Adaptive Oblivious Transfer And Generalization

Olivier Blazy, Céline Chevalier, **Paul Germouty**

December 5, 2016

1. Oblivious Transfer

2. OLBE: A Natural Generalization

3. Adaptive Oblivious Transfer

4. What To Remember

1 Oblivious Transfer

2 OLBE: A Natural Generalization

3 Adaptive Oblivious Transfer

4 What To Remember

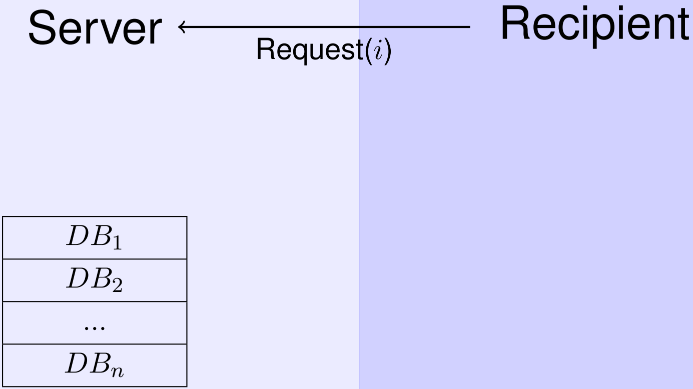
Oblivious Transfer

Oblivious Transfer

Server

DB_1
DB_2
$\ldots$
DB_n

Oblivious Transfer



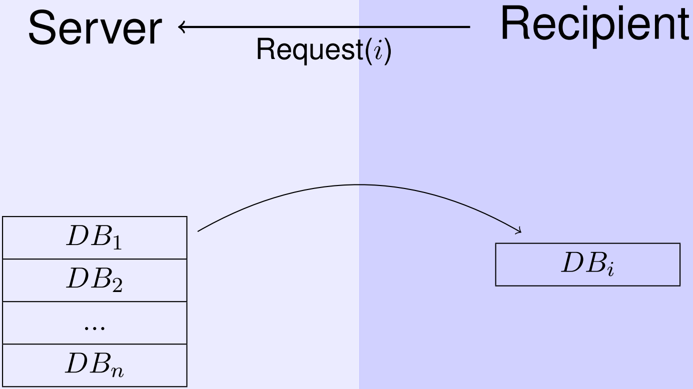
Server

Recipient

Request(i)

DB_1
DB_2
$\dots$
DB_n

Oblivious Transfer



Server

Recipient

Request(i)

DB_1

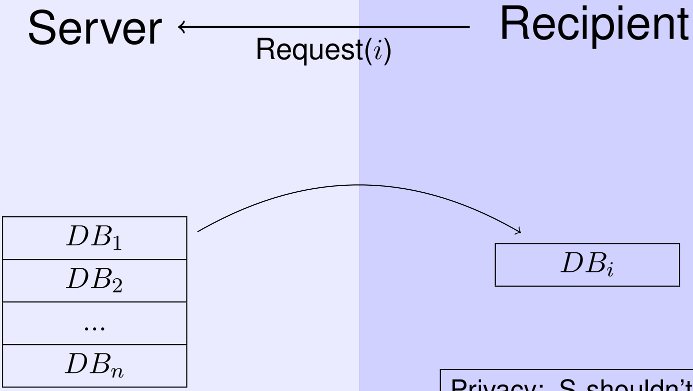
DB_2

$\ldots$

DB_n

DB_i

Oblivious Transfer



Server

Request(i)

Recipient

DB_1
DB_2
...
DB_n

DB_i

Privacy: S shouldn't know i and R shouldn't have any information about other lines.

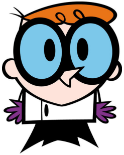
Identity Based Encryption

Identity Based Encryption

Alice

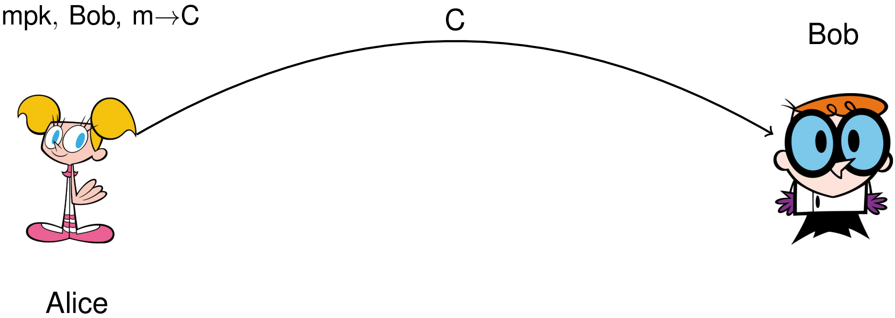
Identity Based Encryption

Bob



Alice

Identity Based Encryption



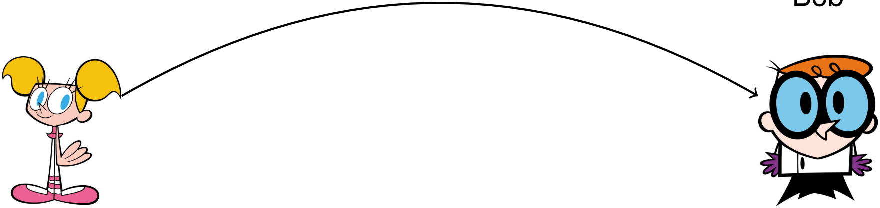
mpk, Bob, m→C

C

Bob

Alice

Identity Based Encryption



mpk, Bob, m→C

C

Bob

Alice

usk[Bob],C→m

Identity Based Key Encapsulation Mechanism

- **Gen**(param): generates $(\mathsf{mpk}, \mathsf{msk})$
- **USKGen**$(\mathbf{msk}, \mathbf{id})$: computes $\mathsf{usk}[\mathsf{id}]$
- **Enc**$(\mathbf{mpk}, \mathbf{id})$: encrypts a key K into C
- **Dec**$(\mathbf{usk}[\mathbf{id}], C)$: decrypts C into K

UC-framework and Security Model

UC-framework and Security Model

- Ideal functionality vs real world

UC-framework and Security Model

- Ideal functionality vs real world
- Adaptive corruptions: the adversary can ask for internal state of the recipient at any moment and then play his role.

