Fully Homomorphic Encryption over the Integers

Many slides borrowed from Craig

Marten van Dijk[1], Craig Gentry[2], Shai Halevi[2], Vinod Vaikuntanathan[2]

1 – MIT, 2 – IBM Research

Computing on Encrypted Data

- Storing my files on the cloud
  - Encrypt them to protect my information
  - Search through them for emails with “homomorphic” in the subject line
    - Cloud should return only these (encrypted) messages, w/o knowing the key
- Private Internet search
  - Encrypt my query, send to Google
  - I still want to get the same results
    - Results would be encrypted too

Public-key Encryption

- Three procedures: KeyGen, Enc, Dec
  - (sk,pk) ← KeyGen($)
    - Generate random public/secret key-pair
  - c ← $Enc_{pk}(m)$
    - Encrypt a message with the public key
  - m ← $Dec_{sk}(c)$
    - Decrypt a ciphertext with the secret key

- E.g., RSA: $c \leftarrow m^e \bmod N$, $m \leftarrow c^d \bmod N$
  - (N,e) public key, d secret key

Homomorphic Public-key Encryption

- Also another procedure: Eval
  - $c^* \leftarrow \mathrm{Eval}_{pk}(\Pi, c_1,\ldots,c_n)$

Encryption of output value $m^*=\Pi(m_1,\ldots,m_n)$

Circuit

Encryption of inputs $m_1,\ldots,m_n$ to Π

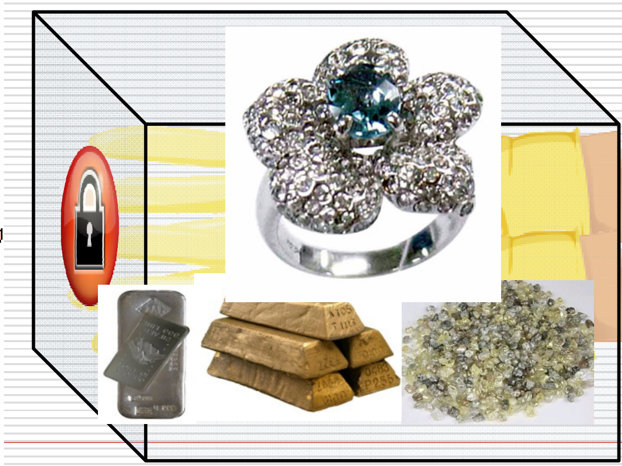
  - Π a Boolean circuit with ADD, MULT mod 2

An Analogy: Alice's Jewelry Store

- Alice's workers need to assemble raw materials into jewelry
- But Alice [illegible]d about theft
    - How ca[illegible]ers process the raw materi[illegible] having access to them?

An Analogy: Alice's Jewelry Store

- Alice puts materials in locked glove box
  - For which only she has the key
- Workers assemble jewelry in the box
- Alice unlocks box to get "results"



The Analogy

- Enc: putting things inside the box
  - Anyone can do this (imagine a mail-drop)
  - $c_i \leftarrow \mathrm{Enc}_{pk}(m_i)$
- Dec: Taking things out of the box
  - Only Alice can do it, requires the key
  - $m^* \leftarrow \mathrm{Dec}_{sk}(c^*)$
- Eval: Assembling the jewelry
  - Anyone can do it, computing on ciphertext
  - $c^* \leftarrow \mathrm{Eval}_{pk}(\Pi, c_1,\ldots,c_n)$
- $m^* = \Pi(m_1,\ldots,m_n)$ is "the ring", made from "raw materials" $m_1,\ldots,m_n$

Can we do it?

- As described so far, sure..
  - $(\Pi, c_1,\ldots,c_n) = c^* \leftarrow \mathrm{Eval}_{pk}(\Pi, c_1,\ldots,c_n)$
  - $\mathrm{Dec}_{sk}(c^*)$ decrypts individual c_i's, apply Π

(the workers do nothing, Alice assembles the jewelry by herself)

This is the main challenge

Of course, this is cheating:

