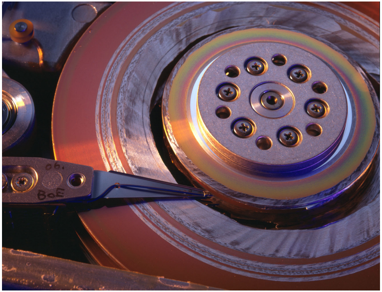
Lossy Trapdoor Functions and Their Applications

Chris Peikert   Brent Waters

SRI International

On Losing Information



On Losing Information

On Losing Information

On Losing Information

On Losing Information

2.3 MB → 0.4 MB

On Losing Information

On Losing Information

Lossy object *indistinguishable from original*

This Talk

1. Trapdoor functions without factoring: discrete log & lattices

This Talk

1. Trapdoor functions without factoring: discrete log & lattices
2. Black-box chosen-ciphertext security via randomness recovery

This Talk

1. Trapdoor functions without factoring: discrete log & lattices

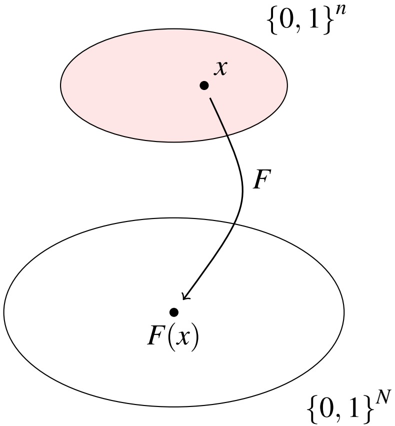
2. Black-box chosen-ciphertext security via randomness recovery