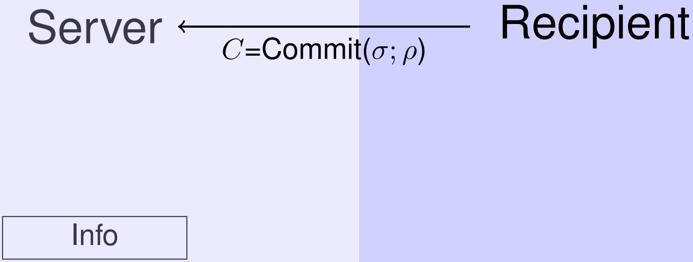
1. Oblivious Transfer
2. OLBE: A Natural Generalization
3. Adaptive Oblivious Transfer
4. What To Remember

The Oblivious Signature Based Envelope Protocol

Server

Info

The Oblivious Signature Based Envelope Protocol



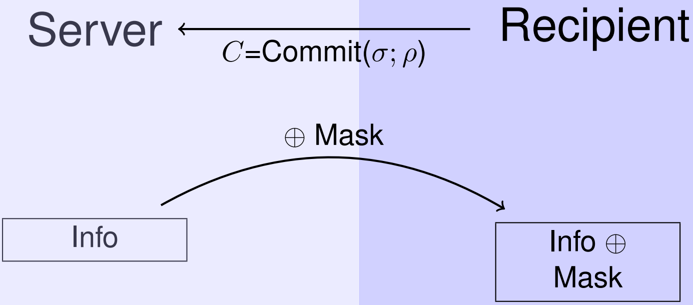
Server

Recipient

C=Commit($\sigma; \rho$)

Info

The Oblivious Signature Based Envelope Protocol



Server

Recipient

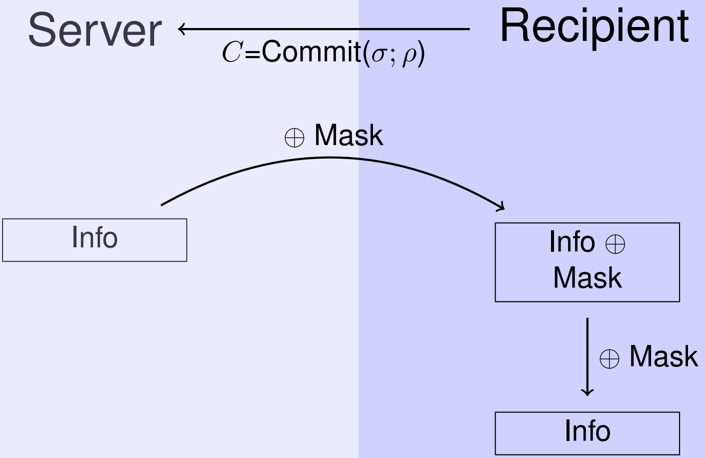
C=Commit($\sigma; \rho$)

$\oplus$ Mask

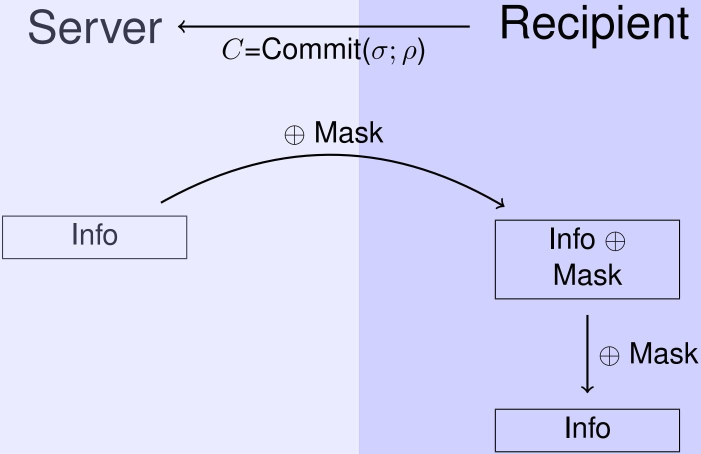
Info

Info $\oplus$ Mask

The Oblivious Signature Based Envelope Protocol



The Oblivious Signature Based Envelope Protocol



Mask computable for the user if and only if C is a commitment of σ

How To Do So: Commitment

How To Do So: Commitment

Commitment:

- Setup
- KeyGen
- Commit
- Decommit

How To Do So: Commitment

Commitment:

- Setup
- KeyGen
- Commit
- Decommit

Properties:

- Extractable
- Equivocable

How To Do So: Commitment

Commitment:

- Setup
- KeyGen
- Commit
- Decommit

Properties:

- Extractable
- Equivocable

Example: Encryption, Chameleon Hash Function: $(\mathsf{KeyGen}, \mathsf{CH}, \mathsf{Coll})$

How To Do So: Commitment

Commitment:

- Setup
- KeyGen
- Commit
- Decommit

Properties:

- Extractable
- Equivocable

Example: Encryption, Chameleon Hash Function: $(\mathsf{KeyGen}, \mathsf{CH}, \mathsf{Coll})$

$$\text{If } \mathsf{CH}(\mathsf{ck}, m; r) = H \text{ then}$$

$$\mathsf{coll}(\mathsf{ck}, \mathsf{tk}, H; m') = r' \text{ s. t. } \mathsf{CH}(\mathsf{ck}, m'; r') = H$$

