Cryptography for Big Data Security

Book Chapter for Big Data: Storage, Sharing, and Security (3S)*

Ariel Hamlin† Nabil Schear† Emily Shen†
Mayank Varia‡ Sophia Yakoubov†
Arkady Yerukhimovich†,§

December 17, 2015

Big Data: Storage, Sharing, and Security. Edited by Fei Hu. Taylor & Francis CRC Press, 2016. https://www.crcpress.com/9781498734868

†MIT Lincoln Laboratory. This work is sponsored by the Assistant Secretary of Defense for Research and Engineering under Air Force Contract #FA8721-05-C-0002. Opinions, interpretations, recommendations and conclusions are those of the authors and are not necessarily endorsed by the United States Government. Email: {ariel.hamlin, nabil, emily.shen, sophia.yakoubov, arkady}@ll.mit.edu.

‡Boston University. Work supported by the MACS project under NSF Frontier grant CNS-1414119. Email: varia@bu.edu

§Corresponding author

Contents

1 Introduction **1**
- 1.1 The Data Lifecycle . . . 2
- 1.2 Chapter Outline . . . 3

2 The Basics of Data Security **3**
- 2.1 Modeling the Adversary . . . 3
- 2.2 Security Goals . . . 4
- 2.3 Basic Cryptographic Tools . . . 5

3 Secure Block Storage and Access Control **6**
- 3.1 Key Management for Access Control . . . 7
  - 3.1.1 Introduction . . . 7
  - 3.1.2 Definition . . . 7
  - 3.1.3 Survey . . . 8
  - 3.1.4 State of the art . . . 9
- 3.2 Attribute-Based Access Control . . . 9
  - 3.2.1 Introduction . . . 9
  - 3.2.2 Definition . . . 10
  - 3.2.3 Survey . . . 10
  - 3.2.4 State of The Art . . . 11

4 Secure Search **11**
- 4.1 Introduction . . . 12
- 4.2 Definition . . . 12
  - 4.2.1 Parties Involved . . . 13
  - 4.2.2 Security Guarantees and Limitations . . . 13
  - 4.2.3 Trust Requirements . . . 14
- 4.3 Survey . . . 14
  - 4.3.1 History: Precursors to Searchable Encryption . . . 14
  - 4.3.2 Searchable Encryption . . . 15
- 4.4 State of the Art . . . 19
  - 4.4.1 Security-Focused Schemes . . . 19
  - 4.4.2 Performance-Focused Schemes . . . 22

5 Secure Data Processing **24**
- 5.1 Homomorphic Encryption . . . 25
  - 5.1.1 Introduction . . . 25
  - 5.1.2 Definition . . . 25
  - 5.1.3 Survey . . . 25
  - 5.1.4 State of the Art . . . 27
- 5.2 Verifiable Computation . . . 27

5.2.1 Introduction 27
5.2.2 Definition 28
5.2.3 Survey 28
5.2.4 State of the Art 29
5.3 Multi-Party Computation 30
5.3.1 Introduction 30
5.3.2 Definition 30
5.3.3 Survey 31
5.3.4 State of the Art 34
5.4 Functional Encryption 34
5.4.1 Introduction 34
5.4.2 Definition 35
5.4.3 Survey 35
5.4.4 State of the Art 36

6 Conclusion **36**

1 Introduction

With the amount of data generated, collected, and analyzed by computing systems growing at an amazing rate, big data processing has become crucial to most enterprise and government applications. For example, the rise of Internet of things (e.g., a connected refrigerator), smart phone location tracking to target advertising, and the growing adoption of health and wellness devices that collect personal statistics all represent new opportunities for novel analytics and services.

However, this proliferation of big data is not without its own dangers. The collected data often contains private information about individuals or corporate secrets that would cause great harm if they fell into the wrong hands. Criminal groups are creating underground markets where one can buy and sell stolen personal information [124]. Government intelligence services are targeting personal, corporate, and adversary government systems for espionage and competitive advantage [119]. This potential for harm is clearly demonstrated by many recent, highly publicized cyber attacks against commercial [164] and government targets [165], costing these organizations millions of dollars and causing serious damage to the individuals and institutions affected.

As these examples make clear, new and improved security tools are needed to protect systems collecting and handling big data to allow applications to reap the benefits of big data analysis without the risk of such catastrophic attacks. Fortunately, modern cryptography offers many powerful technologies that can help protect big data applications throughout the data lifecycle, as it is being collected, stored in repositories, and processed by analysts. In this chapter, we give a brief survey of several of these technologies and explain how they can help big data security. We hope that this survey can provide big data practitioners with a better understanding of what protections are available to them as they design their applications and will also encourage the necessary discussion and collaboration to mature these tools for real-world consumption.

This chapter focuses on state-of-the-art provably secure cryptographic techniques for protecting big data applications. We do not focus on more established, and commonly available cryptographic solutions. The goal is to inform practitioners of new techniques to consider as they develop new big data solutions rather than to summarize the current best practice for securing data.

We note that, in addition to the cryptographic techniques, there are also many non-cryptographic techniques such as firewalls, data guards, mandatory access control, data provenance, and many more that are likely to be a part of any complete solution to secure big data applications. However, such security mechanisms are beyond the scope of this chapter and we refer readers to the excellent book on the topic by Anderson [6] for a good summary. Additionally, we do not cover the security of operation, monitoring, and maintenance of big data systems and refer to Limoncelli et al., for more details [112]. Finally, we focus on software-based solutions and do not discuss the work relying on trusted hardware components to achieve data security [10, 153, 9]

We also do not address the privacy implications of big data collection and processing.

That is, our focus is on keeping the data and computations out of the hands of unintended parties and not on the questions of whether the data should be collected or the analyses performed in the first place or what the implications of such analyses are for the individuals and corporations supplying the data. These are very important considerations in any big data system, but they fall outside the scope of our discussion here.

1.1 The Data Lifecycle

We begin this chapter by presenting the cryptographer's view of the data lifecycle and the stages that big data goes through. To secure big data, it is necessary to understand the threats and protections available at each stage. For this reason, the cryptographic techniques presented in this chapter are organized according to the three stages of the data lifecycle described below.

Data in Transit: The first step in most big data processing architectures is to transmit the data from a user, sensor, or other collection source to a centralized repository where it can be stored and analyzed. To ensure that the data arrives at its destination unmodified and unstolen, it is necessary to ensure that all data is transmitted in a protected form. This is the stage of the lifecycle best served by existing tools with existing technology such as the Internet Protocol Security (IPSec) [61] and the Transport Layer Security (TLS) [62] protocol suites already standardized and widely deployed in many of the big data tools of interest to the reader. Since the focus of this chapter is on newer technology, we do not discuss data in transit security further.

Data in Storage: The next step in the big data lifecycle is to store the data in a repository where it will be stored until it is needed. It is critical that unauthorized parties not be able to read or modify this data in storage. However, authorized parties should be able to efficiently access all or parts of this data as necessary. Thus, more advanced techniques are needed to enforce access control to stored data while allowing for efficient data retrieval.

Data in Use: Finally, once the data has been collected and stored, it is necessary to run analytics over the data to derive value from the collected information. It is necessary to guarantee that only authorized analytics are run on the data by authorized parties and that the computation is performed correctly. This is one area where current best-practices are especially lacking, with all processing done on unprotected data. Modern cryptography offers several techniques to change this status quo and allow data to remain protected even while it is in use.

1.2 Chapter Outline

The remainder of this chapter is organized by the stages of the data lifecycle. We begin in Section 2 with a description of the basic concepts of data security and an overview of the traditional cryptographic tools such as encryption and signatures. In Sections 3 and 4 we describe techniques for protecting and searching stored data. In Section 3 we present techniques such as *broadcast encryption* and *attribute-based encryption* for controlling access to data stored and retrieved via unique identifiers, such as in a file system. In Section 4 we describe *searchable encryption*, which allows complex queries (e.g., SQL queries) to be performed to retrieve allowed subsets of the protected data, such as in a database. Finally, in Section 5, we describe tools such as *homomorphic encryption*, *verifiable computation*, *secure multi-party computation*, and *functional encryption* for protecting computation over sensitive data.

For each of the presented technologies, we use the following outline. We begin with a high-level overview of what protections the cryptographic technology provides and how it can be used. Next, we give a more detailed definition of the security achieved and the critical properties of the technology. Finally, we give an in-depth summary of the history and state of the art developments for the technology to illustrate the differences between individual schemes and their potential uses-cases.

2 The Basics of Data Security

We begin our chapter with an overview of some of the basic goals and tools of data security. First, we discuss the typical adversary models used to capture the threats to data that need to be addressed. Then, we briefly review common security goals and the tools used to achieve them.

2.1 Modeling the Adversary

In this chapter, we restrict our attention to technologies with *provable* statements of security. These provable statements require that we formally model the threats addressed in the form of an adversary. This adversary formally captures the types of attacks as well as the limits on the attacks that we want the technology to withstand. Some properties typically addressed by an adversary model include:

- **Types of Adversarial Behavior:** Cryptographic protocols typically aim to provide security against one of two adversarial behaviors. An *honest-but-curious* or *semi-honest* adversary follows the protocol description but tries to learn unauthorized information from the messages he receives. A *malicious* adversary, on the other hand, may deviate arbitrarily from the protocol in order to learn private information or disrupt the protocol. These, roughly, model an eavesdropper who observes part of the protocol

execution in an effort to learn private information and a malicious insider or hacker trying to actively interfere with the computation.

- **Amount of Collusion:** In cryptographic protocols involving multiple parties, it is necessary to consider the number of parties that may behave adversarially and whether those parties will collude with each other during their attacks. Typically, cryptographers limit the total number of adversarial parties but assume that all such parties will collude. In some settings, such as searchable encryption, weaker models of collusion restrict parties playing different roles in the protocol from colluding with each other. Such restrictions are often justified by the organizational structure of the participating parties and possible adversaries.
- **Computational Limitations:** Most provably secure cryptographic protocols rely on the mathematical difficulty of certain problems like factoring integers or computing discrete logarithms in finite groups. An important consideration for any cryptographic primitive is how strong an assumption is necessary to prove its security. Typically, schemes secure that are based on weaker and more standard mathematical assumptions are preferable, and much effort in cryptography is devoted to reducing the assumptions needed for desired cryptographic tasks.

2.2 Security Goals

Having defined the adversary we want to protect against, we need to describe the security goals. The three most fundamental security goals are *confidentiality*, *integrity*, and *availability*, collectively known as the CIA triad.

Confidentiality: Confidentiality is the goal of keeping all sensitive data secret from an adversary. More formally, traditional definitions of confidentiality guarantee that an adversary should learn no information about the sensitive data, other than its length. Confidentiality is critical in big data applications to guarantee that sensitive data is not revealed to the wrong parties.

Integrity: Integrity is the goal that any unauthorized modification of data should be detectable. That is, a malicious adversary should not be able to modify such data without leaving a trace. This is very important to help guarantee the veracity of data collected in big data applications.

Availability: Availability is the goal of always being able to access one's data and computing resources. In particular, an adversary should not be able to disable access to critical data or resources. This is a very important security goal in big data processing, as the sheer volume and velocity of the data make guaranteeing constant access a difficult task. However, in today's big data systems, availability is typically guaranteed via non-cryptographic means such as replication, and we will not discuss it further in this chapter.

2.3 Basic Cryptographic Tools

We now give a very brief overview of the basic cryptographic tools used to ensure the aforementioned security goals in simple applications such as data in transit. Then, in the remainder of this chapter we present techniques that build on these to guarantee CIA of data while enabling richer uses of the data. For a much more complete presentation of these basic tools, their formal definitions, and detailed constructions we refer the interested reader to the excellent book by Katz and Lindell [99].

Encryption: The main tool for guaranteeing confidentiality of data is data encryption. Encryption takes a piece of data, commonly called the plaintext, together with a cryptographic key and produces a scrambled version of the data called the ciphertext. Using the key it is possible to decrypt the data to recover the plaintext, but without the key the ciphertext hides all information about the original data, other than its length. This security property, commonly known as *semantic security* [82] guarantees that, without the key, an adversary cannot learn any (potentially sensitive) property of the underlying data even if he has a lot of insight as to what the data may be. This is critical in applications where data may have some predefined structure, such as in financial transactions or if partial information about the underlying distribution of data is known, such as when the data is measuring some real-world phenomenon.

A little more formally, encryption consists of the following three protocols:

- KeyGen - a key generation algorithm that generates the necessary cryptographic keys,
- $\mathsf{Enc}(k, p) = c$ - an encryption algorithm that uses a key k to scramble the plaintext p into cipher text c,
- $\mathsf{Dec}(k, c) = p$ - a decryption algorithm that uses the key k to recover the plaintext p from the ciphertext c.

Encryption schemes come in two flavors: secret-key encryption, described above, and public-key encryption. In secret-key encryption the same key is used for encrypting and decrypting data. In public-key encryption, KeyGen produces two keys: a public key and a secret key. The public key is used to encrypt the data, but cannot be used to decrypt the data and thus can be made public. The secret key, which must be kept private, is used to decrypt the data.

Secret-key encryption has been around for thousands of years with many protocols developed over the years [96]. Today there are well established standards for secret-key encryption such as schemes based on the Advanced Encryption Standard (AES) block cipher [130]. In the 1970s, public-key encryption was introduced by Diffie and Hellman [56] and Rivest, Shamir, and Adelman [149] and was formalized by Goldwasser and Micali [82].

Message Authentication Codes and Digital Signatures: The primary tools for guaranteeing integrity of data are message authentication codes (MACs) and digital signatures.

Roughly, they both take a message and a key and generate an authentication value that can be used to verify the integrity of the data. The standard security property, *existential unforgeability*, of these primitives states that without the key it is impossible to forge a valid MAC or signature on any piece of data even after having seen MACs or signatures on other data items. Thus, an adversary cannot create or modify data without being detected. A little more formally, MACs and signatures consist of the following three protocols:

- KeyGen - a key generation algorithm,
- $\mathsf{Sign}(k, m) = \sigma$ - a signing algorithm that uses the key k and the message m to generate an authentication object σ,
- $\mathsf{Verify}(k, m, \sigma) = b$ - a verification algorithm that uses the key k, message m, and the authentication object σ and returns a boolean value b depending on whether or not the message was signed using the corresponding key.

As with encryption, there are two flavors of these schemes: secret-key and public-key. MACs are the secret-key variant, where the same key is used both to sign the data and to verify that the signature is valid. Digital signatures are the public-key variant, where a secret key is needed to sign the data, but a public verification key allows anyone to check the validity of the signature.

