Public-Key Encryption. A public-key (aka. asymmetric) encryption scheme with message-space $\mathcal{M}$ is a tuple of algorithms $\text{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ where: $\mathcal{K}$ on input $1^k$ outputs a public key $pk$ and matching secret key $sk$, $\mathcal{E}$ on inputs $pk$ and message $m \in \mathcal{M}$ outputs a ciphertext $c$, and deterministic $\mathcal{D}$ on inputs $sk$ and $c$ outputs $m$ or $\bot$. For correctness we require that for every $k \in \mathbb{N}$ and for all $(pk, sk)$ that can be output by $\mathcal{K}(1^k)$ we have that $\mathcal{D}(\mathcal{E}(pk, m)) = m$ with probability one (over the coins of $\mathcal{E}$).

IND-CPA. The notion of security for public-key encryption is a straightforward modification of that in the symmetric-key setting to take into account the fact that the adversary is given the public key. For a public-key encryption scheme $\text{AE}$ define games LEFT$_{\text{AE}}$ and RIGHT$_{\text{AE}}$ for all $k \in \mathbb{N}$ as follows:

Game LEFT$_{\text{AE}}$

\begin{align*}
\text{proc INITIALIZE} \\
(pk, sk) &\leftarrow \mathcal{K}(1^k) \\
\text{Return } pk \\
\text{proc LR}(m_0, m_1) \\
\text{Return } \mathcal{E}(pk, m_0) \\
\text{proc FINALIZE}(b) \\
\text{Return } (b = 1)
\end{align*}

Game RIGHT$_{\text{AE}}$

\begin{align*}
\text{proc INITIALIZE} \\
(pk, sk) &\leftarrow \mathcal{K}(1^k) \\
\text{Return } pk \\
\text{proc LR}(m_0, m_1) \\
\text{Return } \mathcal{E}(pk, m_1) \\
\text{proc FINALIZE}(b) \\
\text{Return } (b = 1)
\end{align*}

We define the IND-CPA advantage of adversary $A$ against $\text{AE}$ as

\[
\text{Adv}_{\text{AE}}^{\text{ind-cpa}}(A) = \Pr[\text{RIGHT}_{\text{AE}}(A) \text{ outputs } 1] - \Pr[\text{LEFT}_{\text{AE}}(A) \text{ outputs } 1].
\]

As usual the probability is over any coins used in the game as well as those of the adversary.

It turns out that in the public-key setting we can assume the adversary makes one LR query (with some loss in security):

**Theorem 0.1** For any public-key encryption scheme $\text{AE}$ and for any $q$-query adversary $B$ there is a 1-query adversary $A$ such that

\[
\text{Adv}_{\text{AE}}^{\text{ind-cpa}}(B) \leq q \cdot \text{Adv}_{\text{AE}}^{\text{ind-cpa}}(A).
\]

The running-time of $A$ is about that of $B$. 

1
The proof is a “hybrid argument.” Namely, we consider a sequence of hybrid games where we change the responses to the q queries to the LR-oracle by B one-by-one, from encryptions of the left message in the query to encryptions of the right message in the query.

Key encapsulation and its security. Next in order to build PKE we’ll show it suffices to build a “key-encapsulation mechanism” (KEM). A key-encapsulation mechanism (KEM) is a tuple of algorithms KEM = (K, E, D) where: K on input 1^k outputs a public key pk and matching secret key sk, E on inputs pk a key K ∈ {0, 1}^k and encapsulation c, and deterministic D on inputs sk and c outputs K or ⊥. For correctness we require that for every k ∈ N and for all (pk, sk) that can be output by K(1^k) we have that D(c) outputs K with probability one over (K, c) ← E(pk).

For a key encapsulation scheme KEM define games REAL_{KEM} and RAND_{KEM} for all k ∈ N as follows:

Game LEFT_{AE}

\begin{align*}
\text{proc INITIALIZE} \\
(pk, sk) &\leftarrow K(1^k) \\
\text{Return } pk \\
\end{align*}

\begin{align*}
\text{proc ENC}() \\
K_0 &\leftarrow \{0, 1\}^k; (K_1, c) \leftarrow E(pk) \\
\text{Return } (K_1, c) \\
\end{align*}

Game RIGHT_{AE}

\begin{align*}
\text{proc INITIALIZE} \\
(pk, sk) &\leftarrow K(1^k) \\
\text{Return } pk \\
\end{align*}

\begin{align*}
\text{proc ENC}() \\
K_0 &\leftarrow \{0, 1\}^k; (K_1, c) \leftarrow E(pk) \\
\text{Return } (K_0, c) \\
\end{align*}

We define the ROR-advantage of adversary A against KEM as

\[
\text{Adv}_{\text{KEM}}^\text{ror}(A) = \Pr[\text{REAL}_{\text{KEM}}(A) \text{ outputs } 1] - \Pr[\text{RAND}_{\text{KEM}}(A) \text{ outputs } 1].
\]

Hybrid Encryption. We can use a KEM in combination with symmetric-key encryption to obtain PKE. This is called hybrid encryption. In addition to making PKE easier to construct (since we just have to construct a KEM, which is simpler) this approach offers enormous efficiency benefits as it minimizes the use of the public-key part of the scheme. So it is almost always used in practice.

More formally let KEM = (K_a, E_a, D_a) be a KEM and let SE = (K_s, E_s, D_s) be a symmetric-key encryption scheme with message space M. Define the associated public-key encryption scheme HE = (K, E, D) with message space M as follows: Algorithm K on input 1^k outputs (pk, sk) where (pk, sk) ← K_a(1^k). Algorithm E on inputs pk and m ∈ M runs (K_a, c_a) ← E_a(pk) and c_s ← E_s(K, m), and outputs (c_a, c_s). Algorithm D on inputs sk, (c_a, c_s) runs K ← D_a(sk, c_a) and m ← D_s(K, c_s) and outputs m.
Note that the a hybrid encryption scheme is just a public-key encryption scheme! So to analyze its security we just use IND-CPA for public-key encryption as previously defined. In class we stated and proved the associated security theorem. The idea for the proof was to consider a sequence of games in which we change the response given for the one query to the LR-oracle as follows: First we change the encapsulation component of the ciphertext to that of an independent random symmetric key, then we switch from \( m_0 \) to \( m_1 \) in the symmetric-key encryption component, then we switch the encapsulation component of the ciphertext back to an encapsulation of the same key used in the symmetric-key encryption component.