Fully homomorphic encryption scheme using ideal lattices

Gentry’s STOC’09 paper - Part I
Homomorphic encryption

- **KeyGen**: On input $1^\lambda$, outputs a pair of keys, $(pk, sk)$.

- **Encrypt**: On input a public key $pk$ and a plaintext $\pi \in M_{pk}$, outputs a ciphertext $\psi$. We write $\psi \leftarrow \text{Encrypt}(pk, \pi)$. (The plaintext space $M_{pk}$ may depend on $pk$.)

- **Decrypt**: On input a secret key $sk$ and a ciphertext $\psi$, outputs a plaintext $\pi$. We write $\pi \leftarrow \text{Decrypt}(sk, \psi)$.

- **Evaluate**: On input a circuit $C$, public key $pk$, ciphertexts $(\psi_1, \ldots, \psi_t)$, outputs a ciphertext. We write $\psi \leftarrow \text{Evaluate}(pk, C, \psi_1, \ldots, \psi_t)$. 
Correctness

• $\Sigma = (\text{KeyGen, Encrypt, Decrypt, Evaluate}).$

• The scheme $\Sigma$ is correct for circuit $C$ if for any plaintexts $(\pi_1, \ldots, \pi_t)$ and any ciphertexts $(\psi_1, \ldots, \psi_t)$ with $\psi_i \leftarrow \text{Encrypt}(pk, \pi_i)$, it holds that:

$$\psi \leftarrow \text{Evaluate}(pk, C, \psi_1, \ldots, \psi_t)$$

$$\Rightarrow C(\pi_1, \ldots, \pi_t) = \text{Decrypt}(sk, \psi)$$
Compactness

- \( \Sigma = (\text{KeyGen, Encrypt, Decrypt, Evaluate}) \).
- The scheme \( \Sigma \) is **compact** if the output ciphertext of Evaluate is independent (in length) of the input circuit \( C \); more specifically, Decrypt can be expressed as a circuit of size \( \text{poly}(\lambda) \).
- This is to avoid trivial solutions such as:
  - \( \text{Evaluate}(pk, C, \psi_1, \ldots, \psi_t) \) simply returns \( \psi := (C, \psi_1, \ldots, \psi_t) \) as the ciphertext.
  - \( \text{Decrypt}(sk, \psi) \) decrypts each \( \psi_i \) to \( \pi_i \) and computes \( C(\pi_1, \ldots, \pi_t) \).
Fully homomorphic encryption

- $\Sigma = (\text{KeyGen, Encrypt, Decrypt, Evaluate})$.
- $\mathcal{C}$: a class of circuits (including the identity circuit).
- $\Sigma$ is $\mathcal{C}$-homomorphic if $\Sigma$ is correct and compact for every circuit in $\mathcal{C}$.
- $\Sigma$ is somewhat homomorphic if it is $\mathcal{C}$-homomorphic for some set of circuits $\mathcal{C}$.
- $\Sigma$ is fully homomorphic if it is homomorphic for all circuits (i.e., $\mathcal{C}$-homomorphic for the set of all circuits $\mathcal{C}$).
Leveled fully homomorphic encryption

- $\Sigma^{(d)} = \left(\text{KeyGen}^{(d)}, \text{Encrypt}^{(d)}, \text{Decrypt}^{(d)}, \text{Evaluate}^{(d)}\right)$.

- A family of schemes $\{\Sigma^{(d)} : d \in \mathbb{Z}^+\}$ is said to be leveled fully homomorphic iff:
  - all schemes $\Sigma^{(d)}$ use the same decryption circuit,
  - $\Sigma^{(d)}$ is homomorphic for all circuits of depth up to $d$ (that use some specified set of gates),
  - the computational complexity of $\Sigma^{(d)}$'s algorithms is polynomial in $\lambda$, $d$, and (in the case of $\text{Evaluate}^{(d)}$) the size of $C$. 
Homomorphic encryption before Gentry

- The concept of fully homomorphic encryption, originally called privacy homomorphism, was proposed by Rivest, Adleman and Dertouzos in 1978 (one year after RSA was published).

