# A Review of Homomorphic Encryption and its Applications

Lifang Zhang
Department of Communications and Networking
Aalto University
Espoo, Finland
lifang.zhang@aalto.fi

Yan Zheng
State Key Laboratory on Integrated Services Networks
Xidian University, Xi'an, China
Department of Communications and Networking
Aalto University, Espoo, Finland
zyan@xidian.edu.cn

Raimo Kantoa
Department of Communications and Networking
Aalto University
Espoo, Finland
raimo.kantola@aalto.fi

## ABSTRACT
Homomorphic Encryption (HE) enables meaningful computations on encrypted data without decrypting it. Thus, privacy concerns can be addressed in a satisfactory manner by encrypting data using the homomorphic cryptosystem before uploading the data to a cloud service provider. Recent years have witnessed rapid progress of (HE) development in theory and practice. However, current literature still lacks a thorough review on these new developments and their applications. This paper guides the reader through a journey of these developments by reviewing the state of the art of homomorphic cryptosystems and discussing their advantages, disadvantages as well as applicability. In addition, this work presents the privacy preserving applications of homomorphic cryptosystems in the field of private information retrieval, genomic data, cloud computing and participatory sensing. Moreover, this review further discusses open issues of such applications and future research trends.

## CCS Concepts
**• Security and privacy**

## Keywords
Cryptography, homomorphic encryption, privacy

## 1. INTRODUCTION
Internet of Things (IoT) interconnects "things" such as sensors or mobile devices to collect all types of data and utilizes the data to extract useful information in order to offer advanced and intelligent services for its users. The number of IoT devices has already reached 9 billion and is expected to grow rapidly, reaching 24 billion by 2020 [3]. Such a rapid increase leads to a huge amount of data gathered from these devices, making it impossible to store the data locally.

In addition, in order to make wise use of such big data, the data is going to be further processed, analyzed and mined to form insightful knowledge, which requires strong processing and computation capability. However, IoT end devices are often low cost, and thus suffer from resource and energy constraints. Therefore, the large amount of data collected from IoT devices and its computation need to be outsourced to a third party (e.g. a cloud service provider). However, the collected data from IoT devices in its original form often contains sensitive information such as user location, thus outsourcing such data to a distrusted third party raises privacy concerns. Lack of privacy protection will decrease the adoption of IoT among users and therefore is one of the obstacles to the success of IoT. Thus, to keep the outsourced data practically useful without revealing sensitive data, it is of the utmost importance to develop approaches for outsourcing the collected data in a hostile environment.

Encryption is a well-known privacy-enhancing technique. However, the limitation of this technique arises from necessitating data decryption before any complicated operations can be performed on it. Thus, it is desirable to have such an encryption scheme that allows nontrivial operations on encrypted data without decryption. Such a special encryption scheme is called a homomorphic cryptosystem. The homomorphic cryptosystem allows arithmetic operations to be performed on encrypted data, the result of which is the encrypted result of operations conducted on its corresponding plaintext. Such a cryptosystem is especially desirable for privacy aware applications where a third party needs to perform computation on encrypted data received from its clients, yet without knowing the real contents of the data. For example, when a resource constrained device or a privacy conscious user decides to delegate its computation to a cloud service provider, it will first encrypt the outsourced data to prevent the cloud service provider from reading the plaintext. With homomorphic encryption, the cloud server can perform computations on the encrypted data without knowing the plaintext, and thus data privacy is protected.

Privacy threats play as a strong barrier to the wide adoption of IoT and some other promising technologies, such as cloud computing. This barrier has prompted intensive study on the homomorphic cryptosystems and homomorphic applications in recent years. The main purpose of this paper is to review the recent development of homomorphic techniques and their applications in new application areas such as genomic data storage and sharing. We further discuss a number of open research issues and propose the directions of future research.

MOBIMEDIA 2016, June 18-19, Xi'an, People's Republic of China

DOI 10.4108/eai.18-6-2016.2264201

The rest of this paper is organized as follows. Section 2 formally defines homomorphic encryption (HE) and proposes its evaluation criteria. Section 3 introduces the state of the arts of homomorphic cryptosystems, followed by a study of some privacy preserving applications of HE in section 4. Section 5 proposes open issues and future research trends. Finally, section 6 concludes this paper.

# 2. HOMOMORPHIC ENCRYPTION

## 2.1 Encryption Basics

In cryptography, an encryption scheme encrypts a message that a sender wishes to transmit to a receiver, referred to as plaintext, to a ciphertext that can only be read if decrypted. We can distinguish two kinds of encryption schemes: symmetric encryption and asymmetric encryption. In symmetric encryption, a decryption key is the same as the encryption key. Thus a key must be agreed among the communicating parties before exchanging any message. Therefore, it is impossible for two people who have never met to directly use such a scheme. In contrast, asymmetric encryption, also known as public-key encryption, uses two different keys for encryption and decryption. Although different, the two keys are mathematically linked. The encryption key is made public, whereas the decryption key remains private. Such a scheme is more functional than a symmetric one since there is no need for the sender and the receiver to agree on anything before communications. However, public key encryption schemes are much slower than the symmetric ones. For example, AES, a widely used symmetric scheme, is typically 100 times faster than RSA encryption, which is one of the most prominent public key cryptosystem, and 200 times faster than RSA decryption [19].

## 2.2 Security Notions of Encryption Schemes

We define a cryptosystem here as a tuple consisting of message space, ciphertext space, a key space, encryption and decryption functions. A cryptosystem is computationally secure if it is secure due to the computation cost of cryptanalysis, but would succumb to an attack with unlimited computation capacity. In contrast, a cryptosystem is unconditionally secure if it can resist any cryptanalytic attacks, no matter how strong computation capacity an opponent may possess. The one time pad that combines a plaintext with a randomly chosen key of the same length is the only unconditionally secure cryptosystem in common use. Although such a cryptosystem is provably secure, it is impractical due to the requirement on key size. No other cryptosystems have been proven unconditionally secure. Therefore, the security of any encryption scheme is evaluated based on the computing power of an opponent if one time pad is omitted. To assess the adequacy of a cryptographic system, the first step is to classify the threats to which it is to be subjected. The following threats are some that may apply to a cryptosystem and distinguished according to the capacity of an attacker [18].

***A chosen-plaintext attack (CPA)*** is an attack model where the attacker can choose an unlimited number of plaintexts and obtain corresponding ciphertexts.

***Non-adaptive chosen-ciphertext attack (CCA1)*** is an attack model where the attacker has access to a decryption oracle that takes a ciphertext as input and outputs its corresponding plaintext. But the opponent can only access the decryption oracle before a challenging ciphertext is given. The feature of non-adaptive comes from the fact that queries to the decryption oracle cannot depend on the challenging ciphertext.

***Adaptive chosen-ciphertext attack (CCA2)*** is an attack model where the attacker again can access to a decryption oracle, but this time she can use the decryption oracle even after the challenging ciphertext is given, as long as the attacker does not ask for the decryption of the challenging ciphertext.

