Cryptographic Cloud Storage

Seny Kamara
Microsoft Research
senyk@microsoft.com

Kristin Lauter
Microsoft Research
klauter@microsoft.com

Abstract

We consider the problem of building a secure cloud storage service on top of a public cloud infrastructure where the service provider is not completely trusted by the customer. We describe, at a high level, several architectures that combine recent and non-standard cryptographic primitives in order to achieve our goal. We survey the benefits such an architecture would provide to both customers and service providers and give an overview of recent advances in cryptography motivated specifically by cloud storage.

1 Introduction

Advances in networking technology and an increase in the need for computing resources have prompted many organizations to outsource their storage and computing needs. This new economic and computing model is commonly referred to as cloud computing and includes various types of services such as: infrastructure as a service (IaaS), where a customer makes use of a service provider's computing, storage or networking infrastructure; platform as a service (PaaS), where a customer leverages the provider's resources to run custom applications; and finally software as a service (SaaS), where customers use software that is run on the providers infrastructure.

Cloud infrastructures can be roughly categorized as either private or public. In a private cloud, the infrastructure is managed and owned by the customer and located on-premise (i.e., in the customers region of control). In particular, this means that access to customer data is under its control and is only granted to parties it trusts. In a public cloud the infrastructure is owned and managed by a cloud service provider and is located off-premise (i.e., in the service provider's region of control). This means that customer data is outside its control and could potentially be granted to untrusted parties.

Storage services based on public clouds such as Microsoft's Azure storage service and Amazon's S3 provide customers with scalable and dynamic storage. By moving their data to the cloud customers can avoid the costs of building and maintaining a private storage infrastructure, opting instead to pay a service provider as a function of its needs. For most customers, this provides several benefits including availability (i.e., being able to access data from anywhere) and reliability (i.e., not having to worry about backups) at a relatively low cost.

While the benefits of using a public cloud infrastructure are clear, it introduces significant security and privacy risks. In fact, it seems that the biggest hurdle to the adoption of cloud storage (and cloud computing in general) is concern over the confidentiality and integrity of data. While, so far, consumers have been willing to trade privacy for the convenience of software services (e.g., for web-based email, calendars, pictures etc), this is not the case for enterprises and government

organizations. This reluctance can be attributed to several factors that range from a desire to protect mission-critical data to regulatory obligations to preserve the confidentiality and integrity of data. The latter can occur when the customer is responsible for keeping personally identifiable information (PII), or medical and financial records. So while cloud storage has enormous promise, unless the issues of confidentiality and integrity are addressed many potential customers will be reluctant to make the move.

To address the concerns outlined above and increase the adoption of cloud storage, we argue for designing a *virtual private storage service* based on recently developed cryptographic techniques. Such a service should aim to achieve the best of both worlds by providing the security of a private cloud and the functionality and cost savings of a public cloud. More precisely, such a service should provide (at least):

- confidentiality: the cloud storage provider does not learn any information about customer data
- integrity: any unauthorized modification of customer data by the cloud storage provider can be detected by the customer

while retaining the main benefits of a public storage service:

- availability: customer data is accessible from any machine and at all times
- reliability: customer data is reliably backed up
- efficient retrieval: data retrieval times are comparable to a public cloud storage service
- data sharing: customers can share their data with trusted parties.

An important aspect of a cryptographic storage service is that the security properties described above are achieved based on strong cryptographic guarantees as opposed to legal, physical and access control mechanisms. We believe this has several important benefits which we discuss further in Section 3.

This article is organized as follows. In Section 2 we describe, at a high level, a possible architecture for a cryptographic storage service. We consider both consumer and enterprise scenarios. We stress that this design is not intended to be a formal specification (indeed many important business and engineering questions would need to be addressed) but is only meant to serve as an *illustration* of how some of the new and non-standard cryptographic techniques that have been developed recently could be combined to achieve our goals. In Section 3 we give an overview of the benefits of a cryptographic storage service, e.g., reducing the legal exposure of both customers and cloud providers, and achieving regulatory compliance. In Section 4 we describe in more detail the relevant cryptographic techniques, including searchable encryption, proofs of storage and attribute-based encryption. Finally, in Section 5, we mention some cloud services that could be built on top of a cryptographic storage service such as secure back-ups, archival, health record systems, secure data exchange and e-discovery.

