

Hybrid Encryption, KEM/DEM, and RSA

CS 171: Cryptography

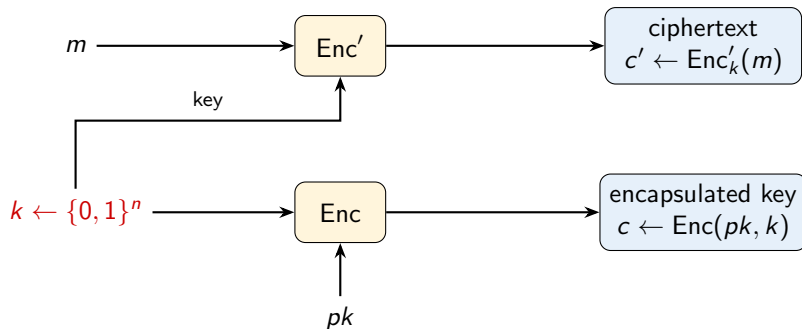
Sanjam Garg

October 20, 2026

Overview

- 1 Hybrid Encryption
- 2 Key Encapsulation Mechanisms (KEM)
- 3 CCA Security and Homomorphism of ElGamal
- 4 RSA Encryption
- 5 An Encryption Combiner
- 6 Summary

Hybrid Encryption: Recall



The promise of hybrid encryption

The **functionality** of public-key encryption at the (asymptotic) **efficiency** of private-key encryption. Decryption is natural: recover k from c , then m from c' .

Hybrid Encryption: More Formally

Construction Π_{hy}

Let $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ be a public-key scheme and $\Pi' = (\text{Gen}', \text{Enc}', \text{Dec}')$ a private-key scheme with keys in $\{0, 1\}^n$.

- $\text{Gen}_{hy} := \text{Gen}$: output (pk, sk) .
- $\text{Enc}_{hy}(pk, m)$:
 - sample $k \leftarrow \{0, 1\}^n$;
 - $c \leftarrow \text{Enc}(pk, k)$ (encapsulate the key)
 - $c' \leftarrow \text{Enc}'_k(m)$ (encrypt the message)
 - output (c, c') .
- $\text{Dec}_{hy}(sk, (c, c'))$: $k := \text{Dec}(sk, c)$, then output $\text{Dec}'_k(c')$.

Security of Hybrid Encryption

Theorem 1 (CPA security)

If Π is CPA-secure and Π' is CPA-secure, then Π_{hy} is CPA-secure.

- In fact, it suffices that Π' is only **EAV-secure**: each key k encrypts a single message.

Theorem 2 (CCA security)

If Π and Π' are CCA-secure, then Π_{hy} is CCA-secure.

CPA Security Proof: Three Hybrids

In $\text{PubK}_{\mathcal{A}, \Pi_{hy}}^{\text{eav}}(n)$, $\mathcal{A}$ gets pk , picks m_0, m_1 , and receives a challenge (c, c') :

	c	c'	
H_0 (real)	$\text{Enc}(pk, k)$	$\text{Enc}'_k(m_b)$	$k \leftarrow \{0, 1\}^n$
H_1	$\text{Enc}(pk, r)$	$\text{Enc}'_k(m_b)$	$k, r \leftarrow \{0, 1\}^n$
H_2	$\text{Enc}(pk, r)$	$\text{Enc}'_k(0^{ m_b })$	$k, r \leftarrow \{0, 1\}^n$

- $H_0 \rightarrow H_1$: the public-key part encrypts an **unrelated** r instead of k (CPA security of Π).
- $H_1 \rightarrow H_2$: now k appears only in c' , so $\text{Enc}'_k(m_b)$ can be swapped for $\text{Enc}'_k(0^{|m_b|})$ (EAV security of Π').
- In H_2 , (c, c') is independent of b : $w_2 = \frac{1}{2}$, where $w_j := \Pr[\mathcal{A} \text{ outputs } b' = b \text{ in } H_j]$.

