \leftarrow \text{Encrypt}(\text{PK}, S, m)$
  - $m/\perp := \text{Decrypt}(\text{PK}, sk_i, c)$

    S no longer provided to Decrypt
- Correctness:
  - For all $S \subseteq \mathcal{U}$, $i \in S$ and $m \in \mathcal{MSP}$, if $sk_i \leftarrow \text{KeyGen}(\text{PK}, \text{MSK}, i)$, then $\text{Decrypt}(\text{PK}, sk_i, \text{Encrypt}(\text{PK}, S, m)) = m$

oABE – The Setting

- Algorithms:
  - $(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, N)$
  - $sk_i \leftarrow \mathsf{KeyGen}(\mathsf{PK}, \mathsf{MSK}, i)$
  - $c \leftarrow \mathsf{Encrypt}(\mathsf{PK}, S, m)$
  - $m/\perp := \mathsf{Decrypt}(\mathsf{PK}, sk_i, c)$

    S no longer provided to Decrypt
- Correctness:
  - For all $S \subseteq \mathcal{U}$, $i \in S$ and $m \in \mathcal{MSP}$, if $sk_i \leftarrow \mathsf{KeyGen}(\mathsf{PK}, \mathsf{MSK}, i)$, then $\mathsf{Decrypt}(\mathsf{PK}, sk_i, \mathsf{Encrypt}(\mathsf{PK}, S, m)) = m$

oABE – The Security Model

The oABE-IND-CCA Game

$\mathcal{C}$ $\mathcal{A}$

1. Setup
2. Oracle Phase 1
3. Challenge
4. Oracle Phase 2
5. Guess

oABE – The Security Model

❶ Setup

oABE – The Security Model

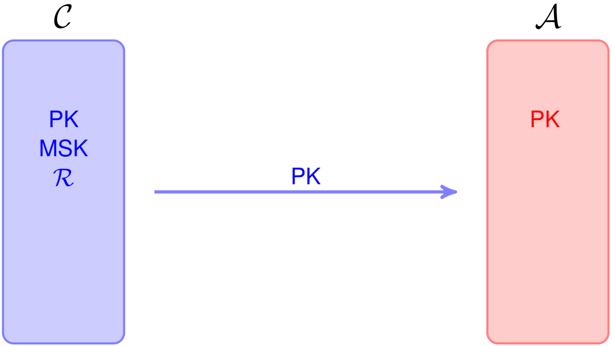
❶ Setup

$(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, N)$

$\mathcal{R} \leftarrow \emptyset$

oABE – The Security Model

❶ Setup



$(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, N)$

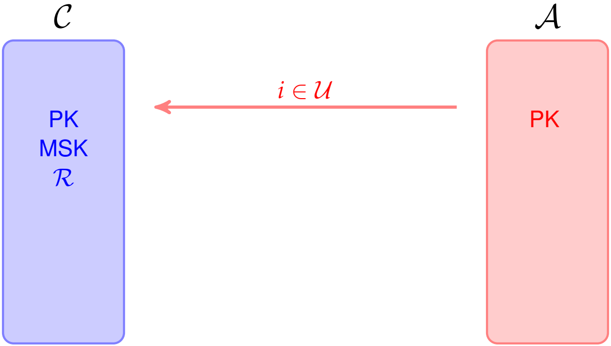
$\mathcal{R} \leftarrow \emptyset$

oABE – The Security Model

❷ Oracle Phase 1

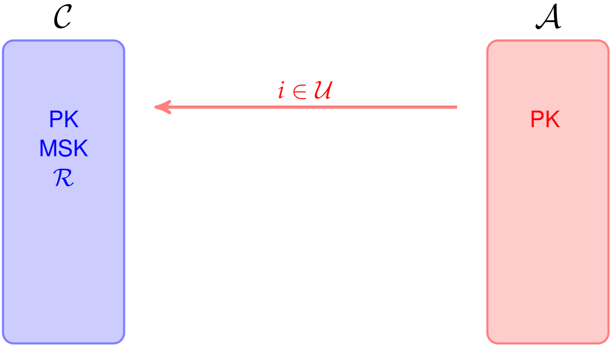
oABE – The Security Model

❷ Oracle Phase 1



oABE – The Security Model

❷ Oracle Phase 1

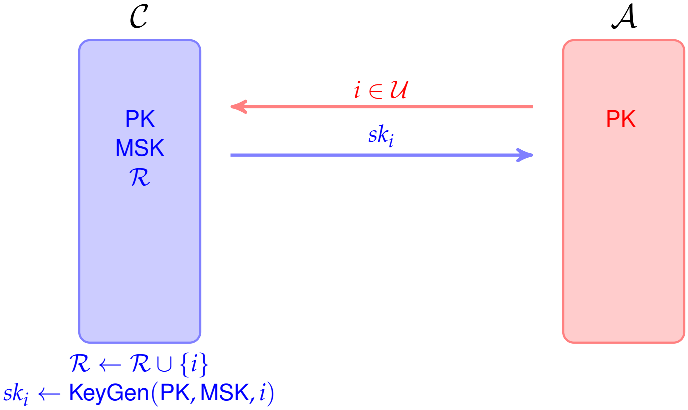


$\mathcal{R} \leftarrow \mathcal{R} \cup \{i\}$

$sk_i \leftarrow \text{KeyGen}(\text{PK}, \text{MSK}, i)$

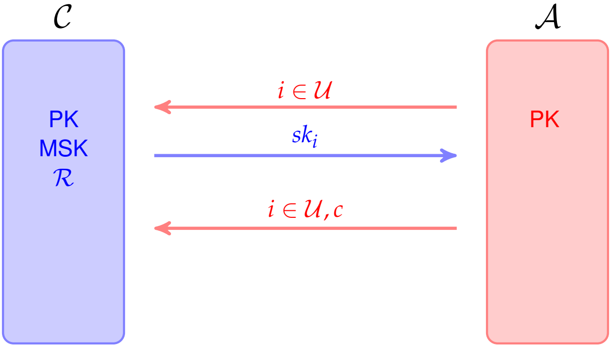
oABE – The Security Model

❷ Oracle Phase 1



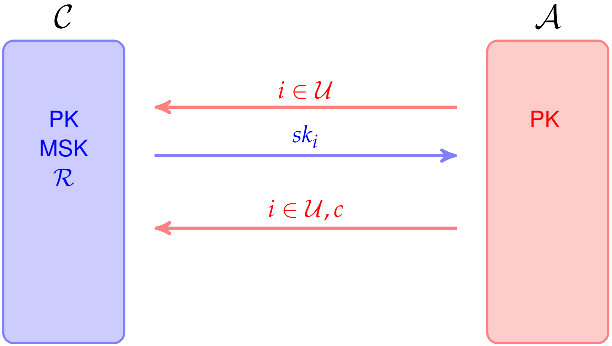
oABE – The Security Model

❷ Oracle Phase 1



oABE – The Security Model

❷ Oracle Phase 1

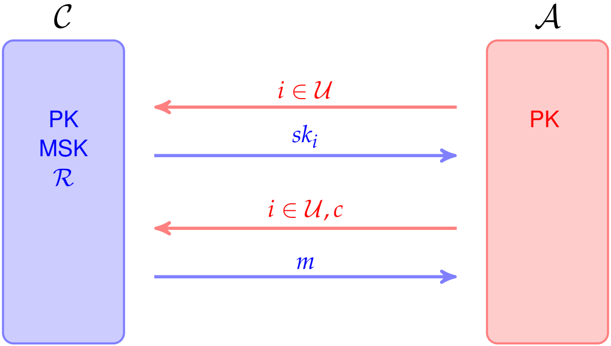


$sk_i \leftarrow \mathsf{KeyGen}(\mathsf{PK}, \mathsf{MSK}, i)$

$m := \mathsf{Decrypt}(\mathsf{PK}, sk_i, c)$

oABE – The Security Model

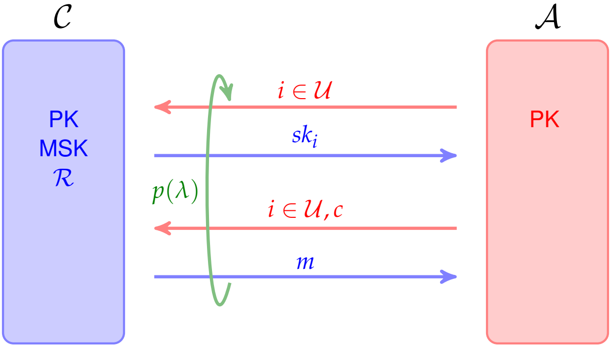
❷ Oracle Phase 1



$sk_i \leftarrow \text{KeyGen}(\text{PK}, \text{MSK}, i)$
$m := \text{Decrypt}(\text{PK}, sk_i, c)$