How To Do So: Smooth Projective Hash Function

Functions over a set X and $\mathfrak{L} \subset X$

How To Do So: Smooth Projective Hash Function

Functions over a set X and $\mathfrak{L} \subset X$

- $\mathsf{HashKG}(\mathfrak{L}, \mathsf{param}) \rightarrow \mathsf{hk}$

How To Do So: Smooth Projective Hash Function

Functions over a set X and $\mathfrak{L} \subset X$

- $\mathsf{HashKG}(\mathfrak{L}, \mathsf{param}) \to \mathsf{hk}$
- $\mathsf{ProjKG}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W) \to \mathsf{hp}$

How To Do So: Smooth Projective Hash Function

Functions over a set X and $\mathfrak{L} \subset X$

- $\mathsf{HashKG}(\mathfrak{L}, \mathsf{param}) \rightarrow \mathsf{hk}$
- $\mathsf{ProjKG}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W) \rightarrow \mathsf{hp}$
- $\mathsf{Hash}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W) \rightarrow H \in \mathcal{G}$

How To Do So: Smooth Projective Hash Function

Functions over a set X and $\mathfrak{L} \subset X$

- $\mathsf{HashKG}(\mathfrak{L}, \mathsf{param}) \rightarrow \mathsf{hk}$
- $\mathsf{ProjKG}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W) \rightarrow \mathsf{hp}$
- $\mathsf{Hash}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W) \rightarrow H \in \mathcal{G}$
- $\mathsf{ProjHash}(\mathsf{hp}, (\mathfrak{L}, \mathsf{param}), W, w) \rightarrow H' \in \mathcal{G}$

How To Do So: Smooth Projective Hash Function

Functions over a set X and $\mathfrak{L} \subset X$

- HashKG$(\mathfrak{L}, \mathsf{param}) \to$ hk
- ProjKG$(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W) \to$ hp
- Hash$(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W) \to H \in \mathcal{G}$
- ProjHash$(\mathsf{hp}, (\mathfrak{L}, \mathsf{param}), W, w) \to H' \in \mathcal{G}$

Hash$(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W) =$ ProjHash$(\mathsf{hp}, (\mathfrak{L}, \mathsf{param}), W, w)$.
If w is a witness for $W \in \mathfrak{L}$.

Properties Of SPHF

- Smoothness: If $W \notin \mathfrak{L}$ nobody can distinguish a hashed value from a random one.

Properties Of SPHF

- Smoothness: If $W \notin \mathfrak{L}$ nobody can distinguish a hashed value from a random one.
- Pseudo-Randomness: Without w, if $W \in \mathfrak{L}$ it is hard to distinguish a hashed value from a random one.

A Simple Example Of SPHF

Here param contains $(g, h) \in \mathcal{G}$

A Simple Example Of SPHF

Here param contains $(g,h) \in \mathcal{G}$

- $\mathfrak{L} = \{(g_1, h_1) | \exists \alpha \in \mathbb{Z}_p, g_1 = g^\alpha \wedge h_1 = h^\alpha\}$, $w = \alpha$.

A Simple Example Of SPHF

Here param contains $(g, h) \in \mathcal{G}$

- $\mathfrak{L} = \{(g_1, h_1) | \exists \alpha \in \mathbb{Z}_p, g_1 = g^\alpha \wedge h_1 = h^\alpha\}$, $w = \alpha$.
- hk $= (\lambda, \mu) \in \mathbb{Z}_p^2$

A Simple Example Of SPHF

Here param contains $(g, h) \in \mathcal{G}$

- $\mathfrak{L} = \{(g_1, h_1) | \exists \alpha \in \mathbb{Z}_p, g_1 = g^\alpha \wedge h_1 = h^\alpha\}, w = \alpha$.
- hk $= (\lambda, \mu) \in \mathbb{Z}_p^2$
- hp $= g^\lambda h^\mu$

A Simple Example Of SPHF

Here param contains $(g, h) \in \mathcal{G}$

- $\mathfrak{L} = \{(g_1, h_1) | \exists \alpha \in \mathbb{Z}_p, g_1 = g^\alpha \wedge h_1 = h^\alpha\}$, $w = \alpha$.
- hk $= (\lambda, \mu) \in \mathbb{Z}_p^2$
- hp $= g^\lambda h^\mu$
- $H = g_1^\lambda h_1^\mu$

A Simple Example Of SPHF

Here param contains $(g, h) \in \mathcal{G}$

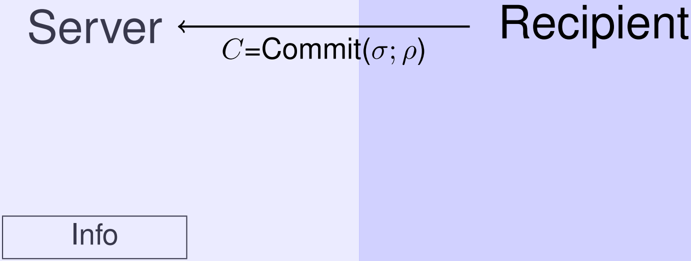
- $\mathfrak{L} = \{(g_1, h_1) | \exists \alpha \in \mathbb{Z}_p, g_1 = g^\alpha \wedge h_1 = h^\alpha\}$, $w = \alpha$.
- hk $= (\lambda, \mu) \in \mathbb{Z}_p^2$
- hp $= g^\lambda h^\mu$
- $H = g_1^\lambda h_1^\mu$
- $H' =$ hp$^\alpha$