2 Architecture of a Cryptographic Storage Service

We now describe, at a high level, a possible architecture for a cryptographic storage service. At its core, the architecture consists of three components: a *data processor* (DP), that processes data

before it is sent to the cloud; a *data verifier* (DV), that checks whether the data in the cloud has been tampered with; and a *token generator* (TG), that generates tokens that enable the cloud storage provider to retrieve segments of customer data; and a *credential generator* that implements an access control policy by issuing credentials to the various parties in the system (these credentials will enable the parties to decrypt encrypted files according to the policy). We describe designs for both consumer and enterprise scenarios.

2.1 A Consumer Architecture

Consider three parties: a user Alice that stores her data in the cloud; a user Bob with whom Alice wants to share data; and a cloud storage provider that stores Alice's data. To use the service, Alice and Bob begin by downloading a client application that consists of a data processor, a data verifier and a token generator. Upon its first execution, Alice's application generates a cryptographic key. We will refer to this key as a master key and assume it is stored locally on Alice's system and that it is kept secret from the cloud storage provider.

Whenever Alice wishes to upload data to the cloud, the data processor is invoked. It attaches some metadata (e.g., current time, size, keywords etc) and encrypts and encodes the data and metadata with a variety of cryptographic primitives (which we describe in more detail in Section 4). Whenever Alice wants to verify the integrity of her data, the data verifier is invoked. The latter uses Alice's master key to interact with the cloud storage provider and ascertain the integrity of the data. When Alice wants to retrieve data (e.g., all files tagged with keyword "urgent") the token generator is invoked to create a token. The token is sent to the cloud storage provider who uses it to retrieve the appropriate (encrypted) files which it returns to Alice. Alice then uses the decryption key to decrypt the files. Data sharing between Alice and Bob proceeds in a similar fashion. Whenever she wishes to share data with Bob, the application invokes the token generator to create an appropriate token, and the credential generator to generate a credential for Bob. Both the token and credential are sent to Bob who, in turn, sends the token to the provider. The latter uses the token to retrieve and return the appropriate encrypted documents which Bob decrypts using his credential.

This process is illustrated in Figure 1. We note that in order to achieve the security properties we seek, it is important that the client-side application and, in particular, the core components be either open-source or implemented or verified by someone other than the cloud service provider.

2.2 An Enterprise Architecture

In the enterprise scenario we consider an enterprise MegaCorp that stores its data in the cloud; a business partner PartnerCorp with whom MegaCorp wants to share data; and a cloud storage provider that stores MegaCorp's data.

To use the service, MegaCorp deploys dedicated machines within its network. Depending on the particular scenario, these dedicated machines will run various core components. Since these components make use of a master secret key, it is important that they be adequately protected and, in particular, that the master key be kept secret from the cloud storage provider and PartnerCorp. If this is too costly in terms of resources or expertise, management of the dedicated machines (or specific components) can alternatively be outsourced to a trusted entity.

In the case of a medium-sized enterprise with enough resources and expertise, the dedicated machines include a data processor, a data verifier, a token generator and a credential generator.

