RSA®CONFERENCE2013

A ROBUST AND PLAINTEXT-AWARE VARIANT OF SIGNED ELGAMAL ENCRYPTION

Joana Treger

ANSSI, France.

Session ID: CRYP-W21

Session Classification: Advanced

RSA CONFERENCE 2013

ELGAMAL ENCRYPTION & BASIC CONCEPTS

CDH / DDH

- Computational Diffie-Hellman Assumption :

G generator of finite cyclic group $\mathbb{G}$. For all efficient algorithms $\mathcal{A}$:

$$\Pr\left(\mathcal{A}(G^a, G^b) = G^{ab}\right) \text{ is at most negligible.}$$

- Decisional Diffie-Hellman Assumption :

G generator of finite cyclic group $\mathbb{G}$. For all efficient algorithms $\mathcal{A}$:

$$|\Pr\left(\mathcal{A}(G^a, G^b, C) = 1\right) - \Pr\left(\mathcal{A}(G^a, G^b, G^{ab}) = 1\right)|$$

is at most negligible.

ELGAMAL PUBLIC KEY ENCRYPTION SCHEME [ElGamal84]

- G is the generator of some finite cyclic group $\mathbb{G}$ of prime order p.

$\underline{\mathtt{Encryption}}$

$$\left[\ \mathtt{pk} : X = G^x\ \right]$$

$$\begin{cases} r \leftarrow_\$ \mathbb{Z}_p^* \\ R \leftarrow G^r \\ Y \leftarrow M X^r \end{cases}$$

$$\boxed{\mathtt{Output} : \psi = (Y, R)}$$

$\underline{\mathtt{Decryption}}$

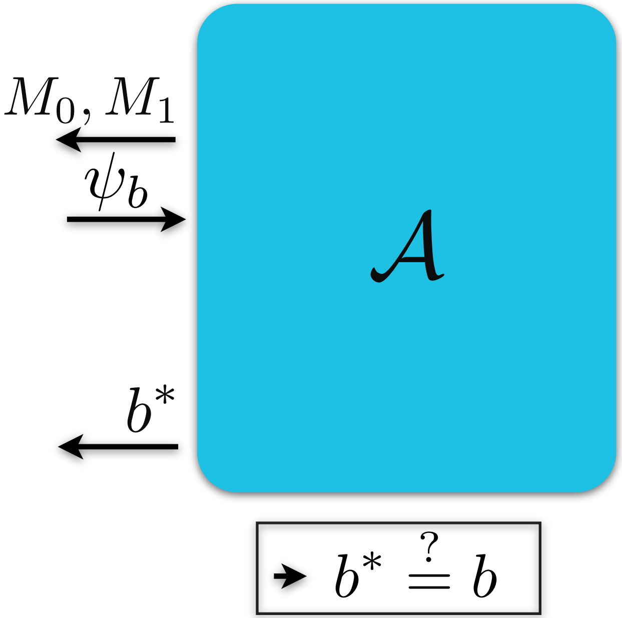
$$\left[\ \mathtt{pk} : X = G^x,\ \ \mathtt{sk} : x\ \right]$$

$$\begin{cases} R' \leftarrow R^x \end{cases}$$

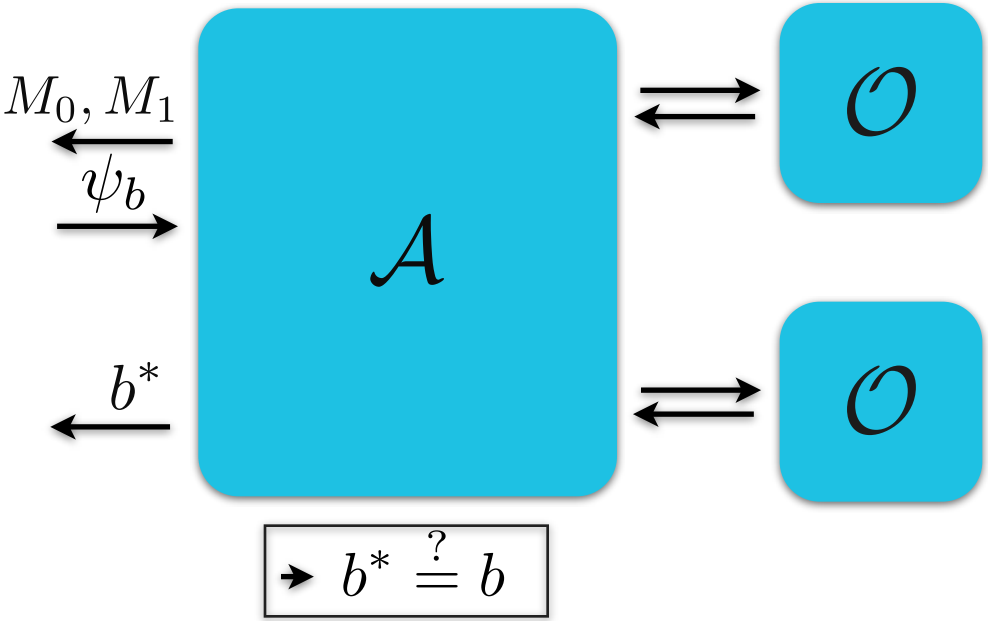
$$\boxed{\mathtt{Output} : M = Y/R'}$$

IND-CPA / IND-CCA2

- $\underline{\texttt{IND} - \texttt{CPA Game}}$



- $\underline{\texttt{IND} - \texttt{CCA2 Game}}$



IND-CCA2 security: **strongest** security notion.

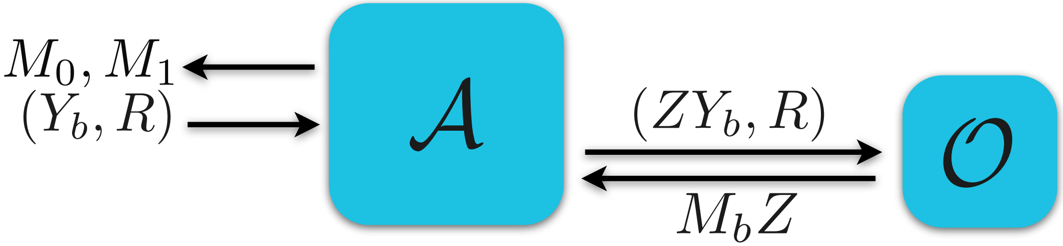
SECURITY OF ELGAMAL

[TsiounisYung98]

ElGamal is **IND-CPA** under **DDH**.

ElGamal is ***not* IND-CCA2.**

- **It is malleable**: it's easy to transform a ciphertext into another one that decrypts to a related plaintext.
- An **IND-CCA2 attacker** can ask for a decryption to win the game.



HOW TO TWEAK ELGAMAL ENCRYPTION TO REACH IND-CCA2 SECURITY?

TWO OPTIONS

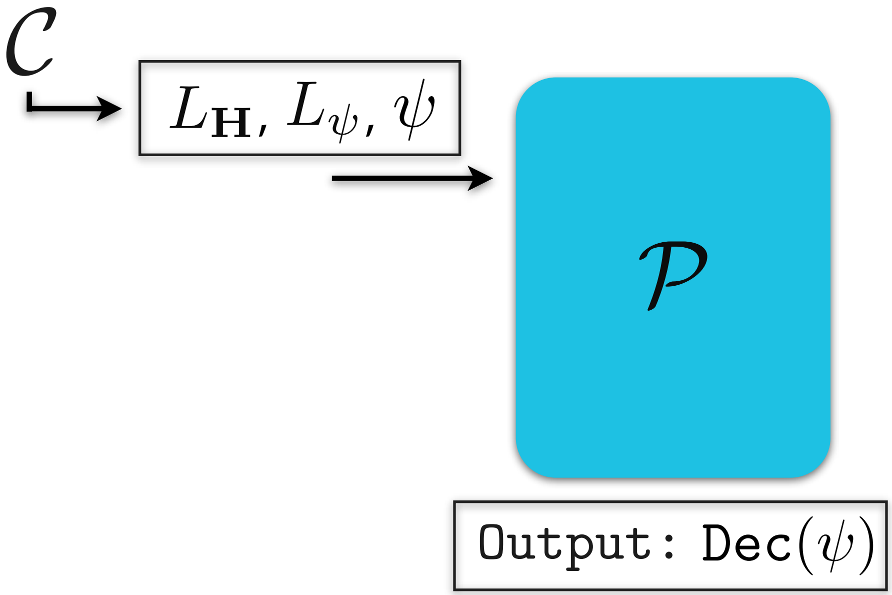
- 1st option: Include hashing [AbdallaBellareRogaway01]
  - *Example*: DHIES - R is **hashed** through some random oracle to create a **symmetric key** used to encrypt the message. $k = H(R, R')$.
  - The resulting scheme is no longer malleable.
  - It requires a **symmetric cipher**.

- 2nd option: Add a non interactive proof of knowledge.
  - *Example*: Schnorr Signed ElGamal (SS-EG) - Add a Schnorr signature as a PoK of r. [Jakobsson98] [TsiounisYung98]

PLAINTEXT-AWARENESS

[BellareRogaway94]
[BelDesaiPointchRog98]

- ROM-PA for a public-key encryption scheme:



$\mathcal{C}$
$L_{\mathbf{H}}, L_{\psi}, \psi$
$\mathcal{P}$
$\texttt{Output: Dec}(\psi)$

$\mathcal{C}$: efficient ciphertext creator.
$\mathcal{P}$: efficient plaintext extractor.

PLAINTEXT-AWARENESS & IND-CCA2 SECURITY

[BelDesaiPointchRog98]

- Intuitively

The **only way** to produce a **valid ciphertext** is to apply the **encryption algorithm** to a public key and a message, rendering a **decryption oracle** available **to an IND-CCA2 adversary useless**.

- PA & IND-CCA2?

$$\begin{gathered}\text{IND-CPA} + \text{PA} \\ \Downarrow \\ \text{IND-CCA2}\end{gathered}$$

LIMITATIONS OF SS-EG

- So...
  - **Adding a Schnorr signature** to ElGamal encryption **would render** the scheme **IND-CCA2**.
- However...
  - ***No proof*** that SS-EG reaches **IND-CCA2** security in the ROM;
  - SS-EG is ***not* plaintext-aware**. [SeurinT13]

SCHNORR SIGNED ELGAMAL ENCRYPTION

Encryption

$[\ \mathtt{pk} : X = G^x\ ]$

$$\begin{cases} r, a \leftarrow_\$ \mathbb{Z}_p^* \\ R \leftarrow G^r,\ A \leftarrow G^a \\ Y \leftarrow M X^r \\ c \leftarrow \mathbf{H}(Y, R, A) \\ s = a + cr \end{cases}$$

$\boxed{\mathtt{Output} : \psi = (Y, R, s, c)}$

Decryption

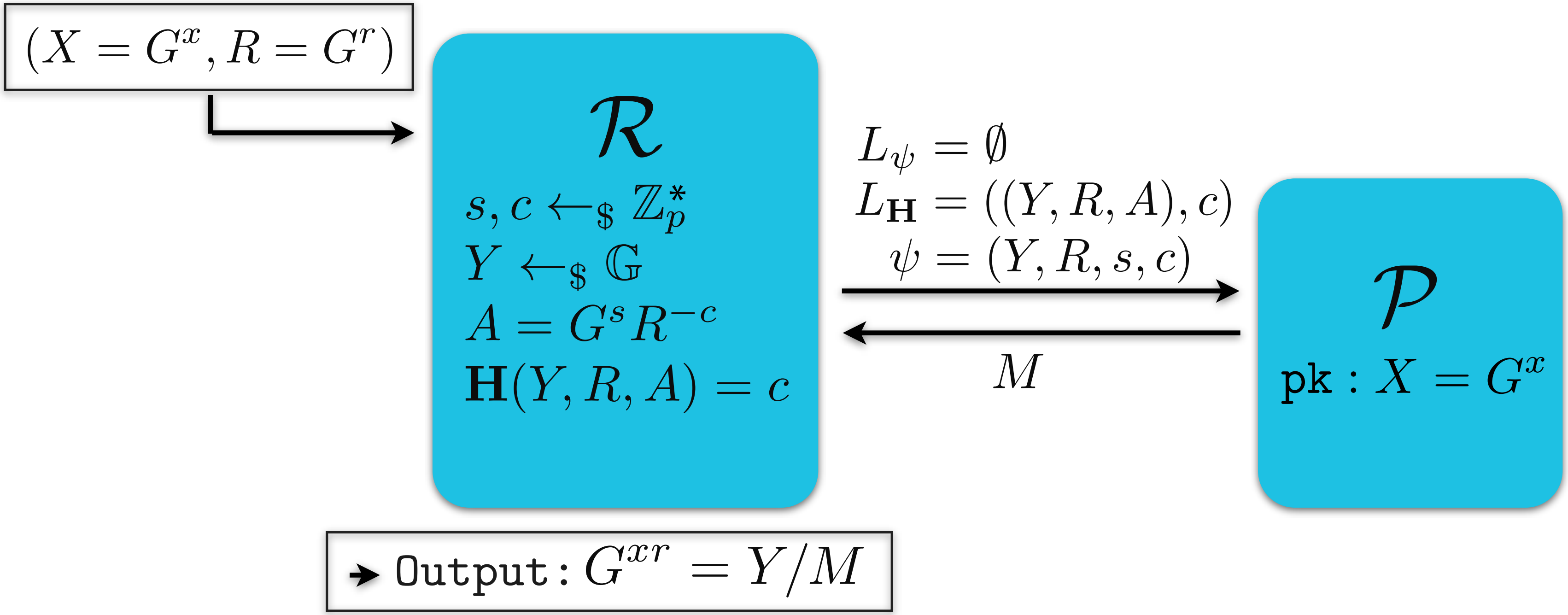
$[\ \mathtt{pk} : X = G^x \quad \mathtt{sk} : x\ ]$

$$\begin{cases} R' \leftarrow R^x \\ A \leftarrow G^s R^{-c} \\ c \overset{?}{=} \mathbf{H}(Y, R, A) \end{cases}$$

$\boxed{\mathtt{Output} : M = Y/R'}$

SS-EG IS NOT PLAINTEXT AWARE [SeurinT13]

- Sketch of the proof:
  - **Assume it is ROM-PA** secure and build a **reduction that solves the CDH problem**.



$(X = G^x, R = G^r)$

$\mathcal{R}$

$s, c \leftarrow_\$ \mathbb{Z}_p^*$

$Y \leftarrow_\$ \mathbb{G}$

$A = G^s R^{-c}$

$\mathbf{H}(Y, R, A) = c$

$L_\psi = \emptyset$

$L_{\mathbf{H}} = ((Y, R, A), c)$

$\psi = (Y, R, s, c)$

M

$\mathcal{P}$

$\mathrm{pk} : X = G^x$

$\texttt{Output} : G^{xr} = Y/M$

CPS-EG: A NEW VARIANT OF SIGNED ELGAMAL ENCRYPTION

Chaum Pedersen Signed ElGamal

[SeurinT13]

Add a CP signature as a PoK of a DL equality : $\log_G(R)=\log_X(R')$.