SPHF And Implicit Decommitment Achieving OSBE

Server

Info

SPHF And Implicit Decommitment Achieving OSBE



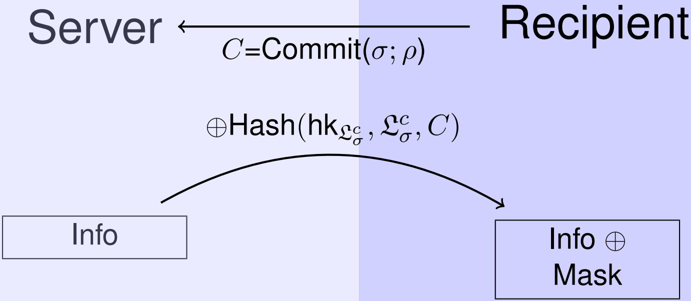
Server

Recipient

C=Commit$(\sigma; \rho)$

Info

SPHF And Implicit Decommitment Achieving OSBE



Server

Recipient

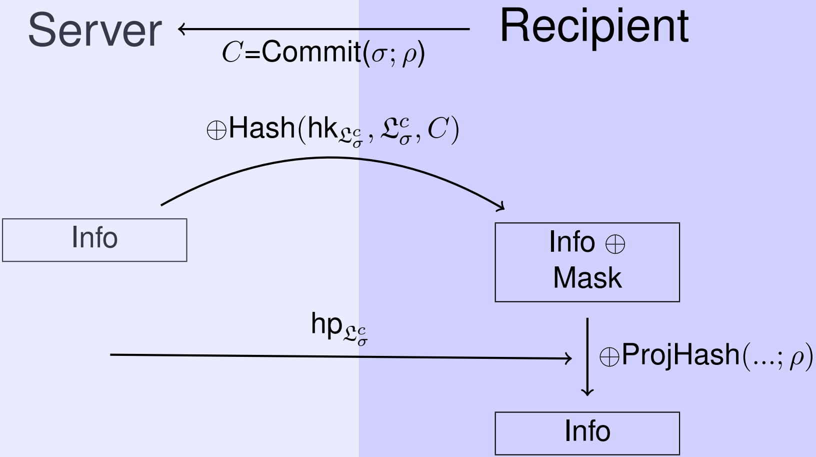
C=Commit$(\sigma;\rho)$

$\oplus$Hash$(\mathrm{hk}_{\mathfrak{L}^c_\sigma}, \mathfrak{L}^c_\sigma, C)$

Info

Info $\oplus$ Mask

SPHF And Implicit Decommitment Achieving OSBE



Server

Recipient

C=Commit$(\sigma;\rho)$

$\oplus$Hash$(\mathsf{hk}_{\mathfrak{L}^c_\sigma}, \mathfrak{L}^c_\sigma, C)$

Info

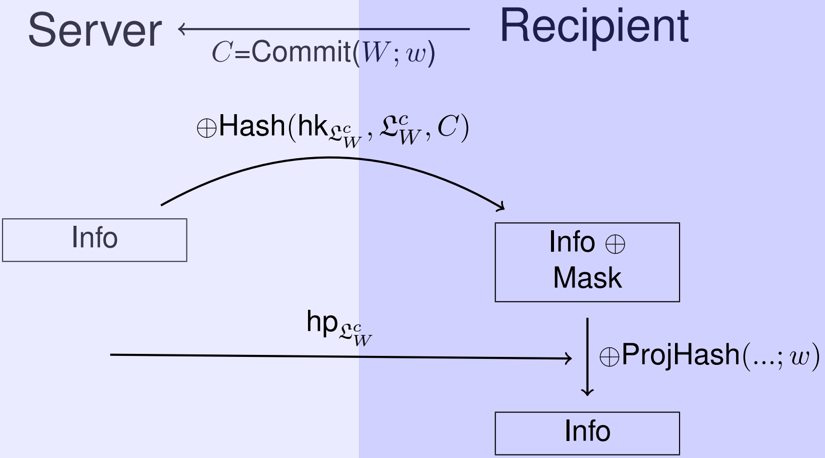
Info $\oplus$ Mask

$\mathsf{hp}_{\mathfrak{L}^c_\sigma}$

$\oplus$ProjHash$(\ldots;\rho)$

Info

Generalization: Oblivious Language Based Envelope



Server

Recipient

C=Commit(W; w)

$\oplus$Hash(hk$_{\mathfrak{L}_W^c}$, $\mathfrak{L}_W^c$, C)

Info

Info $\oplus$ Mask

hp$_{\mathfrak{L}_W^c}$

$\oplus$ProjHash(...; w)

Info

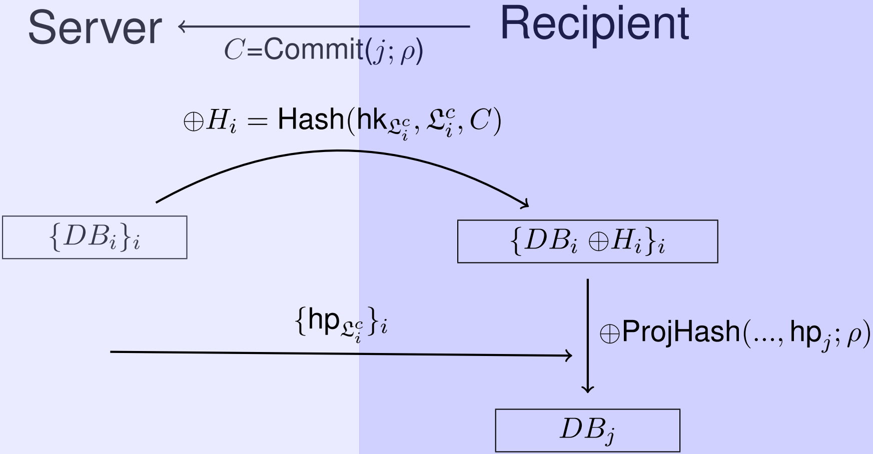
Examples

- Oblivious Signature Based Envelope

Examples

- Oblivious Signature Based Envelope
- Oblivious Transfer

Oblivious Transfer



Server

Recipient

C=Commit$(j;\rho)$

$\oplus H_i = \mathsf{Hash}(\mathsf{hk}_{\mathfrak{L}_i^c}, \mathfrak{L}_i^c, C)$

