Fully homomorphic encryption scheme using ideal lattices

Gentry's STOC'09 paper - Part I

Homomorphic encryption

- KeyGen: On input 1^λ, outputs a pair of keys, (pk, sk).
- Encrypt: On input a public key pk and a plaintext $\pi \in M_{pk}$, outputs a ciphertext ψ. 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 ψ, outputs a plaintext π. 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

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

Correctness

- $\Sigma = \left(\text{KeyGen, Encrypt, Decrypt, Evaluate}\right)$.
- The scheme Σ is correct for circuit C if for any plaintexts $\left(\pi_1, \ldots, \pi_t\right)$ and any ciphertexts $(\psi_1, \ldots, \psi_t)$ with $\psi_i \leftarrow \text{Encrypt}(pk, \pi_i)$, it holds that:

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

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

Compactness

- $\Sigma = (\mathsf{KeyGen},\ \mathsf{Encrypt},\ \mathsf{Decrypt},\ \mathsf{Evaluate})$.
- The scheme Σ is compact if the output ciphertext of Evaluate is independent (in length) of the input circuit C; more specificly, Decrypt can be expressed as a circuit of size poly(λ).
- This is to avoid trivial solutions such as:
  - $\mathsf{Evaluate}(pk,\ C,\ \psi_1,\ \ldots,\ \psi_t)$ simply returns $\psi := (C,\ \psi_1,\ \ldots,\ \psi_t)$ as the ciphertext.
  - $\mathsf{Decrypt}(sk, \psi)$ decrypts each ψ_i to π_i and computes $C(\pi_1,\ \ldots,\ \pi_t)$.

Fully homomorphic encryption

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

- $\Sigma^{(d)} = \left(\mathsf{KeyGen}^{(d)},\ \mathsf{Encrypt}^{(d)},\ \mathsf{Decrypt}^{(d)},\ \mathsf{Evaluate}^{(d)}\right)$.
- A family of schemes $\left\{\Sigma^{(d)} : d \in \mathbb{Z}^{+}\right\}$ 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 λ, d, and (in the case of $\mathsf{Evaluate}^{(d)}$) 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, ElGammal, 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

Gentry's fully homomorphic encryption scheme

- 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?

- $\{\text{AND, XOR}\}$, i.e., $\{+, \times\}$, 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 $0 \le 2r \ll p$. $2r$ is the noise.
- Decryp(p, c): let $m = \left(c \bmod p\right) \bmod 2.$
- 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 $c_1 c_2$ is a ciphertext of $m_1 m_2$, with noise $2(2r_1 r_2 + r_1 m_2 + m_1 r_2)$.
- The noise grows!
- What if the noise becomes too large, say $2r > p$?

Challenge

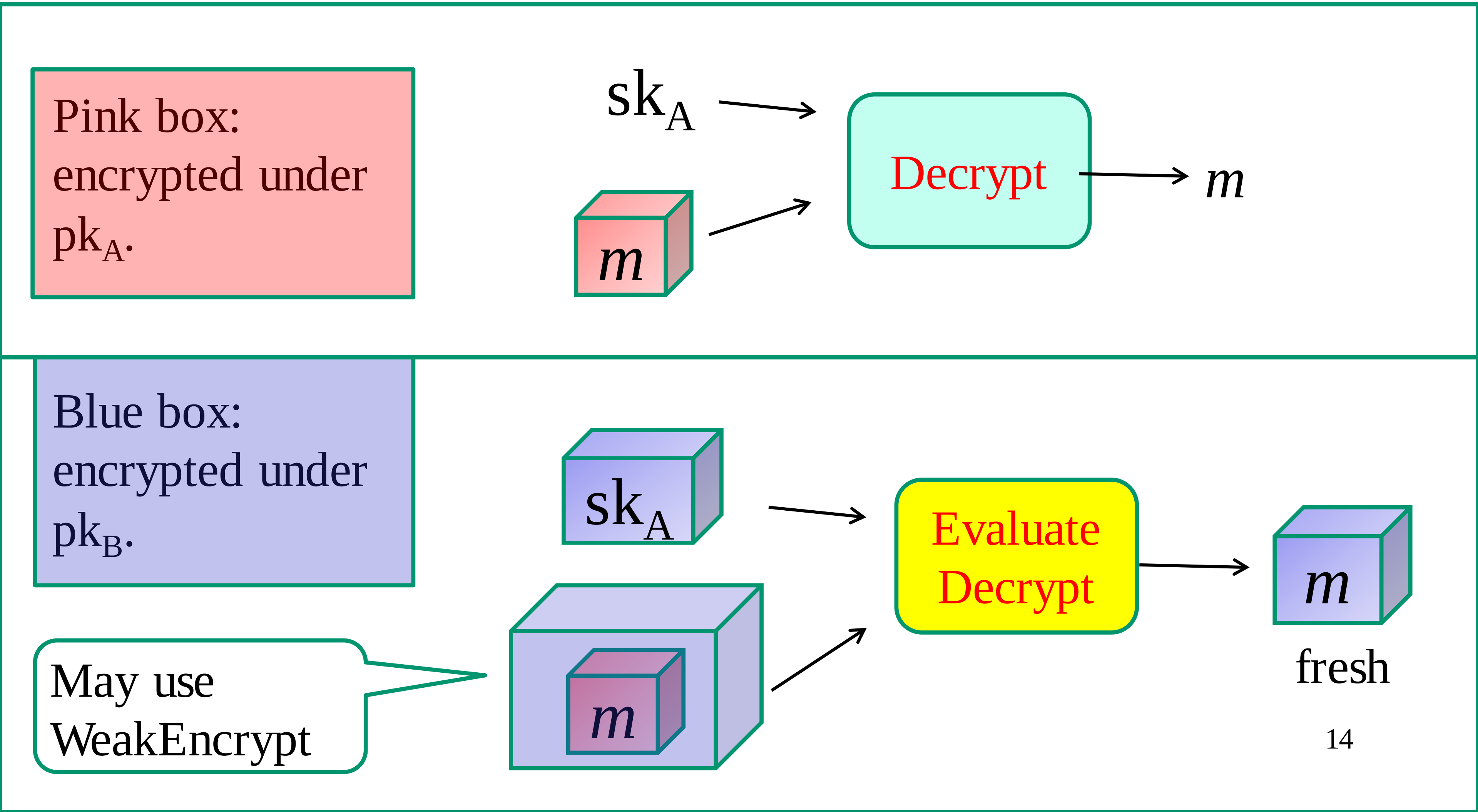
- Can we have a $\{+, \times\}$-homomorphic encryption scheme without noises growing?
- That is, the ciphertexts output by Evaluate is as fresh as those output by Encrypt (in terms of amount of noise).
- Such a scheme will automatically be fully homomorphic.
- Gentry proposed a simple yet powerful strategy 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

