Processing Over Encrypted Data: Between Theory and Practice

Eyad Saleh
Hasso Plattner Institute
Potsdam, Germany
eyad.saleh@hpi.de

Ahmad Alsa'deh
Birzeit University
West Bank, Palestine
asadeh@birzeit.edu

Ahmad Kayed
Middle East University
Amman, Jordan
akayed@meu.edu.jo

Christoph Meinel
Hasso Plattner Institute
Potsdam, Germany
christoph.meinel@hpi.de

ABSTRACT

Data encryption is a common approach to protect the confidentiality of users' data. However, when computation is required, the data must be decrypted before processing. The decryption-for-processing approach causes critical threats. For instance, a compromised server may lead to the leakage of data or cryptographic keys. On the other hand, data owners are concerned since the data is beyond their control. Thus, they look for mechanisms to achieve strong data protection. Accordingly, alternatives for protecting data become essential. Consequently, the trend of processing over encrypted data starts to arise along with a rapidly growing literature. This paper surveys applications, tools, building blocks, and approaches that can be used to directly process encrypted data (i.e., without decrypting it). The purpose of this survey is to provide an overview of existing systems and approaches that can be used to process encrypted data, discuss commercial usage of such systems, and to analyze the current developments in this area.

1. INTRODUCTION

Encryption was previously used to encrypt data during transmission to prevent eavesdroppers from intercepting the communication and revealing the data. In addition, it prevents unauthorized disclosure of confidential data in storage. However, the standard encryption schemes do not allow computations over encrypted data without access to the decryption key. Furthermore, disclosing the decryption key to the server has drawbacks, mainly, the leakage of the key if the server is compromised [46]. Thus,the security challenges for cloud cannot be addressed effectively by classical encryption algorithms. Those challenges can be classified into three groups: Privacy of data (i.e. How to secure shared data), privacy of programs (i.e. How to preserve programs' functionality without leaking their secrets), and integrity of computations (i.e. How to outsource computations over encrypted database for authorized users). Therefore, in the modern era, and motivated by the increasing adoption of the cloud model, the need and possibility of processing over encrypted data is highly desirable.

Developing new constructions that allow operations directly on encrypted data was firstly introduced by Rivest et al. in 1978 [77]. The main hypothesis was that useful privacy homomorphisms (i.e., encryption schemes) may exist to support processing data while being encrypted. They discussed some examples of basic operations that could be applicable, such as addition on integers. In 1985, Blakley and Meadows followed Rivest approach and proposed an encryption scheme that supports some statistical operations such as sum and average [7]. Despite the previous initiation efforts, Feigenbaum in 1986 and Abadi et al. in 1987 can be considered as the first proposals to discuss the concept of processing over encrypted data in its general form, and the first to use formal definitions and strict security requirements [1, 31].

However, the hype of processing over encrypted data did not receive a considerable attention by the database community until 2002, when Hacigümüs et al. discussed the idea in the context of database applications [51]. A restricted version that focuses only on search over encrypted documents has been previously published by Song et al. in 2000 [81]. Since then, a rapidly growing literature evolved, and yielded to several approaches and solutions, such as *Fully Homomorphic Encryption (FHE)* [37], *CryptDB* [71], *CloudProtect* [28], *Silverline* [72], *Cipherbase* [5], *TrustedDB* [6], and *Blind Seer* [69]. However, literature is still evolving and the status

of this new paradigm is yet to be well established. Therefore, we believe that there is a strong need for such a survey that provides a comprehensive view on the developments and advances in this area.

An earlier survey of search over encrypted data has been introduced by Hacigümüs et al. in 2007 [49]. Another survey of homomorphic cryptosystems was also presented by Fontaine and Galand in 2007 [32]. Additionally, In 2013, Ravan et al. wrote a survey paper that introduced some methods for searching on encrypted data and compared between these methods in terms of performance and security level [73]. Although these surveys are helpful, still they focus only on partial issues of the topic. Therefore, our survey provides more in-depth coverage of the topic and presents the current advances in this topic.

Since the objective of this survey is to be a self-contained reference, we include a background section that briefly overview the main encryption categories. In Section 3, we discuss the importance of cryptography, present a detailed description of the homomorphic schemes that are used today, and highlight why they are critical in the cloud environment. Then, the recent advances of processing on encrypted data is presented in Section 4. Section 5 discusses the commercial use of cryptography and processing over encrypted data. Finally, we discuss the limitations and open issues, and conclude the survey in Section 6 and 7 respectively.

2. BACKGROUND

Encryption techniques are used for ensuring the information secrecy. The encryption algorithms can be classified into two categories: (1) symmetric encryption and (2) asymmetric encryption. With symmetric or single-key encryption, the sender and recipient share a single secret key; and they can encrypt and decrypt all messages with this secret key. The symmetric encryption algorithm takes as an input the message (plaintext) and performs various substitutions and transformations on the plaintext based on the secret key value to produce the scrambled message (ciphertext). The two most important symmetric cryptographic algorithms are Data Encryption Standard (DES) and Advanced Encryption Standard (AES). The main challenge with the symmetric encryption is the problem with secret keys exchanging over the Internet. If the secret key falls in an adversary hands, encrypted messages by this secret key can be revealed. One solution to the secret keys exchange problem is the use of asymmetric encryption.

Asymmetric encryption, also known as two-key or public-key encryption uses two related keys for encryption and decryption, a public key and a private key. A private-key known only to one party and a public-key is made freely available to other parties. If Alice encrypts a message by using the Bob's public-key, only Bob can decrypt it using his matching private-key. This means that publishing the public-key on the Internet is safe. If Alice prepares a message to Bob and encrypts it using her private-key, Bob can decrypt the message using Alice's public-key. Because only Alice poses her private-key, the encrypted message with her private-key serves as digital signature. Therefore, the public-key cryptosystems have profound consequences on confidentiality, key exchange, and authentication (digital signature). The most widely used general purpose public-key algorithm is RSA scheme. Public-key algorithms are based on mathematical functions, therefore they are computationally heavy.