We consider two different goals of the cryptosystem here: indistinguishability and non-malleability. Indistinguishability (IND) describes the inability of an adversary to learn any information about plaintext underlying a challenge ciphertext. Non-malleability (NM) describes the inability of an adversary to output a ciphertext $y'$, given a challenge ciphertext $y$, such that the corresponding plaintexts $x$ and $x'$ are meaningfully related (e.g., $x' = 2x$).

Naor [34] suggested a convenient way to organize the definitions of secure encryption: consider various possible goals and various possible attack models separately and then obtain each definition as a pairing of a particular goal and a particular attack model. The two goals and the three attack models can be mixed and matched in any combination, giving rise to six security notions [8]. For example, notion IND-CPA means indistinguishability under a chosen plaintext attack. From the weakest to strongest, we have IND-CPA, IND-CCA1, IND-CCA2, NM-CPA, NM-CCA1 and NM-CCA2. Thus, the strongest security level for an encryption scheme is IND-CCA2, which in particular implies non-malleability.

## 2.3 Homomorphic Encryption

A formal definition of homomorphic encryption is as follows. Let $M$ and $C$ denote the plaintext and ciphertext space, respectively. A probabilistic encryption scheme, which encrypts the same plaintext into many different ciphertexts while keeping the plaintext decodable, is defined to be homomorphic if the encryption function $\mathcal{E}$ satisfies formula (1) for any given encryption key $k$ and some operators $\odot_M$ in $M$ and $\odot_C$ in $C$.

$$\forall m_1, m_2 \in M, \mathcal{E}(m_1 \odot_M m_2) \leftarrow \mathcal{E}(m_1) \odot_C \mathcal{E}(m_2) \quad (1)$$

In formula (1), $\leftarrow$ denotes "can be directly computed from without any intermediate decryption". For example, we say an encryption scheme is additively homomorphic if we consider $\odot_M$ as an addition operator $\oplus_M$, that is

$$\forall m_1, m_2 \in M, \mathcal{E}(m_1 \oplus_M m_2) \leftarrow \mathcal{E}(m_1) \odot_C \mathcal{E}(m_2) \quad (2)$$

Similarly, a scheme is said to be multiplicatively homomorphic if $\odot_M$ is a multiplication operator $\otimes_M$ [19].

The homomorphic encryption schemes that support either addition or multiplication on plaintext are called partially homomorphic encryption (PHE). In contrast, an encryption scheme is called a fully homomorphic encryption (FHE) if it allows $\odot_M$ to be any arithmetic operator including both addition and multiplication. Many PHE schemes have been published and widely applied in many applications. However, the usage of FHE schemes is limited due to low efficiency even though many FHE schemes have been constructed recently in the literature.

Most of the HE schemes are asymmetric, which means they are much slower than the symmetric encryption schemes. Note also that a homomorphic cryptosystem cannot achieve non-malleability. Since under a homomorphic cryptosystem, a ciphertext $y'$ can be computed from a given ciphertext $y$ and decrypted to $x'$ which is meaningfully related to the plaintext of $y$, violating the definition of non-malleability. Thus, a HE scheme cannot reach IND-CCA. IND-CPA is the highest security level it can achieve [19].

## 2.4 Criteria to Evaluate HE Schemes

To evaluate a HE scheme, it is necessary to study its efficiency and security as both of them are essential in practice.

When it comes to efficiency, we observe the length of keys and ciphertext encrypted from a single bit. The length of ciphertext can be expressed by message expansion, which is defined by the ratio of the length in bits of ciphertext to that of plaintext. The message expansion shouldn't be too large since large message expansion will create heavy communication overhead and considerable storage consumption. In addition, we concern the time it takes in encrypting and decrypting a message and the time it takes to perform homomorphic operations on ciphertext. These times are especially important in highly constrained devices such as PDAs, mobile devices, smart cards and RFID tags. Moreover, they are highly important when a HE scheme is utilized to process a huge amount of data, since in this situation slow operations (e.g. encryption, decryption and homomorphic computation) will incur intolerable processing time. Normally, a computationally complex algorithm involves more time consuming computations (e.g. modular exponentiation). Thus, we should evaluate these times according to the computation complexity of their corresponding algorithms.

The security of any encryption scheme is crucial, since it describes how well an encryption scheme can keep our data confidential and private. Usually, an encryption scheme with higher security level can secure our data in a better way. The security of HE schemes can be evaluated by their mathematical structure, since most of them belong to public key encryption based on well-defined mathematical problems.

The range of homomorphic operations ($\odot_M$) supported by an HE scheme affects the possible range of application scenarios. For example, a FHE scheme that allows $\odot_M$ to be any arithmetic operator can better support sophisticated algorithms. Therefore, a FHE scheme can be used in many application scenarios where there are a need to perform complex calculations on encrypted data. In contrast, PHE schemes can only be used to calculate a limited number of simple algorithms such as sum and average, confining its application scenarios.

Therefore, a desirable HE scheme should support complicated algorithms while at the same time provide high efficiency and security. Thus, we suggest evaluating an HE scheme from the perspective of following 6 parameters.

- Message expansion (ME): the ratio of the length in bits of ciphertext to that of plaintext.
- Security: it can be evaluated according to the security notions defined in section 2.2, such as IND-CPA, IND-CCA1, IND-CCA2, NM-CPA, NM-CCA1 and NM-CCA2S.
- Encryption computation complexity (ECC): the computation complexity of encryption algorithm.
- Decryption computation complexity (DCC): the computation complexity of decryption algorithm.
- Homomorphic operation computation complexity (HCC): the computation complexity of the homomorphic operation algorithm.
- Supported homomorphic operations (SHO): the arithmetic operations on plaintext supported by a homomorphic cryptosystem.

# 3. THE STATE OF THE ART OF HE SCHEMES

In this section, we study the state of the art of HE schemes according to the criteria described above. However, this paper does not introduce their underlying mathematical theories. Instead we discuss their advantages and disadvantages.

## 3.1 PHE Schemes

### *3.1.1 RSA Cryptosystem*

Detailed RSA algorithms are as follows.

**RSASetup.** The algorithm randomly chooses two large prime numbers $p$ and $q$ with equal length in bits. It first computes $n = q * p$ and picks a large random integer $d$ such that $d$ is relatively prime to $(p-1) * (q-1)$. The algorithm then computes $e$ such that $e\ d = 1\ mod\ (p-1) * (q-1)$. Then it outputs public key $RSA\_PK = (e, n)$ and secret key $RSA\_SK = (d, n)$.

**RSAEncrypt.** Take as input $m \in \mathbb{Z}_n$ and $RSA\_PK$, the algorithm outputs $c = E(m) = m^e\ mod\ n$ where $c \in \mathbb{Z}_n$.

**RSADecrypt.** Take as input $c \in \mathbb{Z}_n$ and $RSA\_SK$, the algorithm outputs $m = D(c) = c^d\ mod\ n$, where $m \in \mathbb{Z}_n$ [37].

**RSAMul.** Take as input $c_1 \in \mathbb{Z}_n$ and $c_2 \in \mathbb{Z}_n$ and $RSA\_PK$, the algorithm outputs $E(m_1) * \mathrm{E}(m_2) = c_1 * c_2 = (m_1 * m_2)^e\ mod\ n = E(m_1 * m_2)$.

