Fine-Grained Secure Attribute-Based Encryption

Yuyu Wang[1], Jiaxin pan[2], Yu Chen[3]

1. University of Electronic Science and Technology of China
2. NTNU - Norwegian University of Science and Technology
3. Shandong University

Standard cryptography

Honest party

polynomial-time

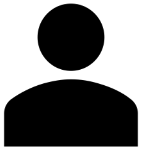
Adversary

polynomial-time

Assumption:
- Basic ones (e.g., one-way function)
- More advanced ones (e.g., factoring, discrete logarithm, DDH, LWE)
- Exotic ones (e.g, generic groups, algebraic groups)

Standard cryptography

Honest party



polynomial-time

Adversary

polynomial-time

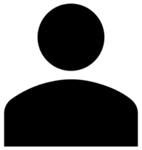
Unproven

Assumption:

- Basic ones (e.g., one-way function)
- More advanced ones (e.g., factoring, discrete logarithm, DDH, LWE)
- Exotic ones (e.g, generic groups, algebraic groups)

Fine-grained cryptography

Honest party

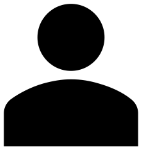


Adversary

An honest party uses less resources than the adversary

Fine-grained cryptography

Honest party



An honest party uses less resources than the adversary

Adversary

The resources of an adversary can be a-prior bounded

Fine-grained cryptography

Honest party

An honest party uses less resources than the adversary

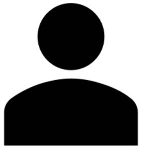
Adversary

The resources of an adversary can be a-prior bounded

- Based only on mild assumption

Fine-grained cryptography

Honest party



An honest party uses less resources than the adversary

Adversary

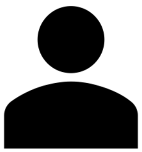
The resources of an adversary can be a-prior

Existing fine-grained primitives:
NIKE [Mer78], OWF [BC20], PKE [DVV16], verifiable computation [CG18], HPS [EWT19], trapdoor one-way functions [EWT21]

- Based only on mild assumption

Fine-grained cryptography

Honest party



An honest party uses less resources than the adversary

Adversary

The resources of an adversary can be a-prior

Existing fine-grained primitives:
NIKE [Mer78], OWF [BC20], PKE [DVV16], verifiable computation [CG18], HPS [EWT19], trapdoor one-way functions [EWT21]
(signature is not among them)

- Based only on mild assumption

Our results

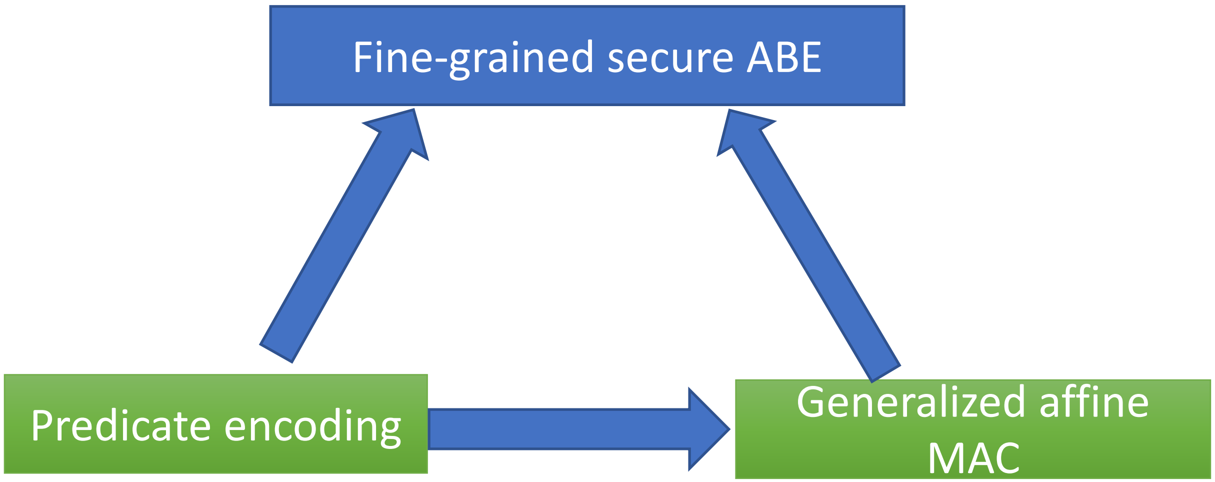
Fine-grained secure ABE

Our results

Fine-grained secure ABE

IBE from affine MAC [BKP14]
+
ABE from predicate encodings [CGW15]

Our results

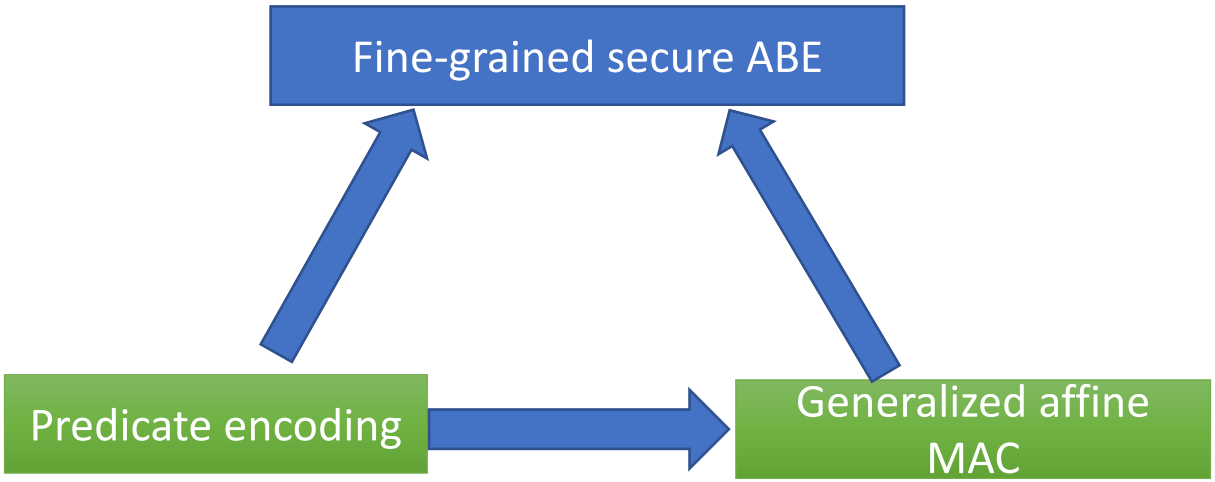


IBE from affine MAC [BKP14]

+

ABE from predicate encodings [CGW15]

Our results



A dual system in the fine-grained-setting

IBE

ABE from predicate encodings [CGW15]

Our results

By suitably instantiating the underlying predicate encoding, we obtain:

1. IBE scheme (which in turn implies a signature scheme)

2. ABEs for
   a. inner-product encryption
   b. non-zero inner-product
   c. encryption spatial encryption
   d. doubly spatial encryption
   e. boolean span programs
   f. arithmetic span programs

3. Broadcast encryption

4. fuzzy IBE

in the fine-grained setting.

Our results

All of the instantiations are computable in AC0[2] and secure against adversaries in NC^1

Our results

All of the instantiations are computable in AC0[2] and secure against adversaries in NC^1 under the assumption: $\mathrm{NC}^1 \neq \oplus \mathrm{L/poly}$.

Our results

All of the instantiations are computable in AC0[2] and secure against adversaries in NC^1 under the assumption: $\mathrm{NC}^1 \neq \oplus \mathrm{L/poly}$. All the computations are over GF(2).

Our results

All of the instantiations are computable in AC0[2] and secure against adversaries in NC^1 under the assumption: $NC^1 \neq \oplus L/poly$. All the computations are over GF(2).

(same as the bounded-circuit setting in previous works on fine-grained cryptography [DVV16,CG18,EWT19])

Our results

All of the instantiations are computable in AC0[2] and secure against adversaries in NC^1 under the assumption: $\mathrm{NC}^1 \neq \oplus \mathrm{L/poly}$. All the computations are over GF(2).

(same as the bounded-circuit setting in previous works on fine-grained cryptography [DVV16,CG18,EWT19])

Circuits with constant depth, polynomial size, and unbounded fan-in using AND, OR, NOT, and PARITY gates.

Our results

All of the instantiations are computable in AC0[2] and secure against adversaries in NC^1 under the assumption: $NC^1 \neq \oplus L/poly$. All the computations are over GF(2).

(same as the bounded-circuit setting in previous works on fine-grained cryptography [DVV16,CG18,EWT19])

Circuits with logarithmic depth, polynomial size, and fan-in 2 gates.

Our results

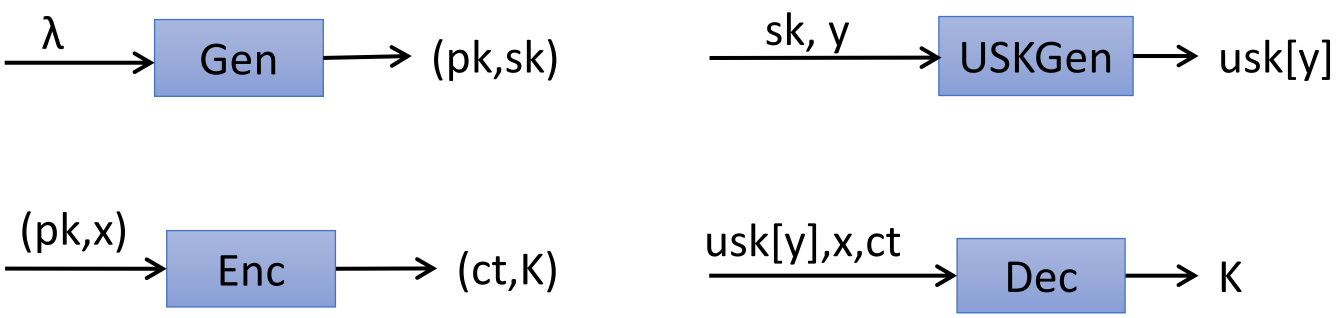
All of the instantiations are computable in AC0[2] and secure against adversaries in NC^1 under the assumption: $NC^1 \neq \oplus L/poly$. All the computations are over GF(2).

(same as the bounded-circuit setting in previous works on fine-grained cryptography [DVV16,CG18,EWT19])

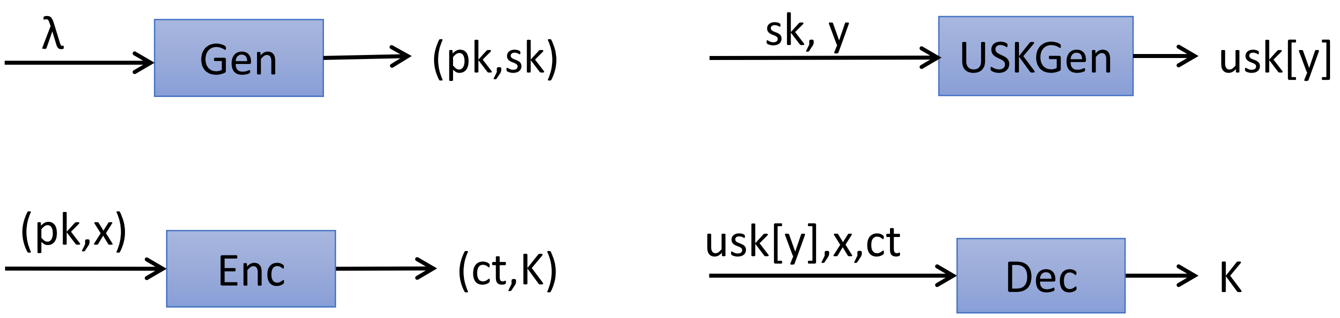
Log space turing machine with parity acceptance.