The computational overhead of current public-key encryption schemes keeps the need for symmetric encryptions because it is faster than the asymmetric encryptions. Diffie state that "the restriction of public-key cryptography to key management and signature applications is almost universally accepted" [29]. In practice, asymmetric encryption used to encrypt small blocks of data, such as encryption keys, while symmetric encryption used to encrypt the contents of blocks or streams of data of any size.

To use asymmetric encryption, there must be a way for the communicating parties to discover other public keys. Therefore, the digital certificates are in use. A certificate is a package that provides information to identify a server or a user. It contains information, such as the certificate holder name, the organization that issued the certificate, the holder's e-mail address and country, and the holder's public key. The digital certificate is forgery resistant and can be verified because it was issued by a trusted certificate authority (CA). When a client want to securely communicate with a server, it sends a query over the network to the sever asking for its certificate. The server responds with a copy of its certificate to the client. The client can extract the server's public-key from the certificate and verify if it is genuine and valid by using CA's public-key.

3. CRYPTOGRAPHY IN THE CLOUD

Recent surveys showed that security and privacy concerns are among the major barriers for cloud adoption [74,76]. Utilization of cryptography in the cloud can be seen as a potential candidate to the

data confidentiality problem. Here, we discuss the recent advances of cryptography in this context.

3.1 Functional Encryption

Originally, the authorized entity who has the decryption key can decrypt and read the encrypted data. Thus, conventional encryption schemes are *all-or-nothing*, where the encrypted data is useless without knowing the decryption key. However, in many contemporary scenarios, such as complex networks and cloud computing, more fine-grained encryption approach is needed to offer more functionality. In some cases, the data owner needs the ability to control not only who should access the encrypted data but also what should they see. To address this problem, the cryptographic community develop what is known as *functional encryption*.

Functional encryption (FE) is a novel public-key encryption scheme that allows both access control flexibility and selective processing on the encrypted data. FE supports having multiple restricted secret keys of the encrypted data, and allows the secret key holder to learn a specific function of the encrypted data but nothing else about the data. For example, consider a financial data for a company uses the cloud encrypted in away that only employees of the finance department working in the headquarter are allowed to decrypt. In the past decade, cumulative efforts have been made to enable fine-grain access control, which resulted in offering some derivatives of FE, such as *Attribute-Based Encryption (ABE)* and *Identity-Based Encryption (IBE)* [10, 16, 23, 48, 56, 67, 78]. More general notion and framework for functional encryption system that offers selective computation have been published in [15, 65].

In a functional encryption system, the data is encrypted once and the appropriate secret keys with different decryption capabilities are distributed to different users according to arbitrary functions that control what each user should learn from the ciphertext. If a user has a key Sk_{f_1} associated to some function f_1, then he can apply the key Sk_{f_1} to decrypt data and learn the output of applying f_1 but nothing else about the plaintext. On the other hand, another user with a different key can learn entirely different things about the encrypted data.

The enhanced flexibility provided by the functional encryption systems that provides partial access and selective computation on encrypted data is very attractive for many applications, such as searching on encrypted data, partial access control, and selective computation on the encrypted data. Accordingly, much progress has been done to realize secure and efficient ABE schemes, such as [13, 47]. Moreover, Garg et al. constructed functional encryption for general circuits that depends on "multilinear maps" [35]. An example of the efforts toward standardization is publishing RFC5091 [18].

An extensive research has recently been pursued to study the functional encryption (FE) schemes in terms of security, implementations, and applications. In particular, multi-input FE [43], functional signatures [19], Fully Key-Homomorphic Encryption [13], secure FE construction [85] and function-private FE [21]. Nevertheless, the main goal of functional encryption is to build secure and efficient schemes that support a wide class of functions and policies.

3.2 Searchable Encryption

Another interesting approach developed by the community is the *Searchable Encryption (SE)*. SE allows the user to encrypt his data using a private-key and store it in the cloud; then, selectively retrieve segments of his encrypted data using keyword search. One approach of SE is the so-called *secure index*. Informally, the user creates an *Index* I over a database $DB = (m_1, m_2, ..., m_n)$ by using some keywords $KW = (kw_1, kw_2, ..., kw_m)$ extracted from DB and encrypted using a private-key K. Next, the user stores the encrypted database and the secure index in the cloud. Later, the user generates a trapdoor T over KW using K, and requests the server to use T to search the secure index and return the segments of data that match the keyword. A pioneered approach to search directly over the ciphertext was introduced by Song et al. [81]. They introduce several schemes that support both search by sequential scan over an encrypted database (to avoid the overhead of keep updating the encrypted index), and the more sophisticated search using an encrypted index without sacrificing security. For more details on SE, we refer the reader to a recent survey which was published during the time of reviewing this article [17].

3.3 Secure Multi-party Computation

Yao introduced the Multi-party Computation in 1982 [86]. Yao asked: How can two millionaires know who is richer without disclosing their individual wealth to each other. Sheikh et al. formalized the problem in the so-called Secure Multi-party Computation (SMC) [79]. SMC provides private computation over data while reveal only the individual item to the respective owner. Given multiple parties $P_1, P_2, ..., P_n$ involved in a computation of some public function of their private inputs

$D_1, D_2, ..., D_n$, respectively. Each party Pi wants to know the common function $f(D_1, D_2, ..., D_n)$ without disclosing value of its data D_i to other parties. *Ideal* and *Real* models are the two well-known paradigms for SMC. In the ideal model, there is some trusted third party (TTP) among the participants while there is no such assumption in the real model. Worth to mention that in the Data-as-a-Service (DaaS) environment and in large volumes of online transactions, the concept of data privacy and SMC has become a matter of great concern [79].

A survey of the main techniques to secure joint computation over private data while preserving the privacy of their individual items has been introduced by Sheikh et al. [79]. They classified the techniques that solve SMC problems into three main groups: randomization, anonymization and cryptographic. In the randomization method, parties use random numbers for hiding their data. Cliftonet et al. proposed a secure sum protocol that computes the sum of several parties while preserving the privacy of their data [26]. In the anonymization method, TTP is required to hide the identities of the parties. Mishra and Chandwani proposed and extend anonymous protocols to hide the TTP identities [62]. Their main protocol unanimously selects one TTP among all TTPs in the SMC architecture to ensure that no single TTP controls the system and no TTP knows where the computation is taking place. In the cryptographic technique, blocks are built to secure computation [64]. Well-known techniques that use cryptographic blocks are: Yao's millionaires problem, homomorphic encryption, oblivious transfer, and private matching.

