CS 127/CSCI E-127: Introduction to Cryptography

Prof. Salil Vadhan Fall 2013

Lecture Notes 20:

Fully Homomorphic Encryption

Reading.

- Barak–Brakerski blog post:
  `http://windowsontheory.org/2012/05/01/the-swiss-army-knife-of-cryptography/`
- Gentry–Sahai–Waters paper §1

1 Homomorphic Encryption

- Example: textbook RSA encryption
  - $[\mathsf{Enc}_{N,e}(m_1) \cdot \mathsf{Enc}_{N,e}(m_2) \bmod N] = \mathsf{Enc}_{N,e}([m_1 \cdot m_2 \bmod N])$
  - Can multiply encrypted messages without decrypting!
  - Previously discussed as a vulnerability in textbook RSA signatures; here we will see this as a useful feature.
- Informally, we want to say that a (public-key) encryption scheme with message space $\mathcal{M}_{pk}$ and ciphertext space $\mathcal{C}_{pk}$ is *homomorphic* with respect to $f : \mathcal{M}_{pk}^k \to \mathcal{M}_{pk}$ if there is a PPT algorithm $\mathsf{Eval}_{pk}^f : \mathcal{C}_{pk}^k \to \mathcal{C}_{pk}$ such that if $c_1, \ldots, c_k$ are encryptions of messages $m_1, \ldots, m_k$, then $f(c_1, \ldots, c_k)$ is an encryption of $f(m_1, \ldots, m_k)$.
- Above can be formalized in several ways for probabilistic encryption:
  - **Strongly Homomorphic:** For *every* $c_1 \in \mathsf{Enc}_{pk}(m_1), \ldots, c_k \in \mathsf{Enc}_{pk}(m_k)$, $\mathsf{Eval}_{pk}^f(c_1, \ldots, c_k)$ is identically distributed to $\mathsf{Enc}_{pk}(f(m_1, \ldots, m_k))$, where these are random variables taken over the coin tosses of Eval and Enc, respectively.
    - ∗ has "rerandomization" built in
  - **Weakly Homomorphic:** For every $m_1, \ldots, m_k \in \mathcal{M}_{pk}$, $\mathsf{Dec}_{sk}(\mathsf{Eval}_{pk}^f(\mathsf{Enc}_{pk}(m_1), \ldots, \mathsf{Enc}_{pk}(m_k))) = f(m_1, \ldots, m_k)$ with probability $1 - \mathrm{neg}(n)$ over coins of Enc and Eval.
    - ∗ has limited usability. why?
    - ∗ trivial construction:
    - ∗ can rule out trivial construction by requiring ciphertext to be *compact* — not grow with homomorphic evaluation.

- Existing constructions of *fully* homomorphic encryption schemes achieve properties somewhere between the above two definitions.

- Some probabilistic encryption schemes that are strongly homomorphic and are CPA-secure under standard crypto assumptions:
  - El Gamal: homomorphic wrt multiplication in $\mathbb{Z}_p^*$
  - Goldwasser–Micali (KL1e §11.1, PS10): homomorphic wrt XOR
  - Paillier (KL1e §11.3): homomorphic wrt addition in $\mathbb{Z}_N$

- Already these single-operation homomorphic properties are very useful in the construction of cryptographic protocols such as electronic voting (see KL2e §14.2) and private information retrieval (PS 10).

- NB: homomorphic encryption schemes *cannot* be chosen-ciphertext secure.

2 Fully Homomorphic Encryption

- What if we had an encryption scheme that is homomorphic with respect to *both* addition and multiplication?
  - $\Rightarrow$ also homomorphic wrt boolean AND, OR, and NOT (for messages in $\{0, 1\}$):
    - $\mathsf{Eval}_{pk}^{\mathsf{N}OT}(c) =$
    - $\mathsf{Eval}_{pk}^{\mathsf{A}ND}(c_1, c_2) =$
  - $\Rightarrow$ homomorphic wrt to any boolean *circuit* C
    - A boolean circuit $C : \{0, 1\}^k \to \{0, 1\}$ computes on boolean inputs by sequence of AND, OR, and NOT gates).
    - $\mathsf{Eval}^C(c_1, \ldots, c_k)$ can take C as an input and runs in time $\mathrm{poly}(|C|, n)$, where $|C|$ is the number of gates in C and n the security parameter.
  - $\Rightarrow$ homomorphic wrt any polynomial-time algorithm $A : \{0, 1\}^* \to \{0, 1\}$
    - Given an algorithm $A : \{0, 1\}^* \to \{0, 1\}$ that runs in time $t(k)$ on inputs of length k, can construct in time $\mathrm{poly}(t(k), k)$ a boolean circuit $C : \{0, 1\}^k \to \{0, 1\}$ that computes the same function as A, restricted to inputs of length k.
    - $\mathsf{Eval}_{pk}^{A,t(k)}(c_1, \ldots, c_k)$ runs in time $\mathrm{poly}(|A|, k, n, t(k))$, where $|A|$ is the length of the program A.

- An encryption scheme satisfying the above properties is called *fully homomorphic*.
  - Idea first proposed in 1978.
  - First candidate construction 2009 (Gentry).

- Canonical Application: cloud computing with privacy
  - B wants to use a cloud provider A to store and compute on his files $m_1, \ldots, m_k$
  - Worried about privacy, B uploads only encrypted files $c_1 = \mathsf{Enc}_{pk}(m_1), \ldots, c_k = \mathsf{Enc}_{pk}(m_k)$ and his public key pk.
  - Later when B wants to run a program F on his files, he sends F and the runtime t of F to A.

- A computes $\mathsf{Eval}_{pk}^{F,t}(c_1, \ldots, c_k)$ and sends B the result c.
- B decrypts c to obtain $\mathsf{Dec}_{sk}(c) = F(m_1, \ldots, m_k)$.

3 Sketch of GSW Construction

- First attempt:
  - Secret key is a vector $v \in \mathbb{Z}_q^n$.
  - Ciphertexts are $n \times n$ matrices C over $\mathbb{Z}_q$, where q is a poly(n)-bit prime, such that $Cv = \mu v$ (matrix-vector product modulo q) for some $\mu \in \mathbb{Z}_q$. μ is the message encrypted by C.
  - $\mathsf{Eval}^{+}(C_1, C_2) = C_1 + C_2$ (componentwise addition).
  - $\mathsf{Eval}^{\times}(C_1, C_2) = C_1 C_2$ (matrix multiplication).
  - Why is this insecure?

- Fix:
  - Cv is only approximately equal to v, i.e. $\|Cv - v\|$ is small.
  - Solving noisy linear systems conjectured to be hard (and in fact can even be proven to be *average-case hard* based on a *worst-case* assumption about the hardness of estimating the length of the shortest vector in high-dimensional lattices).
  - How to decrypt?
  - Problem: homomorphic evaluation increases noise
    - Can only do a bounded number of homomorphic operations.
    - Many beautiful ideas go into controlling the noise blow-up and "'bootstrapping" such a scheme to allow an unbounded number of homomorphic evaluations.
  - See GSW paper for more details (including proof of security).