Fonctions à trappe avec perte d'information et Applications (d'après Peikert, Waters . . .)

(Lossy Trapdoor Functions and Applications - following Peikert, Waters, . . .)

Damien Vergnaud

École normale supérieure – C.N.R.S. – I.N.R.I.A.

8 avril 2011

Lossy Trapdoor Functions and Their Applications
C. Peikert, B. Waters - STOC 2008, 187-196
(Full version to appear in SIAM Journal on Computing)

- new general *primitive*
- realizations under various computational assumptions (DDH, LWE, QR, Paillier ...)
- used in more than 50 subsequent papers (Crypto'08, TCC'09, Eurocrypt'09, TCC'10, Eurocrypt'10, Crypto'10, ...)

  $\rightsquigarrow$ trapdoor functions, collision-resistant hash functions, oblivious transfer, encryption scheme with strong security properties, ...

Lossy Trapdoor Functions and Their Applications
C. Peikert, B. Waters - STOC 2008, 187-196
(Full version to appear in SIAM Journal on Computing)

- new general *primitive*
- realizations under various computational assumptions (DDH, LWE, QR, Paillier ...)
- used in more than 50 subsequent papers (Crypto'08, TCC'09, Eurocrypt'09, TCC'10, Eurocrypt'10, Crypto'10, ...)

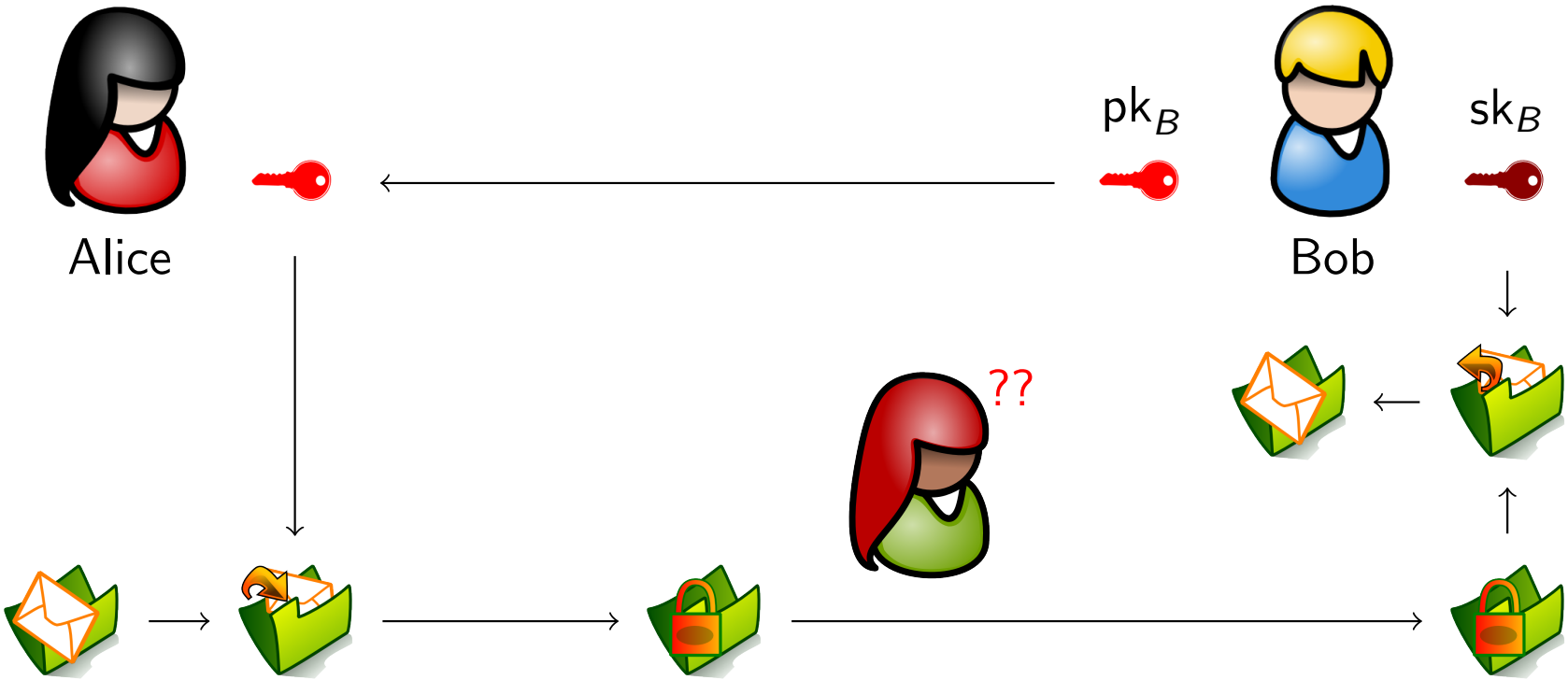
  $\rightsquigarrow$ trapdoor functions, collision-resistant hash functions, oblivious transfer, encryption scheme with strong security properties, ...

Contents

Public Key Cryptosystems

Asymmetric encryption: Bob owns two "keys"

- a public key — known by everybody (including Alice)
- a secret key — known by Bob only



Public-Key Encryption: Security Notions

Encryption is supposed to provide confidentiality of the data.

But what exactly does this mean?

Security goal	But . . .
Recovery of secret key is infeasible	

Public-Key Encryption: Security Notions

Encryption is supposed to provide confidentiality of the data.

But what exactly does this mean?

Security goal	But . . .
Recovery of secret key is infeasible	True if data is sent in the clear

Public-Key Encryption: Security Notions

Encryption is supposed to provide confidentiality of the data.

But what exactly does this mean?

Security goal	But . . .
Recovery of secret key is infeasible	True if data is sent in the clear
Obtaining plaintext from ciphertext is infeasible	

Public-Key Encryption: Security Notions

Encryption is supposed to provide confidentiality of the data.

But what exactly does this mean?

Security goal	But . . .
Recovery of secret key is infeasible	True if data is sent in the clear
Obtaining plaintext from ciphertext is infeasible	Might be able to obtain half the plaintext

Public-Key Encryption: Security Notions

Encryption is supposed to provide confidentiality of the data.

But what exactly does this mean?

Security goal	But . . .
Recovery of secret key is infeasible	True if data is sent in the clear
Obtaining plaintext from ciphertext is infeasible	Might be able to obtain half the plaintext

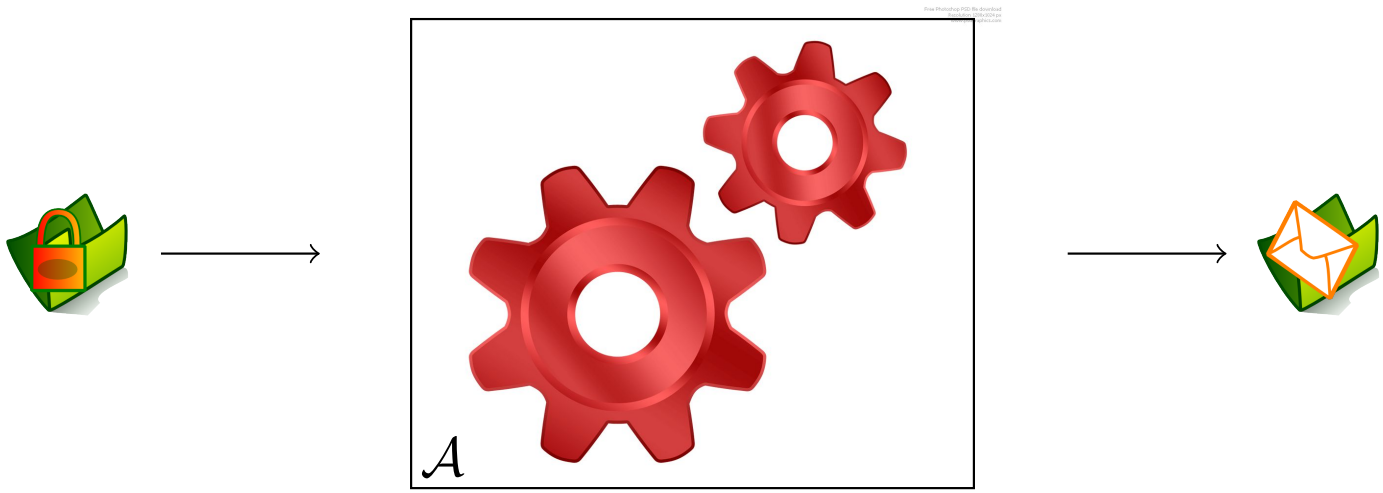
So what is a **secure** encryption scheme ?

Not an easy question to answer . . .

The Methodology of "Provable Security"

1. Define goal of adversary
2. Define security model
3. Define complexity assumptions
4. Provide a proof by reduction
5. Interpret proof

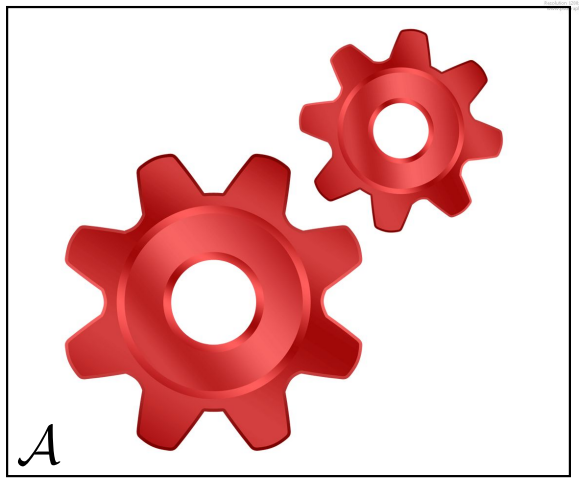
Proof by reduction



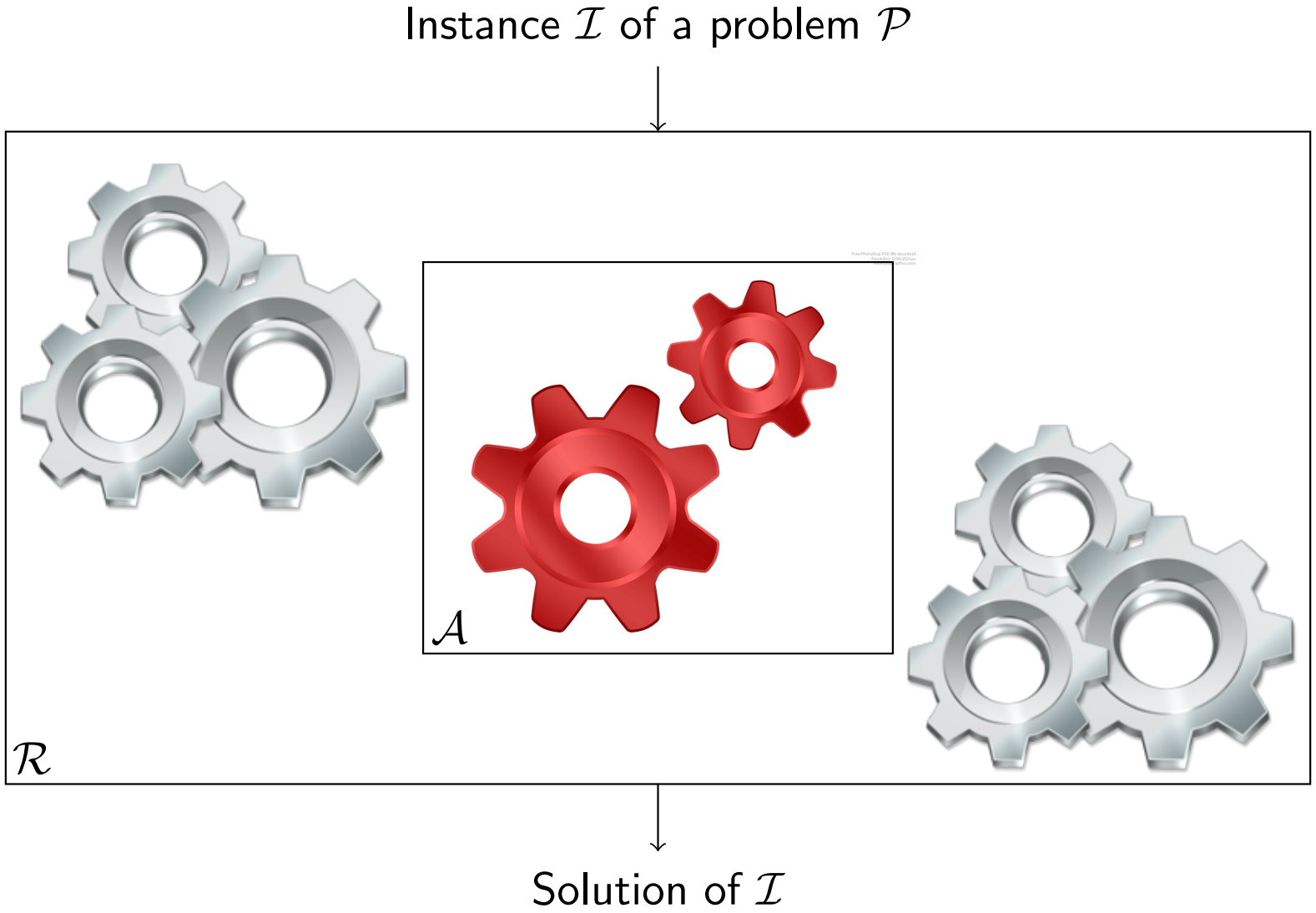
$\mathcal{A}$ adversary against e.g. one-wayness

Proof by reduction

Instance $\mathcal{I}$ of a problem $\mathcal{P}$

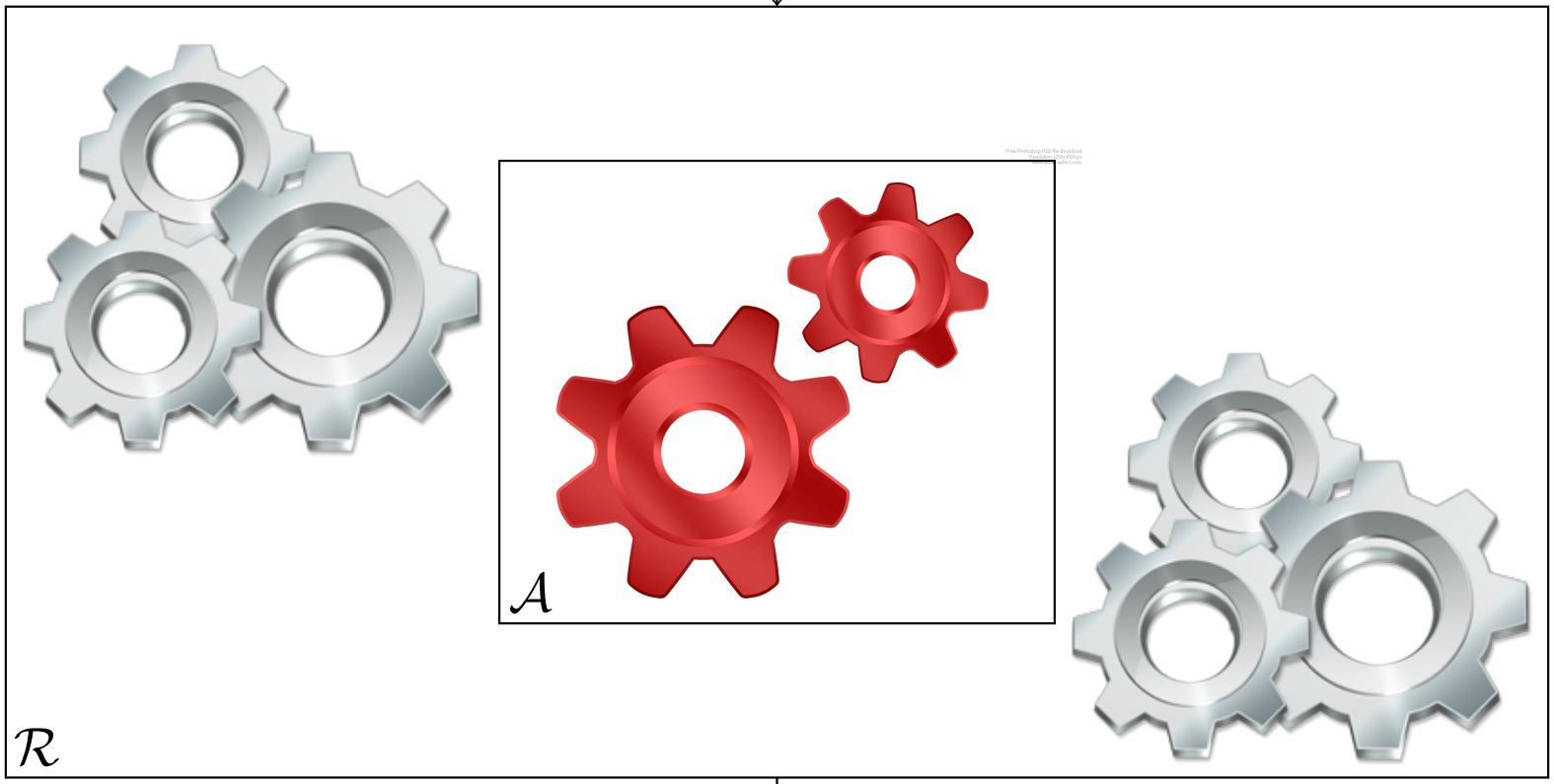


Proof by reduction



Proof by reduction

Instance $\mathcal{I}$ of a problem $\mathcal{P}$



Solution of $\mathcal{I}$

$\mathcal{P}$ intractable $\rightarrow$ contradiction

Public-Key Encryption: Definition

(KeyGen, Encrypt, Decrypt) where:

- KeyGen: $\kappa \longrightarrow (\mathsf{sk}, \mathsf{pk})$
- Encrypt: $\mathsf{pk}, m, \varpi \longrightarrow c$,
- Decrypt: $\mathsf{sk}, c \longrightarrow m$ or $\perp$.

$$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(\kappa) \Rightarrow \mathsf{Decrypt}_{\mathsf{sk}}\left(\mathsf{Encrypt}_{\mathsf{pk}}(m, \varpi)\right) = m$$

Public-Key Encryption: Definition

(KeyGen, Encrypt, Decrypt) where:

- KeyGen: $\kappa \longrightarrow (\mathsf{sk}, \mathsf{pk})$
- Encrypt: $\mathsf{pk}, m, \varpi \longrightarrow c$,
- Decrypt: $\mathsf{sk}, c \longrightarrow m$ or $\perp$.

$$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(\kappa) \Rightarrow \mathsf{Decrypt}_{\mathsf{sk}}\left(\mathsf{Encrypt}_{\mathsf{pk}}(m, \varpi)\right) = m$$

Public-Key Encryption: Definition

(KeyGen, Encrypt, Decrypt) where:

- KeyGen: $\kappa \longrightarrow (\mathsf{sk}, \mathsf{pk})$
- Encrypt: $\mathsf{pk}, m, \varpi \longrightarrow c$,
- Decrypt: $\mathsf{sk}, c \longrightarrow m$ or $\perp$.

$$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(\kappa) \Rightarrow \mathsf{Decrypt}_{\mathsf{sk}}\left(\mathsf{Encrypt}_{\mathsf{pk}}(m, \varpi)\right) = m$$

History of Security Goals

- infeasible to compute sk from pk
  $\rightsquigarrow$ Unbreakability. (Implicitely appeared with public-key crypto)

- infeasible to invert Encrypt(pk, $\cdot$)
  $\rightsquigarrow$ One-wayness. (Diffie and Hellman, late 70's)

- infeasible to recover a **single bit of information** about a plaintext given its encryption under pk
  $\rightsquigarrow$ Indistinguishability of encryptions **(IND)**. (Goldwasser and Micali, 1984).