oABE – The Security Model

❷ Oracle Phase 1

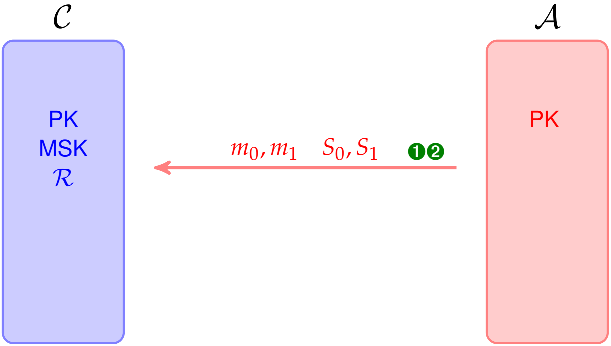


oABE – The Security Model

❸ Challenge

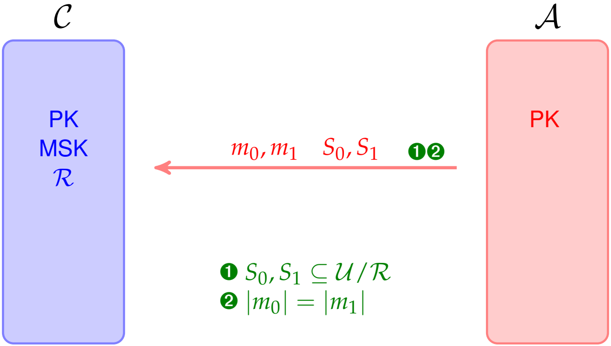
oABE – The Security Model

❸ Challenge



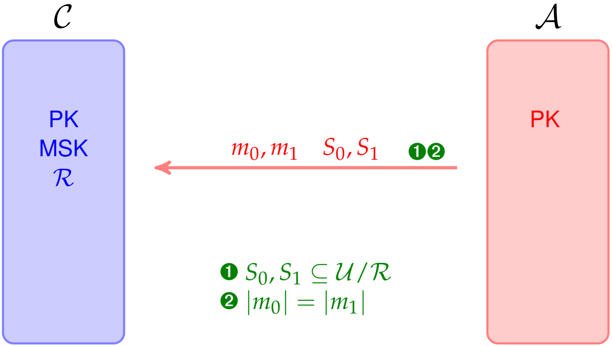
oABE – The Security Model

❸ Challenge



oABE – The Security Model

❸ Challenge

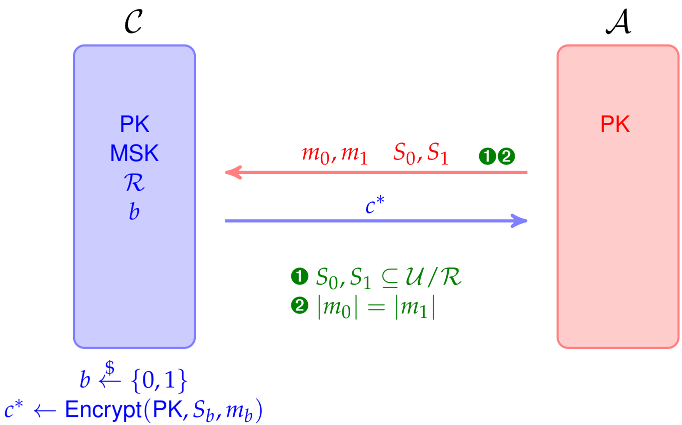


$b \xleftarrow{\$} \{0,1\}$

$c^* \leftarrow \text{Encrypt}(\text{PK}, S_b, m_b)$

oABE – The Security Model

❸ Challenge

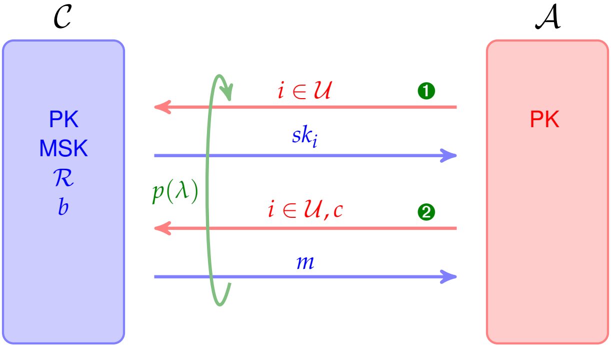


oABE – The Security Model

❹ Oracle Phase 2

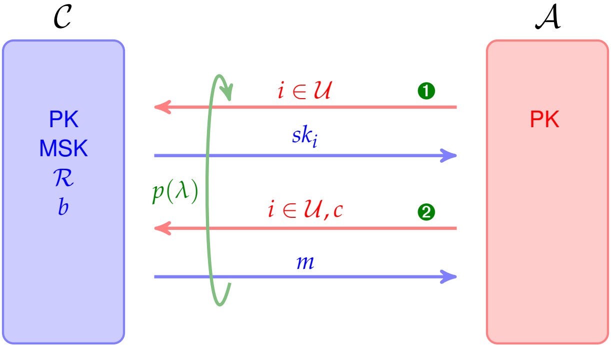
oABE – The Security Model

❹ Oracle Phase 2



oABE – The Security Model

❹ Oracle Phase 2



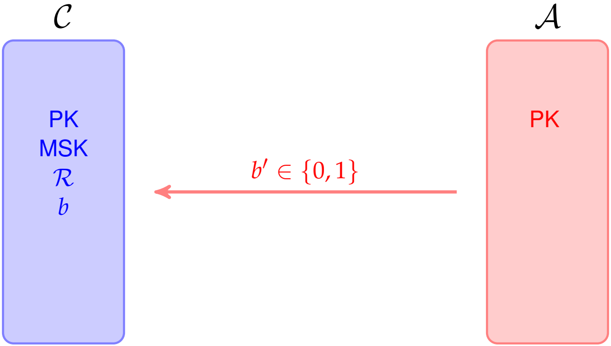
❶ $i \notin S_0 \cup S_1$
❷ if $i \in S_0 \cup S_1$, then $c \neq c^*$

oABE – The Security Model

❺ Guess

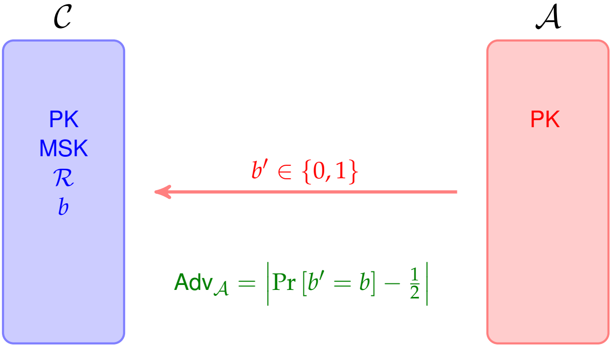
oABE – The Security Model

❺ Guess



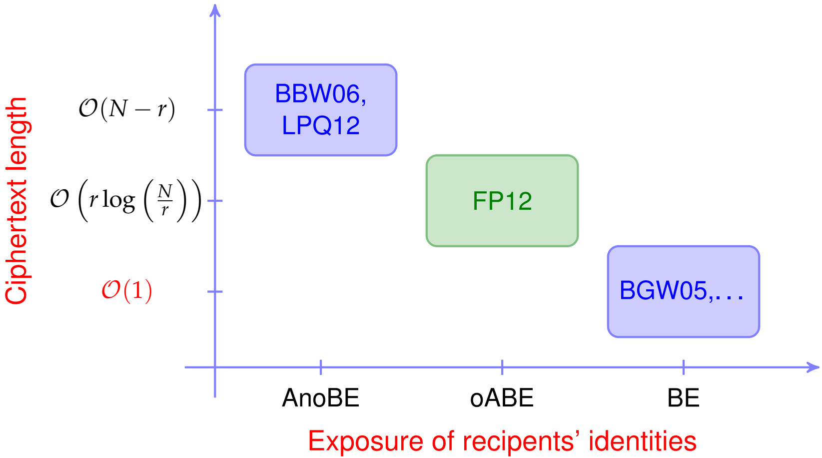
oABE – The Security Model

❺ Guess



$$\text{Adv}_{\mathcal{A}} = \left| \Pr\left[b' = b\right] - \frac{1}{2} \right|$$