Currently used message authentication codes known as HMAC and NMAC were originally proposed by Bellare, Canetti, and Krawczyk [13]. Public-key-based digital signatures were originally proposed by Diffie and Hellman [56] and Rivest, Shamir, and Adelman [149] and formalized by Goldwasser, Micali, and Rivest [83].

Implementations and Standards: All of the cryptographic primitives described above are already widely in use and there are multiple available implementations and standards for how they should be used. The National Institute for Standards and Technology (NIST) maintains a list of standards for all of the above primitives as part of their cryptographic toolkit [129]. For US government applications, the National Security Agency maintains what is known as the Suite B standard [3] for which primitives must be used to protect sensitive US government data. Finally, the use of public-key primitives is standardized by the RSA Laboratories' PKCS #1 standards [110]. We strongly urge readers to follow recommendations and implementations described in these standards and to avoid implementing their own solutions, as it is very difficult to implement bug-free cryptographic protocols and these standards are the results of decades of improvements and bug fixes.

3 Secure Block Storage and Access Control

We now turn to cryptographic techniques to secure data in storage. The primary goal of these techniques is to enforce access control to data stored in potentially untrusted repositories. That is, we want to give authorized parties access to the data they need while ensuring

that unauthorized parties, either outsiders trying to gain access or malicious insiders in the organization managing the repository, cannot access sensitive data. In this section we focus on systems where data is stored in blocks that are stored and retrieved by a unique identifier, such as in a file system. In such systems, we want authorized parties to be able to retrieve data by its identifier, but do not need to enable complex search queries to retrieve subsets of the data.

As may be expected, the first step in any such solution is to encrypt the data to hide it from unauthorized parties. However, the problem now becomes how to give authorized parties access to the data. We discuss two techniques for enabling such access control. The first of these, commonly called *group keying* or *broadcast encryption*, achieves this through careful key management to ensure authorized parties have the necessary keys to decrypt the data. The second of these, a technique known as *attribute-based encryption (ABE)*, instead relies on more powerful cryptographic techniques to automatically enforce the access permissions.

3.1 Key Management for Access Control

3.1.1 Introduction

Key management includes generating and distributing cryptographic keys to system users in such a way that only authorized parties have the necessary keys to decrypt sensitive data. Most modern systems include some form of key management for controlling access to data in this way and there are many commercially available, standardized solutions for generating and managing keys.[1] These typically use a *trusted* key management server to manage all keys in the system and to distribute the necessary keys to authorized parties. Here, we instead focus on a cryptographic technique called *broadcast encryption* or *group keying* which allows a data owner to encrypt data to a designated set of recipients without having to rely on a trusted key manager. This is particularly important in big data applications where the storage may be handled by an untrusted repository on which a trusted key manager may not be available.

3.1.2 Definition

Broadcast encryption or group keying is a cryptographic technique for establishing cryptographic keys shared among a designated set of parties, thus giving these authorized parties access to encrypted data. Specifically, broadcast encryption gives a way to establish a cryptographic key such that all authorized parties receive the key and all unauthorized parties have no information about the key. In particular, even if some number of unauthorized parties collude, they should not be able to learn data that none of them is individually authorized to learn. An important goal of this primitive is to minimize the total size of encrypted data

[1] See, for example, the various vendor implementations of the OASIS Key Management Interoperability Protocol (KMIP) standard for key management systems [128].

that must be generated and the amount of key material that must be stored by each of the participating parties.

3.1.3 Survey

Imagine that a data owner wishes to share data with some subset of the users of a system. One trivial solution is to have him share a different cryptographic key with each of the recipients and separately encrypt the data to each of them. However, this requires the data owner to store a large number of keys, and also the size of the encrypted data grows linearly in the number of recipients. Broadcast encryption gives techniques to achieve this functionality without incurring these costs in key and data storage.

Single Sender Broadcast Encryption: Broadcast encryption was first considered by Fiat and Naor [59] in a setting where there is one data owner who wants to share data with a set of authorized recipients. This construction was able to achieve much shorter keys and ciphertexts when the number of unauthorized users is small. This was further improved by Naor, Naor, and Lotspiech [125], who showed a protocol able to handle an arbitrary number of unauthorized users while only incurring a logarithmic (in the number of parties) overhead both in key and ciphertext size when the number of adversaries isn't too large. Roughly, both of these schemes work as follows. First, they generate cryptographic keys and then distribute these keys among all the possible users. Then to encrypt data to an authorized set of users, they encrypt the data under an appropriately chosen subset of the keys. The partitions of the keys and the subset used to encrypt are chosen to guarantee that all authorized user will know at least one key enabling it to decrypt the data, while all unauthorized users will not know any of these keys. See Figure 1 for an example of the mechanism given by Naor et al. One critical limitation of both of these schemes is that they only allow for one data owner who can share data, but in most big data scenarios there are multiple data providers.

Public-Key Broadcast Encryption: In order to overcome this single sender limitation, the literature turned to public-key broadcast encryption. Such schemes allow anybody to share data, but rely on much stronger computational assumptions. The first scheme to do this was a scheme by Dodis and Fazio [57], who showed how to achieve parameters similar to the Naor et al. scheme in the public-key setting. An alternative construction by Boneh, Gentry, and Waters [27] reduces the size of the secret keys needed by the data recipients down to a constant independent of the number of users, at the cost of (somewhat) increasing the size of the public key and encrypted data. One major drawback of these schemes is that they rely on a relatively novel, powerful, and non-standardized cryptographic building block called bilinear groups. Further work [31, 32] was able to significantly improve on the parameters of these schemes by relying on even stronger computational assumptions.

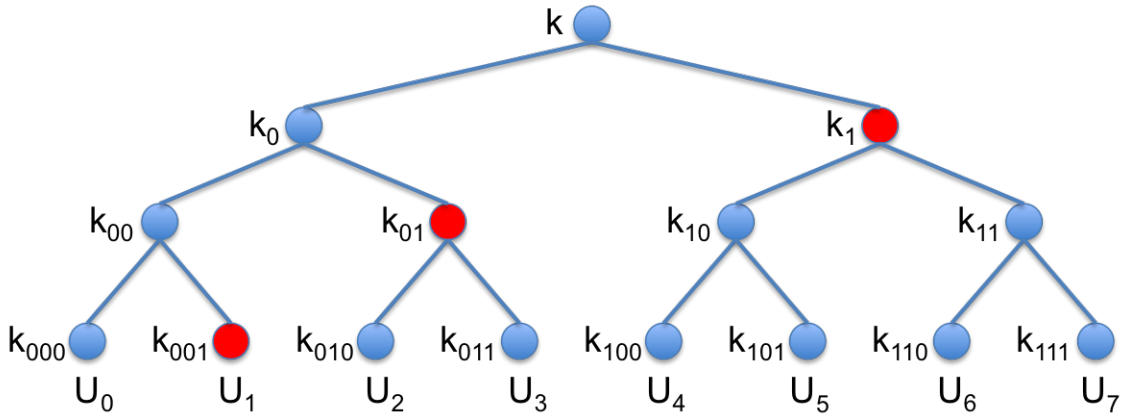

Figure 1: The NNL tree for broadcast encryption for 8 users. Each k_i represents an independent cryptographic key. Each user U_j receives all the keys on the path from the root to their leaf node. To encrypt the data, encrypt a copy of the data under the keys at the roots of subtrees containing only authorized users. For example, to encrypt to the set of all users except U_0, one would use the keys corresponding to the nodes colored in red.

3.1.4 State of the art

There are several available implementations of these broadcast encryption protocols. First, the Lincoln Open Cryptographic Key Management Architecture (LOCKMA) [101, 49] leverages the Naor et al. protocol to establish group keys for efficient communication over tactical networks. Also, an implementation of the public-key scheme by Boneh et al. has been demonstrated on top of the Pairing-Based Cryptography library [117].

3.2 Attribute-Based Access Control

3.2.1 Introduction

Key management based solutions such as above have an inherent limitation. In order to share data with a set of users, it is necessary to know the identities (and keys) of all the authorized users. This is problematic in large systems or in systems with several organizational structures (as is very common in big data architectures where the data is collected, stored, and used in different environments) as the data owner is unlikely to know the identities of all the authorized users. An alternative approach to access control in such settings is a technique called *attribute-based access control (ABAC)*. In ABAC, data is encrypted together with a policy describing the attributes of users authorized to access the data. The users receive keys for the attributes they possess and are able to access the data if and only if those attributes are authorized. This allows for enforcing access to data without knowing the full set of users with the authorized attributes. For example, to encrypt data for analysis by NIH scientists, a data provider could encrypt the data with the policy ("NIH" and "scientist") and only someone possessing both these attributes would be able to decrypt the data.

One approach to ABAC that has gained traction in government and commercial uses

is to have a trusted server evaluate the access policy over a user's attributes and grant or restrict access to data accordingly [132, 63]. However, this requires trusting the server to correctly administer these permissions and is problematic in scenarios where there is no such trusted entity, such as in outsourced storage. We instead focus on solutions for ABAC that do not require a trusted server to evaluate the access policies. Specifically, we present a powerful cryptographic technique known as *attribute-based encryption (ABE)* that can be used to solve this problem cryptographically.

3.2.2 Definition

(Ciphertext-policy) attribute-based encryption, introduced by Sahai and Waters [152], is a form of encryption in which keys are associated with attributes and data is encrypted with a policy specifying which attributes are needed to decrypt the ciphertext. The security of this primitive guarantees that a key will succeed in decrypting a ciphertext if and only if the attributes in the decryption key satisfy the access policy specified in the ciphertext. More formally, the ciphertext contains a boolean formula f over attributes and a key will successfully decrypt the ciphertext if f evaluates to 1 on the attributes contained in the key.

There are two properties that must be addressed in any ABE scheme. The first is the expressivity of the supported policy formulas. Schemes achieving more expressive policies can be used to enforce access control in more diverse settings. The second property is *collusion resistance*. That is, unauthorized users should not be able to combine their keys in order to decrypt data for which neither of their keys individually satisfies the access formula. This is typically the hardest feature of ABE to achieve and requires stronger cryptographic assumptions.

3.2.3 Survey

A First Attempt: As a first attempt to constructing ABE, we can do the following. Let each cryptographic key correspond to an attribute, so k_A is given to all users with attribute A. Now, to encrypt a piece of data so that only those possessing attributes A and B can decrypt, we can encrypt the data first under k_A and then under k_B, resulting in $c = \mathsf{Enc}_{k_B}(\mathsf{Enc}_{k_A}(x))$. Clearly, both keys are needed to decrypt the data. To encrypt to users with either attribute A or B, encrypt the data separately under each key, resulting in $c = (\mathsf{Enc}_{K_A}(x), \mathsf{Enc}_{K_B}(x))$. Now either key will be able to decrypt. It is possible to repeat this procedure to enforce access policies represented by arbitrary boolean formulas over the attributes.

However, this simple solution is not collusion resistant. If one user with attribute A (but not B) colludes with a user with attribute B (but not A) they can decrypt data encrypted to policy (A AND B) even though neither of them satisfies this policy. More complicated cryptography is needed to achieve the necessary collusion resistance.

Attribute-Based Encryption: The first ABE scheme to satisfy full collusion-resistance was given in 2005 by Sahai and Waters [152] supporting a limited set of policies known as

threshold policies, where an authorized user's key has to have large overlap with the set of keys specified in the policy. An implementation of this scheme describing how this can be used for access control was given by Pirretti et al. [138]. The class of supported policies was extended to arbitrary boolean formulas by Goyal et al. [86] and Bethencourt, Sahai, and Waters [19].

All of these schemes take the following approach to prevent collusion attacks. As in the no-collusion scheme above, they generate cryptographic keys corresponding to the possible attributes. However, each generated key is also personalized to the specific user. So, the key for attribute A given to user 1 would be different from the key for attribute A given to user 2. This prevents keys from different users from decrypting the data since the personalized components will not match. However, this personalization approach requires the use of a non-standardized cryptographic building block called bilinear groups and it is unlikely that (collusion resistant) ABE can be based on standard encryption [100].

Since these works there have been a number of improvements optimizing performance, e.g., [102], achieving richer classes of access policies [85], allowing for delegation of access permissions [170], and giving constructions relying on alternative security assumptions, e.g., [68].

3.2.4 State of The Art

There are now a number of available implementations of ABE. The first such implementation was given by Bethencourt et al. [19] on top of the Pairing-Based Crypto library [117]. The latest implementation of ABE, due to Khoury et al. [102], achieves 3ms for ABE encryption and 6ms for decryption, making this primitive fairly practical for securing data storage. On the functionality side, we now have ABE schemes capable of enforcing very rich classes of policies such as policies specified by arbitrary (polynomial-size) boolean circuits over the attributes [85]. We note that all of these schemes still rely on non-standard security assumptions.

4 Secure Search

The previous section dealt with security and access control for block storage where data can only be retrieved via its unique identifier. However, in any big data system, users rarely want to retrieve all available data and instead usually only fetch a subset of the available data based on some specified search criteria. We now switch topics to discuss cryptographic techniques to enable *secure search* allowing for complex, database-style queries to be performed on stored data.

4.1 Introduction

Searchable encryption refers to a collection of cryptographic techniques that enable a server to perform searches directly over encrypted data. Most database technologies in use today require a server to be able to read all data contents in order to perform a fast search over it. By contrast, searchable encryption technology permits a server to perform basic database operations directly over encrypted data. Even though the server cannot read the data, it can still return encrypted versions of matching results back to a querier. In summary, searchable encryption technology provides stronger confidentiality and privacy guarantees than those in databases today by separating the roles of providing and accessing data.

There are a myriad of use-cases for this technology in today's interconnected world; we provide just a few examples here. First, in a secure cloud outsourcing scenario, a client can upload files to the cloud and then make searches on the data later, without giving the cloud provider access to read the data. Second, in a secure publish-subscribe system, a broker can properly route messages from many providers to interested subscribers without being able to read the data itself.[2] Third, in a secure email scenario, many senders can send encrypted mail to one receiver, and then the receiver can make searches to her email host to pull only the emails of interest to her without giving the host access to the contents of her emails. Fourth, in a secure database scenario, a data provider can allow multiple clients to search over her data without knowing the locations of data being requested; note here that preventing the data provider from learning the queriers' access patterns (e.g., frequency of requested records and overlaps between sets of records returned in response to different queries) is critical since the provider already knows the contents of her own data, so encryption alone does not suffice here.

Research into searchable encryption has been ongoing for the past 15 years. While there is still work to be done, the current state of this technology is quite usable. First, modern searchable encryption technologies are quite fast: their performance is only about 20% to 200% slower than an unprotected search over plaintext data. Second, the searchable encryption works described below cover a large subset of the types of searches typically performed in databases, such as those permitted by the `where` clause in the Structured Query Language (SQL). For applications where one is willing to sacrifice a bit in security in exchange for performance, there are alternative searchable encryption schemes that work on top of existing database back-ends (e.g., MySQL and Accumulo) achieving much better performance in exchange for revealing some additional information about the stored data [140, 65].

