On Homomorphic Encryption and Secure Computation

Shai Halevi

IBM|NYU|Columbia Theory Day, May 7, 2010

Computing on Encrypted Data

Wouldn't it be nice to be able to…

- Encrypt my data in the cloud
- While still allowing the cloud to search/sort/edit/… this data on my behalf
- Keeping the data in the cloud in encrypted form
  - Without needing to ship it back and forth to be decrypted

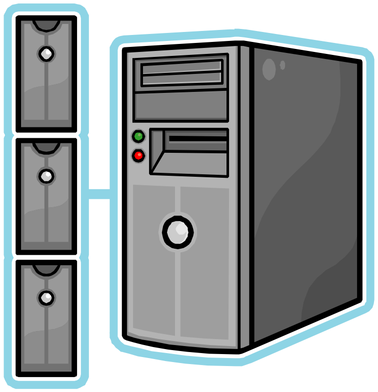
Computing on Encrypted Data

Wouldn't it be nice to be able to…

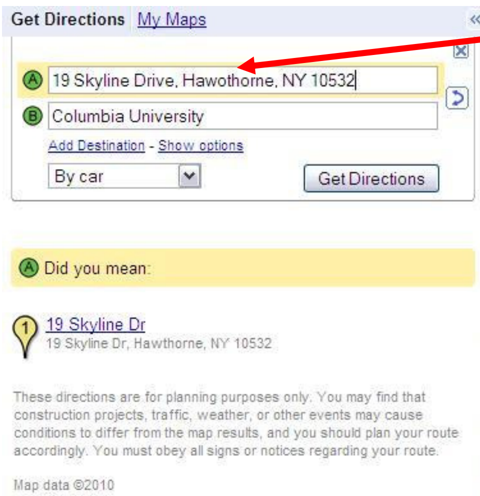
- Encrypt my queries to the cloud
- While still allowing the cloud to process them
- Cloud returns encrypted answers
  - that I can decrypt

Computing on Encrypted Data

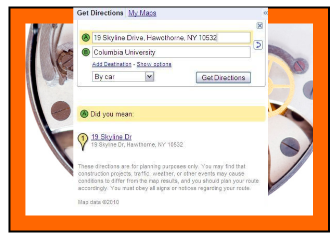
Directions
- From: 19 Skyline Drive, Hawothorne, NY 10532, USA
- To: Columbia University



