CSE208: Advanced Cryptography (FHE)

Daniele Micciancio

UCSD

Fall 2023

Section 1

Defining FHE

Public Key Encryption


```
PKE(Gen,Enc,Dec)
    Gen: () → (pk,sk)
    Enc: (pk,m) → c
    Dec: (sk,c) → m
```

- All algorithms are given an implicit security parameter k as input, and may be randomized
- Gen: Key Generation algorithm. Given a security parameter, produces a pair of matching secret and public keys
- Enc: Encryption algorithm, given the public key and a message, outputs a ciphertext
- Dec: Decryption algorithm, given the secret key and a ciphertext, recovers the message

Correctness of PKE

For every `(sk,pk)` $\leftarrow$ `Gen()` and `m` $\leftarrow$ `[M]`, `r` $\leftarrow$ `[R]`:

```
Dec(sk,Enc(pk,m;r)) = m
```

Notes:

- `[M]`: message space, may be just `{0,1}`, or $\{0,1\}^n$
- `[C]`: ciphertext space, usually $\{0,1\}^m$ for $m > n + k$
- `r` $\leftarrow$ `[R]`: randomness
- `A(x;r)` means run algorithm `A` on input `x` and randomness `r`
- Fixed size message space
  - assume $[M], [C]$ are finite sets
- Variable length messages:
  - break long message into blocks of size n
  - encrypt each block independently

Chosen Plaintext Attack (IND-CPA) security

- Indistinguishability under Chosen Plaintext Attack
- Experiment:

```
GameINDCPA(b:{0,1})
    (sk,pk) ← Gen()
    A(pk) → (m_0,m_1)
    b' ← A(Enc(pk,m_b))
    return b':{0,1}
```

Chosen Plaintext Attack (IND-CPA) security

- Indistinguishability under Chosen Plaintext Attack
- Experiment:

```
GameINDCPA(b:{0,1})
      (sk,pk) ← Gen()
      A(pk) → (m0,m1)
      b' ← A(Enc(pk,mb))
      return b':{0,1}
```

`Adv(A) = |Pr(Game(0)=1) - Pr(Game(1)=1)|`

Definition

(`Gen`,`Enc`,`Dec`) is **IND-CPA** secure if any efficient A has advantage `Adv(A)`$\approx$`0`

(t, ϵ) security: If `Time(A)<` `t` then `Adv(A)`$<\epsilon$

Pseudorandomness under CPA (RND-CPA)

- Similar to IND-CPA, but different experiment:

```
GameRNDCPA(b:{0,1})
      (sk,pk) ← Gen()
      A(pk) → m
      c_0 ←[C]
      c_1 ←Enc(pk,m_b)
      b' ← A(c_b)
      return b':{0,1}
```

Pseudorandomness under CPA (RND-CPA)

- Similar to IND-CPA, but different experiment:

```
GameRNDCPA(b:{0,1})
    (sk,pk) ← Gen()
    A(pk) → m
    c_0 ←[C]
    c_1 ←Enc(pk,m_b)
    b' ← A(c_b)
    return b':{0,1}
```

- How does IND-CPA security compare to RND-CPA?
  - If a scheme is RND-CPA, then it is also IND-CPA secure
  - A scheme can be IND-CPA, but not RND-CPA secure
  - Let's prove these statements on the blackboard

Significance of CPA security

- Adversary can choose messages m_0, m_1
  - No assumption about input distribution
  - Adversary may have partial information about messages
  - Adversary may influence the choice of messages
- Ciphertext c = Enc(pk, m_b) is computed honestly
  - Adversary cannot tamper with ciphertexts
- Adversary models a passive attacker
- Equivalent to many other seemingly stronger definitions
  - Adversary sees encryption of many message pairs $(m_0[i], m_1[i])$ for i=1,2,...
  - Multiuser: Encryptions under many independent keys (pk[i], sk[i])$\leftarrow$ Gen
- Define formally, and give reductions similarly to RND-CPA

Homomorphic Encryption: first attempt

- Assume `f: M` $\rightarrow$`M`, later will extend to multi-input functions
- Intuition: "Encryption commutes with function application"

```
f(Enc(pk,m)) = Enc(pk,f(m))
```

- How to apply `f` to a ciphertext

```
Eval(pk,f, Enc(pk,m)) = Enc(pk,f(m))
```

- Recall, `Enc` is randomized!
  - `Eval` and `Enc` are unlikely to produce the same ciphertext
  - should `Eval` and `Enc` produce identical distribution?
  - should ciphertexts produced by `Eval` be independent?

