Zero Knowledge and the Construction of Secure Encryption Schemes

Giuseppe Persiano
giuper@dia.unisa.it

Dipartimento di Informatica ed Appl. "Renato M. Capocelli"
Università di Salerno

May 21, 2008

Outline

Secure Public-Key Encryption Schemes

Chosen Ciphertext Attack

Dynamic Chosen Ciphertext Attack

Public Key Encryption

A Public Key Encryption Schemes for message space $\mathcal{M} = \cup \mathcal{M}_k$ is a triple $(\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ of algorithms:

- the key-generation algorithm KG that takes as input a security parameter 1^k and outputs a public key $\mathtt{pk}$ and a secret key $\mathtt{sk}$;
- the encryption algorithm Enc that takes as input a plaintext $m \in \mathcal{M}_k$ and a public key $\mathtt{pk}$ and returns a ciphertext $\mathtt{ct}$;
- the decryption algorithm Dec that takes as input a ciphertext $\mathtt{ct}$ and a secret key $\mathtt{sk}$ and returns plaintext $m \in \mathcal{M}_k$.

For all $m \in \mathcal{M}_k$,

$$\textbf{Prob}\left[(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathsf{KG}(1^k); \mathtt{ct} \leftarrow \mathsf{Enc}(\mathtt{pk}, m) : \mathsf{Dec}(\mathtt{ct}, \mathtt{sk}) = m\right] = 1.$$

Public Key Encryption

A Public Key Encryption Schemes for message space $\mathcal{M} = \cup \mathcal{M}_k$ is a triple $(\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ of algorithms:

- the key-generation algorithm KG that takes as input a security parameter 1^k and outputs a public key $\mathtt{pk}$ and a secret key $\mathtt{sk}$;
- the encryption algorithm Enc that takes as input a plaintext $m \in \mathcal{M}_k$ and a public key $\mathtt{pk}$ and returns a ciphertext $\mathtt{ct}$;
- the decryption algorithm Dec that takes as input a ciphertext $\mathtt{ct}$ and a secret key $\mathtt{sk}$ and returns plaintext $m \in \mathcal{M}_k$.

For all $m \in \mathcal{M}_k$,

$$\textbf{Prob}\left[(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathsf{KG}(1^k); \mathtt{ct} \leftarrow \mathsf{Enc}(\mathtt{pk}, m) : \mathsf{Dec}(\mathtt{ct}, \mathtt{sk}) = m\right] = 1.$$

Public Key Encryption

A Public Key Encryption Schemes for message space $\mathcal{M} = \cup \mathcal{M}_k$ is a triple $(\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ of algorithms:

- the key-generation algorithm KG that takes as input a security parameter 1^k and outputs a public key $\mathtt{pk}$ and a secret key $\mathtt{sk}$;
- the encryption algorithm Enc that takes as input a plaintext $m \in \mathcal{M}_k$ and a public key $\mathtt{pk}$ and returns a ciphertext $\mathtt{ct}$;
- the decryption algorithm Dec that takes as input a ciphertext $\mathtt{ct}$ and a secret key $\mathtt{sk}$ and returns plaintext $m \in \mathcal{M}_k$.

For all $m \in \mathcal{M}_k$,

$$\textbf{Prob}\left[(\mathrm{pk}, \mathrm{sk}) \leftarrow \mathsf{KG}(1^k); \mathrm{ct} \leftarrow \mathsf{Enc}(\mathrm{pk}, m) : \mathsf{Dec}(\mathrm{ct}, \mathrm{sk}) = m\right] = 1.$$

Public Key Encryption

A Public Key Encryption Schemes for message space $\mathcal{M} = \cup \mathcal{M}_k$ is a triple $(\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ of algorithms:

- the key-generation algorithm KG that takes as input a security parameter 1^k and outputs a public key $\mathtt{pk}$ and a secret key $\mathtt{sk}$;
- the encryption algorithm Enc that takes as input a plaintext $m \in \mathcal{M}_k$ and a public key $\mathtt{pk}$ and returns a ciphertext $\mathtt{ct}$;
- the decryption algorithm Dec that takes as input a ciphertext $\mathtt{ct}$ and a secret key $\mathtt{sk}$ and returns plaintext $m \in \mathcal{M}_k$.

For all $m \in \mathcal{M}_k$,

$$\textbf{Prob}\left[(\mathrm{pk}, \mathrm{sk}) \leftarrow \mathsf{KG}(1^k); \mathrm{ct} \leftarrow \mathsf{Enc}(\mathrm{pk}, m) : \mathsf{Dec}(\mathrm{ct}, \mathrm{sk}) = m\right] = 1.$$

Public Key Encryption

A Public Key Encryption Schemes for message space $\mathcal{M} = \cup \mathcal{M}_k$ is a triple $(\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ of algorithms:

- the key-generation algorithm KG that takes as input a security parameter 1^k and outputs a public key $\mathtt{pk}$ and a secret key $\mathtt{sk}$;
- the encryption algorithm Enc that takes as input a plaintext $m \in \mathcal{M}_k$ and a public key $\mathtt{pk}$ and returns a ciphertext $\mathtt{ct}$;
- the decryption algorithm Dec that takes as input a ciphertext $\mathtt{ct}$ and a secret key $\mathtt{sk}$ and returns plaintext $m \in \mathcal{M}_k$.

For all $m \in \mathcal{M}_k$,

$$\textbf{Prob}\left[(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathsf{KG}(1^k); \mathtt{ct} \leftarrow \mathsf{Enc}(\mathtt{pk}, m) : \mathsf{Dec}(\mathtt{ct}, \mathtt{sk}) = m\right] = 1.$$

Notation for randomized algorithms

$y \leftarrow A(x)$ means

1. pick r at random;
2. run A on input x using r as random tape;
3. assign the result to y;

Sometimes we write $y = A(x; r)$.

$\nu : \mathbb{N} \rightarrow \mathbb{N}$ is negligible if for any poly$(\cdot)$ there exists n_0 such that for $n > n_0$

$$\nu(n) < 1/\text{poly}(n).$$

Notation for randomized algorithms

$y \leftarrow A(x)$ means

1. pick r at random;
2. run A on input x using r as random tape;
3. assign the result to y;

Sometimes we write $y = A(x; r)$.