Attribute-based key encapsulation (ABKEM)

Attribute-based key encapsulation (ABKEM)

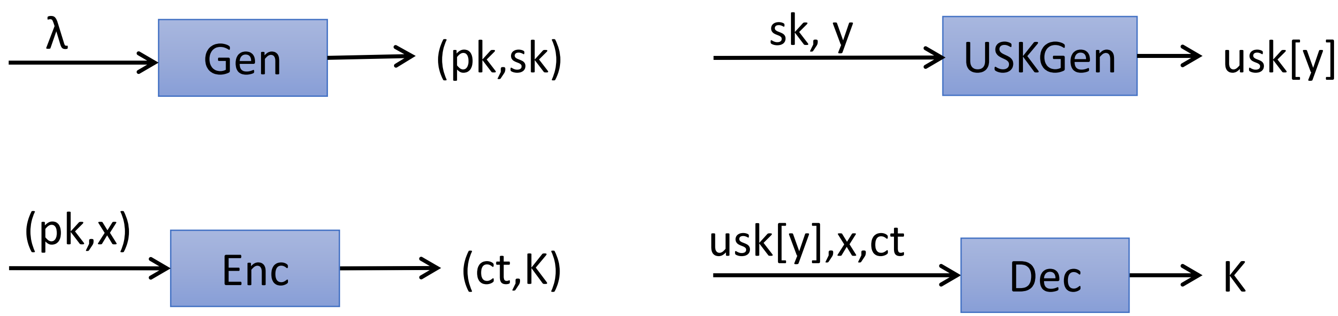


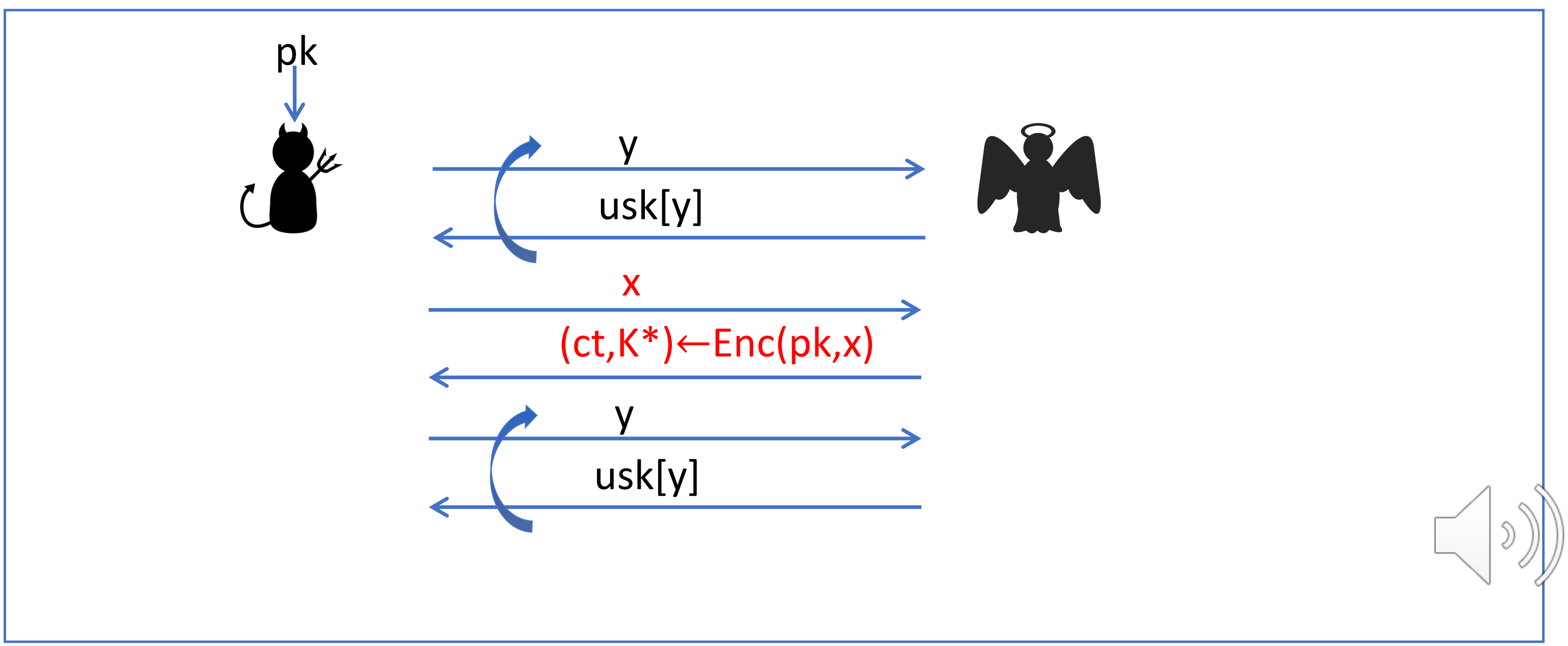
Attribute-based key encapsulation (ABKEM)



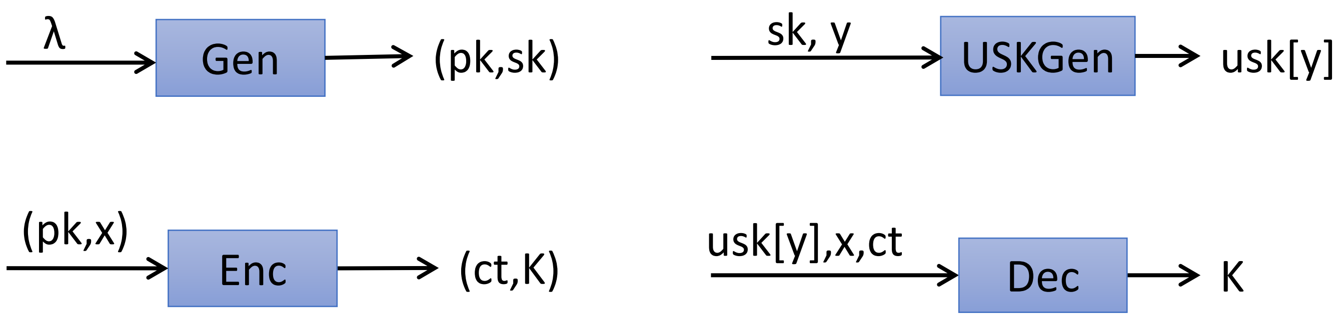
Correctness: K can be correctly recovered by Dec if p(x,y)=1

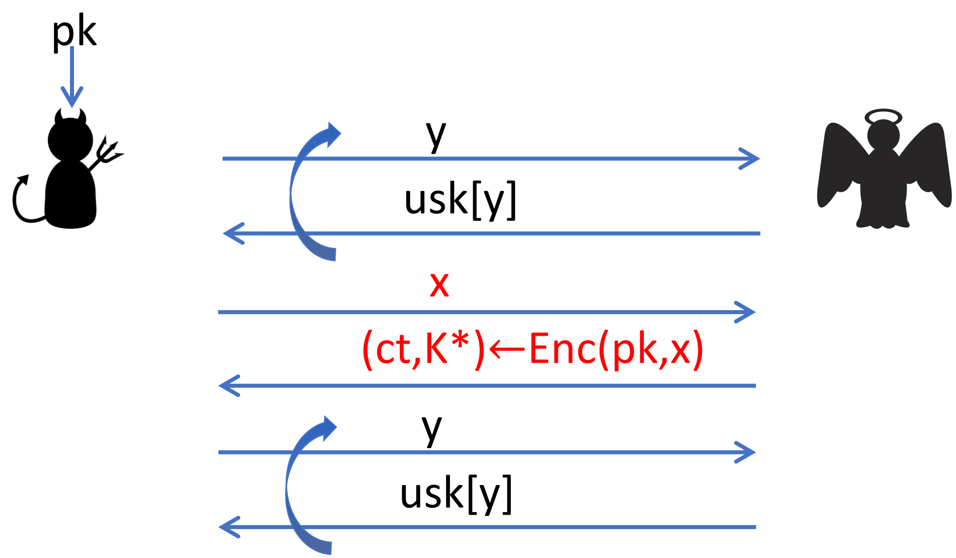
Attribute-based key encapsulation (ABKEM)





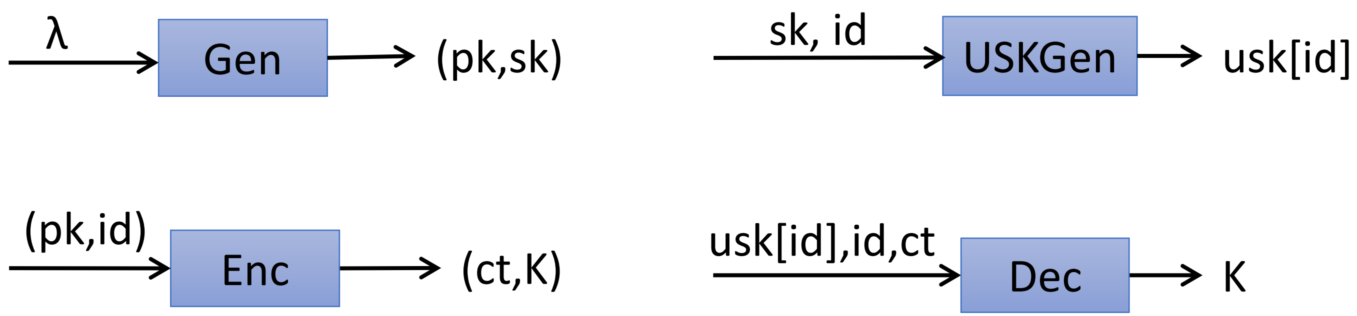
Attribute-based key encapsulation (ABKEM)

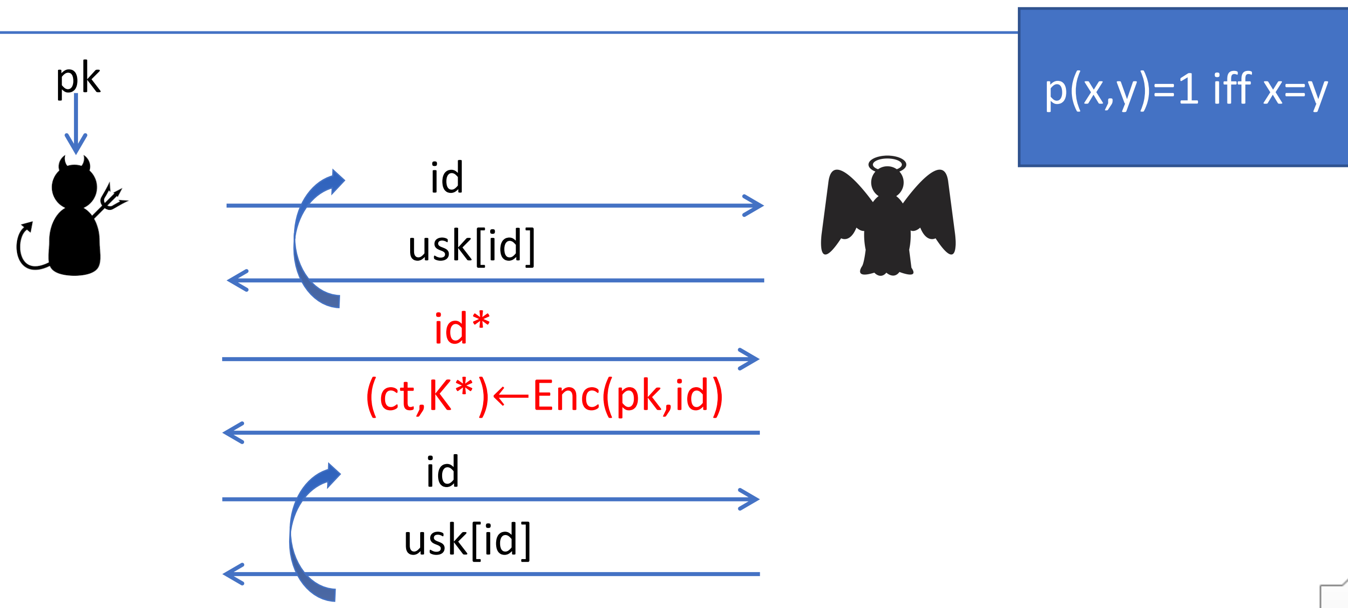




Security: K* is indistinguishable from randomness when $p(x,y) \neq 1$ for all queried y