Homomorphic Encryption: second attempt


```
Dec(sk,Eval(pk,f, Enc(pk,m))) =  f(m)
```

This "homomorphic correctness" definition captures the workflow of a typical application

1. trusted party generates a pair of keys (pk, sk) $\leftarrow$ Gen()
2. data owner encrypts data m under pk, and stores ciphertext on public server
3. server carries out computation of program f on encrypted data
4. final result is decrypted using sk

This is the standard definition of correctness for homomorphic encryption.

Multi-input functions

- Many inputs are encrypted independently

$$c_1 \leftarrow \mathsf{Enc}(\mathsf{pk}, m_1)$$
$$\ldots$$
$$c_k \leftarrow \mathsf{Enc}(\mathsf{pk}, m_k)$$

Multi-input functions

- Many inputs are encrypted independently

$$c_1 \leftarrow \mathtt{Enc}(\mathtt{pk}, m_1)$$
$$\ldots$$
$$c_k \leftarrow \mathtt{Enc}(\mathtt{pk}, m_k)$$

- k-ary function $\mathtt{f}: (m_1, \ldots, m_k) \rightarrow \mathtt{m}$

$$\mathtt{Dec}(\mathtt{sk}, \mathtt{Eval}(\mathtt{pk}, \mathtt{f}, c_1, \ldots c_k)) = \mathtt{f}(m_1, \ldots, m_k)$$

- Different parties provide encrypted data to perform a joint computation
- Only owner of secret key sk can decrypt the result
- For added security, sk may be distributed using secret sharing scheme: this is called "Threshold FHE", and there is much to say about it

Correctness and Function Composition

- Let $x, y, z \in M$ be messages and $f, g : M \to M$ two functions such that $y = f(x)$ and $z = g(y) = (g \circ f)(x)$
- Assume (Gen,Enc,Dec,Eval) can evaluate f and g correctly:

```
Dec(sk,Eval(pk,f, Enc(pk,x))) = f(x)
Dec(sk,Eval(pk,g, Enc(pk,y))) = g(y)
```

Correctness and Function Composition

- Let $x, y, z \in M$ be messages and $f, g : M \to M$ two functions such that $y = f(x)$ and $z = g(y) = (g \circ f)(x)$
- Assume `(Gen,Enc,Dec,Eval)` can evaluate f and g correctly:

```
Dec(sk,Eval(pk,f, Enc(pk,x))) = f(x)
Dec(sk,Eval(pk,g, Enc(pk,y))) = g(y)
```

Question

Does it follow that

```
ctX ← Enc(pk,x)
ctY ← Eval(pk,f,ctX)
ctZ ← Eval(pk,g,ctY)
Dec(sk,ctZ) ≟ z
```

Security of Homomorphic Encryption


```
INDCPAgame(b:{0,1})
    (sk,pk) ← Gen()
    A(pk) → (m_0,m_1)
    return A(Enc(pk,m_b)): {0,1}
```

Remark

The IND-CPA security definition depends only on `Gen` *and* `Enc`*, but not on* `Dec` *(or* `Eval`*)*

Question

Can the IND-CPA security definition be applied as it is to FHE schemes `(Gen,Enc,Dec,Eval)`*?*

A trivial FHE scheme

Consider the following FHE scheme:

- Let `(Gen,Enc,Dec)` be IND-CPA secure
- Define `TrivialFHE = (Gen,Enc',Dec',Eval)`

```
Enc'(pk,m) = (Enc(pk,m),[])
Dec'(sk,(ct,[])) = Dec(sk,ct)
Dec'(sk,(ct,[f;fs])) = f(Dec'(sk,(ct,fs)))
Eval(pk,f,(ct,[fs])) = (ct,[f;fs])
```

Question

- *Is* `TrivialFHE` *a correct FHE scheme?*
- *Is* `TrivialFHE` *a secure FHE scheme?*
- *What makes the above scheme "trivial"?*

Compactness

- The `TrivialFHE` scheme is both correct and secure
- The problem with `TrivialFHE` is that it is not efficient:
  - Computation is performed by `Dec`, not `Eval`!

Definition

A FHE scheme is **compact** if the size of ciphertext `ct = Eval(pk,f,Enc(pk,m))` is independent of `Size(f)`

- Weaker forms of compactness:
  - Ciphertext size may grow logarithmic with `Size(f)`
  - Ciphertext size may depend on `Depth(f)`