$\nu : \mathbb{N} \rightarrow \mathbb{N}$ is negligible if for any poly$(\cdot)$ there exists n_0 such that for $n > n_0$

$$\nu(n) < 1/\text{poly}(n).$$

An example: El Gamal Encryption

- KG on input 1^k
  1. randomly selects k-bit prime $p = 2q + 1$ with q prime;
  2. $\mathbb{Z}_p^\star$ is a cyclic group and consider the subgroup $\mathcal{S}_p$ of squares of $\mathbb{Z}_p^\star$ and let g be a generator of $\mathcal{S}_p$;
  3. randomly select $x \leftarrow \mathbb{Z}_q$ and compute $y = g^x$;
  4. output $(\mathtt{pk}, \mathtt{sk}) = ((p, g, y), (x))$;
- Enc on input $\mathtt{pk} = (p, g, y)$ and $m \in \mathcal{S}_p$
  1. randomly select $r \leftarrow \mathbb{Z}_q$;
  2. compute $\mathtt{ct} = (g^r, y^r \cdot m)$;
- Dec on input $\mathtt{sk} = (x)$ and $\mathtt{ct} = (c_0, c_1)$
  1. outputs $c_1 \cdot c_0^{-x}$;

Note: all operations in $\mathbb{Z}_p^\star$.

An example: El Gamal Encryption

- KG on input 1^k
  1. randomly selects k-bit prime $p = 2q + 1$ with q prime;
  2. $\mathbb{Z}_p^\star$ is a cyclic group and consider the subgroup $\mathcal{S}_p$ of squares of $\mathbb{Z}_p^\star$ and let g be a generator of $\mathcal{S}_p$;
  3. randomly select $x \leftarrow \mathbb{Z}_q$ and compute $y = g^x$;
  4. output $(\mathtt{pk}, \mathtt{sk}) = ((p, g, y), (x))$;

- Enc on input $\mathtt{pk} = (p, g, y)$ and $m \in \mathcal{S}_p$
  1. randomly select $r \leftarrow \mathbb{Z}_q$;
  2. compute $\mathtt{ct} = (g^r, y^r \cdot m)$;

- Dec on input $\mathtt{sk} = (x)$ and $\mathtt{ct} = (c_0, c_1)$
  1. outputs $c_1 \cdot c_0^{-x}$;

Note: all operations in $\mathbb{Z}_p^\star$.

An example: El Gamal Encryption

- KG on input 1^k
  1. randomly selects k-bit prime $p = 2q + 1$ with q prime;
  2. $\mathbb{Z}_p^\star$ is a cyclic group and consider the subgroup $\mathcal{S}_p$ of squares of $\mathbb{Z}_p^\star$ and let g be a generator of $\mathcal{S}_p$;
  3. randomly select $x \leftarrow \mathbb{Z}_q$ and compute $y = g^x$;
  4. output $(\texttt{pk}, \texttt{sk}) = ((p, g, y), (x))$;

- Enc on input $\texttt{pk} = (p, g, y)$ and $m \in \mathcal{S}_p$
  1. randomly select $r \leftarrow \mathbb{Z}_q$;
  2. compute $\texttt{ct} = (g^r, y^r \cdot m)$;

- Dec on input $\texttt{sk} = (x)$ and $\texttt{ct} = (c_0, c_1)$
  1. outputs $c_1 \cdot c_0^{-x}$;

Note: all operations in $\mathbb{Z}_p^\star$.

Secure encryption schemes

- it should not be possible to compute m from ct;
- what if from ct I can reconstruct half of the bits of m?
- from ct it should not be possible to compute any of the bits of m;
- what if from ct I can check whether m has more 1's than 0's?
- from ct it should not be possible to compute any predicate on m;

for $m_0, m_1 \in \mathcal{M}_k$, an encryption of m_0 cannot be distinguished from an encryption of m_1.

Secure encryption schemes

- it should not be possible to compute m from `ct`;
- what if from `ct` I can reconstruct half of the bits of m?
- from `ct` it should not be possible to compute any of the bits of m;
- what if from `ct` I can check whether m has more 1's than 0's?
- from `ct` it should not be possible to compute any predicate on m;

for $m_0, m_1 \in \mathcal{M}_k$, an encryption of m_0 cannot be distinguished from an encryption of m_1.

Secure encryption schemes

- it should not be possible to compute m from ct;
- what if from ct I can reconstruct half of the bits of m?
- from ct it should not be possible to compute any of the bits of m;
- what if from ct I can check whether m has more 1's than 0's?
- from ct it should not be possible to compute any predicate on m;

for $m_0, m_1 \in \mathcal{M}_k$, an encryption of m_0 cannot be distinguished from an encryption of m_1.

Secure encryption schemes

- it should not be possible to compute m from ct;
- what if from ct I can reconstruct half of the bits of m?
- from ct it should not be possible to compute any of the bits of m;
- what if from ct I can check whether m has more 1's than 0's?
- from ct it should not be possible to compute any predicate on m;

for $m_0, m_1 \in \mathcal{M}_k$, an encryption of m_0 cannot be distinguished from an encryption of m_1.

Secure encryption schemes

- it should not be possible to compute m from ct;
- what if from ct I can reconstruct half of the bits of m?
- from ct it should not be possible to compute any of the bits of m;
- what if from ct I can check whether m has more 1's than 0's?
- from ct it should not be possible to compute any predicate on m;

for $m_0, m_1 \in \mathcal{M}_k$, an encryption of m_0 cannot be distinguished from an encryption of m_1.

Secure encryption schemes

- it should not be possible to compute m from ct;
- what if from ct I can reconstruct half of the bits of m?
- from ct it should not be possible to compute any of the bits of m;
- what if from ct I can check whether m has more 1's than 0's?
- from ct it should not be possible to compute any predicate on m;

for $m_0, m_1 \in \mathcal{M}_k$, an encryption of m_0 cannot be distinguished from an encryption of m_1.

Security against Chosen Plaintext Attack (aka *Semantic Security*)

An encryption scheme (KG, Enc, Dec) is *secure against chosen plaintext attack* if for all probabilistic polynomial-time adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ we have that for a negligible function ν

$$\left| \textbf{Prob}\left[\mathsf{CPAExp}^{\mathcal{A}}(1^k) = 1 \right] - \frac{1}{2} \right| \leq \nu(k)$$

where

$\mathsf{CPAExp}^{\mathcal{A}}(1^k)$

$(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathsf{KG}(1^k)$

$(m_0, m_1, \mathtt{state}) \leftarrow \mathcal{A}_0(\mathtt{pk})$

pick $b \leftarrow \{0, 1\}$;

$\mathtt{ct} \leftarrow \mathsf{Enc}(\mathtt{pk}, m_b)$;

$b' = \mathcal{A}_1(\mathtt{ct}, \mathtt{pk}, m_0, m_1, \mathtt{state})$;

if $b = b'$ **then return** 1 **else return** 0;

Decisional Diffie Hellman

Let $\mathcal{D}$ be a probabilistic polynomial-time distiguisher.