$\{DB_i\}_i$

$\{DB_i \oplus H_i\}_i$

$\{\mathsf{hp}_{\mathfrak{L}_i^c}\}_i$

$\oplus\mathsf{ProjHash}(\ldots, \mathsf{hp}_j; \rho)$

DB_j

Examples

- Oblivious Signature Based Envelope
- Oblivious Transfer

Examples

- Oblivious Signature Based Envelope
- Oblivious Transfer
- Conditionned Oblivious Transfer

1 Oblivious Transfer

2 OLBE: A Natural Generalization

3 Adaptive Oblivious Transfer

4 What To Remember

Be Adaptive

- Adaptive in terms of request:

Be Adaptive

- Adaptive in terms of request:
    - Previous works: a new request causes a resent of *DB*.

Be Adaptive

- Adaptive in terms of request:
  - Previous works: a new request causes a resent of *DB*.
  - This work: a new request after the first one has a logarithmic cost in the size of *DB*.

Be Adaptive

- Adaptive in terms of request:
  - Previous works: a new request causes a resent of *DB*.
  - This work: a new request after the first one has a logarithmic cost in the size of *DB*.
- Move the problem to the recipient side:

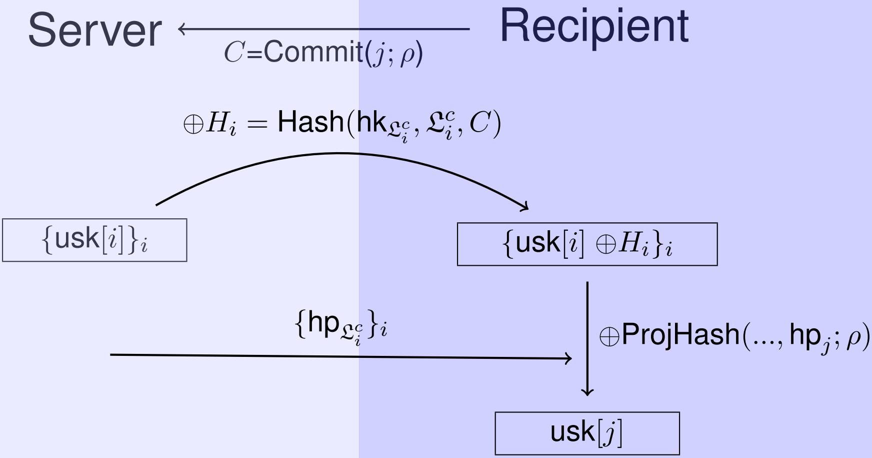
Be Adaptive

- Adaptive in terms of request:
  - Previous works: a new request causes a resent of *DB*.
  - This work: a new request after the first one has a logarithmic cost in the size of *DB*.
- Move the problem to the recipient side:
  - Sends the full encrypted *DB* to $\mathcal{R}$

Be Adaptive

- Adaptive in terms of request:
  - Previous works: a new request causes a resent of *DB*.
  - This work: a new request after the first one has a logarithmic cost in the size of *DB*.
- Move the problem to the recipient side:
  - Sends the full encrypted *DB* to $\mathcal{R}$
  - Do an OT with the keys used to encrypt.

OT On IBE Keys: A Blind IBE



Server

Recipient

C=Commit$(j;\rho)$

$\oplus H_i = \mathsf{Hash}(\mathsf{hk}_{\mathfrak{L}_i^c}, \mathfrak{L}_i^c, C)$

$\{\mathsf{usk}[i]\}_i$

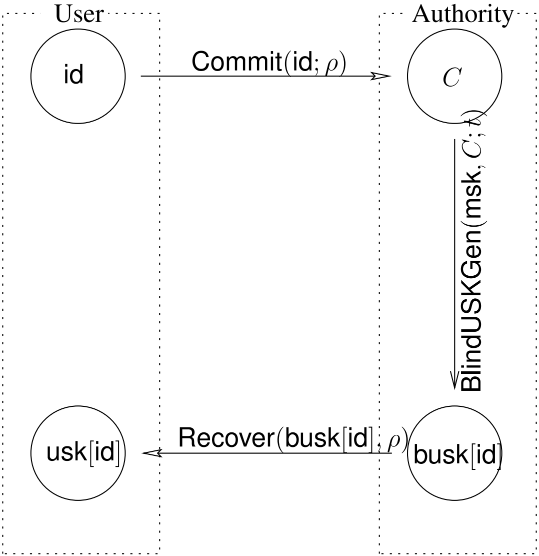
$\{\mathsf{usk}[i] \oplus H_i\}_i$

$\{\mathsf{hp}_{\mathfrak{L}_i^c}\}_i$

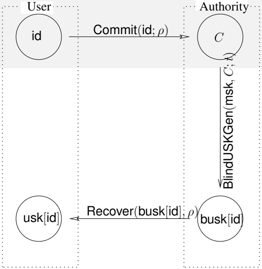
$\oplus\mathsf{ProjHash}(\ldots, \mathsf{hp}_j; \rho)$

