CS 4501-6501 Topics in Cryptography	January 14, 2015
Lecture 2	
Lecturer: Mohammad Mahmoody	*Scribe: Will Hawkins, Sakura Lim*

1 Homomorphic Encryption

1.1 Overview

1.1.1 Motivation

Secure cloud computing: A user wants to store their encrypted data in the cloud. Later, they want to operate on that data without revealing those contents to the service provider.

- Data: D
- Encrypted data: $E_k(D)$
- Function: f

Send $E_k(D)$ to the cloud. Send f to the cloud. Get back $f(D)$.

1.1.2 Implications

If something such as homomorphic encryption exists, it can be used to build many of cryptographic primitives. For example, HE implies *collision resistant hash functions* [IKO05]. A shrinking $h(x) \to y$ for $x \in X$ and $y \in Y$ is a *hash function* if $|x|$ is larger than $|y|$.[1] A *collision-resistant* hash function is such no efficient algorithm (given h chosen from a family) can find collisions $h(x) = h(x')$ in h.

For example, HE implies *electronic voting schemes* [BFP$^+$01]. In such a scheme, the votes are encrypted in voting machines and remain encrypted while date is bring encrypted in the intermediate steps, till the final server decrypts the results.

1.2 (Informal) Definition

HE consists of a "secure" encryption scheme and an evaluation algorithm on top of it. An evaluation algorithm yields a function's encrypted output for a particular plain text input given only that input's cipher text. In other words, given y (the encrypted value of x) $Eval(y, f) \to Enc(f(x))$.

1.3 Implementation with Code Obfuscation

1.3.1 Code Obfuscation

We use $\mathcal{O}$ to denote an "obfuscation" function. To meet the strongest definition of code obfuscation *virtual black box* [BGI$^+$01] $\mathcal{O}$ must meet several criteria:

- For any polynomial time algorithm P the obfuscation $O(P)$ remains polynomial time.
- For any program P, $\mathcal{O}(P) \equiv P$. [2]
- The adversary must be able to learn as much from $\mathcal{O}(P)$ (even when they can inspect its implementation) as they can from arbitrary many oracle queries[3] to P.

[1] $|X|$ denotes the size of the set X.

[2] $\equiv$ means functionally equivalent.

[3] In an oracle query of a function, only the output for an input is given. There is no access to the function's algorithm in oracle queries.

1.3.2 Implementation

It is possible to implement HE with Code Obfuscation and a "secure" public key encryption scheme (with K_e to encrypt and K_d to decrypt).

The evaluation function takes y, an encryption of x and f as input. It has K_d embedded in its source code. It uses the following algorithm to compute:

- Decrypt y to x using embedded K_d.
- Calculate $z = f(x)$.
- $z' = K_e(x)$ (it's public, after all)
- Output z'

Because K_d is embedded in the evaluation function, only the obfuscated version of the above code must be distributed. It still needs a formal proof that virtual black-box obfuscation gives a secure (fully) HE through the construction above, but that can be done.

2 Public Key Encryption

Now we take a step back and study some well known schemes that have some homomorphism built in them.

2.1 Definition

Public Key Encryption (PKE) will encrypt a message $m \in M$ where M is the space of all possible messages. A PKE scheme is fully defined as a tuple of algorithms:

- $Gen(r) = e, d$: a key generation algorithm that uses randomness, r. e is the encryption key and d is the decryption key.
- Enc: an encryption algorithm.
- Dec: a decryption algorithm.

(Gen, Enc, Dec) must satisfy the following *syntactic* requirement: $P_r[Enc_e(x) \rightarrow y \Leftrightarrow Dec_d(y) = x] = 1$ when d, e are generated by $Gen(r)$. d will be kept secret and e is shared publicly.

The security requirement of such a scheme is that no "efficient" algorithm can decide whether $y = Enc_e(x_0 \text{ or } x_1)$ is $Enc_e(x_0)$ or $Enc_e(x_1)$ with probability more than $1/2$[4] (random guessing). This is known as (*semantic*) security.

2.2 RSA

For messages in $[0...N]$ for some $N = p*q$ where p and q are two primes. (p, q) are used to derive a decryption key d. For d there is a corresponding encryption key e. d is private; e is public. They are designed so that $e.d =_{mod\ \phi(N)} 1$ which (using basic number theory) implies $(x^e)^d = x^{ed} \equiv_{mod\ N} x$, therefore:

$$\begin{aligned} Enc_e(x) &=_{mod\ N} x^e \\ Dec_d(y) &=_{mod\ N} y^d \end{aligned}$$

These facts can be used to do multiplication homomorphically. Given $E_e(x_0)$, $E_e(x_1)$ and e,

$$E_e(x_0) * E_e(x_1) = x_0^e * x_1^e = (x_0 * x_1)^e = E_e(x_0 * x_1).$$

The above version of RSA, however, is not even "semantically secure" because it is deterministic. A semantically secure public-key encryption scheme is one that no efficient algorithm can distinguish between encryptions of any two messages (chosen by adversary even given the public key) with more than negligible.[5]

[4] Plus some negligible constant.

[5] A negligible function $\mu(n)$ is one that is smaller than any $1/\text{poly}(n)$ for large enough n.

References

[BFP+01] Baudron, Fouque, Pointcheval, Stern, and Poupard. Practical multi-candidate election system. In *PODC: 20th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, 2001.

[BGI+01] Barak, Goldreich, Impagliazzo, Rudich, Sahai, Vadhan, and Yang. On the (im)possibility of obfuscating programs. In *CRYPTO: Proceedings of Crypto*, 2001.

[IKO05] Yuval Ishai, Eyal Kushilevitz, and Rafail Ostrovsky. Sufficient conditions for collision-resistant hashing. In *In Proceedings of the 2nd Theory of Cryptography Conference*, pages 445–456, 2005.