As shown in algorithm **RSAEncrypt**, the ME of RSA cryptosystem is 1. Both RSA encryption and decryption algorithms require only 1 modular exponentiation. Thus, RSA cryptosystem is efficient. However, it only supports homomorphic multiplication. What is worse, it is not IND-CPA secure unless a message is randomly padded but the padding variant loses the property of multiplicative homomorphism. Because of this, RSA cryptosystem is seldom used although it has a multiplicative homomorphic property.

### *3.1.2 ElGamal Cryptosystem*

ElGamal cryptosystem works as follows.

**ElGSetup.** The algorithm first selects $p$ such that $p-1$ has at least one large prime factor. It then chooses a generator $g \in Z_p$ and a number $s$ uniformly from $Z_p$, and then computes $h = g^s$. The algorithm outputs public key $ElG\_PK = (h, g, p)$ and secret key $ElG\_SK = s$.

**ElGEncrypt.** Take as input $m \in Z_p$ and $ElG\_PK$, the algorithm computes $c_1 = g^a\ mod\ p$ and $c_2 = h^a m\ mod\ p$ where $a$ is randomly selected from $Z_p$. It outputs $c = E(m) = \{c_1, c_2\}$ where $c_1 \in Z_p, c_2 \in Z_p$.

**ElGDecrypt.** Take as input $c \in Z_p$ and $ElG\_SK$, it outputs $m = D(c) = {}^{c_2}/_{c_1{}^s}\ mod\ p$ where $m \in Z_p$ [26].

**ElGMul.** Take as input $c_1 \in Z_p$, $c_2 \in Z_p$ and $ElG\_PK$, the algorithm outputs $c_1 * c_2 = \{g^{a_1 a_2},\ g^s g^{a_1 a_2} m_1 m_2\} = E(m_1 * m_2)$.

The ME of ElGamal cryptosystem is 2, as the size of ElGamal ciphertext is double the size of its corresponding plaintext. ElGamal encryption algorithm **ElGEncrypt** needs 2 modular exponentiations and 1 modular multiplication and its decryption

algorithm **ElGDecrypt** requires 1 modular exponentiation and 1 modular multiplication. Therefore, the ElGamal cryptosystem is not as efficient as RSA. It also only supports homomorphic multiplication. On the other hand, the ElGamal cryptosystem offers a security level of IND-CPA, which is the highest security level an HE scheme can reach. In addition, it works for any family of groups for which the discrete logarithm problem is considered intractable. Therefore, ElGamal is a promising candidate for practical HE schemes. Normally, homomorphic addition is more useful than homomorphic multiplication in practical applications. Thus, Cramer et.al [15] proposed an additively homomorphic variant of ElGamal, but this variant suffers from inefficiency when decrypting big data. Moreover, the variant does not preserve the multiplicatively homomorphic property of the original ElGamal cryptosystem.

### *3.1.3 Goldwasser-Micali (GM) Cryptosystem*

Goldwasser-Micali cryptosystem is constructed as follows.

**GMSetup.** The algorithm randomly chooses two large primes $p$ and $q$ and computes $n = p\,q$. It then selects a quadratic nonresidue module $n$ whose Jacobi symbol is 1, denoted as $g$. It outputs public key $GM_{PK} = (n, g)$ and secret key $GM_{SK} = (p, q)$.

**GMEncrypt.** Take as input $m \in \mathcal{P} = \{0,1\}$ and $GM\_PK$, the algorithm selects a random $r \in \mathbb{Z}_n^*$ and outputs $c = E(m) = g^m r^2 mod\ n$, where $c \in Z_n$ and it is a quadratic residue if and only if $m = 0$.

**GMDecrypt.** Take as input $c \in Z_n$ and $GM\_SK$, the algorithm uses the property that $c/p = (-1)^m$ to determine if $c$ is a quadratic residue or not, namely, decide if $m$ is 0 or not [24].

**GMMul.** $E(m_1) * E(m_2) = E(m_1 + m_2)$.

As shown in algorithm **GMEncrypt**, GM cryptosystem encrypts a single bit of plaintext into an integer modulo $n$, which is $l(n)$ bits. To be secure, $n$ must be at least 1024-bit, if not larger. That is the message expansion must be at least 1024 to make the system secure. This large message expansion rules out the GM cryptosystem as an attractive choice for practical usage.

### *3.1.4 Parillier Cryptosystem*

Paillier cryptosystem is constructed by following algorithms.

**PailSetup.** The algorithm randomly chooses two large primes $p$ and $q$ with equal lengths (in bits). It calculates $n = pq$ and selects a random integer $g$ ($g \in \mathbb{Z}_{n^2}^*$) such that $n$ divides its order, namely $n$ divides the smallest positive integer $o$ satisfying $g^o = e$, where $e$ is a unique identity element of group $\mathbb{Z}_{n^2}^*$. Next it computes $\lambda$ according to equation $\lambda = lcm(p-1, q-1)$, where function $lcm$ calculates the least common multiple of $p-1$ and $q-1$. Then it outputs homomorphic public key $HE\_PK = (n, g)$ and homomorphic private key $HE\_SK = \lambda$.

**PailEncrypt.** Take as input $m \in \mathbb{Z}_n$ and $HE\_PK$, the algorithm returns ciphertext $c = E(m) = g^m * r^n\ mod\ n^2$, where $c \in \mathbb{Z}_{n^2}$ and $r$ is randomly selected from $\mathbb{Z}_n^*$.

**PailDecrypt.** Take as input $c \in \mathbb{Z}_{n^2}$, $HE\_PK$ and $HE\_SK$, the algorithm returns plaintext $m \in \mathbb{Z}_n$ according to $m = \frac{L(c^\lambda\ mod\ n^2)}{L(g^\lambda\ mod\ n^2)}\ mod\ n$, where $L(u) = (u-1)/n$; $\forall_u \in \{u < n^2 \mid u = 1\ mod\ n\}$ [35].

**PailMul.** Take as input $c_1 \in \mathbb{Z}_{n^2}$, $c_2 \in \mathbb{Z}_{n^2}$, and $HE\_PK$, the algorithm outputs $c_1 * c_2\ mod\ n^2 = g^{m1+m2} * r_1 r_2{}^n\ mod\ n^2 = E(m1 + m2)$

**PailExp.** Take as input $c \in \mathbb{Z}_{n^2}$, constant $k$ and $HE\_PK$, the algorithm outputs $c^k\ mod\ n^2 = g^{km} * r^{kn}\ mod\ n^2 = E(km)$.

As shown in algorithm **PailEncrypt**, the ME of Paillier cryptosystem is 2. Paillier encryption algorithm **PailEncrypt** needs 2 modular exponentiations, one of which is based on $g$, its computation cost is not too high by judicious choice of $g$ and applying computing constant parameters in advance. Paillier decryption algorithm **PailDecrypt**, requiring 2 exponentiation modulo $n^2$ to power $\lambda$ and a low cost multiplication modulo $n$, may also be achieved efficiently through Chinese Remainder Theorem and constant parameter pre-computation [35]. In addition, Paillier cryptosystem supports homomorphic addition (shown in **PailMul**) and homomorphic constant multiplication (shown in **PailExp**), which suffice to a number of practical applications (e.g. secure electronic voting and Private Information Retrieval). Moreover, Paillier encryption scheme can reach IND-CPA, the highest security level for HE schemes. Because of this, Paillier crytosystem is one of the most widely known and used HE schemes.