- Homomorphic encryption schemes before 2009:
  - **Multiplicatively homomorphic**: RSA, ElGamal, etc.
  - **Additively homomorphic**: Goldwasser-Micali, Paillier, etc.
  - **Quadratic polynomials**: Boneh-Goh-Nissim
  - **Arbitrary circuits but with exponential ciphertext-size**: "Polly Craker" by Fellows and Koblitz
  - **NC^1 circuits** (poly-size, depth $O(\log n)$, using bounded fan-in AND, OR, and NOT gates): Sanders-Young-Yung
In 2009, Gentry proposed the first FHE scheme.

Three steps:
- Building a somewhat homomorphic encryption scheme using ideal lattices
- Squashing the Decryption Circuit
- Bootstrapping
Bootstrapping
Why does SH not imply FH?

- \{\text{AND, } \text{XOR}\}, \text{i.e., } \{+, \times\}, \text{ is a complete set of gates, from which any Boolean function can be constructed.}

- **False**: If an encryption scheme is \{+, \times\}-homomorphic, then it is fully homomorphic.

- **Reason**: Ciphertexts typically contain an "error" or "noise". When operations are performed on ciphertexts, errors grow. When the error becomes too large, the ciphertext cannot be correctly decrypted.
Example

- **Key:** a large odd integer $p$.

- **Encryp($p, m$):** To encrypt a bit $m \in \{0, 1\}$, let $c = pq + 2r + m$, where $q$, $r$ are random with $0 \leq 2r \ll p$. 2$r$ is the noise.

- **Decryp($p, c$):** let $m = (c \mod p) \mod 2$.

- If $c_1 = pq_1 + 2r_1 + m_1$ and $c_2 = pq_2 + 2r_2 + m_2$, then $c_1 + c_2$ is a ciphertext of $m_1 + m_2$, with noise $2(r_1 + r_2)$, and $c_1c_2$ is a ciphertext of $m_1m_2$, with noise $2(2r_1r_2 + r_1m_2 + m_1r_2)$.

- The noise grows!

- What if the noise becomes too large, say $2r > p$?
Challenge

• Can we have a \( \{+, \times\} \)-homomorphic encryption scheme without noises growing?

• That is, the ciphertexts output by Evaluate is as fresh as those output by Encrypt (in terms of amount of noise).

• Such a scheme will automatically be fully homomorphic.

• Gentry proposed a simple yet powerful strategy to achieve that (no noise growing): **Bootstrapping!**
Bootstrapping

• In a nut shell, bootstrapping is to perform (augmented) Decrypt homomorphically.
If we can evaluate decrypt homomorphically

- We can allow anyone to convert a ciphertext under key $pk_A$ into a ciphertext under key $pk_B$ w/o revealing the message.

**Pink box:**
- encrypted under $pk_A$.

**Blue box:**
- encrypted under $pk_B$.

May use WeakEncrypt
$g$-augmented decryption circuit

- $g$: a gate (with input and output in the plaintext space).
- $g$-augmented decryption circuit: illustrated below.

NAND-augmented Decrypt:

$c_1, c_2$ are ciphertexts of $m_1, m_2$ under key $pk_A$
If we can evaluate NAND-Decrypt homomorphically

- Encrypt all input using $pk_B$ (figuratively, put them in a blue box).
- Evaluate NAND-Decrypt.
- We obtain a "fresh" ciphertext of $m_1 \text{ NAND } m_2$ under key $pk_B$. 

![Diagram showing encryption and decryption process with a NAND gate]
If we can evaluate NAND-Decrypt homomorphically...

- then from the ciphertexts of $m_1$ and $m_2$ under $pk_A$, we can obtain a "fresh" ciphertext of $m_1 \text{ NAND } m_2$ under key $pk_B$, provided that the encryption of $sk_A$ under $pk_B$ is given.