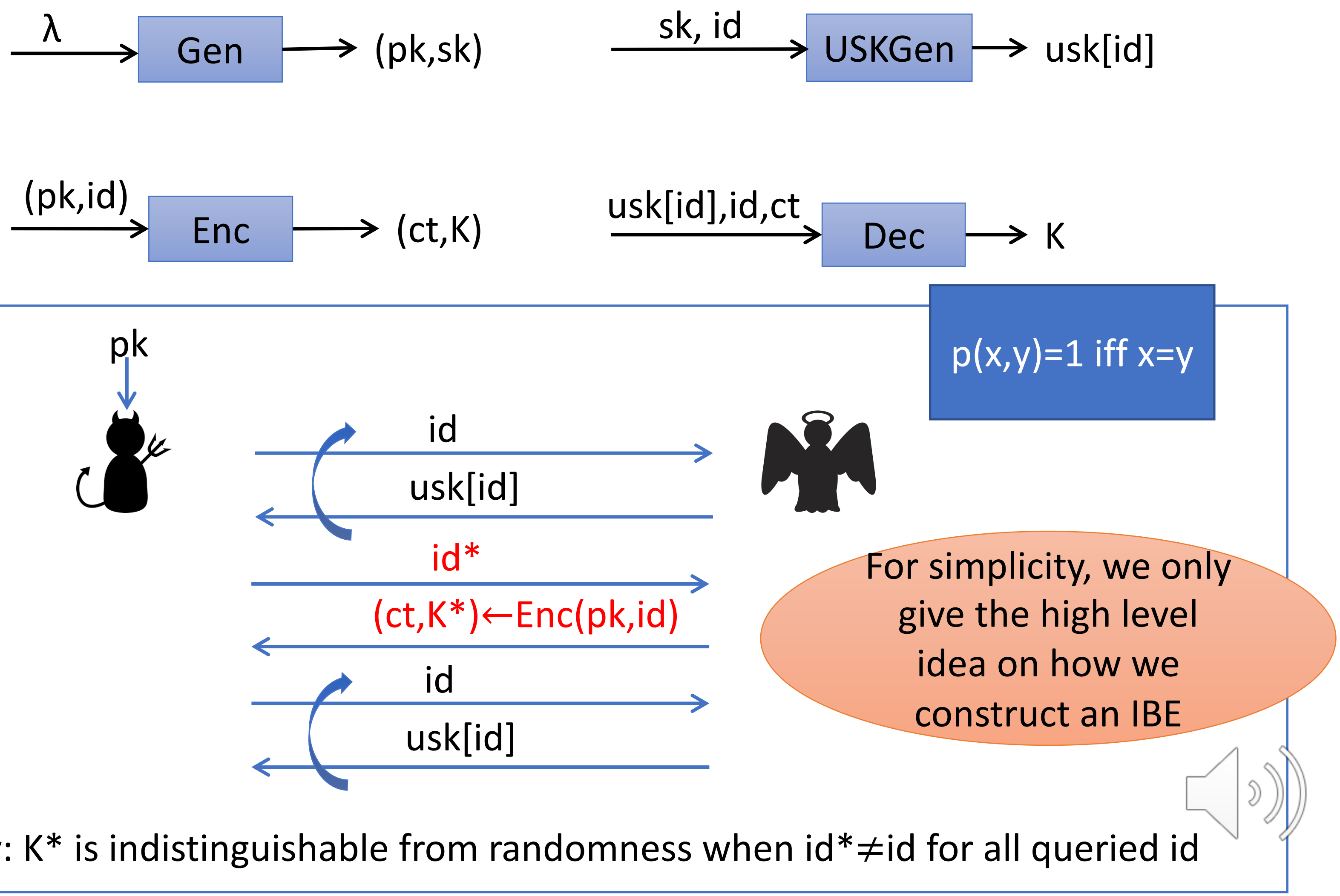
Identity-based key encapsulation (IBKEM)



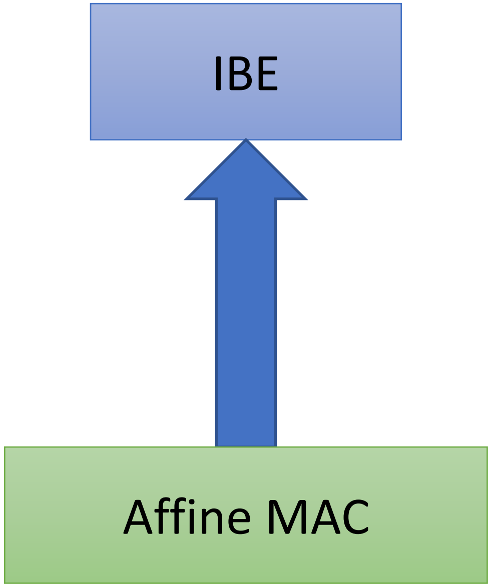


Security: K* is indistinguishable from randomness when id*≠id for all queried id

Identity-based key encapsulation (IBKEM)

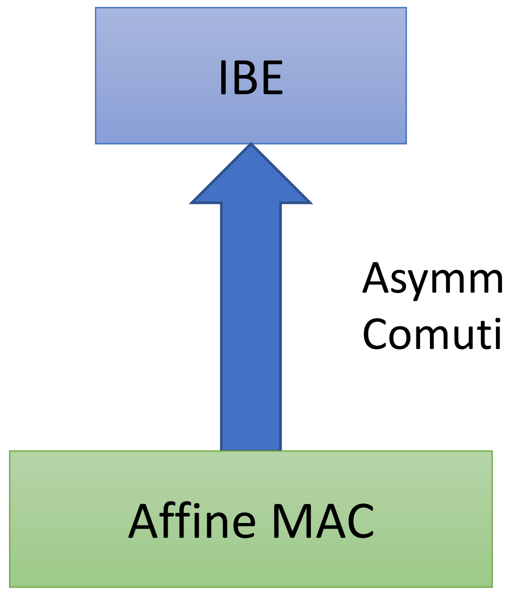


The BKP framework



MDDH assumption

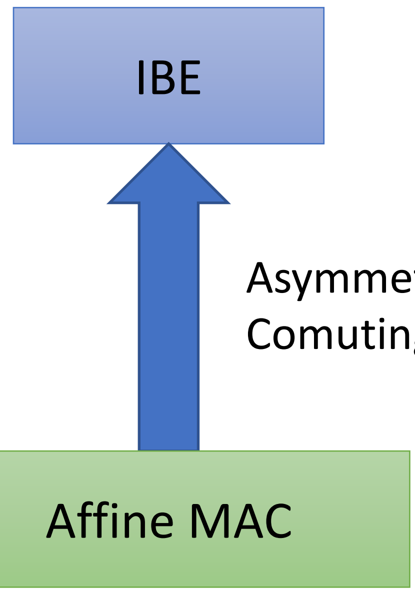
The BKP framework



Asymmetric pairing
Comuting the inverse of a matrix

MDDH assumption

The BKP framework

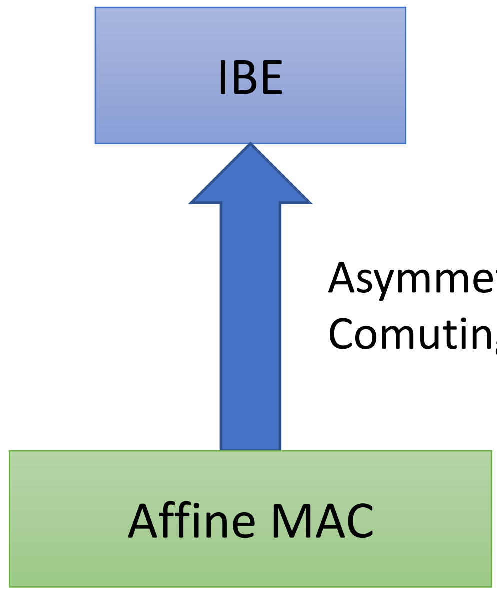


Asymmetric pairing ✗
Comuting the inverse of a matrix

MDDH assumption ✗

We have no pairing and MDDH assumption in NC^1

The BKP framework



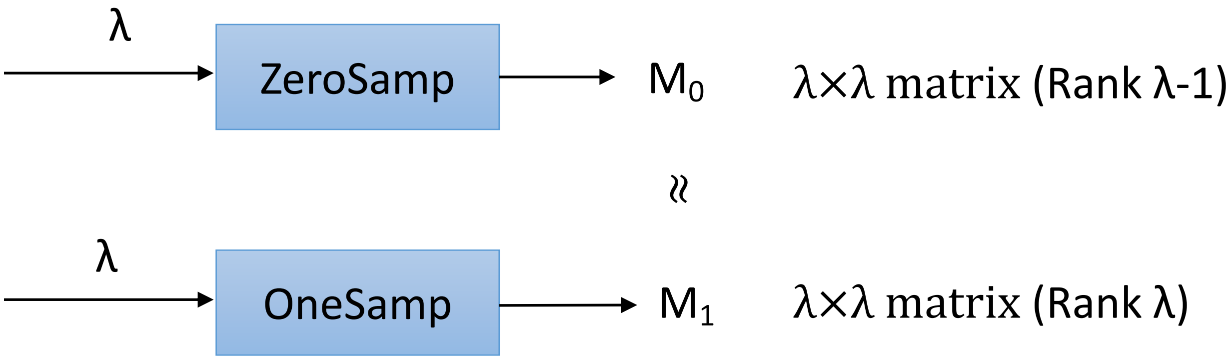
Asymmetric pairing ❌
Comuting the inverse of a matrix ❌