Where oABE Stands



Outline

Anonymous Identity-Based Encryption (AIBE)

- Identity-Based Encryption (IBE)
  - Originally proposed by Shamir (1984)
  - A Public-key encryption scheme
  - The user public key is an arbitrary bit-string
  - Implementations - [BF01,BB04,BGH07,Waters09, . . .]
- Anonymous Identity-Based Encryption (AIBE)
  - Ciphertext hides the identity under which it is encrypted
  - Implementations - [BW06,Gentry06, . . .]

Anonymous Identity-Based Encryption (AIBE)

- Identity-Based Encryption (IBE)
  - Originally proposed by Shamir (1984)
  - A Public-key encryption scheme
  - The user public key is an arbitrary bit-string
  - Implementations - [BF01,BB04,BGH07,Waters09, . . .]
- Anonymous Identity-Based Encryption (AIBE)
  - Ciphertext hides the identity under which it is encrypted
  - Implementations - [BW06,Gentry06, . . .]

AIBE – The Setting

- Algorithms:
  - $(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda)$
  - $sk_{\mathsf{ID}} \leftarrow \mathsf{Ext}(\mathsf{PK}, \mathsf{MSK}, \mathsf{ID})$
    ID– an arbitrary bit-string in $\{0,1\}^*$
  - $c \leftarrow \mathsf{Enc}(\mathsf{PK}, \mathsf{ID}, m)$
  - $m / \perp := \mathsf{Dec}(\mathsf{PK}, sk_{\mathsf{ID}}, c)$
- Correctness:
  - For all ID $\in \{0,1\}^*$ and $m \in \mathcal{MSP}$,
    if $sk_{\mathsf{ID}} \leftarrow \mathsf{Ext}(\mathsf{PK}, \mathsf{MSK}, \mathsf{ID})$, then
    $\mathsf{Dec}(\mathsf{PK}, sk_{\mathsf{ID}}, \mathsf{Enc}(\mathsf{PK}, \mathsf{ID}, m)) = m$

AIBE – The Setting

- Algorithms:
  - $(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda)$
  - $sk_{\mathsf{ID}} \leftarrow \mathsf{Ext}(\mathsf{PK}, \mathsf{MSK}, \mathsf{ID})$
    ID– an arbitrary bit-string in $\{0,1\}^*$
  - $c \leftarrow \mathsf{Enc}(\mathsf{PK}, \mathsf{ID}, m)$
  - $m / \perp := \mathsf{Dec}(\mathsf{PK}, sk_{\mathsf{ID}}, c)$
- Correctness:
  - For all $\mathsf{ID} \in \{0,1\}^*$ and $m \in \mathcal{MSP}$,
    if $sk_{\mathsf{ID}} \leftarrow \mathsf{Ext}(\mathsf{PK}, \mathsf{MSK}, \mathsf{ID})$, then
    $\mathsf{Dec}(\mathsf{PK}, sk_{\mathsf{ID}}, \mathsf{Enc}(\mathsf{PK}, \mathsf{ID}, m)) = m$

AIBE – The Setting

- Algorithms:
  - $(\text{PK}, \text{MSK}) \leftarrow \text{Setup}(1^\lambda)$
  - $sk_{\text{ID}} \leftarrow \text{Ext}(\text{PK}, \text{MSK}, \text{ID})$
    ID– an arbitrary bit-string in $\{0,1\}^*$
  - $c \leftarrow \text{Enc}(\text{PK}, \text{ID}, m)$
  - $m/\perp := \text{Dec}(\text{PK}, sk_{\text{ID}}, c)$
- Correctness:
  - For all ID $\in \{0,1\}^*$ and $m \in \mathcal{MSP}$,
    if $sk_{\text{ID}} \leftarrow \text{Ext}(\text{PK}, \text{MSK}, \text{ID})$, then
    $\text{Dec}(\text{PK}, sk_{\text{ID}}, \text{Enc}(\text{PK}, \text{ID}, m)) = m$

AIBE – The Setting

- Algorithms:
  - $(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda)$
  - $sk_{\mathsf{ID}} \leftarrow \mathsf{Ext}(\mathsf{PK}, \mathsf{MSK}, \mathsf{ID})$
    ID– an arbitrary bit-string in $\{0,1\}^*$
  - $c \leftarrow \mathsf{Enc}(\mathsf{PK}, \mathsf{ID}, m)$
  - $m / \perp := \mathsf{Dec}(\mathsf{PK}, sk_{\mathsf{ID}}, c)$
- Correctness:
  - For all $\mathsf{ID} \in \{0,1\}^*$ and $m \in \mathcal{MSP}$,
    if $sk_{\mathsf{ID}} \leftarrow \mathsf{Ext}(\mathsf{PK}, \mathsf{MSK}, \mathsf{ID})$, then
    $\mathsf{Dec}(\mathsf{PK}, sk_{\mathsf{ID}}, \mathsf{Enc}(\mathsf{PK}, \mathsf{ID}, m)) = m$

AIBE – The Setting

- Algorithms:
  - $(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda)$
  - $sk_{\mathsf{ID}} \leftarrow \mathsf{Ext}(\mathsf{PK}, \mathsf{MSK}, \mathsf{ID})$
    ID– an arbitrary bit-string in $\{0,1\}^*$
  - $c \leftarrow \mathsf{Enc}(\mathsf{PK}, \mathsf{ID}, m)$
  - $m/\perp := \mathsf{Dec}(\mathsf{PK}, sk_{\mathsf{ID}}, c)$
- Correctness:
  - For all $\mathsf{ID} \in \{0,1\}^*$ and $m \in \mathcal{MSP}$,
    if $sk_{\mathsf{ID}} \leftarrow \mathsf{Ext}(\mathsf{PK}, \mathsf{MSK}, \mathsf{ID})$, then
    $\mathsf{Dec}(\mathsf{PK}, sk_{\mathsf{ID}}, \mathsf{Enc}(\mathsf{PK}, \mathsf{ID}, m)) = m$

AIBE – The Setting

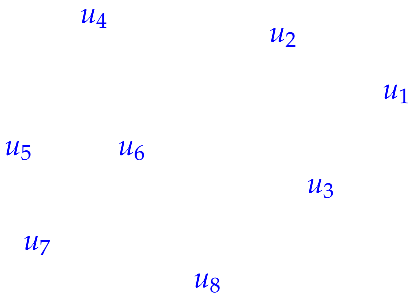
- Algorithms:
  - $(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda)$
  - $sk_{\mathsf{ID}} \leftarrow \mathsf{Ext}(\mathsf{PK}, \mathsf{MSK}, \mathsf{ID})$
    ID– an arbitrary bit-string in $\{0,1\}^*$
  - $c \leftarrow \mathsf{Enc}(\mathsf{PK}, \mathsf{ID}, m)$
  - $m / \perp := \mathsf{Dec}(\mathsf{PK}, sk_{\mathsf{ID}}, c)$
- Correctness:
  - For all $\mathsf{ID} \in \{0,1\}^*$ and $m \in \mathcal{MSP}$,
    if $sk_{\mathsf{ID}} \leftarrow \mathsf{Ext}(\mathsf{PK}, \mathsf{MSK}, \mathsf{ID})$, then
    $\mathsf{Dec}(\mathsf{PK}, sk_{\mathsf{ID}}, \mathsf{Enc}(\mathsf{PK}, \mathsf{ID}, m)) = m$