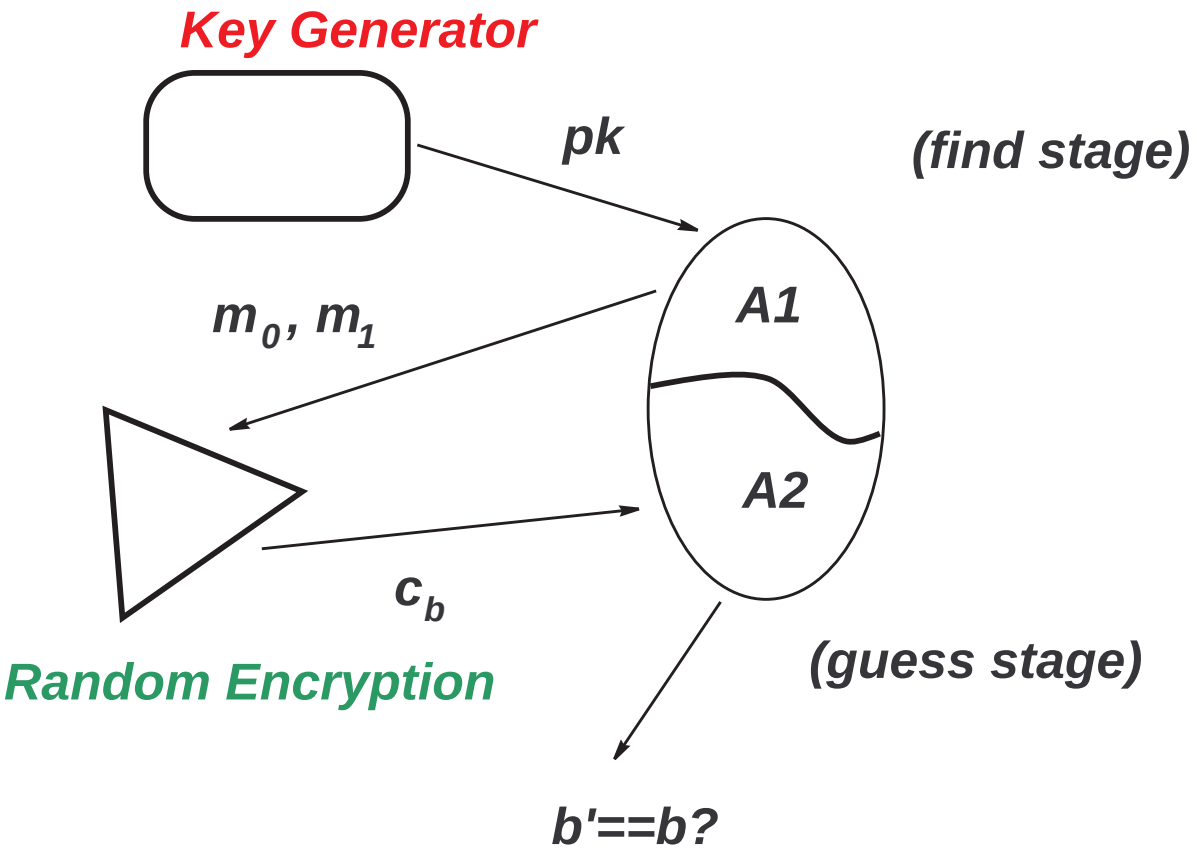
History of Security Goals

- infeasible to compute sk from pk
  $\rightsquigarrow$ Unbreakability. (Implicitely appeared with public-key crypto)

- infeasible to invert Encrypt(pk, $\cdot$)
  $\rightsquigarrow$ One-wayness. (Diffie and Hellman, late 70's)

- infeasible to recover a **single bit of information** about a plaintext given its encryption under pk
  $\rightsquigarrow$ Indistinguishability of encryptions **(IND)**. (Goldwasser and Micali, 1984).

Indistinguishability of encryptions (IND)



History of Adversarial Models

Several types of computational resources an adversary has access to have been considered:

- **chosen-plaintext attacks** (**CPA**), unavoidable scenario.
- **non-adaptive chosen-ciphertext attacks** (**CCA1**),
  the adversary gets access to a decryption oracle before being given the challenge ciphertext. (Naor and Yung, 1990).
- **adaptive chosen-ciphertext attacks** (**CCA2**),
  the adversary queries the decryption oracle before and *after* being challenged (Rackoff and Simon, 1991).

History of Adversarial Models

Several types of computational resources an adversary has access to have been considered:

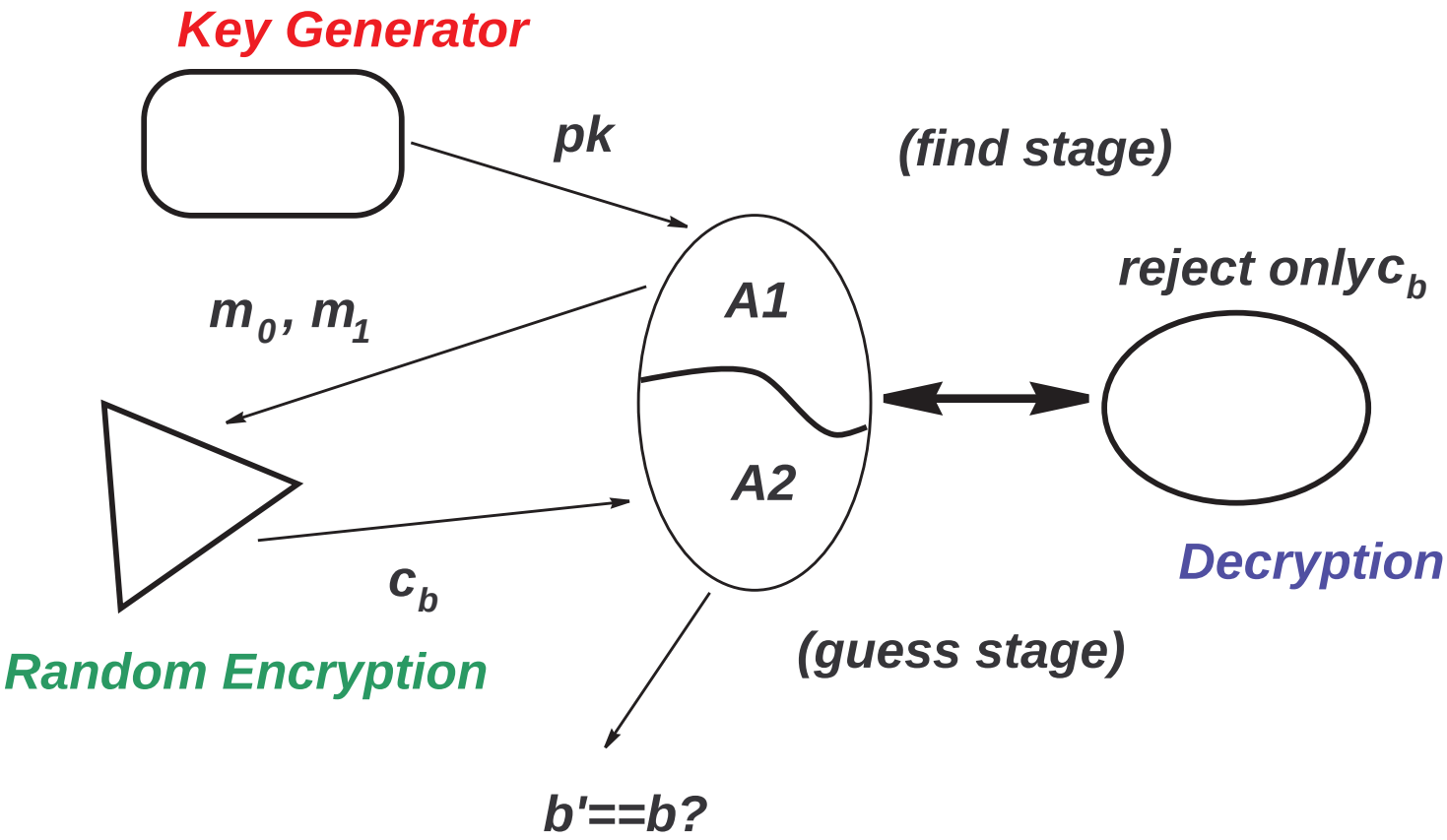
- **chosen-plaintext attacks** (**CPA**), unavoidable scenario.
- **non-adaptive chosen-ciphertext attacks** (**CCA1**),
  the adversary gets access to a decryption oracle before being given the challenge ciphertext. (Naor and Yung, 1990).
- **adaptive chosen-ciphertext attacks** (**CCA2**),
  the adversary queries the decryption oracle before and *after* being challenged (Rackoff and Simon, 1991).

History of Adversarial Models

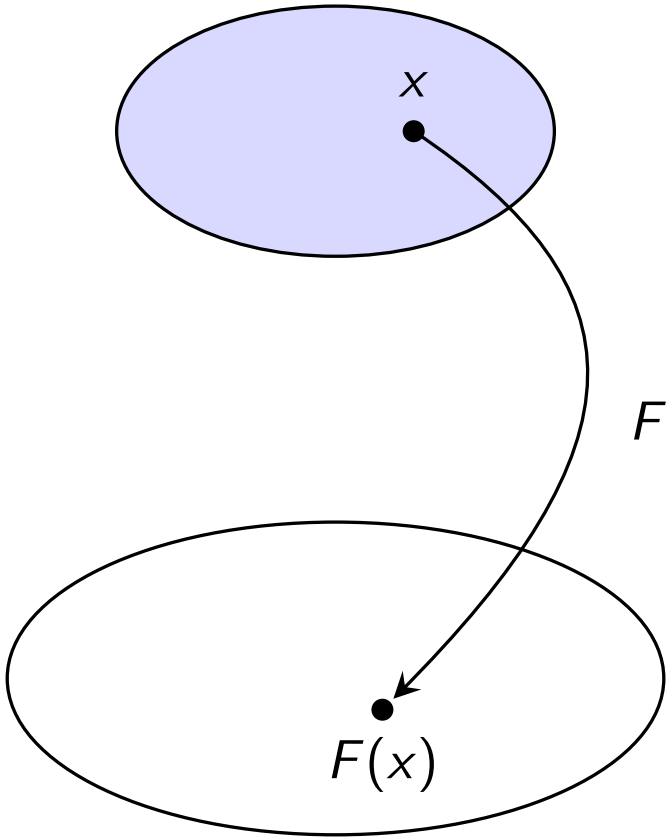
Several types of computational resources an adversary has access to have been considered:

- **chosen-plaintext attacks** (**CPA**), unavoidable scenario.
- **non-adaptive chosen-ciphertext attacks** (**CCA1**),
  the adversary gets access to a decryption oracle before being given the challenge ciphertext. (Naor and Yung, 1990).
- **adaptive chosen-ciphertext attacks** (**CCA2**),
  the adversary queries the decryption oracle before and *after* being challenged (Rackoff and Simon, 1991).