3. A new general primitive: Lossy Trapdoor Functions

Public Key Cryptography

1-1 Trapdoor Functions

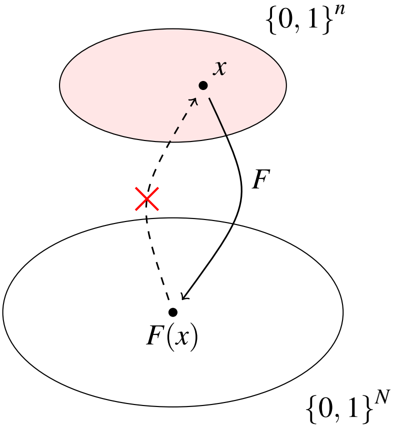
$$(F, F^{-1}) \leftarrow S$$



Public Key Cryptography

1-1 Trapdoor Functions

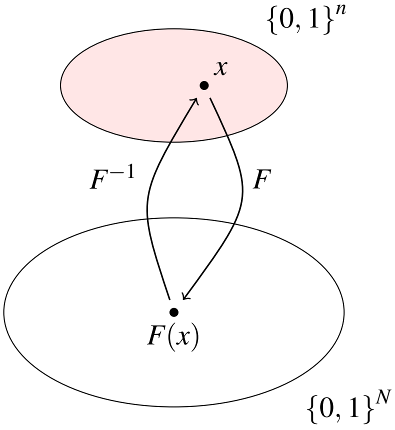
$$(\,F\,,\,F^{-1}\,) \leftarrow S$$



Public Key Cryptography

1-1 Trapdoor Functions

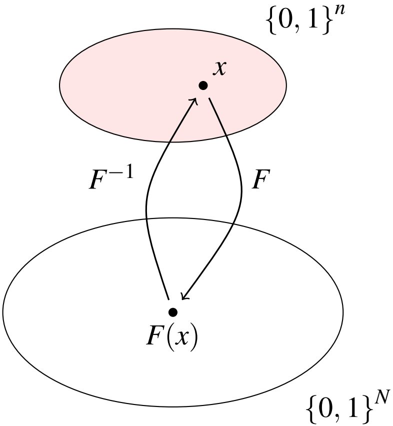
$$(F , F^{-1}) \leftarrow S$$



Public Key Cryptography

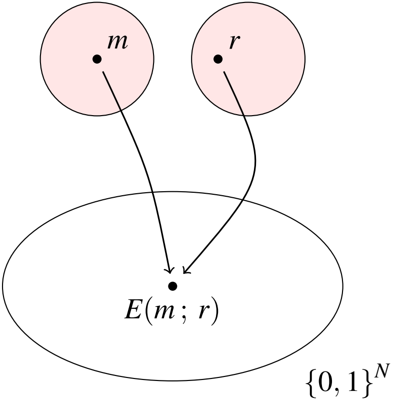
1-1 Trapdoor Functions

$$(F, F^{-1}) \leftarrow S$$



Public Key Encryption

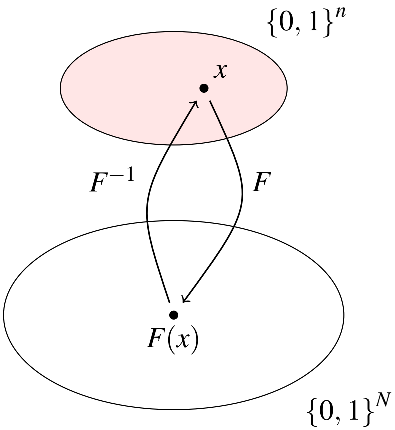
$$(E, D) \leftarrow S$$



Public Key Cryptography

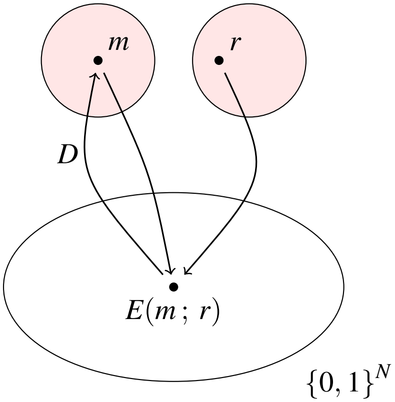
1-1 Trapdoor Functions

$$(F, F^{-1}) \leftarrow S$$



Public Key Encryption

$$(E, D) \leftarrow S$$



Realizing Public Key Crypto

	Factoring	Discrete log	Lattices
PKE	✔ [RSA,. . .]	✔ [ElGamal]	✔ [AD,R1,R2]
CCA	✔ [DDN,. . . ,CS2]	✔ [CS1]	??
TDF	✔ [RSA,R,P]	??	??

Realizing Public Key Crypto

	Factoring	Discrete log	Lattices
PKE	✔ [RSA,. . .]	✔ [ElGamal]	✔ [AD,R1,R2]
CCA	✔ [DDN,. . . ,CS2]	✔ [CS1]	??
TDF	✔ [RSA,R,P]	??	??

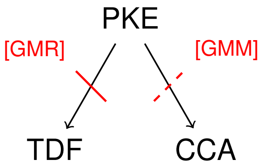
Lattice-Based Crypto:

- Simple & parallelizable
- Resist *quantum* algorithms (so far)
- Security from *worst-case* assumptions [Ajtai,. . .]

Realizing Public Key Crypto

	Factoring	Discrete log	Lattices
PKE	✔ [RSA,. . .]	✔ [ElGamal]	✔ [AD,R1,R2]
CCA	✔ [DDN,. . . ,CS2]	✔ [CS1]	??
TDF	✔ [RSA,R,P]	??	??

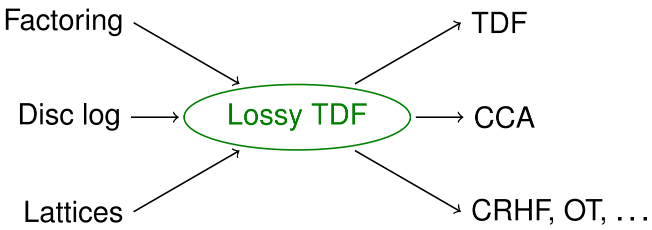
Black-Box Separations:



Realizing Public Key Crypto

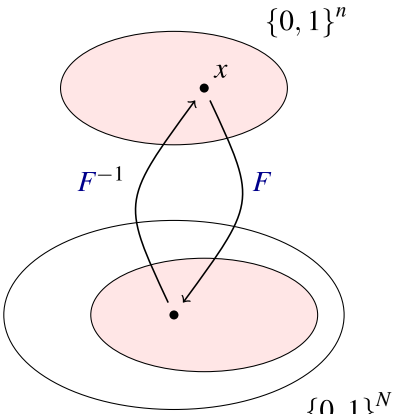
	Factoring	Discrete log	Lattices
PKE	✔ [RSA,. . .]	✔ [ElGamal]	✔ [AD,R1,R2]
CCA	✔ [DDN,. . . ,CS2]	✔ [CS1]	✔
TDF	✔ [RSA,R,P]	✔	✔