- CPS-EG - Definition

Encryption

$[\ \texttt{pk} : X = G^x\ ]$

$$\begin{cases} r, a \leftarrow_{\$} \mathbb{Z}_p^* \\ R \leftarrow G^r, \ A \leftarrow G^a, \ A' \leftarrow X^a \\ Y \leftarrow M X^r \\ c \leftarrow \mathbf{H}(Y, R, R' = X^r, A, A') \\ s = a + cr \end{cases}$$

$\boxed{\texttt{Output} : \psi = (Y, R, A, s)}$

Decryption

$[\ \texttt{pk} : X = G^x \quad \texttt{sk} : x\ ]$

$$\begin{cases} R' \leftarrow R^x, \ A' \leftarrow A^x \\ c \leftarrow \mathbf{H}(Y, R, R', A, A') \\ G^s \overset{?}{=} A R^c \\ X^s \overset{?}{=} A' R'^c \end{cases}$$

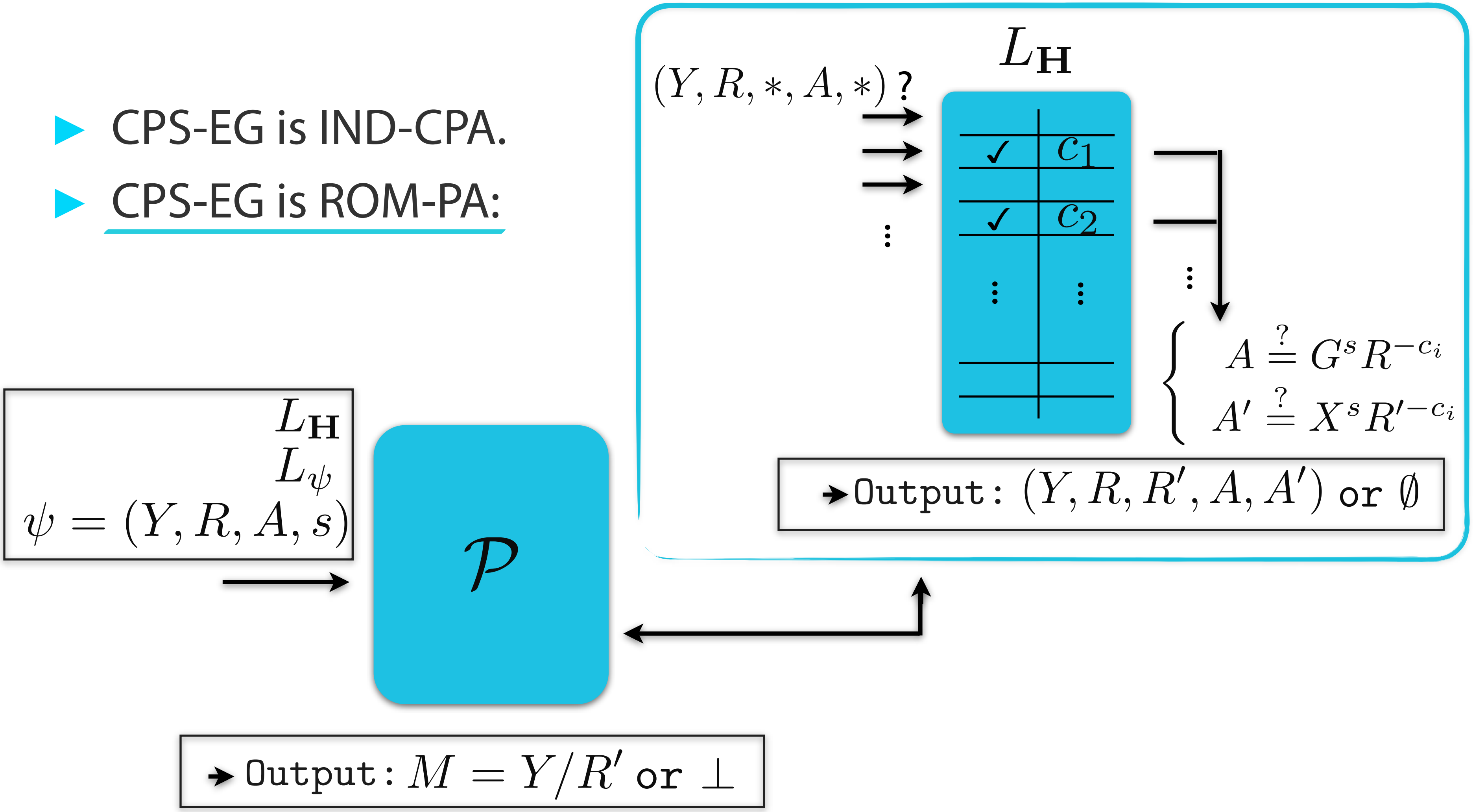
$\boxed{\texttt{Output} : M = Y/R'}$

COMPARISON - EFFICIENCY

Scheme	pub., sec. key size	exponen./ encryption	exponen./ decryption	ciphertext size
ElGamal	G, p	2	1	2G
SS-EG	G, p	3	2	2G+2p
TDH-1 [ShoupGen02]	G, p	3 (+ 2 online)	3	3G+2p
TDH-2	G, 2p	5	3	3G+2p
CPS-EG	G, p	4	4	3G+p

CPS-EG REACHES IND-CCA2 SECURITY [SeurinT13]

- CPS-EG is IND-CPA.
- CPS-EG is ROM-PA:



$L_{\mathbf{H}}$

$(Y, R, *, A, *)$?

✓	c_1
✓	c_2
$\vdots$	$\vdots$

$$\begin{cases} A \overset{?}{=} G^s R^{-c_i} \\ A' \overset{?}{=} X^s R'^{-c_i} \end{cases}$$

→ Output: (Y, R, R', A, A') or $\emptyset$

$L_{\mathbf{H}}$

L_ψ

$\psi = (Y, R, A, s)$

$\mathcal{P}$

→ Output: $M = Y/R'$ or $\perp$

CAUTION

- **Caution** when relying on Chaum & Pedersen's signature:
    - If one uses the **pair (s,c)** in the ciphertext **instead of (A,s)** the scheme does ***not* remain IND-CPA**. [Poettering]
        - $c = \mathbf{H}(Y, R, R', A, A'),\ \ R' = Y/M_b.$

          Part of the ciphertext

          M_0, M_1 known to the attacker.
        - The attacker **tries both values** for R'.
        - A' is deduced from R', s, c and X: $A' = X^s R'^{-c}$.
        - It simply needs to **test the two hash queries** corresponding to the two possible values for R'.
    - This defect is avoided when using the pair (A,s). [SeurinT13-Full]

CPS-EG IS ANONYMOUS & STRONGLY ROBUST

- Anonymity [BellareBoldyrevaDesaiPointcheval01]
  - A **ciphertext** does **not reveal** the **public key** under which it was created.

CPS-EG is ANON-CCA2 under the DDH.

- Strong Robustness [AbdallaBellareNeven10]
  - Hard to create a **ciphertext** that decrypts to a valid plaintext under **2 different secret keys**.

CPS-EG is SROB-CCA when **H** is a collision-resistant hash function.

COMPARISON - SECURITY

Scheme	IND-CCA2 under	ANON+SROB
ElGamal	(CPA under DDH)	✗
SS-EG	~ GGM	✗
TDH-1	DDH	✗
TDH-2	DDH	✗
CPS-EG	DDH	✓

TO SUM UP

- New variant of signed ElGamal encryption : CPS-EG
  - IND-CCA2-secure (plaintext-aware) under DDH assumption;
  - Reaches CCA2-anonymity and strong robustness.
- There exists a hybrid version of CPS-EG.
- Full (and *revised*) version of this paper:
  - Eprint #2012/649. [SeurinT13-Full]

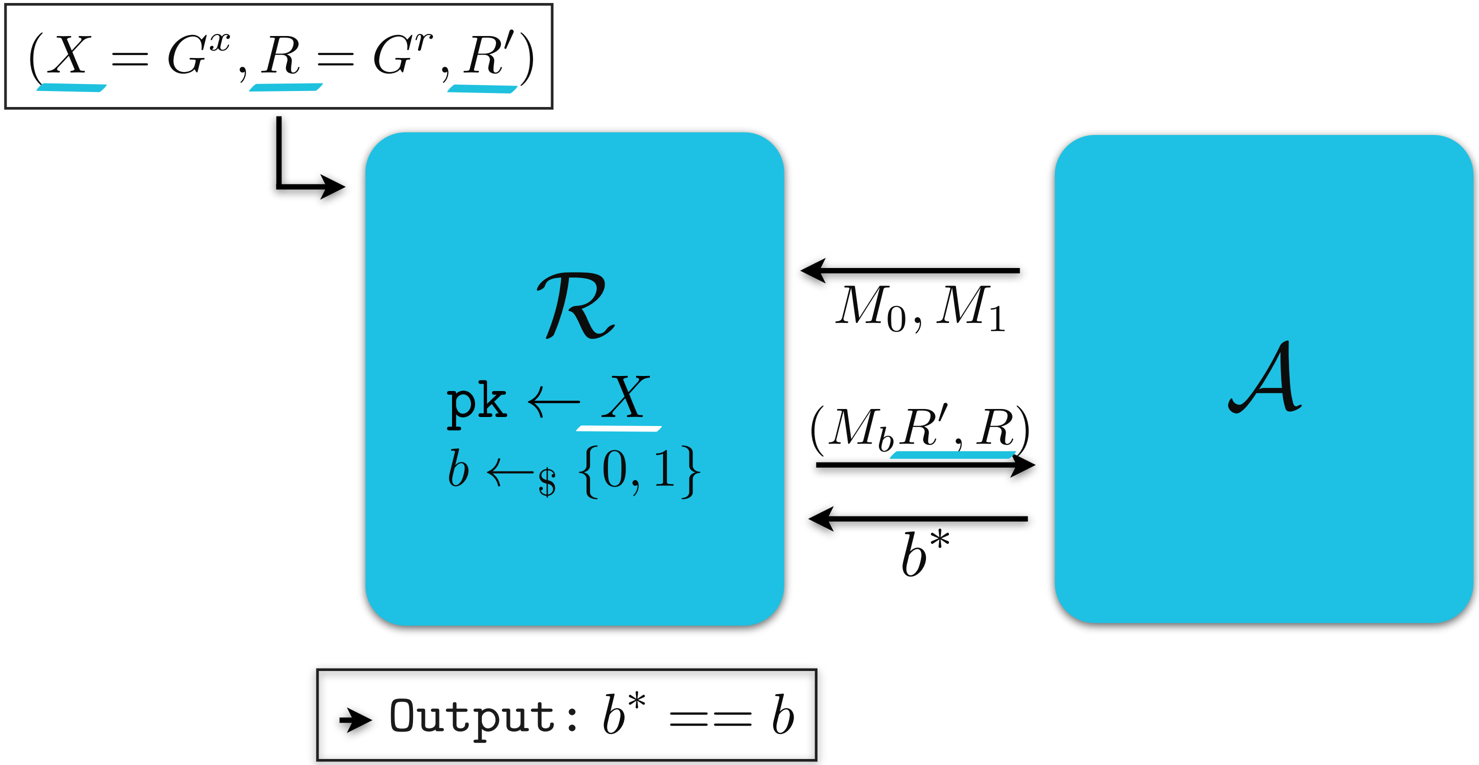
THANKS

ADDITIONAL SLIDES

ELGAMAL

ELGAMAL: IND-CPA UNDER DDH

- Sketch of the proof: [TsiounisYung98]
  - **Reduction** solving the **DDH** prob. from an **IND-CPA attacker**.



$(\underline{X} = G^x, \underline{R} = G^r, \underline{R'})$

$\mathcal{R}$

$\mathrm{pk} \leftarrow \underline{X}$

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

M_0, M_1

$(M_b \underline{R'}, R)$

b^*

$\mathcal{A}$

$\texttt{Output:}\ b^* == b$

DHIES

1ST OPTION: HASHED VARIANT

- Example: DHIES [AbdallaBellareRogaway01]

Encryption

$$\left[\ \mathtt{pk} : X = G^x\ \right]$$

$$\begin{cases} r \leftarrow_\$ \mathbb{Z}_p \\ R \leftarrow G^r,\ R' = X^r \\ k = H(R, R') \end{cases}$$

$$\boxed{\mathtt{Output} : \psi = (R, c = \mathrm{E}_k(M))}$$

Decryption

$$\left[\ \mathtt{pk} : X = G^x \quad \mathtt{sk} : x\ \right]$$

$$\begin{cases} R' \leftarrow R^x \\ k = H(R, R') \end{cases}$$

$$\boxed{\mathtt{Output} : \ M = \mathrm{D}_k(c)}$$

- The resulting scheme is no longer malleable.
- It requires a **symmetric cipher**.

SECURITY OF DHIES

- Strong Diffie-Hellman Assumption

It is hard to compute DH(Y,X), even when additional access to a decision oracle is given, which on input (Z,T) returns 1 if DH(X,Z)=T and 0 otherwise.

LEMMA

CPS-EG - USEFUL LEMMA

Fix $X = G^x$, $x \in \mathbb{F}_p$. Let $R = G^r, R', A = G^a, A' \in \mathbb{G}$ be four group elements such that $r, a \neq 0 \mod p$, and $R' \neq R^x$ or $A' \neq A^x$.
Then there is at most one integer $c \in \mathbb{Z}_p$ such that there exists $s \in \mathbb{Z}_p$ satisfying both $G^s = AR^c$ and $X^s = A'R'^c$.

NO PROOF OF IND-CCA2 SECURITY FOR SS-EG

SCHNORR SIGNATURES & FORKING LEMMA

- Schnorr signatures are not *online-extractable*
  - Suppose **there exists a forger** and build a **reduction** that **solves the DL problem**.
  - The reduction **has to rewind the forger** to obtain two signatures of the same message under the same public key.

$$R = G^s Y^{-c} = G^{s'} Y^{-c'}$$
$$\mathrm{dlog}(Y) = \frac{s - s'}{c - c'}$$

NO PROOF THAT SS-EG IS IND-CCA2

- IND-CCA2 security & SS-EG
  - Schnorr signatures are **not online-extractable**.
  - Setting: We consider a(ny) reduction that would use an IND-CCA2 adversary.
  - The **reduction has to answer** hash queries and decryption queries.
  - **Possible** for the adversary to construct a **list of specific ciphertexts and hash queries**, where one ciphert./h. query is obtained **from the previous ones**.
  - If the attacker then sends its list of queries to the reduction, in the exact **reverse order**, then the reduction has to **rewind** the forger **exponentially**.

ROBUSTNESS AND ANONYMITY

WEAK ROBUSTNESS AND STRONG ROBUSTNESS

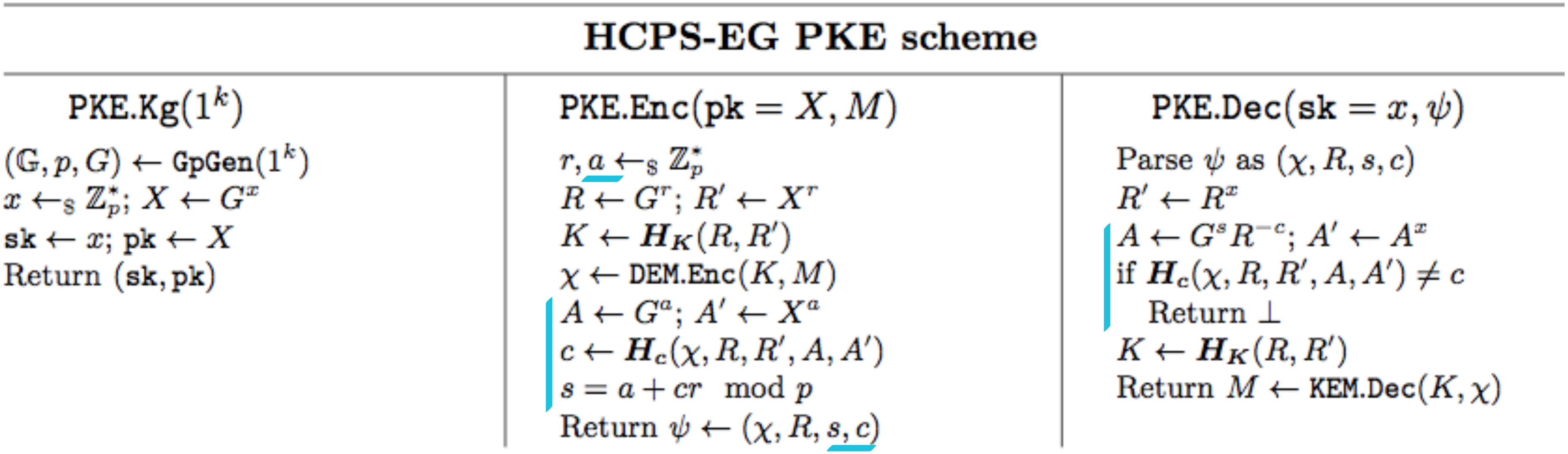
- Strong and Weak Robustness [AbdallaBellareNeven10]
  - *Definition* [Weak robustness]: Hard to create a **ciphertext** that decrypts under **2 different secret keys**.
  - *Definition* [Strong Robustness]: Hard to produce a **plaintext** that, **once encrypted** under some public key, decrypts to a **valid plaintext under another secret key**.
- Robustness can always be achieved
  - How? Append public key of the receiver to the ciphertext.