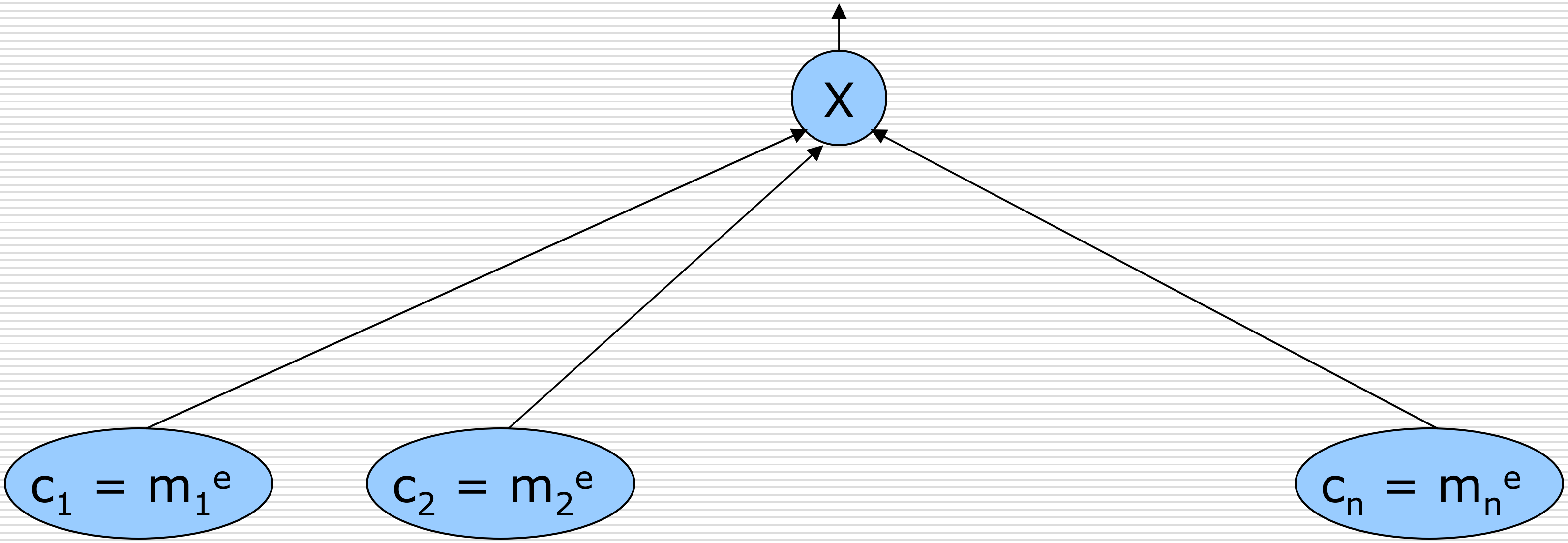
- We want c^* to remain small
  - independent of the size of Π
  - "Compact" homomorphic encryption
- We may also want Π to remain secret

Can be done with "generic tools" (Yao's garbled circuits)

What was known?

- "Somewhat homomorphic" schemes:
  - Only work for some circuits
- E.g., RSA works for MULT gates (mod N)

$c^* = c_1 \times c_2 \ldots \times c_n = (m_1 \times m_2 \ldots \times m_n)^e \pmod{N}$



“Somewhat Homomorphic” Schemes

- RSA, ElGamal work for MULT mod N
- GoMi, Paillier work for XOR, ADD
- BGN05 works for quadratic formulas
- SYY99 works for shallow fan-in-2 circuits
  - c* grows exponentially with the depth of Π
- IP07 works for branching program
- MGH08 works for low-degree polynomials
  - c* grows exponentially with degree

A Recent Breakthrough

- Genrty09: A bootstrapping technique

  Somewhat homomorphic → Fully homomorphic

  Scheme E can evaluate its own decryption circuit → Scheme E* can evaluate any circuit

- Gentry also described a candidate "bootstrappable" scheme
  - Based on ideal lattices

The Current Work

- A second “bootstrappable” scheme
  - Very simple: using only modular arithmetic
- Security is based on the hardness of finding “approximate-GCD”

Outline

1. A homomorphic symmetric encryption
2. Turning it into public-key encryption
   - Result is "almost bootstrappable"
3. Making it bootstrappable
   - Similar to Gentry'09

   Time permitting
4. Security
5. Gentry's bootstrapping technique

   Not today

A homomorphic symmetric encryption

- Shared secret key: odd number p
- To encrypt a bit m:
  - Choose at random large q, small r
  - Output $c = pq + 2r + m$

    2r+m much smaller than p
    - Ciphertext is close to a multiple of p
    - m = LSB of distance to nearest multiple of p
- To decrypt c:
  - Output $m = (c \bmod p) \bmod 2$

Why is this homomorphic?

- $c_1 = q_1 p + 2r_1 + m_1, \quad c_2 = q_2 p + 2r_2 + m_2$

Distance to nearest multiple of p

- $c_1 + c_2 = (q_1 + q_2)p + 2(r_1 + r_2) + (m_1 + m_2)$
  - $2(r_1+r_2)+(m_1+m_2)$ still much smaller than p

  $\rightarrow c_1 + c_2 \bmod p = 2(r_1+r_2) + (m_1+m_2)$

- $c_1 \times c_2 = (c_1 q_2 + q_1 c_2 - q_1 q_2)p$

  $+ 2(2r_1 r_2 + r_1 m_2 + m_1 r_2) + m_1 m_2$
  - $2(2r_1 r_2 + \ldots)$ still much smaller than p

  $\rightarrow c_1 x c_2 \bmod p = 2(2r_1 r_2 + \ldots) + m_1 m_2$

How homomorphic is this?