Lepinksi et al. stated that cryptographic protocol can undo all of the carefully planned measures designed by the auctioneer to prevent collaborative bidding [58]. They define and construct collusion-free protocols in a model in which players can exchange physical envelopes to guarantee that no new method for players to collude are introduced by the protocol itself.

Finally, Alwen et al. addressed the problem of building collusion-free protocols without using physical channels [4]. They suggested a mediated model where all communication passes through a mediator. The goal is to design protocols where collusion-freeness is guaranteed. Recently, Miers et al. proposed Zero-coin, a cryptographic extension to Bitcoin where their protocol allows fully anonymous currency transactions [61]. Their system uses standard cryptographic assumptions and does not introduce new trusted parties.

Current major problems and solutions for SMC can be classified as follows: Private Information Retrieval, Selective Private Function Evaluation, Privacy Preserving Data Mining, Cooperative, Database Query, Geometric Computation, Intrusion Detection, and Statistical Analysis [79].

3.4 Homomorphic Cryptosystems

Existing encryption schemes can be classified into two main categories in terms of homomorphic properties. Namely, *Fully Homomorphic Encryption* and *Partially Homomorphic Encryption*. Homomorphic is an adjective that describes a special property of an encryption scheme. That property, at an abstract level, can be defined as the ability to perform computations on the ciphertext without decrypting it or even knowing the keys.

3.4.1 Fully Homomorphic Encryption (FHE)

In the cryptography communit's Conviction, FHE was impossible to achieve until 2009, when Gentry announced his new approach [38, 39]. It is considered one of the recent breakthrough of cryptography. FHE supports arbitrary computation over encrypted data and remains secure (achieve semantic security) as well. In his PhD thesis, he discussed how his schemes can be constructed [37]. Before Gentry's achievement, all encryption schemes that preserve a homomorphic property were able to support only a single operation over encrypted data. The main contribution of Gentry's work is the supporting of two homomorphic operations at the same time. Namely multiplication and addition. Correspond to AND ($\wedge$) and XOR ($\oplus$) in boolean algebra. The remarkable value of supporting these two boolean functions is that any computation can be converted into a function that contains only ($\wedge$) and ($\oplus$) as we explained below. Finally, an open-source implementation of FHE is available [53, 54].

In algebraic terms, any computation can be expressed as a boolean circuit. For example, to search for a string in a text file, we can convert both the string and the text file into two sequences of binary digits, then we do a bitwise XOR for every bit of the string, when the result of all bits is 1, then there is no match for the current position of the file; Therefore, we shift one bit to the right and compare again. We repeat this process until the result of comparison is 0, which means that we found a match, or the file ends without a match. Usually, several techniques can be used to convert a function (i.e., computation) into a more simple or efficient one. Furthermore, they can also be used to transform a function to use specific boolean operations. For instance, $\neg A$ can be expressed as A

$\oplus$ *1*, another example would be $A \vee B$, this can be transformed into $\neg(\neg A \wedge \neg B)$ which is equivalent to $((A \oplus 1) \wedge (B \oplus 1)) \oplus 1$. By utilizing such techniques, all functions can be converted into a series of ($\wedge$) and ($\oplus$) operations. This is the basis behind the remarkable achievement of Gentry's work.

Clearly, converting even a simple application into a series of boolean circuits requires enormous number of operations. Moreover, both the complexity of encryption and decryption and the size of the ciphertext hugely grow. Despite that Gentry is trying with the support of his colleagues at IBM to optimize the first version of his work [20, 40, 84], his approach remains very expensive and hence impractical.

3.4.2 Partially Homomorphic Encryption (PHE)

Several PHE systems have been discussed in the literature. Rivest et al. in 1978 was the first to introduce the concept of privacy homomorphism [77]. Then, several researchers follow such as ElGamal and paillier [34, 68]. Here is a discussion of the most well-known partially homomorphic cryptosystems and a summary is shown in Table 1 as well.

ElGamal Cryptosystem: T. ElGamal in 1984 proposed what is known as ElGamal cryptosystem [34]. His scheme is based on problem of solving discrete logarithms. The homomorphic operation that ElGamal supports is the multiplication over encrypted messages. Given two ciphertexts c_1 and c_2 that are encryption of m_1 and m_2, α is a generator of a cyclic group G of order p, where p is a large prime number. $y = \alpha^x$ where x is the secret key, k_1 and k_2 are randoms such that $k_1, k_2 \in \{0, ..., p-1\}$, then

$$c_1c_2 = (\alpha^{k_1}\alpha^{k_2} \ mod\ p, ((m_1 \cdot y^{k_1})(m_2 \cdot y^{k_2}))\ mod\ p)$$

$$= (\alpha^{k_1+k_2}, m_1m_2 \cdot y^{k_1+k_2})\ mod\ p$$

is a valid encryption of $m_1 \,.\, m_2$. One notable drawback of ElGamal scheme is that the size of ciphertext is double the size of the plaintext message. Interestingly, several variants of ElGamal have been proposed, such as Cramer et al. that is homomorphic on the additive operation [27].

Paillier Cryptosystem: This scheme is based on the problem of *composite residuosity class*. i.e., given a composite n and an integer z, it is hard to decide whether there exists y such that $z \equiv y^n \ mod\ n^2$ [68]. The difference of paillier from RSA is the usage of square number as modulus, where $n^2 = pq$ is the product of two large primes. As for homomorphic property, the scheme supports two main operations, addition and multiplication by a constant. Next we describe the addition. Let $c_1 = g^{m_1}r_1^n \ mod\ n^2$ and $c_2 = g^{m_2}r_2^n \ mod\ n^2$, then

$$c_1c_2 \ mod\ n^2 = g^{m_1}r_1^n g^{m_2}r_2^n \ \ mod\ n^2$$

$$= g^{m_1+m_2}r_1^n r_2^n \ \ mod\ n^2$$

is a valid encryption of $m_1 + m_2$

Goldwasser-Micali Cryptosystem: Proposed by Goldwasser and Micali as the first *probabilistic encryption* scheme [44, 45]. Also the first to invent the term *semantic security*. The security of the scheme is based on the complexity of deciding whether a number is quadratic residues with respect to composite modulo $n = pq$, where p and q are two distinct prime numbers. The homomorphic property of the scheme is the support of the addition operation modulo 2, or in algebraic terms the XOR ($\oplus$) operation. Given two ciphertexts $c_1 = -1^{x_1}{r_1}^2$ and $c_2 = -1^{x_2}{r_2}^2$, then

$$c_1c_2 \ = (-1^{x_1}{r_1}^2)(-1^{x_2}{r_2}^2)\ mod\ 2$$

$$= -1^{(x_1+x_2)}(r_1r_2)^2\ mod\ 2$$

is a valid encryption of $x_1 + x_2 \ mod\ 2$.