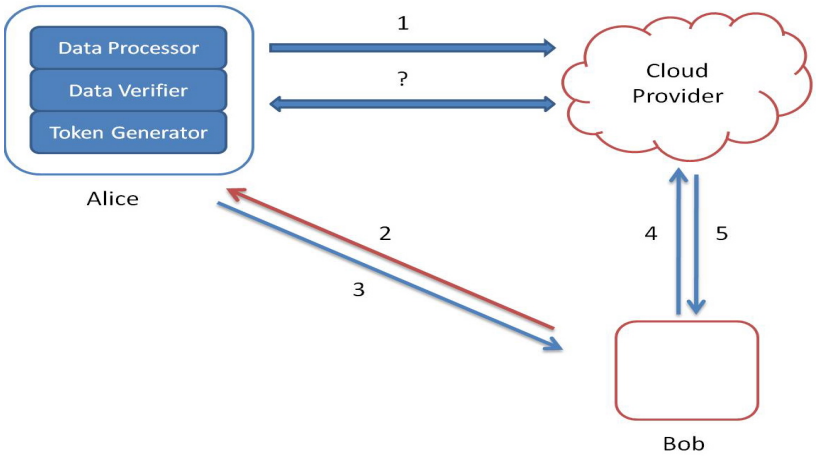

Figure 1: (1) Alice's data processor prepares the data before sending it to the cloud; (2) Bob asks Alice for permission to search for a keyword; (3) Alice's token and credential generators send a token for the keyword and a credential back to Bob; (4) Bob sends the token to the cloud; (5) the cloud uses the token to find the appropriate encrypted documents and returns them to Bob. (?) At any point in time, Alice's data verifier can verify the integrity of the data.

To begin, each MegaCorp and PartnerCorp employee receives a credential from the credential generator. These credentials will reflect some relevant information about the employees such as their organization or team or role. Whenever a MegaCorp employee generates data that needs to be stored in the cloud, it sends the data together with an associated decryption policy to the dedicated machine for processing. The decryption policy specifies the type of credentials necessary to decrypt the data (e.g., only members of a particular team). To retrieve data from the cloud (e.g., all files generated by a particular employee), an employee requests an appropriate token from the dedicated machine. The employee then sends the token to the cloud provider who uses it to find and return the appropriate encrypted files which the employee decrypts using his credentials. Whenever MegaCorp wants to verify the integrity of the data, the dedicated machine's data verifier is invoked. The latter uses the master secret key to interact with the storage provider and ascertain the integrity of the data.

Now consider the case where a PartnerCorp employee needs access to MegaCorp's data. The employee authenticates itself to MegaCorp's dedicated machine and sends it a keyword. The latter verifies that the particular search is allowed for this PartnerCorp employee. If so, the dedicated machine returns an appropriate token which the employee uses to recover the appropriate (encrypted) files from the service provider. It then uses its credentials to decrypt the file. This process is illustrated in Figure 2. Similarly to the consumer architecture, it is imperative that all components be either open-source or implemented by someone other than the cloud service provider.

In the case that MegaCorp is a very large organization and that the prospect of running and maintaining enough dedicated machines to process all employee data is infeasible, consider the following slight variation of the architecture described above. More precisely, in this case the dedicated machines only run data verifiers, token generators and credential generators while the data processing is distributed to each employee. This is illustrated in Figure 3. Note that in this scenario the data processors do not include the master secret key so the confidentiality of the data