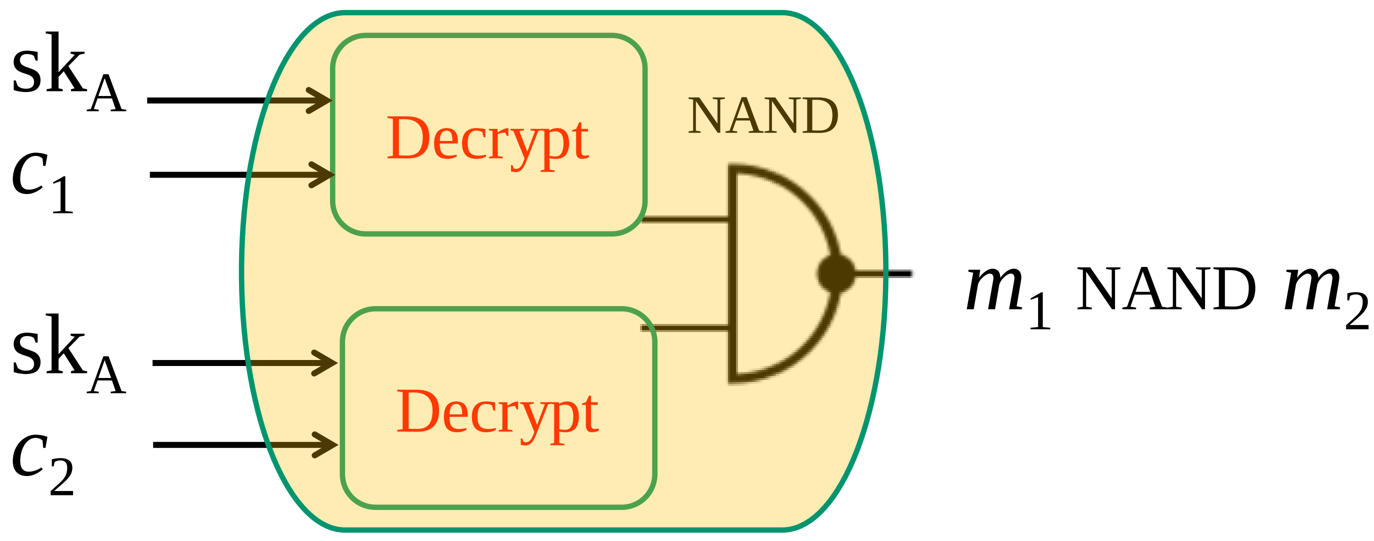
- We can allow anyone to convert a ciphertext under key pk_A into a ciphertext under key pk_B w/o revealing the message.