ROBUSTNESS COMBINED WITH ANONYMITY

- Robustness:
  - *Definition* [Robustness]: Hard to create a **ciphertext** that decrypts under **2 different secret keys**.
- Anonymity:
  - *Definition* [Anonymity]: A **ciphertext** does **not reveal** the **public key** under which it was created.
- Anonymity + Robustness:
  - Help make encryption **more resistant to misuse**.
  - *Ex*. **Anonymously** send a ciphertext to a **particular target of a larger group**. **Who**'s the target? Solution: **Robustness**.
  - **First solution** for robustness **not ok**.

HYBRID VARIANT HCPS-EG

HYBRID VERSION



HCPS-EG PKE scheme

```
$\mathtt{PKE.Kg}(1^k)$
$(\mathbb{G}, p, G) \leftarrow \mathtt{GpGen}(1^k)$
$x \leftarrow_\$ \mathbb{Z}_p^*;\ X \leftarrow G^x$
$\mathbf{sk} \leftarrow x;\ \mathbf{pk} \leftarrow X$
Return $(\mathbf{sk}, \mathbf{pk})$
```

```
$\mathtt{PKE.Enc}(\mathbf{pk} = X, M)$
$r, \underline{a} \leftarrow_\$ \mathbb{Z}_p^*$
$R \leftarrow G^r;\ R' \leftarrow X^r$
$K \leftarrow \boldsymbol{H_K}(R, R')$
$\chi \leftarrow \mathtt{DEM.Enc}(K, M)$
$A \leftarrow G^a;\ A' \leftarrow X^a$
$c \leftarrow \boldsymbol{H_c}(\chi, R, R', A, A')$
$s = a + cr \mod p$
Return $\psi \leftarrow (\chi, R, \underline{s, c})$
```

```
$\mathtt{PKE.Dec}(\mathbf{sk} = x, \psi)$
Parse $\psi$ as $(\chi, R, s, c)$
$R' \leftarrow R^x$
$A \leftarrow G^s R^{-c};\ A' \leftarrow A^x$
if $\boldsymbol{H_c}(\chi, R, R', A, A') \neq c$
    Return $\perp$
$K \leftarrow \boldsymbol{H_K}(R, R')$
Return $M \leftarrow \mathtt{KEM.Dec}(K, \chi)$
```

PROPERTIES OF HCPS-EG

- IND-CCA2 secure in the ROM under CDH.
  - Using the weakest form of security for the data encapsulation mechanism (DEM): ciphertext indistinguishability under one-time attacks (IND-OT).
  - *Example*: With AES in counter mode.
- Anonymous against CCA2-attacks under CDH.
- Strongly robust.

Efficient Public Key Cryptosystem Resilient to Key Leakage Chosen Ciphertext Attacks

Shengli Liu[1], Jian Weng[2] and Yunlei Zhao[3]

[1]Shanghai Jiao Tong University (slliu@sjtu.edu.cn)
[2]Jinan University(cryptjweng@gmail.com)
[3]Fudan University(ylzhao@fudan.edu.cn)

2013-02-27

Introduction and Motivation

Introduction to leakage-resilient PKE

- IND-CCA2 security assumes that the secret key sk is totally randomly distributed, i.e., $Entropy(sk) = |sk|$.

Introduction to leakage-resilient PKE

- IND-CCA2 security assumes that the secret key sk is totally randomly distributed, i.e., $Entropy(sk) = |sk|$.
- Side-channel attack, i.e., "memory attacks", may leak information about the secret key sk, i.e., $Entropy(sk) \leq |sk|$.

Introduction to leakage-resilient PKE

- IND-CCA2 security assumes that the secret key sk is totally randomly distributed, i.e., $Entropy(sk) = |sk|$.
- Side-channel attack, i.e., "memory attacks", may leak information about the secret key sk, i.e., $Entropy(sk) \leq |sk|$.
- Leakage-resilient public key encryption (PKE) schemes are designed to resist key leakage.

Introduction to leakage-resilient PKE

- IND-CCA2 security assumes that the secret key sk is totally randomly distributed, i.e., $Entropy(sk) = |sk|$.
- Side-channel attack, i.e., "memory attacks", may leak information about the secret key sk, i.e., $Entropy(sk) \leq |sk|$.
- Leakage-resilient public key encryption (PKE) schemes are designed to resist key leakage.
- We will consider IND-CCA2 security in **bounded key-leakage model**, where the total amount of leaked information about the key is bounded by some parameter λ (bits).

Related Works

- In 2005, Akavia et.al.[AGV09] showed that the lattice-based PKE scheme proposed by Regev [R05] is resilient to any leakage of $L/\text{polylog}(L)$ bits, with L the bit length of the secret key.

Related Works

- In 2005, Akavia et.al.[AGV09] showed that the lattice-based PKE scheme proposed by Regev [R05] is resilient to any leakage of $L/\mathrm{polylog}(L)$ bits, with L the bit length of the secret key.
- In 2009, Noar and Segev [NS09] suggested that any PKE based on *hash proof systems* [CS02] can be made secure against chosen-plaintext key leakage attacks (IND-KL-CPA) with help of randomness extractors.

Related Works

- In 2005, Akavia et.al.[AGV09] showed that the lattice-based PKE scheme proposed by Regev [R05] is resilient to any leakage of $L/\mathrm{polylog}(L)$ bits, with L the bit length of the secret key.
- In 2009, Noar and Segev [NS09] suggested that any PKE based on *hash proof systems* [CS02] can be made secure against chosen-plaintext key leakage attacks (IND-KL-CPA) with help of randomness extractors.
- For chosen-ciphertext key leakage attacks (IND-KL-CCA2), Naorand Segev [NS09] proved that Noar-Yung 's "double encryption" paradigm works as well.

Related Works

- In 2005, Akavia et.al.[AGV09] showed that the lattice-based PKE scheme proposed by Regev [R05] is resilient to any leakage of $L/\text{polylog}(L)$ bits, with L the bit length of the secret key.
- In 2009, Noar and Segev [NS09] suggested that any PKE based on *hash proof systems* [CS02] can be made secure against chosen-plaintext key leakage attacks (IND-KL-CPA) with help of randomness extractors.
- For chosen-ciphertext key leakage attacks (IND-KL-CCA2), Naorand Segev [NS09] proved that Noar-Yung 's “double encryption” paradigm works as well.

$$\boxed{\text{IND-KL-CPA secure PKE1}} + \boxed{\text{IND-KL-CPA secure PKE2}} + \boxed{\text{NIZK}}$$

Related Works

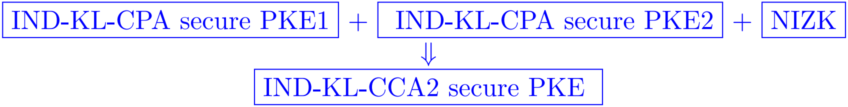
- In 2005, Akavia et.al.[AGV09] showed that the lattice-based PKE scheme proposed by Regev [R05] is resilient to any leakage of $L/\text{polylog}(L)$ bits, with L the bit length of the secret key.
- In 2009, Noar and Segev [NS09] suggested that any PKE based on *hash proof systems* [CS02] can be made secure against chosen-plaintext key leakage attacks (IND-KL-CPA) with help of randomness extractors.
- For chosen-ciphertext key leakage attacks (IND-KL-CCA2), Naorand Segev [NS09] proved that Noar-Yung 's "double encryption" paradigm works as well.

$$\boxed{\text{IND-KL-CPA secure PKE1}} + \boxed{\text{ IND-KL-CPA secure PKE2}} + \boxed{\text{NIZK}}$$

$$\Downarrow$$

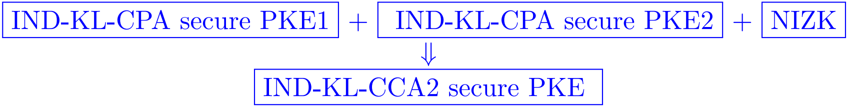
Related Works

- In 2005, Akavia et.al.[AGV09] showed that the lattice-based PKE scheme proposed by Regev [R05] is resilient to any leakage of $L/\mathrm{polylog}(L)$ bits, with L the bit length of the secret key.
- In 2009, Noar and Segev [NS09] suggested that any PKE based on *hash proof systems* [CS02] can be made secure against chosen-plaintext key leakage attacks (IND-KL-CPA) with help of randomness extractors.
- For chosen-ciphertext key leakage attacks (IND-KL-CCA2), Naorand Segev [NS09] proved that Noar-Yung 's "double encryption" paradigm works as well.



Related Works

- In 2005, Akavia et.al.[AGV09] showed that the lattice-based PKE scheme proposed by Regev [R05] is resilient to any leakage of $L/\mathrm{polylog}(L)$ bits, with L the bit length of the secret key.
- In 2009, Noar and Segev [NS09] suggested that any PKE based on *hash proof systems* [CS02] can be made secure against chosen-plaintext key leakage attacks (IND-KL-CPA) with help of randomness extractors.
- For chosen-ciphertext key leakage attacks (IND-KL-CCA2), Naorand Segev [NS09] proved that Noar-Yung 's "double encryption" paradigm works as well.



IND-KL-CPA secure PKE1 + IND-KL-CPA secure PKE2 + NIZK

⇓

IND-KL-CCA2 secure PKE

The key leakage λ is also up to $L(1 - o(1))$.

- In 2010, Dodis et.al. [DHLW10] exploited more efficient ways to achieve IND-KL-CCA2 security.

- In 2010, Dodis et.al. [DHLW10] exploited more efficient ways to achieve IND-KL-CCA2 security.
- They proposed a new concept of *true-simulation extractable* NIZK arguments, and gave a construction of IND-KL-CCA2-secure PKE from an IND-KL-CPA secure one together with a strong f-*true-simulation extractable* NIZK argument. The key leakage λ is also up to $L(1-o(1))$.

- In 2010, Dodis et.al. [DHLW10] exploited more efficient ways to achieve IND-KL-CCA2 security.
- They proposed a new concept of *true-simulation extractable* NIZK arguments, and gave a construction of IND-KL-CCA2-secure PKE from an IND-KL-CPA secure one together with a strong f-*true-simulation extractable* NIZK argument. The key leakage λ is also up to $L(1 - o(1))$.

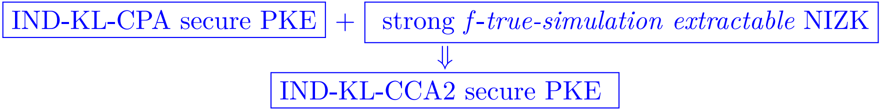
IND-KL-CPA secure PKE + strong f-*true-simulation extractable* NIZK

- In 2010, Dodis et.al. [DHLW10] exploited more efficient ways to achieve IND-KL-CCA2 security.
- They proposed a new concept of *true-simulation extractable* NIZK arguments, and gave a construction of IND-KL-CCA2-secure PKE from an IND-KL-CPA secure one together with a strong f-*true-simulation extractable* NIZK argument. The key leakage λ is also up to $L(1-o(1))$.

$$\boxed{\text{IND-KL-CPA secure PKE}} + \boxed{\text{strong } f\text{-}\textit{true-simulation extractable} \text{ NIZK}}$$

$$\Downarrow$$

- In 2010, Dodis et.al. [DHLW10] exploited more efficient ways to achieve IND-KL-CCA2 security.
- They proposed a new concept of *true-simulation extractable* NIZK arguments, and gave a construction of IND-KL-CCA2-secure PKE from an IND-KL-CPA secure one together with a strong f-*true-simulation extractable* NIZK argument. The key leakage λ is also up to $L(1 - o(1))$.

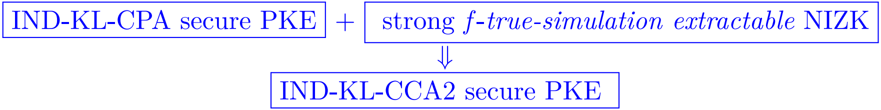


IND-KL-CPA secure PKE + strong f-*true-simulation extractable* NIZK

$\Downarrow$

IND-KL-CCA2 secure PKE

- In 2010, Dodis et.al. [DHLW10] exploited more efficient ways to achieve IND-KL-CCA2 security.
- They proposed a new concept of *true-simulation extractable* NIZK arguments, and gave a construction of IND-KL-CCA2-secure PKE from an IND-KL-CPA secure one together with a strong f-*true-simulation extractable* NIZK argument. The key leakage λ is also up to $L(1 - o(1))$.



IND-KL-CPA secure PKE + strong f-*true-simulation extractable* NIZK

$\Downarrow$

IND-KL-CCA2 secure PKE

The key leakage λ is also up to $L(1 - o(1))$.

Practical PKE with IND-KL-CCA2 Security

- The variant of Cramer-Shoup cryptosystem (“KL-CS-PKE”) presented by Naor-Segev [NS,CRYPTO09] is the most practical one.

Practical PKE with IND-KL-CCA2 Security

- The variant of Cramer-Shoup cryptosystem ("KL-CS-PKE") presented by Naor-Segev [NS,CRYPTO09] is the most practical one.

$$\boxed{\underbrace{x_1, x_2}_{c}, \underbrace{y_1, y_2}_{d}} \qquad \boxed{\underbrace{z_1, z_2}_{h}}$$

Practical PKE with IND-KL-CCA2 Security

- The variant of Cramer-Shoup cryptosystem (“KL-CS-PKE”) presented by Naor-Segev [NS,CRYPTO09] is the most practical one.

$$\boxed{\underbrace{x_1, x_2}_{c}, \underbrace{y_1, y_2}_{d}} \qquad \boxed{\underbrace{z_1, z_2}_{h}}$$

$$\Downarrow \qquad\qquad\qquad \Downarrow$$

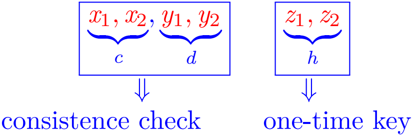
Practical PKE with IND-KL-CCA2 Security

- The variant of Cramer-Shoup cryptosystem ("KL-CS-PKE") presented by Naor-Segev [NS,CRYPTO09] is the most practical one.

$$\begin{array}{cc} \boxed{\underbrace{x_1, x_2}_{c}, \underbrace{y_1, y_2}_{d}} & \boxed{\underbrace{z_1, z_2}_{h}} \\ \Downarrow & \Downarrow \\ \text{consistence check} & \text{one-time key} \end{array}$$

Practical PKE with IND-KL-CCA2 Security

- The variant of Cramer-Shoup cryptosystem ("KL-CS-PKE") presented by Naor-Segev [NS,CRYPTO09] is the most practical one.