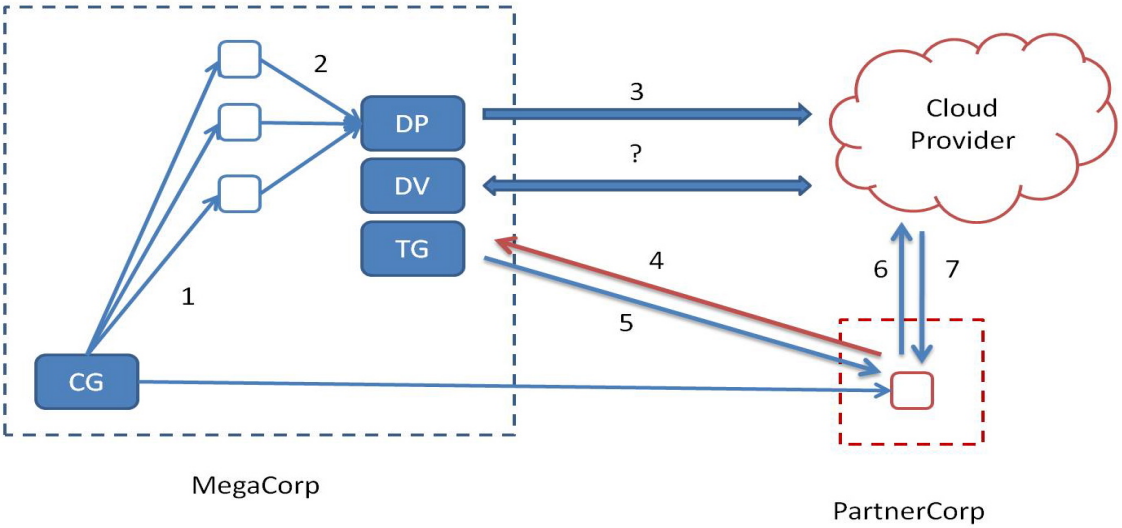

Figure 2: (1) Each MegaCorp and PartnerCorp employee receives a credential; (2) MegaCorp employees send their data to the dedicated machine; (3) the latter processes the data using the data processor before sending it to the cloud; (4) the PartnerCorp employee sends a keyword to MegaCorp's dedicated machine ; (5) the dedicated machine returns a token; (6) the PartnerCorp employee sends the token to the cloud; (7) the cloud uses the token to find the appropriate encrypted documents and returns them to the employee. (?) At any point in time, MegaCorp's data verifier can verify the integrity of MegaCorp's data.

is not affected. The data processors, however, do include some keying material which, if revealed to the service provider, could enable it to compromise the confidentiality of the tokens it receives (i.e,. it could learn which keywords are being searched for).

3 Benefits of a Cryptographic Storage Service

The core properties of a cryptographic storage service are that (1) control of the data is maintained by the customer and (2) the security properties are derived from cryptography, as opposed to legal mechanisms, physical security or access control. Therefore, such a service provides several compelling advantages over other storage services based on public cloud infrastructures. In this section, we recall some of the main concerns with cloud computing as outlined in the Cloud Security Alliances recent report [7] and highlight how these concerns can be mitigated by such an architecture.

Regulatory compliance. Most countries have laws in place that make organizations responsible for the protection of the data that is entrusted to them. This is particularly so for the case of personally identifiable information, medical records and financial records. And since organizations are often held responsible for the actions of their contractors, the use of a public cloud storage service can involve significant legal risks. In a cryptographic storage service, the data is encrypted on-premise by the data processor(s). This way, customers can be assured that the confidentiality of their data is preserved irrespective of the actions of the cloud storage provider. This greatly