g-augumented decryption circuit

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

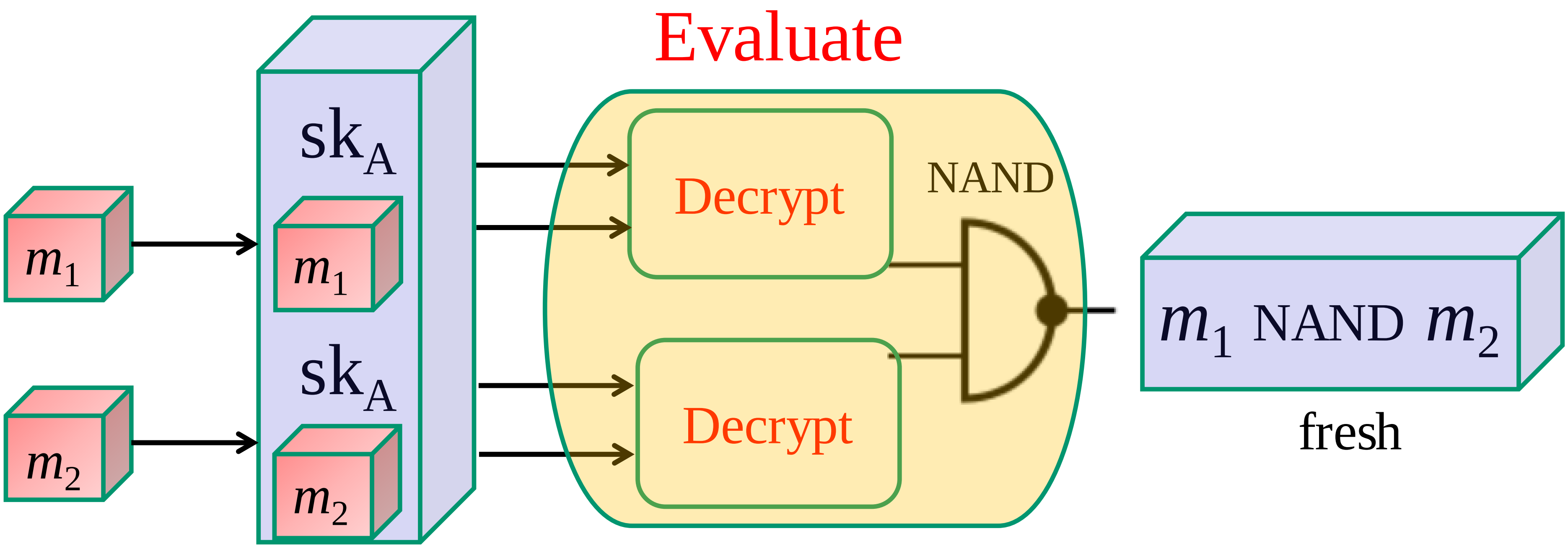
NAND-augmented Decrypt:



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 NAND m_2 under key pk_B.