$\boxed{\underbrace{x_1, x_2}_{c}, \underbrace{y_1, y_2}_{d}}$ $\Downarrow$ consistence check

$\boxed{\underbrace{z_1, z_2}_{h}}$ $\Downarrow$ one-time key

Encryption	Decryption
$u_1 = g_1^r, u_2 = g_2^r, r \in \mathbb{Z}_q^*, s \in \{0,1\}^t$;	$\alpha = T(u_1, u_2, e, s)$;
$e = M \oplus \mathsf{Ext}(h^r, s)$;	If $v \neq u_1^{x_1+y_1\alpha} u_2^{x_2+y_2\alpha}$, output $\perp$;
$\alpha = T(u_1, u_2, e, s)$; $\quad v = c^r d^{r\alpha}$;	otherwise $M = e \oplus \mathsf{Ext}(u_1^{z_1} u_2^{z_2}, s)$;
Output (u_1, u_2, s, e, v).	Output M.

Figure: The KL-CS-PKE Scheme by Naor-Segev.

Practical PKE with IND-KL-CCA2 Security

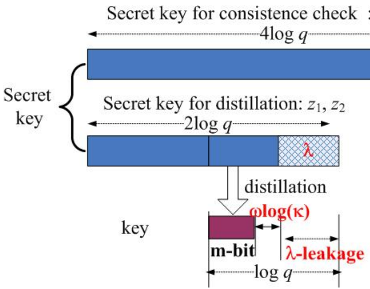
- The variant of Cramer-Shoup cryptosystem (“KL-CS-PKE”) presented by Naor-Segev [NS,CRYPTO09] is the most practical one.

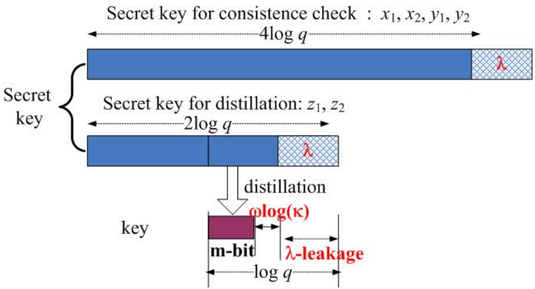
$$\begin{array}{cc} \boxed{\underbrace{x_1, x_2}_{c}, \underbrace{y_1, y_2}_{d}} & \boxed{\underbrace{z_1, z_2}_{h}} \\ \Downarrow & \Downarrow \\ \text{consistence check} & \text{one-time key} \end{array}$$

Encryption	Decryption
$u_1 = g_1^r, u_2 = g_2^r, r \in \mathbb{Z}_q^*, s \in \{0,1\}^t$;	$\alpha = T(u_1, u_2, e, s)$;
$e = M \oplus \mathsf{Ext}(h^r, s)$;	If $v \neq u_1^{x_1+y_1\alpha} u_2^{x_2+y_2\alpha}$, output $\perp$;
$\alpha = T(u_1, u_2, e, s)$; $\quad v = c^r d^{r\alpha}$;	otherwise $M = e \oplus \mathsf{Ext}(u_1^{z_1} u_2^{z_2}, s)$;
Output (u_1, u_2, s, e, v).	Output M.

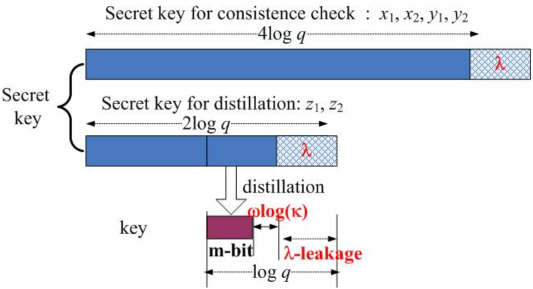
Figure: The KL-CS-PKE Scheme by Naor-Segev.

- It uses extractors to deal with key leakage.


Secret key for consistence check : x_1, x_2, y_1, y_2
4log q
λ
Secret key
Secret key for distillation: z_1, z_2
2log q
λ
distillation
ωlog(κ)
key
m-bit
λ-leakage
log q

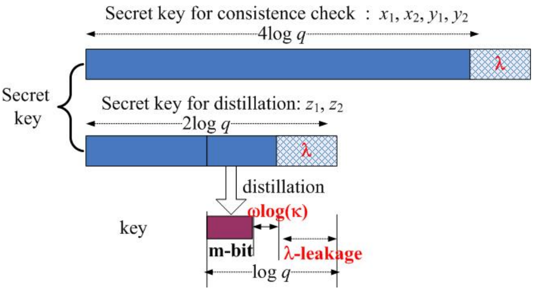

Secret key for consistence check : x_1, x_2, y_1, y_2
4log q
λ
Secret key
Secret key for distillation: z_1, z_2
2log q
λ
distillation
key
ωlog(κ)
m-bit
λ-leakage
log q

Undesirable dependency between λ and m: $\lambda + m \leq \log_2 q - \omega(\log \kappa)$.


Secret key for consistence check : x_1, x_2, y_1, y_2
4log q
λ
Secret key
Secret key for distillation: z_1, z_2
2log q
λ
distillation
ωlog(κ)
key
m-bit
λ-leakage
log q

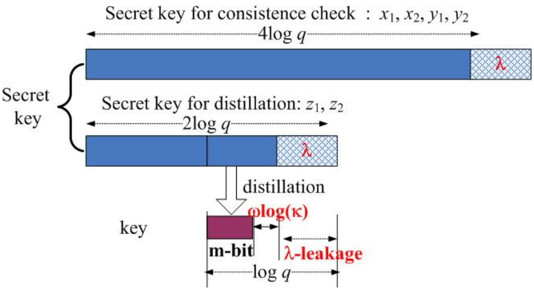
Undesirable dependency between λ and m: $\lambda + m \leq \log_2 q - \omega(\log \kappa)$.

- m is the bit length of plaintext;


Secret key for consistence check : x_1, x_2, y_1, y_2
4log q
λ
Secret key
Secret key for distillation: z_1, z_2
2log q
λ
distillation
ωlog(κ)
key
m-bit
λ-leakage
log q

Undesirable dependency between λ and m: $\lambda + m \leq \log_2 q - \omega(\log \kappa)$.

- m is the bit length of plaintext;
- q is the order of the group that CS is based.

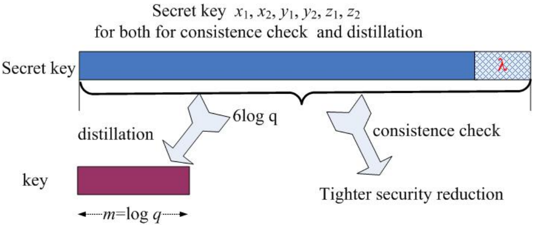

Secret key for consistence check : x_1, x_2, y_1, y_2
4log q
λ
Secret key
Secret key for distillation: z_1, z_2
2log q
λ
distillation
ωlog(κ)
key
m-bit
λ-leakage
log q

Undesirable dependency between λ and m: $\lambda + m \leq \log_2 q - \omega(\log \kappa)$.

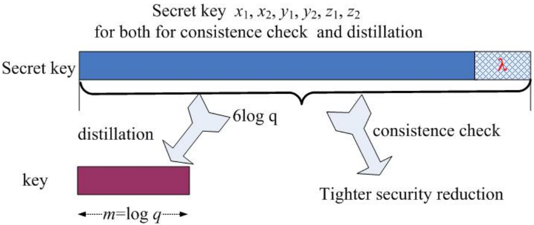
- m is the bit length of plaintext;
- q is the order of the group that CS is based.

Naor and Segev noted this and called for further refinement.

Our contributions


Secret key $x_1, x_2, y_1, y_2, z_1, z_2$
for both for consistence check and distillation
Secret key
λ
6log q
distillation
consistence check
key
Tighter security reduction
m=log q

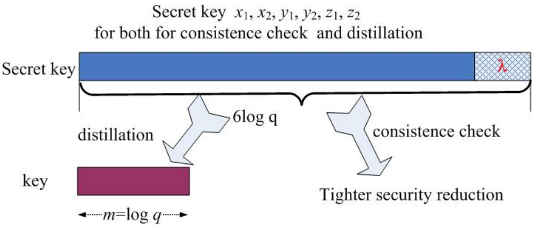
Our contributions



Secret key $x_1, x_2, y_1, y_2, z_1, z_2$
for both for consistence check and distillation
Secret key
λ
6log q
distillation
consistence check
key
Tighter security reduction
m=log q

- As a response to the calling-for of Naor and Segev [NS09], this work answers the open question proposed in [NS09], and follows a new technical line.

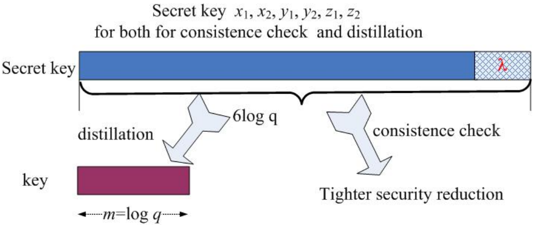
Our contributions



Secret key $x_1, x_2, y_1, y_2, z_1, z_2$
for both for consistence check and distillation
Secret key
λ
6log q
distillation
consistence check
key
Tighter security reduction
m=log q

- As a response to the calling-for of Naor and Segev [NS09], this work answers the open question proposed in [NS09], and follows a new technical line.
- Our proposal is almost efficient as the CS-PKE, and we show it is $\lambda \leq \log q - \omega(\log \kappa)$ leakage resilient. In addition, it has the following two advantages:

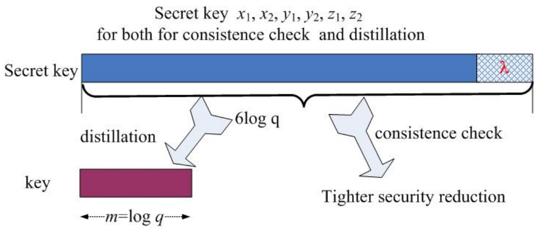
Our contributions



Secret key $x_1, x_2, y_1, y_2, z_1, z_2$
for both for consistence check and distillation
Secret key
λ
6log q
distillation
consistence check
key
Tighter security reduction
m=log q

- As a response to the calling-for of Naor and Segev [NS09], this work answers the open question proposed in [NS09], and follows a new technical line.
- Our proposal is almost efficient as the CS-PKE, and we show it is $\lambda \leq \log q - \omega(\log \kappa)$ leakage resilient. In addition, it has the following two advantages:
    - The plaintext space is the group $\mathbb{G}$, enjoying a constant size q. It is independent of the leakage parameter λ.

Our contributions


Secret key $x_1, x_2, y_1, y_2, z_1, z_2$
for both for consistence check and distillation
Secret key
λ
6log q
distillation
consistence check
key
Tighter security reduction
m=log q

- As a response to the calling-for of Naor and Segev [NS09], this work answers the open question proposed in [NS09], and follows a new technical line.
- Our proposal is almost efficient as the CS-PKE, and we show it is $\lambda \leq \log q - \omega(\log \kappa)$ leakage resilient. In addition, it has the following two advantages:
    - The plaintext space is the group $\mathbb{G}$, enjoying a constant size q. It is independent of the leakage parameter λ.
    - The security reduction is tighter than that of KL-CS-PKE [NS09].

Preliminaries

Statistical Distance, Min-Entropy, and Leftover Hash Lemma

Definition

Let X be a random variable, which takes value from a finite set $\mathcal{X}$.

Statistical Distance, Min-Entropy, and Leftover Hash Lemma

Definition

Let X be a random variable, which takes value from a finite set $\mathcal{X}$.

- The guessing probability of X is defined as $P_g(X) = \max_{x \in \mathcal{X}} \Pr[X = x]$.

Statistical Distance, Min-Entropy, and Leftover Hash Lemma

Definition

Let X be a random variable, which takes value from a finite set $\mathcal{X}$.

- The guessing probability of X is defined as $P_g(X) = \max_{x \in \mathcal{X}} \Pr[X = x]$.
- The min entropy of X is defined as $\mathbf{H}_\infty(X) := -\log P_g(X)$. The average min-entropy of X given Y is defined as the logarithm of the average guessing probability of X given Y, i.e.,

$$\tilde{\mathbf{H}}_\infty(X|Y) := -\log\left(\mathbb{E}_{y \leftarrow \mathcal{Y}}[2^{-\mathbf{H}_\infty(X|Y=y)}].\right)$$

Statistical Distance, Min-Entropy, and Leftover Hash Lemma

Definition

Let X be a random variable, which takes value from a finite set $\mathcal{X}$.

- The guessing probability of X is defined as $P_g(X) = \max_{x \in \mathcal{X}} \Pr[X = x]$.
- The min entropy of X is defined as $\mathbf{H}_\infty(X) := -\log P_g(X)$. The average min-entropy of X given Y is defined as the logarithm of the average guessing probability of X given Y, i.e.,

$$\tilde{\mathbf{H}}_\infty(X|Y) := -\log \left(\mathbb{E}_{y \leftarrow \mathcal{Y}}[2^{-\mathbf{H}_\infty(X|Y=y)}]. \right)$$

- The statistic distance of two distributions of variable X and Y is defined as

$$SD(X, Y) = \frac{1}{2} \sum_{a \in \mathcal{X}} |\Pr[X = a] - \Pr[Y = a]|.$$

Statistical Distance, Min-Entropy, and Leftover Hash Lemma

Definition

Let X be a random variable, which takes value from a finite set $\mathcal{X}$.

- The guessing probability of X is defined as $P_g(X) = \max_{x \in \mathcal{X}} \Pr[X = x]$.
- The min entropy of X is defined as $\mathbf{H}_\infty(X) := -\log P_g(X)$. The average min-entropy of X given Y is defined as the logarithm of the average guessing probability of X given Y, i.e.,

$$\tilde{\mathbf{H}}_\infty(X|Y) := -\log \left(\mathbb{E}_{y \leftarrow \mathcal{Y}}[2^{-\mathbf{H}_\infty(X|Y=y)}]. \right)$$

- The statistic distance of two distributions of variable X and Y is defined as

$$SD(X, Y) = \frac{1}{2} \sum_{a \in \mathcal{X}} |\Pr[X = a] - \Pr[Y = a]|.$$

Universal Hash and Examples

Definition

(Universal hash functions) A family of functions $\{H_k : \mathcal{X} \to \mathcal{Y}, k \in \mathcal{K}\}$ is universal if

$$\Pr_{k\in\mathcal{K}}\left[H_k(x_1) = H_k(x_2)\right] \leq \frac{1}{|\mathcal{Y}|}.$$

for all $x_1 \neq x_2$ with $x_1, x_2 \in \mathcal{X}$.

Universal Hash and Examples

Definition

(Universal hash functions) A family of functions $\{H_k : \mathcal{X} \to \mathcal{Y}, k \in \mathcal{K}\}$ is universal if

$$\Pr_{k\in\mathcal{K}}\left[H_k(x_1) = H_k(x_2)\right] \leq \frac{1}{|\mathcal{Y}|}.$$

for all $x_1 \neq x_2$ with $x_1, x_2 \in \mathcal{X}$.

Shoup05 : Universal hash $\{H_{k_1,k_2,\cdots,k_l} : \mathbb{F}_q^{l+1} \to \mathbb{F}_q, k_i \in \mathbb{F}_q, i = 1, 2, \cdots, l\}$

$$H_{k_1,k_2,\cdots,k_l}(x_0, x_1, \cdots, x_l) = x_0 + k_1x_1 + \cdots + k_lx_l.$$

All operations are in the prime field $\mathbb{F}_q$.