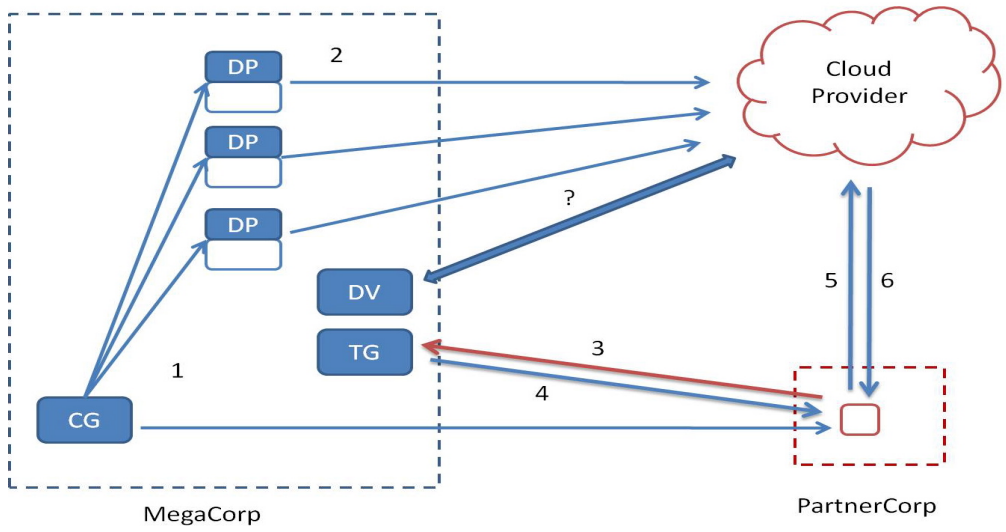

Figure 3: (1) Each MegaCorp and PartnerCorp employee receives a credential; (2) MegaCorp employees process their data using their own data processors and send them to the cloud; (3) the PartnerCorp employee sends a keyword to MegaCorp's dedicated machine; (4) the latter returns a token; (5) the employee sends the token to the cloud; (6) the cloud uses the token to find the appropriate encrypted documents and returns them to the employee. (?) At any point in time, MegaCorp's data verifier can check the integrity of MegaCorp's data.

reduces any legal exposure for both the customer and the provider.

Geographic restrictions. Data that is stored in certain legal jurisdictions may be subject to regulations even if it was not collected there. Because it can be difficult to ascertain exactly where one's data is being stored once it is sent to the cloud (i.e., many service providers have data centers deployed throughout the world) some customers may be reluctant to use a public cloud for fear of increasing their legal exposure. In a cryptographic storage service data is only stored in encrypted form so any law that pertains to the stored data has little to no effect on the customer. This reduces legal exposure for the customer and allows the cloud storage provider to make optimal use of its storage infrastructure, thereby reducing costs.

Subpoenas. If an organization becomes the subject of an investigation, law enforcement agencies may request access to its data. If the data is stored in a public cloud, the request may be made to the cloud provider and the latter could even be prevented from notifying the customer. This can have severe consequences for customers. First, it preempts the customer from challenging the request. Second, it can lead to law enforcement having access to data from clients that are not under investigation (see, e.g., [34]). Such a scenario can occur due to the fact that service providers often store multiple customer's data on the same disks. In a cryptographic storage service, since data is stored in encrypted form and since the customer retains possession of all the keys, any request for the (unencrypted) data must be made directly to the customer.

Security breaches. Even if a cloud storage provider implements strong security practices there is always the possibility of a security breach. If this occurs the customer may be legally responsible. In a cryptographic storage service data is encrypted and data integrity can be verified at any time. Therefore, a security breach poses little to no risk for the customer.

Electronic discovery. Digital information plays an important role in legal proceedings and often organizations are required to preserve and produce records for litigation. Organizations with high levels of litigation may need to keep a copy of large amounts of data on-premise in order to assure its integrity. This can obviously negate the benefits of using a cloud storage service. Since, with a cryptographic storage service, a customer can verify the integrity of its data at any point in time (e.g., every hour) a provider has every incentive to preserve the data's integrity.

Data retention and destruction. In many cases a customer may be responsible for the retention and destruction of the data it has collected. If this data is stored in the cloud, however, it can be difficult for a customer to ascertain the integrity of the data or to verify whether it was properly discarded. A cryptographic storage service alleviates these concerns since data integrity can be verified and since the information necessary to decrypt data (i.e., the master key) is kept on-premise. Secure data erasure can be effectively achieved by just erasing the master key.

4 Implementing the Core Components

The core components of a cryptographic storage service can be implemented using a variety of techniques, some of which were developed specifically for cloud storage. When preparing data for storage in the cloud, the data processor begins by indexing it and encrypting it with a symmetric encryption scheme (e.g., AES) under a unique key. It then encrypts the index using a *searchable encryption* scheme and encrypts the unique key with an *attribute-based encryption* scheme under an appropriate policy. Finally, it encodes the encrypted data and index in such a way that the data verifier can later verify their integrity using a *proof of storage*.