IND-CCA: Playing the Game

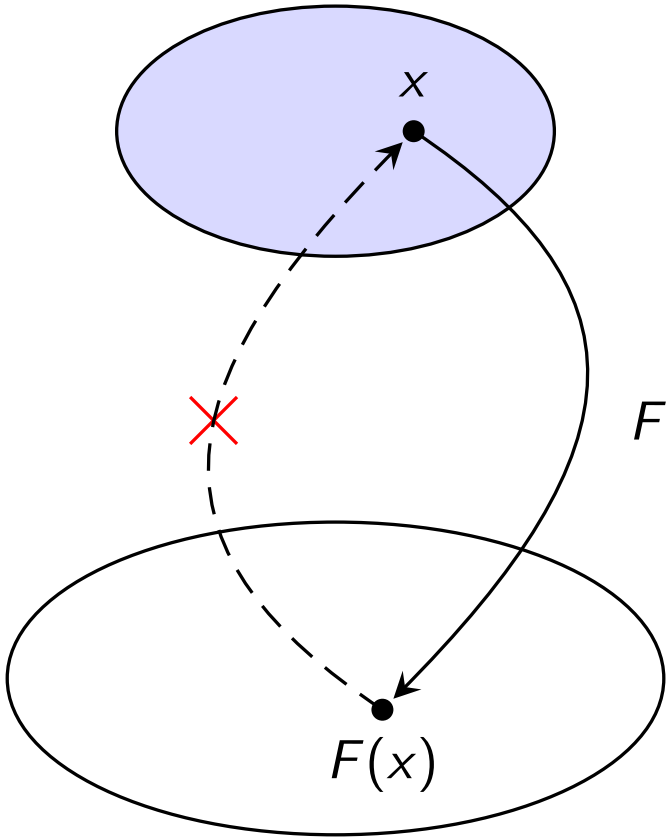


Trapdoor One-Way Functions



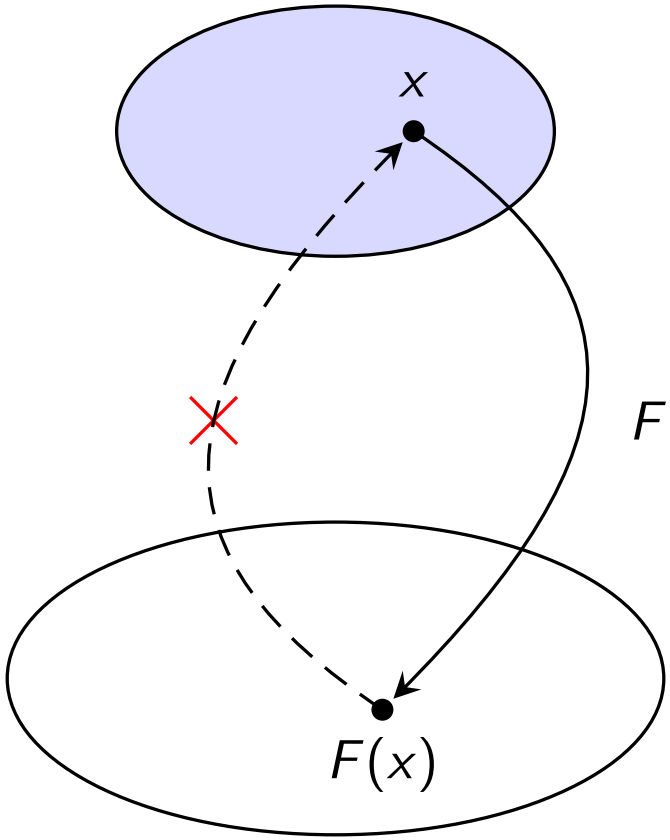
One-Way Function

Trapdoor One-Way Functions

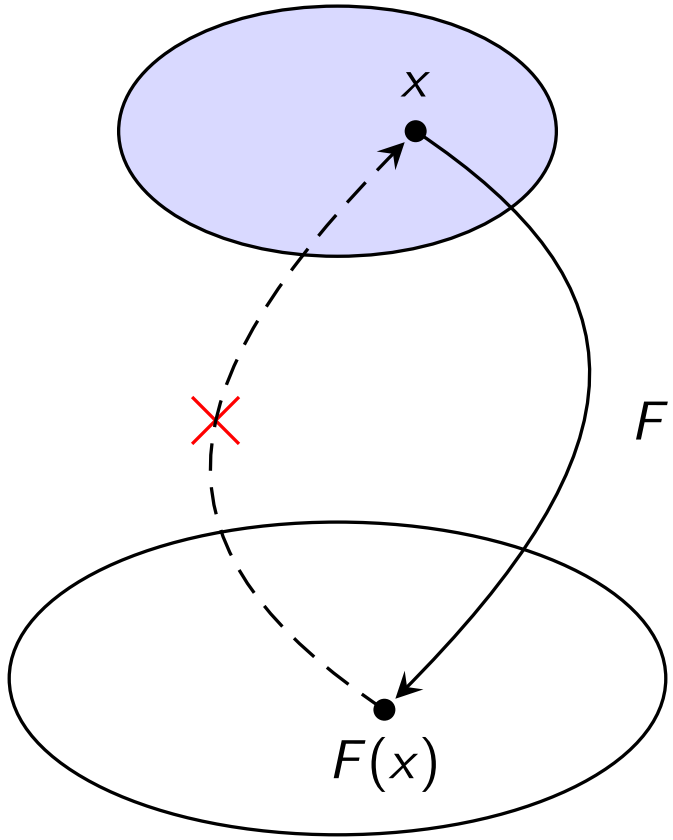


One-Way Function

Trapdoor One-Way Functions

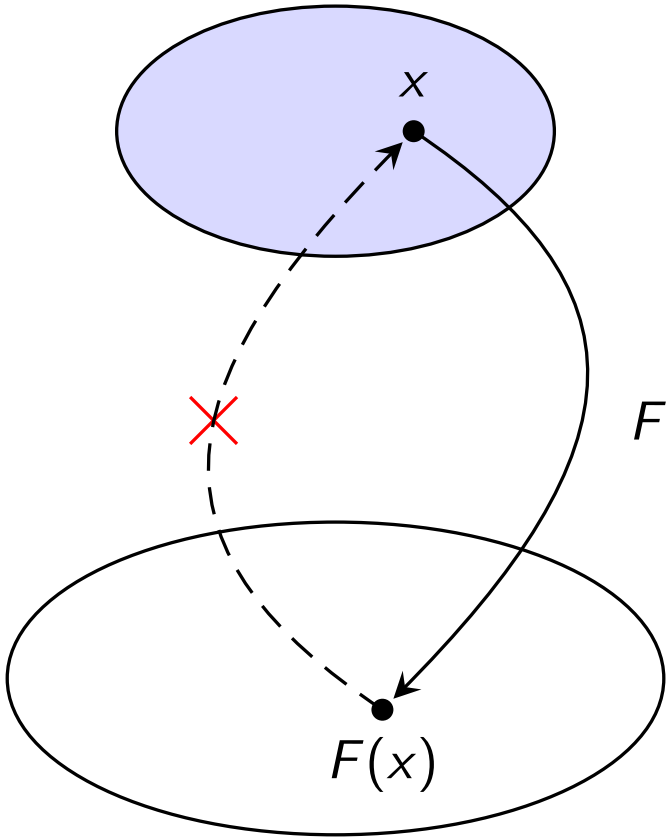


One-Way Function

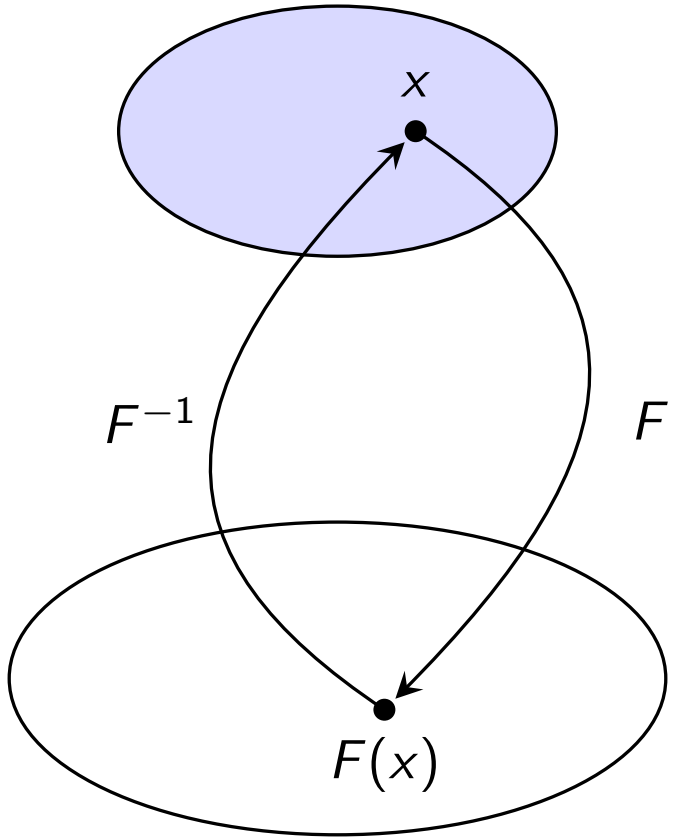


Trapdoor One-Way Function

Trapdoor One-Way Functions



One-Way Function



Trapdoor One-Way Function

Extracting Randomness

F = (trapdoor) one-way function

- **Hard-core bit** $B : \mathrm{Dom}_F \longrightarrow \{0,1\}$ of F:

$$x \longmapsto B(x) \qquad \text{easy}$$
$$F(x) \longmapsto B(x) \qquad \text{difficult !}$$

  - F = RSA, LSB $\rightsquigarrow$ hard-core bit
  - $G : (x, r) \longmapsto (F(x), r)$ with $|x| = |r|$
    $\rightsquigarrow B = (x, r) \longmapsto \langle F(x), r \rangle$ hard-core bit for G. (Goldreich-Levin 1989)

- **Randomness extractor** (Universal pairwise independent hash function)

DO NOT READ !

X, Y random variables in $\{0,1\}^n$ with $H_\infty(X|Y) \geq k$
$\mathcal{H} : \{0,1\}^n \longrightarrow \{0,1\}^\ell$ s.t.

$$\Delta\left[(Y, \mathcal{H}(X)), (Y, U_\ell)\right] \leq \epsilon \text{ for } \ell \leq k - 2\log(1/\epsilon)$$

Extracting Randomness

F = (trapdoor) one-way function

- **Hard-core bit** $B : \mathrm{Dom}_F \longrightarrow \{0,1\}$ of F:

$$x \longmapsto B(x) \qquad \text{easy}$$
$$F(x) \longmapsto B(x) \qquad \text{difficult !}$$

  - F = RSA, LSB $\rightsquigarrow$ hard-core bit
  - $G : (x, r) \longmapsto (F(x), r)$ with $|x| = |r|$
    $\rightsquigarrow$ $B = (x, r) \longmapsto \langle F(x), r \rangle$ hard-core bit for G. (Goldreich-Levin 1989)

- **Randomness extractor** (Universal pairwise independent hash function)

DO NOT READ !

X, Y random variables in $\{0,1\}^n$ with $H_\infty(X|Y) \geq k$
$\mathcal{H} : \{0,1\}^n \longrightarrow \{0,1\}^\ell$ s.t.

$$\Delta\left[(Y, \mathcal{H}(X)), (Y, U_\ell)\right] \leq \epsilon \text{ for } \ell \leq k - 2\log(1/\epsilon)$$

Extracting Randomness

F = (trapdoor) one-way function

- **Hard-core bit** $B : \mathrm{Dom}_F \longrightarrow \{0,1\}$ of F:

$$x \longmapsto B(x) \qquad \text{easy}$$
$$F(x) \longmapsto B(x) \qquad \text{difficult !}$$

  - F = RSA, LSB $\rightsquigarrow$ hard-core bit
  - $G : (x, r) \longmapsto (F(x), r)$ with $|x| = |r|$
    $\rightsquigarrow$ $B = (x, r) \longmapsto \langle F(x), r \rangle$ hard-core bit for G. (Goldreich-Levin 1989)

- **Randomness extractor** (Universal pairwise independent hash function)

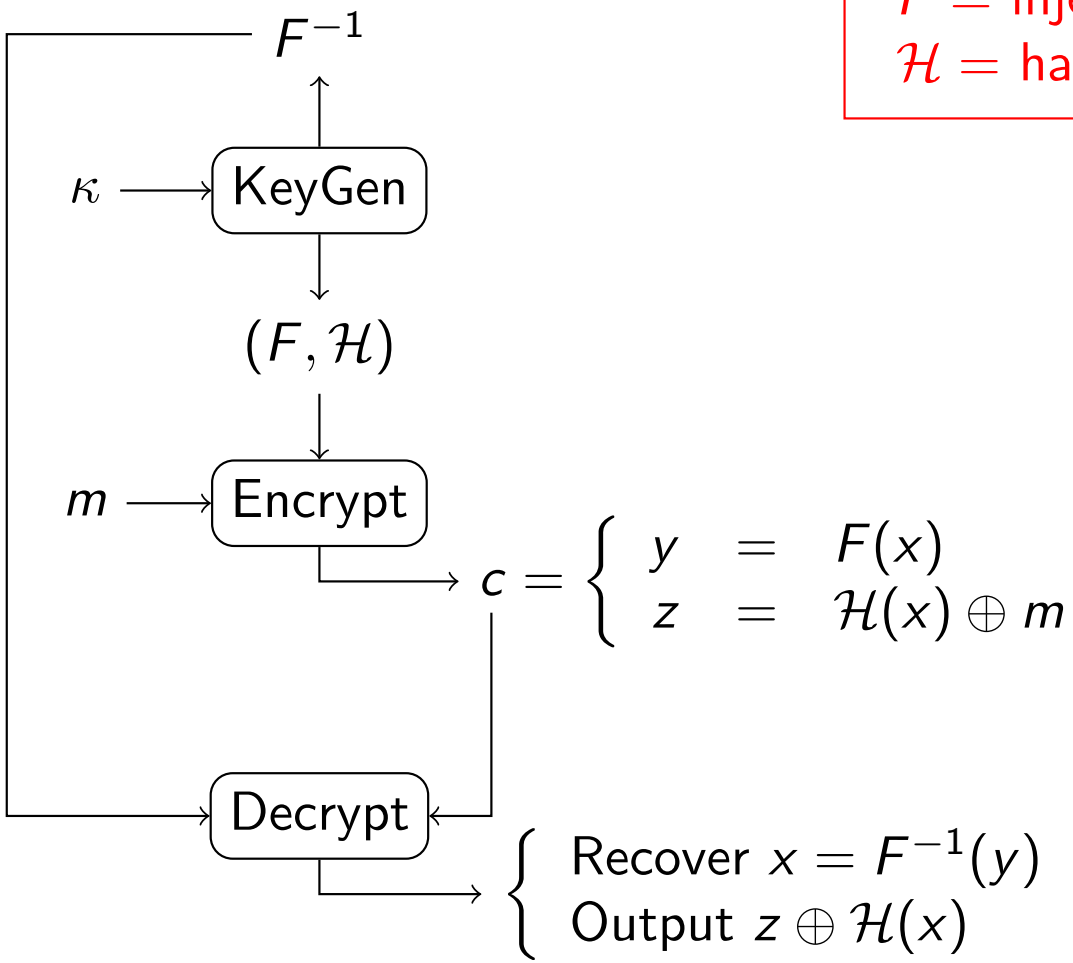
DO NOT READ !

X, Y random variables in $\{0,1\}^n$ with $H_\infty(X|Y) \geq k$
$\mathcal{H} : \{0,1\}^n \longrightarrow \{0,1\}^\ell$ s.t.

$$\Delta\left[(Y, \mathcal{H}(X)), (Y, U_\ell)\right] \leq \epsilon \text{ for } \ell \leq k - 2\log(1/\epsilon)$$

IND-CPA Encryption from TDF

F = injective trapdoor function
$\mathcal{H}$ = hard-core bit for F



How to achieve IND-CCA Security ?

Verify ciphertext is "well-formed"

- **Basic idea:**
  - Add redundancy to the ciphertext
  - Decryption is only performed for ciphertexts with the right redundancy

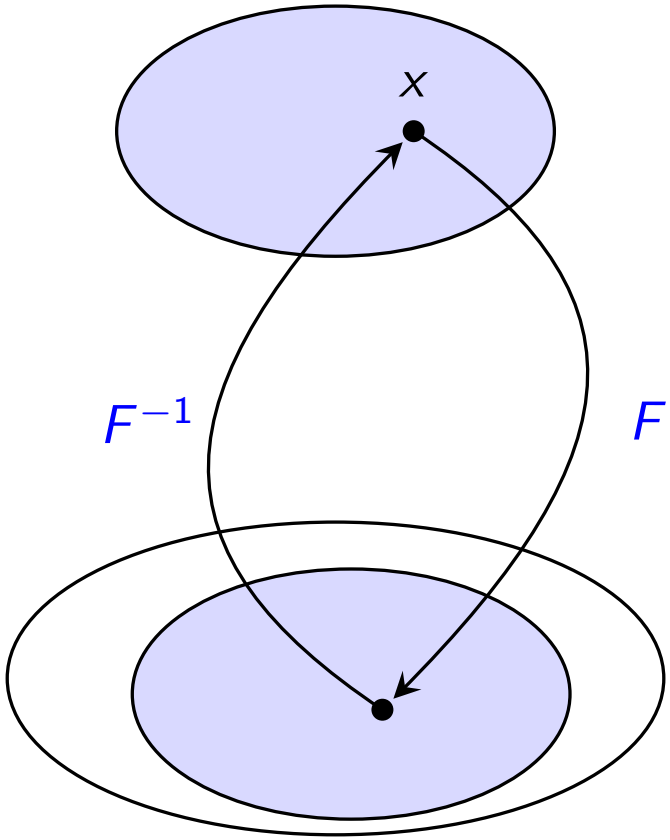
  $\rightsquigarrow \mathcal{A}$ effectively knows the plaintext anyway, the decryption oracle is useless!

- Usually via *zero-knowledge* proof ($\rightsquigarrow$ inefficient)