Benaloh Cryptosystem: Due to the problem of large ciphertext expansion in Goldwasser-Micali cryptosystem, Benaloh proposed his scheme in 1994 that decreased the ciphertext size at the cost of decryption complexity [9]. Benaloh scheme supports both addition and subtraction over ciphertexts. Given two ciphertexts $c_1 = y^{m_1}{u_1}^r \ mod\ n$ and $c_2 = y^{m_2}{u_2}^r$
$mod\ n$, then

$$c_1c_2 = (y^{m_1}{u_1}^r)(y^{m_2}{u_2}^r)\ mod\ n$$

$$= y^{m_1+m_2}(u_1u_2)^r\ mod\ n$$

is a valid encryption of $m_1 + m_2$, and

$$c_1{c_2}^{-1} = (y^{m_1}{u_1}^r)(y^{m_2}{u_2}^r)^{-1}\ mod\ n$$

$$= (y^{m_1}{u_1}^r)(y^{-m_2}({u_2}^{-1})^r)\ mod\ n$$

$$= y^{m_1-m_2}(u_1u_2^{-1})^r\ mod\ n$$

is a valid encryption of $m_1 - m_2$.

Boneh-Goh-Nissim Cryptosystem: This system utilizes the bilinear pairing to supports the homomorphic addition while at the same time allowing the computation of a single homomorphic multiplication of two cipertexts [14]. Let
$c_1 = g^{m_1}h^{r_1} \ mod\ n$ and $c_2 = g^{m_2}h^{r_2} \ mod\ n$, then

$$c_1c_2 \ mod\ n = (g^{m_1}h^{r_1})(g^{m_2}h^{r_2})\ mod\ n$$

$$= (g^{m_1+m_2})(h^{r_1+r_2})\ mod\ n$$

Scheme	Main Homomorphic Properties	Security Assumption
ElGamal [34]	$\boxtimes$	Discrete Logarithms
Paillier [68]	$\boxplus, \boxminus, \boxtimes_c$	Composite Residuosity
Goldwasser-Micali [44, 45]	$\oplus$	Quadratic Residues
Benaloh [9]	$\boxplus, \boxminus$	Quadratic Residues
Boneh-Goh-Nissim [14]	$\boxplus, \boxtimes_{once}$	Subgroup Decision Problem

Table 1: Summary of the most well-known PHE schemes

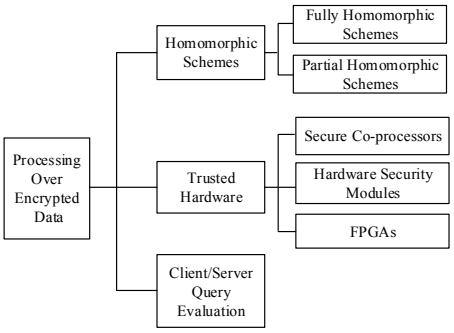

Figure 1: Classification of Processing Over Encrypted Data Models

is a valid encryption of $m_1 + m_2$, and

$$c_1{}^k \ mod\ n = (g^{m_1} h^{r_1})^k \ mod\ n$$

$$= (g^{km_1} h^{r_1 k}) \ mod\ n$$

is a valid encryption of km_1

Based on the above discussion, we argue that homomorphic encryption schemes are possible. However, they lack general computation support since they can perform limited types of operations, and hence the question of designing full functional systems that process encrypted data using only homomorphic schemes is still an open challenge.

4. STATE OF THE ART

As shown in Figure 1, current systems of processing over encrypted data can be classified into three main categories: (i) Systems that utilize homomorphic encryption schemes, (ii) Client-server splitting approaches, and (iii) Trusted-hardware systems. In this section, we discuss systems that fall under these categories.

4.1 Systems Based on Homomorphic

CryptDB is one of the recent "partially" practical systems that utilized several homomorphic schemes to support database functionality [71]. Their approach is basically built on two main ideas. First, use SQL-aware encryption schemes to efficiently execute queries. And second, use onions of encryption and adjust them dynamically at the run-time based on the required functionality. The idea of SQL-aware encryption schemes is a kind of mapping between the operation required and the homomorphic scheme that can support it. However, onions of encryption cause extra overhead. One major draw back of *CryptDB* is the lack of support for *Stored Procedures* (where the SQL code is integrated into the DBMS itself).

CryptDB provides the highest security guarantees when using a probabilistic encryption, which means that encrypting the same value more than once produces different result (even when using the same encryption key). Random(RND) where no computation is supported, and Homomorphic encryption (HOM) where simple computation such as summation is supported, are conventions used by *CryptDB* to refer to such schemes. Better run-time efficiency was achieved by perform aggregation in parallel by simultaneously adding multiple 32-bit integers [36].

To allow more fine-grained operations, *CryptDB* utilizes the scheme proposed by Song et al. to support search over encrypted data [81]. It enables the user to perform search operations over encrypted data. All *text* fields in the database are encrypted using Song et al. approach and stored in the DBMS. By using this approach, they could execute queries to retrieve records that match a certain keyword, such as *SELECT * from Employee where Address Like %Berlin%*

Another important building block of *CryptDB* is the use of *Deterministic Encryption (DET)* that allows equality check operations [8]. DET means that repeating the encryption of any message would always produce the same ciphertext. We cannot achieve semantic security in this scheme, but it still provides high security guarantees. The only information it leaks is the ability to identify which ciphertexts are mapped to the same plaintext, without revealing the actual value of the plaintext. De-

terministic encryption can be constructed by the use of a block cipher such as AES-ECB. Block-size in AES has a fixed length of 128-bit, for lower block-size, such as 64-bit, alternative schemes could be used, such as Blowfish. By utilizing deterministic encryption, the system would be able to execute, for example, queries with equality checks, *GROUP BY*, and some aggregate functions, such as *COUNT*.