In the following we provide high level descriptions of these new cryptographic primitives. While traditional techniques like encryption and digital signatures could be used to implement the core components, they would do so at considerable cost in communication and computation. To see why, consider the example of an organization that encrypts and signs its data before storing it in the cloud. While this clearly preserves confidentiality and integrity it has the following limitations. To enable searching over the data, the customer has to either store an index locally, or download all the (encrypted) data, decrypt it and search locally. The first approach obviously negates the benefits of cloud storage (since indexes can grow large) while the second has high communication complexity. Also, to verify the integrity of the data, the organization would have to retrieve all the data first in order to verify the signatures. If the data is large, this verification procedure is obviously undesirable. Various solutions based on (keyed) hash functions could also be used, but all such approaches only allow a fixed number of verifications.

4.1 Searchable Encryption

At a high level, a searchable encryption scheme provides a way to "encrypt" a search index so that its contents are hidden except to a party that is given appropriate tokens. More precisely, consider

a search index generated over a collection of files (this could be a full-text index or just a keyword index). Using a searchable encryption scheme, the index is encrypted in such a way that (1) given a token for a keyword one can retrieve pointers to the encrypted files that contain the keyword; and (2) without a token the contents of the index are hidden. In addition, the tokens can only be generated with knowledge of a secret key and the retrieval procedure reveals nothing about the files or the keywords except that the files contain a keyword in common.

This last point is worth discussing further as it is crucial to understanding the security guarantee provided by searchable encryption. Notice that over time (i.e., after many searches) knowing that a certain subset of documents contain a word in common *may* leak some useful information. This is because the server could make some assumptions about the client's search pattern and use this information to make a guess about the keywords being searched for. It is important to understand, however, that while searching does leak *some* information to the provider, what is being leaked is exactly what the provider would learn from the act of returning the appropriate files to the customer (i.e., that these files contain some keyword in common). In other words, the information leaked to the cloud provider is not leaked by the cryptographic primitives, but by the manner in which the service is being used (i.e., to fetch files based on exact keyword matches). This leakage seems almost inherent to any efficient and reliable cloud storage service and is, at worst, less information than what is leaked by using a public cloud storage service. The only known alternative, which involves making the service provider return false positives and having the client perform some local filtering, is inefficient in terms of communication and computational complexity.

There are many types of searchable encryption schemes, each one appropriate to particular application scenarios. For example, the data processors in our consumer and small enterprise architectures could be implemented using symmetric searchable encryption (SSE), while the data processors in the large enterprise architecture could be based on asymmetric searchable encryption (ASE). In the following we describe each type of scheme in more detail.

Symmetric searchable encryption. SSE is appropriate in any setting where the party that searches over the data is also the one who generates it. Borrowing from storage systems terminology, we refer to such scenarios as single writer/single reader (SWSR). SSE schemes were introduced in [32] and improved constructions and security definitions were given in [23, 16, 19].

The main advantages of SSE are efficiency and security while the main disadvantage is functionality. SSE schemes are efficient both for the party doing the encryption and (in some cases) for the party performing the search. Encryption is efficient because most SSE schemes are based on symmetric primitives like block ciphers and pseudo-random functions. As shown in [19], search can be efficient because the typical usage scenarios for SSE (i.e., SWSR) allow the data to be pre-processed and stored in efficient data structures.

The security guarantees provided by SSE are, roughly speaking, the following:

1. without any tokens the server learns nothing about the data except its length
2. given a token for a keyword w, the server learns which (encrypted) documents contain w without learning w.

While these security guarantees are stronger than the ones provided by both asymmetric and efficiently searchable encryption (described below), we stress that they do have their limitations. In addition to the issues outlined above, all currently known constructions have deterministic tokens

which essentially means that the service provider can tell if a query is repeated (though it won't know what the query is).