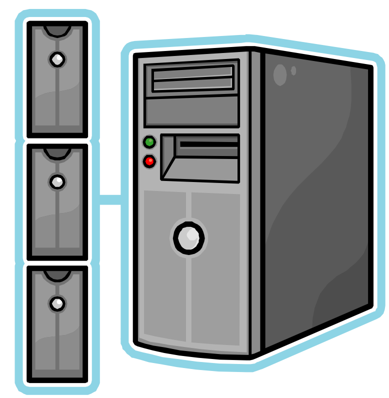
Computing on Encrypted Data



typo

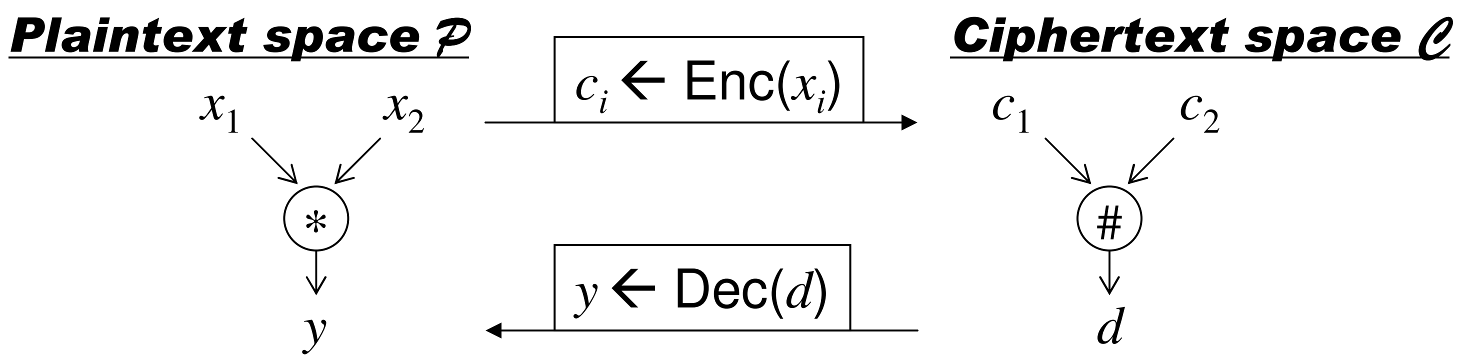


$kjh9*mslt@na0
&maXxjq02bflx
m^00a2nm5,A4.
pE.abxp3m58bsa
(3saM%w,snanba
nq~mD=3akm2,A
Z,ltnhde83|3mz{n
dewiunb4]gnbTa*
kjew^bwJ^mdns0



Part I: Constructing Homomorphic Encryption

Privacy Homomorphisms [RAD78]



Some examples:

- "Raw RSA": $c \leftarrow x^e \bmod N$ ($x \leftarrow c^d \bmod N$)
  - $x_1^e \times x_2^e = (x_1 \times x_2)^e \bmod N$
- GM84: $\mathsf{Enc}(0) \in_R$ QR, $\mathsf{Enc}(1) \in_R$ QNR (in Z_N^*)
  - $\mathsf{Enc}(x_1) \times \mathsf{Enc}(x_2) = \mathsf{Enc}(x_1 \oplus x_2) \bmod N$

More Privacy Homomorphisms

- Mult-mod-p [ElGamal’84]
- Add-mod-N [Pallier’98]
- NC1 circuits [SYY’00]
- Quadratic-polys mod p [BGN’06]
- Poly-size branching programs [IP’07]
- See Part II for a “different type of solution” for any poly-size circuit [Yao’82,…]

(x,+)-Homomorphic Encryption

It will be really nice to have…

- Plaintext space Z_2 (w/ ops +,x)
- Ciphertext space some ring $\mathcal{R}$ (w/ ops +,x)
- Homomorphic for both + and x
  - Enc(x_1) + Enc(x_2) in $\mathcal{R}$ = Enc(x_1+ x_2 mod 2)
  - Enc(x_1) x Enc(x_2) in $\mathcal{R}$ = Enc(x_1 x x_2 mod 2)
- Then we can compute any function on the encryptions
  - Since every binary function is a polynomial
- We won't get exactly this, but it's a good motivation

Some Notations

- An encryption scheme: (KeyGen, Enc, Dec)
  - Plaintext-space = {0,1}
  - $(pk,sk) \leftarrow$ KeyGen(\$), $c \leftarrow \text{Enc}_{pk}(b)$, $b \leftarrow \text{Dec}_{sk}(c)$
- Semantic security [GM'84]:

$$(pk, \text{Enc}_{pk}(0)) \approx (pk, \text{Enc}_{pk}(1))$$

$\approx$ means indistinguishable by efficient algorithms

Homomorphic Encryption

- H = {KeyGen, Enc, Dec, Eval}

  $c^* \leftarrow \text{Eval}_{pk}(f, c)$
- Homomorphic: $\text{Dec}_{sk}(\text{Eval}_{pk}(f, \text{Enc}_{pk}(x))) = f(x)$
  - ("Fully" Homomorphic: for every function f)
  - $\text{Enc}_{pk}(f(x))$, $\text{Eval}_{pk}(f, \text{Enc}_{pk}(x))$ may differ
    - As long as both distributions decrypt to $f(x)$
- Function-private: $\text{Eval}_{pk}(f, \text{Enc}_{pk}(x))$ hides f
- Compact: $|\text{Eval}_{pk}(f, \text{Enc}_{pk}(x))|$ independent of $|f|$

(x,+)-Homomorphic Encryption, the Gentry Way [G'09]

Evaluate any function in four "easy" steps

- Step 1: Encryption from linear ECCs
  - Additive homomorphism
- Step 2: ECC lives inside a ring
  - Also multiplicative homomorphism
  - But only for a few operations (i.e., low-degree poly's)
- Step 3: Bootstrapping
  - Few ops (but not too few) → any number of ops
- Step 4: Everything else

