Classical and (Post-)Quantum Theoretical Cryptography

Computation Theory and Algorithms Laboratory

Kai-Min Chung
Associate Research Fellow

The Computation Theory and Algorithms Laboratory's cryptography group performs research on theoretical cryptography, from various classical cryptographic topics to (post-)quantum cryptography. Over the past decades, cryptography has evolved far beyond its original goal of secure communication, and is allowing us to realize more and more complex tasks with desired security. With the development of the Internet, cryptography has become ubiquitous, such as with e-mail, mobile phones, SSL, e-commerce, and e-voting. It has permeated everyday life and is heavily used by many applications.

To give a sense of recent exciting developments, this article focuses on the fundamental primitive of encryption schemes and introduces two fascinating strengthened notions of it that allow computation over encrypted data in different forms. These two notions — fully homomorphic encryptions (FHE) and functional encryptions (FE) — have been achieved and even defined only within the past ten years, and they remain very active research topics. We will introduce both notions by their natural applications and provide some physical analogies to help readers to understand the notions intuitively. Finally, we briefly introduce some of our group's recent research focuses and research activities.

Historically, cryptography emerged from the need of the military/government to have a means of secure communication between military forces or government agencies (such as depicted in the movie The Imitation Game [1]). For example, consider that Alice wants to send a private message (m) to Bob, who is far away, and that the information transmitted may be listened to by an adversarial eavesdropper, Eve. Clearly, Alice cannot send her message m directly, otherwise it will be learned Eve. The goal of secure communication is to ensure that Bob can correctly receive m from Alice while Eve learns nothing about the message m. For example, in online shopping, credit card numbers and security codes must be transmitted to the bank and secure communication is needed to protect the the information against eavesdroppers.

A Public-key encryption scheme (PKE) is a cryptographic primitive to achieve this goal. A PKE consists of three algorithms (KeyGen, Enc, Dec). One can use the key generation algorithm KeyGen to generate a pair of (randomized) keys, called public key (pk) and secret key (sk). Anyone who knows pk can use the encryption algorithm to encrypt a message m to produce a ciphertext ct = $Enc_{pk}(m)$ such that whoever holds sk can decrypt ct to recover m = $Dec_{sk}(ct)$; but those without sk can learn nothing about the underlying message m. Given such PKE, Bob can generate (pk, sk), sending pk to Alice while keeping sk private. Alice can then encrypt m and send the ciphertext (ct) to Bob, who decrypts ct to recover m. Eve, who may learn ct but does not know sk, is guaranteed to learn nothing about m, as desired.

We note that formalizing what it means by "Eve learns nothing about m from ct" is in fact highly non-trivial, and one of the main reasons that Shafi Goldwasser and Silvio Micali received the Turing Award in 2012 (Interested readers are encouraged to look at the following online material: [2,3,4]). We will keep our discussion on an informal and intuitive level, and provide a physical analogy (see Figure 1): One can think of the secret key as a physical key, and the public key as a mold that can be used to produce an unbreakable box (the ciphertext) that can only be opened by sk. Bob can send Alice the mold, who can produce a box to store what she wants to send to Bob (say, jewelry), lock the box, and send it to Bob, who can unlock the box to obtain the jewelry by using his key.

Fully Homomorphic Encryption (FHE)

In the era of cloud computing, it is common that we delegate our data and computation to a cloud server (e.g., Amazon EC2). However, when the data is sensitive (e.g., personal private data, hospital medical records), delegation to a potential untrusted server may compromise privacy. In an abstract scenario, consider that a client wants to delegate computation of a function f on his data m (e.g., statistical analysis of medical records) to a server. Can this be done in a way that the server learns nothing about the data m?