Outline

The Subset Cover Framework

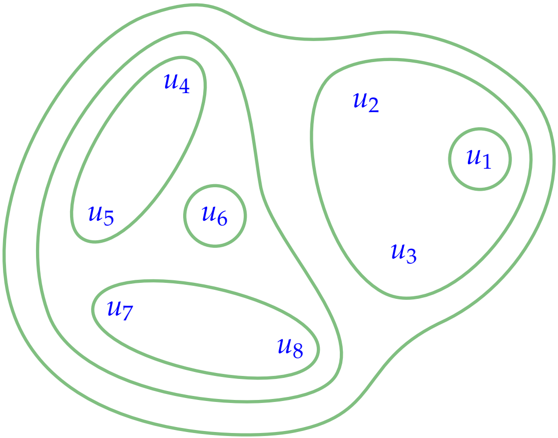
- Proposed by Naor *et al.* (2001)
- *Private-key* setting
- Goal: Define and analyze the security of revocation schemes

The Subset Cover Framework – Idea



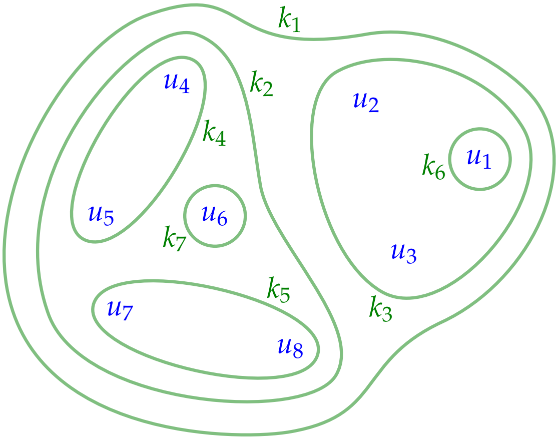
The Subset Cover Framework – Idea

Setup



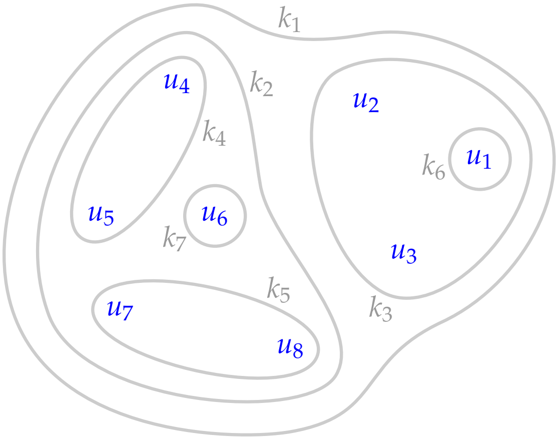
The Subset Cover Framework – Idea

Setup



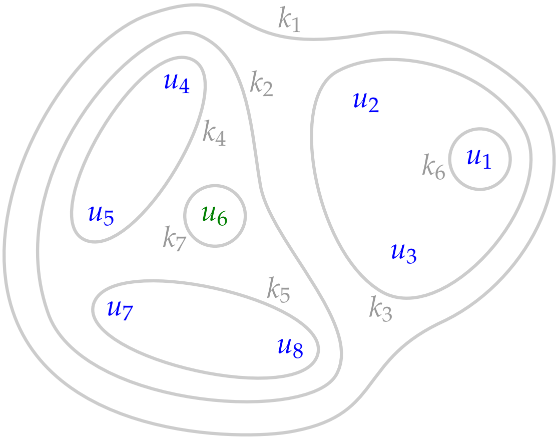
The Subset Cover Framework – Idea

Key Generation



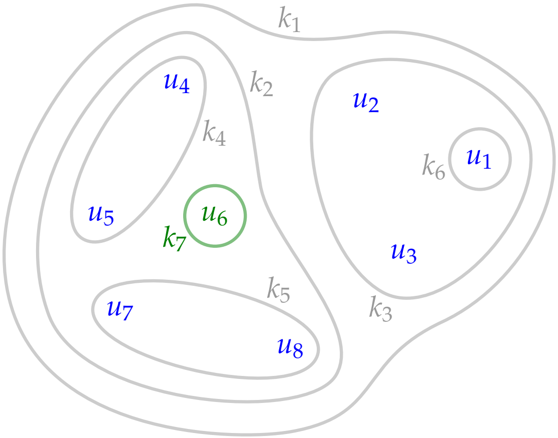
The Subset Cover Framework – Idea

Key Generation



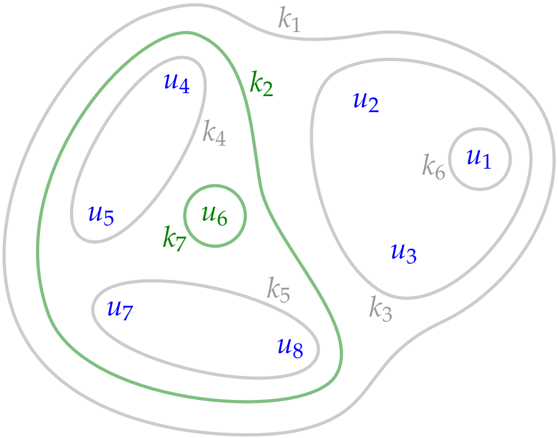
The Subset Cover Framework – Idea

Key Generation



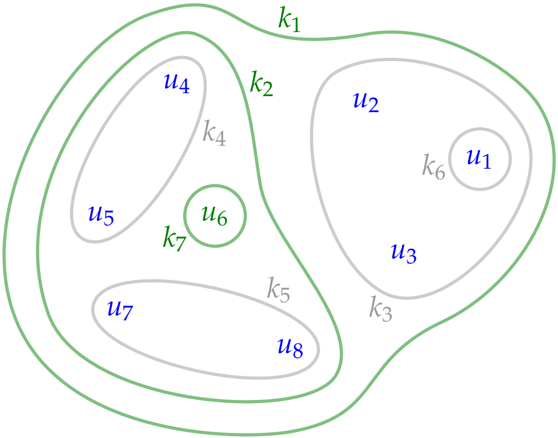
The Subset Cover Framework – Idea

Key Generation



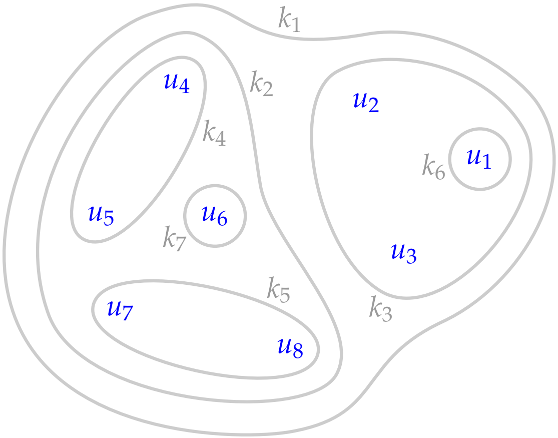
The Subset Cover Framework – Idea

Key Generation



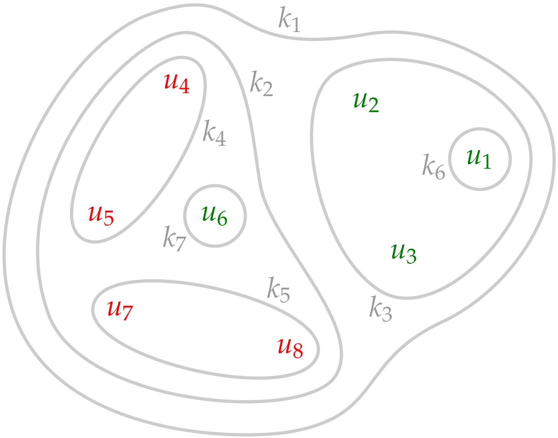
The Subset Cover Framework – Idea

Encryption/Decryption



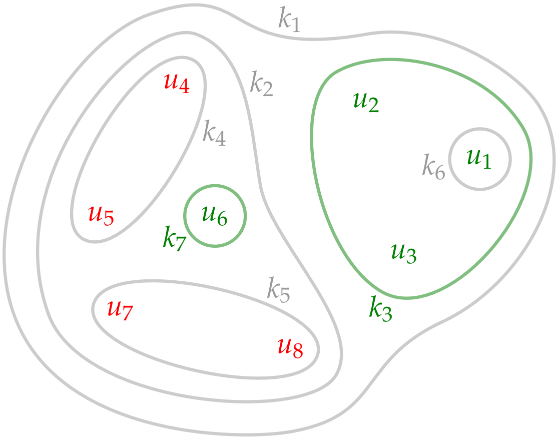
The Subset Cover Framework – Idea

Encryption/Decryption



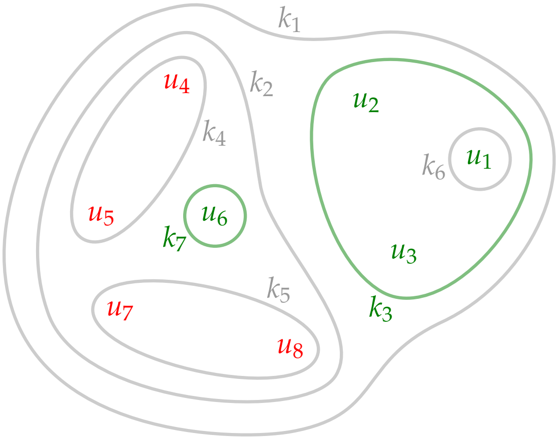
The Subset Cover Framework – Idea

Encryption/Decryption



The Subset Cover Framework – Idea

Encryption/Decryption



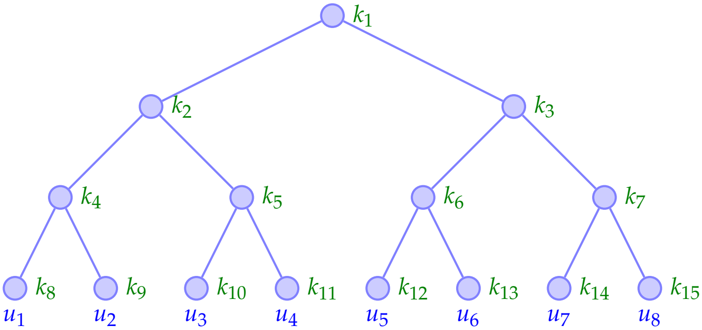
$$c = (E_s(m), E_{k_7}(s), E_{k_3}(s))$$

Two Ways of Defining $\mathcal{S}$

1. Complete Subtree (CS)