Step One:
Encryption from Linear ECCs

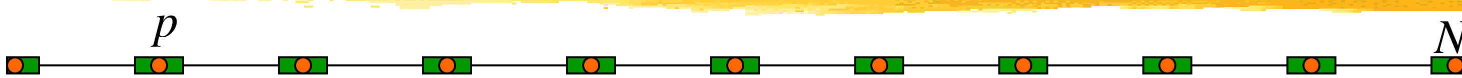
- For "random looking" codes, hard to distinguish close/far from code

- Many cryptosystems built on this hardness
  - E.g., [McEliece'78, AD'97, GGH'97, R'03,…]

Encryption from linear ECCs

- KeyGen: choose a "random" code $\mathcal{C}$
  - Secret key: "good representation" of $\mathcal{C}$
    - Allows correction of "large" errors
  - Public key: "bad representation" of $\mathcal{C}$
- Enc(0): a word close to $\mathcal{C}$
- Enc(1): a random word
  - Far from $\mathcal{C}$ (with high probability)

An Example: Integers mod p (similar to [Regev'03])



- Code determined by an integer p
  - Codewords: multiples of p
- Good representation: p itself
- Bad representation:
  - $N = pq$, and also many many $x_i = pq_i + r_i$

$r_i \ll p$

- Enc(0): subset-sum(x_i's)+r mod N
- Enc(1): random integer mod N

A Different Input Encoding

- Plaintext bit is LSB of dist(c, $\mathcal{C}$)
  - Enc(0/1): close to $\mathcal{C}$, distance is even/odd
  - In our example of integers mod p:
    - Enc(b) = 2(subset-sum(x_i's)+r) +b mod N
    - Dec(c) = (c mod p) mod 2

p is odd

- **Thm:** If "$\mathcal{C}$ co-prime with 2", then Enc(0), Enc(1) indistinguishable
  - w is near-$\mathcal{C}$/random → $2w$+b is Enc(b)/random

Additive Homomorphism

- c_1+c_2 = (codeword$_1$+codeword$_2$) $+2(r_1+r_2)+b_1+b_2$
  - codeword$_1$+codeword$_2$ $\in \mathcal{C}$
  - If $2(r_1+r_2)+b_1+b_2$ < min-dist/2, then it is the distance between c_1+c_2 and $\mathcal{C}$
  - dist(c_1+c_2, $\mathcal{C}$) = b_1+b_2 mod 2
- Additively-homomorphic while close to $\mathcal{C}$

Step 2: ECC Lives in a Ring $\mathcal{R}$

- What happens when multiplying in $\mathcal{R}$:
  - $c_1 c_2$ = (codeword$_1$+$2r_1$+b_1) x (codeword$_2$+$2r_2$+b_2)
    = codeword$_1$ X + Y codeword$_2$
    + $(2r_1+b_1)(2r_2+b_2)$
- If:
  - codeword$_1$ X + Y codeword$_2$ $\in \mathcal{C}$
  - $(2r_1+b_1)(2r_2+b_2)$ < min-dist/2
- Then
  - dist($c_1 c_2$, $\mathcal{C}$) = $(2r_1+b_1)(2r_2+b_2)$ = $b_1 b_2$ mod 2

$\mathcal{C}$ is both a left-ideal and a right-ideal

Product in $\mathcal{R}$ of small elements is small

Integers Rings [vDGHV'10]

- Recall mod-p scheme: $c_i = q_i p + 2r_i + b_i$ (mod $N=qp$)
  - Parameters: $|r_i|=n$, $|p|=n^2$, $|q|=|q_i|=n^5$
- c_1+c_2 mod N = $(q_1+q_2-\kappa q)p$ + $2(r_1+r_2)+(b_1+b_2)$
  - ➔sum mod p = $2(r_1+r_2)$ + (b_1+b_2)
- c_1 x c_2 mod N = $(c_1q_2+q_1c_2-q_1q_2-\kappa q)p$
  $+ 2(2r_1r_2+r_1m_2+m_1r_2) + b_1b_2$
  - ➔product mod p = $2(2r_1r_2+\ldots)$ + b_1b_2
- Can evaluate polynomials of degree ~ n before the distance from $\mathcal{C}$ exceeds $p/2$

Integers Rings [vDGHV'10]

Thm: "Approximate GCD" is hard
➔ Enc(0), Enc(1) are indistinguishable

- Apprixmate-GCD: Given $N=qp$ and many $x_i = pq_i + r_i$, hard to recover p

Polynomial Rings [G'09]