4.2 Definition

A formal definition of searchable encryption is both complex and ever-changing. In this chapter, we present an overview of the concepts behind searchable encryption here and then describe some of the complexities in the next section.

[2]Note that broadcast encryption, described in Section 3.1, achieves a similar goal without using a proxy, but instead requiring providers to know all interested subscribers.

4.2.1 Parties Involved

The standard model of searchable encryption considers three types of parties: an *owner* who initially possesses the data, a *querier* who wants to learn something about the data, and a *server* who handles the bulk of the storage and processing work. We note that in many scenarios, such as the publish-subscribe and email scenarios above, there may be multiple data owners or queriers.

We note that some scenarios envision only two parties, which we call the querier and server. For instance, the cloud outsourcing scenario above considers a client who acts as both the initial data owner and subsequent querier, and the secure database scenario considers a data owner who does her own server processing.[3]

We can fit these scenarios into our three-party framework by thinking of them as instances of *collusion*[4] between two of the three parties: in the cloud outsourcing scenario, the data owner colludes with the querier, and in the second scenario, the data owner colludes with the server.

4.2.2 Security Guarantees and Limitations

In this chapter, we restrict our attention to searchable encryption technologies with *provable* claims about their security. Here, we overview some of the provable security guarantees provided by searchable encryption technology. First, the querier only learns the contents of records that match her query; she learns nothing about the rest of the owner's dataset, except in some cases its size. Second, the data owner doesn't learn anything about queries being performed on her data. Third, the server performs searches in a relatively oblivious fashion, learning almost nothing about either the data or queries involved.

However, most searchable encryption schemes do reveal or *leak* some information about data and queries to the server. We describe two common leakages that most searchable encryption schemes reveal. First, *search pattern leakage* permits the server to determine if two different queries are identical, even if she cannot learn what the queries are. Second, *access pattern leakage* permits the server to determine which records in the dataset are returned in response to each query, even if she cannot read the records. As a consequence, the server can also determine which records are "popular" and being returned in response to many (potentially different) queries.

The precise cryptographic definitions that formalize these security notions have evolved over the years. We describe some of the nuances of these security definitions in Section 4.3 and refer interested readers to Bosch et al.'s survey [33] for the formal definitions of searchable encryption.

[3]The two-party cloud outsourcing scenario in which the owner and querier are the same entity is also used in the model of fully homomorphic encryption, described in detail in Section 5.1.1 below.

[4]Collusion is typically used in cryptography to model a collection of malicious users who are working together, as described in Section 2.1. But, the same concept applies just as well to the benign setting in which two entities are "working together" because they are really the same person.

4.2.3 Trust Requirements

In order to make provable statements of security like the ones above, searchable encryption schemes (and indeed, all of cryptography) require some pre-conditions to be met: namely, they require users to place their trust in certain aspects of the behavior of other technologies or parties involved in the cryptographic protocol. Specifically, the searchable encryption schemes commonly rely on assumptions about the allowed adversary behavior, limits on collusion between the involved parties, and computational assumptions for security (See Section 2.1 for a detailed discussion).

In addition to the above, there are several protocols for secure search that do not require trust in the veracity of computational assumptions but rather in the inability of an adversary to perturb a specialized piece of hardware that performs computations inside of a (hopefully) tamper-proof shell. Put bluntly, these schemes place their trust in an adversary's physical limitations rather than her mathematical ones. However, as this chapter is focused on software-based solutions, we will only briefly mention one example of such schemes and refer interested readers to the original papers [159, 20, 9].

4.3 Survey

In this section, we first provide a brief history of three important cryptographic prmitives that preceded searchable encryption. Next, we describe some of the initial works on searchable encryption. Finally, we detail several innovations that improve the security of searchable encryption or add new dimensions to the functionality, performance, or security of this technology.

We stress that this section only provides a brief overview of the major thematic advances in searchable encryption technology over the years. We purposely do not delve into all technical details and only present the first few papers that tangibly influenced each theme presented below. For a more detailed, technical review of the (more than 100) research works on searchable encryption, we refer readers to the comprehensive survey by Bösch *et al* [33].

4.3.1 History: Precursors to Searchable Encryption

We begin by describing three cryptographic technologies initially developed in the 1990s that solve related problems. All of these works utilize a two-party setup with a querier and a server. Some of them are better suited to the cloud outsourcing scenario (in which the querier provides the data) and some are better suited to the database setting (in which the server provides the data).

First, *private information retrieval*, or PIR, considers the cloud outsourcing problem in which a data owner outsources her data to an untrusted server and later wants to query for a single record. While the database contents themselves may be readable in the clear by the server, the querier wants to hide her interests from the server (i.e., which record is being returned in response to a query). An important goal in PIR works is to minimize the

amount of communication required between the server and querier. Chor et al. [48] constructed the first PIR scheme in an information-theoretic model with multiple non-colluding servers. Shortly thereafter, several works provided PIR under the assumption that certain cryptographic problems are hard [108, 38, 109]. We refer interested readers to Gasarch's excellent survey [70] for more information about PIR.

Second, Chor et al. [47] introduce *private information retrieval by keywords*. This work introduces the concept of searching (as we think of it today) to the cryptographic literature. Specifically, it uses multiple rounds of PIR to search by keyword rather than by the index (i.e., pointer or unique identifier) of the record that the client wishes to retrieve. Nevertheless, data are still stored unencrypted on the server.

Third, *symmetric private information retrieval*, or SPIR, extends PIR's security guarantees to protect the server as well. Specifically, whereas PIR permits the client to view the entire database, SPIR limits the number of records that a querier may retrieve in response to any query. Hence, SPIR is useful both in the cloud outsourcing scenario and the database scenario, and it can also support multiple queriers securely by using its access controls to restrict each querier to reading only her own data on the server. SPIR was introduced by Gertner et al. [76], though it has its roots in another cryptographic primitive (called oblivious transfer) developed by Rabin in the 1970s [143].

Finally, *oblivious random-access memory*, or ORAM, introduces the concept of a server performing keyword searches directly over encrypted data in an *oblivious* fashion; that is, without learning either the contents of any records stored at the server or the access patterns of data returned to the client (such as the simple question "is this identical to data that the server returned to the client in the past?"). Goldreich and Ostrovsky's seminal work [79] introduces multiple ORAM techniques, including an asymptotically-optimal logarithmic-time search and a square-root-time search that turns out to be faster in practice [93]. ORAM has been extensively studied by the cryptographic community since the 1990s, and recent works have substantially improved the server's computational burden [162, 163, 147, 54, 122, 55, 123].

One performance impediment of ORAM schemes is the large number of rounds of communication that they require between the querier and server. By contrast, most of the searchable encryption works discussed below only require one round of communication. In order to achieve this performance improvement, however, most searchable encryption works reveal some search pattern information to the server.

4.3.2 Searchable Encryption

In 2000, Song, Wagner, and Perrig published the seminal work on searchable encryption [160]. They presented a cryptographic protocol in a two-party setting (with a querier and server) to search over encrypted data that provided the following four properties:

1. Provable security for encryption: The untrusted server stores ciphertext data that has been encrypted using a semantically-secure encryption scheme (as described in Section 2.3), so the server cannot learn any information about the corresponding plaintexts.

2. Controlled searching: The untrusted server can only perform searches that have been authorized by the querier. The server cannot make searches on her own.

3. Hidden queries: Song et al. support keyword searches. During a query, the server does not learn the keyword. (This concept is later formalized by Goh [77] as "chosen keyword security" and defined in a similar style to semantic security.)

4. Query isolation: While the server does learn which records are returned to the querier, the server learns nothing more about plaintexts than this information.

Additionally, the Song et al. protocol provides good performance by adhering to a few simple guidelines that subsequent works also follow:

1. The search protocols are simple and reminiscent of their insecure counterparts.

2. The scheme relies on faster symmetric-key cryptography rather than slower public-key cryptography. Song et al., in fact, use symmetric-key cryptography exclusively: schemes that follow their lead are collectively referred to as *searchable symmetric encryption*, or SSE. While some of the subsequent works described below do utilize public-key cryptography to provide additional functionality, they often use public-key cryptography as sparingly as possible.

3. Like most unprotected database search technologies, Song et al. bolster the performance of queries by pre-computing an *index* mapping keywords to records that match the query. While the bulk of Song et al.'s work is on non-indexed search, this observation foreshadows many of the subsequent works on SSE.

After the initial Song et al. result, a myriad of works improved the functionality, performance, and security of searchable encryption technology. We now present the various dimensions along which these works improved searchable encryption. Understanding these dimensions will be important for understanding the state-of-the-art in searchable encryption schemes.

Secure indices: Goh [77] formally defines the concept of index-based SSE. We note that indexing induces a performance tradeoff: to achieve faster query throughput, SSE schemes may require a long time to compute the index whenever the database is created or modified. In fact, Song et al. and Goh do not support efficient database modifications (insertions, updates, or deletions) into their indices due to the difficulty of doing so in a confidential manner (that is, without revealing the desired modification to the server).

Chang and Mitzenmacher [42] make the valuable observation that indexing provides a clean conceptual separation between two acts that the server must perform: determining the indices of records that match a query and fetching the contents of records at those indices. The first action might require complicated data structures to support sophisticated queries, but the contents of those data structures (i.e., the indices themselves) are small. The second action requires a large data structure to hold record contents but only requires that this data structure support the simplest of queries: a simple pointer-lookup.

With this observation, Chang and Mitzenmacher are able to use standard compression and encryption schemes on the record contents. Chang and Mitzenmacher's observation is utilized in many of the SSE schemes that follow, including all of the works presented in the State of the Art section below (Section 4.4).

Error rates: Goh [77] constructs a secure index mechanism whose search time is independent of the number of keywords in the database. To do so, Goh utilizes "Bloom filters," a simple fixed-size data structure that can check for membership in a variable-length set. Goh's indexing mechanism is simple: it builds one Bloom filter per record that includes all of its keywords. Because Bloom filters have a one-sided error, it is possible that the server could return records to the querier that do not actually match the keyword being searched. Subsequent SSE works that utilize Bloom filter technology, such as the Blind Seer [134, 60] scheme discussed in Section 4.4 below, also have non-zero false positive rates.

Modifications: Due to their separation of searching and file storage, Chang and Mitzenmacher [42] provide keyword search data structures that are much simpler than those of Song *et al* or Goh. As a result, Chang and Mitzenmacher's index structure supports modifications: insertions, updates, and deletions. While Chang and Mitzenmacher can only support modifications if they have an *a priori* dictionary or universe of all possible keywords that could be inserted later, subsequent works [167, 140, 98, 97, 40, 134] overcome this restriction and support faster modifications directly over the data structures they use for indexing.

Secure search with multiple queriers: Curtmola et al. [51] formally define SSE with multiple queriers. They do so in two settings: a *non-adaptive* setting in which security is only proved when the adversary must commit to all queries before seeing any responses, and a stronger, *adaptive*, setting in which the adversary is permitted to choose queries based on the responses to prior ones. While adaptively-secure cryptography is preferable for its relevance to real-world usage of SSE, non-adaptive protocols are faster with no known attacks in practice.

To aid in the multiple-querier setting, Chase and Kamara [43] provide an access control mechanism called *controlled disclosure*. This mechanism gives a data owner the ability to grant someone else access to a portion of a large dataset.

Public-key searchable encryption and outsourced SPIR: Public-key cryptography enables three-party scenarios (such as the secure publish-subscribe and email scenarios above) in which the server that stores data never has the ability to read or perform searches over it. Boneh et al. [26] provide the first construction of public-key searchable encryption in which the querier can produce a "trapdoor" token that permits the server (non-interactively) to locate all records containing a given keyword without learning anything else. We refer interested readers to Boneh and Waters [30] for a formal, generic definition of public-key searchable encryption. Many of the works in this space use a relatively novel, powerful,

non-standardized cryptographic building block called bilinear groups.

More recently, some of the works described in Section 4.4 below attempt to satisfy the security constraints of the three-party scenario (namely, having a server who isn't trusted to learn either the data or queries, as described in Section 4.2.2 above) while minimizing or removing the use of public-key cryptography in order to improve performance [41, 95, 40, 60, 134]. Jarecki et al. [95] refer to this concept as *outsourced SPIR*.

Searching over different data structures: Inspired by traditional databases, SSE focuses on the problem of record retrieval via text-based searches. But, just as with the rest of the big data community, encrypted search has progressed beyond this simple setup. Chase and Kamara [43] define *structured encryption*, which enables searches over arbitrary structured data such as matrices and graphs. Chase and Shen [44] generalize this notion even further to *queryable encryption*, which permits queries over potentially-unstructured data. Chase and Shen provide one form of queryable encryption: the ability to do a substring search over a large file.

Query expressivity: Researchers have steadily increased the expressivity of searchable encryption schemes. Collectively, modern schemes support a large percentage of the selections possible in, for instance, the `where` clause of the Structured Query Language (SQL).

From the beginning, searchable encryption schemes (and their predecessors in the history section above) supported equality keyword searches such as `select * where name = Alice`. Researchers subsequently added support for other types of queries, often by finding a new data structure to traverse in a secure fashion. We describe a few of the innovations in data structures below and explain the new query types that they enable.

- *Boolean queries*: Goh's usage of Bloom filters [77], as described above, facilitates secure searching for Boolean combinations (i.e., ANDs and ORs) of keywords. The same technique also permits searches for the number of occurrences of a keyword inside a record. Finally, by building a tree of Bloom filters, Goh's technique also facilitates searches for words that occur infrequently in a large dataset, which can be useful for statistical analyses of large datasets. Pappas et al. [134] use a similar Bloom filter technique, and Cash et al. [41] support Boolean queries in a different fashion.

- *Inequality and range comparisons*: A technique called *order-preserving encryption*, in which the order of ciphertexts matches the order of underlying plaintexts, enables a weak but incredibly fast form of range searches. Research into this technology began in the database community with the work of Agrawal et al. [4], and it has continued in the cryptographic community with several works that improve the performance and characterize the security limitations of this technology [23, 24, 139, 120]. On the other end of the spectrum, Boneh and Waters [30] provide a stronger but slower form of range comparison, subset, and conjunction searches.

- *Substrings*: Chase and Shen [44] use a suffix tree data structure to support searches

for substrings in a single, large file. Their work operates in the cloud outsourcing two-party scenario.

- *Similarity searches*: A few works provide for similarity or fuzzy searches. Park et al. 's scheme can be used for approximate string matching [135], and Adjedj et al.'s scheme can be used for biometric identification [2].
- *Inner products*: A few works provide for searches that involve basic arithmetic operations such as inner product computations [156, 34].