$\mathsf{usk}[j]$

OT On IBE-Keys: A Blind IBE

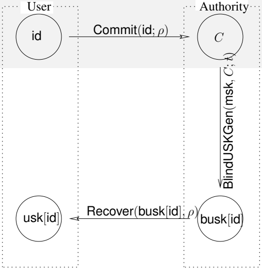

User
Authority
id
$\mathsf{Commit}(\mathsf{id};\rho)$
C
$\mathsf{BlindUSKGen}(\mathsf{msk}, C; t)$
$\mathsf{usk}[\mathsf{id}]$
$\mathsf{Recover}(\mathsf{busk}[\mathsf{id}], \rho)$
$\mathsf{busk}[\mathsf{id}]$

BIBE Generic Construction



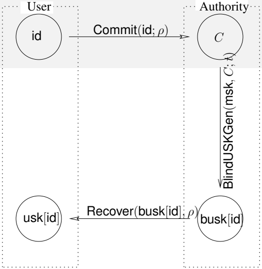
1. User: $C = \mathsf{Encrypt}_{\mathrm{cca}}(\mathsf{id}; \rho)$

BIBE Generic Construction



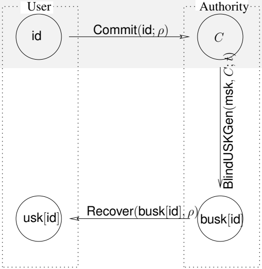
1. User: $C = \mathsf{Encrypt}_{\mathrm{cca}}(\mathsf{id}; \rho)$
2. For every id' $\mathcal{S}$ computes:
   - $(\mathsf{usk}[\mathsf{id}'], (\mathsf{hk}_{\mathsf{id}'}, \mathsf{hp}_{\mathsf{id}'}))$ for SPHF on $\mathfrak{L}^{c}_{\mathsf{id}'}$

BIBE Generic Construction



1. User: $C = \mathsf{Encrypt}_{\mathrm{cca}}(\mathsf{id}; \rho)$
2. For every id' $\mathcal{S}$ computes:
   - $(\mathsf{usk}[\mathsf{id}'], (\mathsf{hk}_{\mathsf{id}'}, \mathsf{hp}_{\mathsf{id}'}))$ for SPHF on $\mathfrak{L}^c_{\mathsf{id}'}$,
   - $H_{\mathsf{id}'}$

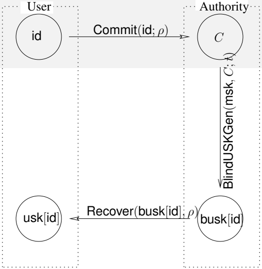
BIBE Generic Construction



1. User: $C = \mathsf{Encrypt}_{\text{cca}}(\mathsf{id}; \rho)$
2. For every id′ $\mathcal{S}$ computes:
   - $(\mathsf{usk}[\mathsf{id}'], (\mathsf{hk}_{\mathsf{id}'}, \mathsf{hp}_{\mathsf{id}'}))$ for SPHF on $\mathfrak{L}^c_{\mathsf{id}'}$
   - $H_{\mathsf{id}'}$

   Sends $(\mathsf{hp}_{\mathsf{id}'}, \mathsf{usk}[\mathsf{id}'] \oplus \mathtt{KDF}(H_{\mathsf{id}'}))$ for every id′

BIBE Generic Construction



1. User: $C = \mathsf{Encrypt}_{\mathrm{cca}}(\mathsf{id}; \rho)$

2. For every id′ $\mathcal{S}$ computes:
   - $(\mathsf{usk}[\mathsf{id}'], (\mathsf{hk}_{\mathsf{id}'}, \mathsf{hp}_{\mathsf{id}'}))$ for SPHF on $\mathfrak{L}^c_{\mathsf{id}'}$
   - $H_{\mathsf{id}'}$

   Sends $(\mathsf{hp}_{\mathsf{id}'}, \mathsf{usk}[\mathsf{id}'] \oplus \mathtt{KDF}(H_{\mathsf{id}'}))$ for every id′

3. User computes $H'_{\mathsf{id}} = \mathsf{ProjHash}(\mathsf{hp}_{\mathsf{id}}, (\mathfrak{L}^c_{\mathsf{id}}, \mathsf{param}), C, \rho)$
   Recovers $\mathsf{usk}[\mathsf{id}]$

3-flow-Adaptive Oblivious Transfer

Database Preparation:
Data encryption, usk computation, channel key generation.

3-flow-Adaptive Oblivious Transfer

Database Preparation:
Data encryption, usk computation, channel key generation.

Index query on s:
Secure channel creation, s commitment computation (keeping rand).

3-flow-Adaptive Oblivious Transfer

Database Preparation:
Data encryption, usk computation, channel key generation.

Index query on s:
Secure channel creation, s commitment computation (keeping rand).

IBE input msk:
BlindUSKGen computation, blind key transmission.

3-flow-Adaptive Oblivious Transfer

Database Preparation:
Data encryption, usk computation, channel key generation.

Index query on s:
Secure channel creation, s commitment computation (keeping rand).

IBE input msk:
BlindUSKGen computation, blind key transmission.

Data recovery:
usk$[s]$ computation (using rand), data recovering.

3-flow-Adaptive Oblivious Transfer

Database Preparation:
Data encryption, usk computation, channel key generation.