This Work:

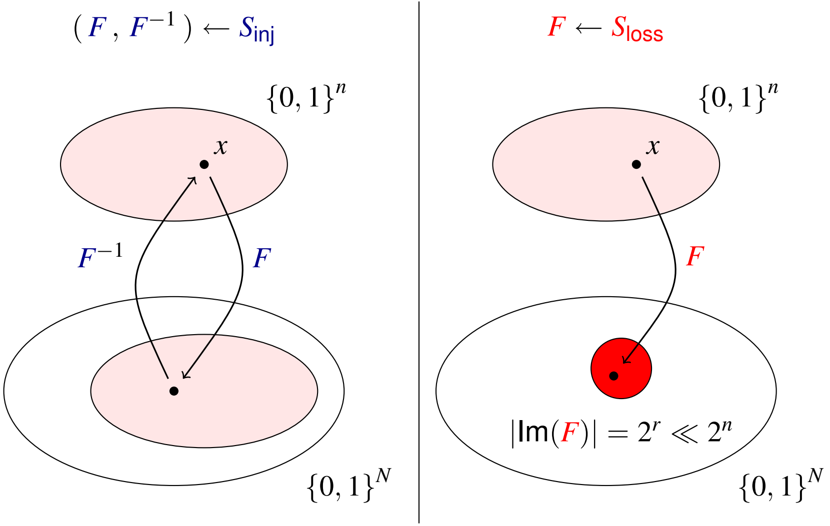


Lossy Trapdoor Functions

$(F, F^{-1}) \leftarrow S_{\text{inj}}$

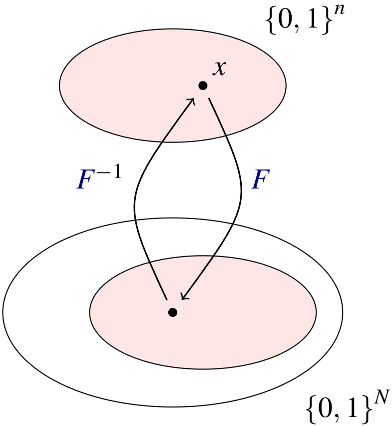


Lossy Trapdoor Functions

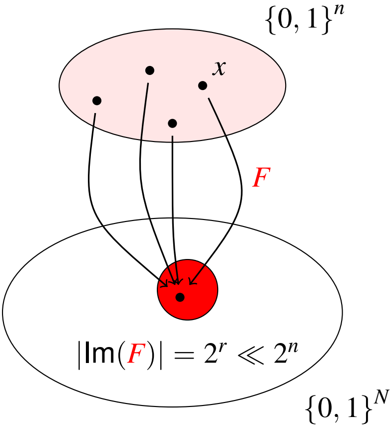


Lossy Trapdoor Functions

$(F, F^{-1}) \leftarrow S_{\text{inj}}$



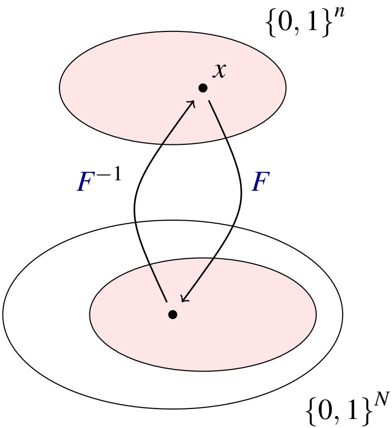
$F \leftarrow S_{\text{loss}}$



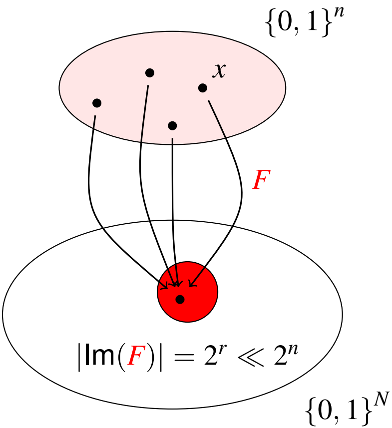
Lossy Trapdoor Functions

$$F \overset{c}{\approx} F$$

$(F, F^{-1}) \leftarrow S_{\text{inj}}$



$F \leftarrow S_{\text{loss}}$



Lossy TDFs $\Rightarrow$ 1-1 Trapdoor Functions

Theorem

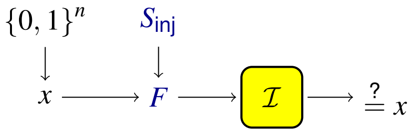
- S_{inj} generates 1-1 trapdoor functions (F, F^{-1}).

Lossy TDFs $\Rightarrow$ 1-1 Trapdoor Functions

Theorem

- S_{inj} generates 1-1 trapdoor functions (F, F^{-1}).

- Efficient $\mathcal{I}$ wants to invert F.

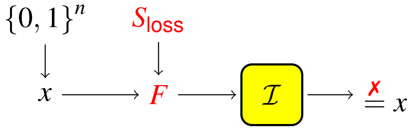


Lossy TDFs ⇒ 1-1 Trapdoor Functions

Theorem

- S_{inj} generates 1-1 trapdoor functions (F, F^{-1}).

- Efficient $\mathcal{I}$ wants to invert F.

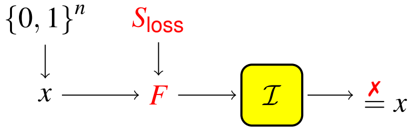


Lossy TDFs ⇒ 1-1 Trapdoor Functions

Theorem

- S_{inj} generates 1-1 trapdoor functions (F, F^{-1}).

- Efficient $\mathcal{I}$ wants to invert F.



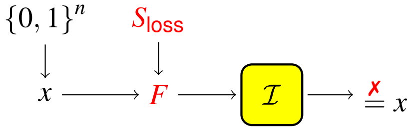