With support for a broad array of query types, searchable encryption technology is approaching the point of commercial interest [25].

Performance and scale improvements: Finally, we examine two methods utilized to improve the performance and scalability of searchable encryption: pre-processing and parallelization.

First, Freedman et al. [64] developed a technique to save on online query processing time by performing some pre-processing work between the querier and server (a technique that is also used in ORAM schemes). Second, Kamara and Papamanthou [97], Cash et al. [40], and Pappas et al. [134] discover opportunities for parallelization, both within a single query's execution and between several queries that are executed concurrently. With these performance improvements, these works are able to handle significantly larger databases, with the latter two works scaling to achieve good performance on terabyte-sized datasets. We discuss these two works, along with other modern, usable searchable encryption techniques, next.

4.4 State of the Art

The state of the art secure search schemes all make different choices in the dimensions described above resulting in different trade-offs between security, functionality, and performance. We now briefly describe several of the latest searchable encryption schemes. We focus on schemes supporting rich sets of query functionality rather than those focused on only one query type. Such schemes largely fall into one of two camps: Schemes focused on achieving very strong security, with minimal leakage, potentially at the cost of performance, and schemes focused on achieving performance close to unencrypted databases at the cost of allowing more leakage to an adversary. We now describe several schemes following each of these approaches.

4.4.1 Security-Focused Schemes

In the past few years there have been several efforts to build searchable encryption schemes providing strong security guarantees while providing rich query expressiveness and being able to scale to terabyte size databases. This work has largely been driven by the Intelligence Advanced Research Projects Activity (IARPA) Secure and Privacy Assurance Research (SPAR)

program [92] which aimed to produce prototype implementations of searchable encryption that could perform within an order of magnitude of standard SQL databases while minimizing the amount of leakage to the parties involved.

The two protocols to come out of this project are Blind Seer [134] and Cash et al.'s scheme [41]. Both of these work in the three party model described earlier and assume no collusion between the three parties: server, querier, and data owner. Both of these protocols achieved strong security against all three parties. Specifically, the data owner cannot learn the querier's queries and the querier can learn only information returned by authorized queries. The server can only learn limited information about the data it is storing and the queries it is serving, such as query search patterns. The exact details of the leakage provided to the server vary between the two protocols resulting in slightly different security guarantees.

Both of these protocols work by constructing secure search indices to allow for sublinear searching. This is done by combining cryptographic techniques with indexing structures from the database community. While the two schemes share this basic premise, the underlying technologies are drastically different and use different cryptographic primitives. We now briefly sketch these two protocols.

Blind Seer: The Blind Seer protocol constructs an index consisting of an encrypted search tree. The leaves of this tree correspond to the individual records and contain all the searchable keywords from that record. Internal nodes contain a Bloom filter[5] storing the set of all keywords stored in their descendants. Now, a boolean query (e.g., A AND B) can be executed as follows:

- Beginning at the root, check to see if the Bloom filter of the current node contains the keywords being searched for in the query. If not, terminate the search.
- If so, visit all the children of the node recursively for the same check until the leaf nodes are reached.
- Return all leaf nodes containing the searched for keywords.

To additionally support range queries and queries including negations, the set of keywords in the leaf nodes is extended to include negated terms and also special range keywords.

However, just performing the queries as described above would leak both the query and the data to the server performing the search. Instead, the Bloom filters are encrypted to protect their content and the Bloom filter check is performed by a secure computation between the client and server which allows one to check membership in a Bloom filter while hiding the content of the Bloom filter and the value being looked for. Now the client and server can together traverse the search tree as before with the server only learning the search pattern of which tree nodes are visited by the search, but not the keywords contained in those nodes or the search terms.

[5] A data structure allowing for efficient set membership queries

Blind Seer also supports modifications (which includes inserts, deletions and updates) within their secure index. Inserts are handled by creating a separate Bloom Filter with is included as part of the search. Deletions are handled by marking the corresponding nodes, both in the original and inserted Bloom filters, as deleted. Updates are implemented as simply an insertion and deletion. From time to time the entire tree of Bloom Filters is recreated to include these inserted records.

Blind Seer's performance was measured as part of the SPAR program by MIT Lincoln Laboratory [168] and can be seen in Figure 2. For the most part they function within an order of magnitude of a MySQL baseline system.

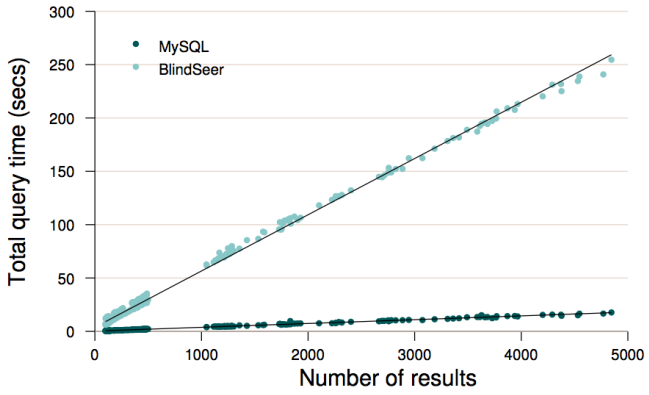

Figure 2: Performance of Blind Seer compared to a MySQL database. Performance cost is within 15x - mostly due to network communication costs. Figure taken from [134]

Cash et al.: The Cash et al. protocol constructs an encrypted index based on the idea of an expanded inverted index. In their basic scheme, without factoring in security, they store two different indices. The first is an inverted index that maps keywords to record identifiers, and the second is an index that maps record identifiers to keywords contained within. In order to do a conjunctive search, the server looks in the first index for the first keyword in the conjunction. For each of the records that matches the keyword, the server checks to see if the rest of the terms of the query are matched by checking the second index. If they do, that particular record is returned as part of the query.

To add security to the above search protocol, first the records in the database are permuted and encrypted. Then, the indices are built for the permuted encrypted database. Furthermore, during a search careful cryptographic techniques guarantee that the server can check the second index for keywords included in the query without learning what the keywords are or learning anything about the remaining keywords contained in the record.

Like Blind Seer, the Cash et al. protocol is able to achieve reasonable performance compared to standard search technologies. Specifically, the query time for queries returning small result sets is essentially independent of the size of the database for databases up to

one billion records. For such queries, this protocol is almost comparable to the runtime of an unencrypted MySQL backend.

Bajaj and Sion: An alternative approach for security-focused secure search was given by Bajaj and Sion [9] who use trusted hardware such as IBM 4758 and 4764 PCI cards in place of cryptographic assumptions. Here security stems from requiring the server to perform the sensitive portion of their searches properly on a trusted hardware device thus guaranteeing security of the data and correctness of the searches. Specifically, security is guaranteed as long as adversaries cannot tamper with the internal state of the trusted hardware.

Bajaj and Sion use this approach to build a full relational database whose data structures and query processing methods are split into secure pieces run in a trusted enclave and insecure components that run on commodity hardware. To analyze the efficiency of this approach Bajaj and Sion estimate the cost of computations, storage, and networking in both small-scale and large-scale ("the cloud") computing environments. They conclude that large-scale computations inside secure processors is only 2-4× more costly than computation in small-scale insecure enterprises, and thus orders of magnitude faster than any cryptographic operation. Thus, this approach allows them to build trusted hardware based secure search schemes achieving strong security while also getting all the desired query functionality and scalability.

4.4.2 Performance-Focused Schemes

On the other end of the spectrum there have been several efforts trying to add a minimal level of security to database search while achieving minimal performance overhead. Moreover, these techniques allow the use of an unmodified traditional database as the server in the secure search protocol thus allowing backward compatibility with existing infrastructure.

CryptDB: CryptDB, built by Popa et al. [140], provides a client side library that supports executing SQL queries over encrypted data. The key insight that allows them to do this is that various forms of encryption allow for different queries to be performed directly on encrypted data. Thus, by encrypting data accordingly, it is possible to allow searching over encrypted data. Specifically, Popa et al. use the following encryption schemes:

- **Semantically Secure Encryption** provides the strongest security guarantees; the encryption process is randomized and encryptions of different messages are indistinguishable. However, this type of encryption, while the most secure, does not allow for any specific queries to be made across it.
- **Deterministic Encryption (DET)** [12] provides a weaker security guarantee; two equal values always encrypt to the same ciphertext, which allows the server to check equality of the underlying plaintexts to perform frequency analysis. However, this allows a client to make targeted keyword search across deterministically encrypted columns as these only require checking equality.

- **Order Preserving Encryption (OPE)** [139], in addition to being deterministic, also preserves the relative order of the underlying plaintexts. So, if $a < b$, then $\mathsf{Enc}(a) < \mathsf{Enc}(b)$. This means it provides an even weaker security guarantee than deterministic encryption, allowing the server to learn the order of the underlying values. It does, however, allow a user to make range queries.
- **Additively Homomorphic Encryption** has similar security to semantically secure encryption but has the added property of allowing limited computations, in this case addition, across the encrypted data in the columns. This is further discussed in Section 5.1.1.

CryptDB uses these different encryption types by encrypting the different fields in a SQL database using different types of encryption. Fields over which ranges may be desired are encrypted using OPE, fields that need to support keyword queries are encrypted using DET, and raw data fields that cannot be searched over can be encrypted using semantically secure encryption. Additionally, CryptDB allows for joins over fields encrypted under OPE and DET as this just requires equality checks. In order to reduce the amount of leakage from such encryptions, all fields encrypted using OPE and DET are *onion encrypted* with a semantically secure encryption scheme. That is, the these ciphertexts are again encrypted under a semantically secure encryption, thus removing all leakage. They remain encrypted in this way until a query is made that requires equality or range search over this field at which point the semantically-secure layer is decrypted off to allow the query to be performed. The benefit of this is that no information is revealed about values in fields that are not searched over.

Even with this measure, CryptDB offers a much more limited form of security than that found in the Blind Seer and Cash et al.'s solutions since once the outer layer of encryption is removed on some field, the server learns some information about the values in that field for all records, even the ones not returned by the query. However, the performance of CryptDB is much closer to traditional search technologies, performing within $2\times$ for most query types as can be seen in Figure 3.

Monomi Monomi [166] takes CryptDB's advances and extends its performance results by introducing a query and schema planner. This planner, based on the variables of client storage and allowed communication complexity (number of rounds of communication and the size of the portion of the database sent over for each query) splits the query execution between server-supported portions (such as range queries over an order preserving column) and client side computation. The schema and additional encrypted columns are also automatically generated based on a sample set of queries that the client inputs at generation time. This effort to apply traditional query planning to encrypted search technology allows for more expressive queries and better performance than CryptDB demonstrates.

Computing on Masked Data Computing on Masked Data (CMD) [65] extends the ideas of CryptDB to NoSQL databases to support big data applications. Specifically, CMD applies

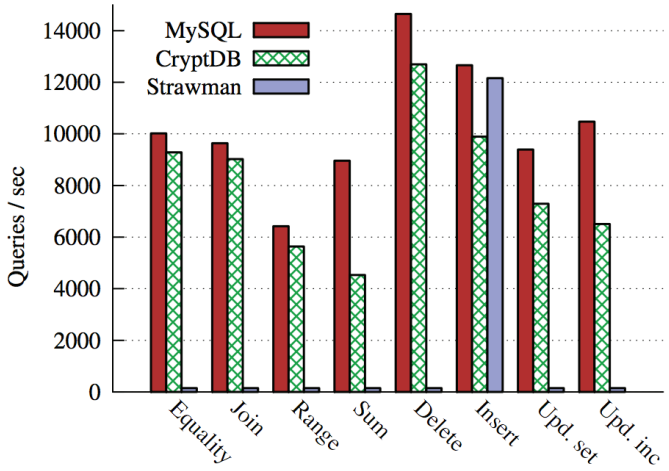

Figure 3: Performance of CryptDB for a variety of query types. Strawman is a naive implementation of the algorithm without optimizations to speed up the construction of the encrypted indexes. Figure taken from [140]

these protections to the Accumulo database [63], a distributed database noted for its high ingest rates and native cell-level security. CMD supports a similar set of encryption types to CryptDB, including semantically secure, deterministic, and order preserving encryption. CMD targets, and achieves, a performance goal of computing on the encrypted data within a factor of 2× of the unencrypted system. They note that compared to the performance cost of performing the query, the performance cost of encrypting and decrypting the data is relatively small.

5 Secure Data Processing

In the previous section, we discussed performing secure searching on encrypted data. However, in many big-data applications it is not enough to be able to simply retrieve stored data. Instead, it is desirable to perform analytic computations over the data and return only the result of these computations rather than the original data. We stress that it is very important that the data and the computations be protected even while the processing is being performed. In this section, we describe four cryptographic techniques that enable generic secure computation: *homomorphic encryption* (HE), *verifiable computation* (VC), *secure multi-party computation* (MPC), and *functional encryption* (FE).

All of these can be used to securely outsource the processing of data but in different scenarios. Homomorphic encryption allows computing on encrypted data while maintaining confidentiality; it can be used to outsource processing of sensitive data to another entity that is trusted to perform the computation correctly but should not learn the data. Verifiable computation allows computing on data and allowing the integrity of the computation to be checked; it can be used to outsource processing of data to another entity that is allowed

to learn the data but not trusted to perform the computation correctly. It is possible to combine homomorphic encryption and verifiable computation to achieve both confidentiality of the input and output and integrity of the computation. Secure multiparty computation allows performing a distributed computation on sensitive inputs held by multiple parties while maintaining confidentiality of each party's inputs from every other party and ensuring that the computation was performed correctly. MPC guarantees these properties as long as no more than some threshold of computing parties are adversarial. MPC can be used in settings where different sensitive inputs are held by different parties or in settings where a single client wants to outsource computation on its sensitive input by distributing the computation over multiple compute nodes. Finally, functional encryption allows a data owner to let others learn specific functions of her sensitive data and nothing else.

5.1 Homomorphic Encryption

5.1.1 Introduction

Imagine that we own a computationally weak computer and wish to outsource some work to a server that we do not trust. The data on which the work is done is sensitive, and we want to make sure that the server can't learn anything about it. Although the idea of delegating a computation without revealing anything about the inputs or outputs may seem very unintuitive, cryptography has a tool enabling just that: this tool is *homomorphic encryption*.