Universal Hash and Examples

Definition

(Universal hash functions) A family of functions $\{H_k : \mathcal{X} \to \mathcal{Y}, k \in \mathcal{K}\}$ is universal if

$$\Pr_{k\in\mathcal{K}}\left[H_k(x_1) = H_k(x_2)\right] \le \frac{1}{|\mathcal{Y}|}.$$

for all $x_1 \neq x_2$ with $x_1, x_2 \in \mathcal{X}$.

Shoup05 : Universal hash $\{H_{k_1,k_2,\cdots,k_l} : \mathbb{F}_q^{l+1} \to \mathbb{F}_q, k_i \in \mathbb{F}_q, i = 1, 2, \cdots, l\}$

$$H_{k_1,k_2,\cdots,k_l}(x_0, x_1, \cdots, x_l) = x_0 + k_1x_1 + \cdots + k_lx_l.$$

All operations are in the prime field $\mathbb{F}_q$.

- Let $\mathbb{G}$ be a multiplicative group of prime order q with generator g. Then $(\mathbb{Z}_q, +) \cong (\mathbb{G}, \cdot)$ with bijection $x \to g^x$.

Universal Hash and Examples

Definition

(Universal hash functions) A family of functions $\{H_k : \mathcal{X} \to \mathcal{Y}, k \in \mathcal{K}\}$ is universal if

$$\Pr_{k\in\mathcal{K}}\left[H_k(x_1) = H_k(x_2)\right] \le \frac{1}{|\mathcal{Y}|}.$$

for all $x_1 \neq x_2$ with $x_1, x_2 \in \mathcal{X}$.

Shoup05 : Universal hash $\{H_{k_1,k_2,\cdots,k_l} : \mathbb{F}_q^{l+1} \to \mathbb{F}_q, k_i \in \mathbb{F}_q, i = 1, 2, \cdots, l\}$

$$H_{k_1,k_2,\cdots,k_l}(x_0, x_1, \cdots, x_l) = x_0 + k_1 x_1 + \cdots + k_l x_l.$$

All operations are in the prime field $\mathbb{F}_q$.

- Let $\mathbb{G}$ be a multiplicative group of prime order q with generator g. Then $(\mathbb{Z}_q, +) \cong (\mathbb{G}, \cdot)$ with bijection $x \to g^x$.
- The family $\{H_{k_1,k_2,\cdots,k_l} : \mathbb{G}^{l+1} \to \mathbb{G}, k_i \in \mathbb{Z}_q, i = 1, 2, \cdots, l\}$ is universal, where

$$H_{k_1,k_2,\cdots,k_l}(g_0, g_1, \cdots, g_l) = g_0 \cdot g_1^{k_1} \cdot \ldots \cdot g_l^{k_l} (= g^{x_0 + k_1 x_1 + \cdots + k_l x_l}),$$

with $g_i = g^{x_i}$ for $i = 0, 1, \cdots, l$.

Leftover Hash Lemma[Dodis08]

Lemma

Assume $\{H_k : \mathcal{X} \to \mathcal{Y}\}_{k \in \mathcal{K}}$ *is a family of universal hash functions. For any random variables* $X \in \mathcal{X}$, $Z \in \mathcal{Z}$, *and* $K \leftarrow \mathcal{K}$,

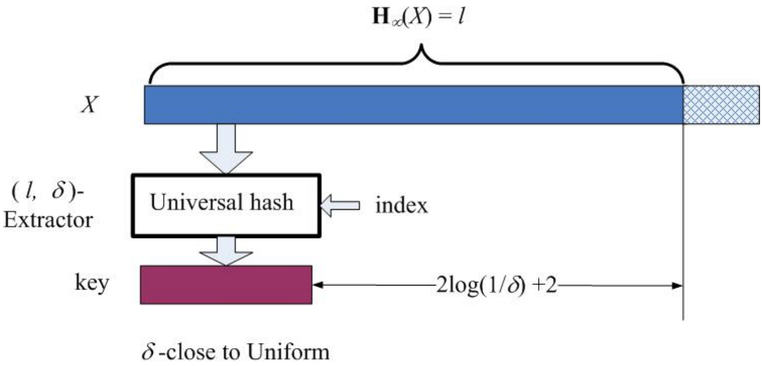
$$SD((H_k(X), K), (U_{\mathcal{Y}}, K)) \leq \frac{1}{2}\sqrt{P_c(X)|\mathcal{Y}|} \leq \frac{1}{2}\sqrt{2^{-\boldsymbol{H}_\infty(X)}|\mathcal{Y}|},$$

and

$$SD((H_k(X), K, Z), (U_{\mathcal{Y}}, K, Z)) \leq \frac{1}{2}\sqrt{2^{-\tilde{\boldsymbol{H}}_\infty(X|Z)}|\mathcal{Y}|},$$

where $U_{\mathcal{Y}}$ *denotes a uniform distribution over* $\mathcal{Y}$.

Extractors with Universal Hashing [Dodis08]



IND-KL-CCA2 Security

KL-CCA2 Game: *Adversary* $\mathcal{A}$ v.s. its *environment*

- Setup $\mathcal{A}$ queries the *key generation oracle*. The key generation oracle computes $(\mathsf{PK},\mathsf{SK})$ and responds with PK.

IND-KL-CCA2 Security

KL-CCA2 Game: *Adversary* $\mathcal{A}$ v.s. its *environment*

Setup $\mathcal{A}$ queries the *key generation oracle*. The key generation oracle computes (PK,SK) and responds with PK.

Morning Besides normal queries to decryption oracle $Dec(sk, \cdot)$, $\mathcal{A}$ is also allowed to make queries to a key-leakage oracle $KL(sk)$, as follows: $\mathcal{A}$ can adaptively query the oracle $\mathbf{KL}(sk)$ with any function $f_i, i \geq 1$, and then gets back $f_i(sk)$.

IND-KL-CCA2 Security

KL-CCA2 Game: *Adversary* $\mathcal{A}$ v.s. its *environment*

Setup $\mathcal{A}$ queries the *key generation oracle*. The key generation oracle computes (PK,SK) and responds with PK.

Morning Besides normal queries to decryption oracle $Dec(sk, \cdot)$, $\mathcal{A}$ is also allowed to make queries to a key-leakage oracle $KL(sk)$, as follows: $\mathcal{A}$ can adaptively query the oracle $\mathbf{KL}(sk)$ with any function $f_i, i \geq 1$, and then gets back $f_i(sk)$.

Noon $\mathcal{A}$ submits two messages (M_0, M_1) of equal length to the *encryption oracle*.

The encryption oracle picks a random bit $\sigma \in \{0, 1\}$ and responds with the “target" ciphertext $\psi^* = \mathsf{Encrypt}(\mathsf{PK}, m_\sigma)$.

IND-KL-CCA2 Security

KL-CCA2 Game: *Adversary* $\mathcal{A}$ v.s. its *environment*

Setup $\mathcal{A}$ queries the *key generation oracle*. The key generation oracle computes (PK,SK) and responds with PK.

Morning Besides normal queries to decryption oracle $Dec(sk, \cdot)$, $\mathcal{A}$ is also allowed to make queries to a key-leakage oracle $KL(sk)$, as follows: $\mathcal{A}$ can adaptively query the oracle $\mathbf{KL}(sk)$ with any function $f_i, i \geq 1$, and then gets back $f_i(sk)$.

Noon $\mathcal{A}$ submits two messages (M_0, M_1) of equal length to the *encryption oracle*.

The encryption oracle picks a random bit $\sigma \in \{0, 1\}$ and responds with the "target" ciphertext $\psi^* = \mathsf{Encrypt}(\mathsf{PK}, m_\sigma)$.

Afternoon $\mathcal{A}$ continues to query the *decryption oracle* with arbitrary ciphertext ψ under the restriction: $\psi \neq \psi^*$, but the $KL(sk)$ oracle access is denied in the afternoon.

IND-KL-CCA2 Security

KL-CCA2 Game: *Adversary* $\mathcal{A}$ v.s. its *environment*

Setup $\mathcal{A}$ queries the *key generation oracle*. The key generation oracle computes (PK,SK) and responds with PK.

Morning Besides normal queries to decryption oracle $Dec(sk, \cdot)$, $\mathcal{A}$ is also allowed to make queries to a key-leakage oracle $KL(sk)$, as follows: $\mathcal{A}$ can adaptively query the oracle $\mathbf{KL}(sk)$ with any function $f_i, i \geq 1$, and then gets back $f_i(sk)$.

Noon $\mathcal{A}$ submits two messages (M_0, M_1) of equal length to the *encryption oracle*.

The encryption oracle picks a random bit $\sigma \in \{0, 1\}$ and responds with the “target" ciphertext $\psi^* = \mathsf{Encrypt}(\mathsf{PK}, m_\sigma)$.

Afternoon $\mathcal{A}$ continues to query the *decryption oracle* with arbitrary ciphertext ψ under the restriction: $\psi \neq \psi^*$, but the $KL(sk)$ oracle access is denied in the afternoon.

Guess $\mathcal{A}$ outputs a guess bit $\hat{\sigma} \in \{0, 1\}$.

IND-KL-CCA2 Security

KL-CCA2 Game: *Adversary* $\mathcal{A}$ v.s. its *environment*

Setup $\mathcal{A}$ queries the *key generation oracle*. The key generation oracle computes $(\mathsf{PK},\mathsf{SK})$ and responds with PK.

Morning Besides normal queries to decryption oracle $Dec(sk,\cdot)$, $\mathcal{A}$ is also allowed to make queries to a key-leakage oracle $KL(sk)$, as follows: $\mathcal{A}$ can adaptively query the oracle $\mathbf{KL}(sk)$ with any function $f_i, i \geq 1$, and then gets back $f_i(sk)$.

Noon $\mathcal{A}$ submits two messages (M_0, M_1) of equal length to the *encryption oracle*.
The encryption oracle picks a random bit $\sigma \in \{0,1\}$ and responds with the “target" ciphertext $\psi^* = \mathsf{Encrypt}(\mathsf{PK}, m_\sigma)$.

Afternoon $\mathcal{A}$ continues to query the *decryption oracle* with arbitrary ciphertext ψ under the restriction: $\psi \neq \psi^*$, but the $KL(sk)$ oracle access is denied in the afternoon.

Guess $\mathcal{A}$ outputs a guess bit $\hat{\sigma} \in \{0,1\}$.

The KL-CCA *advantage of* $\mathcal{A}$ *against* a PKE scheme:

$$\mathsf{Adv}_{\mathcal{A}} = |\Pr[\sigma = \hat{\sigma}] - 1/2|.$$

Overview of KL-CS-PKE by Naor and Segev [CRYPTO09]

Key Generation A group $(\mathbb{G}, q, g)$. T: TCR hash function.

Overview of KL-CS-PKE by Naor and Segev [CRYPTO09]

Key Generation A group $(\mathbb{G}, q, g)$. T: TCR hash function.

$$\begin{cases} \textbf{secret key: } x_1, x_2, y_1, y_2, z_1, z_2 \in_R \mathbb{Z}_q \\ \textbf{public key: } c = g_1^{x_1} g_2^{x_2},\ d = g_1^{y_1} g_2^{y_2},\ h = g_1^{z_1} g_2^{z_2}. \end{cases}$$

Overview of KL-CS-PKE by Naor and Segev [CRYPTO09]

Key Generation A group $(\mathbb{G}, q, g)$. T: TCR hash function.

$$\begin{cases} \textbf{secret key: } x_1, x_2, y_1, y_2, z_1, z_2 \in_R \mathbb{Z}_q \\ \textbf{public key: } c = g_1^{x_1} g_2^{x_2},\ d = g_1^{y_1} g_2^{y_2},\ h = g_1^{z_1} g_2^{z_2}. \end{cases}$$

$$\mathsf{PK} = (\mathbb{G}, q, g, \mathsf{T}, c, d, h), \mathsf{SK} = (x_1, x_2, y_1, y_2, z_1, z_2).$$

Overview of KL-CS-PKE by Naor and Segev [CRYPTO09]

Key Generation A group $(\mathbb{G}, q, g)$. T: TCR hash function.

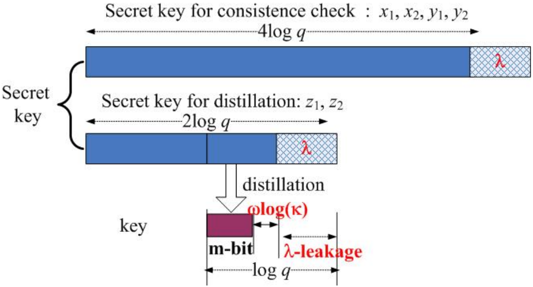
$$\begin{cases} \textbf{secret key: } x_1, x_2, y_1, y_2, z_1, z_2 \in_R \mathbb{Z}_q \\ \textbf{public key: } c = g_1^{x_1} g_2^{x_2},\ d = g_1^{y_1} g_2^{y_2},\ h = g_1^{z_1} g_2^{z_2}. \end{cases}$$

$$\mathsf{PK} = (\mathbb{G}, q, g, \mathsf{T}, c, d, h), \mathsf{SK} = (x_1, x_2, y_1, y_2, z_1, z_2).$$

Encryption	Decryption
$u_1 = g_1^r, u_2 = g_2^r, r \in \mathbb{Z}_q^*, s \in \{0,1\}^t$;	$\alpha = T(u_1, u_2, e, s)$;
$e = M \oplus \mathsf{Ext}(h^r, s)$;	If $v \neq u_1^{x_1 + y_1\alpha} u_2^{x_2 + y_2\alpha}$, output $\perp$;
$\alpha = T(u_1, u_2, e, s)$; $\quad v = c^r d^{r\alpha}$;	otherwise $M = e \oplus \mathsf{Ext}(u_1^{z_1} u_2^{z_2}, s)$;
Output (u_1, u_2, s, e, v).	Output M.

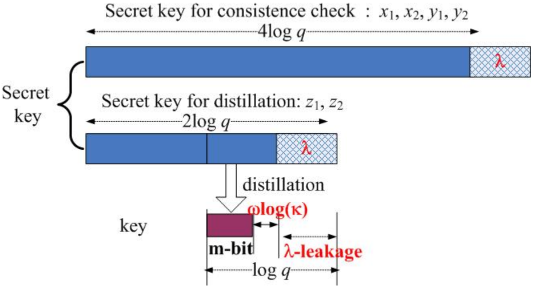
Figure: The KL-CS-PKE Scheme by Naor-Segev.

Overview of KL-CS-PKE by Naor and Segev [CRYPTO09]



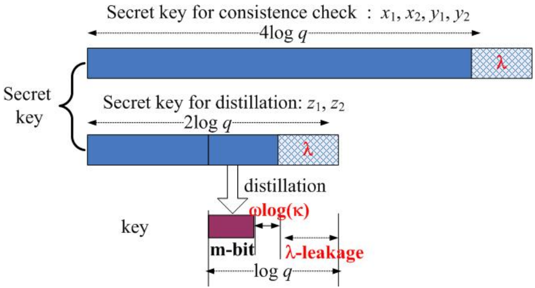
- The key leakage of CS-PKE brings two effects:

Overview of KL-CS-PKE by Naor and Segev [CRYPTO09]


Secret key for consistence check : x_1, x_2, y_1, y_2
4log q
λ
Secret key
Secret key for distillation: z_1, z_2
2log q
λ
distillation
key
ωlog(κ)
m-bit
λ-leakage
log q

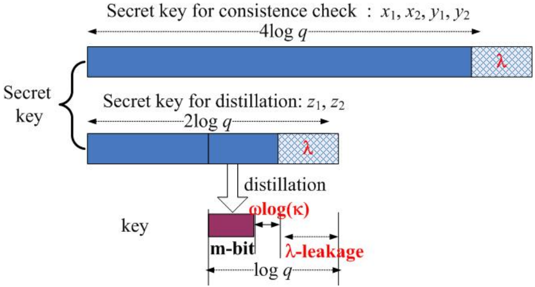
- The key leakage of CS-PKE brings two effects:
  - The information leakage related to (x_1, x_2, y_1, y_2) makes the security reduction looser than that of CS-PKE.