2. Subset Difference (SD)

Complete Subtree (CS) Method

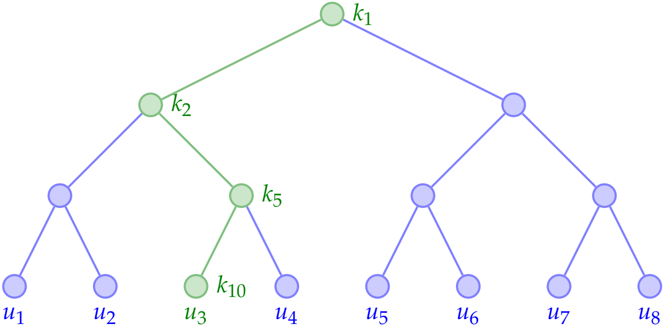
Setup



- The N users are the leaves of a full binary tree $\mathcal{T}$
- $\mathcal{S}$ contains all possible subtrees of $\mathcal{T}$

Complete Subtree (CS) Method

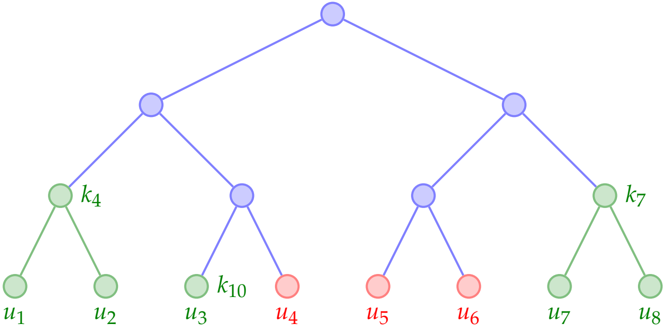
Key Generation



- Each user belongs to $\log(N) + 1$ subtrees
- Each user is given those $\log(N) + 1$ keys

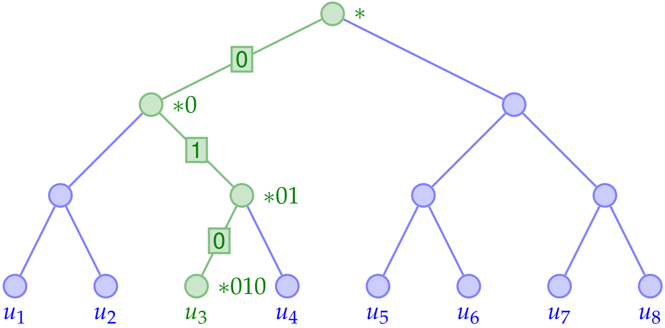
Complete Subtree (CS) Method

Encryption/Decryption



- Find the set $\mathcal{C}$ of subtrees covering the recipients
- Encrypt the session key using all the keys from $\mathcal{C}$

Extension of CS Method to the Public-key Setting



- Dodis and Fazio (2002) extends [NNL01] to *Public-key* setting
- Idea: Novel ID assignment + Identity-Based Encryption (IBE)

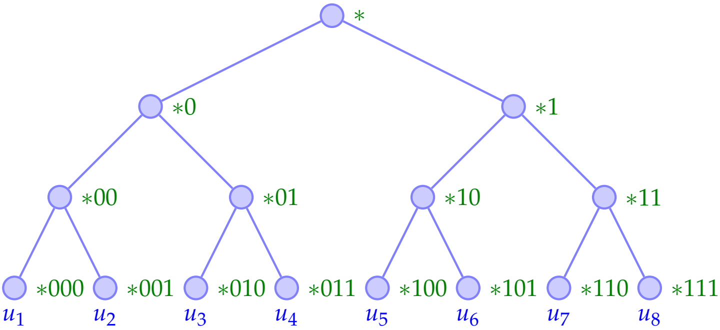
Outline

Our Constructions

- Idea: Extended CS method $+$ AIBE $=$ oABE
- Constructions:
  1. Generic CPA
  2. Generic CCA
  3. CCA with Enhanced Decryption

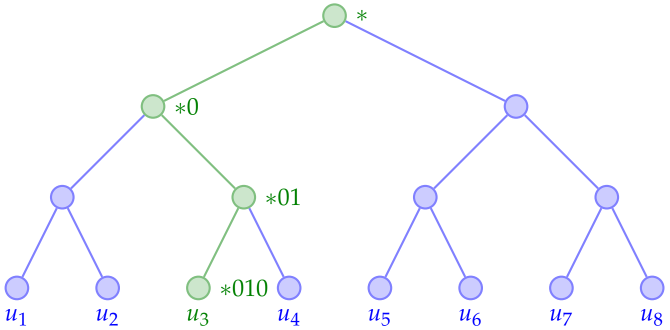
Our Constructions – Idea

Setup



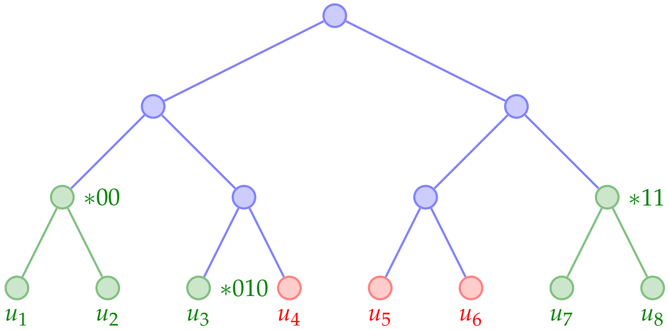
Our Constructions – Idea

Key Generation



Our Constructions – Idea

Encryption/Decryption



Outline

Generic CPA Construction

Given a *weakly robust* AIBE $\Pi' = (\mathsf{Init}, \mathsf{Ext}, \mathsf{Enc}, \mathsf{Dec})$