5.1.2 Definition

Homomorphic encryption is a type of public-key encryption that allows anyone to compute functions of data while it is encrypted without decrypting it or learning anything about it. Specifically, in addition to the algorithms KeyGen, Enc, and Dec associated with any encryption scheme, homomorphic encryption also has a fourth algorithm Eval. This algorithm takes in the public key pk, a function f, and encrypted data c_{in}, and returns an encrypted output $c_{out} = \mathsf{Eval}(pk, f, c_{in})$ such that c_{out} is an encryption of the result of applying f to the originally encrypted data; that is, $f(\mathsf{Dec}(sk, c_{in})) = \mathsf{Dec}(sk, c_{out})$. Anyone can evaluate Eval, since it does not require the secret key sk. Note that the result is an encryption of the result; retrieving the result itself requires the secret key. Moreover, the evaluation does not reveal anything about the data c_{in} or the computation output c_{out}, since that would violate semantic security.

5.1.3 Survey

Many public-key encryption schemes are naturally homomorphic with respect to a limited set of functions f, such as only addition or only multiplication [66, 133, 142, 149]. An intriguing question, initially posed in 1978 by Rivest, Adleman, and Dertouzos [148], is whether there exists an encryption scheme that simultaneously permits the evaluation of (an unbounded

number of) *both* addition and multiplication operations on the underlying plaintexts. Since these two operations constitute a logically complete set of operations, such a scheme would be a *fully homomorphic encryption* (FHE) scheme, meaning that it is homomorphic with respect to all functions f. Boneh, Goh and Nissim [28] give an encryption scheme that is homomorphic with respect to quadratic functions f; that is, their encryption scheme supports the evaluation of arbitrarily many additions, but only one multiplication, thus falling short of this goal.

In 2009, the seminal work of Gentry [71] affirmatively answered this long-standing question by demonstrating an encryption scheme that provides Eval operations for an unbounded number of both additions and multiplications. Gentry split the FHE problem into two components: the design of a *somewhat homomorphic encryption scheme* (SWHE) that permits a limited number of Eval operations, and the insight of a *bootstrapping* algorithm that achieves fully homomorphic encryption through the use of multiple applications of SWHE. A slight tweak of this initial scheme was implemented by Gentry and Halevi [72].

Somewhat Homomorphic Encryption: There are many versions of Gentry's somewhat homomorphic encryption scheme. One of the most straightforward ones uses integers modulo a large prime p, which is the secret key. Ciphertexts take the form of large integers, which are encryptions of 0 if their remainder modulo p is even, and encryptions of 1 if their remainder modulo p is odd. It is straightforward to see that adding (or multiplying) two ciphertexts gives an encryption of the sum (or product) of the plaintexts, as long as the remainders modulo p (i.e., the *noise*) of the two original ciphertexts are small enough. Informally, this scheme is secure because of the *learning with errors* assumption [146], which says that if p is unknown, then multiples of p with some small additive noise look completely random, and an observer can't learn p.

Bootstrapping: In the scheme described above, the error keeps growing until it is impossible to perform any more homomorphic evaluations. However, Gentry made the key observation that if a scheme is homomorphic enough to evaluate its own decryption circuit in addition to at least one other operation, then it can be made fully homomorphic by means of *bootstrapping*. Along with the public key, an encryption under that public key of the corresponding secret key is published. When the ciphertext noise grows so large that no more homomorphic operations can be performed, the ciphertext is encrypted again. This results in two layers of encryption – the original one and the new one. Under the new layer of encryption, the old ciphertext is homomorphically decrypted (using the encryption of the secret key), resulting in a ciphertext with just one layer of encryption – the new layer. The noise of this new ciphertext is only whatever is accrued over the course of homomorphically evaluating a single decryption circuit. So, as long as the decryption circuit isn't deep enough to generate too much error, one or more additional homomorphic operations can be performed before the error grows too large again.

Gate Type	Count	Mean	Std Dev	Min	Max
Add	101	$2.04 \cdot 10^{-4}$	$1.99 \cdot 10^{-4}$	$1.10 \cdot 10^{-5}$	$6.18 \cdot 10^{-4}$
Add constant	101	$1.85 \cdot 10^{-4}$	$1.75 \cdot 10^{-3}$	$6.00 \cdot 10^{-5}$	$4.43 \cdot 10^{-3}$
Multiply	101	$1.45 \cdot 10^{-2}$	$1.39 \cdot 10^{-2}$	$4.76 \cdot 10^{-4}$	$3.00 \cdot 10^{-2}$
Multiply by constant	101	$1.92 \cdot 10^{-3}$	$1.82 \cdot 10^{-3}$	$7.60 \cdot 10^{-5}$	$5.19 \cdot 10^{-3}$

Figure 4: This table describes the evaluation time (in seconds) of some operations in HElib [87]. The rows describe the operations on ciphertexts that correspond to the following operations: addition of two plaintexts, addition of a public constant to a plaintext, multiplication of two plaintexts, and multiplication of a plaintext by a public constant. The columns describe statistics of the time it takes to evaluate a single gate of each type: that is, the number of evaluations performed (count), the mean evaluation time, the standard deviation, the minimum, and the maximum. The numbers are taken from Varia, Yakoubov and Yang [121].

5.1.4 State of the Art

After Gentry's initial work, many cryptographers have designed new FHE algorithms that make substantial improvements along several dimensions:

- Decreasing the ciphertext and key sizes [157].
- Optimizing the FHE for specific common operations such as SIMD [158, 73]. SIMD operations involve performing the same operations on multiple bits in parallel.
- Using hardware (e.g., FPGAs [141]) in order to make FHE faster.
- Increasing the number of Eval operations possible in a SWHE scheme before bootstrapping, eliminating bootstrapping altogether, or making bootstrapping more efficient [35, 75]. Bootstrapping is the most expensive part of homomorphic evaluation, and reducing the frequency with which it needs to be done greatly increases evaluation efficiency. One of the approaches to this is reducing the noise a limited number of times by changing the ciphertext in a different way ("modulus reduction") between every two bootstrapping operations.
- Basing the cryptography upon weaker, more accepted assumptions [36].

Some of these improved algorithms have been implemented in software [74, 87, 88, 58]. Figure 4 describes the homomorphic evaluation time of some operations in the HElib implementation [87].

5.2 Verifiable Computation

5.2.1 Introduction

Imagine that, as in the previous section, we own a computationally weak computer and wish to outsource some work to a server that we do not trust. Unlike the scenario in the previous

section, suppose we now do not care about the confidentiality of the data (perhaps this data is not sensitive), but we do care that the work is done correctly.

There are a number of ways that one might approach this without using cryptographic tools. The most straightforward approach is *replication* – outsourcing the computation to several different servers, and then taking the most common answer as correct. However, this only works as long as the failures of the servers are not correlated (as in Byzantine fault tolerance [111], which is not always the case. It could be that most of the servers in the cluster are controlled by the same adversarial entity, or that they are running a single operating system that is faulty in a specific way. Another way is *auditing* – after outsourcing the work, also performing it oneself with some probability, and if the answer does not agree with the server's answer, ceasing to trust the servers to do any of the work. This only works as long as one believes the servers will fail on a noticeable fraction of the computations, and not on just a few of them (which may be vitally important to get correct). If we use cryptographic tools, the landscape is much more promising. *Verifiable computation* is a cryptographic tool that guarantees the integrity of outsourced computations without making any assumptions about the server failure rate or correlation of failures.

5.2.2 Definition

Typically, verifiable computation is defined in the context of two parties: a computationally weak *verifier* (or *client*), and a computationally strong but untrusted *prover* (or *server*) to whom the verifier delegates some work. Given an input x and a function f to evaluate on x, the prover is expected to produce an output y, as well as a proof π that $y = f(x)$ that the verifier can use to confirm the correctness of the computation. In order for the verifier to want to outsource the computation to the prover in the first place, it should be much more efficient for the verifier to check the proof of correctness π of the output y than for the prover to perform the computation itself. Verifiable computation schemes should make it infeasible for a prover to come up with an incorrect output $y^* \neq f(x)$ and a proof π^* such that π^* convinces the verifier that $y^* = f(x)$.

5.2.3 Survey

Most verifiable computation constructions are based on *probabilistically checkable proofs*, or *PCPs* [8]. A PCP is a (possibly long) string produced by the prover to prove the validity of some statement. The PCP itself can be viewed as the proof of the statement's validity; however, having to read the entire PCP may be too big a burden on the computationally weak verifier. Instead, the special property of a PCP is that the verifier can check the validity of a PCP by looking only at a constant number of random locations in the PCP. This works because any invalid PCP is necessarily inconsistent in a large number of locations, so it is detectable as invalid by the verifier with high probability. This powerful machinery has found ample uses in cryptography and theoretical computer science, and we refer readers to the book by Arora and Barak [7] for an excellent overview.

However, PCPs alone do not give us verifiable computation; there has to be some way for the prover to produce *and fix* a PCP without sending the whole thing to the verifier. Having the prover store the PCP and answer queries from the verifier on the fly won't work, because the prover could then cheat by changing parts of the PCP in reaction to the verifier's queries. In the next few paragraphs, we describe several ways to build verifiable computation on top of PCPs, as presented in [169].

Verifiable Computation Using Interaction: Goldwasser, Kalai, and Rothblum [81] and later Cormode, Mitzenmacher, and Thaler [50] describe using interactive proofs for verifying computation. Allowing the prover and verifier to interact (instead of requiring the prover to send a fixed string π to the verifier) makes it hard for the prover to lie in a way that the verifier won't eventually catch him. Instead of "naively" asking questions about values at specific locations of the PCP, the verifier asks queries in an adaptive fashion such that without a valid PCP, the prover will eventually be forced to contradict himself. Roughly, the prover's early answers commit him to a lie and the verifier's later queries make it impossible for the prover to stick to that lie.

Verifiable Computation Using Commitments: Another line of work (e.g., Killian [103]) uses *cryptographic commitments*. A cryptographic commitment is a digital object that binds the prover to a certain statement without revealing the statement. The commitment can be much smaller than the statement itself. Later, when the prover produces the statement itself (or, for some commitment schemes, even part of the statement), the verifier can use the commitment to check that the statement is in fact the one the prover committed to earlier. If the prover computes a commitment c to the entire PCP, and sends c to the verifier, the verifier can then query the prover on the parts of the PCP he wishes to view, and the prover will have to answer honestly, because if he changes parts of the PCP the verifier will be able to tell that they no longer match the commitment c.

Verifiable Computation Using Homomorphic Encryption: Additively or multiplicatively homomorphic encryption (described in Section 5.1.1) can be used to conceal the verifier's queries about the PCP, but still allow the prover to answer them. Since the prover only sees the queries in encrypted form, he has no way to know how to adapt his PCP in response to them. A few constructions based on this paradigm include Ishai, Kushilevitz and Ostrovsky [94], Ben-Sasson et al.[17], and Bitansky et al.[21]. One advantage of this approach is that it allows the verifier's queries to be reused, reducing the amount of interaction required.

5.2.4 State of the Art

A great survey of practical verifiable computation implementations is given in [169]. Implementations include Pinocchio [136], SNARKs for C [17], and many others. Figure 5 describes some aspects of their relative performance.

	Evaluation and proof generation time	Verification time
Pinocchio [136]	144s for 277,745 multiplications	10ms
SNARKs for C [17]	31s for 262,144 operations	5s

Figure 5: This table describes some aspects of the relative performance of Pinocchio [136] and SNARKs for C [17].

5.3 Multi-Party Computation

5.3.1 Introduction

In many situations, several parties with sensitive data wish to jointly compute some function of the collection of their data. For example, hospitals may want to perform medical research on their combined patient data, companies may want to predict cyber threats by analyzing threat information from companies in the same sector, or individuals may wish to participate in an auction with sealed bids. In all of these situations, each party wants to learn the result of the computation on the data held by all the parties but without sharing their own sensitive information. Furthermore, there may not be a trusted party to whom everyone is willing to give their information and that everyone trusts to perform the computation securely and correctly.

Secure multi-party computation (MPC) is an area of cryptography that addresses this problem. MPC protocols allow people to perform distributed computation on their private data without revealing their data and without the use of a trusted party. As we will describe below, MPC protocols exist for any computable function and for any number of parties. Thus, MPC can not only provide privacy for existing applications in which parties currently share their data but would prefer not to; MPC can also enable new applications that are currently impossible because parties are unwilling or unable to share their data.

5.3.2 Definition

Secure multi-party computation allows two or more parties to compute a joint function of their inputs in such a way that everyone learns the correct output of the function but no one learns any other information, even when some of the parties may be adversarial.

In an ideal world, there would be a party that everyone trusts with their private inputs and trusts to perform the computation correctly. Then the MPC problem would be solved trivially: everyone would send their inputs to the trusted party, and the trusted party would perform the computation and distribute the outputs. However, in the real world, we cannot assume there is a trusted party. The security of an MPC protocol guarantees essentially that it emulates the ideal-world behavior.

MPC generally considers two types of adversaries: semi-honest and malicious (as described in Section 2.1). We assume collusion among adversarial parties (as described in 2.1). In fact, we can think of there being a single adversary who "corrupts" some subset of the parties, observing their inputs and outputs and (in the case of a malicious adversary) controlling their behavior. We will also distinguish between computational security, which is

based on the assumed hardness of some computational problem, and information-theoretic security, which is unconditional security based on bounds on the amount of information an adversary can learn.

5.3.3 Survey

The fundamental results of MPC have demonstrated that protocols exist to compute *any* function securely. In the rest of this section, we will consider *generic* MPC protocols; that is, protocols that describe how to compute any function securely, as opposed to ones that are specific to restricted classes of functions. Generic MPC protocols are typically designed for *boolean* or *arithmetic circuits*. A circuit is a directed acyclic graph consisting of gates, where each gate takes input values and produces an output value that may feed into other gates as input. One can think of a boolean circuit as a circuit operating on bits with XOR and AND gates. An arithmetic circuit operates on elements of a finite field with addition and multiplication gates.

Early results showed that generic MPC is possible. Yao [171] gave the first two-party protocol for computing functions represented as boolean circuits using a technique called *garbled circuits*, providing computational security against a semi-honest adversary. Yao's protocol requires a constant number of rounds of communication. Goldreich, Micali, and Wigderson [78] made two contributions: (1) the first multi-party protocol for two or more parties, also for boolean circuits, with computational security against a semi-honest adversary, and (2) a general compiler for transforming any protocol with semi-honest security to one with malicious security. These protocols requires communication for every multiplication gate, so the number of rounds of communication is proportional to the depth of the circuit. Next, Ben-Or, Goldwasser, and Wigderson [16], and Chaum, Crépeau, and Damgård [45] simultaneously presented the first multi-party protocols with information-theoretic security; these protocols work on functions represented as arithmetic circuits. We now describe the main ideas behind some of the foundational protocols and subsequent improvements for boolean and arithmetic circuits.