MDDH assumption ❌

We have no pairing and MDDH assumption in NC^1

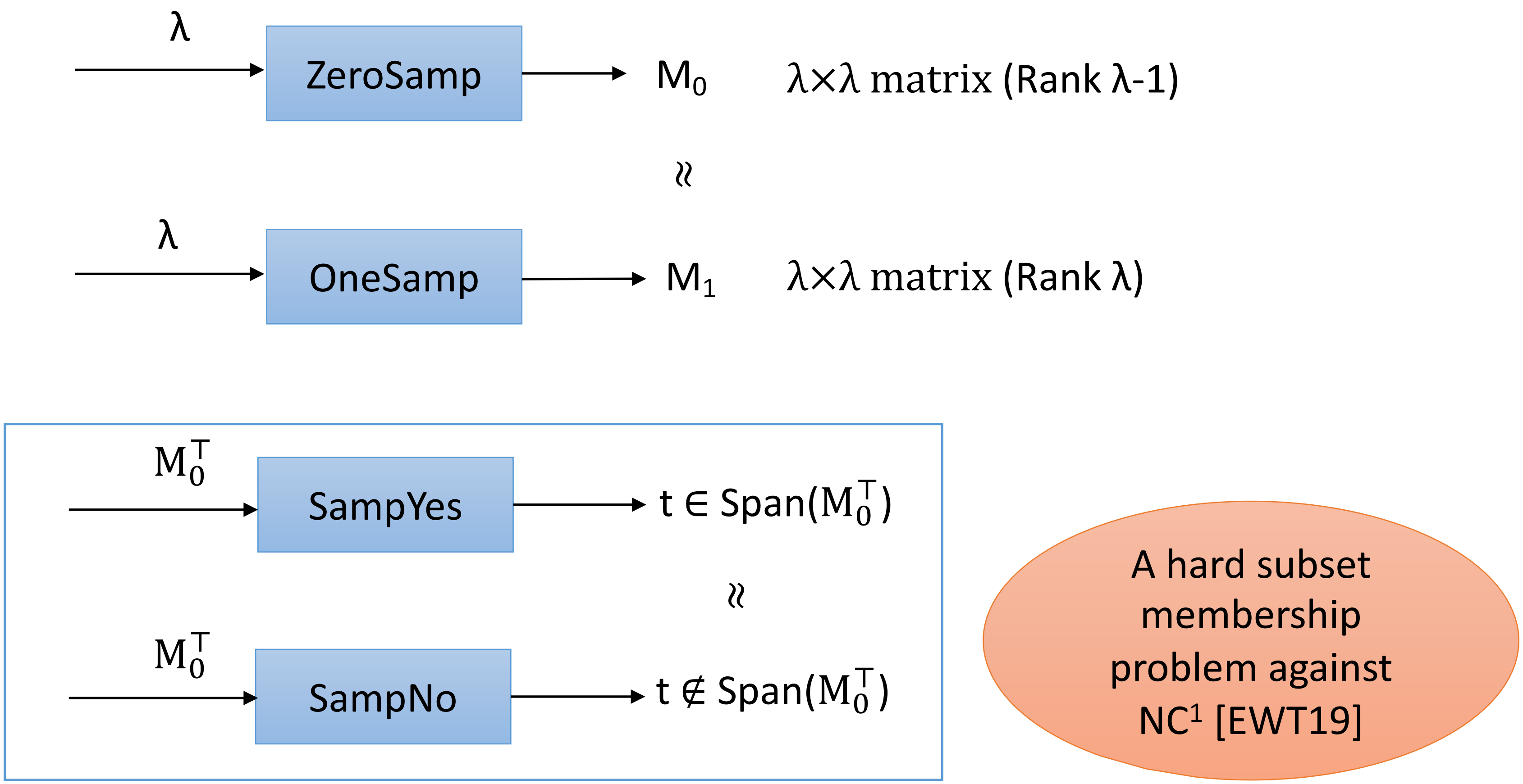
We cannot compute the inverse of of a matrix in NC^1

A counter part of the MDDH assumption

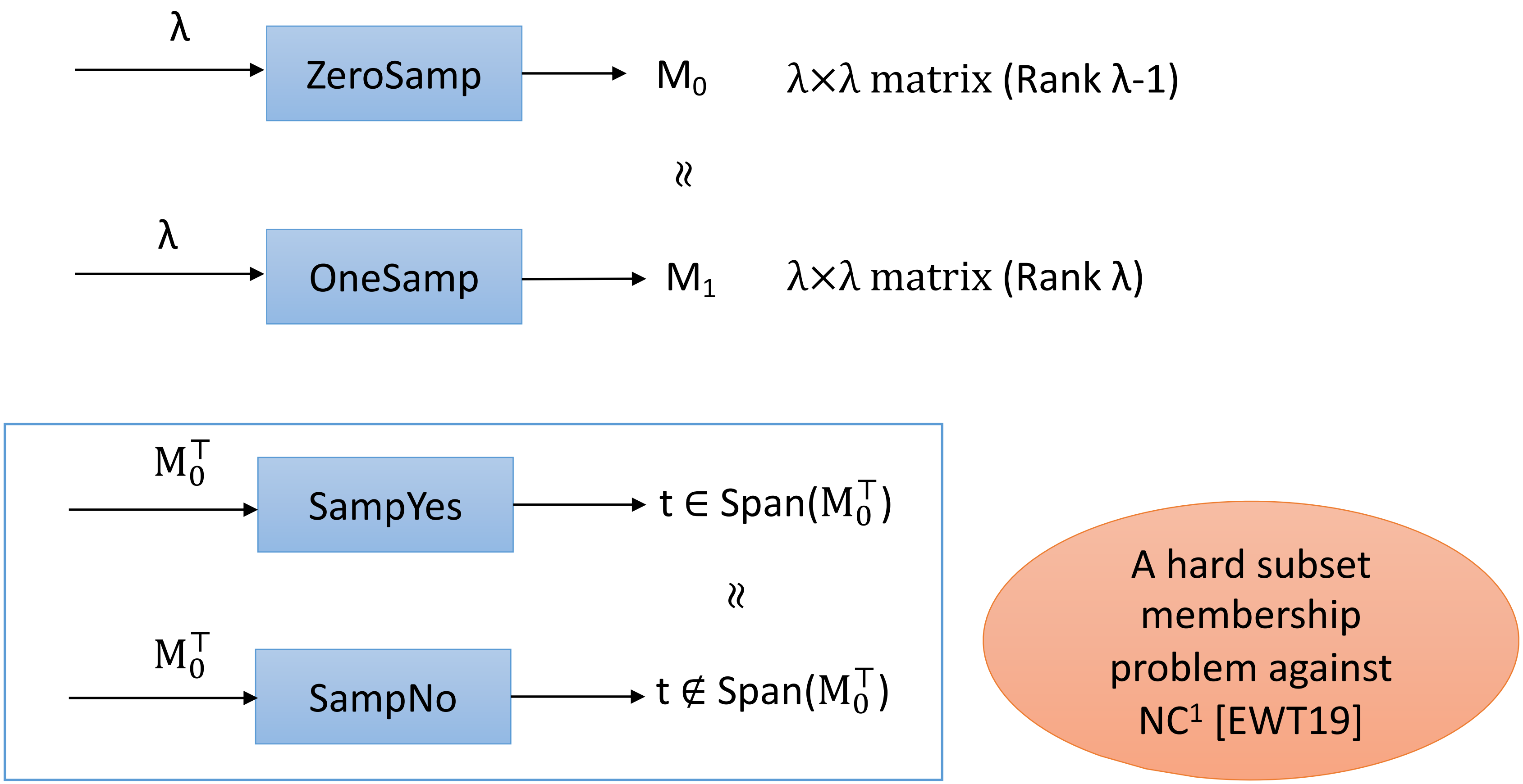


Indistinguishable against NC1 adversaries if $\mathrm{NC}^1 \neq \oplus \mathrm{L/poly}$ [DVV16]

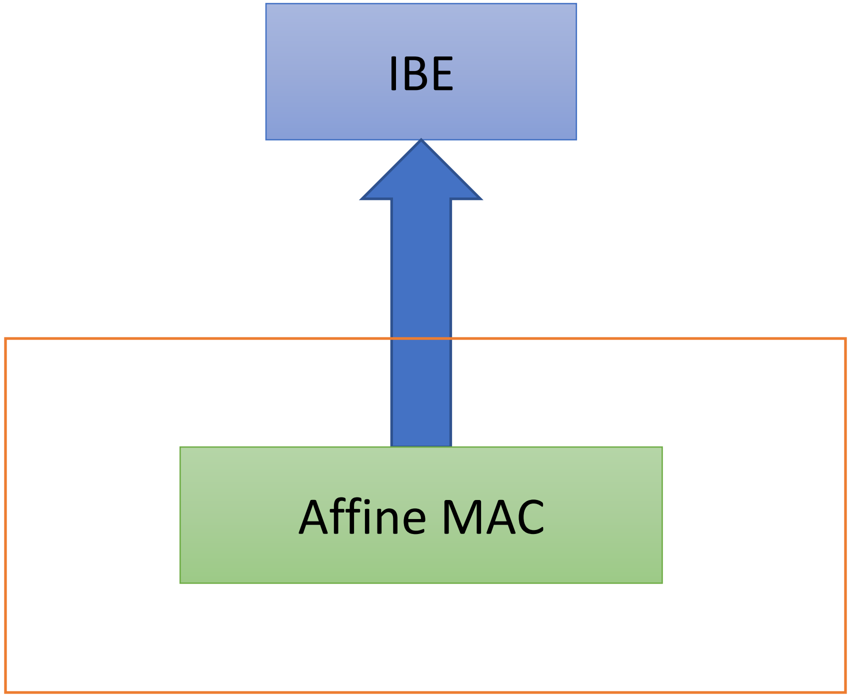
A counter part of the MDDH assumption



A counter part of the MDDH assumption



Affine MAC



Affine MAC

$Gen_{MAC}(\lambda)$:

- $B^{\top} \leftarrow ZeroSamp(\lambda)$
- $x_i \leftarrow \{0,1\}^{\lambda}$ for i=0, ..., n
- $x' \leftarrow \{0,1\}$

$sk_{MAC}=(B,x_0,\ldots, x_n,x')$

Return ε

Affine MAC

$\mathrm{Gen}_{MAC}(\lambda)$:
- $B^{\top} \leftarrow \mathrm{ZeroSamp}(\lambda)$
- $x_i \leftarrow \{0,1\}^{\lambda}$ for i=0, ..., n
- $x' \leftarrow \{0,1\}$

$sk_{MAC}=(B,x_0,\ldots, x_n,x')$
Return ε

$\mathrm{Tag}(sk_{MAC}, id=(id_i)_{i=1,..,n})$:
$t \leftarrow \mathrm{SampYes}(B)$
$$u = x_0^{\top} t + \sum id_i x_i^{\top} t + x'$$
Return (t,u)

Affine MAC

$\text{Gen}_{\text{MAC}}(\lambda)$:

- $B^{\top} \leftarrow \text{ZeroSamp}(\lambda)$
- $x_i \leftarrow \{0,1\}^{\lambda}$ for $i=0, \ldots, n$
- $x' \leftarrow \{0,1\}$

$sk_{\text{MAC}}=(B,x_0,\ldots, x_n,x')$

Return ε

$\text{Tag}(sk_{\text{MAC}}, id=(id_i)_{i=1,\ldots,n})$:

$t \leftarrow \text{SampYes}(B)$

$u = x_0^{\top} t + \sum id_i x_i^{\top} t + x'$

Return (t,u)

Affine equation of $x_i^{\top} t$ and x' with coefficients derived from the message

Affine MAC (security)

$\mathrm{Gen_{MAC}}(\lambda)$:

- $B^\top \leftarrow \mathrm{ZeroSamp}(\lambda)$
- $x_i \leftarrow \{0,1\}^\lambda$ for i=0, …, n
- $x' \leftarrow \{0,1\}$