- That is, we can perform $m_1 \text{ NAND } m_2$ homomorphically without increasing the noise.
Suppose we want to evaluate this circuit homomorphically, with $m_1, m_2, m_3, m_4$ encrypted under $\text{pk}_A$. Evaluate($C, \text{pk}_A, \psi_1, \psi_2, \psi_3, \psi_4$).
Evaluate Decrypt-NAND

$\text{skA}$

$m_1$ NAND $m_2$ NAND $m_3$ NAND $m_4$

Evaluate Decrypt-NAND

$\text{skB}$

$\text{Evaluate}$

$(m_1 \text{ NAND } m_2) \text{ NAND } (m_3 \text{ NAND } m_4)$
Bootstrappable encryption

- $\Sigma = (\text{KeyGen, Encrypt, Decrypt, Evaluate})$.
- $\Gamma$: a set of gates (with input/output in the plaintext space).
- $D_{\Sigma}(\Gamma)$: the set of $g$-augmented Decrypt, $g \in \Gamma$.
- $\mathcal{C}$: a class of circuits (including the identity circuit).
- Suppose $\Sigma$ is $\mathcal{C}$-homomorphic.
- $\Sigma$ is said to be bootstrappable with respect to $\Gamma$ if $D_{\Sigma}(\Gamma) \subseteq \mathcal{C}$.
- If $\Sigma$ is bootstrappable w.r.t. a complete set of gates $\Gamma$ (including the identity gate), then we can construct a leveled fully homomorphic family of schemes $\{\Sigma^{(d)} : d \in \mathbb{Z}^+\}$ (for circuits with gates in $\Gamma$).
\[ \Sigma^{(d)} : \text{homomorphic for circuits of depth} \leq d \]

- Assume \( \Sigma = (\text{KeyGen}, \text{Encrypt}, \text{Decrypt}, \text{Evaluate}) \) is bootstrappable w.r.t. a set of gates \( \Gamma \). We construct from \( \Sigma \)
  \[ \Sigma^{(d)} = (\text{KeyGen}^{(d)}, \text{Encrypt}^{(d)}, \text{Decrypt}^{(d)}, \text{Evaluate}^{(d)}) \].

- \( \text{KeyGen}^{(d)} (\lambda, d) : \) //The same algorithm for all \( d \)//
  - Use \( \text{KeyGen} \) to generate \( d + 1 \) key pairs \( (sk_i, pk_i) \), \( 0 \leq i \leq d \).
  - Represent \( sk_i \) as a sequence of plaintexts: \( sk_i = (sk_{i1}, \ldots, sk_{i\ell}) \).
  - Encrypt (each element of) \( sk_i : \overline{sk_i} \leftarrow \text{Encrypt} (pk_{i-1}, sk_i) \).
  - Secret key: \( sk^{(d)} = sk_0 \).
  - Public key: \( pk^{(d)} = \left\{ \langle pk_i \rangle_{0 \leq i \leq d}, \langle \overline{sk_i} \rangle_{1 \leq i \leq d} \right\} \).
The rest are the evaluation key
- **Encrypt**\(^{(d)}\): 
  - Input: a public key \( pk^{(d)} \) and a plaintext \( \pi \).
  - Output: ciphertext \( \psi \leftarrow \text{Encrypt}(pk^{(d)}, \pi) \).

- **Decrypt**\(^{(d)}\): 
  - Input: a secret key \( sk^{(d)} \) and a ciphertext \( \psi \).
  - Output: ciphertext \( \pi \leftarrow \text{Decrypt}(sk^{(d)}, \psi) \).
  - Remark: \( \psi \) is assumed to be an output of \( \text{Evaluate}^{(d)} \).

