Section 2

Defining FHE

Public Key Encryption


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

Correctness of PKE

For every (sk,pk) ← Gen() and m ← [M], r ← [R]:

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

Chosen Plaintext Attack (CPA) security

- Ciphertext Indistinguishability under Chosen Plaintext Attack
- Experiment:

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

Chosen Plaintext Attack (CPA) security

- Ciphertext Indistinguishability under Chosen Plaintext Attack
- Experiment:

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

Definition

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

Definition

An encryption scheme (`Gen`,`Enc`,`Dec`) is **IND-CPA** secure if any polynomial time A has advantage `Adv(A) ~ 0`

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 $\mathtt{c} = \mathsf{Enc}(\mathsf{pk}, m_b)$ is computed honestly
  - Adversary cannot tamper with ciphertexts
- Adversary models a passive attacker

Definition of CCA security

Definition

An encryption scheme (Gen,Enc,Dec) is **IND-CCA** secure if any polynomial time A has advantage Adv(A) ≈ 0 in the following game.

```
Game(b:{0,1})
    (sk,pk) ← Gen()
    A[D](pk) → (m_0,m_1)
    c ← Enc(pk,m_b)
    b' ← A[D'](c)
    return b':{0,1}
```

- A[D] is an adversary with oracle access to

  D(x) = Dec(sk,x)

- A[D'] uses a modified oracle (next slide)

IND-CCA1 vs IND-CCA2

There are two variants of CCA security, depending on the type of oracle given to the adversary after receiving the challenge ciphertext:

- **IND-CCA1** security: No decryption oracle after receiving the challenge

```
D'(x) = Nil
```

- **IND-CCA2** security: decrypt any ciphertext, except the challenge c

```
D'(x) =
    if (x ?= c)
        then Nil
        else Dec(sk,x)
```

Significance of CCA security

- Goal: model active attacks, where adversary can tamper with ciphertexts
- Standard notion for regular encryption schemes
- IND-CCA2 theoretically equivalent to *non-malleable* encryption
    - Any attempt to modify a ciphertext should be detected

Significance of CCA security

- Goal: model active attacks, where adversary can tamper with ciphertexts
- Standard notion for regular encryption schemes
- IND-CCA2 theoretically equivalent to *non-malleable* encryption
  - Any attempt to modify a ciphertext should be detected
- Seems incompatible with homomorphic encryption
  - Ability to modify ciphertexts can be a useful feature
  - Homomorphic encryption is *perfectly malleable*
- We will not consider CCA security

Homomorphic Encryption: first attempt

- Assume $f\colon M \to M$

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

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

Homomorphic Encryption: second attempt


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

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

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

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

$$\mathtt{Eval}(\mathtt{pk}, \mathtt{f}, c_1, \ldots, c_k)) = \mathtt{Enc}(\mathtt{pk}, \mathtt{f}(m_1, \ldots, m_k)) \text{ ???}$$

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

Multi-key Homomorphic encryption

- Assume multiple users: $P_1, P_2, \ldots$
- Each user has a key (pair): $P_i : (pk_i, sk_i)$
- Data is encrypted and sent to different users

$$c_1 \leftarrow \mathsf{Enc}(pk_1, m_1)$$
$$\ldots$$
$$c_t \leftarrow \mathsf{Enc}(pk_t, m_t)$$

- Users pool data together to perform a joint computation on $c_1, \ldots, c_t$

Multi-key Homomorphic encryption

- Assume multiple users: $P_1, P_2, \ldots$
- Each user has a key (pair): $P_i : (pk_i, sk_i)$
- Data is encrypted and sent to different users

$$
\begin{aligned}
c_1 &\leftarrow \texttt{Enc}(pk_1, m_1) \\
&\ldots \\
c_t &\leftarrow \texttt{Enc}(pk_t, m_t)
\end{aligned}
$$

- Users pool data together to perform a joint computation on $c_1, \ldots, c_t$
- Final result is an encryption of $f(m_1, \ldots, m_t)$ under what key?

$$
\begin{aligned}
&\texttt{Eval}(???, \texttt{f}, c_1, \ldots, c_t)) \\
&\quad \sim \texttt{Enc}(???, \ \texttt{f}(m_1, \ldots, m_t))
\end{aligned}
$$

Restricting Homomorphic Encryption

- FHE is a useful and challenging problem already in the single key setting

Restricting Homomorphic Encryption

- FHE is a useful and challenging problem already in the single key setting
- In order to appropach the problem we will further restrict it by parametrizing by a set of allowed computations/functions `Func = {f: ....}` where each `f: (M,...,M)` $\rightarrow$ `M` may take a different number of arguments

Restricting Homomorphic Encryption

- FHE is a useful and challenging problem already in the single key setting
- In order to appropach the problem we will further restrict it by parametrizing by a set of allowed computations/functions `Func = {f: ....}` where each `f: (M,...,M)`$\to$`M` may take a different number of arguments
- More generally, one many consider functions `f:`$(M_1, \ldots, M_k)$ $\to$`M` taking inputs from different sets (types), e.g., `ifThenElse: (Bool,Int,Int)`$\to$`Int`

Examples and Function Composition

- $(M, +, 0)$: abelian group, e.g., "fixed size" integers (modulo N)
- Addition: $f(x_1, ...x_t) = x_1 + ... + x_t$
- Scalar multiplication: $g_a(x) = a \cdot x$
- Linear combinations: $h(x_1, ...x_t) = \sum_i 2^{i-1} x_i$

Examples and Function Composition

- $(M, +, 0)$: abelian group, e.g., "fixed size" integers (modulo N)
- Addition: $f(x_1, ...x_t) = x_1 + ... + x_t$
- Scalar multiplication: $g_a(x) = a \cdot x$
- Linear combinations: $h(x_1, ...x_t) = \sum_i 2^{i-1} x_i$