- **New idea:** recover randomness

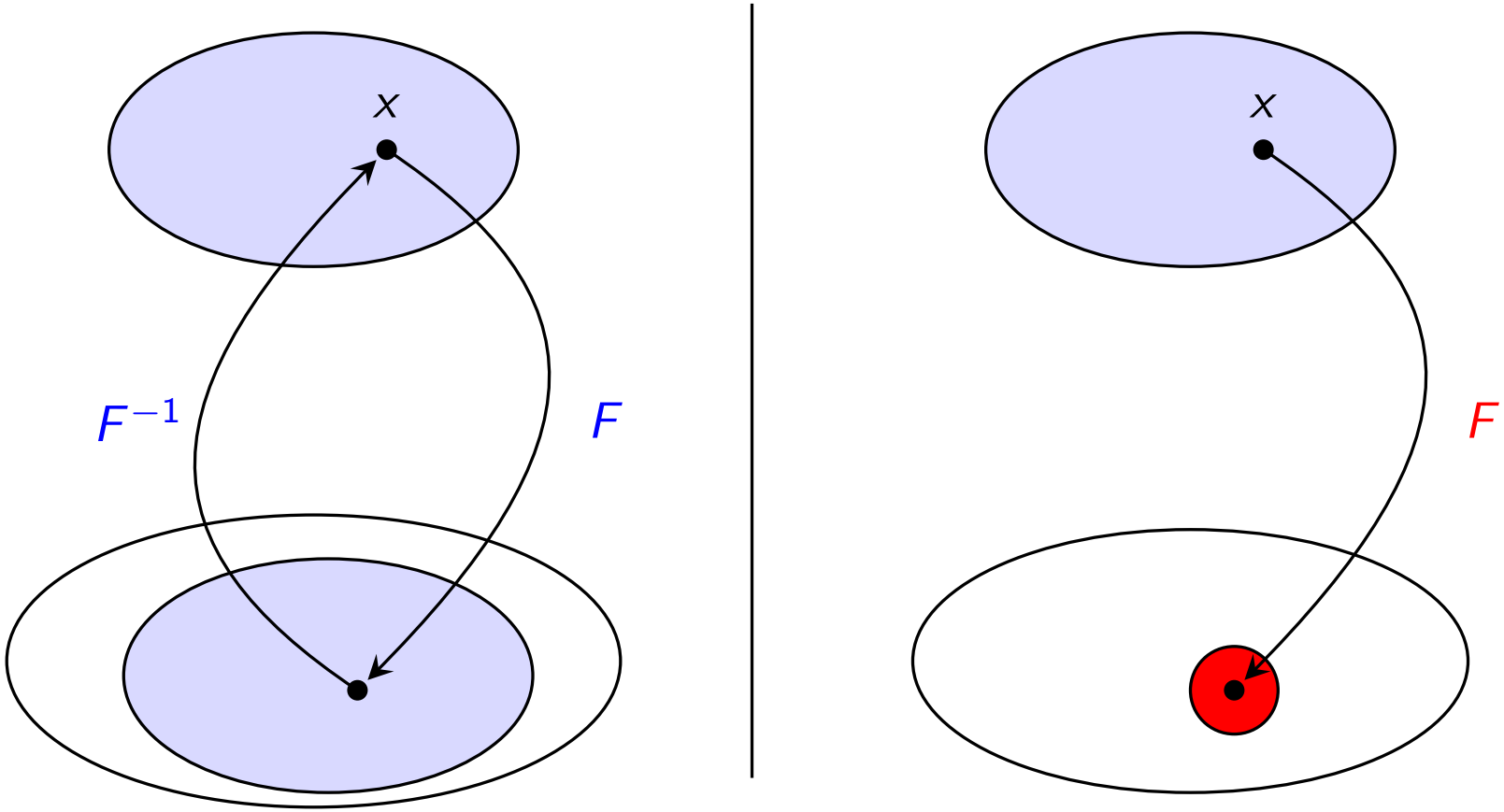
Contents

Lossy Trapdoor One-Way Functions



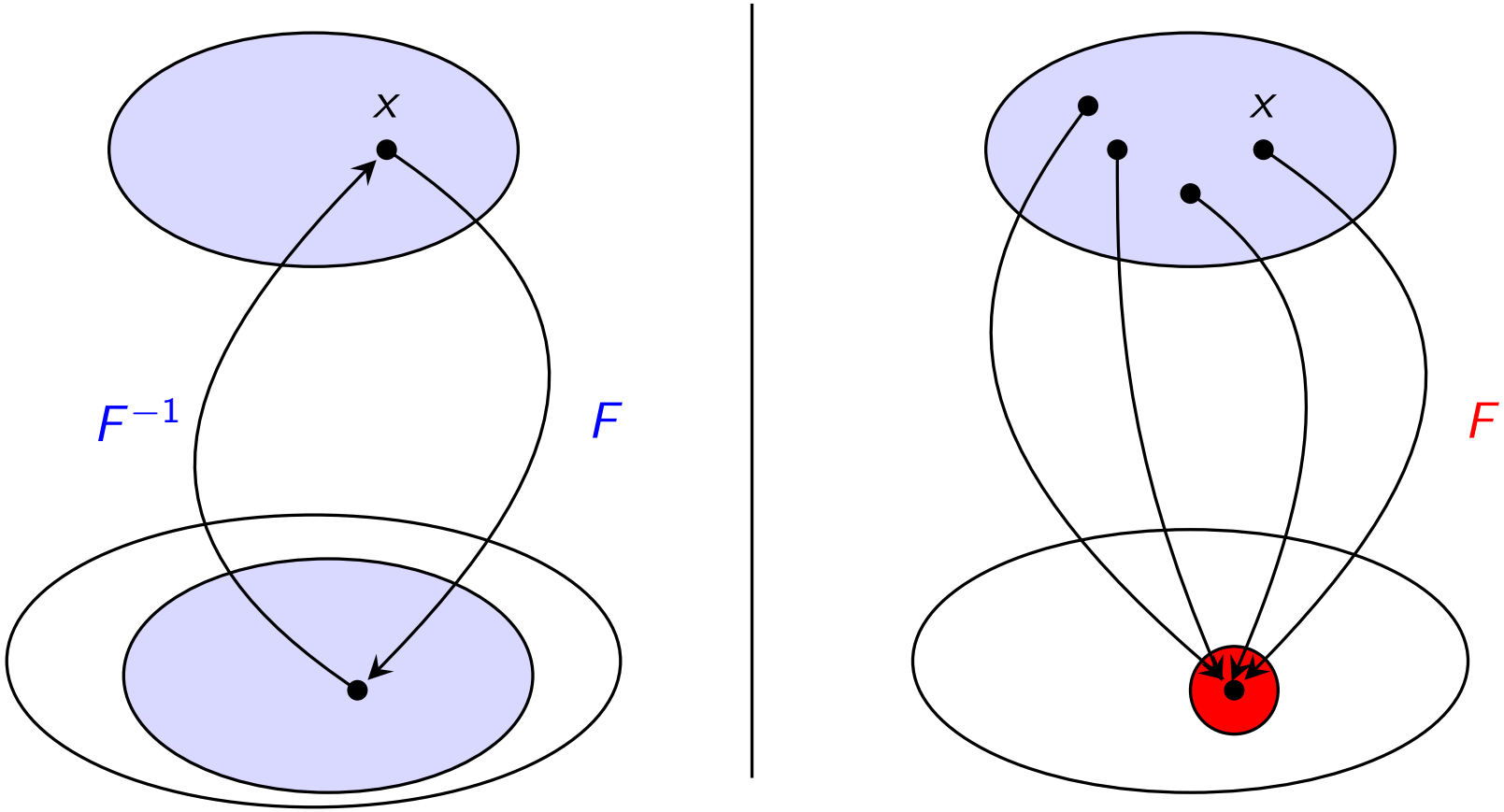
Lossy Trapdoor One-Way Function

Lossy Trapdoor One-Way Functions



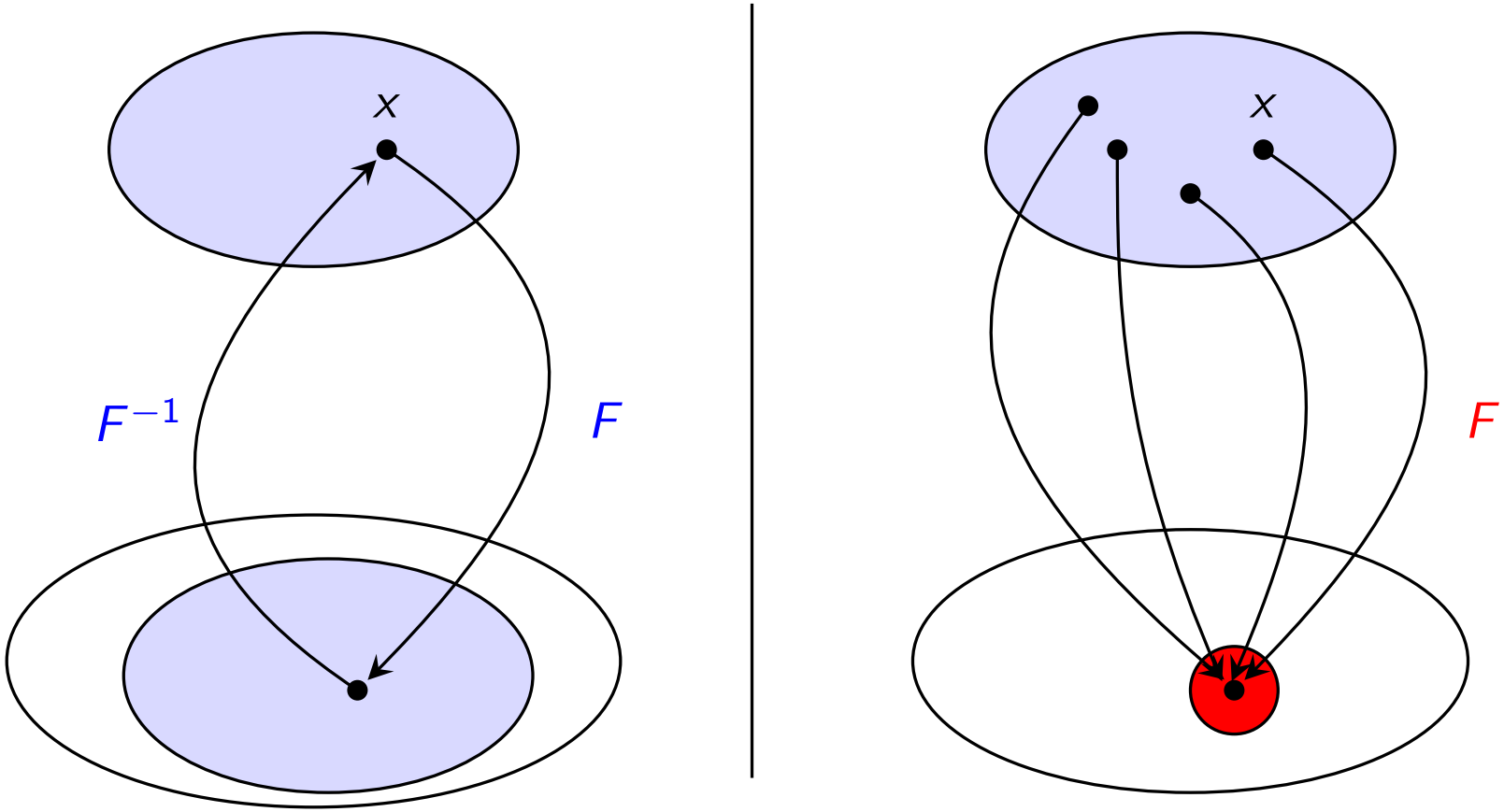
Lossy Trapdoor One-Way Function

Lossy Trapdoor One-Way Functions



Lossy Trapdoor One-Way Function

Lossy Trapdoor One-Way Functions



$$F \simeq_c F$$

Lossy Trapdoor One-Way Function

Lossy TDF: Definition

A Lossy TDF with lossiness k is a tuple $(S_{\text{inj}}, F_{\text{inj}}, F_{\text{inj}}^{-1}, S_{\text{loss}}, F_{\text{loss}})$ where:

- **Easy to sample an injective trapdoor function:**
  - $S_{\text{inj}}(\kappa) \longmapsto (n, s, t)$
  - $F_{\text{inj}}(s) : \{0,1\}^n \longrightarrow \{0,1\}^n, x \longmapsto F_{\text{inj}}(s, x)$
  - $F_{\text{inj}}^{-1}(t) : \{0,1\}^n \longrightarrow \{0,1\}^n, y \longmapsto F_{\text{inj}}^{-1}(t, y)$

  with
  $$F_{\text{inj}}^{-1}(t, F_{\text{inj}}(s, x)) = x$$

- **Easy to sample a lossy trapdoor function:**
  - $S_{\text{loss}}(\kappa) \longmapsto (n, s)$
  - $F_{\text{loss}}(s) : \{0,1\}^n \longrightarrow \{0,1\}^n, x \longmapsto F_{\text{loss}}(s, x)$
  - $\#F_{\text{loss}}(s)(\{0,1\}^n) \leq 2^{n-k}$

- **Hard to distinguish injective from lossy:**
  $$s \leftarrow S_{\text{inj}}(\kappa) \simeq_c s \leftarrow S_{\text{loss}}(\kappa)$$

Lossy TDF: Definition

A Lossy TDF with lossiness k is a tuple $(S_{\text{inj}}, F_{\text{inj}}, F_{\text{inj}}^{-1}, S_{\text{loss}}, F_{\text{loss}})$ where:

- **Easy to sample an injective trapdoor function:**
  - $S_{\text{inj}}(\kappa) \longmapsto (n, s, t)$
  - $F_{\text{inj}}(s) : \{0,1\}^n \longrightarrow \{0,1\}^n, x \longmapsto F_{\text{inj}}(s, x)$
  - $F_{\text{inj}}^{-1}(t) : \{0,1\}^n \longrightarrow \{0,1\}^n, y \longmapsto F_{\text{inj}}^{-1}(t, y)$

  with
  $$F_{\text{inj}}^{-1}(t, F_{\text{inj}}(s, x)) = x$$

- **Easy to sample a lossy trapdoor function:**
  - $S_{\text{loss}}(\kappa) \longmapsto (n, s)$
  - $F_{\text{loss}}(s) : \{0,1\}^n \longrightarrow \{0,1\}^n, x \longmapsto F_{\text{loss}}(s, x)$
  - $\#F_{\text{loss}}(s)(\{0,1\}^n) \leq 2^{n-k}$

- **Hard to distinguish injective from lossy:**

$$s \leftarrow S_{\text{inj}}(\kappa) \simeq_c s \leftarrow S_{\text{loss}}(\kappa)$$

Lossy TDF: Definition

A Lossy TDF with lossiness k is a tuple $(S_{\text{inj}}, F_{\text{inj}}, F_{\text{inj}}^{-1}, S_{\text{loss}}, F_{\text{loss}})$ where:

- **Easy to sample an injective trapdoor function:**
  - $S_{\text{inj}}(\kappa) \longmapsto (n, s, t)$
  - $F_{\text{inj}}(s) : \{0,1\}^n \longrightarrow \{0,1\}^n, x \longmapsto F_{\text{inj}}(s, x)$
  - $F_{\text{inj}}^{-1}(t) : \{0,1\}^n \longrightarrow \{0,1\}^n, y \longmapsto F_{\text{inj}}^{-1}(t, y)$

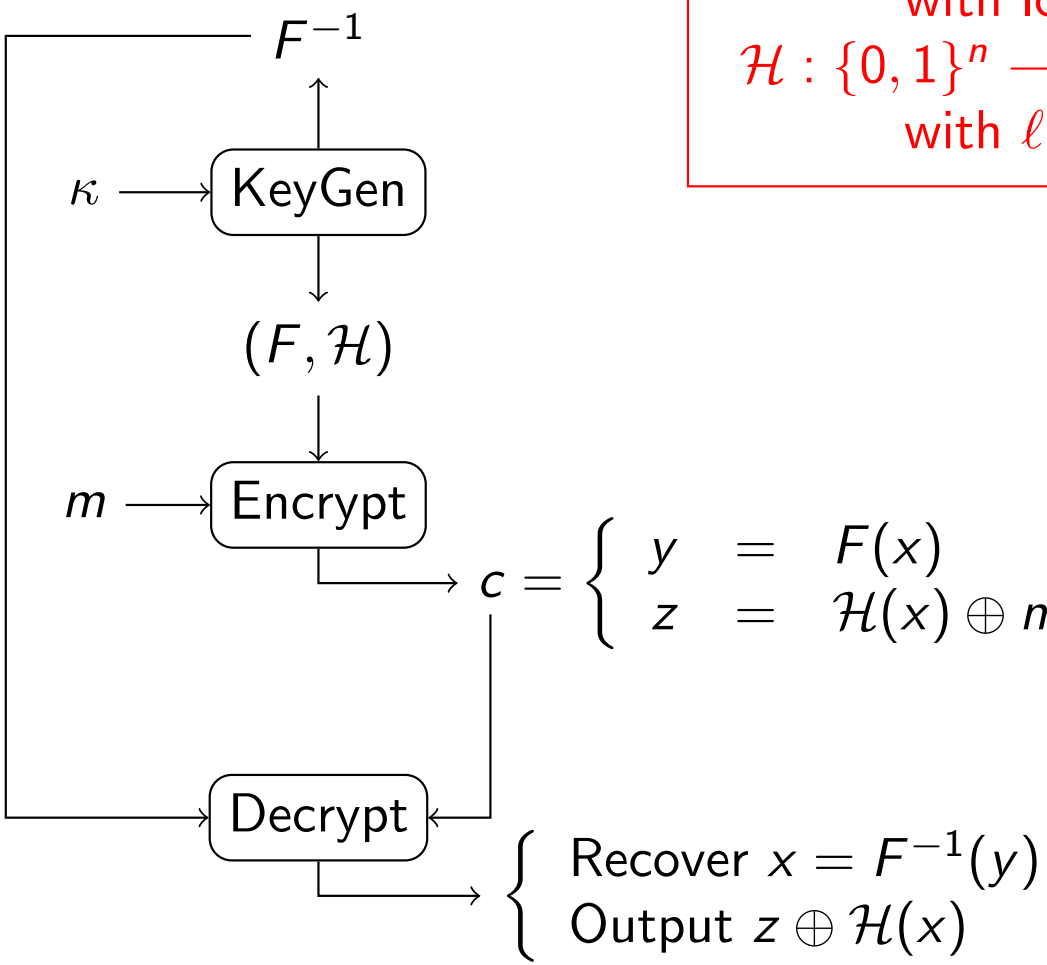
  with
  $$F_{\text{inj}}^{-1}(t, F_{\text{inj}}(s, x)) = x$$

- **Easy to sample a lossy trapdoor function:**
  - $S_{\text{loss}}(\kappa) \longmapsto (n, s)$
  - $F_{\text{loss}}(s) : \{0,1\}^n \longrightarrow \{0,1\}^n, x \longmapsto F_{\text{loss}}(s, x)$
  - $\#F_{\text{loss}}(s)(\{0,1\}^n) \leq 2^{n-k}$

- **Hard to distinguish injective from lossy:**
  $$s \leftarrow S_{\text{inj}}(\kappa) \simeq_c s \leftarrow S_{\text{loss}}(\kappa)$$

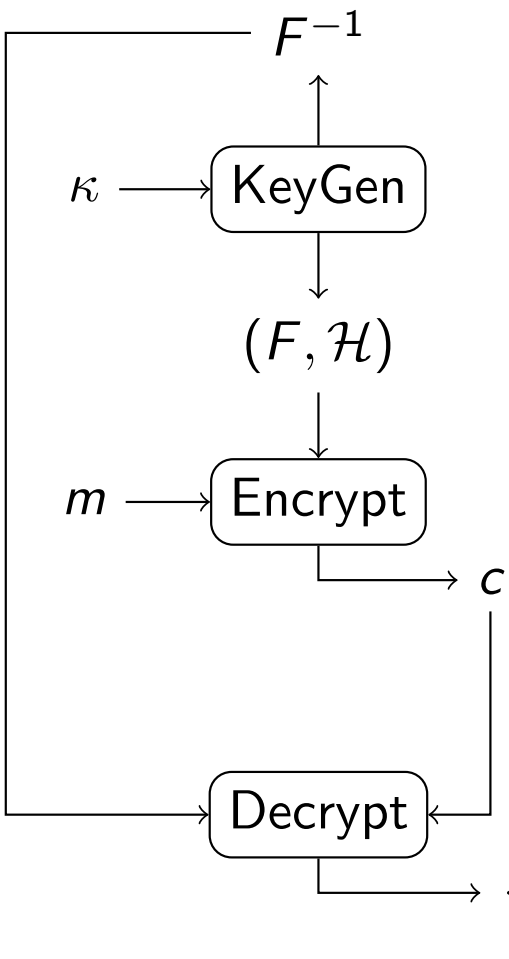
IND-CPA Encryption from Lossy TDF

$F : \{0,1\}^n \longrightarrow \{0,1\}^n$ lossy TDF
with lossiness k
$\mathcal{H} : \{0,1\}^n \longrightarrow \{0,1\}^\ell$ randomness extractor
with $\ell < k$



IND-CPA Encryption from Lossy TDF

$F : \{0,1\}^n \longrightarrow \{0,1\}^n$ lossy TDF
with lossiness k
$\mathcal{H} : \{0,1\}^n \longrightarrow \{0,1\}^\ell$ randomness extractor
with $\ell < k$



$$c = \begin{cases} y & = & F(x) \\ z & = & \mathcal{H}(x) \oplus m \end{cases}$$

$$\begin{cases} \text{Recover } x = F^{-1}(y) \\ \text{Output } z \oplus \mathcal{H}(x) \end{cases}$$