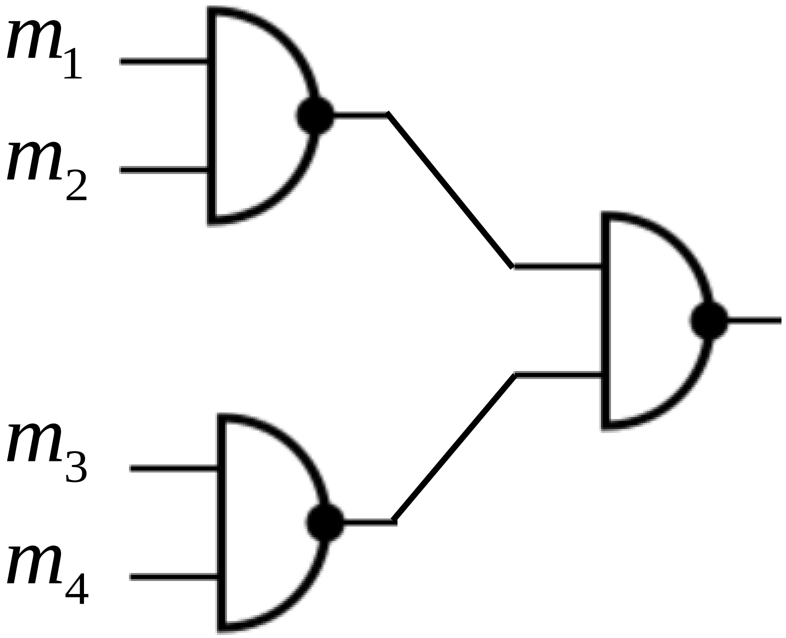


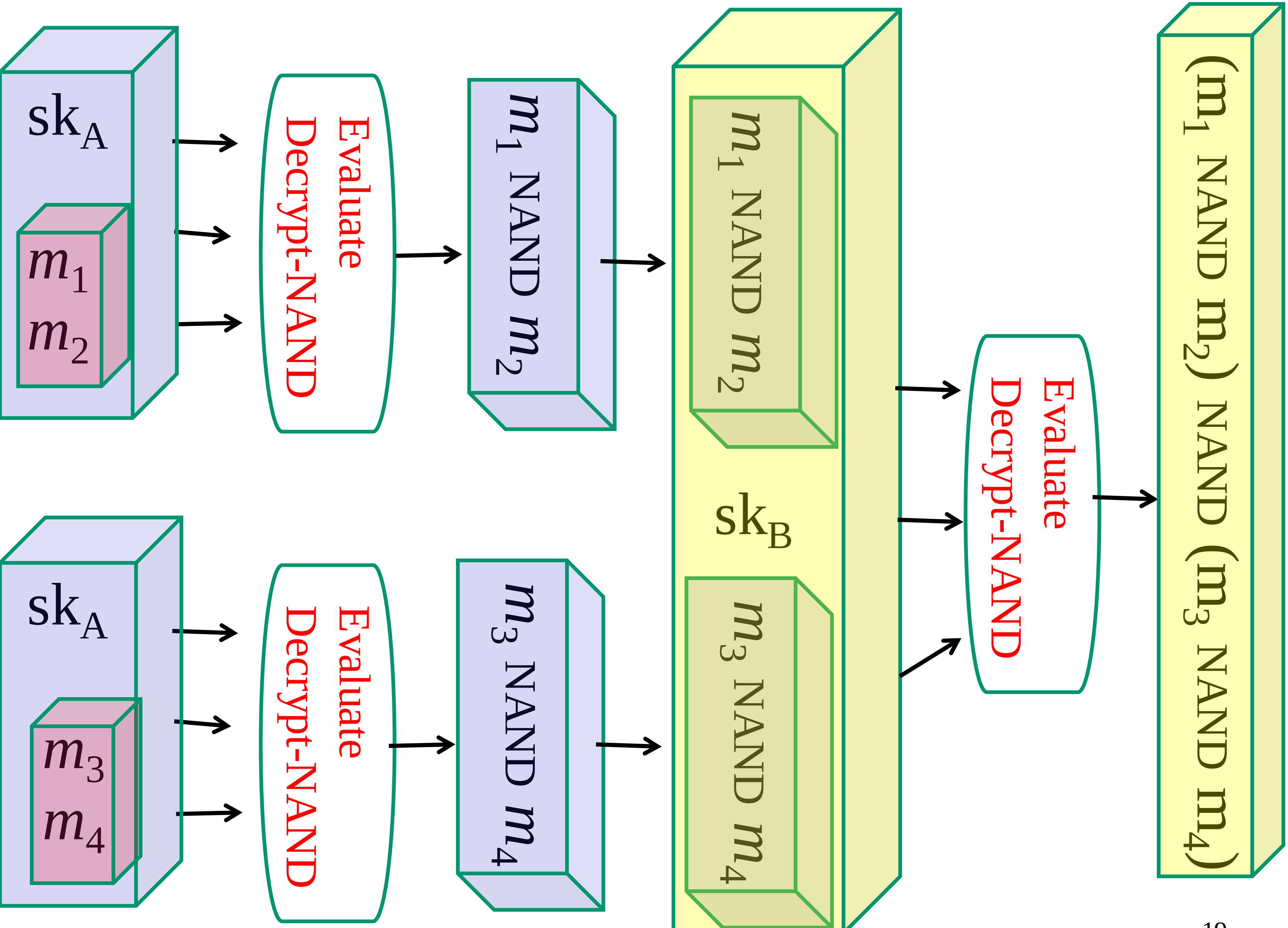
If we can evaluate NAND-Decrypt homomorphically...

- then from the ciphertexts of m_1 and m_2 under pk_A, we can obtain a "fresh" ciphertext of m_1 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 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 pk_A. $\mathsf{Evaluate}\left(C, \mathrm{pk}_A, \psi_1, \psi_2, \psi_3, \psi_4\right)$.




sk$_A$
m_1
m_2
Evaluate
Decrypt-NAND
m_1 NAND m_2
sk$_A$
m_3
m_4
Evaluate
Decrypt-NAND
m_3 NAND m_4
m_1 NAND m_2
sk$_B$
m_3 NAND m_4
Evaluate
Decrypt-NAND
(m_1 NAND m_2) NAND (m_3 NAND m_4)

Bootstrappable encryption

- $\Sigma = (\mathsf{KeyGen},\ \mathsf{Encrypt},\ \mathsf{Decrypt},\ \mathsf{Evaluate})$.
- Γ : 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 Σ is $\mathcal{C}$-homomorphic.
- Σ is said to be bootstrappable with respect to Γ if $D_\Sigma(\Gamma) \subseteq \mathcal{C}$.
- If Σ is bootstrappable w.r.t. a complete set of gates Γ (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 Γ).

$\Sigma^{(d)}$: homomorphic for circuits of depth $\leq d$

- Assume $\Sigma = (\mathsf{KeyGen},\ \mathsf{Encrypt},\ \mathsf{Decrypt},\ \mathsf{Evaluate})$ is bootstrappable w.r.t. a set of gates Γ. We construct from Σ $\Sigma^{(d)} = \left(\mathsf{KeyGen}^{(d)},\ \mathsf{Encrypt}^{(d)},\ \mathsf{Decrypt}^{(d)},\ \mathsf{Evaluate}^{(d)}\right)$.