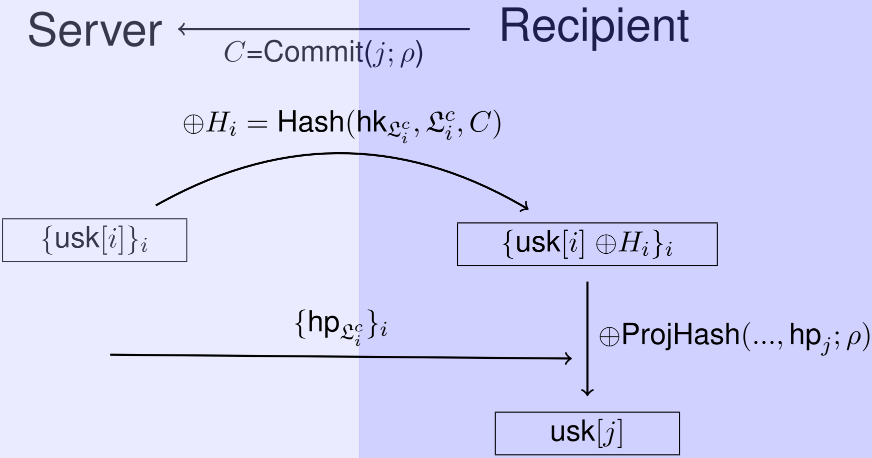
Index query on s:
Secure channel creation, s commitment computation (keeping rand).

IBE input msk:
BlindUSKGen computation, blind key transmission.

Data recovery:
usk$[s]$ computation (using rand), data recovering.

Almost...

What About The Communication Cost?



Server

Recipient

C=Commit$(j;\rho)$

$\oplus H_i = \mathsf{Hash}(\mathsf{hk}_{\mathfrak{L}_i^c}, \mathfrak{L}_i^c, C)$

$\{\mathsf{usk}[i]\}_i$

$\{\mathsf{usk}[i] \oplus H_i\}_i$

$\{\mathsf{hp}_{\mathfrak{L}_i^c}\}_i$

$\oplus\mathsf{ProjHash}(\ldots, \mathsf{hp}_j; \rho)$

$\mathsf{usk}[j]$

Issue With The Communication Cost

Problem: need as many languages as the number of identities.

$\Rightarrow$ communication cost linear in the size of the database.

Issue With The Communication Cost

Problem: need as many languages as the number of identities.

⇒ communication cost linear in the size of the database.

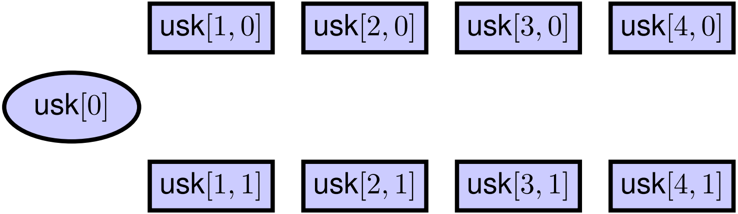
Solution: fragment identities into bits.

⇒ communication cost logarithmic in the size of the database.

Affine IBE

$$\mathsf{usk}[\mathsf{id}] = \mathsf{usk}[0] \oplus \left(\bigoplus_i \mathsf{usk}[i, \mathsf{id}_i] \right)$$

Example with id $= 0010$



$\mathsf{usk}[1,0]$ $\mathsf{usk}[2,0]$ $\mathsf{usk}[3,0]$ $\mathsf{usk}[4,0]$

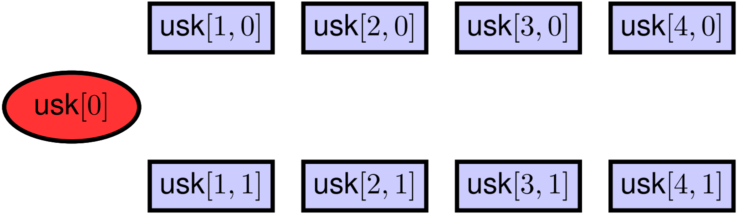
$\mathsf{usk}[0]$

$\mathsf{usk}[1,1]$ $\mathsf{usk}[2,1]$ $\mathsf{usk}[3,1]$ $\mathsf{usk}[4,1]$

Affine IBE

$$\mathsf{usk}[\mathsf{id}] = \mathsf{usk}[0] \oplus \left(\bigoplus_i \mathsf{usk}[i, \mathsf{id}_i] \right)$$

Example with $\mathsf{id} = 0010$



$\mathsf{usk}[1,0]$ $\mathsf{usk}[2,0]$ $\mathsf{usk}[3,0]$ $\mathsf{usk}[4,0]$

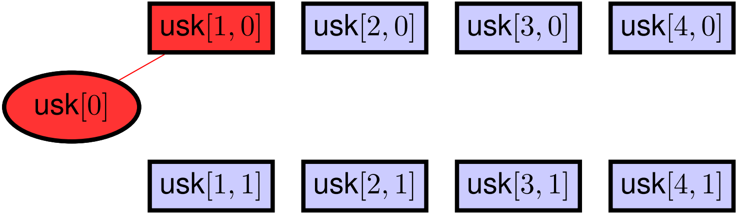
$\mathsf{usk}[0]$

$\mathsf{usk}[1,1]$ $\mathsf{usk}[2,1]$ $\mathsf{usk}[3,1]$ $\mathsf{usk}[4,1]$

Affine IBE

$$\mathsf{usk}[\mathsf{id}] = \mathsf{usk}[0] \oplus \left(\bigoplus_i \mathsf{usk}[i, \mathsf{id}_i] \right)$$

Example with id $= 0010$



$\mathsf{usk}[1,0]$ $\mathsf{usk}[2,0]$ $\mathsf{usk}[3,0]$ $\mathsf{usk}[4,0]$

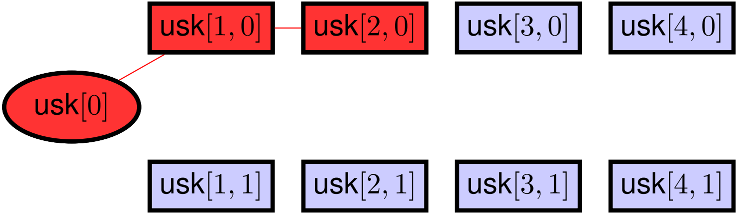
$\mathsf{usk}[0]$

$\mathsf{usk}[1,1]$ $\mathsf{usk}[2,1]$ $\mathsf{usk}[3,1]$ $\mathsf{usk}[4,1]$

Affine IBE

$$\mathsf{usk}[\mathsf{id}] = \mathsf{usk}[0] \oplus \left(\bigoplus_i \mathsf{usk}[i, \mathsf{id}_i] \right)$$