$\text{DDHExp}_b^{\mathcal{D}}(1^k)$

p random k-bit prime such that $p = 2q + 1$, q prime,
and g generator of $\mathcal{S}_p$;
pick $x, y, z \leftarrow \mathbb{Z}_q$;
if $b = 0$ **then return** $\mathcal{D}(p, g, g^x, g^y, g^{xy})$;
if $b = 1$ **then return** $\mathcal{D}(p, g, g^x, g^y, g^z)$;

Decisional Diffie-Hellman Assumption. For all probabilistic polynomial time distinguishers $\mathcal{D}$ we have that

$$\left|\textbf{Prob}\left[\text{DDHExp}_0^{\mathcal{D}}(1^k) = 1\right] - \textbf{Prob}\left[\text{DDHExp}_1^{\mathcal{D}}(1^k) = 1\right]\right| \leq \nu(k)$$

for a negligible function ν.

CPA Security of ELGamal (under DDH)

Suppose there exists a successful CPA adversary $\mathcal{A}$ for ElGamal. Consider following $\mathcal{D}$.

$\mathcal{D}(p, g, X, Y, Z)$
  run $\mathcal{A}_0$ on public key $\texttt{pk} = (p, g, X)$;
  $\mathcal{A}_0$ ouputs messages (m_0, m_1);
  pick $b \leftarrow \{0, 1\}$ and set $C_1 = Y$ and $C_2 = Z \cdot m_b$;
  run $\mathcal{A}_1$ on (C_1, C_2);
  let b' be $\mathcal{A}_1$'s output;
  if $b = b'$ **then return** 1 **else return** 0;

Chosen Ciphertext Attack (aka *CCA1*)

An encryption scheme (KG, Enc, Dec) is *secure against chosen ciphertext attack* if for all probabilistic polynomial-time adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ we have that for a negligible function ν

$$\left| \textbf{Prob}\left[\mathsf{CCAExp}^{\mathcal{A}}(1^k) = 1\right] - \frac{1}{2} \right| \leq \nu(k)$$

where

$\mathsf{CCAExp}^{\mathcal{A}}(1^k)$

$(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathsf{KG}(1^k)$

$(m_0, m_1) \leftarrow \mathcal{A}_0^{\mathsf{Dec}(\cdot,\mathtt{sk})}(\mathtt{pk})$

pick $b \leftarrow \{0, 1\}$;

$\mathtt{ct} \leftarrow \mathsf{Enc}(\mathtt{pk}, m_b)$;

$b' = \mathcal{A}_1(\mathtt{ct})$;

if $b = b'$ **then return** 1 **else return** 0;

How to Combat CCA

Take a CPA Secure scheme (KG, Enc, Dec) and modify it as follows:

CCAEnc($\mathtt{pk}, m$)
  $\mathtt{ct} \leftarrow$ Enc($\mathtt{pk}, m$)
  Add a "proof" Π that
    "sender knows cleartext associated with $\mathtt{ct}$";

CCADec($\mathtt{sk}, \mathtt{ct} + \Pi$)
  check Π is a correct proof;
  if not then reject;
  if it is then return Dec($\mathtt{sk}, \mathtt{ct}$);

Intuition: Adversary gets decryption only for ciphertexts for which he already knows the cleartext. This should not help!

How to Combat CCA

Take a CPA Secure scheme (KG, Enc, Dec) and modify it as follows:

CCAEnc(pk, m)
    ct $\leftarrow$ Enc(pk, m)
    Add a "proof" Π that
        "sender knows cleartext associated with ct";

CCADec(sk, ct + Π)
    check Π is a correct proof;
    if not then reject;
    if it is then return Dec(sk, ct);

Intuition: Adversary gets decryption only for ciphertexts for which he already knows the cleartext. This should not help!

Desiderata for the proof

1. non-interactive;
2. m is not revealed;

CCAKG(1^k)
  $(\mathrm{pk}, \mathrm{sk}) \leftarrow$ KG(1^k);
  pick a random k-bit string Σ;
  return $((\mathrm{pk}, \Sigma), \mathrm{sk})$;

CCAEnc($(\mathrm{pk}, \Sigma), m$)
  $\mathrm{ct} \leftarrow$ Enc(pk, m)
  Add a "proof" Π computed with respect to Σ
    that "sender knows cleartext associated with ct";

CCADec($\mathrm{sk}, \mathrm{ct} + \Pi$)
  check Π is a correct proof with respect to Σ;
  if not then reject;
  if it is then **return** Dec(sk, ct);

Desiderata for the proof

1. non-interactive;
2. m is not revealed;

CCAKG(1^k)

  $(\mathtt{pk}, \mathtt{sk}) \leftarrow$ KG(1^k);
  pick a random k-bit string Σ;
  return $((\mathtt{pk}, \Sigma), \mathtt{sk})$;

CCAEnc($(\mathtt{pk}, \Sigma), m$)

  $\mathtt{ct} \leftarrow$ Enc($\mathtt{pk}, m$)
  Add a "proof" Π computed with respect to Σ
    that "sender knows cleartext associated with ct";

CCADec($\mathtt{sk}, \mathtt{ct} + \Pi$)

  check Π is a correct proof with respect to Σ;
  if not then reject;
  if it is then **return** Dec($\mathtt{sk}, \mathtt{ct}$);

NIZK for a language L

(P, V, E, S, c) is a NIZK for L

Completeness: $\forall x \in L$:

$$\textbf{Prob}\left[\Sigma \leftarrow \{0,1\}^{n^c}; \Pi \leftarrow P(\Sigma, x, w) : V(\Sigma, x, \Pi) = 1\right] = 1$$

w is a *witness* for $x \in L$.

L is an $\mathbb{NP}$ language.

NIZK for a language L

(P, V, E, S, c) is a NIZK for L

Soundness: $\forall P^\star$:

$$\textbf{Prob}\left[\Sigma \leftarrow \{0,1\}^{n^c}; (x,\Pi) \leftarrow P^\star(\Sigma) : V(\Sigma, x, \Pi) = 1 \wedge x \notin L\right] = \nu(n)$$

The scheme – refined

Consider the language L

$$L = \{(\mathtt{ct}, \mathtt{pk}) : \exists m, r \text{ and } \mathtt{ct} = \mathsf{Enc}(\mathtt{pk}, m; r)\}$$

and let (P, V, E, S, c) a NIZK for L.

$\mathsf{CCAKG}(1^k)$
    $(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathsf{KG}(1^k)$;
    pick a random k-bit string Σ;
    return $((\mathtt{pk}, \Sigma), \mathtt{sk})$;

$\mathsf{CCAEnc}((\mathtt{pk}, \Sigma), m)$
    $\mathtt{ct} = \mathsf{Enc}(\mathtt{pk}, m; r)$;
    $\Pi \leftarrow P(\Sigma, (\mathtt{ct}, \mathtt{pk}), (m, r))$;
    return $(\mathtt{ct}, \Pi)$;

$\mathsf{CCADec}(\mathtt{sk}, (\mathtt{ct}, \Pi))$
    if $V(\Sigma, (\mathtt{ct}, \mathtt{pk}), \Pi) = 0$ **then**
        return $\perp$;
    return $\mathsf{Dec}(\mathtt{sk}, \mathtt{ct})$;

Reduction

Assume existence of an adversary $\mathcal{A}$ that wins the CCAExp game for (CCAKG, CCAEnc, CCADec) with prob. $1/2 + 1/k$ then show existence of adversary $\mathcal{B}$ that wins the CPAExp game for (KG, Enc, Dec) with similar probability.