- $F(x)$ has 2^{n-r} preimages (on average).

Lossy TDFs $\Rightarrow$ 1-1 Trapdoor Functions

Theorem

- S_{inj} generates 1-1 trapdoor functions (F, F^{-1}).

- Efficient $\mathcal{I}$ wants to invert F.



- $F(x)$ has 2^{n-r} preimages (on average).

Main Technique

- Swapping F with F yields *statistically secure* system.

Lossy TDFs $\Rightarrow$ Public-Key Encryption

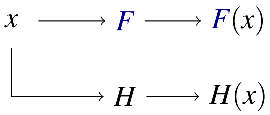
- Hard-core functions [GoldreichLevin] — the lazy way.

Lossy TDFs $\Rightarrow$ Public-Key Encryption

- Hard-core functions [GoldreichLevin] — the lazy way.
  - Pairwise independent $H : \{0,1\}^n \to \{0,1\}^k$ for $k \approx n - r$.

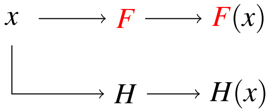
Lossy TDFs ⇒ Public-Key Encryption

- Hard-core functions [GoldreichLevin] — the lazy way.
  - Pairwise independent $H : \{0,1\}^n \to \{0,1\}^k$ for $k \approx n - r$.



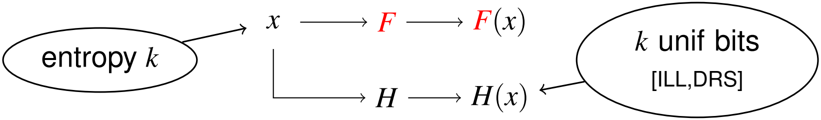
Lossy TDFs ⇒ Public-Key Encryption

- Hard-core functions [GoldreichLevin] — the lazy way.
    - Pairwise independent $H : \{0,1\}^n \to \{0,1\}^k$ for $k \approx n - r$.



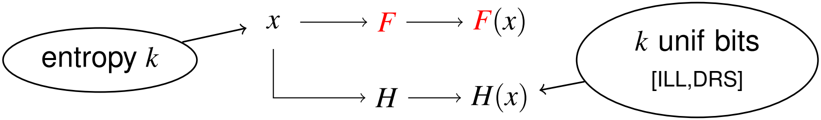
Lossy TDFs $\Rightarrow$ Public-Key Encryption

- Hard-core functions [GoldreichLevin] — the lazy way.
  - Pairwise independent $H : \{0,1\}^n \to \{0,1\}^k$ for $k \approx n - r$.