Table 1 summarizes the analysis of the above PHE schemes.

**Table 1. Comparison of PHE schemes**

| PHE scheme | ME | Security | ECC | DCC | SHO | HCC |
|---|---|---|---|---|---|---|
| RSA | 1 | Not IND-CPA | 1 ModExp | 1 ModExp | Mul | 1 ModMul |
| ElGamal | 2 | IND-CPA | 2 ModExp +1 ModMul | 1ModExp+1ModMul | Mul | 1 ModMul |
| GM | $l(n)$ | IND-CPA | 1 ModeExp+2 ModMul | 1 ModMul | Add | 1 ModMul |
| Paillier | 2 | IND-CPA | 2 ModExp+1 ModMul | 2ModExp+1ModMul | Add | 1ModMul |
| | | | | | CMul | 1 ModeExp |

## 3.2 FHE Schemes

As mentioned before, a FHE scheme enables arbitrary computation to be performed on encrypted data. FHE have been regarded as the holy grail of modern cryptography for a long time. However, it was not until 2009 that a FHE scheme was construable, thanks to the breakthrough work of Gentry [20]. Gentry proposed the first FHE scheme with a blueprint to construct such a system. Even though Gentry's first proposal based on lattice was rather theoretical than implementable, this blueprint has been instantiated with a number of cryptographic assumptions, yielding progressively simpler and more efficient schemes. In this section, we first describe the ideas and methods behind Gentry's blueprint and its open issues. We then present current three main families of FHE schemes (e.g. ideal lattices based FHE, FHE over the integers and learning with errors based FHE) and discuss the practical developments of FHE. Moreover,

in order to better compare the efficiencies of different HE schemes, we summarize their performance in table 2.

### 3.2.1 Gentry's blueprint and its issues

Gentry's construction consists of three main building blocks, starting with the construction of a somewhat homomorphic encryption (SWHE) scheme. A SWHE scheme is one that can homomorphically evaluate "low-degree" polynomials. More specifically, SWHE schemes allow only a limited number of homomorphic computations (e.g. many additions and a small number of multiplications) on ciphertexts. In all proposed FHE schemes, encryption is a process of adding noise to plaintexts, thus resulting ciphertexts to carry a certain amount of built-in noise. This noise would increase in the process of any homomorphic computation and when it becomes too large, correct decryption is impossible even with the right decryption key. Thus, the purpose of limiting the number of homomorphic operations on ciphertexts in SWHE schemes is to avoid decryption failure resulting from the noise growing too much. To overcome such limitations on the number of homomorphic operations and realize FHE, Gentry's construction proceeds to refresh ciphertexts constantly to reduce noise. Such a process is very costly and called bootstrapping. The final step is to squash the SWHE decryption circuit. Squashing aims to simplify the decryption circuit of a SWHE scheme so as to allow bootstrapping, while keeping the homomorphic capacity of the scheme at the same time. Following this framework, several other works [21, 39, 41] construct FHE schemes by building various SWHE schemes. To construct FHE, all these schemes then progress to the squash and bootstrapping transformations.

Even though Gentry's blueprint and its various instantiations can achieve FHE, they also leave a number of important questions unsolved. First, constructions based on Gentry's blueprint inherently suffer the disadvantage of low efficiency. The most efficient FHE scheme currently known is still very expensive. More importantly, the per-gate evaluation overhead – namely, the ratio of the time in evaluating a circuit homomorphically to the time in computing it on plaintext inputs – remains as the bottleneck of FHE practical deployments. Second, all the schemes following the blueprint depend on multiple new and relatively unproven cryptographic assumptions. The most problematic one is the non-standard Sparse Subset Sum Assumption (SSSA) used in the squashing part. More open issues regarding FHE can be found in a paper authored by Vaikuntanathan [40]. It walked us through the development of FHE from the mathematic point of view and gave some insightful thoughts about future research directions of FHE schemes. Refer to [40] for more details.

### 3.2.2 FHE based on ideal lattices

Gentry [21] constructed the first FHE scheme based on ideal lattices about which we know relatively little. Stehlé and Steinfeld [39] improved the scheme by analyzing the SSSA against lattice attacks more aggressively and introducing a probabilistic decryption algorithm with low multiplicative degree, thus producing a faster FHE scheme of lower bit complexity. Moreover, by optimizing the key generation and applying batching technique for encryption, Gentry and Halevi [22] implemented a variant of Gentry's FHE scheme [21], which represents the first implementation of Gentry's blueprint. The authors tested the implementation with four parameter settings ranging from "toy" to "large". For the "large" setting, the public key size is 2.3 GB, and it takes 2.2 hours to generate keys, 3 minutes to encrypt a message and 30 minutes to refresh ciphertext. In addition to the poor efficiency, this family of FHE encounters two more difficulties. First, it relies on multiple new and untested complexity assumptions associated with ideals in various rings. Second, it also has to depend on the most problematic assumption SSSA.

### 3.2.3 FHE over the integers

Instead of utilizing ideal lattices as in [21, 39], Dijk et al. [41] published a FHE scheme over the integers. FHE over the integers has the advantage of conceptual simplicity. However, this scheme is far from practical since it involves large public key size and suffers low efficiency. In order to reduce the public key size of the scheme, Coron et al. [14] proposed to generate a public key from its smaller subset on the fly and implemented their system comparable to that of Gentry-Halevi implementation [22]. Specifically, for the "large" setting, their implementation generates 802 MB public key, and takes 43 minutes to generate a key, 2 minutes and 7 seconds to encrypt a message and 3 minutes 55 seconds to refresh ciphertext, as shown in table 2. However, this family of FHE schemes also encounters the difficulty of having to rely on the most problematic assumption SSSA.

### 3.2.4 FHE based on learning with errors (LWE)

Aiming to base FHE schemes on well-studied mathematical problems, Brakerski and Vaikuntanathanz [10] demonstrated the first FHE scheme deviating from Gentry's blueprint. Such a new FHE scheme is based on the standard LWE assumption. They avoided the untested assumptions on ideal lattices by introducing a technique called re-linearization. Moreover, they applied a technique called dimension-modulus reduction to remove the indispensable squashing procedure of Gentry's blueprint, thus preventing its most problematic SSSA. However, this BV scheme suffers poor efficiency by having to depend on the costly bootstrapping and thus has high per-gate evaluation overhead. In order to improve the efficiency of BV scheme, Gentry et al. [9] constructed a leveled FHE scheme to replace bootstrapping with "modulus switching" technique in managing ciphertext noise. In this scheme, the authors apply bootstrapping and batching to optimize the per-gate evaluation overhead, thus improving the efficiency.