Idea: $\mathcal{B}$ uses $\mathcal{A}$ as subroutine.

Problem: $\mathcal{A}$ is playing a CCAExp game and it expects to have access to a decryption oracle.
$\mathcal{B}$ does **not** have a decryption oracle since it is playing a CPAExp game.

Solution: Use the proof Π given by the adversary.

Reduction

Assume existence of an adversary $\mathcal{A}$ that wins the CCAExp game for (CCAKG, CCAEnc, CCADec) with prob. $1/2 + 1/k$ then show existence of adversary $\mathcal{B}$ that wins the CPAExp game for (KG, Enc, Dec) with similar probability.

Idea: $\mathcal{B}$ uses $\mathcal{A}$ as subroutine.

Problem: $\mathcal{A}$ is playing a CCAExp game and it expects to have access to a decryption oracle.
$\mathcal{B}$ does **not** have a decryption oracle since it is playing a CPAExp game.

Solution: Use the proof Π given by the adversary.

Reduction

Assume existence of an adversary $\mathcal{A}$ that wins the CCAExp game for (CCAKG, CCAEnc, CCADec) with prob. $1/2 + 1/k$ then show existence of adversary $\mathcal{B}$ that wins the CPAExp game for (KG, Enc, Dec) with similar probability.

Idea: $\mathcal{B}$ uses $\mathcal{A}$ as subroutine.

Problem: $\mathcal{A}$ is playing a CCAExp game and it expects to have access to a decryption oracle.
$\mathcal{B}$ does **not** have a decryption oracle since it is playing a CPAExp game.

Solution: Use the proof Π given by the adversary.

NIZK for a language L: Extraction

Extraction: For all $P^\star$

$$\textbf{Prob}\left[\mathsf{EXTExp}^{P^\star}(1^n) = 1\right] = 1 - \nu(n)$$

where

$\mathsf{EXTExp}^{P^\star}(1^n)$

$(\Sigma, \mathtt{aux}) \leftarrow E_0(1^n)$;

$(x_1, \Pi_1^\star, \cdots, x_l, \Pi_l^\star) \leftarrow P^\star(\Sigma)$;

$(w_1, \cdots, w_l) \leftarrow E_1(\Sigma, \mathtt{aux}, x_1, \Pi_1^\star, \cdots, x_l, \Pi_l^\star)$;

if for some i, w_i not a witness for $x_i \in L$;

and $V(\Sigma, x_i, \Pi_i^\star) = 1$ **then return** 0;

else return 1;

The reduction: 1st try

1. $\mathcal{B}_0$ receives $\texttt{pk}$ in input.
2. Use E_0 to construct $(\Sigma, \texttt{aux})$.
3. Run $\mathcal{A}_0$ on input public key $(\texttt{pk}, \Sigma)$.
4. Whenever $\mathcal{A}_0$ submits $(\texttt{ct}, \Pi)$ for decryption:
   4.1 check Π is valid by running V;
   4.2 if it is, use E_1 and $\texttt{aux}$ to get m.
5. $\mathcal{A}_0$ outputs m_0 and m_1.
6. $\mathcal{B}_0$ outputs m_0 and m_1.
7. $\mathcal{B}_1$ receives $\texttt{ct}^\star$ (encryption of m_0 or m_1).
8. $\mathcal{B}_1$ needs to compute ciphertext for $\mathcal{A}_1$.

Problem: B_1 does not know how to compute a proof it knows message encrypted by $\texttt{ct}$.

The reduction: 1st try

1. $\mathcal{B}_0$ receives $\mathtt{pk}$ in input.
2. Use E_0 to construct $(\Sigma, \mathtt{aux})$.
3. Run $\mathcal{A}_0$ on input public key $(\mathtt{pk}, \Sigma)$.
4. Whenever $\mathcal{A}_0$ submits $(\mathtt{ct}, \Pi)$ for decryption:
   4.1 check Π is valid by running V;
   4.2 if it is, use E_1 and $\mathtt{aux}$ to get m.
5. $\mathcal{A}_0$ outputs m_0 and m_1.
6. $\mathcal{B}_0$ outputs m_0 and m_1.
7. $\mathcal{B}_1$ receives $\mathtt{ct}^\star$ (encryption of m_0 or m_1).
8. $\mathcal{B}_1$ needs to compute ciphertext for $\mathcal{A}_1$.

Problem: B_1 does not know how to compute a proof it knows message encrypted by $\mathtt{ct}$.

NIZK for a language L: Simulation

Simulation: For all PPT $\mathcal{A}$,

$$\left|\textbf{Prob}\left[\mathsf{RZKExp}^{\mathcal{A}}(1^n) = 1\right] - \textbf{Prob}\left[\mathsf{SZKExp}^{\mathcal{A}}(1^n) = 1\right]\right| < \nu(n)$$

where

$\mathsf{RZKExp}^{\mathcal{A}}(1^n)$	$\mathsf{SZKExp}^{\mathcal{A}}(1^n)$
$\Sigma \leftarrow \{0,1\}^{n^c}$;	$(\Sigma, \mathtt{aux}) \leftarrow S_0(1^n)$;
$(x, w) \leftarrow \mathcal{A}_0(\Sigma)$;	$(x, w) \leftarrow \mathcal{A}_0(\Sigma)$;
$\Pi \leftarrow P(\Sigma, x, w)$;	$\Pi \leftarrow S_1(\Sigma, x, \mathtt{aux})$;
return $\mathcal{A}_1(\Sigma, \Pi, x)$;	**return** $\mathcal{A}_1(\Sigma, \Pi, x)$;

The reduction

1. $\mathcal{B}_0$ receives $\mathtt{pk}$ in input.
2. Use E_0 to construct $(\Sigma, \mathtt{aux})$.
3. Run $\mathcal{A}_0$ on input public key $(\mathtt{pk}, \Sigma)$.
4. Whenever $\mathcal{A}_0$ submits $(\mathtt{ct}, \Pi)$ for decryption:
   4.1 check Π is valid by running V;
   4.2 if it is, use E_1 and $\mathtt{aux}$ to get m.
5. $\mathcal{A}_0$ outputs m_0 and m_1.
6. $\mathcal{B}_0$ outputs m_0 and m_1.
7. $\mathcal{B}_1$ receives $\mathtt{ct}^\star$ (encryption of m_0 or m_1).
8. Use S to compute $\Pi^\star$ and submit $(\mathtt{ct}^\star, \Pi^\star)$ to $\mathcal{A}_1$.
9. Receive b' from $\mathcal{A}_1$ and **return** b'.