Setup$(1^\lambda, N)$:

1. Obtain $(\mathsf{PK}', \mathsf{MSK}') \leftarrow \mathsf{Init}(1^\lambda)$
2. Output $\mathsf{PK} = (\mathsf{PK}', N)$, $\mathsf{MSK} = \mathsf{MSK}'$

KeyGen$(\mathsf{PK}, \mathsf{MSK}, i)$:

1. Let $\mathsf{HID}_i = (\texttt{Root}, \mathsf{ID}_1, \ldots, \mathsf{ID}_n)$
2. For $k = 1$ to $n+1$:
   Compute $sk_{i,k} \leftarrow \mathsf{Ext}(\mathsf{PK}', \mathsf{MSK}', \mathsf{HID}_{i|k})$
3. Output $sk_i = (sk_{i,1}, \ldots, sk_{i,n+1})$

Generic CPA Construction

Given a *weakly robust* AIBE $\Pi' = (\mathsf{Init}, \mathsf{Ext}, \mathsf{Enc}, \mathsf{Dec})$

Setup$(1^\lambda, N)$:

1. Obtain $(\mathsf{PK}', \mathsf{MSK}') \leftarrow \mathsf{Init}(1^\lambda)$
2. Output $\mathsf{PK} = (\mathsf{PK}', N)$, $\mathsf{MSK} = \mathsf{MSK}'$

KeyGen$(\mathsf{PK}, \mathsf{MSK}, i)$:

1. Let $\mathsf{HID}_i = (\mathtt{Root}, \mathsf{ID}_1, \ldots, \mathsf{ID}_n)$
2. For $k = 1$ to $n+1$:
   Compute $sk_{i,k} \leftarrow \mathsf{Ext}(\mathsf{PK}', \mathsf{MSK}', \mathsf{HID}_{i|k})$
3. Output $sk_i = (sk_{i,1}, \ldots, sk_{i,n+1})$

Generic CPA Construction

Given a *weakly robust* AIBE $\Pi' = (\text{Init}, \text{Ext}, \text{Enc}, \text{Dec})$

Setup$(1^\lambda, N)$:

1. Obtain $(\text{PK}', \text{MSK}') \leftarrow \text{Init}(1^\lambda)$
2. Output $\text{PK} = (\text{PK}', N)$, $\text{MSK} = \text{MSK}'$

KeyGen$(\text{PK}, \text{MSK}, i)$:

1. Let $\text{HID}_i = (\texttt{Root}, \text{ID}_1, \ldots, \text{ID}_n)$
2. For $k = 1$ to $n + 1$:
   Compute $sk_{i,k} \leftarrow \text{Ext}(\text{PK}', \text{MSK}', \text{HID}_{i|k})$
3. Output $sk_i = (sk_{i,1}, \ldots, sk_{i,n+1})$

Generic CPA Construction

Encrypt(PK, S, m):

1. Find the set Cov of subtrees covering the set S
2. Set $r = N - |S|$, $L = \left\lfloor r \log\left(\frac{N}{r}\right)\right\rfloor$
3. For each subtree T_j (with HID_j) $\in$ Cov:
   Compute $c_j \leftarrow \mathsf{Enc}(\mathsf{PK}', \mathsf{HID}_j, m)$
4. Choose $\widetilde{m} \xleftarrow{\$} \{0,1\}^{|m|}$
5. For $|\mathsf{Cov}| + 1 \leq j \leq L$:
   Compute $c_j \leftarrow \mathsf{Enc}(\mathsf{PK}', \mathrm{dummy}, \widetilde{m})$
6. Set $c = \left(c_{\pi(1)}, \ldots, c_{\pi(L)}\right)$
7. Output c

Generic CPA Construction

Encrypt(PK, S, m):

1. Find the set Cov of subtrees covering the set S
2. Set $r = N - |S|$, $L = \left\lfloor r \log\left(\frac{N}{r}\right)\right\rfloor$
3. For each subtree T_j (with HID_j) $\in$ Cov:
   Compute $c_j \leftarrow \mathsf{Enc}(\mathsf{PK}', \mathsf{HID}_j, m)$
4. Choose $\widetilde{m} \xleftarrow{\$} \{0,1\}^{|m|}$
5. For $|\mathsf{Cov}| + 1 \leq j \leq L$:
   Compute $c_j \leftarrow \mathsf{Enc}(\mathsf{PK}', \mathrm{dummy}, \widetilde{m})$
6. Set $c = \left(c_{\pi(1)}, \ldots, c_{\pi(L)}\right)$
7. Output c

Generic CPA Construction

Encrypt(PK, S, m):

1. Find the set Cov of subtrees covering the set S
2. Set $r = N - |S|$, $L = \left\lfloor r \log \left(\frac{N}{r} \right) \right\rfloor$
3. For each subtree T_j (with HID_j) $\in$ Cov:
   Compute $c_j \leftarrow \mathsf{Enc}(\mathsf{PK}', \mathsf{HID}_j, m)$
4. Choose $\widetilde{m} \xleftarrow{\$} \{0,1\}^{|m|}$
5. For $|\mathsf{Cov}| + 1 \leq j \leq L$:
   Compute $c_j \leftarrow \mathsf{Enc}(\mathsf{PK}', \mathtt{dummy}, \widetilde{m})$
6. Set $c = \left(c_{\pi(1)}, \ldots, c_{\pi(L)} \right)$
7. Output c

Generic CPA Construction

Encrypt(PK, S, m):

1. Find the set Cov of subtrees covering the set S
2. Set $r = N - |S|$, $L = \left\lfloor r \log \left(\frac{N}{r}\right) \right\rfloor$
3. For each subtree T_j (with HID_j) $\in$ Cov:
   Compute $c_j \leftarrow \text{Enc}(\text{PK}', \text{HID}_j, m)$
4. Choose $\widetilde{m} \stackrel{\$}{\leftarrow} \{0,1\}^{|m|}$
5. For $|\text{Cov}| + 1 \leq j \leq L$:
   Compute $c_j \leftarrow \text{Enc}(\text{PK}', \texttt{dummy}, \widetilde{m})$
6. Set $c = \left(c_{\pi(1)}, \ldots, c_{\pi(L)}\right)$
7. Output c

Generic CPA Construction

Encrypt(PK, S, m):

1. Find the set Cov of subtrees covering the set S
2. Set $r = N - |S|$, $L = \left\lfloor r \log\left(\frac{N}{r}\right)\right\rfloor$
3. For each subtree T_j (with HID_j) $\in$ Cov:
   Compute $c_j \leftarrow \mathsf{Enc}(\mathsf{PK}', \mathsf{HID}_j, m)$
4. Choose $\tilde{m} \xleftarrow{\$} \{0,1\}^{|m|}$
5. For $|\mathsf{Cov}| + 1 \leq j \leq L$:
   Compute $c_j \leftarrow \mathsf{Enc}(\mathsf{PK}', \mathtt{dummy}, \tilde{m})$
6. Set $c = \left(c_{\pi(1)}, \ldots, c_{\pi(L)}\right)$
7. Output c

Generic CPA Construction

Encrypt(PK, S, m):

1. Find the set Cov of subtrees covering the set S
2. Set $r = N - |S|$, $L = \left\lfloor r \log\left(\frac{N}{r}\right)\right\rfloor$
3. For each subtree T_j (with HID_j) $\in$ Cov:
   Compute $c_j \leftarrow \mathsf{Enc}(\mathsf{PK}', \mathsf{HID}_j, m)$
4. Choose $\widetilde{m} \xleftarrow{\$} \{0,1\}^{|m|}$
5. For $|\mathsf{Cov}| + 1 \leq j \leq L$:
   Compute $c_j \leftarrow \mathsf{Enc}(\mathsf{PK}', \mathtt{dummy}, \widetilde{m})$
6. Set $c = \left(c_{\pi(1)}, \ldots, c_{\pi(L)}\right)$
7. Output c