Overview of KL-CS-PKE by Naor and Segev [CRYPTO09]


Secret key for consistence check : x_1, x_2, y_1, y_2
4log q
λ
Secret key
Secret key for distillation: z_1, z_2
2log q
λ
distillation
ωlog(κ)
key
m-bit
λ-leakage
log q

- The key leakage of CS-PKE brings two effects:
  - The information leakage related to (x_1, x_2, y_1, y_2) makes the security reduction looser than that of CS-PKE.
  - The information leakage related to (z_1, z_2) makes it unsuitable to mask the plaintext directly. Naor and Segev employed an extractor Ext to distill a random shorter key from the ephemeral key $u_1^{z_1} u_2^{z_2}$ that is in turn used to mask the plaintext.

Overview of KL-CS-PKE by Naor and Segev [CRYPTO09]


Secret key for consistence check : x_1, x_2, y_1, y_2
4log q
λ
Secret key
Secret key for distillation: z_1, z_2
2log q
λ
distillation
key
ωlog(κ)
m-bit
λ-leakage
log q

- The key leakage of CS-PKE brings two effects:
  - The information leakage related to (x_1, x_2, y_1, y_2) makes the security reduction looser than that of CS-PKE.
  - The information leakage related to (z_1, z_2) makes it unsuitable to mask the plaintext directly. Naor and Segev employed an extractor Ext to distill a random shorter key from the ephemeral key $u_1^{z_1} u_2^{z_2}$ that is in turn used to mask the plaintext.
- $m \leq \log_2 q - \lambda - \omega(\log \kappa)$.

Our Proposal

New Idea and Key Observation

- The new idea is that all the three parts of secret key, namely (x_1, x_2), (y_1, y_2) and (z_1, z_2), are involved both in the ciphertext consistence check and the random distillation to mask plaintexts.

New Idea and Key Observation

- The new idea is that all the three parts of secret key, namely (x_1, x_2), (y_1, y_2) and (z_1, z_2), are involved both in the ciphertext consistence check and the random distillation to mask plaintexts.
- The key observation is:
  - The three parts of secret key altogether imply larger average min-entropy, even conditioned on all the leaked information bounded by $\lambda \leq \log q - \omega(\log \kappa)$. Larger min-entropy implies more randomness can be distilled.

New Idea and Key Observation

- The new idea is that all the three parts of secret key, namely (x_1, x_2), (y_1, y_2) and (z_1, z_2), are involved both in the ciphertext consistence check and the random distillation to mask plaintexts.
- The key observation is:
    - The three parts of secret key altogether imply larger average min-entropy, even conditioned on all the leaked information bounded by $\lambda \leq \log q - \omega(\log \kappa)$. Larger min-entropy implies more randomness can be distilled.
    - On the other hand, we use a special universal hash function (i.e., $H_s(a, b) = a \cdot b^s$, for $a, b \in \mathbb{G}$ and $s \in \mathbb{Z}_q^*$) as an extractor, where $a = (cd)^r$ and $b = h^r$ for $r \in \mathbb{Z}_q^*$ with our proposal, which allows plaintext space to be $\mathbb{G}$, and makes the security proof neat and tighter.

New Idea and Key Observation

- The new idea is that all the three parts of secret key, namely (x_1, x_2), (y_1, y_2) and (z_1, z_2), are involved both in the ciphertext consistence check and the random distillation to mask plaintexts.
- The key observation is:
  - The three parts of secret key altogether imply larger average min-entropy, even conditioned on all the leaked information bounded by $\lambda \leq \log q - \omega(\log \kappa)$. Larger min-entropy implies more randomness can be distilled.
  - On the other hand, we use a special universal hash function (i.e., $H_s(a, b) = a \cdot b^s$, for $a, b \in \mathbb{G}$ and $s \in \mathbb{Z}_q^*$) as an extractor, where $a = (cd)^r$ and $b = h^r$ for $r \in \mathbb{Z}_q^*$ with our proposal, which allows plaintext space to be $\mathbb{G}$, and makes the security proof neat and tighter.
- The actual design of our proposal was also carefully guided by the underlying analysis, particularly for ensuring non-zero matrix determinant.

Overview of Our Proposal

Encryption	Decryption
$u_1 = g_1^r, u_2 = g_2^r, \quad r, s \in \mathbb{Z}_q^*;$	$\alpha = T(u_1, u_2, e, s);$
$\boxed{e = M \cdot (cd)^r \cdot h^{rs};}$	If $\boxed{v \neq u_1^{x_1+y_1\alpha+z_1} u_2^{x_2+y_2\alpha+z_2},}$ outpu
$\alpha = T(u_1, u_2, e, s);$	otherwise
$\boxed{v = (c \cdot h)^r \cdot d^{r\alpha};}$	$\boxed{M = e \cdot u_1^{-(x_1+y_1+z_1 s)} u_2^{-(x_2+y_2+z_2}}$
Output (u_1, u_2, s, e, v)	Output $M.$

Figure: Our proposal

Proof Outline

Main Theorem

Theorem

The above scheme is $(\log q, \lambda, \epsilon)$*-IND-KL-CCA2 secure public key encryption scheme. Here* q *is the prime order of the group* $\mathbb{G}$ *that* PKE *is based on,* $\lambda \leq \log q - \omega(\log \kappa)$ *(more precisely,* $\lambda \leq \log q - 2\log\frac{1}{\delta} + 2$ *where* $\delta = \frac{2^{\lambda/2-1}}{\sqrt{q}}$*)) and*

$$\epsilon \leq \mathbf{Adv}^{DDH}(\kappa) + \mathbf{Adv}^{TCR}(\kappa) + \frac{2^{\lambda} Q(\kappa)}{q - Q(\kappa)} + \frac{2^{\lambda/2-1}}{\sqrt{q}},$$

where $Q(\kappa)$ *is the number of decryption queries.*

Main Theorem

Theorem

The above scheme is $(\log q, \lambda, \epsilon)$*-IND-KL-CCA2 secure public key encryption scheme. Here* q *is the prime order of the group* $\mathbb{G}$ *that* PKE *is based on,* $\lambda \leq \log q - \omega(\log \kappa)$ *(more precisely,* $\lambda \leq \log q - 2\log\frac{1}{\delta} + 2$ *where* $\delta = \frac{2^{\lambda/2-1}}{\sqrt{q}}$*)) and*

$$\epsilon \leq \mathbf{Adv}^{DDH}(\kappa) + \mathbf{Adv}^{TCR}(\kappa) + \frac{2^{\lambda} Q(\kappa)}{q - Q(\kappa)} + \frac{2^{\lambda/2-1}}{\sqrt{q}},$$

where $Q(\kappa)$ *is the number of decryption queries.*

- Plaintext length $m = \log q$;

Main Theorem

Theorem

The above scheme is $(\log q, \lambda, \epsilon)$*-IND-KL-CCA2 secure public key encryption scheme. Here* q *is the prime order of the group* $\mathbb{G}$ *that* PKE *is based on,* $\lambda \leq \log q - \omega(\log \kappa)$ *(more precisely,* $\lambda \leq \log q - 2\log\frac{1}{\delta} + 2$ *where* $\delta = \frac{2^{\lambda/2-1}}{\sqrt{q}}$*)) and*

$$\epsilon \leq \mathbf{Adv}^{DDH}(\kappa) + \mathbf{Adv}^{TCR}(\kappa) + \frac{2^{\lambda}Q(\kappa)}{q - Q(\kappa)} + \frac{2^{\lambda/2-1}}{\sqrt{q}},$$

where $Q(\kappa)$ *is the number of decryption queries.*

- Plaintext length $m = \log q$;
- Leakage parameter $\lambda = \log q - o(1)$.

Security Proof

We proceed with a series of games played between a simulator $\mathcal{D}$ and an adversary $\mathcal{A}$, and show that Game i and Game $i+1$ are indistinguishable except with negligible probability, $i = 0, 1, 2, 3, 4, 5$. We define S_i as the event that the adversary $\mathcal{A}$ outputs a correct guess of b.

Security Proof

We proceed with a series of games played between a simulator $\mathcal{D}$ and an adversary $\mathcal{A}$, and show that Game i and Game $i+1$ are indistinguishable except with negligible probability, $i = 0, 1, 2, 3, 4, 5$. We define S_i as the event that the adversary $\mathcal{A}$ outputs a correct guess of b.

Game 0: The original KL-CCA2 game.

Security Proof

We proceed with a series of games played between a simulator $\mathcal{D}$ and an adversary $\mathcal{A}$, and show that Game i and Game $i+1$ are indistinguishable except with negligible probability, $i = 0, 1, 2, 3, 4, 5$. We define S_i as the event that the adversary $\mathcal{A}$ outputs a correct guess of b.

Game 0: The original KL-CCA2 game.

Game 1: The same as Game 0 except for the generation of the challenge ciphertext ψ^*. In this game, the simulator generates the target ciphertext $\psi^* = (u_1^*, u_2^*, e^*, s^*, v^*)$ with its secret key SK as follows.

Security Proof

We proceed with a series of games played between a simulator $\mathcal{D}$ and an adversary $\mathcal{A}$, and show that Game i and Game $i+1$ are indistinguishable except with negligible probability, $i = 0, 1, 2, 3, 4, 5$. We define S_i as the event that the adversary $\mathcal{A}$ outputs a correct guess of b.

Game 0: The original KL-CCA2 game.

Game 1: The same as Game 0 except for the generation of the challenge ciphertext ψ^*. In this game, the simulator generates the target ciphertext $\psi^* = (u_1^*, u_2^*, e^*, s^*, v^*)$ with its secret key SK as follows.
$e = M \cdot {u_1}^{-(x_1+y_1+z_1 s)} {u_2}^{-(x_2+y_2+z_2 s)} = M \cdot (cd)^r \cdot h^{rs};$

Security Proof

We proceed with a series of games played between a simulator $\mathcal{D}$ and an adversary $\mathcal{A}$, and show that Game i and Game $i+1$ are indistinguishable except with negligible probability, $i = 0, 1, 2, 3, 4, 5$. We define S_i as the event that the adversary $\mathcal{A}$ outputs a correct guess of b.

Game 0: The original KL-CCA2 game.

Game 1: The same as Game 0 except for the generation of the challenge ciphertext ψ^*. In this game, the simulator generates the target ciphertext $\psi^* = (u_1^*, u_2^*, e^*, s^*, v^*)$ with its secret key SK as follows.
$e = M \cdot u_1^{-(x_1+y_1+z_1 s)} u_2^{-(x_2+y_2+z_2 s)} = M \cdot (cd)^r \cdot h^{rs};$
$v = u_1^{x_1+y_1 \alpha+z_1} u_2^{x_2+y_2 \alpha+z_2} = (c \cdot h)^r \cdot d^{r\alpha}.$

Security Proof

We proceed with a series of games played between a simulator $\mathcal{D}$ and an adversary $\mathcal{A}$, and show that Game i and Game $i+1$ are indistinguishable except with negligible probability, $i = 0, 1, 2, 3, 4, 5$. We define S_i as the event that the adversary $\mathcal{A}$ outputs a correct guess of b.

Game 0: The original KL-CCA2 game.

Game 1: The same as Game 0 except for the generation of the challenge ciphertext ψ^*. In this game, the simulator generates the target ciphertext $\psi^* = (u_1^*, u_2^*, e^*, s^*, v^*)$ with its secret key SK as follows.
$e = M \cdot u_1^{-(x_1+y_1+z_1 s)} u_2^{-(x_2+y_2+z_2 s)} = M \cdot (cd)^r \cdot h^{rs};$
$v = u_1^{x_1+y_1\alpha+z_1} u_2^{x_2+y_2\alpha+z_2} = (c \cdot h)^r \cdot d^{r\alpha}.$

The change is only conceptual, and thus $\Pr[S_1] = \Pr[S_0]$.

Game 2: Same as Game 1 except for the generation of the challenge ciphertext $C^* = (u_1^*, u_2^*, e^*, s^*, v^*)$, where $u_1^* = g_1^{r_1^*}, u_2^* = g_2^{r_2^*}$, with r_1^*, r_2^* chosen uniformly at random from $\mathbb{Z}_q^*$.

Game 2: Same as Game 1 except for the generation of the challenge ciphertext $C^* = (u_1^*, u_2^*, e^*, s^*, v^*)$, where $u_1^* = g_1^{r_1^*}, u_2^* = g_2^{r_2^*}$, with r_1^*, r_2^* chosen uniformly at random from $\mathbb{Z}_q^*$.

- In Game 1: (g_1, g_2, u_1^*, u_2^*) is a DDH tuple, i.e, $u_1^* = g_1^r, u_2^* = g_2^r$.

Game 2: Same as Game 1 except for the generation of the challenge ciphertext $C^* = (u_1^*, u_2^*, e^*, s^*, v^*)$, where $u_1^* = g_1^{r_1^*}, u_2^* = g_2^{r_2^*}$, with r_1^*, r_2^* chosen uniformly at random from $\mathbb{Z}_q^*$.

- In Game 1: (g_1, g_2, u_1^*, u_2^*) is a DDH tuple, i.e, $u_1^* = g_1^r, u_2^* = g_2^r$.
- In Game 2: (g_1, g_2, u_1^*, u_2^*) is a random tuple, i.e, $u_1^* = g_1^{r_1^*}, u_2^* = g_2^{r_2^*}$.

Game 2: Same as Game 1 except for the generation of the challenge ciphertext $C^* = (u_1^*, u_2^*, e^*, s^*, v^*)$, where $u_1^* = g_1^{r_1^*}, u_2^* = g_2^{r_2^*}$, with r_1^*, r_2^* chosen uniformly at random from $\mathbb{Z}_q^*$.

- In Game 1: (g_1, g_2, u_1^*, u_2^*) is a DDH tuple, i.e, $u_1^* = g_1^r, u_2^* = g_2^r$.
- In Game 2: (g_1, g_2, u_1^*, u_2^*) is a random tuple, i.e, $u_1^* = g_1^{r_1^*}, u_2^* = g_2^{r_2^*}$.

By the DDH assumption, $|\Pr[S_2] - \Pr[S_1]|$ is negligible.

Game 3: Same as Game 2 except that the simulator applies a special rejection rule. Let F denote the event that there exists a decryption query of the form $C = (u_1, u_2, e, s, v)$ such that $C \neq C^*$ but $T(u_1, u_2, e, s) = T(u_1^*, u_2^*, e^*, s^*)$, which means a hash collision occurs. The simulator $\mathcal{D}$ rejects the corresponding queried ciphertext C when F occurs.

Game 3: Same as Game 2 except that the simulator applies a special rejection rule. Let F denote the event that there exists a decryption query of the form $C = (u_1, u_2, e, s, v)$ such that $C \neq C^*$ but $T(u_1, u_2, e, s) = T(u_1^*, u_2^*, e^*, s^*)$, which means a hash collision occurs. The simulator $\mathcal{D}$ rejects the corresponding queried ciphertext C when F occurs.

By the TCR property of T, $|\Pr[S_3] - \Pr[S_2]|$ is negligible.

Game 4: Same as Game 3, except that the simulator applies a special rejection rule. If $\mathcal{A}$ asks for decryption of an invalid ciphertext $C = (u_1, u_2, e, s, v)$, i.e., (g_1, g_2, u_1, u_2) is not a DDH tuple, the simulator $\mathcal{D}$ rejects with $\perp$ and the game aborts.