- $\mathcal{R}$ = polynomial ring modulo some $f(x)$
  - E.g., $f(x) = x^n + 1$
- $\mathcal{C}$ is an ideal in $\mathcal{R}$
  - E.g., random $g(x)$, $\mathcal{C}_g = \{\ g \times h \text{ mod } f\ :\ h \in \mathcal{R}\ \}$
    - $\mathcal{C}$ is also a lattice
  - Good representation: g itself
  - Bad representation: Hermite-Normal-Form
- If g has t-bit coefficients, can evaluate polynomials of degree O($t/log\ n$)

Polynomial Rings [G'09]

Thm: Bounded-Distance Decoding in ideal lattices is hard → Enc(0), Enc(1) are indistinguishable

- Bounded-Distance-Decoding: Given x close to the lattice, find dist(x, lattice)

*Doesn't quite fit the mold

Matrix Rings* [GHV'10]

- $\mathcal{R}$ = ring of mxm matrices over Z_q
  - q = poly(n), $m > n \log q$ (n security-parameter)
- $\mathcal{C}$ has low-rank matrices mod q (rank=n)
  - A is a random nxm matrix, $\mathcal{C}_A = \{ AX : X \in \mathcal{R} \}$
  - Bad representation: A itself
  - Good representation: full rank $T_{m\text{x}m}$ (over Z), small entries, $TA = 0 \bmod q$
- Problem: $\mathcal{C}_A$ is left-ideal, but not right-ideal
  - Can still evaluate quadratic formulas, no more

*Doesn't quite fit the mold

Matrix Rings [GHV'10]*

Thm: Learning with Errors hard
➔ Enc(0), Enc(1) are indistinguishable

- Learning with Errors: Given A, $A\boldsymbol{x}+\boldsymbol{e}$ (random $A,\boldsymbol{x}$, small error $\boldsymbol{e}$), find $\boldsymbol{x}$

Step 3: Bootstrapping [G'09]

- So far, can evaluate low-degree polynomials

x_1

x_2

…

x_t

P

$P(x_1, x_2, \ldots, x_t)$

Step 3: Bootstrapping [G'09]

- So far, can evaluate low-degree polynomials

x_1

x_2

…

x_t

P

$P(x_1, x_2, \dots, x_t)$

- Can eval $y=P(x_1,x_2\dots,x_n)$ when x_i's are "fresh"
- But y is an "evaluated ciphertext"
  - Can still be decrypted
  - But eval $Q(y)$ will increase noise too much

Step 3: Bootstrapping [G'09]

- So far, can evaluate low-degree polynomials

x_1

x_2

…

x_t

$P(x_1, x_2, \ldots, x_t)$

- Bootstrapping to handle higher degrees:
- For ciphertext c, consider $\mathbf{D}_c(sk) = \mathrm{Dec}_{sk}(c)$
  - Hope: $D_c(*)$ is a low-degree polynomial in sk
  - Then so are $A_{c_1,c_2}(sk) = \mathrm{Dec}_{sk}(c_1) + \mathrm{Dec}_{sk}(c_2)$
    and $M_{c_1,c_2}(sk) = \mathrm{Dec}_{sk}(c_1) \times \mathrm{Dec}_{sk}(c_2)$

Step 3: Bootstrapping [G'09]

- Include in the public key also $\text{Enc}_{pk}(sk)$

Requires "circular security"

x_1 x_2

c_1 c_2

sk_1

sk_2

...

sk_n

M_{c_1,c_2}

c

$\mathrm{M}_{c_1,c_2}(sk)$

$= \text{Dec}_{sk}(c_1) \times \text{Dec}_{sk}(c_2) = x_1 \times x_2$

Step 3: Bootstrapping [G'09]

- Include in the public key also $\mathrm{Enc}_{pk}(sk)$

Requires "circular security"

x_1 x_2

c_1 c_2

sk_1

sk_2

...