Boolean circuits: Most MPC protocols for securely evaluating boolean circuits are based on Yao's garbled circuits. Yao's garbled circuit approach uses a primitive called *oblivious transfer*. In its simplest form, oblivious transfer involves a sender with two private values and a receiver with a private bit; the result of the oblivious transfer is that the receiver learns the value specified by its bit, while the sender learns nothing about which of its two values the receiver obtained.

Yao's protocol for securely evaluating boolean circuits works as follows. One party will be the circuit generator (call her Alice) and the other will be the circuit evaluator (call him Bob). For each wire in the circuit, Alice randomly chooses two secret keys, one corresponding to each of the input bit choices 0 and 1. For each gate, Alice creates a truth table in which the output wire key is encrypted under the pair of the corresponding input wire keys. Alice permutes the truth table to hide the order of the rows and sends it to Bob along with the

keys corresponding to her inputs. These keys are random, so they do not reveal to Bob anything about Alice's inputs. Then, using oblivious transfer, Bob obtains the keys from Alice corresponding to each of his inputs without Alice learning anything about Bob's inputs. For each gate, Bob decrypts the appropriate entry in the truth table to get the output wire key. After evaluating the entire circuit in this way, Bob tells Alice the final output key and Alice determines whether the output is 0 or 1. Throughout the computation, neither party learns any information about each other's inputs except for the output of the function at the end.

Subsequent works have improved on Yao's protocol in several dimensions, including reducing the ciphertext size, reducing the computational cost, and adding malicious security.

Ciphertext size While Yao's original protocol requires transmitting 4 ciphertexts for the truth table of each gate, subsequent works have reduced the required number of ciphertexts. Naor, Pinkas, and Sumner [126] showed how to use only 3 ciphertexts per truth table, and Pinkas et al. [137] further improved this to only 2 ciphertexts per truth table. Kolesnikov and Schneider [105] showed how to garble XOR gates "for free" without any ciphertexts; however, in this method AND gates still require 3 ciphertexts. Kolesnikov, Mohassel, and Rosulek [104] gave a method that garbles XOR gates using between 0 and 2 ciphertexts while AND gates require only 2 ciphertexts. Finally, Zahur, Rosulek, and Evans [172] show how to garble XOR gates with no ciphertexts and AND gates with only 2 ciphertexts.

Computational cost There has been progress reducing the computational cost of garbling by using different techniques for encrypting a wire label under two keys. Naor, Pinkas, and Sumner [126] showed how to do this using 2 hash evaluations. Lindell, Pinkas, and Smart [114] reduced this to 1 hash evaluation. shelat and Shen [155] showed how to use a single block cipher evaluation, and Bellare et al. [14] further improved the efficiency by eliminating the key schedule, an expensive part of the block cipher evaluation. Another line of work reducing the computational cost of garbling was initiated by Huang et al. [91], who observed that it is possible to pipeline the circuit generation and circuit evaluation so that the entire circuit does not have to generated and stored all at one time before the computation.

Malicious security The original protocol of Yao is not secure against malicious adversaries because the circuit generator could cheat by sending a garbled circuit for a different function. One technique for providing malicious security for Yao's garbled circuits, introduced by Lindell and Pinkas [113], is called "cut and choose." In this technique, the circuit generator generates many copies of the circuit and sends them to the circuit evaluator. The circuit evaluator randomly chooses a subset of these to be opened to check that they indeed evaluate the correct function, and evaluates the other copies of the circuit. If the circuit generator cheats, the circuit evaluator will detect it with high probability.

Arithmetic circuits: Many MPC protocols for evaluating arithmetic circuits are based on a technique called *secret sharing*. Here we describe secret sharing and then present the high-level idea of one secret-sharing based MPC protocol.

A t-out-of-n secret sharing of a secret value x creates n values (shares) such that the

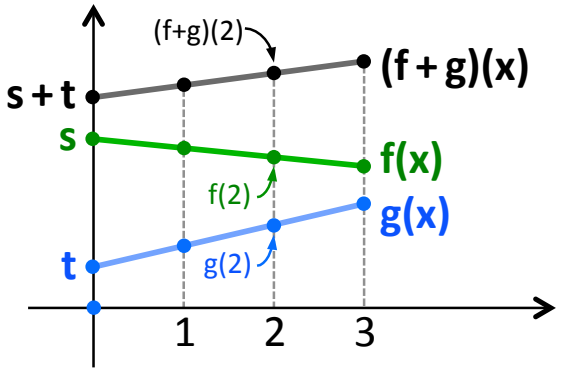

Figure 6: Graphical illustration of the addition of shares in a secret sharing scheme.

combination of any t or more shares can be used to reconstruct x, while any fewer shares reveal no information about x. Secret sharing can be performed using polynomials [154], as shown in Figure 6. To do this, choose a random degree-$(t-1)$ polynomial whose constant term is the secret value, and give each party a distinct point on the polynomial. Then any t parties can perform polynomial interpolation and recover the polynomial and the secret value, while any fewer parties learn no information about the polynomial or the secret value.

The Ben-Or, Goldwasser, Wigderson (BGW) [16] protocol is based on secret sharing. First, each party secret-shares her input with the other parties. Then, the parties perform arithmetic operations on the secret-shared values by manipulating the shares. Specifically, addition of two secret-shared values can be performed by each party adding their shares locally and then jointly reconstructing using the resulting shares. Similarly, multiplication of two values can be performed by multiplying the corresponding shares; however, the resulting shares then lie on a higher-degree polynomial, so an interactive degree reduction procedure is required. Thus, the protocol for any function represented in terms of addition and multiplication operations can be evaluated securely.

Assuming secure pairwise communication channels between the parties, the BGW protocol provides information-theoretic, semi-honest security against up to 1/2 of the parties being corrupted. Using additional techniques, BGW provides malicious security against up to 1/3 of the parties being corrupted. Improvements by Rabin and Ben-Or [144] and by Beaver [11] provide malicious security against up to 1/2 of the parties being corrupted, assuming that the parties have a broadcast channel. A more recent line of work has addressed malicious security against more parties being corrupted. Bendlin et al. [18] and Damgård et al. [53] achieve malicious security against up to all but one of the parties being corrupted, using a preprocessing "offline" phase that contains the expensive public-key operations and is computationally secure, followed by an efficient "online" phase for the actual computation that is information-theoretically secure.

5.3.4 State of the Art

Several MPC compilers and programming frameworks have been implemented and made publicly available.

Two-party computation: Fairplay [118] was the first compiler for secure two-party computation, using garbled circuits. TASTY [89] combines garbled circuits and homomorphic encryption. Since the development of these early compilers, there have been many implementations of garbled circuits with various optimizations. These include FastGC [91], Kreuter, shelat, and Shen [107], CBMC-GC [90], PCF [106], JustGarble [14], and TinyGarble [161]. A couple compilers such as SCVM [115] and ObliVM [116] have support for oblivious RAM (described in Section 4), evaluating secure dynamic memory accesses more efficiently than performing a linear scan of memory as in circuit representations. Compilers such as PCF, SCVM, ObliVM, and Wysteria [145] use an intermediate representation of circuits that allows generating the circuit on the fly instead of all at once before computation, reducing the compilation time.

Multi-party computation: FairplayMP [15] extended Fairplay to multiple parties. VIFF [52], SEPIA [37], Choi et al. [46], Sharemind [22], PICCO [173], and Wysteria [145], compile code to secret-sharing based schemes for more than two parties. SEPIA, PICCO, and Sharemind distinguish between input parties, who secret-share their inputs among the parties but don't necessarily participate in the computation, and computational parties, who perform the actual computation. Sharemind works specifically in the setting of three computational parties. Wysteria offers formal security guarantees through a special type system.

5.4 Functional Encryption

5.4.1 Introduction

We now consider a situation in which a data owner wishes to allow certain computations to be performed on their sensitive data, but wishes to retain strict control over this data and computations. For example, consider a cloud-based email service where users may want the cloud to be able to perform spam filtering on their encrypted emails but learn nothing more about the contents of these emails. Here the user wants to allow the cloud to compute and learn the output of a specific function on their data without giving the cloud any more access to their emails. A cryptographic primitive that enables this capability is *functional encryption*, which achieves both the access control properties of attribute-based encryption (described in Section 3.2) and the computing on encrypted data of homomorphic encryption (described in Section 5.1.1), allowing a data owner to control exactly who can compute on their data and which they functions they can compute.

5.4.2 Definition

Functional encryption (FE) is a public-key encryption scheme where, in addition to regular secret keys used to decrypt the data, there are functional secret keys. These are keys that, instead of decrypting the data, give access to the result of the corresponding function evaluated on the data. More formally, the KeyGen procedure now takes a function f and returns a key sk_f. Then $\mathsf{Dec}_{sk_f}(c)$ returns $f(x)$ where $c = \mathsf{Enc}(x)$. The security guaranteed by FE is that someone holding the key for function f learns no more information about the data x than is revealed by $f(x)$. As in the case of ABE, the critical property here is collusion resistance. Specifically, FE guarantees that even someone holding keys for multiple functions cannot learn more than the outputs of the corresponding functions.

5.4.3 Survey

The concept of functional encryption was first proposed in 2005 by Sahai and Waters [152] as a generalization of attribute-based encryption. The first formal treatment of this powerful primitive was given by Boneh, Sahai, and Waters [29] and O'Neill [131]. However, it was quickly pointed out that there are some very strong limitations to the types of FE schemes that can be constructed in terms of efficiency, security, and generality [5, 39].

It turns out that finding the "correct" definition for security of FE is quite non-trivial. The aforementioned work demonstrated that some natural definitions known as *indistinguishability-based* definitions are in some ways too weak to properly capture the desired security, while other definitions known as *simulation-based* definitions are too strong in that they cannot be satisfied by any scheme that works for all functions, as most existing schemes do. Thus, a lot of work has gone into understanding the relative power and security of these definitions [5, 39].

Despite these limitations, a number of constructions of FE for arbitrary functions have appeared in the academic literature. Most of these focus on a restricted class of schemes, where only a single functional key can be given out over the course of the scheme [151, 80]. This limitation was improved slightly by Gorbunov et al. [84] to allow the scheme to issue a bounded number of keys and thus tolerate a bounded number of collusions. However, the parameters of this scheme grow with the collusion bound, making it unlikely to be practical unless one is willing to assume a very small bound on the number of colluding parties. A somewhat different approach is taken by Naveed et al. [127] who instead require interaction between the data owner, who generates the keys for functions over his data, and any party wishing to evaluate a function on the encrypted data. This solution also seems unlikely to be practical over big data sets unless only a small subset of the data will ever be computed on. Several very recent works have shown ways to achieve security for FE against unbounded collusion. However, the security of all such schemes relies on very strong, though not fully accepted assumptions, such as the existence of indistinguishability obfuscation [67] or the hardness of certain problems over multi-linear maps [69].

5.4.4 State of the Art

FE offers great promise to enable very tightly controlled access to computation over sensitive data. However, it is currently still mostly in the realm of theoretical research. Existing schemes are most likely too inefficient to be used in practice and rely on very strong security assumptions. Additionally, very few reference implementations of FE schemes exist, making it very difficult to evaluate their performance and applicability for real-world use-cases. In fact, at the time of writing, we are only aware of one prototype implementation of FE in existence today [127].

Thus, FE is not currently in a state ready to be used in today's big data applications. However, we believe that this primitive will play an important role in securing computation on sensitive data in the future. By including functional encryption in our survey, we hope to raise awareness about this powerful primitive and draw interest to push researchers to identify appropriate applications for it and to develop schemes achieving appropriate performance for these applications in real-world deployments.

6 Conclusion

In this chapter, we have presented a wide variety of cryptographic techniques for securing data in transit, in storage, and in use. We believe that, as security becomes a critical requirement for sensitive big data processing, these techniques will become an integral part of the big data ecosystem. We hope that the exposition in this chapter will raise awareness of the latest types of tools and protections available for securing big data. We believe better understanding and closer collaboration between the data science and cryptography communities will be critical to enabling the future of big data processing.

References

[1] *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*. IEEE Computer Society, 2013.

[2] Michael Adjedj, Julien Bringer, Hervé Chabanne, and Bruno Kindarji. Biometric identification over encrypted data made feasible. In Atul Prakash and Indranil Gupta, editors, *Information Systems Security, 5th International Conference, ICISS 2009, Kolkata, India, December 14-18, 2009, Proceedings*, volume 5905 of *Lecture Notes in Computer Science*, pages 86–100. Springer, 2009.

[3] National Security Agency. Suite B cryptography. https://www.nsa.gov/ia/programs/suiteb_cryptography/. Accessed: 2015-06-26.

[4] Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu. Order-preserving encryption for numeric data. In Gerhard Weikum, Arnd Christian König, and Stefan Deßloch, editors, *Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France, June 13-18, 2004*, pages 563–574. ACM, 2004.

[5] Shweta Agrawal, Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption: New perspectives and lower bounds. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, pages 500–518, 2013.

[6] Ross J. Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems (2nd ed.)*. Wiley, 2008.

[7] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

[8] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of NP. *J. ACM*, 45(1):70–122, January 1998.

[9] Sumeet Bajaj and Radu Sion. TrustedDB: A trusted hardware-based database with privacy and data confidentiality. *IEEE Trans. Knowl. Data Eng.*, 26(3):752–765, 2014.

[10] Andrew Baumann, Marcus Peinado, and Galen Hunt. Shielding applications from an untrusted cloud with haven. In *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*, pages 267–283, Broomfield, CO, October 2014. USENIX Association.

[11] Donald Beaver. Secure multiparty protocols and zero-knowledge proof systems tolerating a faulty minority. *J. Cryptology*, 4(2):75–122, 1991.

[12] Mihir Bellare, Alexandra Boldyreva, and Adam O'Neill. Deterministic and efficiently searchable encryption. Cryptology ePrint Archive, Report 2006/186, 2006. `http://eprint.iacr.org/`.

[13] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. In *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, pages 1–15, 1996.

[14] Mihir Bellare, Viet Tung Hoang, Sriram Keelveedhi, and Phillip Rogaway. Efficient garbling from a fixed-key blockcipher. In *IEEE Symposium on Security and Privacy*, 2013.

[15] Assaf Ben-David, Noam Nisan, and Benny Pinkas. FairplayMP: A system for secure multi-party computation. In *ACM Conference on Computer and Communications Security (CCS)*, 2008.

[16] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *STOC*, 1988.

[17] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. SNARKs for C: verifying program executions succinctly and in zero knowledge. *IACR*

Cryptology ePrint Archive, 2013:507, 2013.

[18] Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. Semi-homomorphic encryption and multiparty computation. In *Eurocrypt*, 2011.

[19] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *2007 IEEE Symposium on Security and Privacy (S&P 2007), 20-23 May 2007, Oakland, California, USA*, pages 321–334, 2007.

