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.

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 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 $m$?
Paradoxical as it may sound, this task can be achieved by fully homomorphic encryption, an encryption scheme that allows computations to be performed over encrypted data. More precisely, it is a PKC with an additional evaluation algorithm Eval, which can take an encryption $ct = Enc_{pk}(m)$ of $m$ and a function $f$ as input, and produce an encryption $ct' = Enc_{pk}(f(m))$ of $f(m)$ as output. Note that without secret key $sk$, one can still learn nothing from $ct$ and $ct'$. This amazing primitive was first constructed by Craig Gentry in 2009 [5]. Using FHE, the client can send encryption $ct = Enc_{pk}(m)$ of his message $m$ to the server. The server can use Eval to evaluate $f$ on $ct$ to produce $ct' = Enc_{pk}(f(m))$, and send $ct'$ back to the client, who can decrypt to learn $f(m)$. Since the server does not know $sk$, it learns nothing about $m$.

As a physical analogy (see Figure 2), we can think of the ciphertext as an unbreakable glovebox that allows one to manipulate objects inside through the gloves. Thus, Alice (the client) can put raw material inside the glovebox and send it to Bob (the server), who can assemble the material into jewelry inside the glovebox and send it back to Alice, who can unlock the box to get the jewelry.

**Functional Encryption (FE).**

While FHE allows any computation on encrypted data to be performed, the computation result remains encrypted in a ciphertext. Functional encryption (FE) allows delicate access control, such that different users can learn different information from encrypted data. For example, consider that a hospital maintains an encrypted medical database $ct = Enc_{pk}(m)$ and authorizes each department to perform different statistical analysis on the database but does not allow each department to learn any additional information from the sensitive medical database. Functional encryption allows the hospital to produce a different functional key $sk[fi]$ for each department $D_i$ such that given $ct$ and $sk[fi]$, $D_i$ can learn $f_i(m)$ from $ct$ but nothing else (Think of $fi$ as the statistical analysis $Di$ wants to perform on the database $m$). In terms of physical analogy (see Figure 3), now Alice can produce different “magic keys” that can open a box of raw materials for different products made of those materials.

---

**Our Research: Computation Model in Cryptography**

Observe that computation is a common theme in the two amazing primitives FHE and FE; it also plays a central role in cryptography, such as in secure computation and obfuscation. But how do we represent/describe a computation, e.g., the statistical analysis example mentioned above? In computer science, a natural thought is to write a program in a certain language, such as C++ or Python, to implement the computation. However, in cryptography, computation is typically represented as a digital circuit (i.e., half adder in Figure 4) composed of logical gates such as AND, OR, and NOT. The advantage of circuits is its simplicity for cryptographic design, since it can reduce to design the solutions for gates and how to compose them (somewhat oversimplifying). However, unlike most programming languages, circuits cannot express for-loops and conditional branches, which

(cont’d on page 16)
results in large description size and poor efficiency. Recently, researchers (including our group) tackled this issue by designing cryptographic solutions for other computation models, such as the RAM (random-access machine) model, which can be viewed as an abstract model for C++ and Python, and thus can avoid the above-mentioned disadvantage.

Taking one step further, note that "cloud" programming such as MapReduce and GraphLab has emerged to tackle "big data" with massive parallelism, which, however, is not captured by the RAM model. Thus, we propose to consider a PRAM (parallel RAM) model to further capture the power of parallelism in MapReduce and GraphLab. We have initiated the study of cryptography for PRAM and designed cryptographic construction, such as functional encryption, secure computation, and obfuscation in the PRAM model.

Our Research: (Post-)Quantum Cryptography

(Post-)quantum cryptography studies how quantum computation affects cryptography. For example, it is known that quantum computation can solve the factoring problem efficiently. Thus, it can also break any cryptographic constructions that are based on the hardness of the factoring problem. In other words, quantum computation gives more power to the adversary to compromise security. The field of post-quantum cryptography tackles this problem by using new (potentially) quantum-secure computational assumptions, such as lattice-based assumptions, to design quantum-secure cryptographic constructions. On the other hand, quantum can also give more power to the honest party and allow cryptographic tasks that are impossible classically. A prominent example is quantum key-distribution (QKD), which is arguably the most important and close-to-practical application of quantum cryptography. Many countries — such as China, Japan, Switzerland, and the United States — have spent billions of dollars on building QKD networks, with several working prototypes already in existence [6]. QKD allows two parties (e.g., two military camps) to create their secret keys assuming authenticated classical communication. Such a task is impossible to achieve classically without quantum power.

From the above examples, we can see that quantum acts as a two-edged sword in cryptography. In fact, quantum cryptography is still in its infancy, and many new directions remain unexplored. Our group is generally interested in exploring such new directions. In particular, our recent focus is to understand the role of quantum information in post-quantum cryptography. Namely, we consider scenarios where the adversary has access to quantum side information (e.g., through a leakage attack), and develop techniques to ensure security against such adversaries.

Our Group Activities

We aim to create an inspiring environment for theory-prone students to gain exposure to research in theoretical computer science. Currently, we hold weekly seminars and host reading groups on select topics in cryptography and related fields. We also invite international visitors to give talks about their research. Recently, in cooperation with Prof. Ho-Lin Chen of NTU and Prof. Chung-Shou Liao of NTHU, we organized Theory Days in Taiwan (http://theoryday.github.io), a one-day event with four hour-long talks on theoretical computer science by researchers from around the world. All activities were open to the public. The event has two related mailing lists:

- Theory-event-announcement: http://ppt.cc/bpgJF

References