Generic CPA Construction

Encrypt(PK, S, m):

1. Find the set Cov of subtrees covering the set S
2. Set $r = N - |S|$, $L = \left\lfloor r \log\left(\frac{N}{r}\right)\right\rfloor$
3. For each subtree T_j (with HID$_j$) $\in$ Cov:
   Compute $c_j \leftarrow \text{Enc}(\text{PK}', \text{HID}_j, m)$
4. Choose $\widetilde{m} \stackrel{\$}{\leftarrow} \{0,1\}^{|m|}$
5. For $|\text{Cov}| + 1 \leq j \leq L$:
   Compute $c_j \leftarrow \text{Enc}(\text{PK}', \texttt{dummy}, \widetilde{m})$
6. Set $c = \left(c_{\pi(1)}, \ldots, c_{\pi(L)}\right)$
7. Output c

Generic CPA Construction

Encrypt(PK, S, m):

1. Find the set Cov of subtrees covering the set S
2. Set $r = N - |S|$, $L = \left\lfloor r \log\left(\frac{N}{r}\right) \right\rfloor$
3. For each subtree T_j (with HID_j) $\in$ Cov:
   Compute $c_j \leftarrow \text{Enc}(\text{PK}', \text{HID}_j, m)$
4. Choose $\widetilde{m} \xleftarrow{\$} \{0,1\}^{|m|}$
5. For $|\text{Cov}| + 1 \leq j \leq L$:
   Compute $c_j \leftarrow \text{Enc}(\text{PK}', \texttt{dummy}, \widetilde{m})$
6. Set $c = \left(c_{\pi(1)}, \ldots, c_{\pi(L)}\right)$
7. Output c

Generic CPA Construction

Decrypt(PK, sk_i, c):

1. Parse sk_i as $(sk_{i,1}, \ldots, sk_{i,n+1})$ and c as $(c_1, \ldots, c_L)$
2. For $k = 1$ to $n + 1$:
   For $j = 1$ to L:
   Compute $m = \mathsf{Dec}(\mathsf{PK}', sk_{i,k}, c_j)$
   If $m \neq \perp$, return m
3. Return $\perp$

Generic CPA Construction

Decrypt(PK, sk_i, c):

1. Parse sk_i as $(sk_{i,1}, \ldots, sk_{i,n+1})$ and c as $(c_1, \ldots, c_L)$
2. For $k = 1$ to $n + 1$:
   For $j = 1$ to L:
   Compute $m = \mathsf{Dec}(\mathsf{PK}', sk_{i,k}, c_j)$
   If $m \neq \perp$, return m
3. Return $\perp$

Generic CPA Construction

Decrypt(PK, sk_i, c):

1. Parse sk_i as $(sk_{i,1}, \ldots, sk_{i,n+1})$ and c as $(c_1, \ldots, c_L)$
2. For $k = 1$ to $n + 1$:
   For $j = 1$ to L:
   Compute $m = \mathsf{Dec}(\mathsf{PK}', sk_{i,k}, c_j)$
   If $m \neq \perp$, return m
3. Return $\perp$

Generic CPA Construction

Decrypt(PK, sk_i, c):

1. Parse sk_i as $(sk_{i,1}, \ldots, sk_{i,n+1})$ and c as $(c_1, \ldots, c_L)$
2. For $k = 1$ to $n + 1$:
   For $j = 1$ to L:
   Compute $m = \mathsf{Dec}(\mathsf{PK}', sk_{i,k}, c_j)$
   If $m \neq \perp$, return m
3. Return $\perp$

Generic CPA Construction

Theorem

If $\Pi' = (\mathsf{Init}, \mathsf{Ext}, \mathsf{Enc}, \mathsf{Dec})$ *is* (t, q_{sk}, ϵ)-AIBE-IND-CPA *secure, then the above construction is* $(t, q_{sk}, 2\,\epsilon\, r \log\left(\frac{N}{r}\right))$-oABE-IND-CPA *secure.*

Outline

Generic CCA Construction

Given a *weakly robust* AIBE $\Pi' = (\text{Init}, \text{Ext}, \text{Enc}, \text{Dec})$,
and a one-time signature scheme $\Sigma = (\text{Gen}, \text{Sign}, \text{Vrfy})$

$\text{Setup}(1^\lambda, N)$:

1. Obtain $(\text{PK}', \text{MSK}') \leftarrow \text{Init}(1^\lambda)$
2. Output $\text{PK} = (\text{PK}', N)$, $\text{MSK} = \text{MSK}'$

$\text{KeyGen}(\text{PK}, \text{MSK}, i)$:

1. Let $\text{HID}_i = (\texttt{Root}, \text{ID}_1, \ldots, \text{ID}_n)$
2. For $k = 1$ to $n + 1$:
   Compute $sk_{i,k} \leftarrow \text{Ext}(\text{PK}', \text{MSK}', \text{HID}_{i|k})$
3. Output $sk_i = (sk_{i,1}, \ldots, sk_{i,n+1})$

Generic CCA Construction

Encrypt(PK, S, m):

1. Generate $(\mathsf{VK}, \mathsf{SK}) \leftarrow \mathsf{Gen}(1^\lambda)$
2. Find the set Cov of subtrees covering the set S
3. Set $r = N - |S|$, $L = \left\lfloor r \log\left(\frac{N}{r}\right)\right\rfloor$
4. For each subtree T_j (with HID_j) $\in$ Cov:
   Compute $c_j \leftarrow \mathsf{Enc}(\mathsf{PK}', \mathsf{HID}_j, \mathsf{VK}||m)$
5. Choose $\widetilde{m} \xleftarrow{\$} \{0,1\}^{|\mathsf{VK}||m|}$
6. For $|\mathsf{Cov}| + 1 \leq j \leq L$:
   Compute $c_j \leftarrow \mathsf{Enc}(\mathsf{PK}', \mathtt{dummy}, \widetilde{m})$
7. Set $c = \left(c_{\pi(1)}, \ldots, c_{\pi(L)}\right)$
8. Generate $\sigma \leftarrow \mathsf{Sign}(\mathsf{SK}, \mathsf{VK}||c)$
9. Output $C = (\sigma, c)$

Generic CCA Construction

Decrypt(PK, sk_i, c):

1. Parse sk_i as $(sk_{i,1}, \ldots, sk_{i,n+1})$ and C as $(\sigma, c = (c_1, \ldots, c_L))$
2. For $k = 1$ to $n+1$:
   For $j = 1$ to L:
   Compute $m' = \mathsf{Dec}(\mathsf{PK}', sk_{i,k}, c_j)$
   If $m' \neq \perp \wedge\ m' = \mathsf{VK}||m \wedge\ \mathsf{Vrfy}(\mathsf{VK}, \sigma, \mathsf{VK}||c)$,
   return m
3. Return $\perp$

Generic CCA Construction

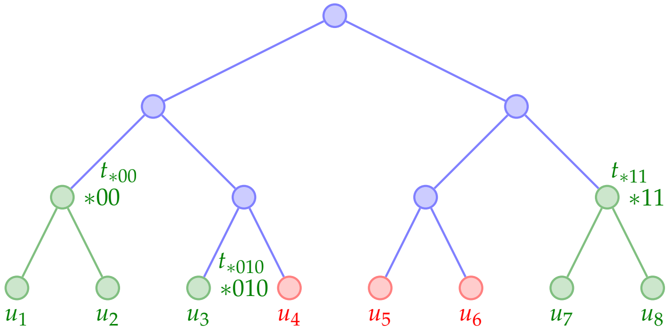
Theorem

If $\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ *is* (t, ϵ_1)*-strongly existentially unforgeable and* $\Pi' = (\mathsf{Init}, \mathsf{Ext}, \mathsf{Enc}, \mathsf{Dec})$ *is* $(t, q_{sk}, q_d, \epsilon_2)$-AIBE-IND-CCA *secure, then the above construction is* $(t, q_{sk}, q_d, 2(\epsilon_1 + \epsilon_2)\, r \log\left(\frac{N}{r}\right))$-oABE-IND-CCA *secure.*