$sk_{MAC}=(B,x_0,\ldots, x_n,x')$

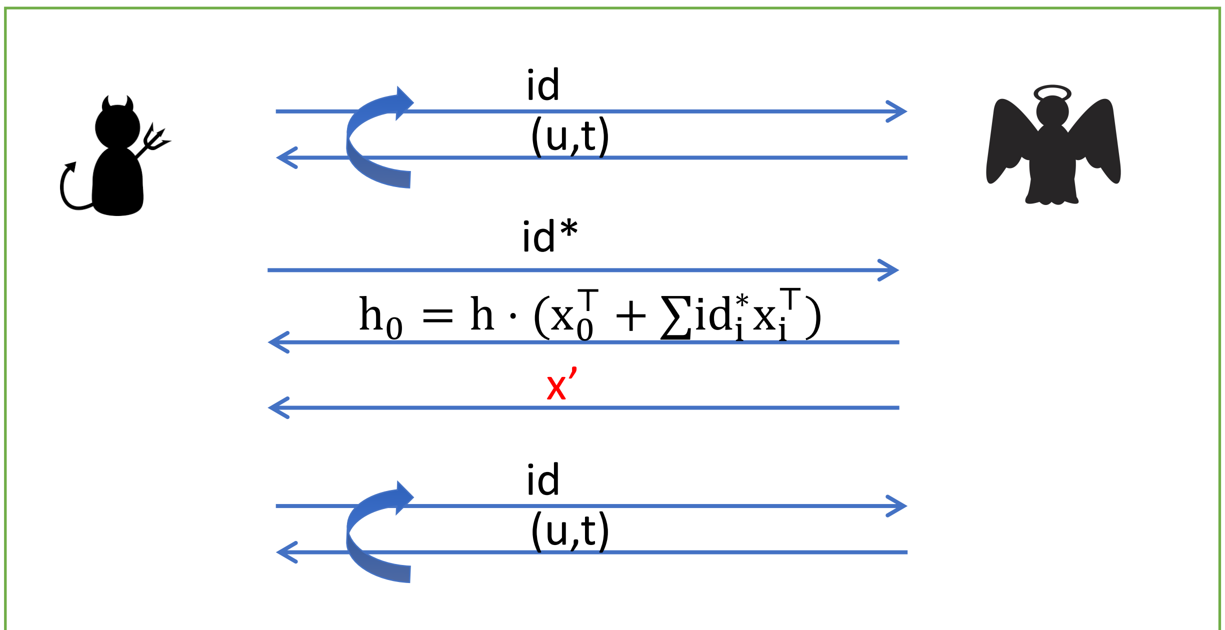
Return ε

$\mathrm{Tag}(sk_{MAC}, id=(id_i)_{i=1,..,n})$:

$t \leftarrow \mathrm{SampYes}(B)$

$u = x_0^\top t + \sum id_i x_i^\top t + x'$

Return (t,u)

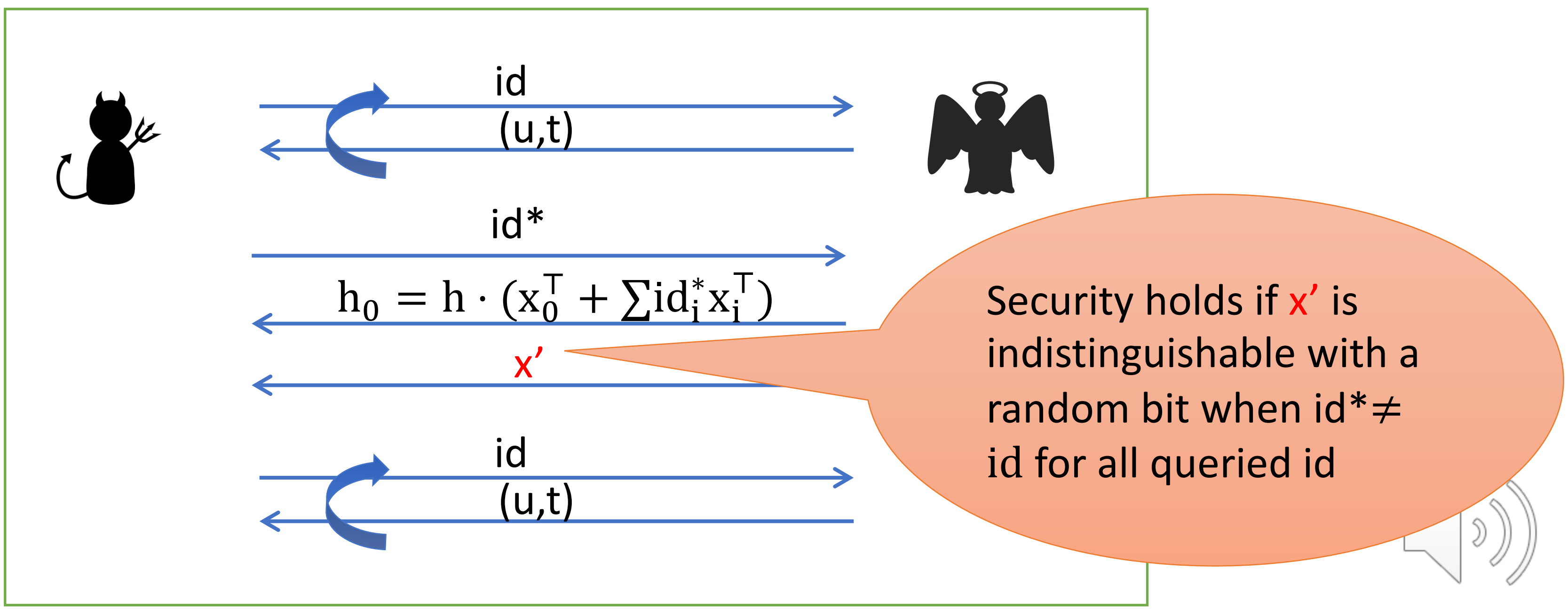


Affine MAC (security)

$Gen_{MAC}(\lambda)$:
- $B^{\top} \leftarrow ZeroSamp(\lambda)$
- $x_i \leftarrow \{0,1\}^{\lambda}$ for i=0, ..., n
- $x' \leftarrow \{0,1\}$

$sk_{MAC}=(B, x_0, \ldots, x_n, x')$
Return ε

$Tag(sk_{MAC}, id=(id_i)_{i=1,\ldots,n})$:
$t \leftarrow SampYes(B)$
$u = x_0^{\top} t + \sum id_i x_i^{\top} t + x'$
Return (t,u)



Affine MAC (security)

$\text{Gen}_{MAC}(\lambda)$:

- $B^{\top} \leftarrow \text{ZeroSamp}(\lambda)$
- $x_i \leftarrow \{0,1\}^{\lambda}$ for i=0, …, n
- $x' \leftarrow \{0,1\}$

$sk_{MAC}=(B,x_0,\ldots, x_n,x')$

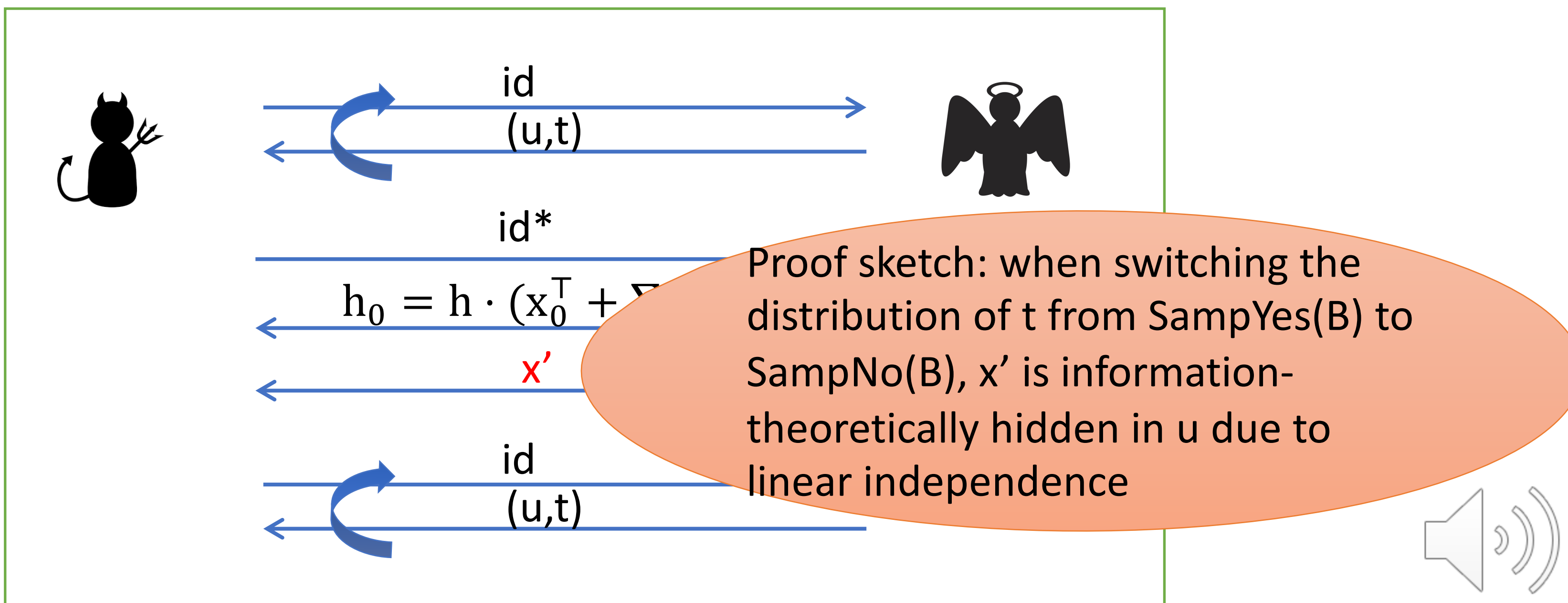
Return ε

$\text{Tag}(sk_{MAC}, id=(id_i)_{i=1,..,n})$:

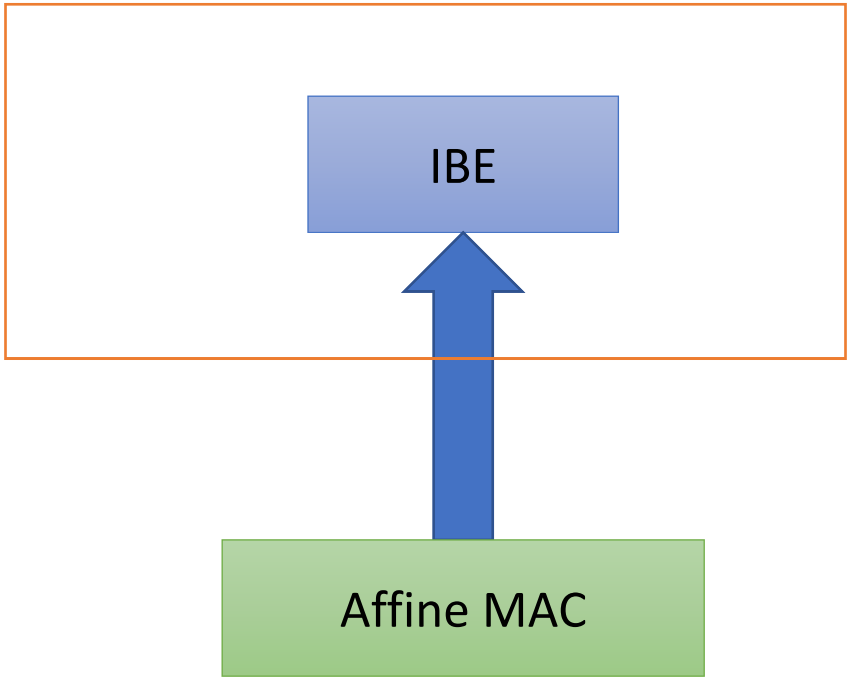
$t \leftarrow \text{SampYes}(B)$

$u = x_0^{\top} t + \sum id_i x_i^{\top} t + x'$

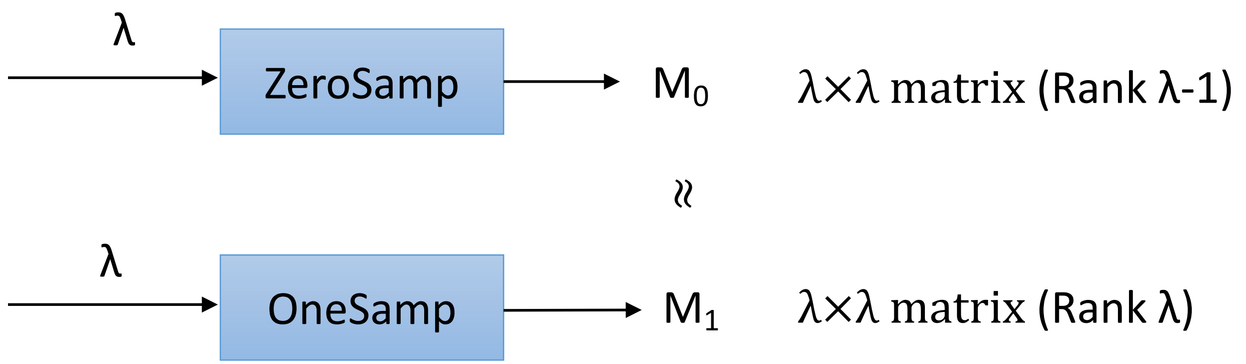
Return (t,u)



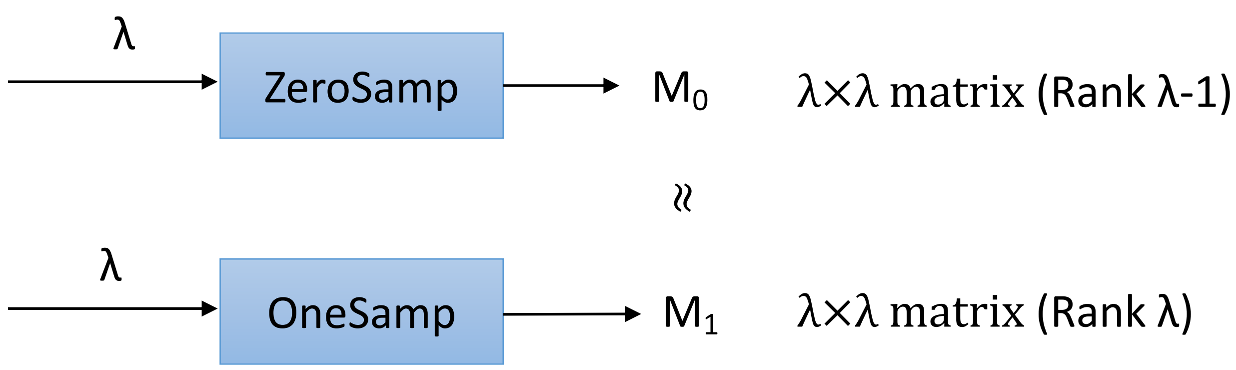
Construction of IBKEM



Two facts on ZeroSamp and OneSamp [EWT19]



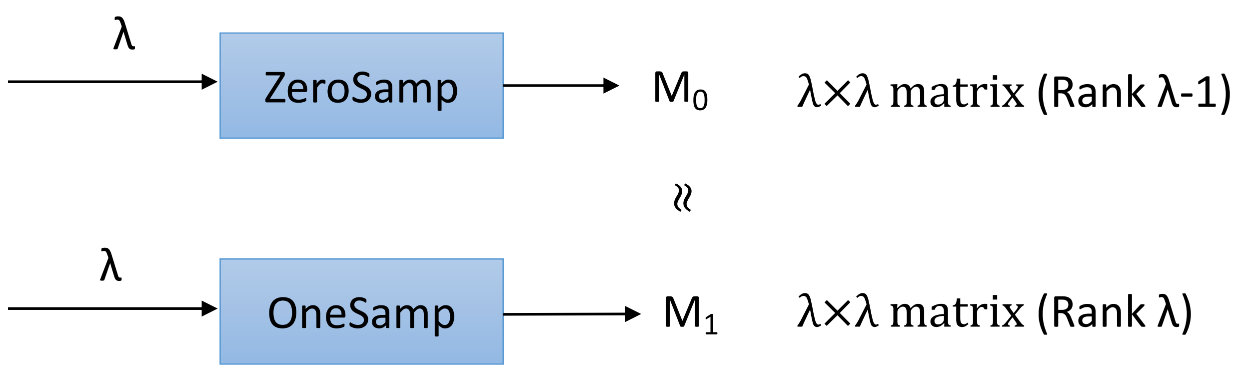
Two facts on ZeroSamp and OneSamp [EWT19]



The distribution of $M_0^{\top}$ +N is identical to that of $M_1^{\top}$

$$N = \begin{pmatrix} 0 & \cdots & & 0 \\ \vdots & 0 & \cdots & 0 \\ 0 & & \ddots & \vdots \\ 1 & 0 & \cdots & 0 \end{pmatrix}$$

Two facts on ZeroSamp and OneSamp [EWT19]



The distribution of M_0^T +N is identical to that of M_1^T

$$N = \begin{pmatrix} 0 & \cdots & & 0 \\ \vdots & 0 & \cdots & 0 \\ 0 & & \ddots & \vdots \\ 1 & 0 & \cdots & 0 \end{pmatrix}$$

The distributions of $M_0^T r_0$ and $M_0^T r_1$ are identical where

$r_0 \leftarrow \{0\}\times\{0,1\}^{\lambda-1}$ and $r_1 \leftarrow \{1\}\times\{0,1\}^{\lambda-1}$

Construction of IBKEM

Gen(λ):

- $A^{\top}$ ←ZeroSamp(λ), $sk_{MAC}=(B,x_0,\ldots,x_n,x')$ ←Gen_{MAC}
- For i=0,…,n, $Y_i \leftarrow \{0,1\}^{(\lambda-1)\times\lambda}$, $Z_i = (Y_i^{\top}||x_i)A$
- y'← $\{0,1\}^{(\lambda-1)}$, $z' = (y'^{\top}||x')A$

pk=$(A,(Z_i)_i,z')$, sk=$(sk_{MAC},(Y_i)_i,y')$

Construction of IBKEM

Gen(λ):

- A^T ←ZeroSamp(λ), sk_{MAC}=(B,x_0,…, x_n,x') ←Gen_{MAC}
- For i=0,…,n, $Y_i \leftarrow \{0,1\}^{(\lambda-1)\times\lambda}$, $Z_i = (Y_i^T||x_i)A$
- y'← $\{0,1\}^{(\lambda-1)}$, $z' = (y'^T||x')A$

pk=(A,$(Z_i)_i$,z'), sk=(sk_{MAC},$(Y_i)_i$,y')

Committing sk_{MAC}

Construction of IBKEM

Gen(λ):

- $A^{\top}$ ←ZeroSamp(λ), sk_{MAC}=(B,x_0,…, x_n,x’) ←Gen_{MAC}
- For i=0,…,n, $Y_i \leftarrow \{0,1\}^{(\lambda-1)\times\lambda}$, $Z_i = (Y_i^{\top}||x_i)A$
- y’← $\{0,1\}^{(\lambda-1)}$, $z' = (y'^{\top}||x')A$

pk=(A,$(Z_i)_i$,z’), sk=(sk_{MAC},$(Y_i)_i$,y’)

USKGen(sk_{MAC}, id):

- (t,u) ←Tag(sk_{MAC},id), $v = t^{\top}\left(Y_0^{\top} + \sum id_i Y_i^{\top}\right) + y'^{\top}$

usk[id]=(t,u,v)

Affine equation of
Y_it and y’
(a proof that the tag
was correctly
computed)

Construction of IBKEM

Gen(λ):

- $A^\top \leftarrow$ZeroSamp(λ), $sk_{MAC}=(B,x_0,\ldots,x_n,x') \leftarrow Gen_{MAC}$
- For i=0,…,n, $Y_i \leftarrow \{0,1\}^{(\lambda-1)\times\lambda}$, $Z_i = (Y_i^\top || x_i)A$
- $y' \leftarrow \{0,1\}^{(\lambda-1)}$, $z' = (y'^\top || x')A$

pk=(A,$(Z_i)_i$,z'), sk=(sk_{MAC},$(Y_i)_i$,y')

USKGen(sk_{MAC}, id):

- (t,u) ←Tag(sk_{MAC},id), $v = t^\top\left(Y_0^\top + \sum id_i Y_i^\top\right) + y'^\top$

usk[id]=(t,u,v)

Enc(pk, id):

- $r \leftarrow \{0\}\times\{0,1\}^{\lambda-1}$, c_0=Ar, $c_1 = (Z_0 + \sum id_i Z_i)r$

ct=(c_0,c_1), K=z'r

Affine equation of $Z_i r$

Construction of IBKEM

Gen(λ):

- $A^{\top}$ ←ZeroSamp(λ), sk_{MAC}=(B,x_0,…, x_n,x') ←Gen_{MAC}
- For i=0,…,n, $Y_i \leftarrow \{0,1\}^{(\lambda-1)\times\lambda}$, $Z_i = (Y_i^{\top}||x_i)A$
- y'← $\{0,1\}^{(\lambda-1)}$, $z' = (y'^{\top}||x')A$

pk=(A,$(Z_i)_i$,z'), sk=(sk_{MAC},$(Y_i)_i$,y')

USKGen(sk_{MAC}, id):

- (t,u) ←Tag(sk_{MAC},id), $v = t^{\top}\left(Y_0^{\top} + \sum id_i Y_i^{\top}\right) + y'^{\top}$

usk[id]=(t,u,v)

Enc(pk, id):

- r ← $\{0\}\times\{0,1\}^{\lambda-1}$, c_0=Ar, $c_1 = (Z_0 + \sum id_i Z_i)r$

ct=(c_0,c_1), K=z'r

Dec(usk[id], ct):
K=(v|u)c_0-$t^{\top}c_1$

Pairing is not necessary now since the computations are not in groups

Construction of IBKEM

Gen(λ):
- $A^{\top}$ ←ZeroSamp(λ), sk_{MAC}=(…
- For i=0,…,n, $Y_i \leftarrow \{0,1$…
- y'← $\{0,1\}^{(\lambda-1)}$, z'…

pk=(A,$(Z_i)_i$,z'), sk=(s…

Crucial step in the security game: to construct a reduction breaking the security of the affine MAC

USKGen(sk_{MAC}, id):
- (t,u) ←Tag(sk_{MAC},id), v …

usk[id]=(t,u,v)

Enc(pk, id):
- r ← $\{0\}\times\{0,1\}^{\lambda-1}$, c_0=Ar, c_1= $(Z_0 + \sum id_i Z_i)r$

ct=(c_0,c_1), K=z'r

Dec(usk[id], ct):

K=(v|u)c_0-$t^{\top}c_1$

Construction of IBKEM

Gen(λ):
- $A^{\top}$ ←ZeroSamp(λ), sk_{MAC}=(...
- For i=0,…,n, $Y_i \leftarrow \{0$...
- y'← $\{0,1\}^{(\lambda-1)}$, z'...

pk=(A,$(Z_i)_i$,z'), sk=(s...