What if \( \psi \) was produced by \( \text{Encrypt}^{(d)} \)?
Recursive procedure:

- **Evaluate**$^{(d)}(pk^{(d)}, C_d, \Psi_d)$:
  - Recursive procedure: **Evaluate**$^{(\delta)}(pk^{(\delta)}, C_\delta, \Psi_\delta)$.
  - $C_\delta$ has exactly $\delta$ levels; gates at level $i$ are connected to gates at level $i - 1$. (Any circuit of depth $\leq \delta$ can be converted to such a circuit by inserting identity gates.)
  - $\Psi_\delta$ is a tuple of ciphertexts under $pk_\delta$.
  - Initial call: **Evaluate**$^{(d)}(pk^{(d)}, C_d, \Psi_d)$. 
Evaluate\(^{(\delta)}\) \((pk^{(\delta)}, C_\delta, \Psi_\delta)\)

\[\Psi_\delta \Rightarrow \text{under } pk_\delta\]

\[C_\delta\]
Evaluate$^{(δ)} \left( pk^{(δ)}, C_δ, Ψ_δ \right)$

\[ \begin{align*}
\Psi_δ \\
sk_δ
\end{align*} \] \[ \Rightarrow \]

$C_δ$ augmented with decryption circuits
Evaluate\(^{(\delta)}\) \(\left( pk^{(\delta)}, C_\delta, \Psi_\delta \right) \)

\[
\begin{align*}
&\Psi_\delta, sk_\delta \\
\Rightarrow &\text{Decrypt circuits} \\
\Rightarrow &\Psi_{\delta-1}
\end{align*}
\]

\(\Psi_\delta, sk_\delta\) encrypted under \(pk_{\delta-1}\)

\(C_{\delta-1}\)

level \(\delta-1\)  
level 1
Call Evaluate\(^{(\delta-1)}\left(pk^{(\delta-1)}, C_{\delta-1}, \Psi_{\delta-1}\right)\)

\[\Psi_{\delta-1} \Rightarrow \text{under } pk_{\delta-1}\]
Evaluate\(^{(0)}\) \(\left( pk^{(0)}, C_0, \Psi_0 \right) \)

When \( \delta = 0 \), simply return \( \Psi_0 \),

which is under \( pk_0 \) and can be decrypted with \( sk^{(d)} = sk_0 \).
Correctness

- Theorem. If $\Sigma$ is bootstrappable w.r.t. a complete set of gates $\Gamma$ (including the identity gate), then the family $\left\{ \Sigma^{(d)} : d \in \mathbb{Z}^+ \right\}$ constructed above is leveled fully homomorphic (for circuits with gates in $\Gamma$).
- That is, $\text{Decrypt}^{(d)}$ correctly evaluate any circuit (composed of gates in $\Gamma$) of depth at most $d$. 

Complexity

- **Theorem.** For a circuit $C$ of depth $d$ and size $s$ (the number of wires), the time complexity of evaluating $C$ is dominated by $O(s \cdot l)$ applications of $\text{Encrypt}$ and $O(s)$ applications of $\text{Evaluate}$ to $(g \in \Gamma)$-augmented decryption circuits, where $l = \ell(\lambda)$ is the number of "bits" of each ciphertext and sk.

- **Remark:** If the given circuit $C$ has depth $< d$ and size $s$, it can be converted into a circuit of depth $d$ and size at most $sd$.

- **Theorem.** For a circuit $C$ of depth $\leq d$ and size $s$ (the number of wires), the time complexity of evaluating $C$ is dominated by $O(s \cdot l \cdot d)$ applications of $\text{Encrypt}$ and $O(s \cdot d)$ applications of $\text{Evaluate}$ to $(g \in \Gamma)$-augmented decryption circuits.
Theorem. If \( \Sigma \) is semantically secure, then 
\[ \Sigma^{(d)} \] is semantically secure for each \( d \).

Two questions:

- What's the meaning of semantic security for homomorphic encryption schemes?
- How to prove the theorem?
Semantic security game for public-key encryption

- **Challenger:** on input the security parameter $\lambda$,
  - generates a key pair $(pk, sk)$,
  - sends $pk$ to the adversary.

- **Adversary:** produces two messages $m_0$, $m_1$, and sends them to the challenger.

- **Challenger:** chooses a random bit $b \leftarrow \{0, 1\}$ and sends $c \leftarrow Enc_{pk}(m_b)$ to the adversary.

- **Adversary:** determines whether $b = 0$ or $b = 1$.