$\rightsquigarrow$ Oblivious transfer

All-But-One TDF

- $G(b, x)$ has an extra parameter: branch $b \in \{0,1\}^n$
- Generate (G, G^{-1}) with hidden lossy branch ℓ
- Lossy TDF $\Longleftrightarrow$ All-But-One TDF

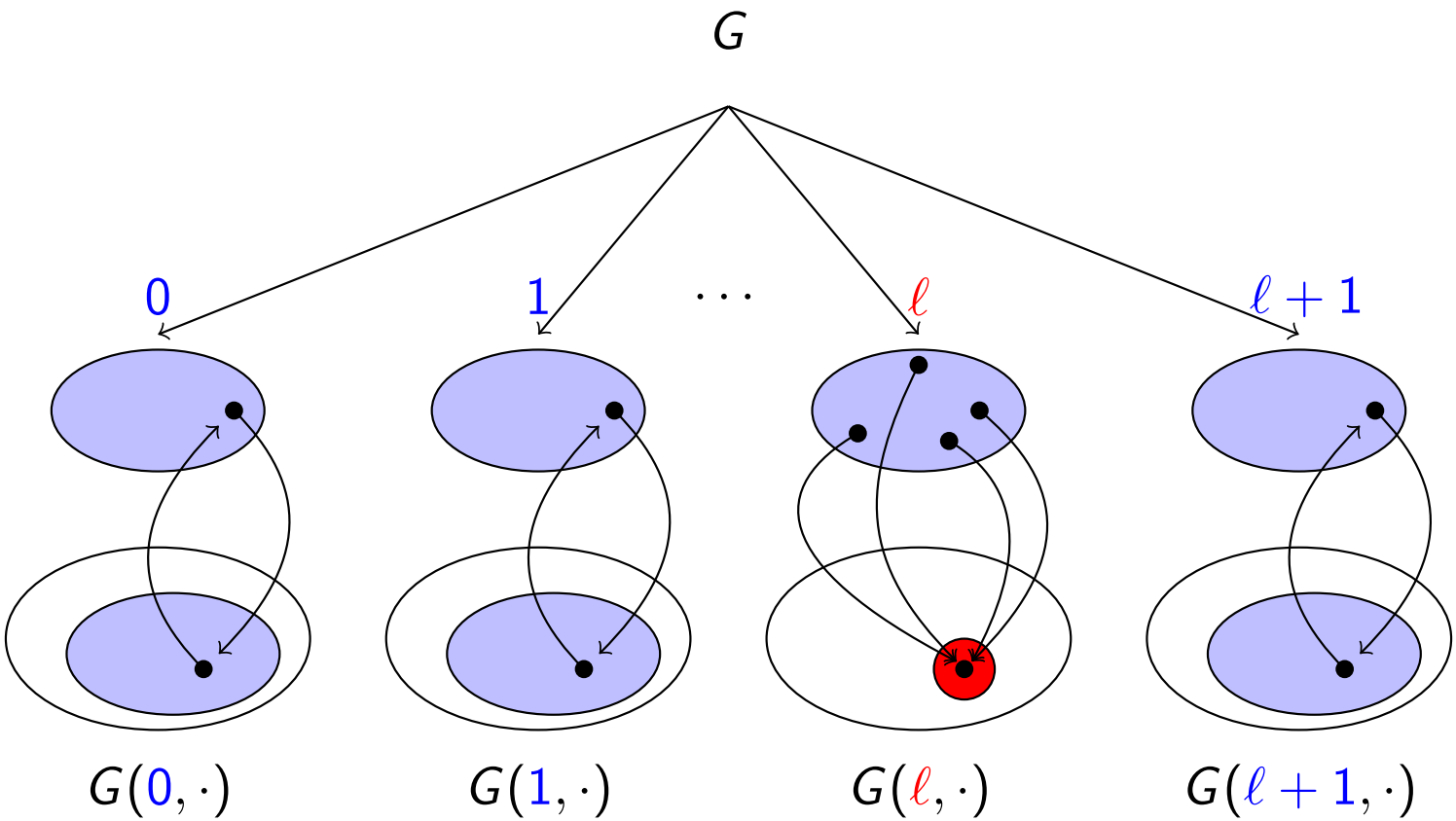
All-But-One TDF

- $G(b, x)$ has an extra parameter: branch $b \in \{0,1\}^n$
- Generate (G, G^{-1}) with hidden lossy branch ℓ

- Lossy TDF $\Longleftrightarrow$ All-But-One TDF

All-But-One TDF

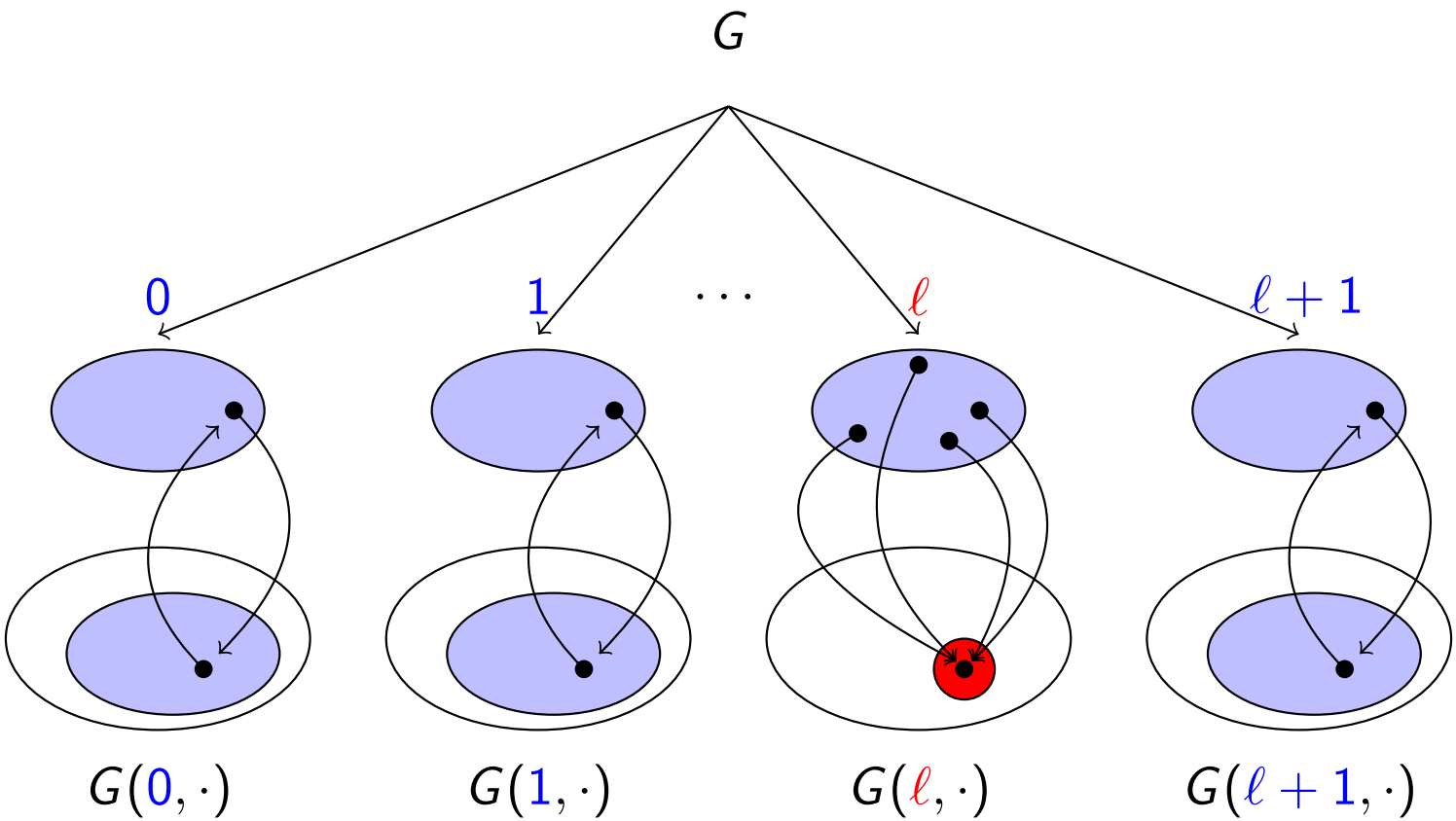
- $G(b, x)$ has an extra parameter: branch $b \in \{0,1\}^n$
- Generate (G, G^{-1}) with hidden lossy branch ℓ



- Lossy TDF $\iff$ All-But-One TDF

All-But-One TDF

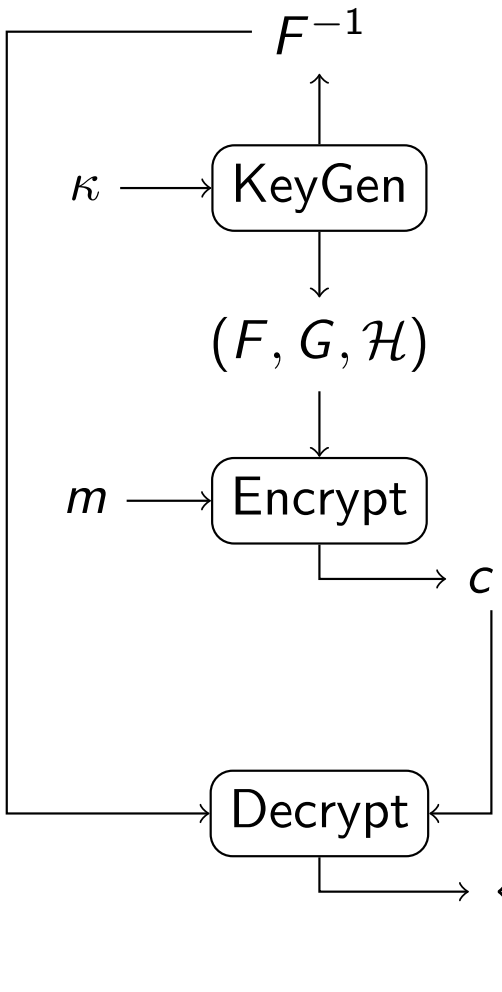
- $G(b, x)$ has an extra parameter: branch $b \in \{0,1\}^n$
- Generate (G, G^{-1}) with hidden lossy branch ℓ



- Lossy TDF $\iff$ All-But-One TDF

IND-CCA Encryption from LTDF: Construction

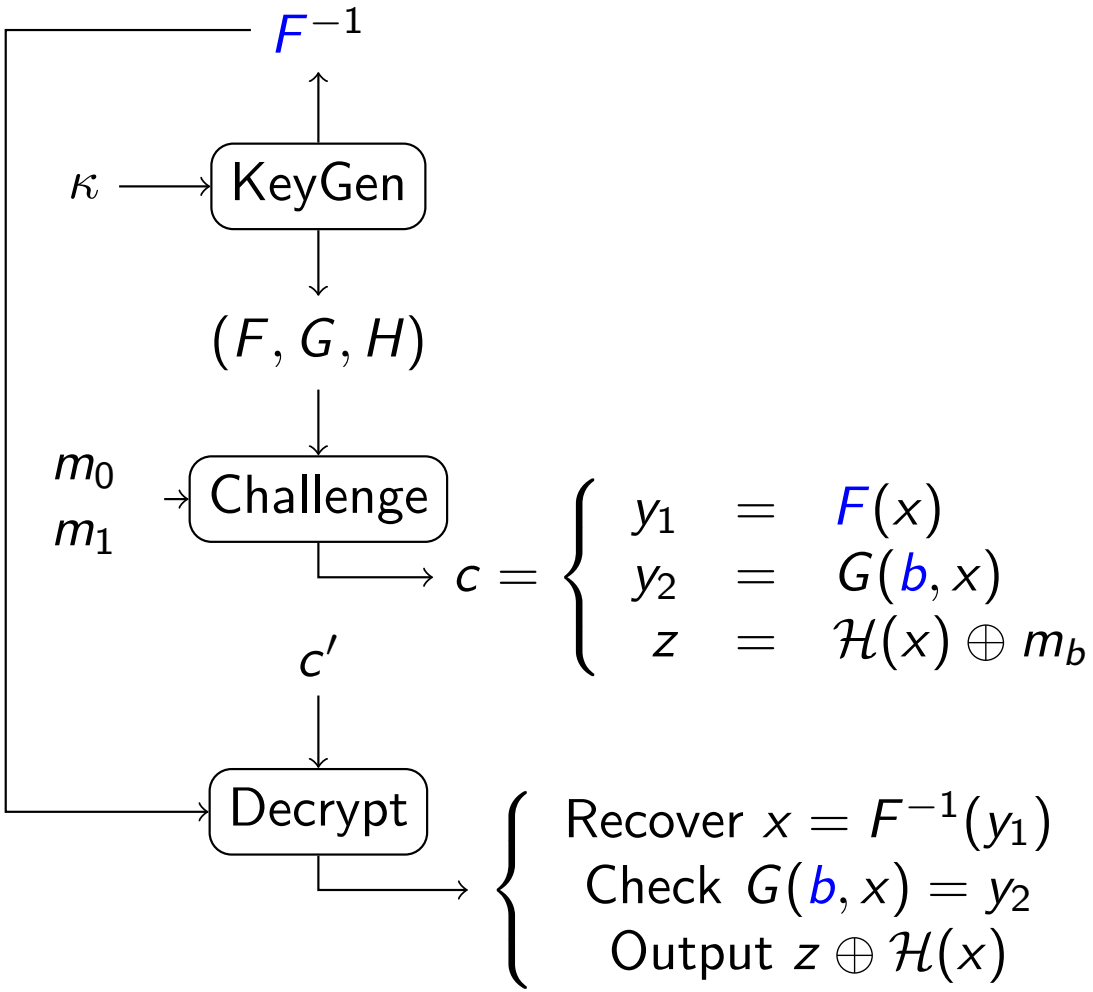
F = lossy TDF (lossiness k)
G = all-but-one TDF (lossiness k')
$\mathcal{H} : \{0,1\}^n \longrightarrow \{0,1\}^\ell$ randomness extractor for F
$(\ell < k + k')$



$$c = \begin{cases} y_1 &= F(x) \\ y_2 &= G(b, x) \\ z &= \mathcal{H}(x) \oplus m \end{cases}$$

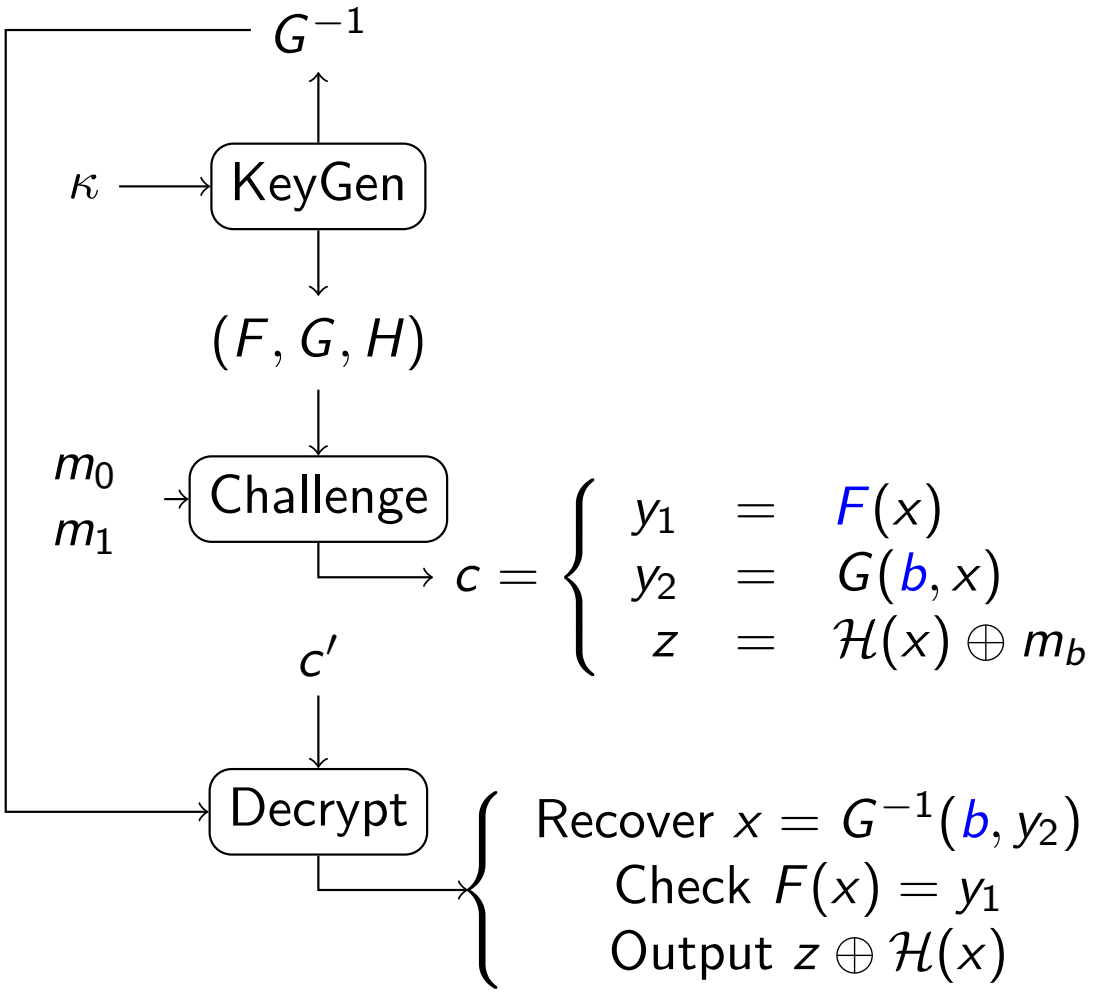
$$\begin{cases} \text{Recover } x = F^{-1}(y_1) \\ \text{Check } G(b, x) = y_2 \\ \text{Output } z \oplus \mathcal{H}(x) \end{cases}$$