Outline

Our Constructions – Idea

Enhanced Decryption



Enhanced CCA Construction

Given, a *weakly robust* AIBE $\Pi' = (\mathsf{Init}, \mathsf{Ext}, \mathsf{Enc}, \mathsf{Dec})$
a one-time signature scheme $\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$
a group $\mathbb{G} = \langle g \rangle$ with prime order $q > 2^{\lambda}$ where CDH is hard and DDH is easy
a cryptographic hash $H' : \mathbb{G} \to \{0,1\}^{\lambda}$

Setup$(1^{\lambda}, N)$:

1. Obtain $(\mathsf{PK}', \mathsf{MSK}') \leftarrow \mathsf{Init}(1^{\lambda})$
2. For each subtree T (with HID) $\in \mathcal{T}$:
   Draw $a_{\mathsf{HID}} \xleftarrow{\$} \mathbb{Z}_q^*$
   Compute $A_{\mathsf{HID}} = g^{a_{\mathsf{HID}}}$
3. Output
   $\mathsf{PK} = (\mathsf{PK}', N, \mathbb{G}, g, \{A_{\mathsf{HID}}\}_{\mathsf{HID} \in \mathcal{T}})$
   $\mathsf{MSK} = (\mathsf{MSK}', \{a_{\mathsf{HID}}\}_{\mathsf{HID} \in \mathcal{T}})$

Enhanced CCA Construction

KeyGen$(\mathsf{PK}, \mathsf{MSK}, i)$:

1. Let $\mathsf{HID}_i = (\texttt{Root}, \mathsf{ID}_1, \ldots, \mathsf{ID}_n)$
2. For $k = 1$ to $n + 1$:
   Set $\overline{sk}_{i,k} = a_{\mathsf{HID}_{i|k}}$
   Compute $sk_{i,k} \leftarrow \mathsf{Ext}(\mathsf{PK}', \mathsf{MSK}', \mathsf{HID}_{i|k})$
3. Output $sk_i = \left(\left(\overline{sk}_{i,1}, sk_{i,1}\right), \ldots, \left(\overline{sk}_{i,n+1}, sk_{i,n+1}\right)\right)$

Enhanced CCA Construction

Encrypt(PK, S, m):

1. Generate $(\mathsf{VK}, \mathsf{SK}) \leftarrow \mathsf{Gen}(1^\lambda)$
2. Find the set Cov of subtrees covering the set S
3. Set $r = N - |S|$, $L = \left\lfloor r \log\left(\frac{N}{r}\right)\right\rfloor$
4. Draw $s \xleftarrow{\$} \mathbb{Z}_q^*$, and compute $\bar{c}_0 = g^s$
5. For each subtree T_j (with HID_j) $\in$ Cov:
   - Set $\bar{c}_j = H'(A^s_{\mathsf{HID}_j})$
   - Compute $c_j \leftarrow \mathsf{Enc}(\mathsf{PK}', \mathsf{HID}_j, \mathsf{VK}||A^s_{\mathsf{HID}_j}||m)$
6. Choose $\widetilde{m} \xleftarrow{\$} \{0,1\}^{|\mathsf{VK}||\bar{c}_0||m|}$
7. For $|\mathsf{Cov}| + 1 \leq j \leq L$:
   - Draw $s_j \xleftarrow{\$} \mathbb{Z}_q^*$
   - Compute $\bar{c}_j = H'(g^{s_j})$
8. Set $c = \left(\bar{c}_0, \left(\bar{c}_{\pi(1)}, c_{\pi(1)}\right), \ldots, \left(\bar{c}_{\pi(L)}, c_{\pi(L)}\right)\right)$
9. Generate $\sigma \leftarrow \mathsf{Sign}(\mathsf{SK}, \mathsf{VK}||c)$
10. Output $C = (\sigma, c)$

Enhanced CCA Construction

Decrypt(PK, sk_i, C):

1. Parse sk_i as $((\overline{sk}_{i,1}, sk_{i,1}), \ldots, (\overline{sk}_{i,n+1}, sk_{i,n+1}))$ and C as $(\sigma, c = (\bar{c}_0, (\bar{c}_1, c_1), \ldots, (\bar{c}_L, c_L)))$.
2. For $k = 1$ to $n+1$:
   Compute $tag_k = H'(\bar{c}_0^{\overline{sk}_{i,k}})$
3. If $\exists\, k \in [1, n+1]$, $\exists\, j \in [1, L]$ $tag_k = \bar{c}_j$
   Compute $m' = \mathsf{Dec}(\mathsf{PK}', sk_{i,k}, c_j)$
   Set $\overline{m} = \bar{c}_0^{\overline{sk}_{i,k}}$
   If $m' = \mathsf{VK}||\overline{m}||m \wedge \mathsf{Vrfy}(\mathsf{VK}, \sigma, \mathsf{VK}||c)$,
   return m
4. Return $\perp$

Enhanced CCA Construction

Theorem

If $\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ *is* (t, ϵ_1)*-strongly existentially unforgeable,* $\Pi' = (\mathsf{Init}, \mathsf{Ext}, \mathsf{Enc}, \mathsf{Dec})$ *is* $(t, q_{sk}, q_d, \epsilon_2)$-AIBE-IND-CCA *secure, and* CDH *is* (t, ϵ_3)*-hard in* $\mathbb{G}$ *and* DDH *is efficiently computable in* $\mathbb{G}$*, then the above construction is* $(t, q_{sk}, q_d, 2(\epsilon_1 + \epsilon_2 + \epsilon_3)\, r \log\left(\frac{N}{r}\right))$-oABE-IND-CCA *secure, in the random oracle model.*

Outline

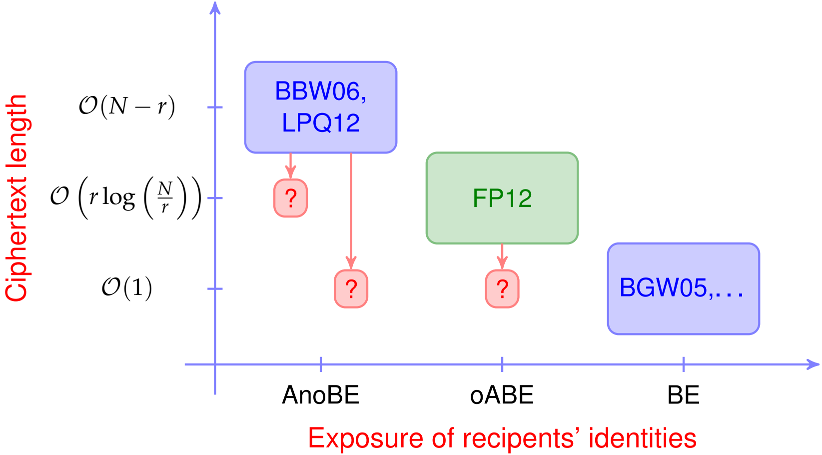
Comparisons

	Scheme	PK Length	SK Length	CT Length	Decryption Attempts
Regular	BBW06	$\mathcal{O}(N)$	$\mathcal{O}(1)$	$\mathcal{O}(N-r)$	$\mathcal{O}(N-r)$
	LPQ12	$\mathcal{O}(N)$	$\mathcal{O}(1)$	$\mathcal{O}(N-r)$	$\mathcal{O}(N-r)$
	FP12a	$\mathcal{O}(1)$	$\mathcal{O}(\log N)$	$\mathcal{O}(r\log(\frac{N}{r}))$	$\mathcal{O}(r\log(\frac{N}{r})\log N)$
Enhanced	BBW06	$\mathcal{O}(N)$	$\mathcal{O}(1)$	$\mathcal{O}(N-r)$	1
	LPQ12	$\mathcal{O}(N)$	$\mathcal{O}(1)$	$\mathcal{O}(N-r)$	1
	FP12a	$\mathcal{O}(N)$	$\mathcal{O}(\log N)$	$\mathcal{O}(r\log(\frac{N}{r}))$	1
	FP12b	$\mathcal{O}(N\log N)$	$\mathcal{O}(N)$	$\mathcal{O}(r)$	1

N: total number of users. r: number of revoked users.

Outline

Open Problems



Thank You!