- $\mathsf{KeyGen}^{(d)}(\lambda,\ d)$: //The same algorithm for all d.//
  - Use 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 : $\overline{sk_i} \leftarrow \mathsf{Encrypt}(pk_{i-1},\ sk_i)$.
  - Secret key: $sk^{(d)} = sk_0$.
  - Public key: $pk^{(d)} = \left\{ \langle pk_i \rangle_{0 \leq i \leq d},\ \langle \overline{sk_i} \rangle_{1 \leq i \leq d} \right\}$.

Encryption key

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

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

The rest are the evaluation key

Decryption key

- $\mathsf{Encrypt}^{(d)}$:
  - Input: a public key $pk^{(d)}$ and a plaintext π.
  - Output: ciphertext $\psi \leftarrow \mathsf{Encrypt}(pk_d, \ \pi)$.

- $\mathsf{Decrypt}^{(d)}$:
  - Input: a secret key $sk^{(d)}$ and a ciphertext ψ.
  - Output: ciphertext $\pi \leftarrow \mathsf{Decrypt}(sk_0, \ \psi)$.
  - Remark: ψ is assumed to be an output of $\mathsf{Evaluate}^{(d)}$.

    What if ψ was produced by $\mathsf{Encrypt}^{(d)}$?

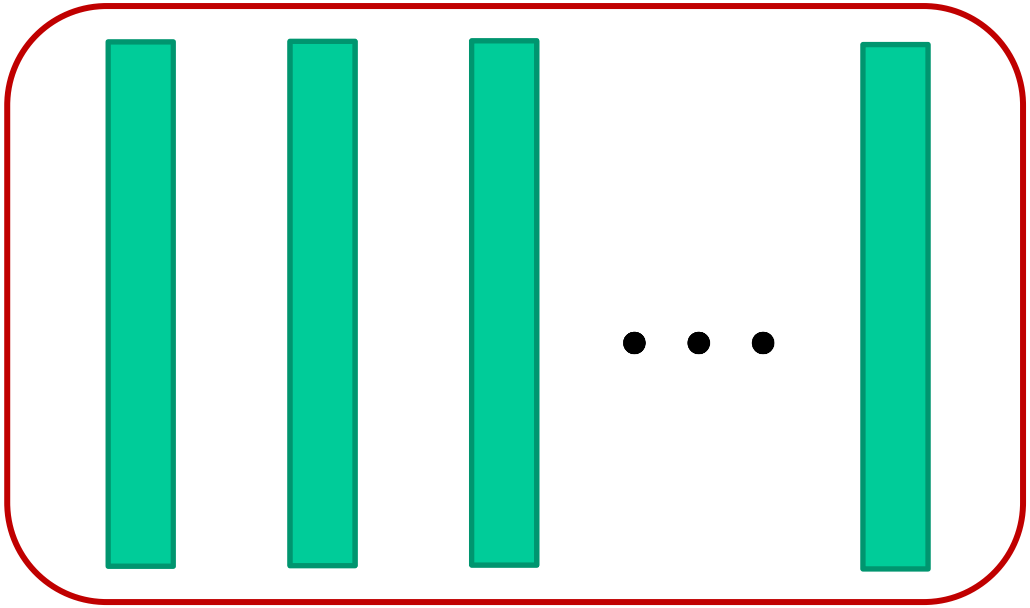
- $\text{Evaluate}^{(d)}\left(pk^{(d)},\ C_d,\ \Psi_d\right)$:
  - Recursive procudure: $\text{Evaluate}^{(\delta)}\left(pk^{(\delta)},\ C_\delta,\ \Psi_\delta\right)$.
  - C_δ has exactly δ levels; gates at level i are connected to gates at level $i-1$. (Any circuit of depth $\leq \delta$ can be converted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