Core of the proof:
A new technique to extract the forgery of the affine MAC from the adversary.
=>switching the distribution of A twice and changing the distribution of r during the switching procedure

USKGen(sk_{MAC}, id):
- (t,u) ←Tag(sk_{MAC},id), v ...

usk[id]=(t,u,v)

Enc(pk, id):
- r ← $\{0\}\times\{0,1\}^{\lambda-1}$, c_0=Ar, c_1= $(Z_0 + \sum id_i Z_i)r$

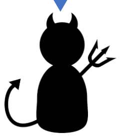
ct=(c_0,c_1), K=z'r

Dec(usk[id], ct):

K=(v|u)c_0-$t^{\top}c_1$

Proof sketch (Game 0)

$A^{\top} \leftarrow \text{ZeroSamp}(\lambda)$, $(Z_i = (Y_i^{\top} || x_i)A)_i$, $z' = (y^{\top} || x')A$



id

(t, u, v)

$(t,u) \leftarrow \text{Tag}(sk_{MAC}, id)$

$v = t^{\top}\left(Y_0^{\top} + \sum id_i Y_i^{\top}\right) + y'^{\top}$

id*

c_0, c_1, K

$r \leftarrow \{0\} \times \{0,1\}^{\lambda-1}$

$c_0 = Ar$

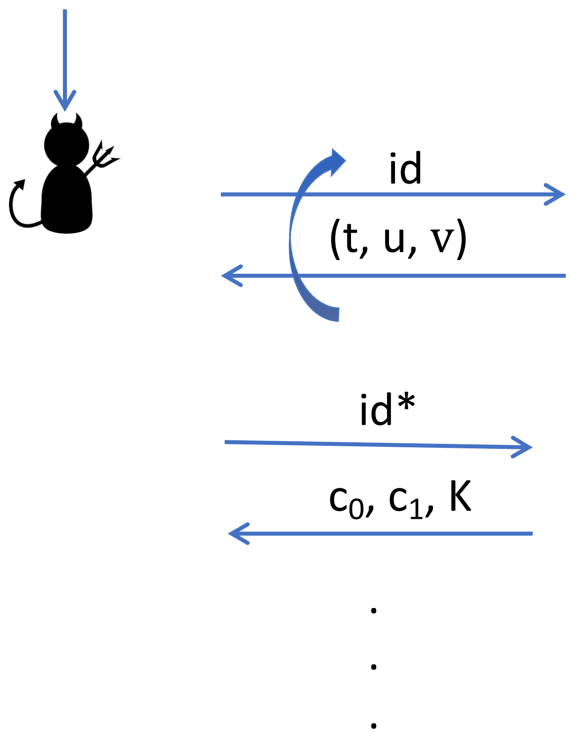
$c_1 = (Z_0 + \sum id_i Z_i)r$

$K = z'r$

.
.
.

Proof sketch (Game 1)

$A^{\top} \leftarrow \mathrm{ZeroSamp}(\lambda)$, $(Z_i = (Y_i^{\top} || x_i)A)_i$, $z' = (y^{\top} || x')A$



$(t,u) \leftarrow \mathrm{Tag}(sk_{MAC}, id)$

$v = t^{\top}\left(Y_0^{\top} + \sum id_i Y_i^{\top}\right) + y'^{\top}$

$r \leftarrow \{0\}\times\{0,1\}^{\lambda-1}$

$c_0 = (A+N)r$

$c_1 = (Y_0^{\top} | x_0)(A+N)r + \sum id_i (A+N)r$

$K = (y'^{\top} | x')(A+N)r$

Proof sketch (Game 1)

$A^\top \leftarrow$ ZeroSamp(λ), $(Z_i = (Y_i^\top || x_i)A)_i$, $z' = (y^\top || x')A$

id

(t, u, v)

(t,u) ←Tag(sk_{MAC},id)

$v = t^\top(Y_0^\top + \sum id_i Y_i^\top)$

The distribution of c_1 does not change since Nr=0

id*

c_0, c_1, K

$r \leftarrow \{0\} \times \{0,1\}^{\lambda-}$

$c_0 = (A+N)r$

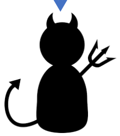
.
.
.

$c_1 = (Y_0^\top | x_0)(A+N)r + \sum id_i (A+N)r$

$K = (y'^\top | x')(A+N)r$

Proof sketch (Game 2)

$A^\top \leftarrow$ OneSamp(λ), $(Z_i = (Y_i^\top || x_i)A)_i$, $z' = (y^\top || x')A$



Switch the distribution of $A^\top$ to OneSamp

,id)

$v = t^\top \left(Y_0^\top + \sum id_i Y_i^\top\right) + y'^\top$

id*

c_0, c_1, K

$r \leftarrow \{0\} \times \{0,1\}^{\lambda-1}$

$c_0 = (A+N)r$

$c_1 = (Y_0^\top | x_0)(A+N)r + \sum id_i (A+N)r$

$K = (y'^\top | x')(A+N)r$

.
.
.

Proof sketch (Game 3)

$A^{\top} \leftarrow$ OneSamp(λ), $(Z_i = (Y_i^{\top} || x_i)A)_i$, $z' = (y^{\top} || x')A$

id

(t, u, v)

(t,u) ←Tag(sk_{MAC},id)

$v = t^{\top}\left(Y_0^{\top} + \sum id_i Y_i^{\top}\right) + y'^{\top}$

id*

c_0, c_1, K

$r \leftarrow \{1\}\times\{0,1\}^{\lambda-1}$

$c_0 = (A+N)r$

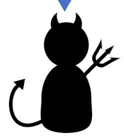
$c_1 = (Y_0^{\top}|x_0)(A+N)r + \sum id_i (A+N)r$

$K = (y'^{\top}|x')(A+N)r$

.
.
.

Proof sketch (Game 3)

$A^{\top} \leftarrow$ OneSamp(λ), $(Z_i = (Y_i^{\top}||x_i)A)_i$, $z' = (y^{\top}||x')A$



id

(t, u, v)

This does not affect the distribution of (A+N)r, since $(A+N)^{\top} \in$ ZeroSamp(λ)

id*

c_0, c_1, K

.
.
.

$r \leftarrow \{1\}\times\{0,1\}^{\lambda-1}$

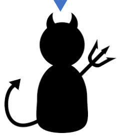
$c_0 = (A+N)r$

$c_1 = (Y_0^{\top}|x_0)(A+N)r + \sum id_i (A+N)r$

$K = (y'^{\top}|x')(A+N)r$

Proof sketch (Game 3)

$A^{\top} \leftarrow$ OneSamp(λ), $(Z_i = (Y_i^{\top}||x_i)A)_i$, $z' = (y^{\top}||x')A$



id →

← (t, u, v)

(t,u) ←Tag(sk_{MAC},id)

$v = t^{\top}(Y_0^{\top} + \sum id_i Y_i^{\top}) + y'^{\top}$

id* →

← c_0, c_1, K

.
.
.

$r \leftarrow \{1\} \times \{0,1\}^{\lambda-1}$

$c_0 = (A+N)r$

$c_1 = (Y_0^{\top}|x_0)(A+N)r + \sum id_i (A+N)r$

$K = (y'^{\top}|x')(A+N)r$

Also, notice that $(Y_0^{\top}|x_0)Nr = x_0$ and $(y'^{\top}|x')Nr = x'$ now

Proof sketch (Game 4)

$A^T \leftarrow$ ZeroSamp(λ), $(Z_i = (Y_i^T || x_i)A)_i$, $z' = (y^T || x')A$

Switch the distribution of A^T back to ZeroSamp

$v = t^T(Y_0^T + \sum id_i Y_i^T) + y'^T$

id*

c_0, c_1, K

$r \leftarrow \{1\} \times \{0,1\}^{\lambda-1}$

$c_0 = (A+N)r$

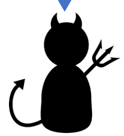
$c_1 = (Y_0^T | x_0)(A+N)r + \sum id_i (A+N)r$

$K = (y'^T | x')(A+N)r$

.
.
.

Proof sketch (Game 5)

$A^\top \leftarrow \text{ZeroSamp}(\lambda)$, $(Z_i = (0||D_i)R_0^\top)_i$, $z' = (0||d')R_0^\top$



id →

← (t, u, v)

$(t,u) \leftarrow \text{Tag}(sk_{MAC}, id)$

$v = t^\top(D_0 + \sum id_i D_i) + d' + us^\top$

id* →

← c_0, c_1, K

$r \leftarrow \{1\} \times \{0,1\}^{\lambda-1}$

$c_0 = (A+N)r$

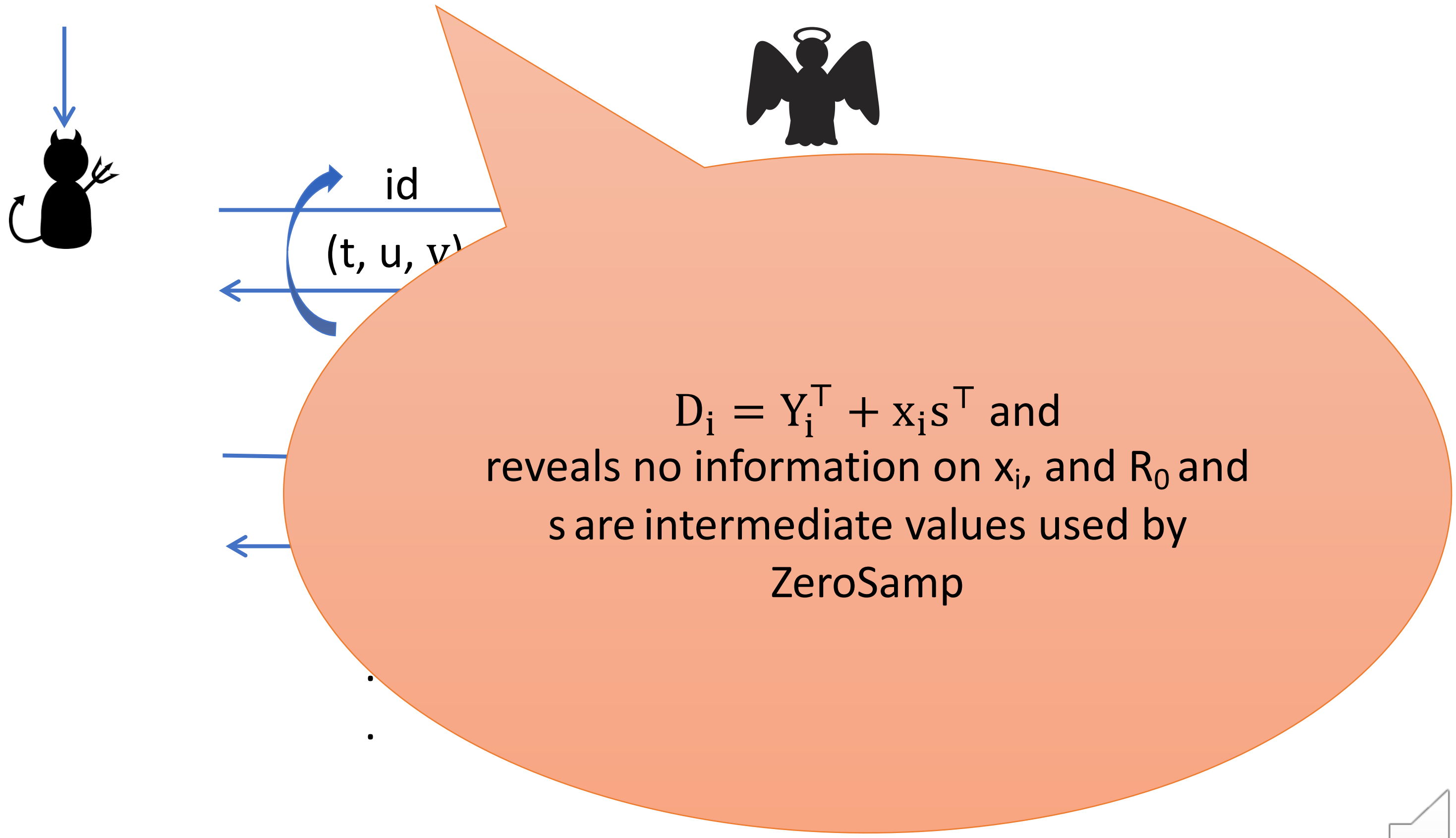
$c_1 = Z_0 r + x_0 + \sum id_i (Z_i r + x_i)$

$K = z'r + x'$

.
.
.