- Can keep adding and multiplying until the "noise term" grows larger than q/2
  - Noise doubles on addition, squares on multiplication
- We choose $r \sim 2^n$, $p \sim 2^{n^2}$ (and $q \sim 2^{n^5}$)
  - Can compute polynomials of degree ~n before the noise grows too large

Homomorphic Public-Key Encryption

- Secret key is an odd p as before
- Public key is many "encryptions of 0"
  - $x_i = [q_i p + 2r_i]_{x0}$ for $i=1,2,\ldots,n$
- $\mathrm{Enc}_{pk}(m) = [\text{subset-sum}(x_i\text{'s})+m+2r]_{x0}$
- $\mathrm{Dec}_{sk}(c) = (c \bmod p) \bmod 2$
- Eval as before

Keeping it small

- The ciphertext's bit-length doubles with every multiplication
  - The original ciphertext already has n^6 bits
  - After ~log n multiplications we get $\sim n^7$ bits
- We can keep the bit-length at n^6 by adding more "encryption of zero"
  - $|y_1|=n^6+1$, $|y_2|=n^6+2$, …, $|y_m|=2n^6$
  - Whenever the ciphertext length grows, set $c' = c \bmod y_m \bmod y_{m-1} \ldots \bmod y_1$

Bootstrappable yet?

c/p, rounded to nearest integer

- Almost, but not quite:
- Decryption is $m = c - (p \times [c/p]) \bmod 2$
  - Same as $c-[c/p] \bmod 2$, since p is odd
  - Computing $[c/p] \bmod 2$ takes degree O(n)
  - But O() has constant bigger than one
    - Our scheme only supports degree $< n$
- To get a bootstrappable scheme, use Gentry09 technique to "squash the decryption circuit"

Squashing the decryption circuit

- Add to public key many real numbers
  - $r_1, r_2, \ldots, r_t \in [0,2]$
  - $\exists$ sparse set S for which $\Sigma_{i\in S}\ r_i = 1/p \bmod 2$
- Enc, Eval output $\psi_i = c \times r_i \bmod 2,\ i=1,\ldots,t$
  - Together with c itself
- New secret key is bit-vector $\sigma_1,\ldots,\sigma_t$
  - $\sigma_i=1$ if $i\in S$, $\sigma_i=0$ otherwise
- New Dec(c) is $c - [\Sigma_i\ \sigma_i \Psi_i] \bmod 2$
  - Can be computed with a "low-degree circuit" because S is sparse

Security

- The approximate-GCD problem:
  - Input: integers x_1, x_2, $x_{3,}$ ...
    - Chosen as $x_i = q_i p + r_i$ for a secret odd p
    - $p \in_\$ [0,P]$, $q_i \in_\$ [0,Q]$, $r_i \in_\$ [0,R]$ (with $R \ll P \ll Q$)
  - Task: find p
- Thm: If we can distinguish Enc(0)/Enc(1) for some p, then we can find that p
  - Roughly: the LSB of r_i is a "hard core bit"

  ➔ Scheme is secure if approx-GCD is hard
- Is approx-GCD really a hard problem?

Hardness of Approximate-GCD

- Several lattice-based approaches for solving approximate-GCD
  - Related to Simultaneous Diophantine Approximation (SDA)
  - Studied in [Hawgrave-Graham01]
    - We considered some extensions of his attacks
- All run out of steam when $|q_i|>|p|^2$
  - In our case $|p|\sim n^2$, $|q_i|\sim n^5 \gg |p|^2$

Relation to SDA

- $x_i = q_i p + r_i$ ($r_i \ll p \ll q_i$), $i = 0,1,2,\ldots$
  - $y_i = x_i/x_0 = (q_i+s_i)/q_0$, $s_i \sim r_i/p \ll 1$
  - y1, y2, … is an instance of SDA
    - q_0 is a denominator that approximates all y_i's
- Use Lagarias'es algorithm:
  - Consider the rows of this matrix:
  - Find a short vector in the lattice that they span
  - $\langle q_0,q_1,\ldots,q_t\rangle \cdot L$ is short
  - Hopefully we will find it

$$L=\begin{pmatrix} R & x_1 & x_2 & \ldots & x_t \\ & -x_0 & & & \\ & & -x_0 & & \\ & & & \ldots & \\ & & & & -x_0 \end{pmatrix}$$

Relation to SDA (cont.)

- When will Lagarias'es algorithm succeed?
  - $<q_0,q_1,\ldots,q_t>\cdot L$ should be shortest in lattice
    - In particular shorter than $\sim\det(L)^{1/t+1}$
  - This only holds for $t > \log Q/\log P$
  - The dimension of the lattice is t+1
  - Quality of lattice-reduction deteriorates exponentially with t
  - When $\log Q > (\log P)^2$ (so $t>\log P$), LLL-type reduction isn't good enough anymore

Minkowski bound

Conclusions

- Fully Homomorphic Encryption is a very powerful tool
- Gentry09 gives first feasibility result
  - Showing that it can be done "in principle"
- We describe a "conceptually simpler" scheme, using only modular arithmetic

- What about efficiency?
  - Computation, ciphertext-expansion are polynomial, but a rather large one…

Thank you