[20] Bishwaranjan Bhattacharjee, Naoki Abe, Kenneth Goldman, Bianca Zadrozny, Vamsavardhana R. Chillakuru, Marysabel del Carpio, and Chidanand Apté. Using secure coprocessors for privacy preserving collaborative data mining and analysis. In Anastassia Ailamaki, Peter A. Boncz, and Stefan Manegold, editors, *Workshop on Data Management on New Hardware, DaMoN 2006, Chicago, Illinois, USA, June 25, 2006*, page 1. ACM, 2006.

[21] Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Omer Paneth, and Rafail Ostrovsky. Succinct non-interactive arguments via linear interactive proofs. In *Proceedings of the 10th Theory of Cryptography Conference on Theory of Cryptography*, TCC'13, pages 315–333, Berlin, Heidelberg, 2013. Springer-Verlag.

[22] Dan Bogdanov, Sven Laur, and Jan Willemson. Sharemind: A framework for fast privacy-preserving computations. In *Proceedings of the 13th European Symposium on Research in Computer Security - ESORICS'08*, 2008.

[23] Alexandra Boldyreva, Nathan Chenette, Younho Lee, and Adam O'Neill. Order-preserving symmetric encryption. In Antoine Joux, editor, *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, volume 5479 of *Lecture Notes in Computer Science*, pages 224–241. Springer, 2009.

[24] Alexandra Boldyreva, Nathan Chenette, and Adam O'Neill. Order-preserving encryption revisited: Improved security analysis and alternative solutions. In Phillip Rogaway, editor, *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, volume 6841 of *Lecture Notes in Computer Science*, pages 578–595. Springer, 2011.

[25] Alexandra Boldyreva and Paul Grubbs. The cloud encryption handbook: Encryption schemes and their relative strengths and weaknesses, 2014.

[26] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, volume 3027 of *Lecture Notes in Computer Science*, pages 506–522. Springer, 2004.

[27] Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, pages 258–275, 2005.

[28] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In *Theory of Cryptography (TCC)*, pages 325–341, 2005.

[29] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *Theory of Cryptography - 8th Theory of Cryptography Conference, TCC 2011, Providence, RI, USA, March 28-30, 2011. Proceedings*, pages 253–273, 2011.

[30] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In Salil P. Vadhan, editor, *Theory of Cryptography, 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007, Proceedings*, volume 4392 of *Lecture Notes in Computer Science*, pages 535–554. Springer, 2007.

[31] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In *Advances in Cryptology - ASIACRYPT 2013 - 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part II*, pages 280–300, 2013.

[32] Dan Boneh, Brent Waters, and Mark Zhandry. Low overhead broadcast encryption from multilinear maps. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, pages 206–223, 2014.

[33] Christoph Bösch, Pieter Hartel, Willem Jonker, and Andreas Peter. A survey of provably secure searchable encryption. *ACM Comput. Surv.*, 47(2):18:1–18:51, August 2014.

[34] Christoph Bösch, Qiang Tang, Pieter H. Hartel, and Willem Jonker. Selective document retrieval from encrypted database. In Dieter Gollmann and Felix C. Freiling, editors, *Information Security - 15th International Conference, ISC 2012, Passau, Germany, September 19-21, 2012. Proceedings*, volume 7483 of *Lecture Notes in Computer Science*, pages 224–241. Springer, 2012.

[35] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, pages 309–325, New York, NY, USA, 2012. ACM.

[36] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS*, pages 97–106, 2011.

[37] Martin Burkhart, Mario Strasser, Dilip Many, and Xenofontas Dimitropoulos. SEPIA: Privacy-preserving aggregation of multi-domain network events and statistics. In *Proceedings of the 19th USENIX Conference on Security*, 2010.

[38] Christian Cachin, Silvio Micali, and Markus Stadler. Computationally private information retrieval with polylogarithmic communication. In Jacques Stern, editor, *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, volume 1592 of *Lecture Notes in Computer Science*, pages 402–414. Springer, 1999.

[39] Angelo De Caro, Vincenzo Iovino, Abhishek Jain, Adam O'Neill, Omer Paneth, and Giuseppe Persiano. On the achievability of simulation-based security for functional encryption. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, pages 519–535, 2013.

[40] David Cash, Joseph Jaeger, Stanislaw Jarecki, Charanjit S. Jutla, Hugo Krawczyk, Marcel-Catalin Rosu, and Michael Steiner. Dynamic searchable encryption in very-large databases: Data structures and implementation. In *21st Annual Network and Distributed System Security Symposium, NDSS 2014, San Diego, California, USA, February 23-26, 2014*. The Internet Society, 2014.

[41] David Cash, Stanislaw Jarecki, Charanjit S. Jutla, Hugo Krawczyk, Marcel-Catalin Rosu, and Michael Steiner. Highly-scalable searchable symmetric encryption with support for boolean queries. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 353–373. Springer, 2013.

[42] Yan-Cheng Chang and Michael Mitzenmacher. Privacy preserving keyword searches on remote encrypted data. In John Ioannidis, Angelos D. Keromytis, and Moti Yung, editors, *Applied Cryptography and Network Security, Third International Conference, ACNS 2005, New York, NY, USA, June 7-10, 2005, Proceedings*, volume 3531 of *Lecture Notes in Computer Science*, pages 442–455, 2005.

[43] Melissa Chase and Seny Kamara. Structured encryption and controlled disclosure. In Masayuki Abe, editor, *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, volume 6477 of *Lecture Notes in Computer Science*, pages 577–594. Springer, 2010.

[44] Melissa Chase and Emily Shen. Substring-searchable symmetric encryption. In *Privacy Enhancing Technologies Symposium (PETS)*, 2015.

[45] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 11–19, 1988.

[46] Seung Geol Choi, Kyung-Wook Hwang, Jonathan Katz, Tal Malkin, and Dan Rubenstein. Secure multi-party computation of boolean circuits with applications to privacy in on-line marketplaces. In *CT-RSA*, 2012.

[47] Benny Chor, Niv Gilboa, and Moni Naor. Private information retrieval by keywords. *IACR Cryptology ePrint Archive*, 1998:3, 1998.

[48] Benny Chor, Eyal Kushilevitz, Oded Goldreich, and Madhu Sudan. Private information retrieval. *J. ACM*, 45(6):965–981, 1998.

[49] Joseph A. Cooley, Roger I. Khazan, Benjamin W. Fuller, and Galen E. Pickard. GROK: A practical system for securing group communications. In *Proceedings of The Ninth IEEE International Symposium on Networking Computing and Applications, NCA 2010, July 15-17, 2010, Cambridge, Massachusetts, USA*, pages 100–107, 2010.

[50] Graham Cormode, Michael Mitzenmacher, and Justin Thaler. Practical verified computation with streaming interactive proofs. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, pages 90–112, New York, NY, USA, 2012. ACM.

[51] Reza Curtmola, Juan A. Garay, Seny Kamara, and Rafail Ostrovsky. Searchable symmetric encryption: Improved definitions and efficient constructions. *Journal of Computer Security*, 19(5):895–934, 2011.

[52] Ivan Damgård, Martin Geisler, Mikkel Krøigaard, and Jesper Buus Nielsen. Asynchronous multiparty computation: Theory and implementation. In *Public Key Cryptography (PKC)*, 2009.

[53] Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In *CRYPTO*, 2012.

[54] Jonathan Dautrich, Emil Stefanov, and Elaine Shi. Burst ORAM: Minimizing ORAM response times for bursty access patterns. In Kevin Fu and Jaeyeon Jung, editors, *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014.*, pages 749–764. USENIX Association, 2014.

[55] Srinivas Devadas, Marten van Dijk, Christopher W. Fletcher, and Ling Ren. Onion ORAM: A constant bandwidth and constant client storage ORAM (without FHE or SWHE). *IACR Cryptology ePrint Archive*, 2015:5, 2015.

[56] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[57] Yevgeniy Dodis and Nelly Fazio. Public key broadcast encryption for stateless receivers. In *Security and Privacy in Digital Rights Management, ACM CCS-9 Workshop, DRM 2002, Washington, DC, USA, November 18, 2002, Revised Papers*, pages 61–80, 2002.

[58] Yarkin Doröz, Yin Hu, and Berk Sunar. Homomorphic AES evaluation using NTRU. *IACR Cryptology ePrint Archive*, 2014:39, 2014.

[59] Amos Fiat and Moni Naor. Broadcast encryption. In *Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22-26, 1993, Proceedings*, pages 480–491, 1993.

[60] Ben A. Fisch, Binh Vo, Fernando Krell, Abishek Kumarasubramanian, Vladimir Kolesnikov, Tal Malkin, and Steven M. Bellovin. Malicious-client security in Blind

Seer: A scalable private DBMS. In *IEEE Symposium on Security and Privacy*, pages 395–410. IEEE Computer Society, 2015.
[61] Internet Engineering Task Force. Request for comments 4301. Security architecture for the internet protocol. https://tools.ietf.org/html/rfc4301. Accessed: 2015-07-09.
[62] Internet Engineering Task Force. Request for comments 5246. The Transport Layer Security (TLS) protocol version 1.2. https://tools.ietf.org/html/rfc5246. Accessed: 2015-07-09.
[63] The Apache Software Foundation. Accumulo. https://accumulo.apache.org/. Accessed: 2015-07-09.
[64] Michael J. Freedman, Yuval Ishai, Benny Pinkas, and Omer Reingold. Keyword search and oblivious pseudorandom functions. In Joe Kilian, editor, *Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005, Proceedings*, volume 3378 of *Lecture Notes in Computer Science*, pages 303–324. Springer, 2005.
[65] Vijay Gadepally, Braden Hancock, Benjamin Kaiser, Jeremy Kepner, Peter Michaleas, Mayank Varia, and Arkady Yerukhimovich. Computing on masked data to improve the security of big data. *CoRR*, abs/1504.01287, 2015.
[66] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.
[67] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 40–49, 2013.
[68] Sanjam Garg, Craig Gentry, Shai Halevi, Amit Sahai, and Brent Waters. Attribute-based encryption for circuits from multilinear maps. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, pages 479–499, 2013.
[69] Sanjam Garg, Craig Gentry, Shai Halevi, and Mark Zhandry. Fully secure functional encryption without obfuscation. *IACR Cryptology ePrint Archive*, 2014:666, 2014.
[70] William I. Gasarch. A survey on private information retrieval (column: Computational complexity). *Bulletin of the EATCS*, 82:72–107, 2004.
[71] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. `crypto.stanford.edu/craig`.
[72] Craig Gentry and Shai Halevi. Implementing Gentry's fully-homomorphic encryption scheme. In *EUROCRYPT 2011*, volume 6632, pages 129–148. Springer, 2011.
[73] Craig Gentry, Shai Halevi, and Nigel Smart. Fully homomorphic encryption with polylog overhead. In *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 465–482. Springer, 2012. Full version at `http://eprint.iacr.org/2011/566`.
[74] Craig Gentry, Shai Halevi, and Nigel Smart. Homomorphic evaluation of the AES circuit. In *CRYPTO 2012*, volume 7417 of *LNCS*, pages 850–867. Springer, 2012. Full version at `http://eprint.iacr.org/2012/099`.
[75] Craig Gentry, Shai Halevi, and Nigel P. Smart. Better bootstrapping in fully homomorphic encryption. In *Public Key Cryptography - PKC 2012*, volume 7293 of *LNCS*, pages 1–16. Springer, 2012.
[76] Yael Gertner, Yuval Ishai, Eyal Kushilevitz, and Tal Malkin. Protecting data privacy in private information retrieval schemes. *J. Comput. Syst. Sci.*, 60(3):592–629, 2000.
[77] Eu-Jin Goh. Secure indexes. *IACR Cryptology ePrint Archive*, 2003:216, 2003.
[78] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, 1987.
[79] Oded Goldreich and Rafail Ostrovsky. Software protection and simulation on oblivious rams. *J. ACM*, 43(3):431–473, 1996.
[80] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and

Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 555–564, 2013.

[81] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: Interactive proofs for muggles. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, pages 113–122, New York, NY, USA, 2008. ACM.

[82] Shafi Goldwasser and Silvio Micali. Probabilistic encryption & how to play mental poker keeping secret all partial information. In *Proceedings of the fourteenth annual ACM symposium on Theory of computing*, STOC '82, pages 365–377, New York, NY, USA, 1982. ACM.

[83] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988.

[84] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, pages 162–179, 2012.

[85] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 545–554, 2013.

[86] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, Ioctober 30 - November 3, 2006*, pages 89–98, 2006.

[87] Shai Halevi and Victor Shoup. HElib. `https://github.com/shaih/HElib`. Accessed: 2014-09-23.

[88] Shai Halevi and Victor Shoup. Algorithms in HElib. In *CRYPTO 2014*, volume 8616, pages 554–571. Springer, 2014.

[89] Wilko Henecka, Stefan Køgl, Ahmad-Reza Sadeghi, Thomas Schneider, and Immo Wehrenberg. TASTY: Tool for automating secure two-party computations. In *ACM Conference on Computer and Communications Security (CCS)*, 2010.

[90] Andreas Holzer, Martin Franz, Stefan Katzenbeisser, and Helmut Veith. Secure two-party computations in ANSI C. In *ACM Conference on Computer and Communications Security (CCS)*, 2012.

[91] Yan Huang, David Evans, Jonathan Katz, and Lior Malka. Faster secure two-party computation using garbled circuits. In *USENIX Security Symposium*, 2011.

[92] IARPA. Broad agency announcement IARPA-BAA-11-01: Security and privacy assurance research (SPAR) program. `https://www.fbo.gov/notices/c55e38dbde30cb668f687897d8f01e69`, February 2011.

[93] Alexander Iliev and Sean W. Smith. Private information storage with logarithm-space secure hardware. In Yves Deswarte, Frédéric Cuppens, Sushil Jajodia, and Lingyu Wang, editors, *Information Security Management, Education and Privacy, IFIP 18th World Computer Congress, TC11 19th International Information Security Workshops, 22-27 August 2004, Toulouse, France*, volume 148 of *IFIP*, pages 199–214. Kluwer, 2004.

[94] Yuval Ishai, Eyal Kushilevitz, and Rafail Ostrovsky. Efficient arguments without short PCPs. In *Proceedings of the Twenty-Second Annual IEEE Conference on Computational Complexity*, CCC '07, pages 278–291, Washington, DC, USA, 2007. IEEE Computer Society.

[95] Stanislaw Jarecki, Charanjit S. Jutla, Hugo Krawczyk, Marcel-Catalin Rosu, and Michael Steiner. Outsourced symmetric private information retrieval. In Sadeghi et al. [150], pages 875–888.

[96] David Kahn. *The Codebreakers: The Comprehensive History of Secret Communication from Ancient Times to the Internet*. Scribner, rev sub edition, December 1996.