**Table 2 The performance comparison of different HE schemes**

| Scheme | Encryption | Decryption | KeyGen | PK size |
|---|---|---|---|---|
| Paillier (1024b) [32] | 20ms | 45ms | - | - |
| BGV [32] for "large" setting | 3min | - | 2.2hour | 2.3GB |
| DGHV [14] for "large" setting | 2min 57s | 0.05s | 43min | 802MB |

### 3.2.5 Practical HE

Another line of FHE development is to investigate a practical HE, as opposed to inefficient FHE. Since SWHE schemes provide much better efficiency guarantee than their FHE counterparts, practical homomorphic encryption focuses on practical applications with concrete mathematical functions. Lots of functions such as average and linear regressions require a limited number of multiplications on ciphertext but already suffice for some useful applications. Starting with some basic functions and statistical tools that people want to compute on data, some efficient SWHE implementations already existed and found their way in some useful applications [27]. For example, Lauter et al. [33] implemented BGV [9] scheme in the Matrix Algebra on GPU and Multicore Architectures (MAGMA) through optimizations and concrete parameters settings, making the scheme comparatively efficient. Moreover, efforts have been made to

implement a leveled homomorphic encryption scheme instead of implementing SWHE. A leveled homomorphic encryption scheme, an important relaxation of FHE, is one that sets parameters particularly for a certain computation in order to be able to evaluate it. Specifically, given a particular computation to evaluate, leveled homomorphic encryption avoids the costly bootstrapping step by selecting scheme parameters in a particular way, achieving high efficiency for the specific computation. Even though leveled homomorphic encryption is not as practical as SWHE schemes, they are quite useful. Moreover, they enjoy the flexibility of FHE schemes since they enable the homomorphic evaluation of any function. One example implementation of a leveled homomorphic encryption scheme is shown in [29]. In a word, due to the tremendous efforts made for improving the efficiency of FHE, FHE is already practical on small data sets and functions with relatively low complexity. Despite such a progress, future work is still needed to make FHE more practical and scalable, especially for those functions with high complexity.

# 4. APPLICATIONS OF HOMOMORPHIC SCHEMES

## 4.1 Private Information Retrieval (PIR)

Private information retrieval (PIR) schemes, introduced by Chor et al [12], allow a user to retrieve one of $N$ records from a database while hiding the identity of the record from a curious database operator. More specifically, a user attempting to access $i$th item in a database reveals no information about the index $i$ to the database owner. PIR can solve many privacy issues in practical e-commerce applications and location based services by enabling a user to retrieve a record of his choice from a database in a way that no one, including the database server, can see the identity of the record. Specifically, PIR can prevent a database server (e.g. eBay database center), from learning the items a consumer is buying, thus protecting the shopping preferences of the consumer from being tracked by eBay and being harassed by unexpected advertisements.

Of course, the privacy of a user can be completely guaranteed by sending the whole database to the user, but this method may incur enormous communication costs and intolerable query response time. Chor et al [12] also showed that a PIR scheme can be constructed with sub-linear communication complexity by replicating a database in several servers and allowing users to give separate queries to each server. However, such a PIR scheme incurs storage consumption and requires the servers not to communicate with each other. A good PIR scheme, therefore, should first have considerably low communication complexity and involve no database replication. In addition to these requirements, a practical scheme should also be able to address deducing attack and greedy user problem as defined below.

> **Deducing attack (DA):** a malicious database operator can deduce if a user tries to retrieve $ith$ record by replacing the $ith$ record with nonsense and checking if the user refuses such fabricated answer after executing the PIR scheme.
>
> **Greedy user problem (GUP):** For real applications where a user is required to pay for a record (e.g. financial enterprise), the user should not be able to get more records than he has queried.

When speaking of PIR communication complexity, Kushilevitz and Ostrovsky [28] demonstrated that the replication of a database is unnecessary. Additionally, they presented how to construct single-server PIR protocols with sub-linear communication complexity from any additive HE schemes, representing an important application of HE. In other words, HE helps to construct a good PIR scheme with sub-linear communication complexity and without database replication. In this case, any HE scheme that is additively homomorphic will suffice to construct such desirable PIR protocols. For example, Chang applied Paillier cryptosystem to construct a single-database PIR scheme with logarithmic communication complexity [11].

Table 2 summarizes the PIR schemes mentioned above. As shown in Table 2, Chang's scheme that applies HE improves the communication complexity significantly and requires only one server. However, all these schemes neglect important practical concerns such as the deducing attack and the greedy user problem. Thus, future work should focus on the design of practical and efficient single-database PIR schemes.

**Table 3. Comparison of PIR schemes**

| PIR scheme | Communication complexity | Numbers of servers | DA | GUP |
|---|---|---|---|---|
| Entire database sending scheme | $\mathcal{O}(n)$ | 1 | - | - |
| Chor's scheme [12] | $\mathcal{O}(n^{1/3})$ | 2 | - | - |
| Chang's scheme [11] | $\mathcal{O}(\log n)$ | 1 | - | - |
| $n$: the size of the whole database. | | | | |

## 4.2 Privacy of Genomic Data

The advances in genome sequencing enable and promote the development of personalized genomic medicine, which represents as a significant paradigm shift and a major trend in health care. However, this remains only a prospect for now until we can fully untangle the complex relationship between genome and health, which necessitates lots of scientific research. To perform such a research, collecting a considerable large amount of genetic data is a must but is not an easy task because of legal restrictions over the use of genetic information. The privacy regulations restrict the ability of hospitals and research groups to store genetic data, but no law stops individuals from sharing their own information. Based on this fact, some well-known corporations such as Apple, Google and Microsoft are seeking to make profits by utilizing apps on mobile devices to supply health care services and products. For example, Apple released a platform called ResearchKit that allows the integration of genetic information into its medical research. Another example is Google Genetics that offers cheap storage spaces for its customers to store their DNA data. Microsoft has also been analyzing genetic information by introducing a product like HealthVault, attempting to create a health-data platform for consumers and doctors [2]. Uploading genetic information via mobile devices enjoys many advantages: collecting a considerable large number of DNA data in a shorter time than it used to be with the traditional way, reducing the cost of data collection as well as making ongoing contacts easy. Yet disclosing personal genetic data raises many serious privacy concerns since a genome uniquely identifies its owner and carries

sensitive information of its owner, such as his ethnic heritage, his proneness to many physical and mental health conditions and other phonotypic traits. Moreover, the privacy issue of genetic data is exacerbated by the fact that the genomes of any two closely related individuals are highly similar. Such a fact implies that disclosing the genome of a person would cause the leakage of the genomic information of his close relatives. Thus, the genomic data privacy must be addressed seriously before we can enjoy the benefits of personalized genomic healthcare [5]. Due to this reason, the genomic privacy has been intensively studied in recent years.

HE schemes can be used to address the genomic privacy concerns. In this case, two most important properties must be considered regarding the HE scheme. That is its efficiency and the set of homomorphic computations it can support, since normally the amount of genetic data is huge and genetic algorithms are usually sophisticated.