IND-CCA Encryption from LTDF: Proof



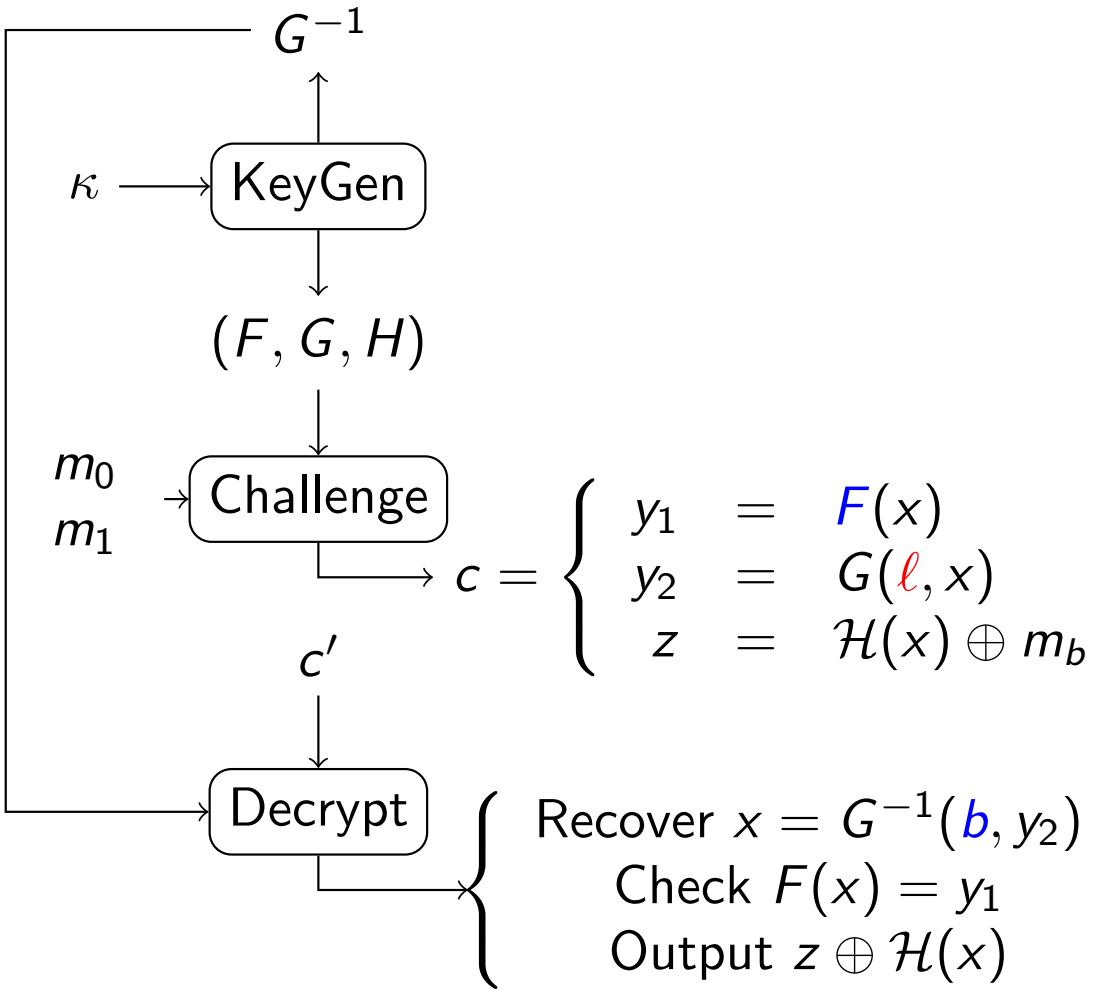
1. Real attack game

IND-CCA Encryption from LTDF: Proof



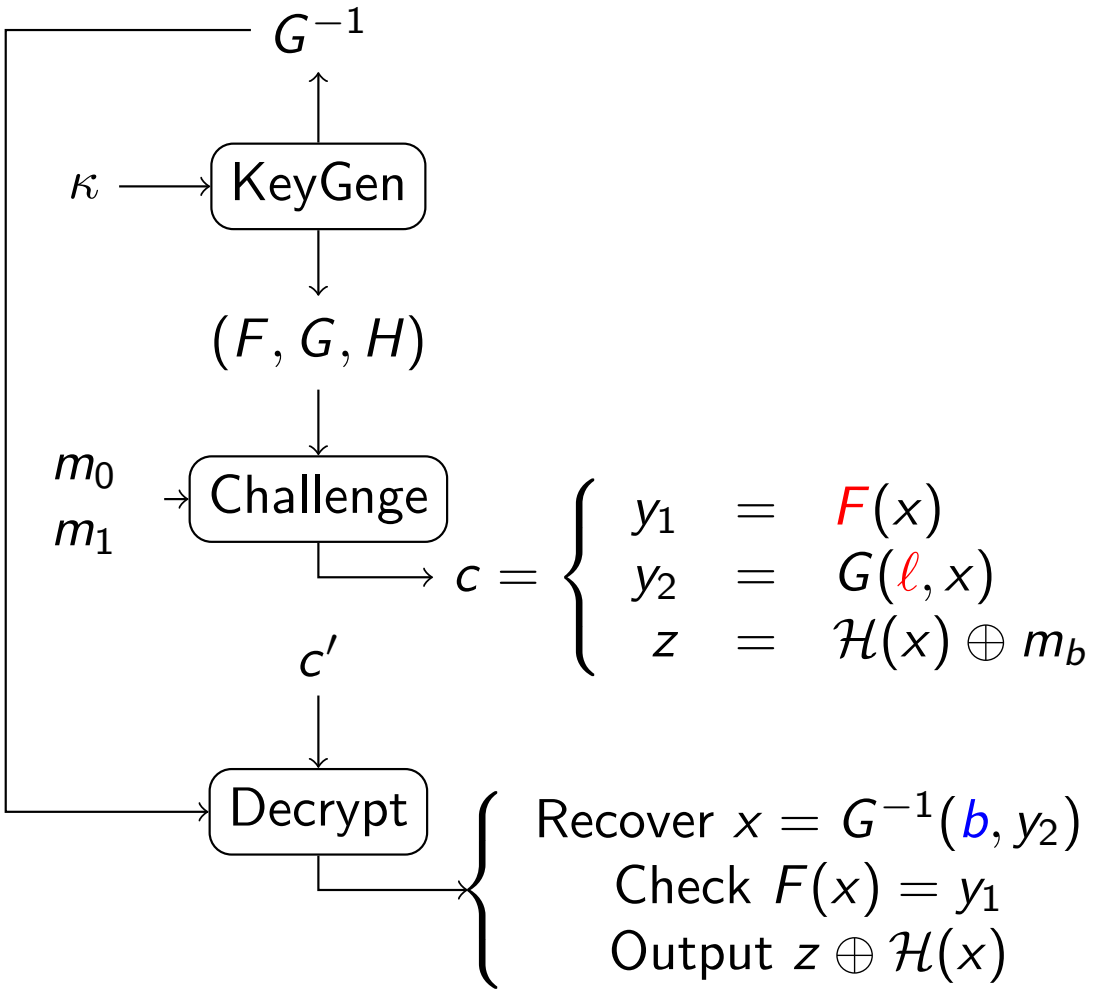
1. Real attack game
2. Perfect simulation

IND-CCA Encryption from LTDF: Proof



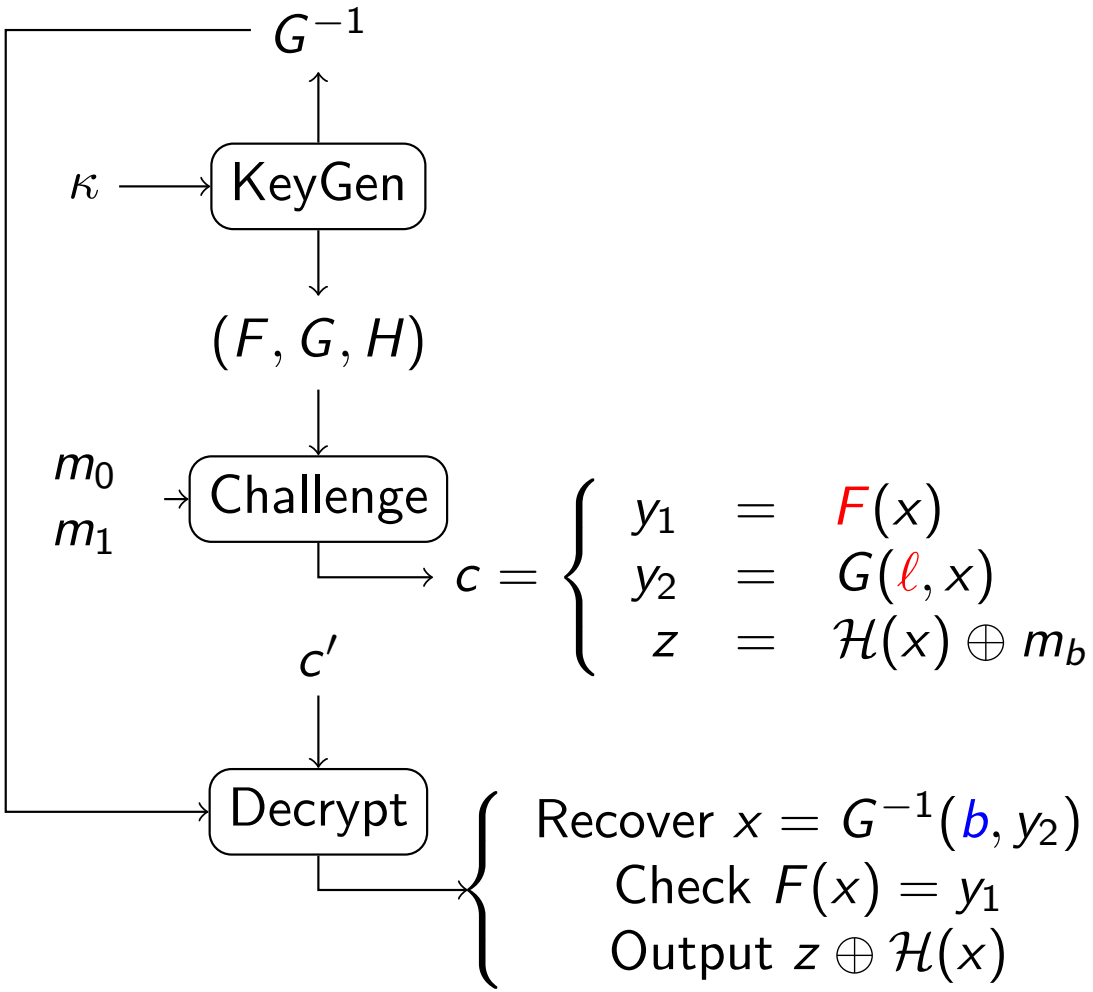
1. Real attack game
2. Perfect simulation
3. $G(\ell, \cdot) \simeq G(b, \cdot)$

IND-CCA Encryption from LTDF: Proof



1. Real attack game
2. Perfect simulation
3. $G(\ell, \cdot) \simeq G(b, \cdot)$
4. $F \simeq F$

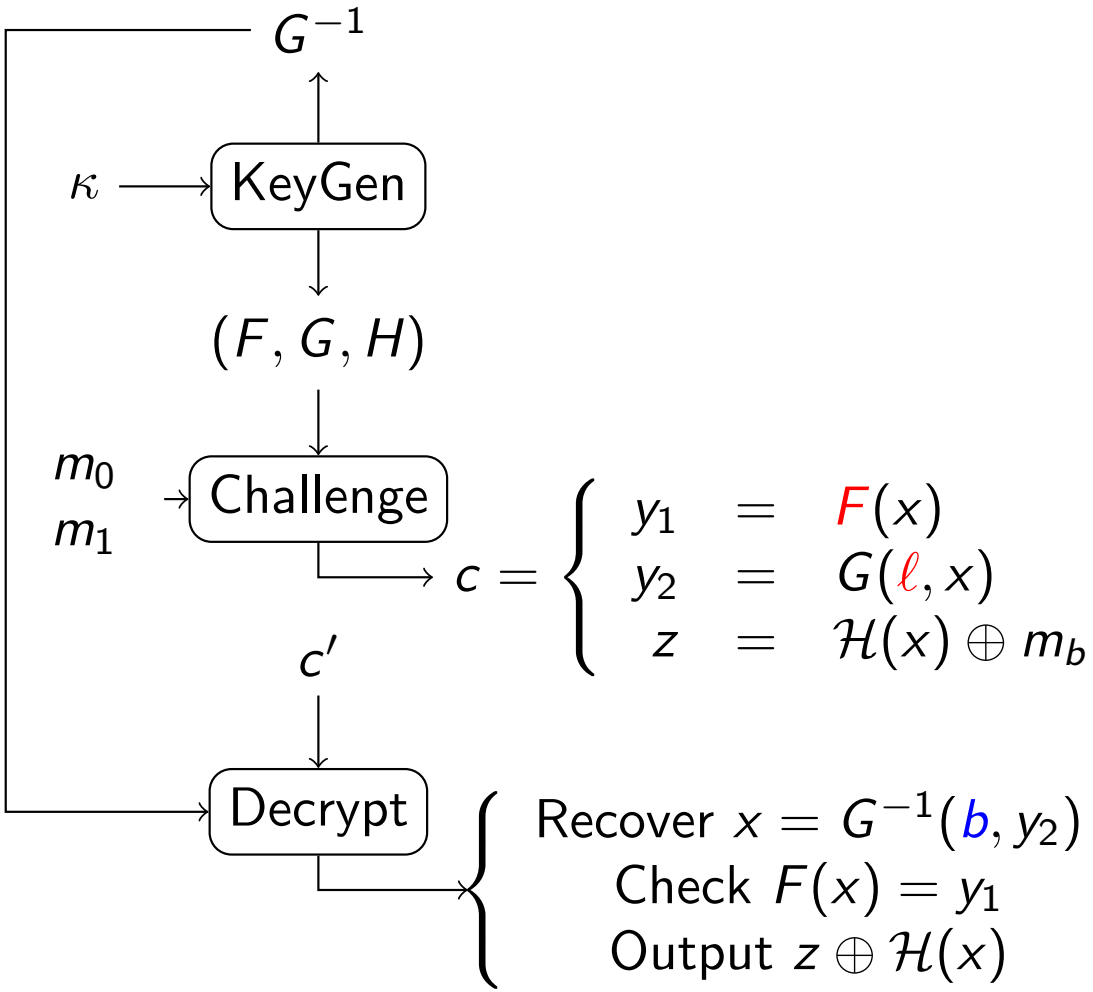
IND-CCA Encryption from LTDF: Proof



1. Real attack game
2. Perfect simulation
3. $G(\ell, \cdot) \simeq G(b, \cdot)$
4. $F \simeq F$

   $\rightsquigarrow$ m_b **statistically hidden**

IND-CCA Encryption from LTDF: Proof



1. Real attack game
2. Perfect simulation
3. $G(\ell, \cdot) \simeq G(b, \cdot)$
4. $F \simeq F$
   $\rightsquigarrow$ m_b **statistically hidden**

- (One-time) signature for CCA2 Security (Dolev-Dwork-Naor's trick)

Other Applications

- **Deterministic encryption**
  - A. Boldyreva, S. Fehr, A. O'Neill - Crypto 2008
- **Selective-opening secure encryption**
  - M. Bellare, D. Hofheinz, S. Yilek - Eurocrypt 2009
  - S. Fehr, D. Hofheinz, E. Kiltz, H. Wee - Eurocrypt 2010
  - B. Hemenway, B. Libert, R. Ostrovsky, D. V. - 2011+
- **Encryption with weak randomness**
  - M. Bellare, Z. Brakerski, M. Naor, T. Ristenpart, G. Segev, H. Shacham, S. Yilek - Asiacrypt 2009
  - G. Castagnos, M. Izabachène, B. Libert, D. V. - 2011+
- **IND-CPA Security of OAEP**
  - E. Kiltz, A. O'Neill, A. Smith - Crypto 2010
- ...