Game 4: Same as Game 3, except that the simulator applies a special rejection rule. If $\mathcal{A}$ asks for decryption of an invalid ciphertext $C = (u_1, u_2, e, s, v)$, i.e., (g_1, g_2, u_1, u_2) is not a DDH tuple, the simulator $\mathcal{D}$ rejects with $\perp$ and the game aborts.

- In Game 3: $C = (u_1, u_2, e, s, v)$ is rejected if it is not consistent, i.e., $v \neq u_1^{x_1 + y_1 \alpha} u_2^{x_2 + y_2 \alpha}$.

Game 4: Same as Game 3, except that the simulator applies a special rejection rule. If $\mathcal{A}$ asks for decryption of an invalid ciphertext $C = (u_1, u_2, e, s, v)$, i.e., (g_1, g_2, u_1, u_2) is not a DDH tuple, the simulator $\mathcal{D}$ rejects with $\perp$ and the game aborts.

- In Game 3: $C = (u_1, u_2, e, s, v)$ is rejected if it is not consistent, i.e., $v \neq u_1^{x_1+y_1\alpha} u_2^{x_2+y_2\alpha}$.
- In Game 4: $C = (u_1, u_2, e, s, v)$ is rejected if it is not consistent, i.e. $(\log_{g_1} u_1 \neq \log_{g_2} u_2)$.

Game 4: Same as Game 3, except that the simulator applies a special rejection rule. If $\mathcal{A}$ asks for decryption of an invalid ciphertext $C = (u_1, u_2, e, s, v)$, i.e., (g_1, g_2, u_1, u_2) is not a DDH tuple, the simulator $\mathcal{D}$ rejects with $\perp$ and the game aborts.

- In Game 3: $C = (u_1, u_2, e, s, v)$ is rejected if it is not consistent, i.e., $v \neq u_1^{x_1+y_1\alpha} u_2^{x_2+y_2\alpha}$.
- In Game 4: $C = (u_1, u_2, e, s, v)$ is rejected if it is not consistent, i.e. $(\log_{g_1} u_1 \neq \log_{g_2} u_2)$.

Let F' be the event that $\mathcal{D}$ outputs $\perp$ for a consistent but valid ciphertext, we have:
$|\Pr[S_4| \neq F'] - \Pr[S_3| \neq F']| \leq \Pr[F']$,
$|\Pr[S_4] - \Pr[S_3]| \leq \Pr[F']$.

Game 4: Same as Game 3, except that the simulator applies a special rejection rule. If $\mathcal{A}$ asks for decryption of an invalid ciphertext $C = (u_1, u_2, e, s, v)$, i.e., (g_1, g_2, u_1, u_2) is not a DDH tuple, the simulator $\mathcal{D}$ rejects with $\perp$ and the game aborts.

- In Game 3: $C = (u_1, u_2, e, s, v)$ is rejected if it is not consistent, i.e., $v \neq u_1^{x_1+y_1\alpha} u_2^{x_2+y_2\alpha}$.
- In Game 4: $C = (u_1, u_2, e, s, v)$ is rejected if it is not consistent, i.e. $(\log_{g_1} u_1 \neq \log_{g_2} u_2)$.

Let F' be the event that $\mathcal{D}$ outputs $\perp$ for a consistent but valid ciphertext, we have:
$|\Pr[S_4| \neq F'] - \Pr[S_3| \neq F']| \leq \Pr[F']$,
$|\Pr[S_4] - \Pr[S_3]| \leq \Pr[F']$.

Below, we analyze the probability that the event F' occurs.

Analyzing the Event F'

Let $\beta = \log_{g_1} g_2$. Firstly, note that by submitting valid ciphertexts to the decryption oracle, the adversary only learns linear combinations of the constraints $\log_{g_1} c = x_1 + \beta x_2$, $\log_{g_1} d = y_1 + \beta y_2$ and $\log_{g_1} h = z_1 + \beta z_2$, which are already known from the public-keys. Also note that $q, g_1, g_2, T, u_1^*, u_2^*, s^*, \sigma$ all are independent of the secret key.

Analyzing the Event F'

Let $\beta = \log_{g_1} g_2$. Firstly, note that by submitting valid ciphertexts to the decryption oracle, the adversary only learns linear combinations of the constraints $\log_{g_1} c = x_1 + \beta x_2$, $\log_{g_1} d = y_1 + \beta y_2$ and $\log_{g_1} h = z_1 + \beta z_2$, which are already known from the public-keys. Also note that $q, g_1, g_2, T, u_1^*, u_2^*, s^*, \sigma$ all are independent of the secret key.

From the view of the attack and the λ-bit key leakage, what can be learnt about the secret-keys can be formulated by the following equations, where

Analyzing the Event F'

Let $\beta = \log_{g_1} g_2$. Firstly, note that by submitting valid ciphertexts to the decryption oracle, the adversary only learns linear combinations of the constraints $\log_{g_1} c = x_1 + \beta x_2$, $\log_{g_1} d = y_1 + \beta y_2$ and $\log_{g_1} h = z_1 + \beta z_2$, which are already known from the public-keys. Also note that $q, g_1, g_2, T, u_1^*, u_2^*, s^*, \sigma$ all are independent of the secret key.

From the view of the attack and the λ-bit key leakage, what can be learnt about the secret-keys can be formulated by the following equations, where

$$\begin{array}{rcl}
\log_{g_1} c & = & x_1 + \beta x_2 \\
\log_{g_1} d & = & y_1 + \beta y_2 \\
\log_{g_1} h & = & z_1 + \beta z_2 \\
\log_{g_1} v^* & = & r_1^* x_1 + r_2^* \beta x_2 + \alpha^* r_1^* y_1 + \alpha^* r_2^* \beta y_2 + r_1^* z_1 + r_2^* \beta z_2 \\
\log_{g_1} e^*/M_b & = & r_1^* x_1 + r_2^* \beta x_2 + r_1^* y_1 + r_2^* \beta y_2 + s^* r_1^* z_1 + s^* r_2^* \beta z_2; \\
 & & \lambda\text{-bit leakage of } (x_1, x_2, y_1, y_2, z_1, z_2).
\end{array}$$

As the secret key elements $(x_1, x_2, y_1, y_2, z_1, z_2)$ are uniformly chosen from $\mathbb{Z}_q^6$, according to the average min-entropy theory, we have

$$\begin{aligned}
& \tilde{\mathrm{H}}_\infty((x_1, x_2, y_1, y_2, z_1, z_2)|c, d, h, C^*, M_b, \lambda\text{-leakage}) \\
= \; & \tilde{\mathrm{H}}_\infty((x_1, x_2, y_1, y_2, z_1, z_2)|c, d, h, u_1^*, u_2^*, s^*, v^*, e^*/M_b, \lambda\text{-leakage}) \\
= \; & \tilde{\mathrm{H}}_\infty((x_1, x_2, y_1, y_2, z_1, z_2)|c, d, h, v^*, e^*/M_b, \lambda\text{-leakage}) \qquad (1) \\
\geq \; & \log q - \lambda. \qquad (2)
\end{aligned}$$

As the secret key elements $(x_1, x_2, y_1, y_2, z_1, z_2)$ are uniformly chosen from $\mathbb{Z}_q^6$, according to the average min-entropy theory, we have

$$\begin{aligned}
& \tilde{\mathrm{H}}_\infty((x_1, x_2, y_1, y_2, z_1, z_2)|c, d, h, C^*, M_b, \lambda\text{-leakage}) \\
= \;& \tilde{\mathrm{H}}_\infty((x_1, x_2, y_1, y_2, z_1, z_2)|c, d, h, u_1^*, u_2^*, s^*, v^*, e^*/M_b, \lambda\text{-leakage}) \\
= \;& \tilde{\mathrm{H}}_\infty((x_1, x_2, y_1, y_2, z_1, z_2)|c, d, h, v^*, e^*/M_b, \lambda\text{-leakage}) \quad (1) \\
\geq \;& \log q - \lambda. \quad (2)
\end{aligned}$$

Let $C = (u_1, u_2, e, s, v)$ be the first invalid ciphertext submitted by $\mathcal{A}$. Let $r_1 = \log_{g_1} u_1$ and $r_2 = \log_{g_1} u_2$, then $r_1 \neq r_2$. We must have:

$$\begin{array}{rcl}
\log_{g_1} c & = & x_1 + \beta x_2; \\
\log_{g_1} d & = & y_1 + \beta y_2; \\
\log_{g_1} h & = & z_1 + \beta z_2; \\
\log_{g_1} v^* & = & r_1^* x_1 + r_2^* \beta x_2 + \alpha^* r_1^* y_1 + \alpha^* r_2^* \beta y_2 + r_1^* z_1 + r_2^* \beta z_2; \\
\log_{g_1} e^*/M_b & = & r_1^* x_1 + r_2^* \beta x_2 + r_1^* y_1 + r_2^* \beta y_2 + s^* r_1^* z_1 + s^* r_2^* \beta z_2; \\
\log_{g_1} v & = & r_1 x_1 + r_2 \beta x_2 + \alpha r_1 y_1 + \alpha r_2 \beta y_2 + r_1 z_1 + r_2 \beta z_2.
\end{array} \quad (3)$$

Let $\mathbf{A}=\begin{pmatrix} 1 & \beta & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & \beta & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \beta \\ r_1^* & r_2^*\beta & \alpha^* r_1^* & \alpha^* r_2^*\beta & r_1^* & r_2^*\beta \\ r_1^* & r_2^*\beta & r_1^* & r_2^*\beta & s^* r_1^* & s^* r_2^*\beta \\ r_1 & r_2\beta & \alpha r_1 & \alpha r_2\beta & r_1 & r_2\beta \end{pmatrix}$. This is distilled into:

$$\begin{pmatrix} 1 & \beta & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & \beta & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \beta \\ r_1^* & r_2^*\beta & \alpha^* r_1^* & \alpha^* r_2^*\beta & r_1^* & r_2^*\beta \\ r_1^* & r_2^*\beta & r_1^* & r_2^*\beta & s^* r_1^* & s^* r_2^*\beta \\ r_1 & r_2\beta & \alpha r_1 & \alpha r_2\beta & r_1 & r_2\beta \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ y_1 \\ y_2 \\ z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} \log_{g_1}\ c \\ \log_{g_1}\ d \\ \log_{g_1}\ h \\ \log_{g_1}\ v^* \\ \log_{g_1}\ e^*/M_b \\ \log_{g_1}\ v \end{pmatrix}. \tag{4}$$

Let $\mathbf{A}=\begin{pmatrix} 1 & \beta & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & \beta & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \beta \\ r_1^* & r_2^*\beta & \alpha^* r_1^* & \alpha^* r_2^*\beta & r_1^* & r_2^*\beta \\ r_1^* & r_2^*\beta & r_1^* & r_2^*\beta & s^* r_1^* & s^* r_2^*\beta \\ r_1 & r_2\beta & \alpha r_1 & \alpha r_2\beta & r_1 & r_2\beta \end{pmatrix}$. This is distilled into:

$$\begin{pmatrix} 1 & \beta & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & \beta & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \beta \\ r_1^* & r_2^*\beta & \alpha^* r_1^* & \alpha^* r_2^*\beta & r_1^* & r_2^*\beta \\ r_1^* & r_2^*\beta & r_1^* & r_2^*\beta & s^* r_1^* & s^* r_2^*\beta \\ r_1 & r_2\beta & \alpha r_1 & \alpha r_2\beta & r_1 & r_2\beta \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ y_1 \\ y_2 \\ z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} \log_{g_1} c \\ \log_{g_1} d \\ \log_{g_1} h \\ \log_{g_1} v^* \\ \log_{g_1} e^*/M_b \\ \log_{g_1} v \end{pmatrix}. \tag{4}$$

The determinant of matrix $\mathbf{A}$ is given by

$$det(\mathbf{A}) = \beta^3(r_1^* - r_2^*)^2(r_1 - r_2)(\alpha^* - \alpha)(s^* - 1).$$

Then $det(\mathbf{A}) \neq 0$ except with a negligible probability $1/q$. Hence Eq. (4) is an injective function.

An injective function preserves its min-entropy. Then

$$\tilde{\mathrm{H}}_\infty((x_1, x_2, y_1, y_2, z_1, z_2)|c, d, h, C^*, M_b, \lambda\text{-leakage})$$

$$= \tilde{\mathrm{H}}_\infty((c, d, h, v^*, e^*/M_b, v)|c, d, h, C^*, M_b, \lambda\text{-leakage})$$

$$= \tilde{\mathrm{H}}_\infty(v|c, d, h, C^*, M_b, \lambda\text{-leakage}) \leq \log q - \lambda.$$

An injective function preserves its min-entropy. Then

$$\tilde{\mathrm{H}}_\infty((x_1, x_2, y_1, y_2, z_1, z_2)|c, d, h, C^*, M_b, \lambda\text{-leakage})$$

$$= \tilde{\mathrm{H}}_\infty((c, d, h, v^*, e^*/M_b, v)|c, d, h, C^*, M_b, \lambda\text{-leakage})$$

$$= \tilde{\mathrm{H}}_\infty(v|c, d, h, C^*, M_b, \lambda\text{-leakage}) \leq \log q - \lambda.$$

Therefore, the first invalid ciphertext C is accepted by $\mathcal{D}$ with probability at most $2^\lambda/q$.

An injective function preserves its min-entropy. Then

$$\tilde{\mathrm{H}}_\infty((x_1, x_2, y_1, y_2, z_1, z_2)|c, d, h, C^*, M_b, \lambda\text{-leakage})$$

$$= \tilde{\mathrm{H}}_\infty((c, d, h, v^*, e^*/M_b, v)|c, d, h, C^*, M_b, \lambda\text{-leakage})$$

$$= \tilde{\mathrm{H}}_\infty(v|c, d, h, C^*, M_b, \lambda\text{-leakage}) \leq \log q - \lambda.$$

Therefore, the first invalid ciphertext C is accepted by $\mathcal{D}$ with probability at most $2^\lambda/q$.
Similarly, the i-th invalid ciphertext is accepted by $\mathcal{D}$ with probability at most $2^\lambda/(q - i + 1) \leq 2^\lambda/(q - Q(\kappa))$, where $Q(\kappa)$ is the total number of decryption queries.

An injective function preserves its min-entropy. Then

$$\tilde{\mathrm{H}}_\infty((x_1, x_2, y_1, y_2, z_1, z_2)|c, d, h, C^*, M_b, \lambda\text{-leakage})$$

$$= \tilde{\mathrm{H}}_\infty((c, d, h, v^*, e^*/M_b, v)|c, d, h, C^*, M_b, \lambda\text{-leakage})$$

$$= \tilde{\mathrm{H}}_\infty(v|c, d, h, C^*, M_b, \lambda\text{-leakage}) \leq \log q - \lambda.$$

Therefore, the first invalid ciphertext C is accepted by $\mathcal{D}$ with probability at most $2^\lambda/q$.
Similarly, the i-th invalid ciphertext is accepted by $\mathcal{D}$ with probability at most $2^\lambda/(q - i + 1) \leq 2^\lambda/(q - Q(\kappa))$, where $Q(\kappa)$ is the total number of decryption queries.
By the union bound, we have

$$\Pr[F'] \leq \frac{2^\lambda Q(\kappa)}{q - Q(\kappa)}$$

and

$$|\Pr[S_4] - \Pr[S_3]| \leq \Pr[F'] \leq \frac{2^\lambda Q(\kappa)}{q - Q(\kappa)}.$$

Game 5: Same as Game 4 except for the generation of the challenge ciphertext $C^* = (u_1^*, u_2^*, e^*, s^*, v^*)$. The only change is that e^* is replaced with an element $\hat{e}$ chosen uniformly at random from G.