Evaluate$^{(\delta)}\left(pk^{(\delta)},\, C_\delta,\, \Psi_\delta\right)$

$\Psi_\delta \Rightarrow$

under pk_δ

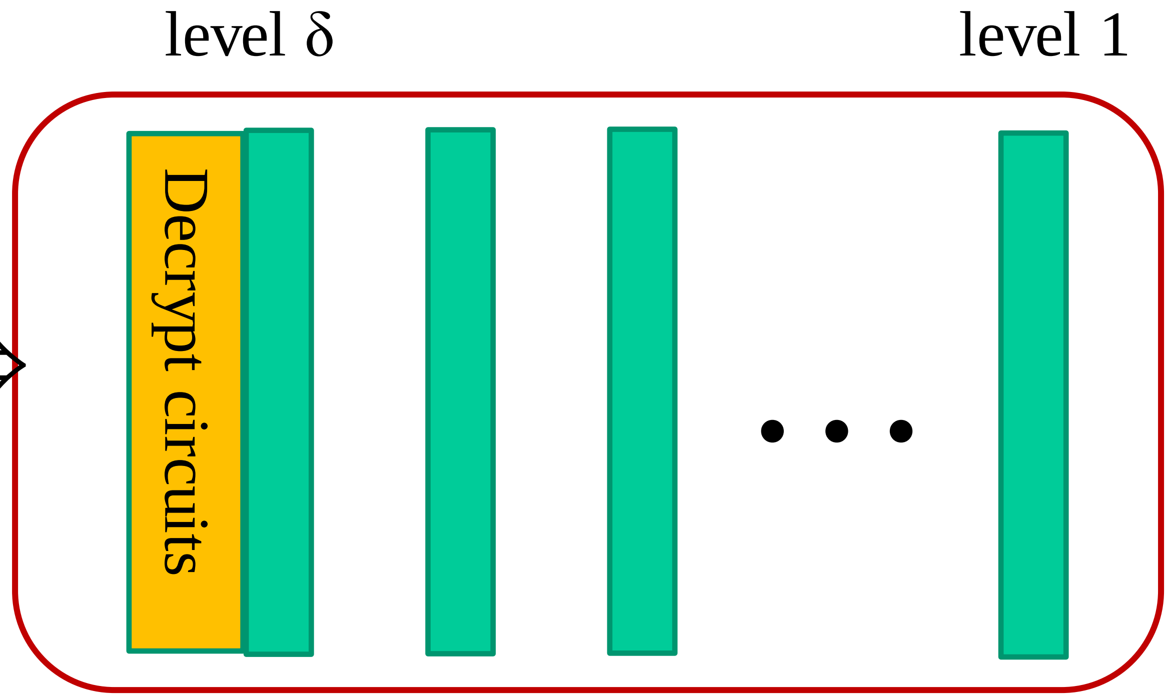
level δ level 1



C_δ

$\text{Evaluate}^{(\delta)}\left(pk^{(\delta)},\ C_\delta,\ \Psi_\delta\right)$

$$\left.\begin{matrix}\overline{\Psi_\delta}\\ \overline{sk_\delta}\end{matrix}\right\}\Rightarrow$$

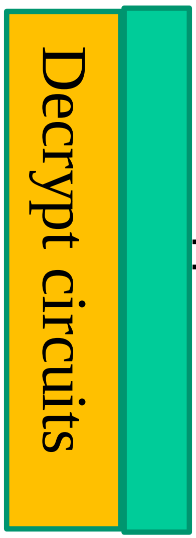


$\Psi_\delta,\ sk_\delta$ encrypted under $pk_{\delta-1}$

C_δ augmented with decryption circuits

$$\text{Evaluate}^{(\delta)}\left(pk^{(\delta)},\ C_{\delta},\ \Psi_{\delta}\right)$$

$$\left.\begin{array}{c}\overline{\Psi_{\delta}}\\ \overline{sk_{\delta}}\end{array}\right\} \Rightarrow$$