Lossy TDFs $\Rightarrow$ Public-Key Encryption

- Hard-core functions [GoldreichLevin] — the lazy way.
  - Pairwise independent $H : \{0,1\}^n \to \{0,1\}^k$ for $k \approx n - r$.



- Public key (F, H), secret key F^{-1}.

  Encrypt $m \in \{0,1\}^k$ as $(F(x),\ m \oplus H(x))$.

Chosen Ciphertext-Secure Encryption

Intuitive Definition [DDN,NY,RS]

- Encryption hides message, even with decryption oracle

Chosen Ciphertext-Secure Encryption

Intuitive Definition [DDN,NY,RS]

- Encryption hides message, even with decryption oracle

Why It Matters

- “Correct” security notion for active adversaries
- Real-world attacks on protocols [Bleichenbacher,JKS]

Chosen Ciphertext-Secure Encryption

Intuitive Definition [DDN,NY,RS]

- Encryption hides message, even with decryption oracle

Why It Matters

- “Correct” security notion for active adversaries
- Real-world attacks on protocols [Bleichenbacher,JKS]

Technical Difficulty

- Verify ciphertext is “well-formed”
- Usually via zero-knowledge proof
- Our approach: recover randomness

All-But-One TDFs

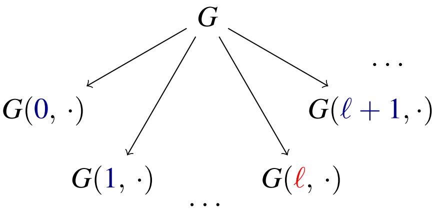
- $G(b, x)$ has extra parameter: *branch* $b \in \{0, 1\}^n$.

All-But-One TDFs

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

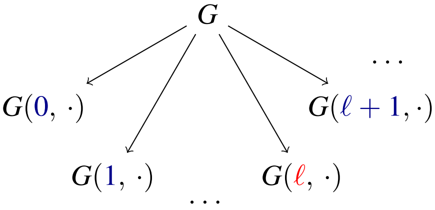
All-But-One TDFs

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



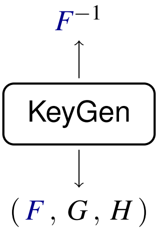
All-But-One TDFs

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

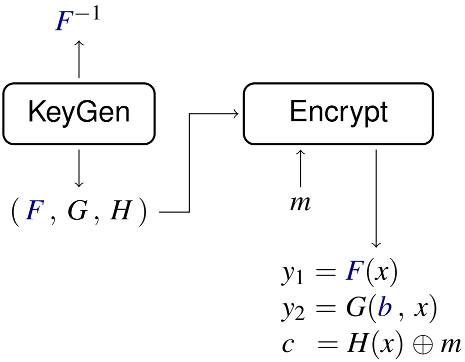


- Lossy TDFs $\Leftrightarrow$ all-but-one TDFs.