Each Step Is a Reduction

$\mathcal{B}_1$ plays $\text{PubK}_{\mathcal{B}_1, \Pi}^{\text{eav}}$

Get pk ; give it to $\mathcal{A}$. On m_0, m_1 : sample k, r and submit the challenge pair (k, r) ; get c^* . Sample b ; give $\mathcal{A}$ $(c^*, \text{Enc}'_k(m_b))$. Output 0 if $\mathcal{A}$ outputs $b' = b$, else 1.

$c^* = \text{Enc}(pk, k)$: $\mathcal{A}$ is in H_0 .

$c^* = \text{Enc}(pk, r)$: in H_1 .

$$\begin{aligned}\Pr[\mathcal{B}_1 \text{ wins}] &= \frac{1}{2}w_0 + \frac{1}{2}(1 - w_1) \\ &= \frac{1}{2} + \frac{1}{2}(w_0 - w_1).\end{aligned}$$

$\mathcal{B}_2$ plays $\text{PrivK}_{\mathcal{B}_2, \Pi'}^{\text{eav}}$

Run Gen itself; give pk to $\mathcal{A}$. On m_0, m_1 : sample b, r ; submit $(m_b, 0^{|m_b|})$; get c^* . Give $\mathcal{A}$ $(\text{Enc}(pk, r), c^*)$. Output 0 if $b' = b$, else 1.

$c^* = \text{Enc}'_k(m_b)$: $\mathcal{A}$ is in H_1 .

$c^* = \text{Enc}'_k(0^{|m_b|})$: in H_2 .

$$\Pr[\mathcal{B}_2 \text{ wins}] = \frac{1}{2} + \frac{1}{2}(w_1 - w_2).$$

CPA Security Proof: Putting It Together

Success probability of $\mathcal{A}$

$$\begin{aligned}\Pr [\text{PubK}_{\mathcal{A}, \Pi_{hy}}^{\text{eav}}(n) = 1] &= w_0 = w_2 + (w_0 - w_1) + (w_1 - w_2) \\ &= \frac{1}{2} + 2(\Pr[\mathcal{B}_1 \text{ wins}] - \frac{1}{2}) + 2(\Pr[\mathcal{B}_2 \text{ wins}] - \frac{1}{2}) \\ &\leq \frac{1}{2} + 2 \text{negl}_1(n) + 2 \text{negl}_2(n),\end{aligned}$$

since Π is CPA-secure and Π' is EAV-secure. □

Remark

$\mathcal{B}_2$ uses its challenge once and never needs an encryption oracle for Π' : that is why EAV security of Π' suffices.

ElGamal Hybrid Encryption

The encoding problem

To use ElGamal as the public-key part, the private key $k \in \{0, 1\}^n$ must be encoded as a **group element** — and it is not clear how to do this!

Fix

Rather than encrypting a specific key k , encrypt a **random group element** M , and **derive** the key as

$$k := H(M),$$

where H is a hash / key-derivation function.

Key Encapsulation Mechanism (KEM)

Lesson

We do **not** need a CPA-secure PKE for CPA-secure hybrid encryption: a **key encapsulation mechanism** (KEM) suffices.

Definition 3 (KEM)

(Gen, Encaps, Decaps): Encaps(pk) outputs a ciphertext c and a key k *encapsulated* in c .

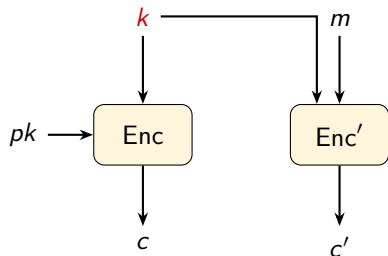
- **Correctness:** Decaps(sk, c) = k .
- **Security:** k is **indistinguishable from uniform** given pk and c .

