Advanced Topics in Cryptography

Lecture 5: Homomorphic encryption

Benny Pinkas

Related papers

- Paillier's cryptosystem
  – Pascal Paillier, Public-Key Cryptosystems Based on Composite Degree Residuosity Classes, Eurocrypt '99, pp. 223-238.
  – Pascal Paillier**,** Composite-residuosity based cryptography: An overview, *Cryptobytes*, **5**(1):20-26, Winter/Spring 2002.

Homomorphic encryption

- Public key encryption
  – Given E(x) it is possible to compute, without knowledge of the secret key, E(c·x), for every c.
  – Given E(x) and E(y), it is possible to compute E(x+y)
- Actually, we can define it for any group operation $\circ$
  – Namely, Given E(x) and E(y), it is easy to compute E(x $\circ$ y)
- Applications
  – Voting
  – Many cryptographic protocols, e.g. keyword search, oblivious transfer…

Homomorphic encryption

- "Standard" public key encryption schemes support Homomorphic operations with relation to multiplication
  – RSA
    - Public key: N, e. Private key: d.
    - $E(m) = m^e \bmod N$
    - $E(m_1)\, E(m_2) = E(m_1 \cdot m_2)$
  – El Gamal
    - Public key : p (or a similar group), $y=g^x$. Private key: x.
    - $E(m) = (g^r, y^r m)$
    - $E(m_1) \cdot E(m_2) = E(m_1 \cdot m_2)$

Modified El Gamal

- $E(m) = (g^r, y^r g^m)$
- $E(m_1) \cdot E(m_2) = (g^r, y^r g^{m_1 + m_2}) = E(m_1 + m_2)$
- Decryption reveals $g^{m_1 + m_2}$
- Computing m_1+m_2 is only possible if discrete log is easy. For example, if m_1+m_2 is relatively small.

Types of public key cryptosystems

- Mostly based on number theory assumptions.
- Can be categorized in one of three main families:
- Based on root extraction over finite Abelian groups of secret order
  - Root extraction is easy when the group order is known
  - RSA, Rabin.
- Based on exponentiation over finite cyclic groups
  - Depend on discrete log and Diffie-Hellman assumptions
  - The trapdoor is knowledge of the discrete log of a public group element
  - El Gamal
- Based on residuocity classes
  - Godwasser-Micali, Paillier.

Paillier's cryptosystem

- Based on composite residuocity classes
- A very useful building block for cryptographic protocols

- Mathematical background
  - $n = p \cdot q$. p,q are large primes.
  - $\phi = \phi(n) = (p-1)(q-1)$
  - $\lambda = \lambda(n) = \mathrm{lcm}(p-1,q-1)$ Carmichael number
  - We work in the group $Z^*_{n^2}$, which has $\phi(n^2)=n\phi(n)$ elements.
  - For any $w \in Z^*_{n^2}$,
    - $w^{\lambda} = 1 \bmod n$
    - $w^{n\lambda} = 1 \bmod n^2$

N^{th} residues

- An integer z is an n^{th} residue modulo n^2 if there exists an integer y such that $z=y^n \bmod n^2$.
- The set of n^{th} residues is a multiplicative subgroup of order $\phi(n)$.
- The number roots of degree n of 1 is n: 1, n+1, 2n+1,…
- Each n^{th} residue has exactly n roots of degree n.

- Decisional Composite Residuocity Assumption:
  - There is no polynomial time algorithm which can decide for n=pq whether a number is an n^{th} residue or not in $Z_n{}^2*$.

  - Homework:
    - Show that this problem is random self reducible.
    - Show that it easy to solve it given a factoring of n.