Game 5: Same as Game 4 except for the generation of the challenge ciphertext $C^* = (u_1^*, u_2^*, e^*, s^*, v^*)$. The only change is that e^* is replaced with an element $\hat{e}$ chosen uniformly at random from G.

By definition, $\Pr[S_5] = \frac{1}{2}$.

Game 5: Same as Game 4 except for the generation of the challenge ciphertext $C^* = (u_1^*, u_2^*, e^*, s^*, v^*)$. The only change is that e^* is replaced with an element $\hat{e}$ chosen uniformly at random from G.

By definition, $\Pr[S_5] = \frac{1}{2}$.

What left to establish in the rest is to show: $|\Pr[S_5] - \Pr[S_4]|$ is negligible.

$|\Pr[S_5] - \Pr[S_4]|$ Is Negligible

- Since all the invalid ciphertext queries are rejected, decryption oracle cannot help gain more information about the secret key.

$|\Pr[S_5] - \Pr[S_4]|$ Is Negligible

- Since all the invalid ciphertext queries are rejected, decryption oracle cannot help gain more information about the secret key.
- The only information related to secret key known by the adversary is still characterized by the public key elements (c, d, h), the λ-bit leakage, and (v^*, e^*) in C^*.

$|\Pr[S_5] - \Pr[S_4]|$ Is Negligible

- Since all the invalid ciphertext queries are rejected, decryption oracle cannot help gain more information about the secret key.
- The only information related to secret key known by the adversary is still characterized by the public key elements (c, d, h), the λ-bit leakage, and (v^*, e^*) in C^*.
- Next, we will show that e^*/M_b is in fact the output a $(2\log_2 q - \lambda, \delta)$ extractor with $\underbrace{(u_1^*)^{x_1+y_1}(u_2^*)^{x_2+y_2}}_{a}$ and $\underbrace{(u_1^*)^{z_1}(u_2^*)^{z_2}}_{b}$ as input.

$|\Pr[S_5] - \Pr[S_4]|$ Is Negligible

- Since all the invalid ciphertext queries are rejected, decryption oracle cannot help gain more information about the secret key.
- The only information related to secret key known by the adversary is still characterized by the public key elements (c, d, h), the λ-bit leakage, and (v^*, e^*) in C^*.
- Next, we will show that e^*/M_b is in fact the output a $(2\log_2 q - \lambda, \delta)$ extractor with $\underbrace{(u_1^*)^{x_1+y_1}(u_2^*)^{x_2+y_2}}_{a}$ and $\underbrace{(u_1^*)^{z_1}(u_2^*)^{z_2}}_{b}$ as input.
- Given the information c, d, h, v^* and the λ-bit leakage, we determine the average min-entropy

$$\tilde{\mathrm{H}}_\infty\left(\underbrace{(u_1^*)^{x_1+y_1}(u_2^*)^{x_2+y_2}}_{a}, \underbrace{(u_1^*)^{z_1}(u_2^*)^{z_2}}_{b} \mid c, d, h, v^*, \lambda\text{-leakage}\right).$$

Let us check the following equations in $x_1, x_2, y_1, y_2, z_1, z_2$.

$$\begin{array}{rcl}
\log_{g_1} c & = & x_1 + \beta x_2; \\
\log_{g_1} d & = & y_1 + \beta y_2; \\
\log_{g_1} h & = & z_1 + \beta z_2; \\
\log_{g_1} v^* & = & r_1^* x_1 + r_2^* \beta x_2 + \alpha^* r_1^* y_1 + \alpha^* r_2^* \beta y_2 + r_1^* z_1 + \\
\log_{g_1} \left(\underbrace{(u_1^*)^{x_1+y_1} (u_2^*)^{x_2+y_2}}_{a} \right) & = & r_1^* x_1 + r_2^* \beta x_2 + r_1^* y_1 + r_2^* \beta y_2; \\
\log_{g_1} \left(\underbrace{(u_1^*)^{z_1} (u_2^*)^{z_2}}_{b} \right) & = & r_1^* z_1 + r_2^* \beta z_2.
\end{array} \tag{5}$$

Let us check the following equations in $x_1, x_2, y_1, y_2, z_1, z_2$.

$$\begin{aligned}
\log_{g_1} c &= x_1 + \beta x_2; \\
\log_{g_1} d &= y_1 + \beta y_2; \\
\log_{g_1} h &= z_1 + \beta z_2; \\
\log_{g_1} v^* &= r_1^* x_1 + r_2^* \beta x_2 + \alpha^* r_1^* y_1 + \alpha^* r_2^* \beta y_2 + r_1^* z_1 + \\
\log_{g_1} \left(\underbrace{(u_1^*)^{x_1+y_1} (u_2^*)^{x_2+y_2}}_{a} \right) &= r_1^* x_1 + r_2^* \beta x_2 + r_1^* y_1 + r_2^* \beta y_2; \\
\log_{g_1} \left(\underbrace{(u_1^*)^{z_1} (u_2^*)^{z_2}}_{b} \right) &= r_1^* z_1 + r_2^* \beta z_2.
\end{aligned} \tag{5}$$

Equivalently,

$$\underbrace{\begin{pmatrix} 1 & \beta & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & \beta & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \beta \\ r_1^* & r_2^* \beta & \alpha^* r_1^* & \alpha^* r_2^* \beta & r_1^* & r_2^* \beta \\ r_1^* & r_2^* \beta & r_1^* & r_2^* \beta & 0 & 0 \\ 0 & 0 & 0 & 0 & r_1^* & r_2^* \beta \end{pmatrix}}_{B} \cdot \begin{pmatrix} x_1 \\ x_2 \\ y_1 \\ y_2 \\ z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} \log_{g_1} c \\ \log_{g_1} d \\ \log_{g_1} h \\ \log_{g_1} v^* \\ \log_{g_1} a \\ \log_{g_1} b \end{pmatrix}. \tag{6}$$

The determinant of the above matrix B is $-\beta^3(\alpha^* - 1)(r_1^* - r_2^*)^3 \neq 0$.

The determinant of the above matrix B is $-\beta^3(\alpha^* - 1)(r_1^* - r_2^*)^3 \neq 0$. The function $(x_1, x_2, y_1, y_2, z_1, z_2) \rightarrow (c, d, h, v^*, a, b)$ is injective, and applying an injective function to a function preserves its min-entropy.

The determinant of the above matrix B is $-\beta^3(\alpha^* - 1)(r_1^* - r_2^*)^3 \neq 0.$ The function $(x_1, x_2, y_1, y_2, z_1, z_2) \to (c, d, h, v^*, a, b)$ is injective, and applying an injective function to a function preserves its min-entropy. Hence,

$$\begin{aligned}
& \tilde{\mathrm{H}}_\infty\left(a, b \mid c, d, h, v^*, \lambda\text{-leakage}\right) && (7)\\
= \; & \tilde{\mathrm{H}}_\infty\left((u_1^*)^{x_1+y_1}(u_2^*)^{x_2+y_2}, (u_1^*)^{z_1}(u_2^*)^{z_2} | c, d, h, v^*, \lambda\text{-leakage}\right) && (8)\\
= \; & \tilde{\mathrm{H}}_\infty((x_1, x_2, y_1, y_2, z_1, z_2) | c, d, h, v^*, \lambda\text{-leakage}) && (9)\\
\geq \; & \tilde{\mathrm{H}}_\infty((x_1, x_2, y_1, y_2, z_1, z_2) | c, d, h, v^*) - \lambda \geq 2\log q - \lambda, . && (10)
\end{aligned}$$

- Applying the universal hash function $H_{s^*} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}$ (i.e, $H_{s^*}(a, b) = a \cdot b^{s^*}$ where $a = (u_1^*)^{x_1+y_1}(u_2^*)^{x_2+y_2}$ and $b = (u_1^*)^{z_1}(u_2^*)^{z_2}$) as a $(2\log_2 q - \lambda, \delta)$ extractor to the two variables $(u_1^*)^{x_1+y_1}(u_2^*)^{x_2+y_2}, (u_1^*)^{z_1}(u_2^*)^{z_2}$, we have

$$\begin{aligned} e^*/M_b &= H_{s^*}\left((u_1^*)^{x_1+y_1}(u_2^*)^{x_2+y_2}, (u_1^*)^{z_1}(u_2^*)^{z_2}\right) & (11)\\ &= (u_1^*)^{x_1+y_1}(u_2^*)^{x_2+y_2}\left((u_1^*)^{z_1}(u_2^*)^{z_2}\right)^{s^*} & (12)\\ &= (u_1^*)^{x_1+y_1+z_1 s^*} \cdot (u_2^*)^{x_2+y_2+z_2 s^*}. & (13) \end{aligned}$$

- Applying the universal hash function $H_{s^*} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}$ (i.e, $H_{s^*}(a, b) = a \cdot b^{s^*}$ where $a = (u_1^*)^{x_1+y_1}(u_2^*)^{x_2+y_2}$ and $b = (u_1^*)^{z_1}(u_2^*)^{z_2})$ as a $(2\log_2 q - \lambda, \delta)$ extractor to the two variables $(u_1^*)^{x_1+y_1}(u_2^*)^{x_2+y_2}, (u_1^*)^{z_1}(u_2^*)^{z_2}$, we have

$$\begin{aligned} e^*/M_b &= H_{s^*}\left((u_1^*)^{x_1+y_1}(u_2^*)^{x_2+y_2}, (u_1^*)^{z_1}(u_2^*)^{z_2}\right) && (11)\\ &= (u_1^*)^{x_1+y_1}(u_2^*)^{x_2+y_2}\left((u_1^*)^{z_1}(u_2^*)^{z_2}\right)^{s^*} && (12)\\ &= (u_1^*)^{x_1+y_1+z_1 s^*} \cdot (u_2^*)^{x_2+y_2+z_2 s^*}. && (13) \end{aligned}$$

- According to the generalized leftover lemma, $SD(e^*, U) \leq \frac{1}{2}\sqrt{q \cdot \frac{2^\lambda}{q^2}} = \frac{2^{\lambda/2-1}}{\sqrt{q}}$.

- Applying the universal hash function $H_{s^*} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}$ (i.e, $H_{s^*}(a, b) = a \cdot b^{s^*}$ where $a = (u_1^*)^{x_1+y_1}(u_2^*)^{x_2+y_2}$ and $b = (u_1^*)^{z_1}(u_2^*)^{z_2})$ as a $(2\log_2 q - \lambda, \delta)$ extractor to the two variables $(u_1^*)^{x_1+y_1}(u_2^*)^{x_2+y_2}, (u_1^*)^{z_1}(u_2^*)^{z_2}$, we have

$$\begin{aligned} e^*/M_b &= H_{s^*}\left((u_1^*)^{x_1+y_1}(u_2^*)^{x_2+y_2}, (u_1^*)^{z_1}(u_2^*)^{z_2}\right) && (11) \\ &= (u_1^*)^{x_1+y_1}(u_2^*)^{x_2+y_2}\left((u_1^*)^{z_1}(u_2^*)^{z_2}\right)^{s^*} && (12) \\ &= (u_1^*)^{x_1+y_1+z_1 s^*} \cdot (u_2^*)^{x_2+y_2+z_2 s^*}. && (13) \end{aligned}$$

- According to the generalized leftover lemma, $SD(e^*, U) \leq \frac{1}{2}\sqrt{q \cdot \frac{2^\lambda}{q^2}} = \frac{2^{\lambda/2-1}}{\sqrt{q}}$.
- Hence, $|\Pr[S_5] - \Pr[S_4]| \leq \delta = \frac{2^{\lambda/2-1}}{\sqrt{q}}$

Performance

Parameters of CS-PKE, KL-CS-PKE and Our Proposal

Let $\epsilon_1 = \mathbf{Adv}^{DDH}(\kappa), \epsilon_2 = \mathbf{Adv}^{\mathbf{TCR}}(\kappa)$, and $|M|$ denote the plaintext bit-length. Let λ be the amount of leakage bits, and $Q(\kappa)$ be the number of decryption queries.

Scheme	$\|M\|$	leakage	$\mathsf{Adv}^{\mathbf{IND\text{-}KL\text{-}CCA2}}_{\mathsf{PKE},\mathcal{A}}(1^\kappa)$
CS [CS03]	$\log q$	—	$\epsilon_1 + \epsilon_2 + \frac{Q(\kappa)}{q-Q(\kappa)}$
KL-CS [NS09] $(m + \lambda \leq \log q - \omega(\log \kappa))$	m	λ	$\epsilon_1 + \epsilon_2 + \frac{2^\lambda Q(\kappa)}{q-Q(\kappa)} + \frac{2^{(\lambda+m)/2-1}}{\sqrt{q}}$
Ours $(\lambda \leq \log q - \omega(\log \kappa))$	$\log q$	λ	$\epsilon_1 + \epsilon_2 + \frac{2^\lambda Q(\kappa)}{q-Q(\kappa)} + \frac{2^{\lambda/2-1}}{\sqrt{q}}$

Table: Parameters of CS-PKE, KL-CS-PKE and our proposals

Efficiency and Ciphertext Sizes of CS-PKE, KL-CS-PKE and Our Proposal

Scheme	KeyGen	Enc	Dec	Ciphertext Size
CS [CS03]	3 SE	3 Ex +1 S	2 SE	$4\mathbb{G}$
KL-CS [NS09]	3 SE	3 Ex + 1 SE+ 1 Ext	2 SE	$4\mathbb{G} + t$-bit
Ours	3 SE	2 Ex + 2 SE	2 SE	$4\mathbb{G} + \log q$-bit

Table: Efficiency and ciphertext sizes of CS-PKE, KL-CS-PKE and our proposal

Conclusion

- A response to Naor and Segev's calling for further refinement of key leakage resilient variant of Cramer-Shoup Cryptosystem in order to get rid of the dependency between plaintext length m and leakage parameter λ.

Conclusion

- A response to Naor and Segev's calling for further refinement of key leakage resilient variant of Cramer-Shoup Cryptosystem in order to get rid of the dependency between plaintext length m and leakage parameter λ.
- With some careful observations and a calculation guided design, our proposal follows a new line:

Conclusion

- A response to Naor and Segev's calling for further refinement of key leakage resilient variant of Cramer-Shoup Cryptosystem in order to get rid of the dependency between plaintext length m and leakage parameter λ.
- With some careful observations and a calculation guided design, our proposal follows a new line:
  - The whole secret key is involved in both ciphertext consistence checking and randomness distillation;

Conclusion

- A response to Naor and Segev's calling for further refinement of key leakage resilient variant of Cramer-Shoup Cryptosystem in order to get rid of the dependency between plaintext length m and leakage parameter λ.
- With some careful observations and a calculation guided design, our proposal follows a new line:
  - The whole secret key is involved in both ciphertext consistence checking and randomness distillation;
  - A special universal hashing based extractor is employed; alternatively, randomness extractor is only implicitly used with our proposal).

- Our scheme is IND-KL-CCA2 secure with a tighter reduction, $\lambda = \log q - \omega(\log \kappa)$ leakage resilient, and the plaintext space is the whole group that the scheme is based on and is independent of the leakage parameter. The performance of our proposal is comparable to the original Cramer-Shoup cryptosystem.

- Our scheme is IND-KL-CCA2 secure with a tighter reduction, $\lambda = \log q - \omega(\log \kappa)$ leakage resilient, and the plaintext space is the whole group that the scheme is based on and is independent of the leakage parameter. The performance of our proposal is comparable to the original Cramer-Shoup cryptosystem.
- To the best of our knowledge, the first leakage-resilient CS-type cryptosystem whose plaintext length is independent of the key leakage parameter, and is also the most efficient IND-CCA2 PKE scheme resilient to up to $\log q - \omega(\log \kappa)$ leakage.

Thanks