Building a NIZK for Hamiltonian graphs [FLS]

Good adjacency matrix

Adjacency matrix of a Hamiltonian cycle.

- exactly one 1 in each row;
- exactly one 1 in each column;
- it encodes a cycle;

$$\begin{matrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{matrix}$$

Cycle: $1 \to 2 \to 3 \to 4 \to 1$.

NIZK for HAM

- Prover has the adjacency matrix of graph G and a Hamiltonian Cycle C in G.
- Suppose there is a good adjacency matrix in the sky.
- Each bit is in an envelope
  - Prover can read the bit.
  - Verifier cannot.

Graph G

0	1	0	1
1	0	1	0
0	1	0	1
1	0	0	0

Hidden Cycle H

0	0	1	0
1	0	0	0
0	0	0	1
0	1	0	0

NIZK for HAM

- Prover permutes the entries of G.

Graph G			
0	1	1	0
1	0	0	0
1	0	0	1
0	1	1	0

Hidden Cycle H			
0	0	1	0
1	0	0	0
0	0	0	1
0	1	0	0

- Prover opens all the envelopes corresponding to 0 entries of the permuted graph G.

Graph G			
0	1	1	0
1	0	0	0
1	0	0	1
0	1	1	0

Hidden Cycle H			
0	0	1	0
1	0	0	0
0	0	0	1
0	1	0	0

- Verifier checks all opened entries are 0.

NIZK for HAM

- **Completeness:** Obvious.
- **Soundness:** Obvious.
- **Simulation:** Verifier sees just
  - a random permutation;
  - 0 bits;

NIZK for HAM

- Prover has the adjacency matrix of graph G and a Hamiltonian Cycle C in G.
- Suppose there are n matrices in the sky and at least one is good.
- Each bit is in an envelope
  - Prover can read the bit.
  - Verifier cannot.
- do protocol on the one good matrix;
- open all non-good matrices completely;

NIZK for HAM

- Prover has the adjacency matrix of graph G and a Hamiltonian Cycle C in G.
- Suppose there are n matrices in the sky and at least one is good.
- Each bit is in an envelope
  - Prover can read the bit.
  - Verifier cannot.

- do protocol on the one good matrix;
- open all non-good matrices completely;

NIZK for HAM

Suppose we have mn^4 random bits with $m = \log n^3$.
Each bit is in an envelope

- Prover can read the bit.
- Verifier cannot.

NIZK for HAM

1. Divide the bits in n^4 groups of m bits.
2. For each group define one bit b as

$$b = \begin{cases} 1, & \text{if all bits of the group are 1;} \\ 0, & \text{otherwise;} \end{cases}$$

3. Now we have a $n^2 \times n^2$ matrix B;
4. $$\textbf{Prob}\,[B \text{ has exactly } n \text{ 1s}] > \binom{n^4}{n} \left(\frac{1}{n^3}\right)^n \left(1 - \frac{1}{n^3}\right)^{n^4 - n} > \frac{1}{4\sqrt{n}}.$$
5. Probability they are in distict rows and colums is at least $d > 0$.
6. Probability it is a cycle is $1/n$.

Probability that we have a good adjacency matrix is $\Omega(n^{-3/2})$. If we have mn^6 bits then with very high probability there will be at least one good adjacency matrix.

NIZK for HAM

1. Divide the bits in n^4 groups of m bits.
2. For each group define one bit b as

$$b = \begin{cases} 1, & \text{if all bits of the group are } 1; \\ 0, & \text{otherwise;} \end{cases}$$

3. Now we have a $n^2 \times n^2$ matrix B;
4. $$\textbf{Prob}\,[B \text{ has exactly } n \text{ 1s}] > \binom{n^4}{n} \left(\frac{1}{n^3}\right)^n \left(1 - \frac{1}{n^3}\right)^{n^4 - n} > \frac{1}{4\sqrt{n}}.$$
5. Probability they are in distict rows and colums is at least $d > 0$.
6. Probability it is a cycle is $1/n$.

Probability that we have a good adjacency matrix is $\Omega(n^{-3/2})$. If we have mn^6 bits then with very high probability there will be at least one good adjacency matrix.

NIZK for HAM

1. Divide the bits in n^4 groups of m bits.
2. For each group define one bit b as

$$b = \begin{cases} 1, & \text{if all bits of the group are } 1; \\ 0, & \text{otherwise;} \end{cases}$$

3. Now we have a $n^2 \times n^2$ matrix B;
4. $$\textbf{Prob}\,[B \text{ has exactly } n \text{ 1s}] > \binom{n^4}{n} \left(\frac{1}{n^3}\right)^n \left(1 - \frac{1}{n^3}\right)^{n^4 - n} > \frac{1}{4\sqrt{n}}.$$
5. Probability they are in distict rows and colums is at least $d > 0$.
6. Probability it is a cycle is $1/n$.

Probability that we have a good adjacency matrix is $\Omega(n^{-3/2})$. If we have mn^6 bits then with very high probability there will be at least one good adjacency matrix.

NIZK for HAM

1. Divide the bits in n^4 groups of m bits.
2. For each group define one bit b as

$$b = \begin{cases} 1, & \text{if all bits of the group are } 1; \\ 0, & \text{otherwise}; \end{cases}$$

3. Now we have a $n^2 \times n^2$ matrix B;
4. $$\textbf{Prob}\left[B \text{ has exactly } n \text{ 1s}\right] > \binom{n^4}{n}\left(\frac{1}{n^3}\right)^n\left(1-\frac{1}{n^3}\right)^{n^4-n} > \frac{1}{4\sqrt{n}}.$$
5. Probability they are in distict rows and colums is at least $d > 0$.
6. Probability it is a cycle is $1/n$.

Probability that we have a good adjacency matrix is $\Omega(n^{-3/2})$. If we have mn^6 bits then with very high probability there will be at least one good adjacency matrix.

NIZK for HAM

1. Divide the bits in n^4 groups of m bits.
2. For each group define one bit b as

$$b = \begin{cases} 1, & \text{if all bits of the group are } 1; \\ 0, & \text{otherwise;} \end{cases}$$

3. Now we have a $n^2 \times n^2$ matrix B;
4. $$\textbf{Prob}\left[B \text{ has exactly } n \text{ 1s}\right] > \binom{n^4}{n} \left(\frac{1}{n^3}\right)^n \left(1 - \frac{1}{n^3}\right)^{n^4 - n} > \frac{1}{4\sqrt{n}}.$$
5. Probability they are in distict rows and colums is at least $d > 0$.
6. Probability it is a cycle is $1/n$.