- 1-hop, n-hop, multi-hop: can functions f be composed?

$$h(x_1, ..., x_t) = f(g_1(x_1), ..., g_{2^t-1}(x_t))$$

Correctness of 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 of 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) $\stackrel{?}{=}$ z

Formalizing Restricted Composition

- Restrict scheme to a set $\mathcal{F}$ of strongly typed functions:

$$f : M_1 \times \ldots M_k \to M_0$$

- `Enc`, `Dec`, `Eval` are given type information

Formalizing Restricted Composition

- Restrict scheme to a set $\mathcal{F}$ of strongly typed functions:

$$f : M_1 \times \ldots M_k \to M_0$$

- `Enc`,`Dec`,`Eval` are given type information
- We can use types to bound computation depth:
  - Start from $f : M \to M$
  - Define $M_i = M$ for $i = 1, \ldots, n$
  - Define $f_i : M_i \to M_{i+1}$, where $f_i(x) = f(x)$
- $\mathcal{F} = \{f\}$ allows arbitrary composition
- $\mathcal{F} = \{f_0\}$: no composition
- $\mathcal{F} = \{f_0, f_1, \ldots, f_n\}$: bounded depth composition

(Multi-hop) Correctness Game

- State: (initially empty) list L of message-ciphertext pairs

```
CorrectFHEgame() = (sk,pk) ← Gen()
                   L ← []
                   A[E,F](pk)
                   (m,c) ← last(L)
                   return (Dec(sk,c) ≠ m)

E(m) = c ← Enc(pk,m)
       L ← L;(m,c)
       return c
F(f,I) = (ms,cs) ← unzip L[I]
         m ← f(ms)
         c ← Eval(pk,f,cs)
         L ← L;(m,c)
         return c
```

Terminology

Reading papers, you will find references to

- Fully Homomorphic Encryption
- Somewhat Homomorphic Encryption
- Leveled Fully Homomorphic Encryption
- etc.

Terminology

Reading papers, you will find references to

- Fully Homomorphic Encryption
- Somewhat Homomorphic Encryption
- Leveled Fully Homomorphic Encryption
- etc.

We will use FHE as a catchall term

- Definition is parametrized by a set of functions $\mathcal{F}$
- Functions in $\mathcal{F}$ can be composed only if their types match
- $\mathcal{F}$ is closed under composition
- Can use "phantom" types to limit composition

We will rarely define $\mathcal{F}$ formally, but it is a useful exercise

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)`

Function Privacy


```
f0(x,y) = x + y
f1(x,y) = y + x

Game[A](b: {0,1})
  (sk,pk) ← Gen()
  ctX ← Enc(pk,x)
  ctY ← Enc(pk,y)
  ct ← Eval(pk,fb,ctX,ctY)
  return A(ct)
```

Question

Assume (Gen,Enc,Dec,Eval) *is a secure FHE scheme. Can an efficient adversary* A *recover the bit* b = Game[A](b) *?*

Passive Attacks to FHE


```
Game[A](b: {0,1})
  (sk,pk) ← Gen()
  State ← []
  b' ← A[E,D,F](pk)
  return b'
```

Adversary has access to three stateful oracles:

- Encryption oracle: $\texttt{E}(m_0, m_1)$
- Function Evaluation oracle: $\texttt{F}(f_0, f_1, \texttt{I})$
- Decryption oracle: $\texttt{D(i)}$
- Joint State: List of message-message-ciphertext triplets $(m_0, m_1, \texttt{ct})$

Passive Attack (oracles)


```
E(m_0,m_1) = ct ← Enc(pk,m_b)
             State ← (State;(m_0,m_1,ct))
             return ct

F(f_0,f_1,I) = (ms_0,ms_1,cts) ← unzip State[I]
               ct ← Eval(pk,f_b,cts)
               m_0 ← f_0(ms_0)
               m_1 ← f_1(ms_1)
               State ← State;(m_0,m_1,ct)
               return ct

D(i): (m_0,m_1,ct) ← State[i]
      if (m_0 ≡ m_1)
        then return Dec(sk,ct)
        else return Nil
```

Passive Attack with/without function privacy

- The game we just described guarantees function privacy
- A similar definition without function privacy can be obtained by requiring `f0`$\equiv$`f1` in the function evaluation queries

$$
\begin{aligned}
\texttt{F'(f,I):}\ & (ms_0, ms_1, \texttt{cts}) \leftarrow \textbf{unzip}\ \texttt{State[I]} \\
& \texttt{ct} \leftarrow \texttt{Eval(pk,f,cts)} \\
& m_0 = \texttt{f}(ms_0) \\
& m_1 = \texttt{f}(ms_1) \\
& \texttt{State} \leftarrow (\texttt{State};(m_0, m_1, \texttt{ct})) \\
& \textbf{return}\ \texttt{ct}
\end{aligned}
$$

Example: Circuit Privacy

- Assume messages are single bits `m: {0,1}`
- Let `FHE=(Gen,Enc,Dec,Eval)` a function private FHE scheme supporting `NAND(x,y)= not (x && y)`
- `EvalC(pk,C,...)`: evaluates boolean circuit `C:` $\{0,1\}^n \rightarrow$ `{0,1}` one gate at a time using `Eval(pk,NAND,...)`
- Let C_0, C_1 : NAND circuits with the same number of inputs and NAND gates
- `(sk,ps)` $\leftarrow$ `Gen()`
- Let xs_0, xs_1 be input bits such that $C_0(xs_0) = C_1(xs_1)$

Question

Are the following two distributions indistinguishable?

(pk, EvalC(pk, C_0, Enc(pk, xs_0)))
(pk, EvalC(pk, C_1, Enc(pk, xs_1)))