Finally, *Order-Preserving Encryption (OPE)* algorithms preserve the numerical order of the ciphertext in a way equivalent to the plaintext [2, 11]. One potential use case of such schemes is to perform range queries on encrypted data. For instance, given two plaintext values m_1 and m_2, where $m_1 < m_2$, then f is order-preserving encryption function if

$$f(m_1) < f(m_2)$$

4.2 Client-Server Splitting Approaches

Several approaches that utilize the concept of client-server query split have been discussed by the community [50–52, 55, 72, 83].

Silverline keeps the data at the server-side confidential by encrypting it in away that is transparent to the application and being able to have some functionality on it as well [72]. *Silveline* proposed to dynamically analyse the application to determine which parts of the data can be functionally encryptable; it assumes that any data that is never interpreted or manipulated by the application is encryptable. For instance, a SELECT query in typical human-resource applications that searches for all records match the employeeID 'Jan' is not required to interpret the actual string 'Jan' and hence can execute the query if it would be encrypted. As for key-management, it divides the users into groups, and assigns a single encryption key to this group, facilitates encryption and information sharing at the same time. While *Silverline* seems to be practical to some extent, the main drawback is that it requires analysis of the application and the data to determine which parts can be encrypted. Such an analysis would be an expensive task; also a repetition of this process will be required whenever a change to the application or upgrade is taking place. Furthermore, major part of the data will still be stored in plaintext, thus privacy and data compromise issues still open.

In contrast to *Silverline*, Hacigümüs et al. proposed to store the entire data in an encrypted form on the provider's side, and introduced an algebraic framework for query rewriting [51]. The framework divides every query into two parts, execute the first part on the encrypted version (i.e., stored on the server's side), and then perform client-side post-processing on the result come from the server. The efficiency of this approach relies on how data partitioning and query splitting and rewriting is accomplished.

Monomi utilizes both techniques, PHE and split client-server query execution [83]. In contrast to CryptDB that focuses on transactional workloads, Monomi is mainly targeting analytical workloads. Since queries are not known ahead of time, and to maximize efficiency, Monomi introduces an optimization designer that chooses an appropriate database design (on the server) according to the target workload. Further, it provides a planner that selects the query execution path for every query. Additionally, it provides some techniques such as per-row pre-computation and pre-filtering. However, Monomi is far from being generally practical for several reasons. First, in real-world enterprise environments, it could be inefficient since queries over analytical workloads contain complex computations that is hard to partition between client and server. Second, Performance cost is very expensive. Queries over large (plain) datasets often have the problem of i/o bottlenecks, imagine adding the cost of using cryptography techniques. Finally, choosing a physical design at the runtime, pre-filtering and pre-computation are complex tasks and depend mainly on the targeted workload. Thus, the task need to be repeated for every workload or application.

4.3 Trusted-Hardware Systems

To perform a computation on encrypted data, the keys need to be present at the server to decrypt the data, compute, and then encrypt again. The drawback of this model is the vulnerability of compromising cryptographic keys. Therefore, several techniques and approaches have been discussed to overcome such vulnerabilities. These approaches use secure, tamper-proof hardware components attached to the server to store cryptographic keys and perform computation over encrypted data [5,6]. Examples of industrial solutions that are in use include secure co-processors, Hardware Security Modules (HSM), and Field-Programmable Gate Arrays (FPGAs).

In contrast to software-based approaches, Trusted DB uses IBM's 4764 cryptographic co-processors to execute SQL queries while maintaining confidentiality [6]. Since it is implemented entirely using hardware components, the overhead of query execution is lower by orders of magnitude in comparison to other approaches. Additionally, They in-

troduced cost-models and insights for the advantages of using trusted, hardware-based solutions for outsourced data processing. Finally, they recommended that trusted-hardware approach be a first-class candidate for remote and secure data management. Different from TrustedDB, Cipherbase key idea is to simulate fully-homomorphic encryption on top of non-homomorphic encryption schemes by using trusted hardware [5].

5. CURRENT INDUSTRY OFFERINGS

Industry offerings can be classified into two categories: encryption at rest and computing on encrypted data. In this section, we discuss the latest technologies provided by the pioneered providers.

Oracle introduced *Transparent Data Encryption (TDE)* that provides data-at-rest encryption [66]. The data will be stored on the file systems as encrypted. Yet, and upon request, it transparently decrypt the data for the application to process. TDE supports both column-level and table-level encryption. However, a single key is used for the entire table regardless of how many columns are encrypted. By default, TDE utilizes AES with 192-bit key as a standard encryption algorithm. However, 128 and 256 bits are also supported. In addition, 3DES can be used as an alternative encryption algorithm. To prevent unauthorized disclosure, the keys for all tables are encrypted with a database-server master key and then stored in a dictionary table in the database. Afterwards, the master key is stored in an external secure module outside the database and is accessible only to the security administrator.

Similar to Oracle, Microsoft offers TDE as well [59]. The main concept of securing data at-rest by utilizing encryption remains the same. However, few differences exist, such as storing the keys for encrypting data in the database boot record in comparison to a dictionary in the case of Oracle. Another major difference is that Microsoft TDE uses three-levels of encryption along with two master keys and one certificate. Namely *Service Master Key (SMK)* and *Database Master Key (DMK)*. First, the SMK is created at the time of SQL Server setup. The *Windows OS-Level Data Protection API (DPAPI)* is used to encrypt the SMK so it remains protected. Second, The DMK is created and then protected by encrypting it using the SMK. Finally, a certificate is generated using the DMK and stored in the master database that is consequently used to encrypt the data encryption key. In addition to *TDE*, Microsoft developed a new Always Encrypted feature for protecting sensitive data, such as credit card number that sorted in Azure SQL Database [60]. Always Encrypted is a client-side technology to ensure that sensitive data is encrypted and decrypted at the client side and the database system does not have access to the encryption keys. Consequently, database administrator or attackers gaining illegal access to the database are not able to retrieve data from encrypted database.