**Question:** Does this model apply to homomorphic encryption?
Semantic security for homomorphic encryption

- Is it different from that for ordinary public-key encryption? We will argue that it is the same.
- Since ciphertexts may be produced by Evaluate, a natural modification to the model is to let the adversary provide a circuit $C$ and two inputs $m_0 = (m_{01}, \ldots, m_{0t})$, $m_1 = (m_{11}, \ldots, m_{1t})$.
- The challenger chooses $b \leftarrow \{0,1\}$, encrypts $m_b$ as $\psi$, runs $\psi \leftarrow \text{Evaluate}(pk, C, \psi)$, and gives $\psi$ to the adversary as the challenge ciphertext.
- The challenger may simply give $\psi$ as the challenge ciphertext, since the adversary can run $\psi \leftarrow \text{Evaluate}(pk, C, \psi)$ itself.
• So, the semantic security game for homomorphic encryption is the same as the **multi-ciphertext** semantic security game for ordinary public-key encryption.

• It has been shown that an algorithm A that breaks the semantic security of the game with multiple ciphertexts can be used to construct an algorithm B that breaks the semantic security of the ordinary game. That is, breaking single-ciphertext semantic security $\leq$ breaking multi-ciphertext semantic security.

• Therefore, to prove semantic security of a homomorphic encryption scheme, we can just use the semantic game for ordinary public-key encryption.
Why is it not trivial?

- Theorem. If $\Sigma$ is semantically secure (and bootstrappable), then $\Sigma^{(d)}$ is semantically secure for each $d$.

$$pk_d \quad pk_{d-1} \quad \cdots \quad pk_1 \quad pk_0$$

$$sk_d \quad sk_{d-1} \quad \cdots \quad sk_1 \quad sk_0$$

These encrypted keys $sk_i$ might leak information about the ciphertext (under $pk_d$), unless we prove otherwise.
Semantic Security Game $k$, $d \geq k \geq 0$.

- Game $k$ is the same as the game for $\Sigma^{(d)}$ except that each $sk_i$, $d \geq i \geq 1$, is replaced by some $sk_i'$ unrelated to $pk_i$:
  - $(sk_i', pk_i') \leftarrow \text{KeyGen}(1^\lambda)$
  - $sk_i' \leftarrow$ encryption of $sk'$ under $pk_{i-1}$

- Game $d = \text{game for } \Sigma$. Game $0 = \text{game for } \Sigma^{(d)}$.

\[
\begin{array}{ccccccc}
  & pk_d & \cdots & pk_k & \cdots & pk_1 & pk_0 \\
\hline
  sk_d & \cdots & sk_k' & \cdots & sk_1' & sk_0
\end{array}
\]
To prove the theorem, assume the existence of an adversary $A$ that has a non-negligible advantage against $\Sigma^{(d)}$ (Game 0). We construct an algorithm $B$ that breaks $\Sigma$ (Game $d$) with a non-negligible advantage. ($B$ will use $A$ as a "subroutine").

Let $\varepsilon_k(\lambda) = A$'s advantage in Game $k$. Apparently, $\varepsilon_d(\lambda) \leq \varepsilon_{d-1}(\lambda) \leq \cdots \leq \varepsilon_0(\lambda)$.

Two cases:

- $\varepsilon_d(\lambda)$ is non-negligible ($A$ breaks $\Sigma$ and we are done).
- $\varepsilon_d(\lambda)$ is negligible.

Assume $\varepsilon_d(\lambda)$ is negligible. There must exist a $d > k \geq 0$ such that $\varepsilon_k(\lambda)$ is non-negligible and $\varepsilon_{k+1}(\lambda)$ is negligible.

Fix this $k$ and consider Games $k$ and $k + 1$. 
• $\varepsilon_k(\lambda)$ is non-negligible and $\varepsilon_{k+1}(\lambda)$ is negligible.

\[
pk_d \; \ldots \; pk_{k+1} \; pk_k \; \ldots \; pk_0
\]

\[
\overline{sk_d} \; \ldots \; \overline{sk_{k+1}} \; \overline{sk_k} \; \ldots \; sk_0
\]

insecure against $A$, but secure if $\overline{sk_{k+1}}$ is replaced by $\overline{sk'_{k+1}}$.