Composite residuocity classes

- Let $g \in Z^*_{n^2}$ s.t. the order of g is a multiple of *n*. (For example, g=n+1).
- Then the following mapping is one-to-one and onto:
  - $Z_n \times Z^*_n \rightarrow Z^*_{n^2}$
  - $(x,y) \rightarrow g^x y^n \bmod n^2$
- Namely, for every $w \in Z^*_{n^2}$ there are unique (x,y) such that $w = g^x y^n \bmod n^2$.
  - This $x \in [1,n]$ is called the (unique) residuocity class of w with respect to g, and is denoted by $[w]_g$.
  - All w values with the same x are in the same residuocity class.
  - $[w]_g=0$ iff w is an n^{th} residue.
  - $[w_1 \cdot w_2]_g = [w_1]_g + [w_2]_g \bmod n$

Computing composite residuocity classes

- Let $S_n = \{u \mid u < n^2, u = 1 \bmod n\}$
  - Namely, $u = c \cdot n + 1$.
- For $u \in S_n$, the following function is well defined
  - $L(u) = (u-1)/n$
- It is easy to compute discrete logs in $Z^*_{n^2}$ for elements in S_n:
  - For $u \in S_n$, $L(u^r) / L(u) = r = [u^r]_u$
    - Namely, L(w) / L(u) is the discrete log of w to the base u, or the residuocity class of w with respect to u, $[w]_u$.
  - True since $(1+c \cdot n)^r = 1 + r \cdot c \cdot n + \ldots$

The Paillier cryptosystem

- Initialization:
  - $n = p \cdot q$, $g \in Z^*_{n^2}$. n divides the order of g.
  - Public key: n, g.
  - Private key: $\lambda = \mathrm{lcm}(p-1, q-1)$.
- Encryption:
  - Plaintext: $m \in Z_n$.
  - Select a random $r \in Z^*_{n^2}$.
  - Ciphertext: $c = g^m \cdot r^n \bmod n^2$.
- Decryption:
  - $m = L(c^\lambda \bmod n^2) / L(g^\lambda \bmod n^2)$

Correctness

- Ciphertext: $c = g^m \cdot r^n \bmod n^2$.
- Decryption: $m = L(c^\lambda \bmod n^2) / L(g^\lambda \bmod n^2)$
- Explanation:
  - $c^\lambda = (g^m \cdot r^n)^\lambda = g^{m\lambda} \, r^{n\lambda} = g^{m\lambda} \bmod n^2$

    $\boxed{= 1 \bmod n}$ $\boxed{= 1 \bmod n^2}$
  - $c^\lambda = g^\lambda = 1 \bmod n$
  - Therefore, $c^\lambda, g^\lambda \in S_n$.
  - $L(c^\lambda \bmod n^2) / L(g^\lambda \bmod n^2) = L(c) / L(g) = [c]_g = m$
- Truly additive Homomorphic property:
  - $E(m_1) \cdot E(m_2) = (g^{m_1} \cdot r_1^{\,n}) \cdot (g^{m_2} \cdot r_2^{\,n}) = (g^{m_1+m_2} \cdot (r_1 r_2)^n) \bmod Z^*_{n^2}$ $= E(m_1+m_2)$

Security

- Decisional Composite Residuocity Assumption:
  - There is no polynomial time algorithm which can decide whether a number is an n^{th} residue or not.
  - Corollary: There is no polynomial time algorithm which can decide, given w,g,x, whether $x=[w]_g$
- *Ciphertext:* $c = g^m \cdot r^n \mod Z^*_{n^2}$.
- c is an encryption of m, iff $c=[g]_m$.
- Suppose that there is an algorithm which distinguishes between encryptions of m_1 and of m_2
  - Namely, the algorithm decides, given c,m_1,m_2,g, whether $c=[m_1]_g$ or $c=[m_2]_g$
  - This algorithm solves the decisional composite residouocity problem

Keyword search

- Motivation: sometimes OT or PIR are not enough
- Bob:
  - Has a list of *N* numbers of fraudulent credit cards
  - His business is advising merchants on credit card fraud
- Alice (merchant):
  - Received a credit card c, wants to check if it's in Bob's list
  - Wants to hide card details from Bob