Navajo Systems (acquired by Salesforce in 2011) [33], CipherCloud [25], and SQLCipher [82] all provide techniques to encrypt enterprise data before storing them in the cloud. For instance, CipherCloud offers, in addition to key management and other things, what they call *Tokenization*. It generates a random values to substitute the original data and store them in the cloud. The mapping between the random values and the original data is stored at the client's side. Finally, Google is implementing and testing some partially homomorphic encryptions in a new command-line client-tool that accesses their BigQuery service [75].

The above industry offerings are mainly targeted to protect data at-rest and in transit. Although we introduced Microsoft Always Encrypted and Skyhighy, supporting functionality over encrypted data, other than basic search or limited queries, remains a challenge and an open issue for both industry and academia.

6. LIMITATIONS AND OPEN ISSUES

We point out inherited limitations of current schemes and discuss some open problems in the domain of processing over encrypted data.

6.1 FHE is Impractical

Despite the improvements that follow Gentry's scheme [20, 22, 42, 84], current proposals of FHE are far from being practical due to the expensive cost to perform operations. For example, An evaluation performed by Gentry et al. in 2012 for AES-128 circuit showed that it cost about 40 minutes per AES block on an Intel core i5-3320M machine running at 2.6GHz with 256 GB of RAM [41]. The computation model required by FHE is complex due to the need of converting the application into a boolean circuit that may results in a very large, non-trivial one. Therefore, designing an efficient and practical FHE scheme remains an open issue.

6.2 PHE Schemes are Limited

In contrast to FHE, PHE schemes are more efficient. This is due to the support of only limited functionality. For instance, paillier takes about $0.005\,ms$ to perform an addition on two ciphertexts [71]. PHE schemes are crucial for systems to

process encrypted data because of their practicality. However, they only support partial computations, and hence, cannot be used to build complete functional systems. Yet, and motivated by the previous schemes and advances in cryptography, we believe that more schemes to come that can help in bridging this gap.

6.3 Strong Order-Preserving Encryption

Order-Preserving Encryption (OPE) schemes in [2,11] are shown to be insecure and reveal about half of the plaintext [70]. An extension to improve the security of [11] was presented by the same authors in [12]. However, the leakage of nothing except order remains questionable. More recent approaches claim that their schemes achieve ideal security of OPE (i.e., they leaks nothing but order) [57,70]. Finally, although *SkyhighNetworks* implemented OPE solution in their cloud security [80], the security of the best practical OPE schemes is still not well understood [24].

6.4 Trusted-Hardware is Expensive

In spite of the fact that the benefits of hardware-based solutions, they require fundamental changes to the service provider's model. Consequently, their usage is limited to specific environments. However, and due to the limitation of software-based solutions, the integration of trusted-hardware with commodity servers has received a considerable attention recently. In order to bring the trusted-hardware model into practice, we believe that in the near future, several IaaS providers will start to offer secure co-processors, FPGAs, and HSM in their settings. A more detailed discussion about processing on encrypted data using secure hardware is presented in [30, 63].

7. CONCLUSION

This paper discussed the main applications, tools, and techniques for processing over encrypted data. We reviewed both PHE and FHE schemes. PHE encryption schemes that preserve homomorphic property can be discussed from two different perspectives. On one hand, it is a desirable property that allows the user to perform computations on the encrypted data without decrypting it or even knowing the decryption keys. An interesting example for such a need is electronic voting. On the other hand, it is perceived as a drawback or a weakness in the encryption scheme since it cannot satisfy indistinguishability under adaptive chosen ciphertext attack (IND-CCA2) requirements, and hence, can be broken. This is drawn from the fact that PHE schemes are malleable by design. For instance, a chosen-ciphertext attack by Ahituv et al. was reported against a homomorphic scheme where the addition operation is supported [3]. Unlike PHE, and to overcome the security issues of the current schemes, a breakthrough in 2009 introduced by Gentry for his proposal of the FHE scheme [38,39]. FHE supports arbitrary computation over encrypted data and remains secure. Despite Gentry's achievement, his approach remains very expensive and impractical. Also, we discussed and classified several aspects of processing over encrypted data, such as functional encryption, searchable encryption, multiparty computation, and the recent industry offerings. Finally, we believe that an obvious shift in the field of processing over encrypted data is in the integration of trusted-hardware components with commodity servers. Interestingly, some researchers foresee the future of secure remote data management as infeasible without the usage of the trusted-hardware model.

8. REFERENCES

[1] M. Abadi, J. Feigenbaum, and J. Kilian. On hiding information from an oracle. In *ACM Symp. on Theory of Computing*, New York, USA, 1987.

[2] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Order-preserving encryption for numeric data. In *ACM SIGMOD Conference*, Paris, France, 2004.

[3] N. Ahituv, Y. Lapid, and S. Neumann. Processing encrypted data. *Communications of the ACM*, 30(9):777–780, 1987.

[4] J. Alwen, A. Shelat, and I. Visconti. Collusion-free protocols in the mediated model. In *CRYPTO*, pages 497–514, Santa Barbara, California, USA, 2008. Springer.

[5] A. Arasu, S. Blanas, K. Eguro, R. Kaushik, D. Kossmann, R. Ramamurthy, and R. Venkatesan. Orthogonal security with cipherbase. In *CIDR*, California, USA, 2013.

[6] S. Bajaj and R. Sion. Trusteddb: A trusted hardware-based database with privacy and data confidentiality. In *ACM SIGMOD Conference*, California, USA, 2011.

[7] G. R. Balkley and C. Meadows. A database encryption scheme which allows the computation of statistics using encrypted data. In *IEEE S&P*, Oakland, CA, USA, 1985.

[8] M. Bellare, A. Boldyreva, and A. O'Neill. Deterministic and efficiently searchable encryption. In *CRYPTO*, pages 535–552, California, USA, 2007.