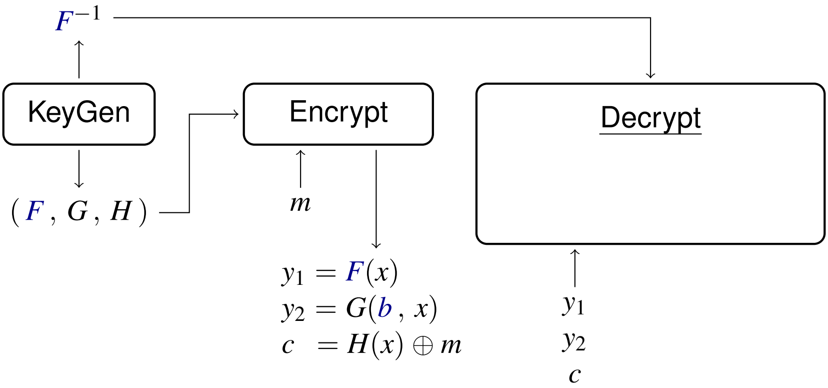
Lossy TDFs ⇒ CCA-Secure Encryption



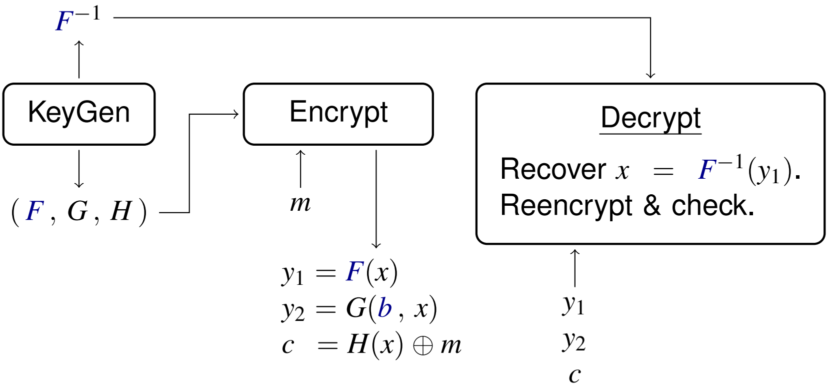
Lossy TDFs ⇒ CCA-Secure Encryption



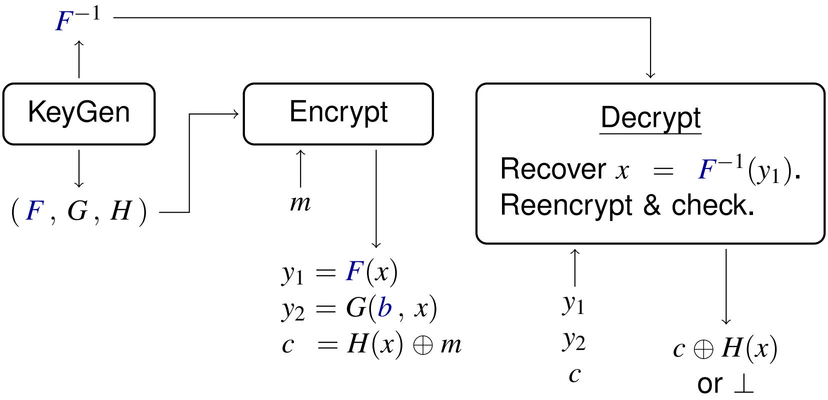
Lossy TDFs ⇒ CCA-Secure Encryption



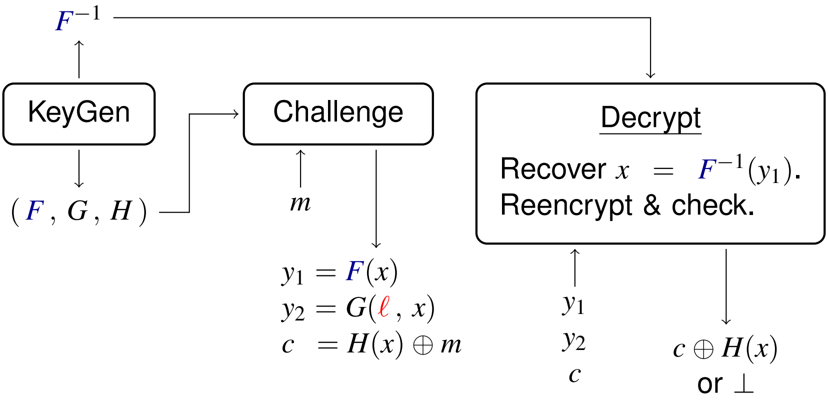
Lossy TDFs ⇒ CCA-Secure Encryption



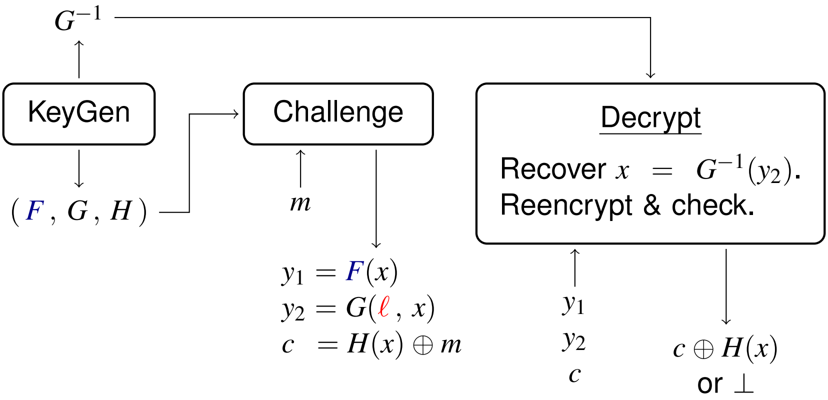
Lossy TDFs $\Rightarrow$ CCA-Secure Encryption