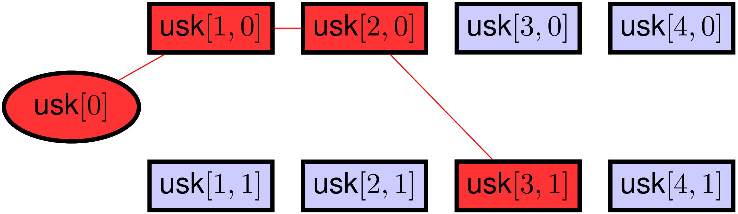
Example with $\mathsf{id} = 0010$



Affine IBE

$$\mathsf{usk}[\mathsf{id}] = \mathsf{usk}[0] \oplus \left(\bigoplus_i \mathsf{usk}[i, \mathsf{id}_i] \right)$$

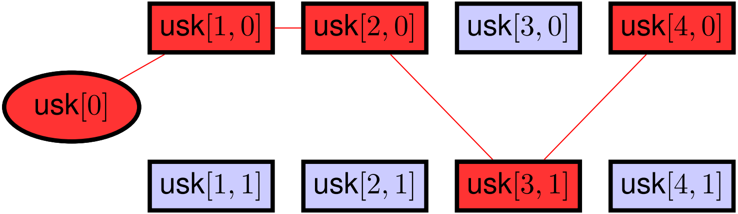
Example with $\mathsf{id} = 0010$



Affine IBE

$$\mathsf{usk}[\mathsf{id}] = \mathsf{usk}[0] \oplus \left(\bigoplus_i \mathsf{usk}[i, \mathsf{id}_i] \right)$$

Example with id $= 0010$



Fragmented BIBE Construction

2. For all i, b S computes:
   - $(\mathsf{usk}[i,b], (\mathsf{hk}_{i,b}, \mathsf{hp}_{i,b}))$ for SPHF on $\mathfrak{L}^c_{i,b}$
   - $H_{i,b}$
   - $Z = \mathsf{usk}[0] \ominus \left(\bigoplus_i z_i \right)$

   Sends $(Z, \mathsf{hp}_{i,b}, \mathsf{usk}[i,b] \oplus \mathtt{KDF}(H_{i,b}) \oplus z_i)$ for each (i, b)

Fragmented BIBE Construction

2. For all i, b S computes:
   - $(\mathsf{usk}[i,b], (\mathsf{hk}_{i,b}, \mathsf{hp}_{i,b}))$ for SPHF on $\mathfrak{L}^c_{i,b}$
   - $H_{i,b}$
   - $Z = \mathsf{usk}[0] \ominus \left(\bigoplus_i z_i\right)$

   Sends $(Z, \mathsf{hp}_{i,b}, \mathsf{usk}[i,b] \oplus \mathtt{KDF}(H_{i,b}) \oplus z_i)$ for each (i, b)

3. User computes $H'_{i,b} = \mathsf{ProjHash}(\mathsf{hp}_{i,b}, (\mathfrak{L}^c_{i,b}, \mathsf{param}), C, \rho)$

   Recovers $\mathsf{usk}[\mathsf{id}] = \left(\bigoplus_i (\mathsf{usk}[i, \mathsf{id}_i] \oplus z_i)\right) \oplus Z$

An Affine IBKEM Scheme

- **Gen** : $Y_i \xleftarrow{\$} \mathbb{Z}_p^2$, $Z_i = Y_i^\top \cdot A$, $y' \xleftarrow{\$} \mathbb{Z}_p^2$, $z' = y'^\top \cdot A$,
  $\Rightarrow$ mpk $= (g_1^A, g_1^{Z_i}, g_1^{z'})$, msk $=(Y_i, y')$
- **USKGen**: $s \xleftarrow{\$} \mathbb{Z}_p$, $t = Bs$, $w = (Y_0 + \sum h_i(id_i)Y_i)t + y'$,
  $\Rightarrow$ usk[id] $= (g_2^t, g_2^w)$
- **Enc**(**mpk**, **id**): $r \xleftarrow{\$} \mathbb{Z}_p$, $c_0 = Ar$, $c_1 = (Z_0 + \sum h_i(id_i)Z_i) \cdot r$,
  $\Rightarrow K = z' \cdot r$,$C = (g_1^{c_0}, g_1^{c_1})$, sk $= g_T^K$
- **Dec**(**usk**[**id**], **id**, C): sk $=e(g_1^{c_0}, g_2^t) \cdot e(g_1^{c_1}, g_2^w)^{-1}$

1. Oblivious Transfer
2. OLBE: A Natural Generalization
3. Adaptive Oblivious Transfer
4. What To Remember

The Talk In One Slide

The Talk In One Slide

- $\{$OSBE, OT$\} \subset$ OLBE

The Talk In One Slide

- $\{$OSBE, OT$\} \subset$ OLBE
- Affine IBE + OT $\Rightarrow$ Fragmented BIBE

The Talk In One Slide

- $\{$OSBE, OT$\} \subset$ OLBE
- Affine IBE + OT $\Rightarrow$ Fragmented BIBE
- Fragmented BIBE + UC folklore $\Rightarrow$ UC secure Adaptive OT