Proof sketch (Game 5)

$A^{\top} \leftarrow \text{ZeroSamp}(\lambda)$, $(Z_i = (0||D_i)R_0^{\top})_i$, $z' = (0||d')R_0^{\top}$



Proof sketch (Game 5)

$A^{\top} \leftarrow \mathrm{ZeroSamp}(\lambda)$, $(Z_i = (0||D_i)R_0^{\top})_i$, $z' = (0||d')R_0^{\top}$

Redefine z' in the same way

(t, u, v)

$v = t^{\top}(D_0 + \sum id_i D_i) + d' + us^{\top}$

id*

$r \leftarrow \{1\} \times \{0,1\}^{\lambda-1}$

c_0, c_1, K

$c_0 = (A+N)r$

$c_1 = Z_0 r + x_0 + \sum id_i (Z_i r + x_i)$

$K = z'r + x'$

.
.
.

Proof sketch (Game 5)

$A^{\top} \leftarrow \mathrm{ZeroSamp}(\lambda)$, $(Z_i = (0||D_i)R_0^{\top})_i$, $z' = (0||d')R_0^{\top}$

v reveals no information on the secrets except for u

id →

(t, u, v) ←

$(t,u) \leftarrow \mathrm{Tag}(sk_{MAC}, id)$

$v = t^{\top}(D_0 + \sum id_i D_i) + d' + us^{\top}$

id* →

c_0, c_1, K ←

$r \leftarrow \{1\} \times \{0,1\}^{\lambda-1}$

$c_0 = (A+N)r$

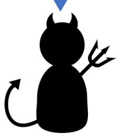
$c_1 = Z_0 r + x_0 + \sum id_i (Z_i r + x_i)$

$K = z' r + x'$

.
.
.

Proof sketch (Game 5)

$A^{\top} \leftarrow \text{ZeroSamp}(\lambda)$, $(Z_i = (0||D_i)R_0^{\top})_i$, $z' = (0||d')R_0^{\top}$



c_0, c_1, K

.
.
.

$c_0 = (A+N)r$

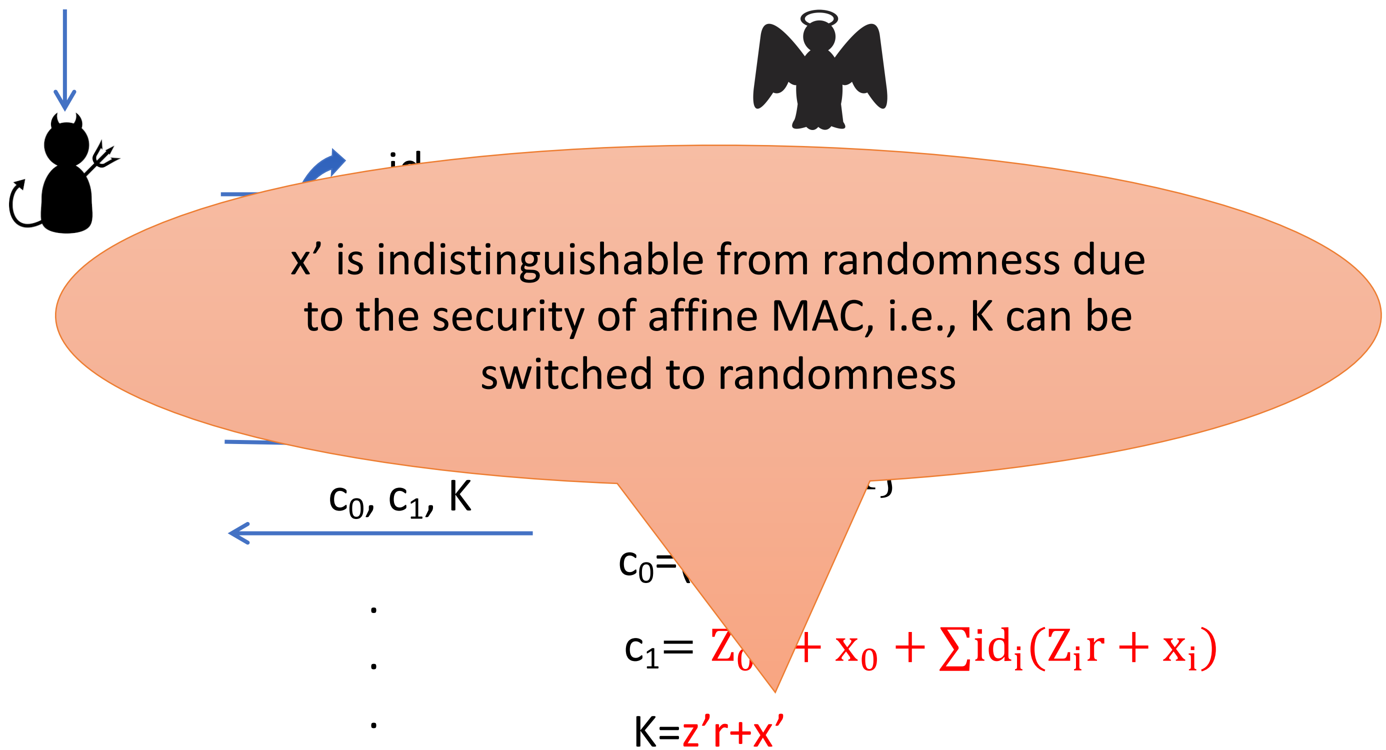
$c_1 = Z_0 r + x_0 + \sum id_i(Z_i r + x_i)$

$K = z'r + x'$

c_1 reveals no information on x_i except for $x_0 + \sum id_i x_i$ (exactly part of the token in the security game for the affine MAC)

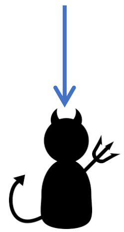
Proof sketch (Game 5)

$A^{\top} \leftarrow \text{ZeroSamp}(\lambda)$, $(Z_i = (0||D_i)R_0^{\top})_i$, $z' = (0||d')R_0^{\top}$



Proof sketch (Game 0)

$A^T \leftarrow \text{ZeroSamp}(\lambda), (Z_i = (Y_i^T || x_i)A)_i, z' = (y^T || x')A$



id

By doing the steps in the reverse order, we can prove that K is indistinguishable from randomness in the original game.

.

.

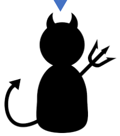
$\ldots + \sum id_i Z_i)r$

.

K=z’r

Proof sketch (Game 0)

$A^{\top} \leftarrow \text{ZeroSamp}(\lambda)$, $(Z_i = (Y_i^{\top} || x_i)A)_i$, $z' = (y^{\top} || x')A$



id

(t, u, v)

$(t,u) \leftarrow \text{Tag}(sk_{MAC}, id)$

$$v = t^{\top}\left(Y_0^{\top} + \sum id_i Y_i^{\top}\right) + y'^{\top}$$

id*

c_0, c_1, K

$r \leftarrow \{0\}\times\{0,1\}^{\lambda-1}$

In the security proof, all the computations are in NC1.

Extension to ABKEM

The red parts essentially use encoding for equality and can be generalized as predicate encodings [CGW15] to achieve ABKEM

Gen(λ):

- $A^\top \leftarrow$ZeroSamp(λ), $sk_{MAC}=(B,x_0,\ldots, x_n,x) \leftarrow Gen_{MAC}$
- For i=0,…,n, $Y_i \leftarrow \{0,1\}^{(\lambda-1)\times\lambda}$, $Zi = (Y_i^\top || x_i)A$
- $y' \leftarrow \{0,1\}^{(\lambda-1)}$, $z' = (y^\top || x')A$

$pk=(A,(Z_i)_i,z')$, $sk=(sk_{MAC},(Y_i)_i,y')$

Tag(sk_{MAC}, $id=(id_i)_{i=1,..,n}$):

$t \leftarrow$SampYes(B)

$u = x_0^\top t + \sum id_i x_i^\top t + x'$

Return (t,u)

USKGen(sk_{MAC}, id):

- $(t,u) \leftarrow$Tag(sk_{MAC},id), $v = t^\top (Y_0^\top + \sum id_i Y_i^\top) + y'^\top$

usk[id]=(t,u,v)

Enc(pk, id):

- $r \leftarrow \{0\}\times\{0,1\}^{\lambda-1}$, $c_0=Ar$, $c_1 = (Z_0 + \sum id_i Z_i)r$

$ct=(c_0,c_1)$, $K=z'r$

Dec(usk[id], ct):

$K=(v|u)c_0-t^\top c_1$

Conclusion

- Generic construction of fine-grained ABE secure against NC^1 adversaries under $\mathrm{NC}^1 \neq \oplus \mathrm{L/poly}$ and computable in AC0[2].

Conclusion

- Generic construction of fine-grained ABE secure against NC^1 adversaries under $\mathrm{NC}^1 \neq \oplus \mathrm{L/poly}$ and computable in AC0[2].

  Instantiations:

  1. IBE scheme (which in turn implies a signature scheme)
  2. ABEs for
     - inner-product encryption
     - non-zero inner-product
     - encryptionspatial encryption
     - doubly spatial encryption
     - boolean span programs
     - arithmetic span programs
  3. Broadcast encryption
  4. fuzzy IBE

Conclusion

- Generic construction of fine-grained ABE secure against NC^1 adversaries under $\mathrm{NC}^1 \neq \oplus \mathrm{L/poly}$ and computable in AC0[2].

  Instantiations:

  1. IBE scheme (which in turn implies a signature scheme)
  2. ABEs for
     - inner-product encryption
     - non-zero inner-product
     - encryptionspatial encryption
     - doubly spatial encryption
     - boolean span programs
     - arithmetic span programs
  3. Broadcast encryption
  4. fuzzy IBE

- More application of our techniques : an efficient fine-grained QA-NIZK.