Game $\text{KEM}_{\mathcal{A}, \Pi}^{\text{cpa}}(n)$

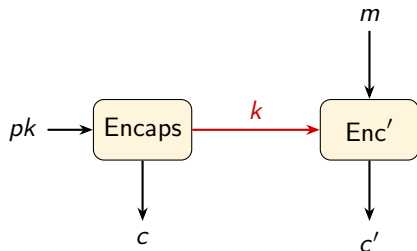
(pk, sk) $\leftarrow$ Gen(1^n), (c, k) $\leftarrow$ Encaps(pk), $b \leftarrow \{0, 1\}$; $\hat{k} := k$ if $b = 0$, uniform if $b = 1$. $\mathcal{A}$ gets $1^n, pk, c, \hat{k}$ and outputs b' . Secure: $\Pr[b' = b] \leq \frac{1}{2} + \text{negl}(n)$. (CCA: add a Decaps oracle, c not allowed.)

Hybrid Encryption (PKE vs. KEM)

Hybrid encryption (PKE + private-key)



KEM/DEM



Difference

In KEM/DEM, the key k is *generated* by $Encaps$ together with c , rather than sampled first and then encrypted under pk .

Theorem 4 (CPA security)

If the KEM Π is CPA-secure and Π' is CPA-secure, then Π_{hy} is CPA-secure.

- In fact, even if Π' is only **EAV-secure**. (Same proof: replace the encapsulated k by a uniform key, then swap m_b for $0^{|m_b|}$.)

Theorem 5 (CCA security)

If the KEM Π and Π' are CCA-secure, then Π_{hy} is CCA-secure.

Efficiency

Short messages: use PKE directly. **Long** messages: use hybrid encryption — this is how things are done in practice.

KEM Based on ElGamal

Gen(1^n)

$(G, q, g) \leftarrow \mathcal{G}(1^n); \quad x \leftarrow \mathbb{Z}_q, \quad h := g^x.$
 $pk = (G, q, g, h), \quad sk = x.$

Encaps(pk)

$r \leftarrow \mathbb{Z}_q; \quad c := g^r, \quad k := H(h^r).$ Output $(c, k).$

Decaps(sk, c)

Output $k := H(c^x).$ ($c^x = g^{rx} = h^r.$)

- KEM security holds based on **DDH**, provided H can extract the randomness in its input.
- Without H : $(pk, c, h^r) = (g^x, g^r, g^{xr})$ is exactly a DH key exchange, and h^r looks like a uniform *group element* (lecture 13) — but we need a uniform *bit string* k .

Is ElGamal Encryption CCA-Secure?

ElGamal ciphertext

$$c_1 = g^r \text{ and } c_2 = m \cdot h^r.$$

Given this ciphertext, construct *another* ciphertext that encrypts the **same** message:

- Sample a uniform $s \leftarrow \mathbb{Z}_q$; set $c'_1 := c_1 \cdot g^s$ and $c'_2 := c_2 \cdot h^s$.
- Then $(c'_1, c'_2) = (g^{r+s}, m \cdot h^{r+s})$: a fresh, valid encryption of m .

No! A CCA attack

$\mathcal{A}$ outputs any $m_0 \neq m_1$, gets $c^* = (c_1, c_2)$, re-randomizes it to $(c'_1, c'_2) \neq c^*$, and queries the decryption oracle: the answer is m_b . $\mathcal{A}$ wins PubK^{cca} with probability 1.

Homomorphic Properties of ElGamal Encryption

Multiplicative homomorphism

Given (c_1, c_2) encrypting m and (c'_1, c'_2) encrypting m' , can we obtain an encryption of $m \cdot m'$?

Yes!

$(c_1 \cdot c'_1, c_2 \cdot c'_2) = (g^{r+r'}, (m \cdot m') \cdot h^{r+r'})$ is an encryption of $m \cdot m'$.

- Any **homomorphic** encryption scheme cannot be CCA-secure: combine the challenge with an encryption of a known message, and ask for the decryption of the result.