There have been many attempts to preserve the genomic privacy by using HE. Ayday et al. [6] introduced a privacy-preserving scheme to utilize the genomic data in medical tests and personalized medicine. The scheme applies a modified Paillier cryptosystem and proxy re-encryption to perform scientific investigations on integrated genomic data. In the scheme, the privacy of a patient is guaranteed by computing the probability that the patient will develop a disease over his encrypted genomic data instead of the plaintext, by applying the homomorphic properties of a modified Paillier cryptosystem. Ayday et al. [7] proposed another privacy preserving system to evaluate the risk of developing a disease based on genomic and non-genomic (e.g. clinical and environmental) data. In this scheme, the privacy of the genomic data is also ensured by a modified Paillier cryptosystem. However, PHE involves only a single operation on ciphertexts (e.g., either additions or multiplications, not both), thus the evaluation algorithm it can support is limited.

FHE, on the contrary, allows us to evaluate arbitrary arithmetic circuits over encrypted data, providing more genomic queries than the PHE schemes do. Kim and Lauter [27] secured the computation of Hamming distance and approximate Edit distance between DNA sequences by using practical HE schemes. Lauter et al. [29] applied a FHE scheme to perform some fundamental genomic algorithms that are frequently used in genetic association studies on encrypted genetic data. However, these approaches can only evaluate polynomials with a small degree since current implementations of FHE are normally inefficient. A practical privacy preserving system that can handle all kinds of genomic queries is contingent on the efficiency improvement of FHE.

In a word, PHE does not suffice for complicated genomic algorithms even though it is more efficient than FHE. On the other hand, FHE is not efficient enough. Thus, future work must focus on improving the efficiency of FHE schemes or extending the homomorphic operations a PHE scheme can support.

## 4.3 Privacy in Cloud

The development of cloud storage and computing platforms coupled with advances in networking technology has promoted both companies and individual users to outsource their data storage and computing needs. However, by depriving clients of direct control over their data, the cloud platforms raise many new privacy issues that present a major barrier for businesses and consumers to adopt such cloud services. These new privacy issues include data privacy of consumers, function privacy, query privacy and server privacy. In this section, we investigate the development of HE for addressing those privacy concerns.

The data privacy regarding consumers can be satisfactorily addressed by encrypting the outsourced data. But encrypting data under standard encryption systems such as AES secures the data at the expense of making it unusable. In other words, the servers that power a cloud have little ability to operate on the encrypted data in a way that can produce meaningful results. An excellent way to address the privacy concerns is to encrypt the data using such an encryption scheme that allows meaningful computation on encrypted data, namely, a HE scheme. Thus HE schemes have been known for a while as the Holy Grail of cloud computing security and privacy. For example, Microsoft applied HE to devise a prototype storage system for cloud platforms that can perform statistical analyses on encrypted data despite never decrypting it. The prototype, regarded as the most practical example yet of HE, protects the privacy of consumers since their data would always remain encrypted [1]. However, to this end, only data privacy of consumers is regarded, some other privacy issues such as function privacy, query privacy and server privacy remain neglected.

In some potential application scenarios such as the financial industry, both the data and the function to be performed on the data are private and should be protected from cloud servers. For example, data about corporations such as their stock price or their performance is a necessary prior knowledge when making investment decisions. Computing functions based on new predictive models for those data may also be a valuable intellectual property since such functions may be the products of costly research done by financial specialists. Thus, it is reasonable and even important for a company to keep these function models private in order to preserve their advantage in their future investment [33]. However, little research focuses on protecting such function privacy, thus deeper exploration is needed.

In the application scenarios where data owners outsource both data and querying services to a cloud service provider, not only should the data be protected against the cloud service provider and querying clients, but the query information of the clients should also be protected against the cloud service provider and the data owner. This is because the query might involve sensitive personal information or reveal confidential business intelligence. Unfortunately, existing studies have addressed data privacy and query privacy separately, and thus cannot solve this new problem arising in cloud computing. Hu et al [25] applied a HE scheme and a secure traversal framework to secure private query processing over a distrusted data cloud server. This work shows us that HE is useful in constructing a system that can protect both data privacy and query privacy at the same time. However, the applied HE scheme increases the size of a plaintext from 1 component to $t$ components, where $t$ is the number of components the plaintext is split into during the encryption process. Moreover, the security of the applied HE scheme is contingent on the number of plaintext-ciphertext pairs an adversary can obtain, instead of a well-studied mathematic assumption. Thus, deeper exploration is still essential in order to yield some efficient and holistic solutions by applying HE schemes to protect both data and query privacy.

In the scenario of outsourcing computation, HE guarantees the client privacy but not the server privacy. A HE scheme that can protect the input of a server– namely, the computation function on encrypted client data – from the client is called a circuit private scheme. In such a scheme, the output of an evaluation function

discloses no more information about the function to the client than the evaluation result. To fight against malicious or semi-honest clients, and reduce the requirement so that the client is allowed to learn some limited information about the function, a number of variants of circuit privacy have been explored [39].

From the above analysis, we can see the great potential of HE in addressing these privacy concerns. However, current research normally addressed them separately, and scheme inefficiency is still a big problem. Thus, future work should make efforts to design schemes that can comprehensively solve all these privacy concerns simultaneously and with high efficiency.

## 4.4 HE in Multiparty Participatory Sensing

Participatory sensing is a data collection and interpretation technology that has attracted extensive applications, such as personal health monitoring, road and traffic condition monitoring, and air quality monitoring. Its key idea is to involve citizens and community groups to contribute sensor information in order to mine it to provide intelligent services for the participants. The improved sensing capability of mobile devices coupled with their sheer ubiquity across the demographic and geographic spectrum paves the way for the participatory sensing applications [23]. The collected sensor data is crucial to any participatory application. However, a malicious party may abuse such data to extract or infer the sensitive information (e.g. habits, acts and relations) of a user, thereby discouraging the user to adopt the participatory sensing applications. Therefore, to achieve widespread adoption of participatory mining applications, the privacy concerns should be addressed first [13]. To this end, HE schemes described above can only solve the privacy issues of the outsourced computing over data collected from a single user. Therefore, they are not applicable for securing the multiparty participatory sensing applications where the body of knowledge is formed according to computations over data contributed by multiple users. Secure Multiparty Computation (SMC) is much more complicated and necessitates multiple harsh requirements as listed below:

**Low interaction complexity.** The server should be able to accomplish the bulk of computations on the ciphertext of participants with the least interaction.

**Low communication complexity.** The communication complexity of participants should rely only on the size of the individual inputs and the output, but neither the complexity of evaluation function nor the total number of system users that could be very large.

**Low computation complexity.** Similarly, the computation complexity of participants should be independent of both the function being evaluated and the total number of system users.

Thus, a good SMC scheme should have low communication, computation and interaction costs. Much work has been carried out on SMC based on threshold homomorphic encryption (THE). THE is basically a HE scheme, with the difference that THE uses N-party protocols to generate keys and to decrypt a ciphertext aggregated from *N* users.