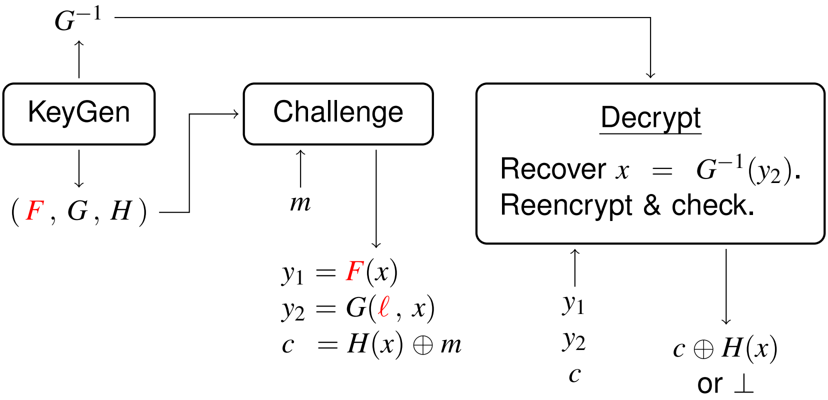
Lossy TDFs ⇒ CCA-Secure Encryption



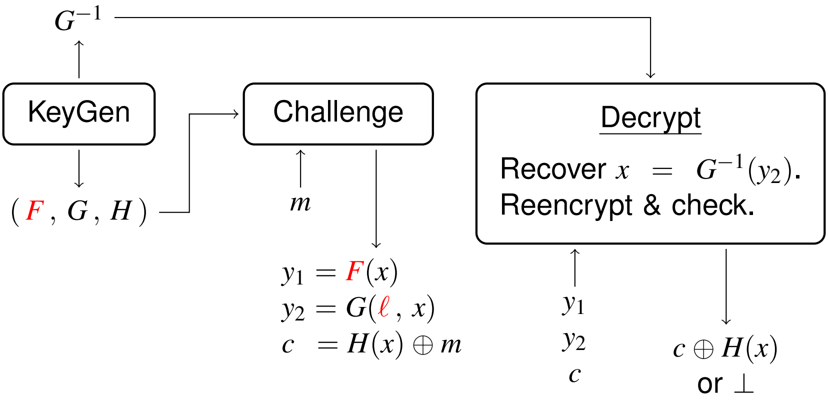
Lossy TDFs ⇒ CCA-Secure Encryption



Lossy TDFs $\Rightarrow$ CCA-Secure Encryption

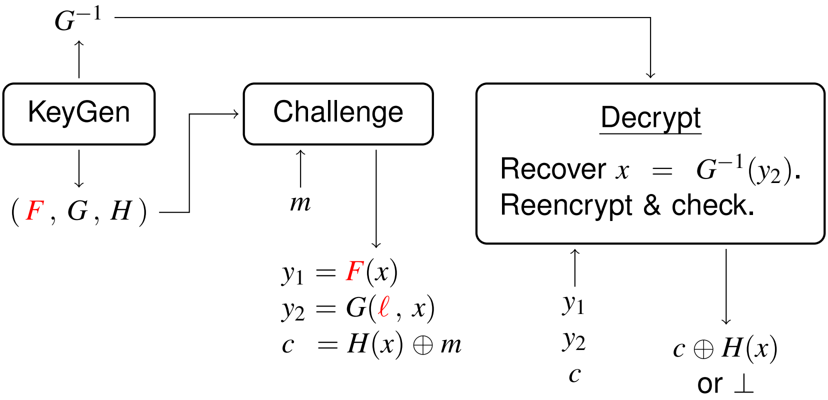


Lossy TDFs ⇒ CCA-Secure Encryption



- Challenge ciphertext hides m statistically.