RSA Encryption

Plan

- The group $\mathbb{Z}_N^*$ and the **factoring** problem.
- The **RSA** problem and its hardness assumption.
- “Plain” RSA encryption — and why it is insecure.
- A secure variant of RSA encryption.

The Group $\mathbb{Z}_N^*$

- Consider $\mathbb{Z}_N = \{0, \dots, N-1\}$: a group under **addition** modulo N .
- Is it a group under **multiplication** modulo N ?
- **No**: 0 has no inverse, and neither does any b with $\gcd(b, N) > 1$. Consider instead

$$\mathbb{Z}_N^* = \{ b \in \{1, \dots, N-1\} \mid \gcd(b, N) = 1 \},$$

a group under multiplication modulo N , of order $\phi(N) := |\mathbb{Z}_N^*|$.

- We are interested in $N = p \cdot q$ with p, q distinct primes:

$$\phi(N) = (p-1)(q-1).$$

Factoring Problem

- Multiplication can be done in polynomial time; **factoring** in general is believed to be hard.
- It is *not* hard to factor **most** numbers:
 - half of all numbers are even;
 - a third are divisible by 3; and so on.
- Products of **two equal-length primes** are the hardest to factor.

Caveat

The RSA cryptosystem is **not** known to be as hard as factoring.

The RSA Problem

- Let $N = p \cdot q$, where p and q are distinct odd primes.
- Fix an e with $\gcd(e, \phi(N)) = 1$. Then $x \mapsto x^e \bmod N$ is a **permutation** of $\mathbb{Z}_N^*$.
- If $ed = 1 \bmod \phi(N)$, then $(x^e)^d = x \bmod N$:

$$x^{ed} = x^{1+t\phi(N)} = x \cdot (x^{\phi(N)})^t = x,$$

since $x^{|\mathbb{Z}_N^*|} = 1$ (lecture 13). So $x \mapsto x^d$ is the inverse permutation.

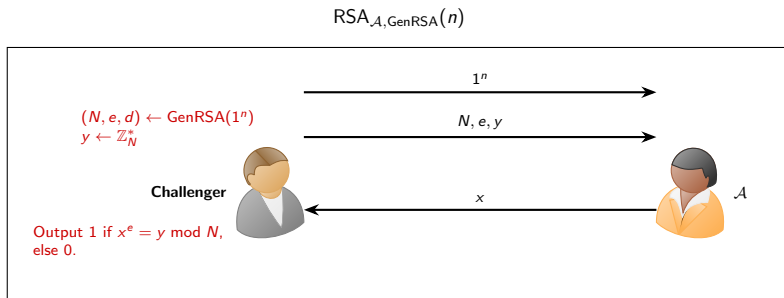
Remark

$\phi(N)$ — and hence d — can be computed if p and q are known!

GenRSA(1^n)

Sample distinct n -bit primes p, q ; $N := pq$; pick e with $\gcd(e, \phi(N)) = 1$;
 $d := e^{-1} \bmod \phi(N)$. Output (N, e, d) .

$\text{RSA}_{\mathcal{A}, \text{GenRSA}}$: Challenger vs. Adversary



Remark

$\mathcal{A}$ gets e but not d : computing e -th roots modulo N is assumed hard without the factors of N . So $x \mapsto x^e \bmod N$ is a candidate one-way *permutation* — with a trapdoor d .

The RSA Problem (Hardness)

Game $\text{RSA}_{\mathcal{A}, \text{GenRSA}}(n)$

- 1 $(N, e, d) \leftarrow \text{GenRSA}(1^n)$, and $y \leftarrow \mathbb{Z}_N^*$ uniform.
- 2 $\mathcal{A}$ is given 1^n and (N, e, y) , and outputs $x \in \mathbb{Z}_N^*$.
- 3 Output 1 if $x^e = y \bmod N$, and 0 otherwise.

