Section 3

Bootstrapping

Bootstrapping

- For simplicity: fix message space to $\{0,1\}$
- `HE=(Gen,Enc,Dec,Eval)`
  - Homomorphic functions: `Func = {` **nand** `}`
  - Supports only bounded computations: `Depth(C)< D`

Bootstrapping

- For simplicity: fix message space to $\{0, 1\}$
- `HE=(Gen,Enc,Dec,Eval)`
  - Homomorphic functions: `Func = {` **nand** `}`
  - Supports only bounded computations: `Depth(C)< D`

Question

Can we use HE *to build a FHE scheme supporting arbitrary circuits/functions?*

- The process of building FHE from HE is called "bootstrapping"

Decryption as a boolean function

- Everything is a sequence of bits
  - Secret key sk: $\{0,1\}^k$
  - Ciphertext ct: $\{0,1\}^l$
- Dec(sk,ct): {0,1}

Decryption as a boolean function

- Everything is a sequence of bits
  - Secret key `sk`: $\{0, 1\}^k$
  - Ciphertext `ct`: $\{0, 1\}^l$
- `Dec(sk,ct): {0,1}`
- Usually we think of `Dec` as a function
  - described by secret key `sk`
  - mapping ciphertext `ct` to message bit `Dec(sk,ct): {0,1}`

Decryption as a boolean function

- Everything is a sequence of bits
  - Secret key `sk`: $\{0,1\}^k$
  - Ciphertext `ct`: $\{0,1\}^l$
- `Dec(sk,ct): {0,1}`
- Usually we think of `Dec` as a function
  - described by secret key `sk`
  - mapping ciphertext `ct` to message bit `Dec(sk,ct): {0,1}`
- But we can also think of `Dec` as a function
  - described by ciphertext `ct`
  - mapping secret key `sk` to message bit `Dec(sk,ct): {0,1}`

Homomorphic Decryption

- Fix a ciphertext c
- Define $f_c : sk \mapsto Dec(sk, c)$
- Assume $Size(f_c) < S$, $Depth(f_c) < D$
- Let `bk[1..k] = Enc(pk,sk[1..k])`

Question

What is the result of the following computation?

`EvalC(pk,` f_c`,bk[1..k])`

Proxy Re-encryption

- Primary key: `(pk,sk)`
- Secondary key: `(pk1,sk1)`
- Re-encryption key: `rk = Enc(pk1,sk[1..k])`
- Input ciphertext `c = Enc(pk,m)`
- Decryption function f_c`(sk) = Dec(sk,c)`

Question

What is the result of the following computation?

`EvalC(pk1,`f_c`,rk)`

Decrypt and compute (unary)

- Homomorphic Encryption (Gen,Enc,Dec,Eval)
- Assume Func = { f_c | c: CipherText } where

  f_c(sk) = not (Dec(sk,c))

Decrypt and compute (unary)

- Homomorphic Encryption `(Gen,Enc,Dec,Eval)`
- Assume `Func = {` f_c `| c: CipherText }` where

f_c`(sk) = not (Dec(sk,c))`

```
(pk,sk) ← Gen()
ek  = Enc(pk,sk)
c   = Enc(pk,m)
```

Question

What is the result of the following computation?

`EvalC(pk,`f_c`,ek)`

Decrypt and compute (binary)

- Homomorphic Encryption `(Gen,Enc,Dec,Eval)`
- Assume `Func = {` $f_{c,c'}$ `| c,c': CipherText }` where

$f_{c,c'}$`(sk) = Dec(sk,c) `**`nand`**` Dec(sk,c')`

Decrypt and compute (binary)

- Homomorphic Encryption (Gen,Enc,Dec,Eval)
- Assume Func = { $f_{c,c'}$ | c,c': CipherText } where

$f_{c,c'}$(sk) = Dec(sk,c) **nand** Dec(sk,c')

```
(pk,sk) ← Gen()
ek  ← Enc(pk,sk)
c   ← Enc(pk,m)
c'  ← Enc(pk,m')
```

Question

What is the result of the following computation?

EvalC(pk, $f_{c,c'}$, ek)

Bootstrapping

- Given (1-hop) (`Gen`,`Enc`,`Dec`,`Eval`) supporting functions

  $f_{c,c'}$`(sk) = Dec(sk,c)` **`nand`** `Dec(sk,c')`

Bootstrapping

- Given (1-hop) `(Gen,Enc,Dec,Eval)` supporting functions

  $f_{c,c'}$`(sk) = Dec(sk,c) nand Dec(sk,c')`

- Define (multi-hop) FHE scheme with `Func = { nand }`

```
Gen'() = (sk,pk) ← Gen()
  ek ← Enc(pk,sk)
  return (sk,(pk,ek))

Enc'((pk,ek),m) = Enc(pk,m)

Eval'((pk,ek),nand,c,c')
  = EvalC(pk,f_{c,c'},ek)
```

Correctness

Let `(Gen',Enc',Dec,Eval')` be the new encryption scheme

Theorem

If `Dec(sk,c)= m` *and* `Dec(sk,c')= m'`*, then*

`Dec(sk,Eval'((pk,ek),`**`nand`**`,c,c') = m `**`nand`**` m'`

Correctness

Let `(Gen',Enc',Dec,Eval')` be the new encryption scheme

Theorem

If `Dec(sk,c)= m` *and* `Dec(sk,c')= m'`, *then*

`Dec(sk,Eval'((pk,ek),`**`nand`**`,c,c') = m `**`nand`**` m'`

Strong correctness property:

```
Dec(sk,Eval'((pk,ek),nand,c,c')
    = Dec(sk,c) nand Dec(sk,c')
```

for **any** ciphertexts `c,c'` !

Security

- Assume `FHE = (Gen,Enc,Dec,Eval)` is IND-CPA secure
- Build new scheme `FHE'`:

```
Gen'() = (sk,pk) ← Gen()
   ek ← Enc(pk,sk)
   return (sk,(pk,ek))

Enc'((pk,ek),m) = Enc(pk,m)
```

Is `FHE'` IND-CPA secure?

Leveled Homomorphic Encryption

- Goal: build a FHE supporting NAND circuits of depth up to L, for any given L
- Key generation procedure takes L as input:

Leveled Homomorphic Encryption

- Goal: build a FHE supporting NAND circuits of depth up to L, for any given L
- Key generation procedure takes `L` as input:

```
Gen'(L) =
  for (i=0..L)
    (sk[i],pk[i]) ← Gen()
  for (i=1..L)
    ek[i] = Enc(pk[i],sk[i-1])
  sk' = sk[0..L]
  pk' = pk[0..L],ek[1..L]
  return (sk',pk')

Enc'(pk',m) = Enc(pk[0],m)
```

FHE Today

State of the art

We can build leveled FHE from standard LWE assumption

- Built using bootstrapping
- Inefficient, but better than nothing

Open problem

Build (non-leveled) FHE from standard LWE

- In practice, one can apply bootstrapping with $\mathsf{ek} = \mathsf{Enc}(\mathsf{pk}, \mathsf{sk})$
- Much smaller key than leveled FHE
- No known attacks to circular security
- Still, it is not known how to prove security