[97] Seny Kamara and Charalampos Papamanthou. Parallel and dynamic searchable symmetric encryption. In *Financial Cryptography and Data Security - 17th International Conference, FC 2013, Okinawa, Japan, April 1-5, 2013, Revised Selected Papers*, pages 258–274, 2013.

[98] Seny Kamara, Charalampos Papamanthou, and Tom Roeder. Dynamic searchable symmetric encryption. In *the ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012*, pages 965–976, 2012.

[99] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. Chapman and Hall/CRC Press, 2007.

[100] Jonathan Katz and Arkady Yerukhimovich. On black-box constructions of predicate encryption from trapdoor permutations. In *Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings*, pages 197–213, 2009.

[101] Roger I. Khazan and Dan Utin. Lincoln Open Cryptographic Key Management Architecture. https://www.ll.mit.edu/publications/technotes/TechNote_LOCKMA.pdf. Accessed: 2015-06-26.

[102] Joud Khoury, Gregory Lauer, Partha Pratim Pal, Bishal Thapa, and Joseph P. Loyall. Efficient private publish-subscribe systems. In *17th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing, ISORC 2014, Reno, NV, USA, June 10-12, 2014*, pages 64–71, 2014.

[103] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *Proceedings of the Twenty-fourth Annual ACM Symposium on Theory of Computing*, STOC '92, pages 723–732, New York, NY, USA, 1992. ACM.

[104] Vladimir Kolesnikov, Payman Mohassel, and Mike Rosulek. FleXOR: Flexible garbling for XOR gates that beats free-XOR. In *CRYPTO*, 2014.

[105] Vladimir Kolesnikov and Thomas Schneider. Improved garbled circuit: Free XOR gates and applications. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, 2008.

[106] Ben Kreuter, Benjamin Mood, Abhi Shelat, and Kevin Butler. PCF: A portable circuit format for scalable two-party secure computation. In *USENIX Security Symposium*, 2013.

[107] Benjamin Kreuter, Abhi Shelat, and Chih-Hao Shen. Billion-gate secure computation with malicious adversaries. In *USENIX Security Symposium*, 2012.

[108] Eyal Kushilevitz and Rafail Ostrovsky. Replication is NOT needed: SINGLE database, computationally-private information retrieval. In *38th Annual Symposium on Foundations of Computer Science, FOCS '97, Miami Beach, Florida, USA, October 19-22, 1997*, pages 364–373. IEEE Computer Society, 1997.

[109] Eyal Kushilevitz and Rafail Ostrovsky. One-way trapdoor permutations are sufficient for non-trivial single-server private information retrieval. In Bart Preneel, editor, *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding*, volume 1807 of *Lecture Notes in Computer Science*, pages 104–121. Springer, 2000.

[110] RSA Laboratories. Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1. https://tools.ietf.org/html/rfc3447. Accessed: 2015-06-26.

[111] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, July 1982.

[112] Thomas A. Limoncelli, Strata R. Chalup, and Christina J. Hogan. *The Practice of*

Cloud System Administration: Designing and Operating Large Distributed Systems (Volume 2). Addison-Wesley Professional, 1 edition, September 2014.

[113] Yehuda Lindell and Benny Pinkas. Secure two-party computation via cut-and-choose oblivious transfer. In *Theory of Cryptography Conference (TCC)*, 2011.

[114] Yehuda Lindell, Benny Pinkas, and Nigel Smart. Implementing two-party computation efficiently with security against malicious adversaries. In *Security and Cryptography for Networks (SCN)*, 2008.

[115] Chang Liu, Yan Huang, Elaine Shi, Jonathan Katz, and Michael W. Hicks. Automating efficient RAM-model secure computation. In *2014 IEEE Symposium on Security and Privacy, SP 2014*, pages 623–638, 2014.

[116] Chang Liu, Xiao Shaun Wang, Kartik Nayak, Yan Huang, and Elaine Shi. ObliVM: A programming framework for secure computation. In *IEEE Symposium on Security and Privacy*, 2015.

[117] Ben Lynn. PBC: The pairing-based crypto library. http://crypto.stanford.edu/pbc/. Accessed: 2015-06-26.

[118] Dahlia Malkhi, Noam Nisan, Benny Pinkas, and Yaron Sella. Fairplay – A secure two-party computation system. In *USENIX Security Symposium*, 2004.

[119] Mandiant. http://intelreport.mandiant.com/Mandiant_APT1_Report.pdf, Feb 2013.

[120] Charalampos Mavroforakis, Nathan Chenette, Adam O'Neill, George Kollios, and Ran Canetti. Modular order-preserving encryption, revisited. In Timos Sellis, Susan B. Davidson, and Zachary G. Ives, editors, *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015*, pages 763–777. ACM, 2015.

[121] MIT Lincoln Laboratory. HEtest. `https://www.ll.mit.edu/mission/cybersec/softwaretools/hetest/hetest.html`, February 2011.

[122] Tarik Moataz, Travis Mayberry, and Erik-Oliver Blass. Resizable tree-based oblivious RAM. *IACR Cryptology ePrint Archive*, 2014:732, 2014.

[123] Tarik Moataz, Travis Mayberry, and Erik-Oliver Blass. Constant communication oblivious RAM. *IACR Cryptology ePrint Archive*, 2015:570, 2015.

[124] Marti Motoyama, Damon McCoy, Kirill Levchenko, Stefan Savage, and Geoffrey M. Voelker. An analysis of underground forums. In *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, IMC '11, pages 71–80, New York, NY, USA, 2011. ACM.

[125] Dalit Naor, Moni Naor, and Jeffery Lotspiech. Revocation and tracing schemes for stateless receivers. In *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, pages 41–62, 2001.

[126] Moni Naor, Benny Pinkas, and Reuban Sumner. Privacy preserving auctions and mechanism design. In *ACM Conference on Electronic Commerce (EC)*, 1999.

[127] Muhammad Naveed, Shashank Agrawal, Manoj Prabhakaran, XiaoFeng Wang, Erman Ayday, Jean-Pierre Hubaux, and Carl A. Gunter. Controlled functional encryption. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*, pages 1280–1291, 2014.

[128] OASIS. Key management interoperability protocol specification version 1.2. http://docs.oasis-open.org/kmip/spec/v1.2/cos01/kmip-spec-v1.2-cos01.html. Accessed: 2015-06-26.

[129] National Institute of Standards and Technology. Cryptographic toolkit. http://csrc.nist.gov/groups/ST/toolkit/index.html. Accessed: 2015-06-26.

[130] National Institute of Standards and Technology. Federal Information Processing Standards Publication 197: Announcing the Advanced Encryption Standard (AES). http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf. Accessed: 2015-07-09.

[131] Adam O'Neill. Definitional issues in functional encryption. *IACR Cryptology ePrint*

Archive, 2010:556, 2010.

[132] Oracle. Oracle label security with oracle database 12c. http://www.oracle.com/technetwork/database/options/label-security/label-security-wp-12c-1896140.pdf?ssSourceSiteId=ocomen, June 2013.

[133] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*, pages 223–238. Springer, 1999.

[134] Vasilis Pappas, Fernando Krell, Binh Vo, Vladimir Kolesnikov, Tal Malkin, Seung Geol Choi, Wesley George, Angelos D. Keromytis, and Steve Bellovin. Blind Seer: A scalable private DBMS. In *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*, pages 359–374. IEEE Computer Society, 2014.

[135] Hyun-A Park, Bum Han Kim, Dong Hoon Lee, Yon Dohn Chung, and Justin Zhan. Secure similarity search. In *2007 IEEE International Conference on Granular Computing, GrC 2007, San Jose, California, USA, 2-4 November 2007*, page 598. IEEE, 2007.

[136] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *Proceedings of the 2013 IEEE Symposium on Security and Privacy*, SP '13, pages 238–252, Washington, DC, USA, 2013. IEEE Computer Society.

[137] Benny Pinkas, Thomas Schneider, Nigel P. Smart, and Stephen C. Williams. Secure two-party computation is practical. In *ASIACRYPT*, 2009.

[138] Matthew Pirretti, Patrick Traynor, Patrick McDaniel, and Brent Waters. Secure attribute-based systems. In *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, Ioctober 30 - November 3, 2006*, pages 99–112, 2006.

[139] Raluca A. Popa, Frank H. Li, and Nickolai Zeldovich. An ideal-security protocol for order-preserving encoding. In *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013* [1], pages 463–477.

[140] Raluca Ada Popa, Catherine M. S. Redfield, Nickolai Zeldovich, and Hari Balakrishnan. CryptDB: Protecting confidentiality with encrypted query processing. In *ACM Symposium on Operating Systems Principles (SOSP 2011)*, 2011.

[141] Thomas Pöppelmann, Michael Naehrig, Andrew Putnam, and Adrian Macias. Accelerating homomorphic evaluation on reconfigurable hardware. Cryptology ePrint Archive, Report 2015/631, 2015. `http://eprint.iacr.org/`.

[142] Michael O. Rabin. Digitalized signatures and public-key functions as intractable as factorization. MIT Laboratory for Computer Science. `http://publications.csail.mit.edu/lcs/pubs/pdf/MIT-LCS-TR-212.pdf`, January 1979.

[143] Michael O. Rabin. How to exchange secrets with oblivious transfer. *IACR Cryptology ePrint Archive*, 2005:187, 2005.

[144] Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *ACM Symposium on Theory of Computing (STOC)*, 1989.

[145] Aseem Rastogi, Matthew A. Hammer, and Michael Hicks. Wysteria: A programming language for generic, mixed-mode multiparty computations. In *IEEE Symposium on Security and Privacy*, 2014.

[146] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, STOC '05, pages 84–93, New York, NY, USA, 2005. ACM.

[147] Ling Ren, Christopher W. Fletcher, Albert Kwon, Emil Stefanov, Elaine Shi, Marten van Dijk, and Srinivas Devadas. Ring ORAM: Closing the gap between small and large client storage oblivious RAM. *IACR Cryptology ePrint Archive*, 2014:997, 2014.

[148] Ronald L. Rivest, Len Adleman, and Michael L. Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation*, pages 169–180, 1978.

[149] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining

digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.
[150] Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors. *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*. ACM, 2013.
[151] Amit Sahai and Hakan Seyalioglu. Worry-free encryption: Functional encryption with public keys. In *Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, October 4-8, 2010*, pages 463–472, 2010.
[152] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, pages 457–473, 2005.
[153] Felix Schuster, Manuel Costa, Cédric Fournet, Christos Gkantsidis, Marcus Peinado, Gloria Mainar-Ruiz, and Mark Russinovich. VC3: trustworthy data analytics in the cloud using SGX. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, pages 38–54, 2015.
[154] Adi Shamir. How to share a secret. In *Communications of the ACM*, 1979.
[155] Abhi Shelat and Chih-Hao Shen. Fast two-party secure computation with minimal assumptions. In Sadeghi et al. [150], pages 523–534.
[156] Emily Shen, Elaine Shi, and Brent Waters. Predicate privacy in encryption systems. In Omer Reingold, editor, *Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009, San Francisco, CA, USA, March 15-17, 2009. Proceedings*, volume 5444 of *Lecture Notes in Computer Science*, pages 457–473. Springer, 2009.
[157] Nigel P. Smart and F. Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In *Public Key Cryptography – PKC 2010*, volume 6056 of *LNCS*, pages 420–443, 2010.
[158] Nigel P. Smart and F. Vercauteren. Fully homomorphic SIMD operations. In *Designs, Codes and Cryptography*. Springer, 2011.
[159] Sean W. Smith and David Safford. Practical server privacy with secure coprocessors. *IBM Systems Journal*, 40(3):683–695, 2001.
[160] Dawn Xiaodong Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In *2000 IEEE Symposium on Security and Privacy, Berkeley, California, USA, May 14-17, 2000*, pages 44–55. IEEE Computer Society, 2000.
[161] Ebrahim M. Songhori, Siam U. Hussain, Ahmad-Reza Sadeghi, Thomas Schneider, and Farinaz Koushanfar. TinyGarble: Highly compressed and scalable sequential garbled circuits. In *IEEE Symposium on Security and Privacy*, 2015.
[162] Emil Stefanov and Elaine Shi. ObliviStore: High performance oblivious cloud storage. In *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013* [1], pages 253–267.
[163] Emil Stefanov, Marten van Dijk, Elaine Shi, Christopher W. Fletcher, Ling Ren, Xiangyao Yu, and Srinivas Devadas. Path ORAM: An extremely simple oblivious RAM protocol. In Sadeghi et al. [150], pages 299–310.
[164] New York Times. 9 recent cyberattacks against big businesses. http://www.nytimes.com/interactive/2015/02/05/technology/recent-cyberattacks.html, February 5, 2015. Accessed: 2015-07-09.
[165] New York Times. Hacking linked to China exposes millions of U.S. workers. http://www.nytimes.com/2015/06/05/us/breach-in-a-federal-computer-system-exposes-personnel-data.html, June 4, 2015. Accessed: 2015-07-09.
[166] Stephen Tu, M. Frans Kaashoek, Samuel Madden, and Nickolai Zeldovich. Processing analytical queries over encrypted data. *Proc. VLDB Endow.*, 6(5):289–300, March 2013.
[167] Peter van Liesdonk, Saeed Sedghi, Jeroen Doumen, Pieter H. Hartel, and Willem

Jonker. Computationally efficient searchable symmetric encryption. In *Secure Data Management, 7th VLDB Workshop, SDM 2010, Singapore, September 17, 2010. Proceedings*, pages 87–100, 2010.

[168] Mayank Varia, Benjamin Price, Nicholas Hwang, Ariel Hamlin, Jonathan Herzog, Jill Poland, Michael Reschly, Sophia Yakoubov, and Robert K. Cunningham. Automated assessment of secure search systems. *Operating Systems Review*, 49(1):22–30, 2015.

[169] Michael Walfish and Andrew J. Blumberg. Verifying computations without reexecuting them. *Commun. ACM*, 58(2):74–84, 2015.

[170] Guojun Wang, Qin Liu, and Jie Wu. Hierarchical attribute-based encryption for fine-grained access control in cloud storage services. In *Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, October 4-8, 2010*, pages 735–737, 2010.

[171] Andrew C. Yao. Protocols for secure computations (extended abstract). In *Foundations of Computer Science (FOCS)*, 1982.

[172] Samee Zahur, Mike Rosulek, and David Evans. Two halves make a whole: Reducing data transfer in garbled circuits using half gates. In *Eurocrypt*, 2015.

[173] Yihua Zhang, Aaron Steele, and Marina Blanton. PICCO: A general-purpose compiler for private distributed computation. In *ACM Conference on Computer and Communications Security (CCS)*, 2013.