Li et al. [30] proposed a privacy preserving protocol to obtain Sum and Min aggregation of time-series data on the basis of additive HE. The scheme does not require bidirectional communication between the aggregator and users in every aggregation period, realizing low interaction. Rastogi and Nath [36] used a threshold Paillier cryptosystem to build a system called PASTE to realize private aggregation of distributed time-series. Both [30] and [36] protect user's data from a distrusted aggregator. By applying PHE, these two schemes have the advantage of high efficiency. However, using PHE also implies that they cannot support complex aggregation algorithms. Moreover, it cannot fulfill the requirement that the aggregator is not allowed to know the aggregation result. Some other studies [16, 17] used somewhat HE to get some of the most efficient SMC implementations. However, since the schemes are not fully homomorphic, the rounds of interaction are high and linearly changed with the depth of the circuit.

Another line of work [4, 20] focused on extending single-key FHE schemes to multiple parties in the following way: first, the parties run a relatively short key generation protocol to jointly generate a common FHE public key along with its secret key that is shared among all the parties. The parties then encrypt their inputs with the common FHE public key and send the corresponding ciphertext to a powerful computing server (e.g. a cloud service) that will perform a function over the ciphertexts from all the parties. Finally, the parties execute a decryption protocol to recover the evaluation result without learning anything but the plaintext. In [4], the scheme achieves low interaction (5 rounds), and low communication and computation complexity that are independent on the function being performed. However, this line of work requires the parties to be online so as to generate a joint FHE public key once a subset of parties and a function are selected. It is preferred to create a scheme that enables an external entity to perform expensive evaluation computations non-interactively.

López-Alt et al. [31] proposed an on-the-fly SMC protocol to protect the data and intermediate results of a user from being snooped by the server and other users. In this paper, the authors first constructed a multi-key FHE scheme that can operate on inputs encrypted under multiple independent keys. Such a multi-key FHE scheme is capable of decrypting a ciphertext resulting from a multi-key evaluation with the secrete keys of all the users associated with the computation. The authors then constructed an on-the-fly SMC protocol for the parties whose data was involved in the evaluation phase to decrypt the evaluation result cooperatively. The scheme achieves low user communication and computation complexity in the sense that the communication and computation of users rely neither on the function being evaluated nor the number of system users. Compared to [4], this proposal also achieves a lower interaction cost since it does not require users to be online to compute a common PHE public key. Instead, it requires users to be involved only in the phase of uploading their encrypted data and in the final decryption phase. This is qualitatively the best from the perspective of minimizing interaction, since the user interaction in the decryption phase has been shown inevitable. Moreover, the multi-key FHE scheme can dynamically select the participants associated with the evaluation and the function to be evaluated on-the-fly. However, the number of users involved in any computation must be restricted in this scheme. The computation that is not restricted by the number of involved users remains an open problem.

**Table 4. Comparison of different SMC schemes**

| Scheme | Interaction in key generation | Interaction in decryption |
|---|---|---|
| [30] | Y | N |
| [36] | Y | Y |
| [4, 20] | Y | Y |
| [31] | N | Y |

Compared to PHE based schemes [30, 36], the FHE based schemes [2, 31, 20] suffer from low efficiency. Moreover, whether [2, 31, 20] are suitable for participatory sensing applications was not carefully analyzed. We remark that interaction is very important for participatory sensing applications due to user mobility and heterogeneity. So SMC schemes that have high interaction cost are not practical. Table 3 shows that none of the SMC schemes listed in the table can eliminate interaction in both key generation and decryption phases. Therefore, it is fair to say that the problem of building a practical privacy-preserving system for participatory sensing still remains unsolved.

# 5. OPEN PROBLEMS AND FUTURE RESEARCH DIRECTIONS

Many privacy preserving applications based on HE schemes have been proposed in the literature. However, some issues still remain to be solved in the future.

First, PHE should be further studied in order to make it support more homomorphic computations. Current PHE schemes are more efficient than FHE schemes, but most of them support only one type of homomorphic operation. Thus, they are not sufficient for many privacy-preserving applications.

Second, computation cost and efficiency are crucial for FHE schemes to be deployed in real privacy preserving applications (e.g. genomic data privacy). FHE schemes are highly valued in privacy preserving applications, but they suffer from low efficiency. As a result, tremendous amount of efforts has been put on improving the efficiency of FHE. Despite such efforts, FHE is still only practical on small data sets and relatively low complicated functions. FHE schemes that can work on functions with any complexity seem to still have a long way to go before they can be used in practice. Thus, future research efforts should still focus on improving the efficiency of FHE.

Third, a PIR scheme that can be applied in practice is still needed. Currently, with the help of HE, PIR schemes that can achieve low communication complexity without the need to duplicate a database already exist. However, they ignore some important practical security issues such as deducing attacks and greedy user problems. Thus, future work should focus on extending current PIR schemes to a more powerful one that can solve such additional security issues, thus promoting wide adoption of PIR applications.

In addition, in the case of cloud computing, a scheme that can efficiently address multiple privacy issues at the same time is needed. Current research has focused on user data privacy, but function privacy, query privacy and server privacy haven't been paid enough attention or they are addressed separately. Thus, further research must be carried out to address these privacy concerns together in an efficient way.

Moreover, it remains open how to build a practical system that protects privacy for a participatory sensing application.

# 6. CONCLUSION

In this paper, we give a review on HE developments and its privacy preserving applications. We first showed that it is essential to address the privacy issues in some promising technologies such as IoT and cloud computing before they can be widely adopted. HE schemes were considered highly valuable in ensuring data privacy since they allow meaningful computations to be performed in an encrypted form without decrypting the data, especially for outsourcing data to a distrusted party. We then proceeded to review the state of the art of HE schemes and compared their pros and cons from the perspective of efficiency, security and generality in terms of supported homomorphic operations. We also reviewed the HE privacy preserving applications in the field of PIR, genetic data computation, cloud computing privacy, and multiparty participatory sensing. Based on the review, we further summarized the open research issues of HE with regard to privacy preserving applications and proposed the future research directions based on our review and analysis.

# 7. ACKNOWLEDGMENTS

This work is sponsored by the NSFC (grant U1536202), the 111 project (grant B08038), the PhD grant of the Chinese Educational Ministry (grant JY0300130104), the Natural Science Basic Research Plan in Shaanxi Province of China (Program No. 2016ZDJC-06), and Aalto University.

# 8. REFERENCES

[1] A cloud that can't leak. http://www.technologyreview.com/news/424942/a-cloud-that-cant-leak/. Accessed: 2015-11-26.

[2] Here's how apple, google, and Microsoft are trying to get inside your genes. http://fusion.net/story/131791/heres-how-apple-google-and-microsoft-are-trying. Accessed: 2015-11-27.

[3] Aazam, M., Khan, I., Alsaffar, A.A., and Huh, E.N. Cloud of Things: Integrating Internet of Things and cloud computing and the issues involved. In *Applied Sciences and Technology (IBCAST), 2014 11th International Bhurban Conference*, (Islamabad, Pakistan, 2014). IEEE, 414-419.

[4] Asharov, G., Jain, A., López-Alt, A., Tromer, E., Vaikuntanathan, V. and Wichs, D. Multiparty computation with low communication, computation and interaction via threshold FHE. In *Advances in Cryptology-EUROCRYPT 2012,* Springer Berlin Heidelberg. 483-501.