Contents

Realizing Lossy TDF: Rabin-Williams

$$N = pq, \qquad p, q \equiv 3 \bmod 4$$

- Squaring function $x \longmapsto x^2 \bmod N$ is lossy
  - 4-to-1 map on $\mathbb{Z}_N^* \rightsquigarrow$ 2 bits of lossiness

- but, inversion is possible if we know its
  - **Jacobi symbol**: $\left(\frac{x}{N}\right)$
  - **"Sign"**: $(x \text{ rem } N) \in [\![-N/2, \ldots, N/2]\!]$

$\rightsquigarrow$ a square $y \in \mathbb{Z}_N^*$ has four square roots $(\pm x_0, \pm x_1)$ with

$$\left(\frac{x_0}{N}\right) = \left(\frac{-x_0}{N}\right) = -\left(\frac{x_1}{N}\right) = -\left(\frac{-x_1}{N}\right)$$

Realizing Lossy TDF: Rabin-Williams

- How to make squaring **injective** ?
  $\rightsquigarrow$ encode two extra bits in the output
- How to encode these bits in a **computationally indistinguishable way** ?
  $\rightsquigarrow$ put them in the exponent of quadratic non-residues !

$$r, s \in \mathbb{Z}_N^* \text{ with } \left(\frac{r}{N}\right) = -1 \text{ and } \left(\frac{s}{N}\right) = 1 \text{ but quadratic non residue}$$

$$F : x \longmapsto y = x^2 \cdot r^{j(x)} \cdot s^{h(x)}$$

where

$$j(x) = 1 \iff \left(\frac{x}{N}\right) = -1 \text{ and } h(x) = 1 \iff (x \text{ rem } N) \in [\![-N/2, \ldots, -1]\!]$$

Realizing Lossy TDF: Rabin-Williams

- How to make squaring **injective** ?
  $\rightsquigarrow$ encode two extra bits in the output
- How to encode these bits in a **computationally indistinguishable way** ?
  $\rightsquigarrow$ put them in the exponent of quadratic non-residues !

$$r, s \in \mathbb{Z}_N^* \text{ with } \left(\frac{r}{N}\right) = -1 \text{ and } \left(\frac{s}{N}\right) = 1 \text{ but quadratic non residue}$$

$$F : x \longmapsto y = x^2 \cdot r^{j(x)} \cdot s^{h(x)}$$

where

$$j(x) = 1 \iff \left(\frac{x}{N}\right) = -1 \text{ and } h(x) = 1 \iff (x \text{ rem } N) \in [\![-N/2, \ldots, -1]\!]$$

Realizing Lossy TDF: Rabin-Williams

- How to make squaring **injective** ?
  $\rightsquigarrow$ encode two extra bits in the output
- How to encode these bits in a **computationally indistinguishable way** ?
  $\rightsquigarrow$ put them in the exponent of quadratic non-residues !

$$r, s \in \mathbb{Z}_N^* \text{ with } \left(\frac{r}{N}\right) = -1 \text{ and } \left(\frac{s}{N}\right) = 1 \text{ but quadratic non residue}$$

$$F : x \longmapsto y = x^2 \cdot r^{j(x)} \cdot s^{h(x)}$$

where

$$j(x) = 1 \iff \left(\frac{x}{N}\right) = -1 \text{ and } h(x) = 1 \iff (x \text{ rem } N) \in [\![-N/2, \ldots, -1]\!]$$

Realizing Lossy TDF: Rabin-Williams

$$F : x \longmapsto y = x^2 \cdot r^{j(x)} \cdot s^{h(x)}$$

where

$$j(x) = 1 \iff \left(\frac{x}{N}\right) = -1 \text{ and } h(x) = 1 \iff (x \text{ rem } N) \in [\![-N/2, \ldots, -1]\!]$$

$$r, s \in \mathbb{Z}_N^* \text{ with } \left(\frac{r}{N}\right) = -1 \text{ and } \left(\frac{s}{N}\right) = 1 \text{ quadratic } \textbf{non} \text{ residue}$$

$$r, s \in \mathbb{Z}_N^* \text{ with } \left(\frac{r}{N}\right) = -1 \text{ and } \left(\frac{s}{N}\right) = 1 \text{ quadratic residue}$$

- lossy ! (1 bit of lossiness)
- indistinguishable from injective under quadratic residuosity assumption

Realizing Lossy TDF: Homomorphic encryption

- Use an IND-CPA (additively) homomorphic cryptosystem

$$\text{Encrypt}(m) \odot \text{Encrypt}(m') \simeq \text{Encrypt}(m + m')$$

- **Idea:**
  - Encrypted $n \times n$ matrix: I_n for F and 0_n for F
  - $F(x)$ is computed by encrypted linear algebra

- but,

Realizing Lossy TDF: Homomorphic encryption

- Use an IND-CPA (additively) homomorphic cryptosystem

$$\text{Encrypt}(m) \odot \text{Encrypt}(m') \simeq \text{Encrypt}(m + m')$$

- **Idea:**
  - Encrypted $n \times n$ matrix: I_n for F and 0_n for F
  - $F(x)$ is computed by encrypted linear algebra
- but,

Realizing Lossy TDF: Homomorphic encryption

- Use an IND-CPA (additively) homomorphic cryptosystem

$$\text{Encrypt}(m) \odot \text{Encrypt}(m') \simeq \text{Encrypt}(m + m')$$

- **Idea:**
  - Encrypted $n \times n$ matrix: I_n for F and 0_n for F
  - $F(x)$ is computed by encrypted linear algebra

- but,

Realizing Lossy TDF: Homomorphic encryption

- Use an IND-CPA (additively) homomorphic cryptosystem

$$\text{Encrypt}(m) \odot \text{Encrypt}(m') \simeq \text{Encrypt}(m + m')$$

- **Idea:**
  - Encrypted $n \times n$ matrix: I_n for F and 0_n for F
  - $F(x)$ is computed by encrypted linear algebra

$$\begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

- but, randomness in each 0 leaks information on x ...

Realizing Lossy TDF: Homomorphic encryption

- Use an IND-CPA (additively) homomorphic cryptosystem

$$\text{Encrypt}(m) \odot \text{Encrypt}(m') \simeq \text{Encrypt}(m + m')$$

- **Idea:**
  - Encrypted $n \times n$ matrix: I_n for F and 0_n for F
  - $F(x)$ is computed by encrypted linear algebra

$$\begin{pmatrix} 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

- but, randomness in each 0 leaks information on x ...

Realizing Lossy TDF: Homomorphic encryption

- Use an IND-CPA (additively) homomorphic cryptosystem

$$\text{Encrypt}(m) \odot \text{Encrypt}(m') \simeq \text{Encrypt}(m + m')$$

- **Idea:**
  - Encrypted $n \times n$ matrix: I_n for F and 0_n for F
  - $F(x)$ is computed by encrypted linear algebra

$$\begin{pmatrix} 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

- but, randomness in each 0 leaks information on x ...

Realizing Lossy TDF: Homomorphic encryption

- $\rightsquigarrow$ Homomorphic cryptosystem with special properties:
  1. Secure to reuse randomness across different keys
  2. Homomorphism isolates randomness

Realizing Lossy TDF: Homomorphic encryption

- $\rightsquigarrow$ Homomorphic cryptosystem with special properties:
  1. Secure to reuse randomness across different keys
  2. Homomorphism isolates randomness

Realizing Lossy TDF: Homomorphic encryption

- $\rightsquigarrow$ Homomorphic cryptosystem with special properties:
  1. Secure to reuse randomness across different keys
  2. Homomorphism isolates randomness

$$\begin{pmatrix} 0; r_1 & 0; r_2 & \dots & 0; r_n \\ 0; r_1 & 0; r_2 & \dots & 0; r_n \\ \vdots & & \ddots & \vdots \\ 0; r_1 & 0; r_2 & \dots & 0; r_n \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} 0; R \\ 0; R \\ \vdots \\ 0; R \end{pmatrix}$$

- Lossy when $n > |R|$!

Realizing Lossy TDF: Homomorphic encryption

- $\rightsquigarrow$ Homomorphic cryptosystem with special properties:
  1. Secure to reuse randomness across different keys
  2. Homomorphism isolates randomness

$$\begin{pmatrix} 0; r_1 & 0; r_2 & \dots & 0; r_n \\ 0; r_1 & 0; r_2 & \dots & 0; r_n \\ \vdots & & \ddots & \vdots \\ 0; r_1 & 0; r_2 & \dots & 0; r_n \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} 0; R \\ 0; R \\ \vdots \\ 0; R \end{pmatrix}$$

- Lossy when $n > |R|$!

Realizing Lossy TDF: Homomorphic encryption

- **Decision Diffie-Hellman** (ElGamal)
  - message in exponent $\rightsquigarrow$ additive homomorphism
  - secure when reusing randomness (Kurosawa . . .)
- **Learning with Errors** (Regev)
  - bounded homomorphism
  - reusing most randomness (not error terms)
- **Composite Residuosity** (Paillier)
  - more efficient construction

Realizing Lossy TDF: Homomorphic encryption

- **Decision Diffie-Hellman** (ElGamal)
  - message in exponent $\rightsquigarrow$ additive homomorphism
  - secure when reusing randomness (Kurosawa . . .)
- **Learning with Errors** (Regev)
  - bounded homomorphism
  - reusing most randomness (not error terms)
- **Composite Residuosity** (Paillier)
  - more efficient construction

Realizing Lossy TDF: Homomorphic encryption

- **Decision Diffie-Hellman** (ElGamal)
  - message in exponent $\rightsquigarrow$ additive homomorphism
  - secure when reusing randomness (Kurosawa . . .)
- **Learning with Errors** (Regev)
  - bounded homomorphism
  - reusing most randomness (not error terms)
- **Composite Residuosity** (Paillier)
  - more efficient construction

Realizing Lossy TDF: Paillier

$$N = pq, \qquad \psi(m, r) = (1 + N)^m r^N \bmod N^2$$

(can be inverted given $\lambda(N) = \mathrm{lcm}(P - 1, Q - 1)$)

- $S_{\mathrm{inj}} \longrightarrow s = (N = pq, c = (1 + N)r^N \bmod N^2), t = \lambda(N)$
- $S_{\mathrm{loss}} \longrightarrow s = (N = pq, c = r^N \bmod N^2)$
- $\left.\begin{array}{c} F_{\mathrm{inj}}(s) \\ F_{\mathrm{loss}}(s) \end{array}\right\} : (x, y) \longmapsto c^x y^N \bmod N^2$

Lossy ! ($n/2$ bit of lossiness)

Realizing Lossy TDF: Paillier

$$N = pq, \qquad \psi(m, r) = (1 + N)^m r^N \bmod N^2$$

(can be inverted given $\lambda(N) = \mathrm{lcm}(P - 1, Q - 1)$)

- $S_{\mathrm{inj}} \longrightarrow s = (N = pq, c = (1 + N)r^N \bmod N^2), t = \lambda(N)$
- $S_{\mathrm{loss}} \longrightarrow s = (N = pq, c = r^N \bmod N^2)$
- $\left.\begin{array}{c} F_{\mathrm{inj}}(s) \\ F_{\mathrm{loss}}(s) \end{array}\right\} : (x, y) \longmapsto c^x y^N \bmod N^2$

Lossy ! ($n/2$ bit of lossiness)

Contents

Extensions of Lossy TDF

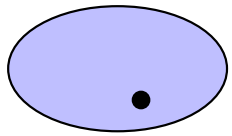
- Correlated Products Secure Trapdoor Functions
  - A. Rosen G. Segev - TCC 2009
- Towards a Theory of Extractable Functions
  - R. Canetti, R. R. Dakdouk - TCC 2009
- Adaptive Trapdoor Functions
  - E. Kiltz, P. Mohassel, A. O'Neill - Eurocrypt 2010
- Adaptive Trapdoor Relations
  - H. Wee - Crypto 2010
- ...

Extensions of Lossy TDF

- Correlated Products Secure Trapdoor Functions
  - A. Rosen G. Segev - TCC 2009
- Towards a Theory of Extractable Functions
  - R. Canetti, R. R. Dakdouk - TCC 2009
- Adaptive Trapdoor Functions
  - E. Kiltz, P. Mohassel, A. O'Neill - Eurocrypt 2010
- Adaptive Trapdoor Relations
  - H. Wee - Crypto 2010
- ...

Correlation-Secure TDF

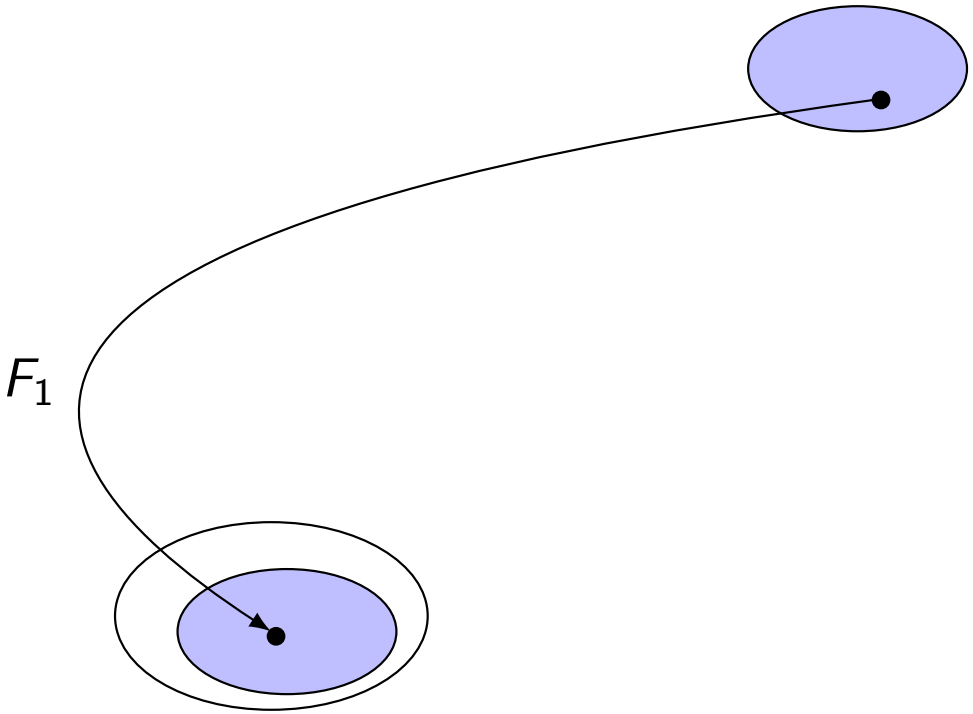
$\mathcal{F} = (F_1, \ldots, F_\ell)$ TDF family s.t. $x \longmapsto F_1(x) \| \ldots \| F_\ell(x)$ is one way



Correlated-Secure Trapdoor One-Way Function

Correlation-Secure TDF

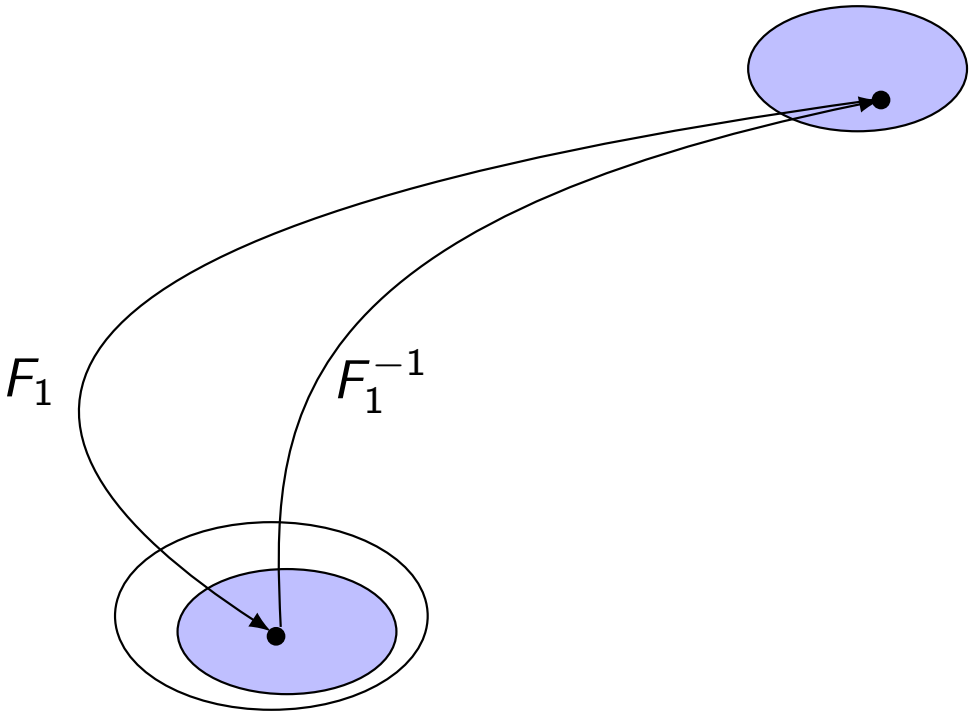
$\mathcal{F} = (F_1, \ldots, F_\ell)$ TDF family s.t. $x \longmapsto F_1(x) \| \ldots \| F_\ell(x)$ is one way



Correlated-Secure Trapdoor One-Way Function

Correlation-Secure TDF

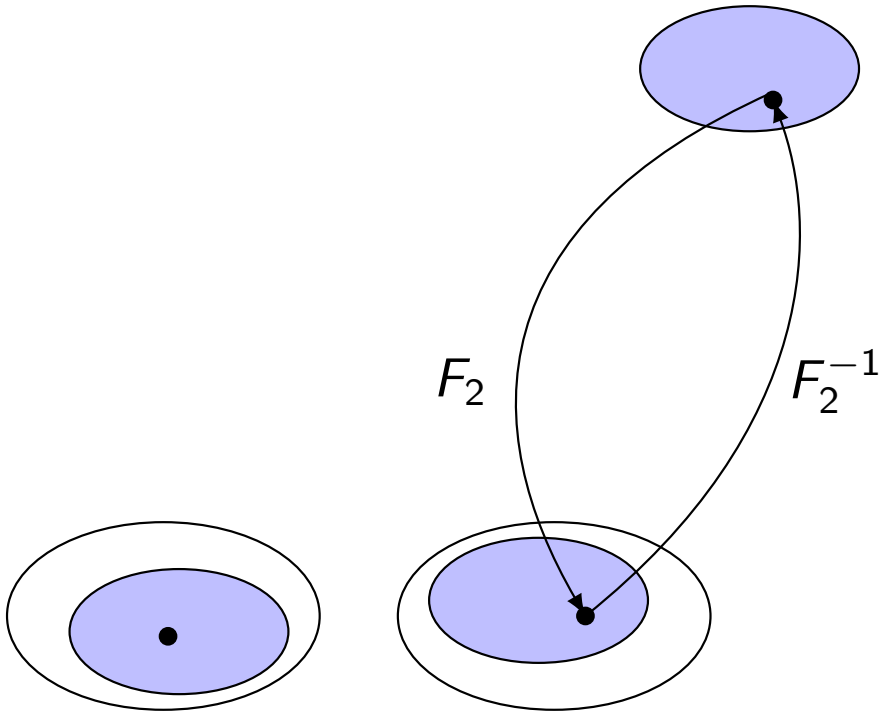
$\mathcal{F} = (F_1, \ldots, F_\ell)$ TDF family s.t. $x \longmapsto F_1(x) \| \ldots \| F_\ell(x)$ is one way