Definition 6 (RSA assumption)

The RSA problem is **hard relative to** GenRSA if for all PPT $\mathcal{A}$:

$$\Pr[\text{RSA}_{\mathcal{A}, \text{GenRSA}}(n) = 1] \leq \text{negl}(n).$$

“Plain” RSA Encryption

Construction

- $\text{Gen}(1^n)$: $(N, e, d) \leftarrow \text{GenRSA}(1^n)$; $pk = (N, e)$, $sk = d$.
- $\text{Enc}(pk, m)$, for $m \in \mathbb{Z}_N^*$: output $c := m^e \bmod N$.
- $\text{Dec}(sk, c)$: output $m := c^d \bmod N$.

Correctness: $c^d = m^{ed} = m \bmod N$, by the previous slides.

Is This Secure?

- **CPA-secure?**
- **No**: the scheme is **deterministic**, so it is not CPA-secure: $\mathcal{A}$ computes $m_0^e \bmod N$ itself and compares it with the challenge.
- Even beyond that, c leaks partial information about m .

Warning

Plain RSA should **never** be used.

Secure RSA Encryption

Construction (one-bit messages)

- $\text{Gen}(1^n)$: $(N, e, d) \leftarrow \text{GenRSA}(1^n)$; $pk = (N, e)$, $sk = d$.
 - $\text{Enc}(pk, m)$, for $m \in \{0, 1\}$: sample $r \leftarrow \mathbb{Z}_N^*$; output $c_1 := r^e \bmod N$ and $c_2 := hc(r) \oplus m$, where hc is a Hardness Concentration Bit of the RSA permutation.
 - $\text{Dec}(sk, c)$: output $m := hc(c_1^d \bmod N) \oplus c_2$.
-
- **Why secure?** Given r^e , the bit $hc(r)$ looks uniform (lecture 12), so c_2 is a one-time pad on m .
 - Interestingly, $\text{lsb}(r)$ is a Hardness Concentration Bit for the RSA function.

RSA in Hybrid Encryption: RSA-KEM

KEM from RSA

- Gen: as for RSA, $pk = (N, e)$, $sk = d$; a hash function H .
 - Encaps(pk): sample $r \leftarrow \mathbb{Z}_N^*$; output $c := r^e \bmod N$ and $k := H(r)$.
 - Decaps(sk, c): output $k := H(c^d \bmod N)$.
-
- The deterministic RSA function is fine here: it is applied to a fresh uniform r , never to the message.
 - Given $c = r^e$, computing r is the RSA problem; $k = H(r)$ should then look uniform — provably so if H is modeled as a truly random function (next week).
 - With a DEM for the message, this is the efficient way to encrypt long messages with RSA.

Encryption Combiner

Problem

Given public-key schemes $\Pi_1 = (\text{Gen}_1, \text{Enc}_1, \text{Dec}_1)$ and $\Pi_2 = (\text{Gen}_2, \text{Enc}_2, \text{Dec}_2)$, construct a scheme Π that is CPA-secure as long as **at least one** of Π_1, Π_2 is CPA-secure — without knowing which.

- **Why?** Π_1 and Π_2 may rest on **different hardness assumptions**: Π stays secure if one of the assumptions falls.
 - Done today: browsers' TLS connections combine a classical key exchange (X25519) with a post-quantum one (ML-KEM), and stay secure as long as either one is.
- **First try:** $c = (\text{Enc}_1(pk_1, m), \text{Enc}_2(pk_2, m))$?
- **No:** if Π_1 is insecure, $\mathcal{A}$ attacks c_1 alone and ignores c_2 .
- **Fix:** give each scheme only a **share** of m that is useless on its own.