[5] Ayday, E., De Cristofaro, E., Hubaux, J., and Tsudik, G. The chills and thrills of whole genome sequencing. 2013.

[6] Ayday, E., Raisaro, J.L., Hubaux, J.P. and Rougemont, J. Protecting and evaluating genomic privacy in medical tests and personalized medicine. In *Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society,* (Berlin, Germany, 2013), ACM. 95-106.

[7] Ayday, E., Raisaro, J.L., Laren, M., Jack, P., Fellay, J. and Hubaux, J.P. Privacy-preserving computation of disease risk by using genomic, clinical, and environmental data. In *Proceedings of USENIX Security Workshop on Health Information Technologies (HealthTech" 13),* (No. EPFL-CONF-187118). 2013.

[8] Bellare, M., Desai, A., Pointcheval, D. and Rogaway, P. Relations among notions of security for public-key encryption schemes. In *Advances in Cryptology—CRYPTO'98,* Springer, 26-45. 1998.

[9] Brakerski, Z., Gentry, C. and Vaikuntanathan, V. (Leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference,* (Cambridge, MA, USA, 2012)*,* ACM, 309-325.

[10] Brakerski, Z. and Vaikuntanathan, V. Efficient fully homomorphic encryption from (standard) LWE. *SIAM Journal on Computing, 43(2), pp.831-871*. 2014.

[11] Chang, Y.C. Single database private information retrieval with logarithmic communication. In *Information Security and Privacy,* Springer, 50-61. 2004.

[12] Chor, B., Kushilevitz, E., Goldreich, O. and Sudan, M. Private information retrieval. *Journal of the ACM (JACM), 45(6).* 965-981. 1998.

[13] Christin, D., Reinhardt, A., Kanhere, S.S. and Hollick, M. A survey on privacy in mobile participatory sensing applications. *Journal of Systems and Software, 84(11).* 1928-1946. 2011.

[14] Coron, J.S. Mandal, A. Naccache, D. and Tibouchi, M. Fully homomorphic encryption over the integers with shorter public keys. In *Advances in Cryptology–CRYPTO 2011 (pp. 487-504)*. Springer Berlin Heidelberg.

[15] Cramer, R., Gennaro, R. and Schoenmakers, B. A secure and optimally efficient multi-authority election scheme. *European transactions on Telecommunications, 8(5).* 481-490. 1997.

[16] Damgård, I. and Nielsen, J.B. Universally composable efficient multiparty computation from threshold homomorphic encryption. In *Advances in Cryptology-CRYPTO 2003*, Springer Berlin Heidelberg, 247-264.

[17] Damgård, I., Pastro, V., Smart, N. and Zakarias, S. Multiparty computation from somewhat homomorphic encryption. In *Advances in Cryptology–CRYPTO 2012,* Springer Berlin Heidelberg, 643-662.

[18] Diffie, W. and Hellman, M.E. New directions in cryptography. *Information Theory, IEEE Transactions on, 22(6).* 644-654. 1976.

[19] Fontaine, C. and Galand, F. A survey of homomorphic encryption for nonspecialists. *EURASIP Journal on Information Security, 2007*. 15.

[20] Gentry, C. A fully homomorphic encryption scheme *(Doctoral dissertation, Stanford University).* 2009.

[21] Gentry, C. Fully homomorphic encryption using ideal lattices. In *STOC (Vol. 9* pp. 169-178). 2009.

[22] Gentry, C. and Halevi, S. Implementing Gentry's fully-homomorphic encryption scheme. In *Advances in Cryptology–EUROCRYPT 2011 (pp. 129-148)*. Springer Berlin Heidelberg.

[23] Goldman, J., Shilton, K., Burke, J., Estrin, D., Hansen, M., Ramanathan, N., Reddy, S., Samanta, V., Srivastava, M. and West, R. Participatory Sensing: A citizen-powered approach to illuminating the patterns that shape our world. *Foresight & Governance Project, White Paper,* 1-15. 2009.

[24] Goldwasser, S. and Micali, S. Probabilistic encryption. *Journal of computer and system sciences, 28(2).* 270-299. 1984.

[25] Hu, H., Xu, J., Ren, C. and Choi, B. Processing private queries over untrusted data cloud through privacy homomorphism. In *Data Engineering (ICDE), 2011 IEEE 27th International Conference*, (Lower Saxony, Germany, 2011), IEEE. 601-612.

[26] Hwang, M.S., Chang, C.C. and Hwang, K.F. An ElGamal-like cryptosystem for enciphering large messages. *Knowledge and Data Engineering, IEEE Transactions on, 14(2).* 445-446. 2002.

[27] Kim, M. and Lauter, K. Private genome analysis through homomorphic encryption. *BMC medical informatics and decision making, 15(Suppl 5).* S3. 2015.

[28] Kushilevita, E. and Ostrovsky, R. Singal-database computationally private information retrieval. In *Proc. of 38th FOCS*. 1997.

[29] Lauter, K., López-Alt, A. and Naehrig, M. Private computation on encrypted genomic data. In *Progress in Cryptology-LATINCRYPT 2014*, Springer International Publishing, 3-27.

[30] Li, Q., Cao, G. and La Porta, T. Efficient and privacy-aware data aggregation in mobile sensing. *Dependable and Secure Computing, IEEE Transactions on, 11(2).* 115-129. 2014.

[31] López-Alt, A., Tromer, E. and Vaikuntanathan, V. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, (NY, USA, 2012), ACM, 1219-1234.

[32] Malina, L. Hajny, J. Fujdiak, R. and Hosek, J. On perspective of security and privacy-preserving solutions in the internet of things. *Computer Networks, 102, pp.83-95*. 2016.

[33] Naehrig, M., Lauter, K. and Vaikuntanathan, V. Can homomorphic encryption be practical? *In Proceedings of the 3rd ACM workshop on Cloud computing security workshop,* ACM, (Chicago, USA, 2011), 113-124.

[34] Naor. M, private communication, March 1998.

[35] Paillier, P. May. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in cryptology—EUROCRYPT'99.* Springer Berlin Heidelberg, 223-238. 1999.

[36] Rastogi, V. and Nath, S. Differentially private aggregation of distributed time-series with transformation and encryption. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data,* (Indianapolis, USA, 2010), ACM, 735-746.

[37] Rivest, R.L., Shamir, A. and Adleman, L. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM, 21(2).*120-126. 1978.

[38] Smart, N.P. and Vercauteren, F. Fully homomorphic encryption with relatively small key and ciphertext sizes. In *Public Key Cryptography–PKC 2010.* Springer Berlin Heidelberg, 420-443.

[39] Stehlé, D. and Steinfeld, R. Faster fully homomorphic encryption. In *Advances in Cryptology-ASIACRYPT 2010 (pp. 377-394)*. Springer Berlin Heidelberg.

[40] Vaikuntanathan, V. Computing blindfolded: New developments in fully homomorphic encryption. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium,* IEEE, 5-16.

[41] Van Dijk, M. Gentry, C. Halevi, S. and Vaikuntanathan, V. Fully homomorphic encryption over the integers. In *Advances in cryptology–EUROCRYPT 2010 (pp. 24-43)*. Springer Berlin Heidelberg.