ElGamal KEM. Now we look at a KEM based on Diffie-Hellman key exchange (and the work of ElGamal, who proposed a public-key encryption scheme based on this key exchange protocol; here we adapt this public-key encryption scheme to the simpler case of a KEM).

Let $G = \langle g \rangle$ be a cyclic group of order $m$ and let $H : \{0,1\}^* \to \{0,1\}^k$ be a hash function. Define $EG$-KEM = $(K,E,D)$ as follows. Algorithm $K$ computes $x \leftarrow \mathbb{Z}_m$, $X \leftarrow g^x$, and returns $(X,x)$. Algorithm $E(X)$ computes $y \leftarrow \mathbb{Z}_m$, $C_a \leftarrow g^y$, $Z \leftarrow X^y$, $K \leftarrow H(C_a||Z)$, and returns $(K,C_a)$. Algorithm $D(x,C_a)$ computes $Z \leftarrow (C_a)^x$, $K \leftarrow H(C_a||Z)$, and returns $K$.

Random oracle model. We analyze this protocol assuming the hash function is “perfect.” This means that $H$ is modeled in the following way. Everyone (all algorithms, adversaries) have access to an oracle $H$ that on input $W$ computes: If $H[W] = \perp$ then $H[W] \leftarrow \{0,1\}^k$; return $H[W]$. Probabilities taken over games now include the coins for this oracle.

Main theorem. We have the following theorem.

Theorem 0.1 Let $A$ be a ROR-adversary making 1 encryption query and $q$ hash queries. Then there is a CDH-adversary $B$ such that

$$\text{Adv}^{\text{ror}}_{\text{EG-KEM}}(A) \leq 2q \cdot \text{Adv}^{\text{cdh}}_{G,g}(B).$$

The running-time of $B$ is that of $A$ plus minor overhead.

Here is an overview of the proof. Assume $A$ doesn’t repeat any hash queries. Define adversary $B$ as follows: On inputs $g^x, g^y$, $B$ chooses $K \leftarrow \{0,1\}^k$, $i^* \leftarrow \mathbb{Z}_q$ and runs $A$ on input $g^x$. When $A$ makes its single encryption query, $B$ returns $(K,g^y)$. When $A$ makes a hash query $W$, $B$ does: $Y||Z \leftarrow W$; If $Y = g^x$ then $i \leftarrow i + 1$, If $i^* = i$ then HALT with output $Z$; $H[W] \leftarrow \{0,1\}^k$ return $H[W]$.

Now we’ll create games $G_0,G_1$ such that we have the following inequalities:

$$1/2 + 1/2\text{Adv}^{\text{ror}}_{\text{EG-KEM}}(A) = \Pr \left[ G_0 \text{ outputs 1} \right] \leq \Pr \left[ G_1 \text{ outputs 1} \right] + \Pr \left[ G_1 \text{ sets bad} \right] \leq 1/2 + 1/q \cdot \text{Adv}^{\text{cdh}}_{G,g}(B).$$

Here are the games:
Games \( G_0 \), \[ G_1 \]

**proc Initialize**

\( x, y \leftarrow Z_m \)

\( K_0, K_1 \leftarrow \{0, 1\}^k \)

\( b \leftarrow \{0, 1\} \)

\( X \leftarrow g^x \)

Return \( X \)

**proc ENC()**

Return \( (K_b, g^y) \)

**proc H(W)**

\( Y \parallel Z \leftarrow W \)

\( H[W] \leftarrow \{0, 1\}^k \)

If \( Z = g^{xy} \) and \( Y = g^y \) then

\( \text{bad} \leftarrow \text{true}; \quad H[W] \leftarrow K_1 \)

Return \( H[W] \)

**proc Finalize(b')**

Return \( (b = b') \)

**RSA.** Let \( K_{rsa} \) be an *RSA parameters generation algorithm* that on input \( 1^k \) outputs \((N, p, q, e, d)\) where \( N = pq \) is a product of \( k/2 \)-bit primes \( p, q \), and \( ed = 1 \mod \phi(N) \). Define the function \( RSA_{N,e}(x) = x^e \mod N \); this is a permutation on \( \mathbb{Z}_N^* \) with inverse \( RSA_{N,e}^{-1}(y) = y^d \mod N \). Informally we want that Given \( N, e, y \) where \( y = RSA_{N,e}(x) \) it’s hard to recover \( x \). Formally for \( K_{rsa} \) we consider the game

Game \( OW_{K_{rsa}} \)

**proc Initialize**

\( (N, p, q, e, d) \leftarrow K_{rsa}(1^k) \)

\( x \leftarrow \mathbb{Z}_N^* ; \quad y \leftarrow RSA_{N,e}(x) \)

Return \( (N, e, y) \)

**proc Finalize(x')**

Return \( (x = x') \)

We define the OW-advantage of adversary \( A \) against \( K_{rsa} \) as

\[
\text{Adv}_{K_{rsa}}^{ow}(A) = \Pr \{ OW_{K_{rsa}}(A) \text{ outputs 1} \}
\]

There are several ways one can consider encrypting with RSA. One is plain RSA (bad). Others are the RSA-KEM and RSA-OAEP, which are IND-CPA secure in the RO model.