Combiner: Construction

Construction Π

- $\text{Gen}(1^n)$: $(pk_1, sk_1) \leftarrow \text{Gen}_1(1^n)$ and $(pk_2, sk_2) \leftarrow \text{Gen}_2(1^n)$; $pk = (pk_1, pk_2)$, $sk = (sk_1, sk_2)$.
- $\text{Enc}(pk, m)$: sample $r \leftarrow \{0, 1\}^{|m|}$; output $c = (c_1, c_2)$, where

$$c_1 \leftarrow \text{Enc}_1(pk_1, r) \quad \text{and} \quad c_2 \leftarrow \text{Enc}_2(pk_2, m \oplus r).$$

- $\text{Dec}(sk, c)$: ?? $\rightsquigarrow \text{Dec}_1(sk_1, c_1) \oplus \text{Dec}_2(sk_2, c_2)$, or $\perp$ if either part gives $\perp$.

Correctness

$$\text{Dec}_1(sk_1, c_1) \oplus \text{Dec}_2(sk_2, c_2) = r \oplus (m \oplus r) = m.$$

Combiner: CPA Security

Theorem 7

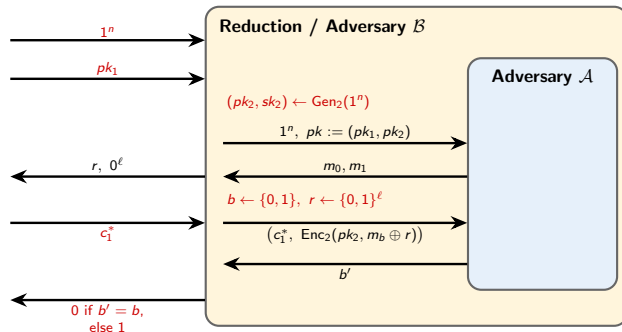
If Π_1 or Π_2 is CPA-secure, then Π is CPA-secure.

Idea: secret sharing

r and $m \oplus r$ are two **shares** of m : each one alone is uniform and says nothing about m .

- If Π_1 is secure, c_1 hides r , so $m \oplus r$ is a one-time pad on m — even if c_2 reveals it completely.
- If Π_2 is secure, c_2 hides $m \oplus r$, and c_1 encrypts an r that is independent of m .
- For public-key schemes, EAV security is CPA security (lecture 14), so we work with PubK^{eav} . Say Π_1 is secure; the other case is symmetric. The proof is a **single** reduction — hybrid encryption needed two.

Combiner Is CPA-Secure: Proof by Reduction



Given / To prove $(\ell = |m_0| = |m_1|)$

Given: $\Pr[\text{PubK}_{\mathcal{A}, \Pi}^{\text{eav}}(n) = 1] \geq \frac{1}{2} + \epsilon(n)$ for non-negligible ϵ .

To prove: $\Pr[\text{PubK}_{\mathcal{B}, \Pi_1}^{\text{eav}}(n) = 1] \geq \frac{1}{2} + \delta(n)$ for non-negligible δ .

Combiner: Two Cases

Case $c_1^* \leftarrow \text{Enc}_1(pk_1, r)$ (real)

$\mathcal{A}$'s view is exactly $\text{PubK}_{\mathcal{A}, \Pi}^{\text{eav}}(n)$:

$$\Pr[\mathcal{B} \text{ correct}] = \Pr[b' = b] \geq \frac{1}{2} + \epsilon(n).$$

$pk = (pk_1, pk_2)$ is distributed as Gen outputs it, and $(\text{Enc}_1(pk_1, r), \text{Enc}_2(pk_2, m_b \oplus r))$ with r uniform is an honest encryption of m_b .

Case $c_1^* \leftarrow \text{Enc}_1(pk_1, 0^\ell)$

$\mathcal{A}$ learns nothing about b :

$$\Pr[\mathcal{B} \text{ correct}] = \Pr[b' \neq b] = \frac{1}{2}.$$

c_1^* does not depend on r , so r is used only in $m_b \oplus r$, which is uniform in $\{0, 1\}^\ell$ for either b — a one-time pad. This holds even if c_2 reveals $m_b \oplus r$ completely.