Lossy TDFs ⇒ CCA-Secure Encryption



- Challenge ciphertext hides m statistically.
- (One-time signature for CCA2 security. [DolevDworkNaor])

Realizing Lossy TDFs

- Use any (additively) homomorphic cryptosystem.

Realizing Lossy TDFs

- Use any (additively) homomorphic cryptosystem.
- Encrypted $n \times n$ matrix: $\mathbf{I}$ for F, $\mathbf{0}$ for F.

  F^{-1} is decryption key.

Realizing Lossy TDFs

- Use any (additively) homomorphic cryptosystem.

- Encrypted $n \times n$ matrix: $\mathbf{I}$ for F, $\mathbf{0}$ for F.

  F^{-1} is decryption key.

- $F(x)$ computed by "encrypted linear algebra."

Realizing Lossy TDFs

- Use any (additively) homomorphic cryptosystem.
- Encrypted $n \times n$ matrix: $\mathbf{I}$ for F, $\mathbf{0}$ for F.

  F^{-1} is decryption key.
- $F(x)$ computed by "encrypted linear algebra."

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

Realizing Lossy TDFs

- Use any (additively) homomorphic cryptosystem.
- Encrypted $n \times n$ matrix: $\mathbf{I}$ for F, $\mathbf{0}$ for F.

  F^{-1} is decryption key.
- $F(x)$ computed by "encrypted linear algebra."

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

Realizing Lossy TDFs

- Use any (additively) homomorphic cryptosystem.
- Encrypted $n \times n$ matrix: $\mathbf{I}$ for F, $\mathbf{0}$ for F.

  F^{-1} is decryption key.
- $F(x)$ computed by "encrypted linear algebra."

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

- Randomness in each 0 leaks information!

Realizing Lossy TDFs (Really)

- Homomorphic cryptosystem with special properties:

Realizing Lossy TDFs (Really)

- Homomorphic cryptosystem with special properties:
  1. Secure to reuse randomness across different keys

Realizing Lossy TDFs (Really)

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

Realizing Lossy TDFs (Really)

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

$$\begin{pmatrix} 0\,;\, r_1 & & \\ 0\,;\, r_1 & & \\ \vdots & & \\ 0\,;\, r_1 & & \end{pmatrix}$$

Realizing Lossy TDFs (Really)

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

$$\begin{pmatrix} 0\,;\,r_1 & 0\,;\,r_2 & & \\ 0\,;\,r_1 & 0\,;\,r_2 & & \\ \vdots & & & \\ 0\,;\,r_1 & 0\,;\,r_2 & & \end{pmatrix}$$

Realizing Lossy TDFs (Really)

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

$$\begin{pmatrix} 0\,;\,r_1 & 0\,;\,r_2 & \cdots & 0\,;\,r_n \\ 0\,;\,r_1 & 0\,;\,r_2 & & 0\,;\,r_n \\ \vdots & & \ddots & \\ 0\,;\,r_1 & 0\,;\,r_2 & & 0\,;\,r_n \end{pmatrix}$$

Realizing Lossy TDFs (Really)

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

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

Realizing Lossy TDFs (Really)

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

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

- Just need $n > |R|$ for lossiness.

Concrete Assumptions

1. Decisional Diffie-Hellman (DDH) on cyclic groups
   - Additive homomorphism in ElGamal: message in the exponent
   - Reusing randomness [NaorReingold,Kurosawa,. . .]

Concrete Assumptions

1. Decisional Diffie-Hellman (DDH) on cyclic groups
   - Additive homomorphism in ElGamal: message in the exponent
   - Reusing randomness [NaorReingold,Kurosawa,. . .]
2. Learning With Errors (LWE) on lattices [Regev]
   - Bounded homomorphism
   - Reuse most randomness — but not the error terms

Future Directions

- Other applications of lossy TDFs (NIZK, PIR, ...?)

Future Directions

- Other applications of lossy TDFs (NIZK, PIR, . . . ?)
- “Natural” trapdoors for lattices [GPV]

Future Directions

- Other applications of lossy TDFs (NIZK, PIR, . . . ?)
- "Natural" trapdoors for lattices [GPV]
- Other indistinguishable properties of "huge" objects?