[9] J. Benaloh. Dense probabilistic encryption. In *Selected Areas of Cryptography*, pages 120–128, Ontario, Canada, 1994.
[10] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *IEEE S&P*, pages 321–334. IEEE, 2007.
[11] A. Boldyreva, N. Chenette, Y. Lee, and A. O'Neill. Order-preserving symmetric encryption. In *EUROCRYPT*, pages 224–241, Cologne, Germany, 2009.
[12] A. Boldyreva, N. Chenette, Y. Lee, and A. O'Neill. Order-preserving encryption revisited: improved security analysis and alternative solutions. In *CRYPTO*, pages 578–595, California, USA, 2011.
[13] D. Boneh, C. Gentry, S. Gorbunov, S. Halevi, V. Nikolaenko, G. Segev, V. Vaikuntanathan, and D. Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit abe, and compact garbled circuits. In *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 533–556, 2014.
[14] D. Boneh, E.-J. Goh, and K. Nissim:. Evaluating 2-dnf formulas on ciphertexts. In *Theory of Cryptography*, volume 3378, pages 325–341, 2005.
[15] D. Boneh, A. Sahai, and B. Waters. Functional encryption: Definitions and challenges. In *Theory of Cryptography*, volume 6597, pages 253–273. Springer, 2011.
[16] D. Boneh and B. Waters. Conjunctive, subset, and range queries on encrypted data. In *Theory of Cryptography*, volume 4392, pages 535–554. Springer, 2007.
[17] C. Bösch, P. Hartel, W. Jonker, and A. Peter. A survey of provably secure searchable encryption. *ACM Computing Surveys (CSUR)*, 47(2):18, 2015.
[18] X. Boyen and L. Martin. Identity-based cryptography standard (ibcs) #1: Supersingular curve implementations of the bf and bb1 cryptosystems. RFC5091, December 2007.
[19] E. Boyle, S. Goldwasser, and I. Ivan. Functional signatures and pseudorandom functions. In *PKC 2014*, volume 8383 of *LNCS*, pages 501–519. Springer, 2014.
[20] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *Innovations in (Theoretical) CS*, Cambridge, MA, USA, 2012.
[21] Z. Brakerski and G. Segev. Function-private functional encryption in the private-key setting. Technical Report Report 2014/550, Cryptology ePrint Archive, 2014.
[22] Z. Brakerski and V. Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. In *CRYPTO*, pages 505–524, California, USA, 2011.
[23] M. Chase. Multi-authority attribute based encryption. In *Theory of Cryptography*, volume 4392 of *Lecture Notes in CS*, pages 515–534. Springer, 2007.
[24] N. Chenette, K. Lewi, S. A. Weis, and D. J. Wu. Practical order-revealing encryption with limited leakage, 2015.
[25] CipherCloud. Cloud data protection. [retrieved: Oct, 2014].
[26] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Y. Zhu. Tools for privacy preserving distributed data mining. *ACM SIGKDD*, 4(2):28–34, December 2002.
[27] R. Cramer, R. Gennaro, and B. Schoenmakers. A secure and optimally efficient multiauthority election scheme. In *EUROCRYPT*, pages 103–118, NY, USA, 1997.
[28] M. H. Diallo, B. Hore, E. C. Chang, S. Mehrotra, and N. Venkatasubramanian. Cloudprotect: Managing data privacy in cloud applications. In *IEEE Cloud*, Hawaii, USA, 2012.
[29] W. Diffie. The first ten years of public-key cryptography. *Proceedings of the IEEE*, 76(5):560 – 577, May 1988.
[30] K. Eguro and R. Venkatesan. Fpgas for trusted cloud computing. In *Field-Programmable Logic and Applications*, Oslo, Norway, 2012.
[31] J. Feigenbaum. Encrypting problem instances, or, ..., can you take advantage of someone without having to trust him? In *CRYPTO*. Springer-Verlag, 1986.
[32] C. Fontaine and F. Galand. A survey of homomorphic encryption for nonspecialists. *EURASIP Journal on Information Security*, pages 1–15, 2007.
[33] Forbes. Salesforce.com brings navajo into camp to boost cloud security. http://www.forbes.com/sites/greatspeculations/2011/08/30/salesforce-com-brings-navajo-into-camp-to-boost-cloud-security, 2011.
[34] T. E. Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *CRYPTO*, pages 10–18, Santa

Barbara, California, USA, 1984.
[35] S. Garg, C. G. S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS '13*, pages 40–49. IEEE Computer Society, 2013.
[36] T. Ge and S. Zdonik. Answering aggregation queries in a secure system model. In *VLDB*, pages 519–530, 2007.
[37] C. Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford, 2009.
[38] C. Gentry. Fully homomorphic encryption using ideal lattices. In *ACM Symp. on the Theory of Computing*, pages 169–178, Maryland, USA, 2009.
[39] C. Gentry. Computing arbitrary functions of encrypted data. *Comm. of the ACM*, 53(3):97–105, 2010.
[40] C. Gentry, S. Halevi, and N. P. Smart. Better bootstrapping in fully homomorphic encryption. In *Public Key Cryptography*, Darmstadt, Germany, 2012.
[41] C. Gentry, S. Halevi, and N. P. Smart. Homomorphic evaluation of the aes circuit. In *CRYPTO*, pages 850–867, California, USA, 2012.
[42] C. Gentry, A. Sahai, and B. Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO*, pages 75–92, California, USA, 2013.
[43] S. Goldwasser, S. D. Gordon, V. Goyal, A. Jain, J. Katz, F.-H. Liu, A. Sahai, E. Shi, and H.-S. Zhou. Multi-input functional encryption. In *EUROCRYPT*, volume 8441 of *LNCS*, pages 578–602. Springer, 2014.
[44] S. Goldwasser and S. Micali. Probabilistic encryption & how to play mental poker keeping secret all partial information. In *ACM Symp. on Theory of Computing*, pages 365–377, California, USA, 1982.
[45] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.
[46] S. Gorbunov. *Cryptographic Tools for the Cloud*. PhD thesis, MIT, 2015.
[47] S. Gorbunov, V. Vaikuntanathan, and H. Wee. Attribute-based encryption for circuits. In *STOC '13*, pages 545–554. ACM, 2013.
[48] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *13th ACM Conference on CCS*, pages 89–98. ACM, 2006.
[49] H. Hacigümüs, B. Hore, B. Iyer, and S. Mehrotra. *Search on Encrypted Data*, volume 33, chapter Secure Data Management in Decentralized Systems, pages 383–425. Springer, 2007.
[50] H. Hacigümüs, B. Lyer, , and S. Mehrotra. Query optimization in encrypted database systems. In *Database Systems for Advanced Applications*, Beijing, China, 2005.
[51] H. Hacigümüs, B. Lyer, C. Li, , and S. Mehrotra. Executing sql over encrypted data in the database-service-provider model. In *ACM SIGMOD Conference*, Wisconsin, USA, 2002.
[52] H. Hacigümüs, B. Lyer, and S. Mehrotra. Efficient execution of aggregation queries over encrypted relational database. In *Database Systems for Advanced Applications*, Jeju Island, Korea, 2004.
[53] S. Halevi. Helib: an implementation of homomorphic encryption. https://github.com/shaih/HElib. [retrieved: Oct, 2014].
[54] S. Halevi and V. Shoup. Algorithms in helib. In *CRYPTO*, California, USA, 2014.
[55] B. Hore, S. Mehrotra, , and G. Tsudik. A privacy-preserving index for range queries. In *VLDB*, pages 720–731, Toronto, Canada, 2004.
[56] J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *EUROCRYPT*, volume 4965, pages 146–162. International Association for Cryptologic Research, 2008.
[57] F. Kerschbaum and A. Schroepfer. Optimal average-complexity ideal-security order-preserving encryption. In *ACM Conference on CCS*, Arizona, USA, 2014.
[58] M. Lepinksi, S. Micali, and A. Shelat. Collusion-free protocols. In *ACM Symp. on Theory of Computing*, 2005.
[59] Microsoft. Transparent data encryption. http://msdn.microsoft.com/en-us/library/bb934049.aspx. [retrieved: Oct, 2014].
[60] Microsoft. Always encrypted (database engine), February 3 2016.
[61] I. Miers, C. Garman, M. Green, and A. D. Rubin. Zerocoin: Anonymous distributed e-cash from bitcoin. In *IEEE S&P*, pages 397–411, San Francisco, California, USA,