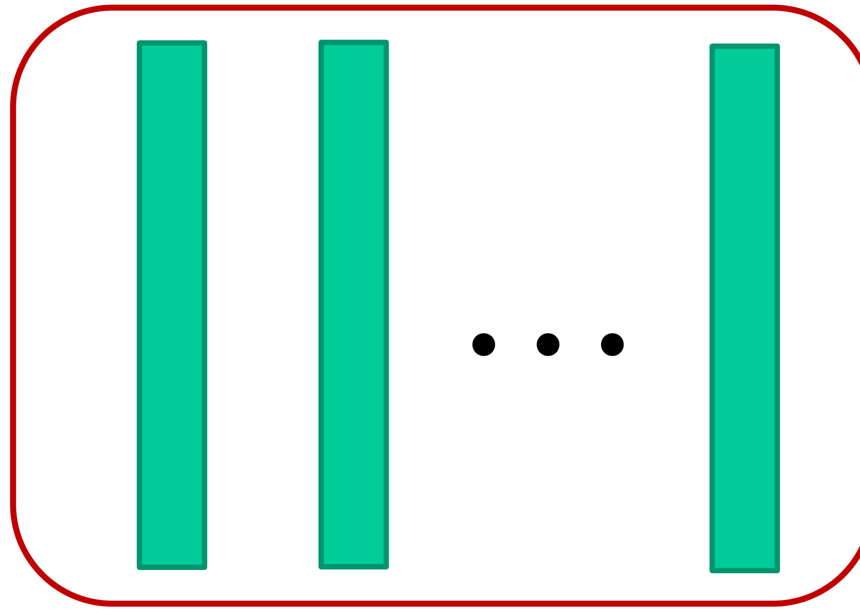


$$\Rightarrow \Psi_{\delta-1}$$

Ψ_{δ}, sk_{δ} encrypted under $pk_{\delta-1}$

under $pk_{\delta-1}$

level $\delta-1$ level 1

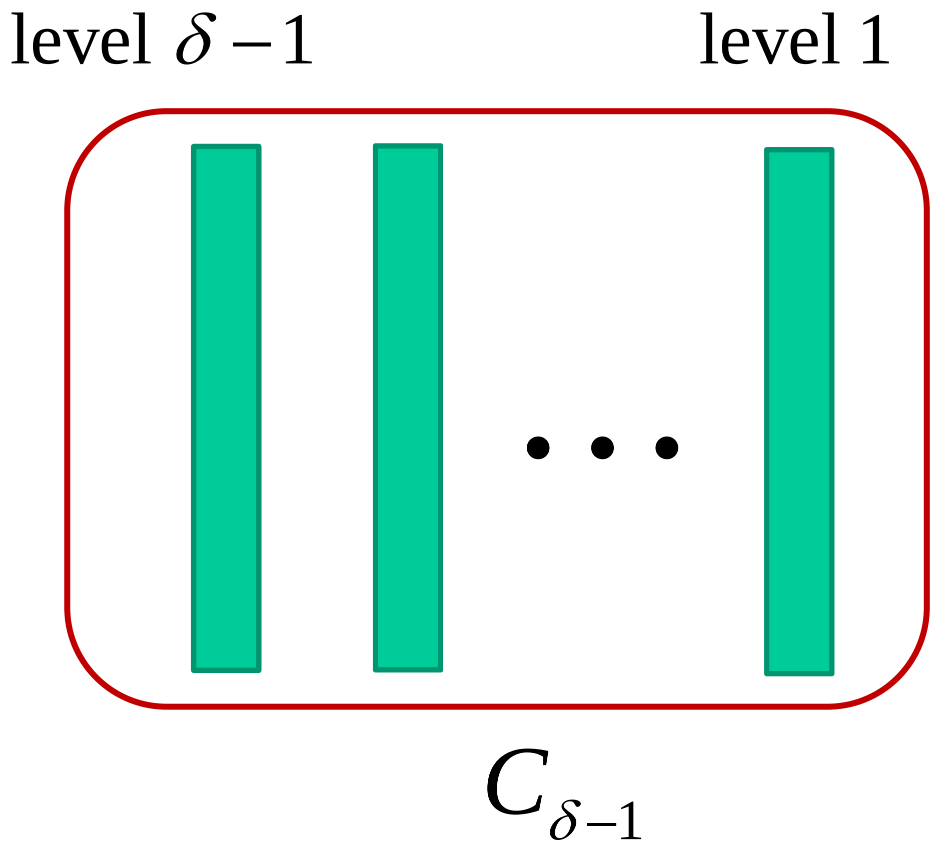


$C_{\delta-1}$

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

$\Psi_{\delta-1} \Rightarrow$

under $pk_{\delta-1}$



$$\text{Evaluate}^{(0)}\left(pk^{(0)},\ C_0,\ \Psi_0\right)$$

When $\delta = 0,$ simply return $\Psi_0,$

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

$$\Psi_\delta \Rightarrow \boxed{} \Rightarrow \Psi_\delta$$

under pk_δ

C_δ

Correctness

- Theorem. If Σ is bootstrappable w.r.t. a complete set of gates Γ (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 Γ).
- That is, $\mathsf{Decrypt}^{(d)}$ correctly evaluate any circuit (composed of gates in Γ) 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 Encrypt and $O(s)$ applications of Evaluate to $(g \in \Gamma)$-augmented decryption circuits, where $\ell = \ell(\lambda)$ 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 Encrypt and $O(s \cdot d)$ applications of Evaluate to $(g \in \Gamma)$-augmented decryption circuits.

Security

- Theorem. If Σ 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 λ,
  - 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 $c \leftarrow \mathrm{Enc}_{pk}(m_b)$ 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 $\mathbf{m}_0 = (m_{01}, \ldots, m_{0t})$, $\mathbf{m}_1 = (m_{11}, \ldots, m_{1t})$.

- The challenger chooses $b \leftarrow \{0,1\}$, encrypts $\mathbf{m}_b$ as $\boldsymbol{\psi}$, runs $\psi \leftarrow \text{Evaluate(pk}, C, \boldsymbol{\psi})$, and gives ψ to the adversary as the challenge ciphertext.

- The challenger may simply give $\boldsymbol{\psi}$ as the challenge ciphertext, since the adversary can run $\psi \leftarrow \text{Evaluate(pk}, C, \boldsymbol{\psi})$ 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 ordinry game. That is, breaking single-ciphertext semantic security ≤ 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 Σ is semantically secure (and bootstrappable), then $\Sigma^{(d)}$ is semantically secure for each d.

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

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

These encrypted keys $\overline{sk_i}$ 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 $\overline{sk_i}$, $d \geq i \geq 1$, is replaced by some $\overline{sk_i'}$ unrelated to pk_i :
  - $(sk_i', pk_i') \leftarrow \mathsf{KeyGen}(1^\lambda)$
  - $\overline{sk_i'} \leftarrow$ encryption of sk' under pk_{i-1}
- Game d = game for Σ. Game 0 = game for $\Sigma^{(d)}$.

$$pk_d \quad \cdots \quad pk_k \quad \cdots \quad pk_1 \quad pk_0$$

$$\overline{sk_d} \quad \cdots \quad \boxed{\overline{sk_k'} \quad \cdots \quad \overline{sk_1'}} \quad sk_0$$