Correlated-Secure Trapdoor One-Way Function

Correlation-Secure TDF

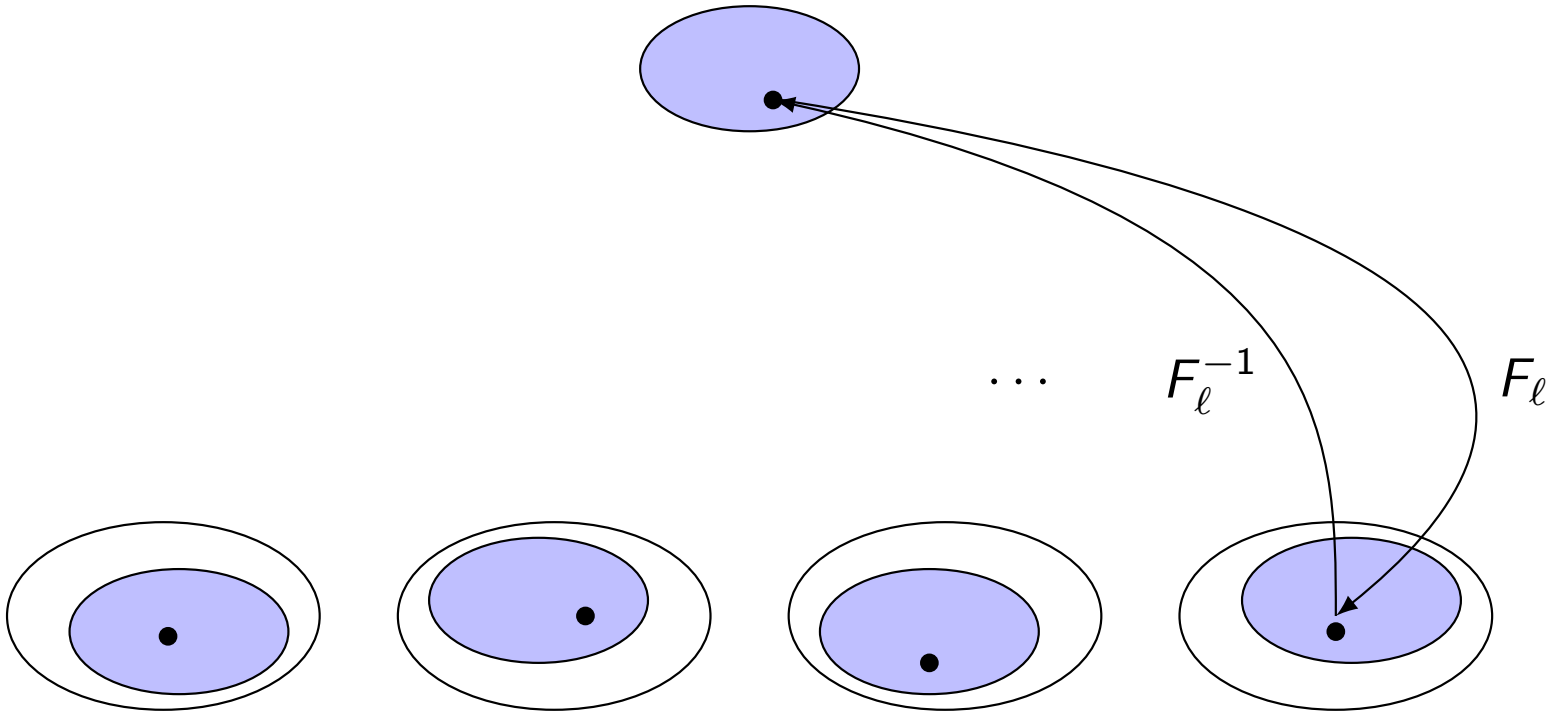
$\mathcal{F} = (F_1, \ldots, F_\ell)$ TDF family s.t. $x \longmapsto F_1(x) \| \ldots \| F_\ell(x)$ is one way



Correlated-Secure Trapdoor One-Way Function

Correlation-Secure TDF

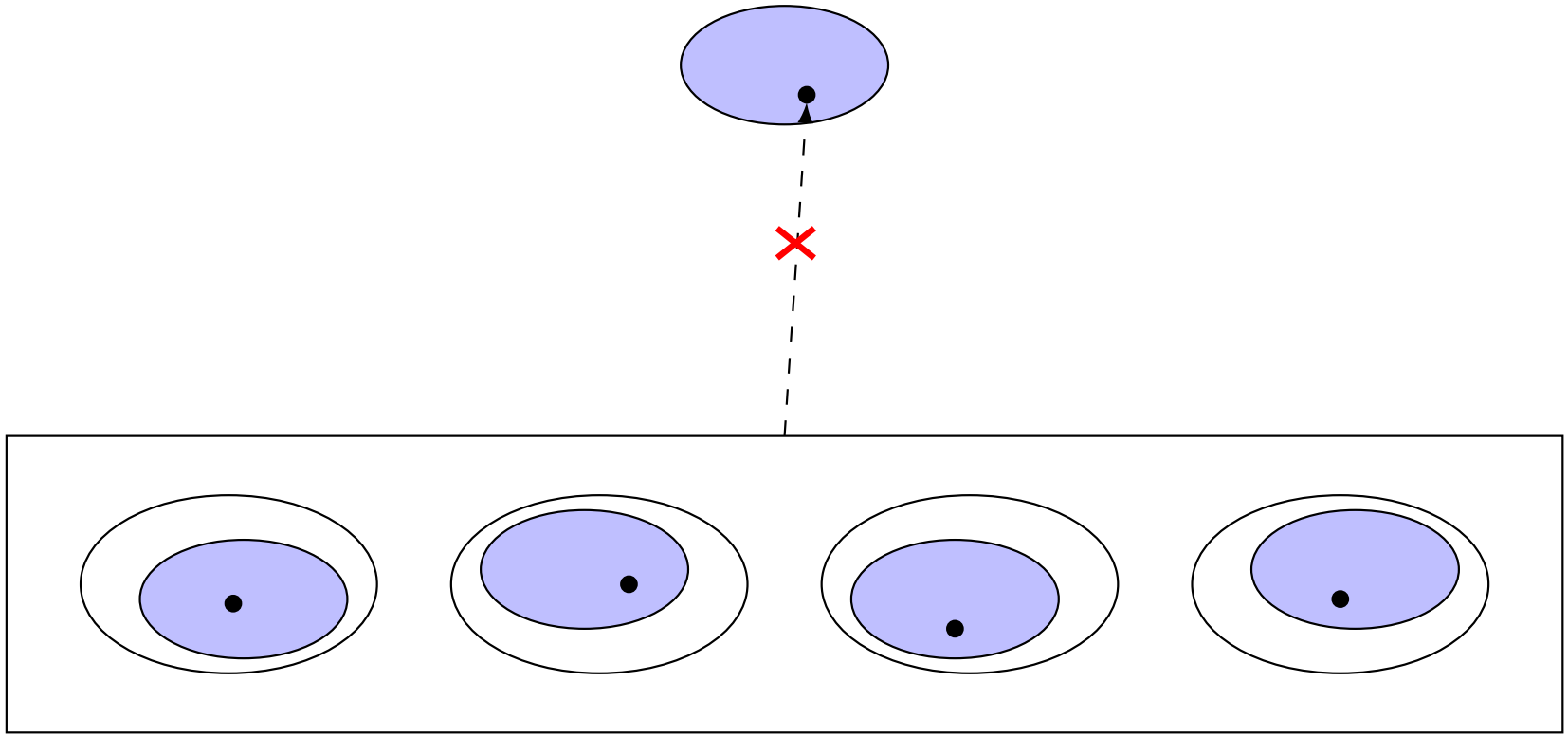
$\mathcal{F} = (F_1, \ldots, F_\ell)$ TDF family s.t. $x \longmapsto F_1(x) \| \ldots \| F_\ell(x)$ is one way



Correlated-Secure Trapdoor One-Way Function

Correlation-Secure TDF

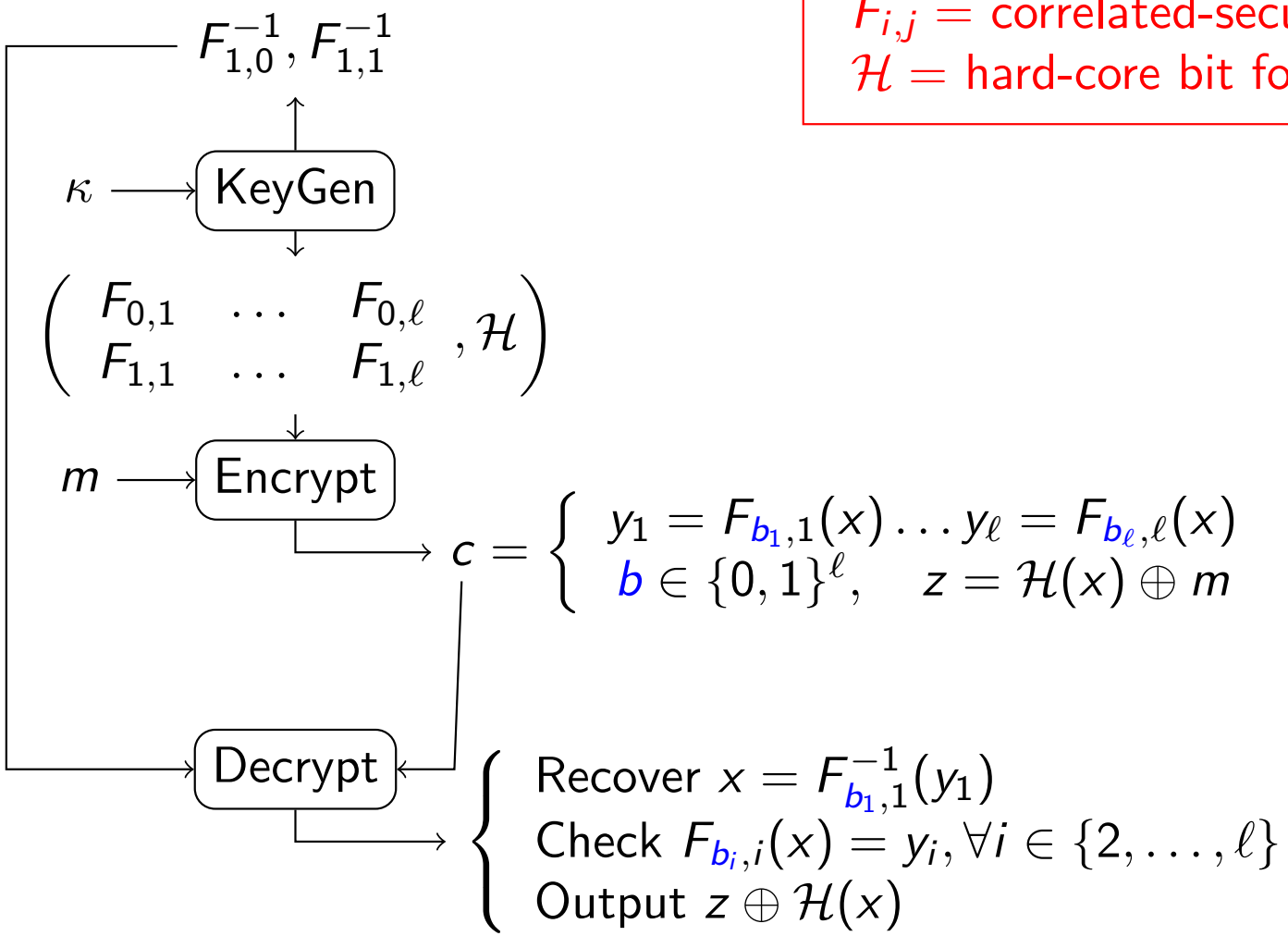
$\mathcal{F} = (F_1, \ldots, F_\ell)$ TDF family s.t. $x \longmapsto F_1(x) \| \ldots \| F_\ell(x)$ is one way



Correlated-Secure Trapdoor One-Way Function

IND-CCA Security from Correlation-Secure TDF: Construction

$F_{i,j}$ = correlated-secure TDF
$\mathcal{H}$ = hard-core bit for $(F_{i,j})_{i,j}$



Realizing Correlation-Secure TDF

Generic constructions (Rosen and Segev, 2009)

- Lossy TDF : $\{0,1\}^n \longrightarrow \{0,1\}^n$ with $n(1-\rho)$ lossiness
  $\Rightarrow$ Correlated Secure TDF under $1/\rho$-repetition

- Correlated Secure TDF $\not\Rightarrow$ LossyTDF

Specific constructions

- RSA ? $\rightsquigarrow$ No ! (Hastad)
- Niederreiter/McEliece ? $\rightsquigarrow$ Yes !
  (Fischer and Stern, 1996) - (Freeman *et al.*, 2010)

Realizing Correlation-Secure TDF

Generic constructions (Rosen and Segev, 2009)

- Lossy TDF : $\{0,1\}^n \longrightarrow \{0,1\}^n$ with $n(1-\rho)$ lossiness
  $\Rightarrow$ Correlated Secure TDF under $1/\rho$-repetition

- Correlated Secure TDF $\not\Rightarrow$ LossyTDF

Specific constructions

- RSA ? $\rightsquigarrow$ No ! (Hastad)
- Niederreiter/McEliece ? $\rightsquigarrow$ Yes !
  (Fischer and Stern, 1996) - (Freeman *et al.*, 2010)

Realizing Correlation-Secure TDF

Generic constructions (Rosen and Segev, 2009)

- Lossy TDF : $\{0,1\}^n \longrightarrow \{0,1\}^n$ with $n(1-\rho)$ lossiness
  $\Rightarrow$ Correlated Secure TDF under $1/\rho$-repetition

- Correlated Secure TDF $\not\Rightarrow$ LossyTDF

Specific constructions

- RSA ? $\rightsquigarrow$ No ! (Hastad)
- Niederreiter/McEliece ? $\rightsquigarrow$ Yes !
  (Fischer and Stern, 1996) - (Freeman *et al.*, 2010)

Realizing Correlation-Secure TDF

Generic constructions (Rosen and Segev, 2009)

- Lossy TDF : $\{0,1\}^n \longrightarrow \{0,1\}^n$ with $n(1-\rho)$ lossiness
  $\Rightarrow$ Correlated Secure TDF under $1/\rho$-repetition

- Correlated Secure TDF $\not\Rightarrow$ LossyTDF

Specific constructions

- RSA ? $\rightsquigarrow$ No ! (Hastad)
- Niederreiter/McEliece ? $\rightsquigarrow$ Yes !
  (Fischer and Stern, 1996) - (Freeman *et al.*, 2010)

Realizing Correlation-Secure TDF

Generic constructions (Rosen and Segev, 2009)

- Lossy TDF : $\{0,1\}^n \longrightarrow \{0,1\}^n$ with $n(1-\rho)$ lossiness
  $\Rightarrow$ Correlated Secure TDF under $1/\rho$-repetition

- Correlated Secure TDF $\not\Rightarrow$ LossyTDF

Specific constructions

- RSA ? $\rightsquigarrow$ No ! (Hastad)
- Niederreiter/McEliece ? $\rightsquigarrow$ Yes !
  (Fischer and Stern, 1996) - (Freeman *et al.*, 2010)

Contents

Open Problems

Morgan, Pierre-Louis, Léonard, Julien, ...

- LTDF from codes ?
- Amplify lossiness ?

Yuanmi, Léo, Mariya, Xavier ...

- Efficient LTDF from LWE ?
- Efficient Special signatures from Lattices ?

Julien, Iwen, Sophie, Mario, ...

- Compressed LTDF from DDH ?
- Efficient NIZK Proof System from LTDF ?

Clément, Thomas, Soline, Vanessa, ...

- Lossy/Correlated Secure TDF from RSA ?
- Efficient LTDF from Rabin-Williams ?

Open Problems

Morgan, Pierre-Louis, Léonard, Julien, ...

- LTDF from codes ?
- Amplify lossiness ?

Yuanmi, Léo, Mariya, Xavier ...

- Efficient LTDF from LWE ?
- Efficient Special signatures from Lattices ?

Julien, Iwen, Sophie, Mario, ...

- Compressed LTDF from DDH ?
- Efficient NIZK Proof System from LTDF ?

Clément, Thomas, Soline, Vanessa, ...

- Lossy/Correlated Secure TDF from RSA ?
- Efficient LTDF from Rabin-Williams ?

Open Problems

Morgan, Pierre-Louis, Léonard, Julien, ...

- LTDF from codes ?
- Amplify lossiness ?

Yuanmi, Léo, Mariya, Xavier ...

- Efficient LTDF from LWE ?
- Efficient Special signatures from Lattices ?

Julien, Iwen, Sophie, Mario, ...

- Compressed LTDF from DDH ?
- Efficient NIZK Proof System from LTDF ?

Clément, Thomas, Soline, Vanessa, ...

- Lossy/Correlated Secure TDF from RSA ?
- Efficient LTDF from Rabin-Williams ?

Open Problems

Morgan, Pierre-Louis, Léonard, Julien, ...

- LTDF from codes ?
- Amplify lossiness ?

Yuanmi, Léo, Mariya, Xavier ...

- Efficient LTDF from LWE ?
- Efficient Special signatures from Lattices ?

Julien, Iwen, Sophie, Mario, ...

- Compressed LTDF from DDH ?
- Efficient NIZK Proof System from LTDF ?

Clément, Thomas, Soline, Vanessa, ...

- Lossy/Correlated Secure TDF from RSA ?
- Efficient LTDF from Rabin-Williams ?