So, $A$ can help us distinguish between $\overline{sk_{k+1}}$ and $\overline{sk'_{k+1}}$.

• Three players, two games:

\[
\begin{array}{ccc}
\text{Game against $\Sigma$} & \text{Game against $\Sigma^{(d)}$} \\
C \text{ (challenger)} & B \text{ (challenger)} & A \text{ (adversary)} \\
\text{(adversary)} & \text{(challenger)} & \text{(adversary)}
\end{array}
\]

• Remark: between $B$ and $C$ is a multi-ciphertext game.
• $\varepsilon_k(\lambda)$ is non-negligible and $\varepsilon_{k+1}(\lambda)$ is negligible.

\[
pk_d \ldots \pk_{k+1} \pk \ldots \pk_0
\]
\[
sk_d \ldots \psi \sk_k \ldots \sk_0
\]

\[\downarrow \text{insecure if } \psi = \sk_{k+1}\]

secure if $\psi = \sk'_{k+1}$.

$A$ can help us distinguish between $\sk_{k+1}$ and $\sk'_{k+1}$. 
Game against $\Sigma$

<table>
<thead>
<tr>
<th>C (challenger)</th>
<th>B (adversary)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. generate $pk$, $sk$;</td>
<td>5. send $\pi_0 = sk_{k+1}$, $\pi_1 = sk'_{k+1}$ to C;</td>
</tr>
<tr>
<td>2. send $pk$ to B;</td>
<td>8. (B is to guess $b$, with A's help);</td>
</tr>
<tr>
<td>6. choose $b$;</td>
<td>14. if $\beta = \beta'$ then $b' = 0$ else $b' = 1$;</td>
</tr>
<tr>
<td>7. send $\psi \leftarrow E_{pk}(\pi_b)$ to B;</td>
<td>15. send $b'$ to C.</td>
</tr>
</tbody>
</table>

Game against $\Sigma^{(d)}$

<table>
<thead>
<tr>
<th>B (challenger)</th>
<th>A (adversary)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3. set up the game with A;</td>
<td>11. send plaintexts $\pi'_0$, $\pi'_1$ to B;</td>
</tr>
<tr>
<td>4. replace $pk_k$ by $pk$;</td>
<td>13. send its guess $\beta'$ to B;</td>
</tr>
<tr>
<td>9. replace $sk_{k+1}$ by $\psi$;</td>
<td>12. choose $\beta$ and send $\psi' \leftarrow E_{pk_d}(\pi'_\beta)$ to A;</td>
</tr>
<tr>
<td>10. send the &quot;keys&quot; to A;</td>
<td></td>
</tr>
<tr>
<td>12. choose $\beta$ and send $\psi' \leftarrow E_{pk_d}(\pi'_\beta)$ to A;</td>
<td></td>
</tr>
</tbody>
</table>
• In summary, if $A$ has a non-negligible advantage against $\Sigma^{(d)}$, then $B$ has a non-negligible advantage against the multi-ciphertext version of $\Sigma$, from which one can construct an algorithm $B'$ against (the single-ciphertext version of) $\Sigma$ with a non-negligible advantage. This proves the theorem.

• Theorem. If $\Sigma$ is semantically secure (and bootstrappable), then $\Sigma^{(d)}$ is semantically secure for each $d$. 
Can we use just one pair of keys?

- The public key of $\Sigma^{(d)}$ (including the evaluation key) contains $d + 1$ $\Sigma$-public keys and a chain of $d$ encrypted $\Sigma$-secret keys.