Combiner: Putting It Together

Success probability of $\mathcal{B}$

$$\begin{aligned}\Pr [\text{PubK}_{\mathcal{B}, \Pi_1}^{\text{eav}}(n) = 1] &= \frac{1}{2} \Pr[\mathcal{B} \text{ correct} \mid c_1^* \text{ encrypts } r] \\ &\quad + \frac{1}{2} \Pr[\mathcal{B} \text{ correct} \mid c_1^* \text{ encrypts } 0^\ell] \\ &\geq \frac{1}{2} \left(\frac{1}{2} + \epsilon(n) \right) + \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{2} + \frac{\epsilon(n)}{2},\end{aligned}$$

contradicting the CPA security of Π_1 . □

If instead Π_2 is CPA-secure

Symmetric: $\mathcal{B}$ gets pk_2 from its challenger, samples (pk_1, sk_1) itself, submits $(m_b \oplus r, 0^\ell)$, and gives $\mathcal{A}$ $(\text{Enc}_1(pk_1, r), c_2^*)$. If c_2^* encrypts 0^ℓ , then $\mathcal{A}$'s view is independent of b .

Remark (running time)

$\mathcal{B}$ runs $\mathcal{A}$ once, plus one run of Gen_2 and one of Enc_2 : PPT.

Is the Combiner CCA-Secure?

Even if Π_1 and Π_2 are **both** CCA-secure? *Try it first: recall how we attacked ElGamal.*

No! A CCA attack

- 1 $\mathcal{A}$ outputs any $m_0 \neq m_1$ of length ℓ , and gets $c^* = (c_1^*, c_2^*)$ with $c_1^* \leftarrow \text{Enc}_1(pk_1, r)$, $c_2^* \leftarrow \text{Enc}_2(pk_2, m_b \oplus r)$.
- 2 It asks to decrypt $(c_1^*, \text{Enc}_2(pk_2, 0^\ell))$ and learns $r \oplus 0^\ell = r$.
- 3 It asks to decrypt $(\text{Enc}_1(pk_1, 0^\ell), c_2^*)$ and learns $0^\ell \oplus m_b \oplus r = m_b \oplus r$.
- 4 The XOR of the two answers is m_b : output the matching b' .

A query equals c^* only if a share is 0^ℓ (probability $2^{-\ell}$ each), so $\mathcal{A}$ wins PubK^{cca} with probability $\geq 1 - 2^{1-\ell}$.

Takeaway

The attack breaks neither Π_1 nor Π_2 : pairing a challenge share with a fresh one gives a valid new ciphertext, and the decryption oracle opens c^* one share at a time — as with ElGamal's re-randomization.

Summary

- **Hybrid encryption:** encrypt a fresh key k under pk , the message under k — PKE functionality at private-key efficiency. *Proved:* CPA-secure if Π is CPA-secure and Π' is only EAV-secure (two hybrid steps).
- **KEM:** Encaps outputs (c, k) with k indistinguishable from uniform (game KEM^{cpa}); $\text{KEM} + \text{DEM}$ is how hybrid encryption is done in practice. ElGamal gives a KEM: $c = g^r$, $k = H(h^r)$.
- **ElGamal** is re-randomizable and multiplicatively homomorphic — hence not CCA-secure.
- **RSA:** inverting $x \mapsto x^e \bmod N$ is assumed hard (game $\text{RSA}_{\mathcal{A}, \text{GenRSA}(n)}$); d is a trapdoor. Plain (deterministic) RSA is insecure; randomize with a Hardness Concentration Bit, or use **RSA-KEM** ($c = r^e$, $k = H(r)$).
- **Combiner:** $(\text{Enc}_1(pk_1, r), \text{Enc}_2(pk_2, m \oplus r))$ is CPA-secure if *either* scheme is (*proved* by a reduction), but not CCA-secure even if both are: decryption queries open the challenge one share at a time.