- Can they use oblivious transfer or PIR?
  - Bob sets a table of $N=10^{16} \approx 2^{53}$ entries, with 1 for each of the m corrupt credit cards, and 0 in all other entries.
  - Run an oblivious transfer with the new table…
  - …but Bob's list is much shorter than 2^{53}

Keyword Search (KS): definition

- Input:
  - Server/Bob $X=\{ (x_i,p_i) \}, 1 \le i \le N$.
    - x_i is a keyword (e.g. number of a corrupt credit card)
    - p_i is the payload (e.g. explanation why the card is corrupt)
  - Client/Alice: w (search word) (e.g. credit card number)

- Output:
  - Server: nothing
  - Client:
    - p_i if $\exists i$ s.t. $x_i=w$
    - nothing otherwise

Server: (X_1,P_1) | (X_2,P_2) | … | (X_n,P_n)

Client: w

Client output: (X_j,P_j) if $w=x_j$

- Privacy: Server learns nothing about w, Client learns nothing about (x_i,p_i) for $x_i \neq w$

KS protocols using polynomials

- Tool: Oblivious Polynomial Evaluation (OPE)
  - Server input: $P(x) = \Sigma_{i=0\ldots d}\, a_i x^i$, polynomial of degree d.
  - Client Input: w.
  - Client's output: $P(w)$
  - Privacy: server doesn't learn anything about w. Client learns nothing but $P(w)$.
  - Common usage: source of $(d+1)$-wise independence.

- Implementation based on homomorphic encryption
  - Client sends $E(w), E(w^2), \ldots, E(w^d)$.
  - Sender returns $\Sigma_{i=0\ldots d}\, E(a_i w^i) = E(\Sigma_{i=0\ldots d}\, a_i w^i) = E(P(w))$.

KS using OPE (basic method)

- Server's input $X=\{(x_i,p_i)\}$.
- Server defines
  - Polynomial $P(x)$ s.t. $P(x_i)=0$ for $x_i \in X$. (degree = N)
  - Polynomial $Q(x)$ s.t. $Q(x_i)=p_i|0^k$ for $x_i \in X$. *(k=20?)*
  - $Z(x) = r \cdot P(x)+Q(x)$, with a random r.
    - $Z(x) = p_i|0^k$ for $w \in X$
    - $Z(w)$ is random for $w \notin X$
- Client/server run OPE of $Z(w)$
  - If $w \notin X$ client learns nothing
  - If $w \in X$ client learns p_i
  - Overhead is $O(N)$

Reducing the Overhead using Hashing

- Server
  - defines $L=N^{1/2}$ bins, maps L inputs to every bin (arbitrarily). (Essentially defines L different databases.)
  - Defines polynomial Z_j for bin j. (Each Z_j uses a different random coefficient r for $Z_j(x) = r \cdot P_j(x)+Q_j(x)$.)
- Parties do an OPE of L polynomials of degree L.
  - Compute $Z_1(w), Z_2(w),\ldots, Z_L(w)$,
- Overhead:
  - $O(L)=O(N^{1/2})$ communication.
  - $O(N)$ computation at the server.
  - $O(L)=O(N^{1/2})$ computation at the client.

Reducing the overhead using PIR (slightly more theoretical…)

- Server:
  - Defines $L= N / \log N$ bins, and uses a *public* hash function H, *chosen independently of* X, to map inputs to bins.
  - Whp, at most $m=O(\log(N))$ items in every bin.
  - Therefore, define polynomials of degree m for every bin.
- Client:
  - Does, in parallel, an OPE for all polynomials.
  - Server has intermediate results $E(Z_1(w)),\ldots,E(Z_L(w))$.
  - Uses PIR to obtain answer from bin $H(w)$, i.e. $E(Z_{H(w)}(w))$.
- Overhead:
  - Communication: logN + overhead of PIR. A total of polylog(N) bits.
  - Client computation is $O(m)=O(\log N)$
  - Server computation is $O(N)$