- 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 Σ (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, $\varepsilon_d(\lambda) \le \varepsilon_{d-1}(\lambda) \le \cdots \le \varepsilon_0(\lambda)$.
- Two cases:
  - $\varepsilon_d(\lambda)$ is non-negligible (A breaks Σ and we are done).
  - $\varepsilon_d(\lambda)$ is negligible.
- Assume $\varepsilon_d(\lambda)$ is negligible. There must exist a $d > k \ge 0$ 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.

$$
\begin{array}{cccccc}
pk_d & \cdots & pk_{k+1} & pk_k & \cdots & pk_0 \\
\overline{sk_d} & \cdots & \overline{sk_{k+1}} & \overline{sk'_k} & \cdots & sk_0
\end{array}
$$

↖ 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:

$$
\overbrace{C \text{ (challenger)} \qquad \text{(adversary)}}^{\text{Game against } \Sigma} \; B \; \overbrace{\text{(challenger)} \qquad A \text{ (adversary)}}^{\text{Game against } \Sigma^{(d)}}
$$

- Remark: between B and C is a multi-ciphertext game.

- $\varepsilon_k(\lambda)$ is non-negligible and $\varepsilon_{k+1}(\lambda)$ is negligible.

$$
\begin{array}{cccccc}
pk_d & \cdots & pk_{k+1} & pk & \cdots & pk_0 \\
\overline{sk_d} & \cdots & \psi & \overline{sk'_k} & \cdots & sk_0
\end{array}
$$

$\nwarrow$ insecure if $\psi = \overline{sk_{k+1}}$

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

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

$$\overbrace{\hspace{20em}}^{\text{Game against } \Sigma}$$

C (challenger) | B (adversary)

1. generate $pk,\ sk$;
2. send pk to B;

5. send $\pi_0 = sk_{k+1},\ \pi_1 = sk'_{k+1}$ to C;

6. choose b;
7. send $\psi \leftarrow E_{pk}(\pi_b)$ to B;

8. (B is to guess b, with A's help);

14. if $\beta = \beta'$ then $b' = 0$ else $b' = 1$;
15. send b' to C.

$$\overbrace{\hspace{20em}}^{\text{Game against } \Sigma^{(d)}}$$

B (challenger) | A (adversary)

3. set up the game with A;
4. replace pk_k by pk;
9. replace $\overline{sk_{k+1}}$ by ψ;
10. send the "keys" to A;

11. send plaintexts $\pi'_0,\ \pi'_1$ to B;

12. choose β and send $\psi' \leftarrow E_{pk_d}(\pi'_\beta)$ to A;

13. send its guess β' to B;

- 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 Σ, from which one can construct an algorithm B' against (the single-ciphertext version of) Σ with a non-negligible advantage. This proves the theorem.

- Theorem. If Σ 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 $d+1$ Σ-public keys and a chain of d encrypted Σ-secret keys.

- Question: why don't we use just one pair of keys?

$$\boxed{\begin{array}{ccccc} pk_d & pk_{d-1} & \cdots & pk_1 & pk_0 \\ \\ \overline{sk_d} & \overline{sk_{d-1}} & \cdots & \overline{sk_1} & sk_0 \end{array}} \overset{?}{\Rightarrow} \boxed{\begin{array}{ccccc} pk_0 & pk_0 & \cdots & pk_0 & pk_0 \\ \\ \overline{sk_0} & \overline{sk_0} & \cdots & \overline{sk_0} & sk_0 \end{array}}$$

Leveled FHE becomes FHE if Σ is KDM-secure

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

$$\boxed{\begin{array}{ccccc} pk_d & pk_{d-1} & \cdots & pk_1 & pk_0 \\ \\ \overline{sk_d} & \overline{sk_{d-1}} & \cdots & \overline{sk_1} & sk_0 \end{array}} \Rightarrow \boxed{\begin{array}{ccccc} pk_0 & pk_0 & \cdots & pk_0 & pk_0 \\ \\ \overline{sk_0} & \overline{sk_0} & \cdots & \overline{sk_0} & sk_0 \end{array}}$$

KDM-Security

(KDM: Key-Dependent Message)

Recall: IND- CPA (semantic security)

- In the IND-CPA game,

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

- Define the adversary's advantage to be $\left|\Pr\left[A \text{ wins}\right] - 1/2\right|$.
- 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?
- Σ' 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 λ, 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 $(pk_1, sk_1), \ \dots, \ (pk_n, sk_n)$, and sends public keys $(pk_1, \ \dots, \ pk_n)$ to the adversary.

- Queries. The adversary issues queries of the form $(i, \ f)$ with $1 \le i \le n$ and $f \in C$. The challenger responds with

$$c \leftarrow \begin{cases} E(pk_i, \ m) & \text{if } b = 0 \\ E(pk_i, \ 0^{|m|}) & \text{if } b = 1 \end{cases} \quad \text{where } m = f(sk_1, \ \dots, \ sk_n).$$

- 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: $f_m(x_1, \ldots, x_n) = m$ for $m \in M$.
  - all selector functions $f_i(x_1, \ldots, x_n) = x_i$ for $1 \le i \le n$.

- 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, $1 \le i \le n$, under $pk_{(i \bmod n)+1}$).

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?