- Question: why don't we use just one pair of keys?
Leveled FHE becomes FHE if $\Sigma$ is KDM-secure

- Theorem. If $\Sigma$ is KDM-secure, then we can shorten $pk^{(d)}$ to \( \{ pk_0, \overline{sk_0} \} \), with $\overline{sk_0} \leftarrow \text{Encrypt}(pk_0, sk_0)$. Then, all $\Sigma^{(d)}$ are the same and we have an FHE scheme.

\[
\begin{array}{ccccccc}
 pk_d & pk_{d-1} & \cdots & pk_1 & pk_0 \\
 \overline{sk_d} & \overline{sk_{d-1}} & \cdots & \overline{sk_1} & sk_0 \\
\end{array}
\Rightarrow
\begin{array}{ccccccc}
 pk_0 & pk_0 & \cdots & pk_0 & pk_0 \\
 \overline{sk_0} & \overline{sk_0} & \cdots & \overline{sk_0} & sk_0 \\
\end{array}
\]
KDM-Security

(KDM: Key-Dependent Message)
Recall: IND-CPA (semantic security)

• In the IND-CPA game,

\[
\Pr[ A \text{ wins}] \triangleq \Pr \left[ A^E_k \left( 1^\lambda, m_0, m_1, E_k(m_b) \right) = b : \\
\begin{bmatrix}
k \leftarrow G(1^\lambda),
\ b \leftarrow \{0,1\},
\ m_0, m_1 \leftarrow_A M
\end{bmatrix}
\right].
\]

• Define the adversary's advantage to be \(|\Pr[ A \text{ wins}] - 1/2|\).

• An encryption scheme is IND-CPA if all polynomial-time adversaries have negligible advantages.

• Remark: The game for asymmetric encryption is similar.
- Semantic security assumes that the messages to be encrypted are independent of the secret key.

- Suppose \( \Sigma = (G, E, D) \) is semantically secure (IND-CPA). Suppose we modify the encryption algorithm such that

\[
E'_k(m) = \begin{cases} 
0 || E_k(m) & \text{if } m \neq k \\
1 || k & \text{otherwise}
\end{cases}
\]

- Q: Is \( \Sigma' = (G, E', D) \) semantically secure?

- \( \Sigma' \) is apparently insecure if it is used to encrypt the key itself, and potentially insecure if used to encrypt key-dependent messages.

- This suggests the notion of KDM security.
KDM-security game (for asymmetric encryption)

- Parameters: security parameter $\lambda$, an integer $n > 0$, a class $C$ of functions that map $n$ secret keys to a message.

- Setup. The challenger chooses a random bit $b \leftarrow \{0, 1\}$, generates $n$ key pairs $(pk_1, sk_1), \ldots, (pk_n, sk_n)$, and sends public keys $(pk_1, \ldots, pk_n)$ to the adversary.

- Queries. The adversary issues queries of the form $(i, f)$ with $1 \leq i \leq n$ and $f \in C$. The challenger responds with
  
  $$
  c \leftarrow \begin{cases} 
  E(pk_i, m) & \text{if } b = 0 \\
  E(pk_i, 0^{\lvert m \rvert}) & \text{if } b = 1
  \end{cases}
  $$

  where $m = f(sk_1, \ldots, sk_n)$.

- Finish. The adversary guesses whether $b = 0$ or $b = 1$. 

KDM-security

- A public-key encryption scheme is \( n \)-way KDM-secure with respect to \( C \) if all polynomial-time adversaries have negligible advantages in the KDM-security game.

- Boneh et al (Crypto'08) proposed a KDM-secure encryption scheme w.r.t. the following class of functions:
  - all constant functions: \( f_m(x_1, ..., x_n) = m \) for \( m \in M \).
  - all selector functions \( f_i(x_1, ..., x_n) = x_i \) for \( 1 \leq i \leq n \).

- KDM-security for this class of functions implies semantic security as well as circular security. (In circular security, we have a cycle of \( n \) key pairs, and we are allowed to encrypt each \( sk_i, 1 \leq i \leq n \), under \( pk_{(i \mod n + 1)} \).)
The KDM-security needed for FHE

- The KDM-security needed to convert leveled FHE to FHE is circular security for some $n > 0$.
- Since the underlying SHE is bootstrappable, using multiple key-pairs ($n > 1$) does not seem to be more secure than using just one pair ($n = 1$). Why?