Probability that we have a good adjacency matrix is $\Omega(n^{-3/2})$. If we have mn^6 bits then with very high probability there will be at least one good adjacency matrix.

NIZK for HAM

1. Divide the bits in n^4 groups of m bits.
2. For each group define one bit b as

$$b = \begin{cases} 1, & \text{if all bits of the group are } 1; \\ 0, & \text{otherwise;} \end{cases}$$

3. Now we have a $n^2 \times n^2$ matrix B;
4. $$\textbf{Prob}\left[B \text{ has exactly } n \text{ 1s}\right] > \binom{n^4}{n} \left(\frac{1}{n^3}\right)^n \left(1 - \frac{1}{n^3}\right)^{n^4 - n} > \frac{1}{4\sqrt{n}}.$$
5. Probability they are in distict rows and colums is at least $d > 0$.
6. Probability it is a cycle is $1/n$.

Probability that we have a good adjacency matrix is $\Omega(n^{-3/2})$. If we have mn^6 bits then with very high probability there will be at least one good adjacency matrix.

NIZK for HAM

1. Divide the bits in n^4 groups of m bits.
2. For each group define one bit b as

$$b = \begin{cases} 1, & \text{if all bits of the group are } 1; \\ 0, & \text{otherwise}; \end{cases}$$

3. Now we have a $n^2 \times n^2$ matrix B;
4. $$\textbf{Prob}\left[B \text{ has exactly } n \text{ 1s}\right] > \binom{n^4}{n}\left(\frac{1}{n^3}\right)^n\left(1 - \frac{1}{n^3}\right)^{n^4-n} > \frac{1}{4\sqrt{n}}.$$
5. Probability they are in distict rows and colums is at least $d > 0$.
6. Probability it is a cycle is $1/n$.

Probability that we have a good adjacency matrix is $\Omega(n^{-3/2})$. If we have mn^6 bits then with very high probability there will be at least one good adjacency matrix.

NIZK for HAM

1. Divide the bits in n^4 groups of m bits.
2. For each group define one bit b as

$$b = \begin{cases} 1, & \text{if all bits of the group are 1;} \\ 0, & \text{otherwise;} \end{cases}$$

3. Now we have a $n^2 \times n^2$ matrix B;
4. $$\textbf{Prob}\left[B \text{ has exactly } n \text{ 1s}\right] > \binom{n^4}{n}\left(\frac{1}{n^3}\right)^n\left(1 - \frac{1}{n^3}\right)^{n^4-n} > \frac{1}{4\sqrt{n}}.$$
5. Probability they are in distict rows and colums is at least $d > 0$.
6. Probability it is a cycle is $1/n$.

Probability that we have a good adjacency matrix is $\Omega(n^{-3/2})$. If we have mn^6 bits then with very high probability there will be at least one good adjacency matrix.

NIZK for HAM

Simulation:

1. Generating the random string Σ;
   1.1 Generate n^2 groups of mn^4 bits.
   1.2 If a groups gives a good adjacency matrix replace entries so that it is all zero. Call it special.
2. Receive graph G.
3. Perform proof for each of the n^2 groups.

NIZK for HAM

Simulation:

1. Generating the random string Σ;
   1.1 Generate n^2 groups of mn^4 bits.
   1.2 If a groups gives a good adjacency matrix replace entries so that it is all zero. Call it special.
2. Receive graph G.
3. Perform proof for each of the n^2 groups.

NIZK for HAM

Simulation:

1. Generating the random string Σ;
   1.1 Generate n^2 groups of mn^4 bits.
   1.2 If a groups gives a good adjacency matrix replace entries so that it is all zero. Call it special.
2. Receive graph G.
3. Perform proof for each of the n^2 groups.
   3.1 if group is special then easy;
   if group is not special then

NIZK for HAM

Simulation:

1. Generating the random string Σ;
   1.1 Generate n^2 groups of mn^4 bits.
   1.2 If a groups gives a good adjacency matrix replace entries so that it is all zero. Call it special.
2. Receive graph G.
3. Perform proof for each of the n^2 groups.
   3.1 if group is special then easy;
   3.2 if group is not special then

NIZK for HAM

Simulation:

1. Generating the random string Σ;
   1.1 Generate n^2 groups of mn^4 bits.
   1.2 If a groups gives a good adjacency matrix replace entries so that it is all zero. Call it special.
2. Receive graph G.
3. Perform proof for each of the n^2 groups.
   3.1 if group is special then easy;
   3.2 if group is not special then easy.

NIZK for HAM

Simulation:

1. Generating the random string Σ;
   1.1 Generate n^2 groups of mn^4 bits.
   1.2 If a groups gives a good adjacency matrix replace entries so that it is all zero. Call it special.
2. Receive graph G.
3. Perform proof for each of the n^2 groups.
   3.1 if group is special then easy;
   3.2 if group is not special then easy;

NIZK for HAM

Simulation:

1. Generating the random string Σ;
   1.1 Generate n^2 groups of mn^4 bits.
   1.2 If a groups gives a good adjacency matrix replace entries so that it is all zero. Call it special.
2. Receive graph G.
3. Perform proof for each of the n^2 groups.
   3.1 if group is special then easy;
   3.2 if group is not special then easy;

NIZK for HAM

Simulation:

1. Generating the random string Σ;
   1.1 Generate n^2 groups of mn^4 bits.
   1.2 If a groups gives a good adjacency matrix replace entries so that it is all zero. Call it special.
2. Receive graph G.
3. Perform proof for each of the n^2 groups.
   3.1 if group is special then easy;
   3.2 if group is not special then easy;

NIZK for HAM

Simulation:

1. Generating the random string Σ;
   1.1 Generate n^2 groups of mn^4 bits.
   1.2 If a groups gives a good adjacency matrix replace entries so that it is all zero. Call it special.
2. Receive graph G.
3. Perform proof for each of the n^2 groups.
   3.1 if group is special then easy;
   3.2 if group is not special then easy;

Implementing envelopes

Use encryption

1. Prover picks a $(\mathtt{pk}, \mathtt{sk})$ for CPA secure cryptosystem (El-Gamal);
2. Each sequence of bits is seen as an encryption;
3. Opening a bit corresponds to decryption;

Extraction [DP]

Let (P, V, S) be a NIZK for language

$$L = \{(\mathtt{pk}, \mathtt{ct}, G) | \mathtt{ct} \text{ is an encryption of a hamiltonian cycle for } G\}$$

Extractable NIZK for HAM (NIZKPoK).