sk_n

c

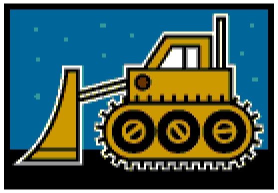
$\mathrm{M}_{c_1,c_2}(sk)$

$= \mathrm{Dec}_{sk}(c_1) \times \mathrm{Dec}_{sk}(c_2) = x_1 \times x_2$

- Homomorphic computation applied only to the "fresh" encryption of sk

Step 4: Everything Else

- Cryptosystems from [G’09, vDGHV’10] cannot handle their own decryption as-is
- Apply some tricks to “squash” the decryption procedure



Part II:
Homomorphic Encryption vs. Secure Computation

Secure Function Evaluation (SFE)

Client Alice has data x

Server Bob has function f

Alice wants to learn $f(x)$

1. Without telling Bob what x is
2. Bob may not want Alice to know f
3. Client Alice may also want server Bob to do most of the work computing $f(x)$

Two-Message SFE [Yao'82,…]

Alice(x)

Bob(f)

$(c,s) \leftarrow$ SFE1(x) $\xrightarrow{c}$

$r \leftarrow$ SFE2(f,c)

$y \leftarrow$ SFE3(s,r) $\xleftarrow{r}$

- Many different instantiations are available
  - Based on hardness of factoring/DL/lattices/…
- Alice's x and Bob's f are kept private
- But Alice does as much work as Bob
  - Bob's reply of size poly(n) x ($|f|+|x|$)

Recall: Homomorphic Encryption

- H = {KeyGen, Enc, Dec, Eval}
- Semantic security: $(pk, \mathrm{Enc}_{pk}(0)) \approx (pk, \mathrm{Enc}_{pk}(1))$
- Homomorphic: $\mathrm{Dec}_{sk}(\mathrm{Eval}_{pk}(f, \mathrm{Enc}_{pk}(x))) = f(x)$
  - (“Fully” Homomorphic: for every function f)
  - $\mathrm{Enc}_{pk}(f(x))$, $\mathrm{Eval}_{pk}(f, \mathrm{Enc}_{pk}(x))$ may differ
    - As long as both distributions decrypt to $f(x)$
- Function-private: $\mathrm{Eval}_{pk}(f, \mathrm{Enc}_{pk}(x))$ hides f
- Compact: $|\mathrm{Eval}_{pk}(f, \mathrm{Enc}_{pk}(x))|$ independent of $|f|$

Aside: a Trivial Solution

- Eval(f,c) = <f,c>, Dec*(<f,c>) = f(Dec(c))
- Neither function-private, nor compact
- Not very useful in applications

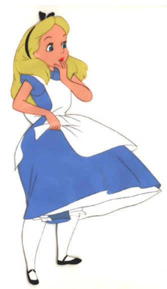
HE → Two-Message SFE

- Alice encrypts data x
  - sends to Bob $c \leftarrow \text{Enc}(x)$
- Bob computes on encrypted data
  - sets $c^* \leftarrow \text{Eval}(f, c)$
  - c^* is supposed to be an encryption of $f(x)$
  - Hopefully it hides f (function-private scheme)
- Alice decrypts, recovers $y \leftarrow \text{Dec}(c^*)$

Two-Message SFE → HE

- Roughly:
  - Alice's message $c \leftarrow$ SFE1(x) is Enc(x)
  - Bob's reply $r \leftarrow$ SFE2(f,c) is Eval(f,c)
- Not quite public-key encryption yet
  - Where are (pk, sk)?
  - Can be fixed with an auxiliary PKE scheme

Two-Message SFE → HE



Alice(pk, x) Bob(f) Dora(sk)

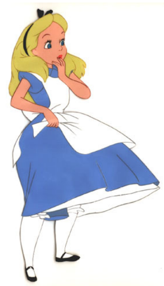
$(c,s) \leftarrow$ SFE1(x) $\xrightarrow{\;c\;}$

$r \leftarrow$ SFE2(f,c)

$y \leftarrow$ SFE3(s,r) $\xleftarrow{\;r\;}$

- Add an auxiliary encryption scheme
  - with (pk,sk)

Two-Message SFE → HE



Alice(pk, x)

Bob(f)

Dora(sk)