2013.
[62] D. K. Mishra and M. Chandwani. Extended protocol for secure multiparty computation using ambiguous identity. *WSEAS Transaction on Computer Research*, 2(2):227–233, February 2007.
[63] R. Müller, J. Teubner, and G. Alonso. Data processing on fpgas. *PVLDB*, 2(1):910–921, 2009.
[64] V. Oleshchuk and V. Zadorozhny. Secure multi-party computations and privacy preservation: Results and open problems. *Telektronikk*, 103(2):20–26, 2007.
[65] A. O'Neill. Definitional issues in functional encryption. *IACR Cryptology ePrint Archive*, 556, 2010.
[66] Oracle. Transparent data encryption. http://www.oracle.com/technetwork/database/options/advanced-security/index-099011.html.
[67] R. Ostrovsky, A. Sahai, and B. Waters. Attribute-based encryption with non-monotonic access structures. In *14th ACM Conference on CCS*, pages 195–203. ACM, 2007.
[68] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*, pages 223–238, Prague, Czech Republic, 1999.
[69] V. Pappas, F. Krell, B. Vo, V. Kolesnikov, T. Malkin, S. G. Choi, W. George, A. Keromytis, and S. Bellovin. Blind seer: A scalable private dbms. In *IEEE S&P*, Oakland, CA, USA, 2014.
[70] R. A. Popa, F. H. Li, and N. Zeldovich. An ideal-security protocol for order-preserving encoding. In *IEEE S&P*, Berkeley, California, USA, 2013.
[71] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan. Cryptdb: Protecting confidentiality with encrypted query processing. In *ACM Symp. on OSP*, Cascais, Portugal, 2011.
[72] K. P. N. Puttaswamy, C. Kruegel, , and B. Y. Zhao. Silverline: toward data confidentiality in storage-intensive cloud applications. In *ACM SOCC*, Cascais, Portugal, 2011.
[73] R. R. Ravan, N. B. Idris, and Z. Mehrabani. A survey on querying encrypted data for database as a service. In *CyberC*, pages 14–18, Beijing, Oct. 2013. IEEE Computer Society.
[74] European Union Agency for Network and I. Security. Survey: An sme perspective on cloud computing. http://www.enisa.europa.eu/activities/risk-management/files/deliverables/cloud-computing-sme-survey/at_download/fullReport, 2009. [retrieved: Oct, 2014].
[75] relax Google BigQuery. Encrypted bigquery client. https://code.google.com/p/encrypted-bigquery-client. [retrieved: Sep, 2014].
[76] relax North Bridge. Cloud adoption survey. http://www.northbridge.com/2013-future-cloud-computing-survey-reveals-business-driving-cloud-adoption-everything-service-era-it. [retrieved: Dec, 2014].
[77] R. L. Rivest, L. Adleman, and M. L. Dertouzos. *On Data Banks and Privacy Homomorphisms*, pages 169–179. Academic Press, New York, 1982.
[78] A. Sahai and B. Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, volume 3494, pages 457–473. Springer, 2005.
[79] R. Sheikh, D. K. Mishra, and B. Kumar. Secure multiparty computation: From millionaires problem to anonymizer. *Information Security Journal: A Global Perspective*, 20(1):25–33, January 2011.
[80] Skyhigh. Cloud security and enablement. https://www.skyhighnetworks.com/. [retrieved: Feb, 2016].
[81] D. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In *IEEE S&P*, Berkeley, USA, 2000.
[82] SqlCipher. Database encryption. https://www.zetetic.net/sqlcipher/. [retrieved: Oct, 2014].
[83] S. Tu, M. F. Kaashoek, S. Madden, and N. Zeldovich. Processing analytical queries over encrypted data. In *VLDB*, volume 6 of *5*, pages 289–300, Trento, Italy, 2013.
[84] M. v. Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. Fully homomorphic encryption over the integers. In *EUROCRYPT*, Nice, France, 2010.
[85] B. Waters. A punctured programming approach to adaptively secure functional encryption. Technical report, University of Texas at Austin, 2014.
[86] A. C. Yao. Protocols for secure computations. In *23rd Symp. on Foundations of CS*, pages 160–164, Indore, India, 1982.