- two random strings Σ_1, Σ_2;
- Prover has graph G and hamiltonian cycle C for G;
  - see Σ_1 as public key of cryptosystem;
  - compute $\mathtt{ct} = \mathsf{Enc}(\Sigma_1, C)$;
  - compute proof Π using Σ_2 that $(\Sigma_1, \mathtt{ct}, G) \in L$;
  - return $(\mathtt{ct}, \Pi)$;

The Extractor

- E_0 sets $(\Sigma_1, \mathtt{sk}) \leftarrow \mathsf{KG}(1^n)$ and pick Σ_2 at random;
- E_1 receives $(\mathtt{ct}, \Pi)$;
  - verify Π is valid;
  - use $\mathtt{sk}$ to decrypt $\mathtt{ct}$;

Extraction [DP]

Let (P, V, S) be a NIZK for language

$$L = \{(\mathrm{pk}, \mathtt{ct}, G) | \mathtt{ct} \text{ is an encryption of a hamiltonian cycle for } G\}$$

Extractable NIZK for HAM (NIZKPoK).

- two random strings Σ_1, Σ_2;
- Prover has graph G and hamiltonian cycle C for G;
  - see Σ_1 as public key of cryptosystem;
  - compute $\mathtt{ct} = \mathsf{Enc}(\Sigma_1, C)$;
  - compute proof Π using Σ_2 that $(\Sigma_1, \mathtt{ct}, G) \in L$;
  - return $(\mathtt{ct}, \Pi)$;

The Extractor

- E_0 sets $(\Sigma_1, \mathrm{sk}) \leftarrow \mathsf{KG}(1^n)$ and pick Σ_2 at random;
- E_1 receives $(\mathtt{ct}, \Pi)$;
  - verify Π is valid;
  - use sk to decrypt $\mathtt{ct}$;

Extraction [DP]

Let (P, V, S) be a NIZK for language

$$L = \{(\mathtt{pk}, \mathtt{ct}, G) | \mathtt{ct} \text{ is an encryption of a hamiltonian cycle for } G\}$$

Extractable NIZK for HAM (NIZKPoK).

- two random strings Σ_1, Σ_2;
- Prover has graph G and hamiltonian cycle C for G;
  - see Σ_1 as public key of cryptosystem;
  - compute $\mathtt{ct} = \mathsf{Enc}(\Sigma_1, C)$;
  - compute proof Π using Σ_2 that $(\Sigma_1, \mathtt{ct}, G) \in L$;
  - return $(\mathtt{ct}, \Pi)$;

The Extractor

- E_0 sets $(\Sigma_1, \mathtt{sk}) \leftarrow \mathsf{KG}(1^n)$ and pick Σ_2 at random;
- E_1 receives $(\mathtt{ct}, \Pi)$;
  - verify Π is valid;
  - use $\mathtt{sk}$ to decrypt $\mathtt{ct}$;

Extraction and Simulation on Same String [DP]

Let (P, V, S, E) be any NIZK for language

$$L = \{(\mathtt{pk}, \Sigma_1, \cdots, \Sigma_n, \mathtt{ct}, G) | \mathtt{ct} \text{ is an encryption of } n/2 \text{ valid proofs that } G \text{ has a hamiltonian cycle } \}$$

Dynamic Chosen Ciphertext Attack (aka *CCA2*)

An encryption scheme (KG, Enc, Dec) is *secure against dynamic chosen ciphertext attack* if for all probabilistic polynomial-time adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ we have that for a negligible function ν

$$\left| \textbf{Prob}\left[\mathsf{DCCAExp}^{\mathcal{A}}(1^k) = 1\right] - \frac{1}{2} \right| \leq \nu(k)$$

where

$\mathsf{DCCAExp}^{\mathcal{A}}(1^k)$

$(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathsf{KG}(1^k)$

$(m_0, m_1) \leftarrow \mathcal{A}_0^{\mathsf{Dec}(\cdot,\mathtt{sk})}(\mathtt{pk})$

pick $b \leftarrow \{0, 1\}$;

$\mathtt{ct} \leftarrow \mathsf{Enc}(\mathtt{pk}, m_b)$;

$b' = \mathcal{A}_1^{\mathsf{Dec}^{\neg \mathtt{ct}}(\cdot,\mathtt{sk})}(\mathtt{ct}, \mathtt{pk}, m_0, m_1)$;

if $b = b'$ **then return** 1 **else return** 0;

El Gamal is not DCCA secure

- public key $\mathtt{pk} = (p, g, y)$;
- ciphertext $(c_0, c_1) = (g^r, y^r \cdot m)$;
- pick $s \leftarrow \mathbb{Z}_p$ and compute $(c_0 \cdot g^s, c_1 \cdot y^s)$;

Dynamic Chosen Ciphertext Attack

In the second query phase, $\mathcal{A}$ can ask for decryptions of $\mathtt{ct}^\star \neq \mathtt{ct}$.

Problem: We cannot extract witness from $\Pi^\star$. Adversary might change slighlty $\Pi^\star$, obtain a new valid proof and submit for decryption.

Solution: Require that extraction succeeds even after seeing a "simulated" proof.
See Robust NIZK [DDOPS].

Dynamic Chosen Ciphertext Attack

In the second query phase, $\mathcal{A}$ can ask for decryptions of $\mathtt{ct}^\star \neq \mathtt{ct}$.

Problem: We cannot extract witness from $\Pi^\star$. Adversary might change slighlty $\Pi^\star$, obtain a new valid proof and submit for decryption.

Solution: Require that extraction succeeds even after seeing a "simulated" proof.
See Robust NIZK [DDOPS].

Dynamic Chosen Ciphertext Attack

In the second query phase, $\mathcal{A}$ can ask for decryptions of $\mathtt{ct}^\star \neq \mathtt{ct}$.

Problem: We cannot extract witness from $\Pi^\star$. Adversary might change slighlty $\Pi^\star$, obtain a new valid proof and submit for decryption.

Solution: Require that extraction succeeds even after seeing a "simulated" proof.
See Robust NIZK [DDOPS].