$\underbrace{\boxed{\begin{array}{r}(c,s) \leftarrow \text{SFE1}(x) \\ c' \leftarrow \text{Enc}_{pk}(s)\end{array}}}_{\text{Enc}'_{pk}(x)}$ $\xrightarrow{c,\ c'}$ $\underbrace{\boxed{r \leftarrow \text{SFE2}(f,c)}}_{\text{Eval}_{pk}(f,c,c')}$ $\xrightarrow{r,\ c'}$ $\underbrace{\boxed{\begin{array}{l}s \leftarrow \text{Dec}_{sk}(c') \\ y \leftarrow \text{SFE3}(s,r)\end{array}}}_{\text{Dec}_{sk}(r,c')}$

- Recall: $|r|$ could be as large as poly(n)($|f|+|x|$)
  - Not compact

A More Complex Setting: i-Hop HE [GHV10b]

Alice(x) Bob(f) Charlie(g) Dora(sk)

$c_0 \leftarrow \text{Enc}(x) \xrightarrow{c_0} c_1 \leftarrow \text{Eval}(f, c_0) \xrightarrow{c_1} c_2 \leftarrow \text{Eval}(g, c_1) \xrightarrow{c_2} y \leftarrow \text{Dec}(c_2)$

$\underbrace{c_1 \leftarrow \text{Eval}(f, c_0) \rightarrow c_2 \leftarrow \text{Eval}(g, c_1)}_{\text{2-Hop Homomorphic Encryption}}$

- c_1 is not a fresh ciphertext
  - May look completely different
- Can Charlie process it at all?
  - What about security?

Multi-Hop Homomorphic Encryption

- H = {KeyGen, Enc, Eval, Dec} as before
- i-Hop Homomorphic (i is a parameter)

$$x \to \boxed{\mathrm{Enc}_{pk}(x)} \xrightarrow{c_0} \underbrace{\boxed{\mathrm{Eval}_{pk}(f_1,c_0)} \xrightarrow{c_1} \boxed{\mathrm{Eval}_{pk}(f_2,c_1)} \xrightarrow{c_2} \cdots \xrightarrow{c_j}}_{\text{Any number } j \le i \text{ hops}} \boxed{\mathrm{Dec}_{sk}(x)} \to y$$

  - $y = f_j(f_{j-1}(\ldots f_1(x) \ldots))$ for any $x, f_1, \ldots, f_j$

- Similarly for i-Hop function-privacy, compactness
- Multi-Hop: i-Hop for any i

1-Hop → multi-Hop HE

- (KeyGen,Enc,Eval,Dec) is 1-Hop HE
  - Can evaluate any single function on ctxt
- We have $c_1 = \mathrm{Eval}_{pk}(f_1, c_0)$, and some other f_2

Bootstrapping:

- Include with pk *also* $c^* = \mathrm{Enc}_{pk}(sk)$
- Consider $F_{c_1, f_2}(sk) = f_2(\ \mathrm{Dec}_{sk}(c_1)\ )$
  - Let $c_2 = \mathrm{Eval}_{pk}(F_{c_1, f_2},\ c^*)$

1-Hop → multi-Hop HE

c^* $\quad f_i \quad c_{i-1}$

sk $\quad x_{i-1}$

F_{c_{i-1},f_i}

c_{i+1}

$F_{c_{i-1},f_i}(sk)$

$= f_i(\,\mathsf{Dec}_{sk}(c_{i-1})\,) = f_i(x_{i-1})$

- Drawback: $|c_i|$ grows exponentially with i:
  - $|F_{c_{i-1},f_i}| \geq |c_{i-1}|+|f_i|$
  - $|c_i| = |\mathsf{Eval}_{pk}(F_{c_{i-1},f_i}, c^*)| \geq \mathrm{poly}(n)(|c_{i-1}|+|f_i|)$
- Does not happen if underlying scheme is compact

Or even $|\mathsf{Eval}_{pk}(F_{c_{i-1},f_i}, c^*)| = |c_{i-1}|+\mathrm{poly}(n)|f_i|$

Other Constructions

- Private 1-hop HE + Compact 1-hop HE
  ➔ Compact, Private 1-hop HE
  ➔ Compact, Private multi-hop HE
- A direct construction of multi-hop HE from Yao’s protocol

Summary

- Homomorphic Encryption is useful
  - Especially multi-hop HE
- A method for constructing HE schemes from linear ECCs in rings
  - Two (+ε) known instances so far
- Connection to two-message protocols for secure computation

Thank You