The Double-Encryption Approach [NY]

DCCAKG(1^k)

$(\mathtt{pk}_0, \mathtt{sk}_0) \leftarrow$ KG(1^k);

$(\mathtt{pk}_1, \mathtt{sk}_1) \leftarrow$ KG(1^k);

pick a random k-bit string Σ;

return $((\mathtt{pk}_0, \mathtt{pk}_1, \Sigma), (\mathtt{sk}_0, \Sigma))$;

DCCAEnc($(\mathtt{pk}_0, \mathtt{pk}_1, \Sigma), m$)

$\mathtt{ct}_0 \leftarrow$ Enc($\mathtt{pk}_0, m$);

$\mathtt{ct}_1 \leftarrow$ Enc($\mathtt{pk}_1, m$);

Add a "proof" Π computed with respect to Σ
that $\mathtt{ct}_0$ and $\mathtt{ct}_1$ are encryption of same message;

DCCADec($(\mathtt{sk}_0, \Sigma), (\mathtt{ct}_0, \mathtt{ct}_1, \Pi)$)

check Π is a correct proof with respect to Σ;

if not then reject;

if it is then **return** Dec($\mathtt{sk}_0, \mathtt{ct}_0$);

The scheme – refined

Consider the language L

$$L = \{(\mathtt{ct}_0, \mathtt{ct}_1, \mathtt{pk}_0, \mathtt{pk}_1) : \exists m, r_0, r_1 \text{ and } \mathtt{ct}_0 = \mathsf{Enc}(\mathtt{pk}_0, m; r_0)$$
$$\text{and } \mathtt{ct}_1 = \mathsf{Enc}(\mathtt{pk}_1, m; r_1)\}$$

and let (P, V, S, c) a NIZK for L.

$\mathsf{DCCAKG}(1^k)$
- $(\mathtt{pk}_0, \mathtt{sk}_0) \leftarrow \mathsf{KG}(1^k)$;
- $(\mathtt{pk}_1, \mathtt{sk}_1) \leftarrow \mathsf{KG}(1^k)$;
- pick a random k-bit string Σ;
- **return** $((\mathtt{pk}_0, \mathtt{pk}_1, \Sigma), (\mathtt{sk}_0, \Sigma))$;

$\mathsf{DCCAEnc}((\mathtt{pk}_0, \mathtt{pk}_1, \Sigma), m)$
- $\mathtt{ct}_0 = \mathsf{Enc}(\mathtt{pk}_0, m; r_0)$;
- $\mathtt{ct}_1 = \mathsf{Enc}(\mathtt{pk}_1, m; r_1)$;
- $\Pi \leftarrow P(\Sigma, (\mathtt{ct}_0, \mathtt{ct}_1, \mathtt{pk}_0, \mathtt{pk}_1), (m, r_0, r_1))$;
- **return** $(\mathtt{ct}_0, \mathtt{ct}_1, \Pi)$;

$\mathsf{DCCADec}((\mathtt{sk}_0, \Sigma), (\mathtt{ct}_0, \mathtt{ct}_1, \Pi))$
- **if** $V((\mathtt{ct}_0, \mathtt{ct}_1, \mathtt{pk}_0, \mathtt{pk}_1), \Sigma, \Pi) = 0$ **then**
  - **return** $\perp$;
- **return** $\mathsf{Dec}(\mathtt{sk}_0, \mathtt{ct}_0)$;

Reduction

Assume existence of an adversary $\mathcal{A}$ that wins the DCCAExp game for (DCCAKG, DCCAEnc, DCCADec) with prob. $1/2 + 1/k$ then show existence of adversary $\mathcal{B}$ that wins the CPAExp game for (KG, Enc, Dec) with similar probability.

Idea: $\mathcal{B}$ uses $\mathcal{A}$ as subroutine.

Problem: $\mathcal{A}$ is playing a DCCAExp game and it expects to have access to a decryption oracle.
$\mathcal{B}$ does **not** have a decryption oracle since it is playing a CPAExp game.

Solution: $\mathcal{B}$ has to break one public key. Use the other one to decrypt ciphertexts from $\mathcal{A}$.

Reduction

Assume existence of an adversary $\mathcal{A}$ that wins the DCCAExp game for (DCCAKG, DCCAEnc, DCCADec) with prob. $1/2 + 1/k$ then show existence of adversary $\mathcal{B}$ that wins the CPAExp game for (KG, Enc, Dec) with similar probability.

Idea: $\mathcal{B}$ uses $\mathcal{A}$ as subroutine.

Problem: $\mathcal{A}$ is playing a DCCAExp game and it expects to have access to a decryption oracle.
$\mathcal{B}$ does **not** have a decryption oracle since it is playing a CPAExp game.

Solution: $\mathcal{B}$ has to break one public key. Use the other one to decrypt ciphertexts from $\mathcal{A}$.

Reduction

Assume existence of an adversary $\mathcal{A}$ that wins the DCCAExp game for (DCCAKG, DCCAEnc, DCCADec) with prob. $1/2 + 1/k$ then show existence of adversary $\mathcal{B}$ that wins the CPAExp game for (KG, Enc, Dec) with similar probability.

Idea: $\mathcal{B}$ uses $\mathcal{A}$ as subroutine.

Problem: $\mathcal{A}$ is playing a DCCAExp game and it expects to have access to a decryption oracle.
$\mathcal{B}$ does **not** have a decryption oracle since it is playing a CPAExp game.

Solution: $\mathcal{B}$ has to break one public key. Use the other one to decrypt ciphertexts from $\mathcal{A}$.

The reduction.

1. $\mathcal{B}_0$ receives $\mathtt{pk}$ in input.
2. Use KG to generate $(\mathtt{pk}_1, \mathtt{sk}_1)$.
3. Randomly pick Σ.
4. Run $\mathcal{A}_0$ on input public key $(\mathtt{pk}, \mathtt{pk}_1, \Sigma)$.
5. Whenever $\mathcal{A}_0$ submits $(\mathtt{ct}_0, \mathtt{ct}_1, \Pi)$ for decryption:
   5.1 check Π is valid by running V;
   5.2 if it is, use $\mathtt{sk}_1$ to get m;
6. $\mathcal{A}_0$ outputs m_0 and m_1.
7. $\mathcal{B}_0$ outputs m_0 and m_1.
8. $\mathcal{B}_1$ receives $\mathtt{ct}^\star$ (encryption of m_0 or m_1).
9. $\mathcal{B}_1$ computes $\mathtt{ct}_1 \leftarrow \mathsf{Enc}(\mathtt{pk}_1, m_b)$.
10. Use simulator to produce proof Π.
11. Run $\mathcal{A}_1$ on input $(\mathtt{ct}_0, \mathtt{ct}_1, \Pi)$.
12. Answer queries as in frist phase.

Simulation Soundness

Simulation-Soundness: For all PPT $\mathcal{A}$

$$\textbf{Prob}\left[\mathsf{SSExp}^{\mathcal{A}}(1^n) = 1\right] < \nu(n)$$

where

$\mathsf{SSExp}^{\mathcal{A}}(1^n)$
$(\Sigma, \mathtt{aux}) \leftarrow S_1(1^n)$;
$x \leftarrow \mathcal{A}_1(\Sigma)$;
$\Pi \leftarrow S_2(\Sigma, x, \mathtt{aux})$;
$(x', \Pi') \leftarrow \mathcal{A}_2(\Sigma, \Pi)$;
if $x' \notin L$ and $(x', \Pi') \neq (x, \Pi)$
and $V(x', \Sigma, \Pi') = 1$ **then return** 1;

That's all, Folks!