The main disadvantage of SSE is that the known solutions tradeoff efficiency and functionality. This is easiest to see by looking at two of the main constructions proposed in the literature. In the scheme proposed by Curtmola et al. [19], search time for the server is optimal (i.e., linear in the number of documents that contain the keyword) but updates to the index are inefficient. On the other hand, in the scheme proposed by Goh [23], updates to the index can be done efficiently but search time for the server is slow (i.e., linear in the total number of documents). We also remark that neither scheme handles searches that are composed of conjunctions or disjunction of terms. The only SSE scheme we are aware of that handles conjunctions [24] is based on pairings on elliptic curves and is as inefficient as the asymmetric searchable encryption schemes discussed below. Another limitation of some searchable encryption constructions is that they are only secure in a setting where the queries are generated non-adaptively (i.e., without seeing the answers to previous queries). Schemes secure in an adaptive setting (i.e., where queries can depend on the answers of previous queries) are considered in [19].

Asymmetric searchable encryption (ASE). ASE schemes are appropriate in any setting where the party searching over the data is different from the party that generates it. We refer to such scenarios as many writer/single reader (MWSR). ASE schemes were introduced in [11] while improved definitions were given in [1]. Several works have shown how to achieve more complex search queries in the public-key setting, including conjunctive searches [28, 13] and range queries [13, 31]. Other issues related to the application of ASE in practical systems have been studied [5, 6, 22], as well as very strong notions of ASE that can guarantee the complete privacy of queries (at the cost of efficiency) [12].

The main advantage of ASE is functionality while the main disadvantages are inefficiency and weaker security guarantees. Since the writer and reader can be different, ASE schemes are usable in a larger number of settings than SSE schemes. The inefficiency comes from the fact that all known ASE schemes require the evaluation of pairings on elliptic curves which is a relatively slow operation compared to evaluations of (cryptographic) hash functions or block ciphers. In addition, in the typical usage scenarios for ASE (i.e., MWSR) the data cannot be stored in efficient data structures.

The security guarantees provided by ASE are, roughly speaking, the following:

1. without any tokens the server learns nothing about the data except its length,
2. given a token for a keyword w, the server learns which (encrypted) documents contain w.

Notice that 2 here is weaker than in the SSE setting. In fact, as pointed out by Byun et al. [15], when using an ASE scheme, the server can mount a dictionary attack against the token and figure out which keyword the client is searching for. It can then use the token (for which it now knows the underlying keyword) and do a search to figure out which documents contain the (known) keyword.

Efficient ASE (ESE). ESE schemes are appropriate in any setting where the party that searches over the data is different from the party that generates it and where the keywords are hard to guess. This falls into the MWSR scenario as well. ESE schemes were introduced in [8]. The main advantage

of efficient ASE is that search is more efficient than (plain) ASE. The main disadvantage, however, is that ESE schemes are also vulnerable to dictionary attacks. In particular, the dictionary attacks against ESE can be performed directly against the encrypted index (as opposed to against the token like in ASE).

Multi-user SSE (mSSE). mSSE schemes are appropriate in any setting where many parties wish to search over data that is generated by a single party. We refer to such scenarios as single writer/many reader (SWMR). Multi-user SSE schemes were introduced in [19].

In a mSSE scheme, in addition to being able to encrypt indexes and generate tokens, the owner of the data can also add and revoke users' search privileges over his data.

4.2 Attribute-based Encryption

Another set of cryptographic techniques that has emerged recently allows the specification of a decryption policy to be associated with a ciphertext. More precisely, in a (ciphertext-policy) attribute-based encryption scheme each user in the system is provided with a decryption key that has a set of attributes associated with it (this is how the "credentials" in Section 2 would be implemented). A user can then encrypt a message under a public key and a policy. Decryption will only work if the attributes associated with the decryption key match the policy used to encrypt the message. Attributes are qualities of a party that can be established through relevant credentials such as being a PartnerCorp employee or living in Washington State.

Attribute-based encryption was introduced in [29]. Improved constructions are given in [25, 27, 10]. The setting where attributes can be distributed by multiple parties is considered in [17, 18].

4.3 Proofs of Storage

A proof of storage is a protocol executed between a client and a server with which the server can prove to the client that it did not tamper with its data. The client begins by encoding the data before storing it in the cloud. From that point on, whenever it wants to verify the integrity of the data it runs a proof of storage protocol with the server. The main benefits of a proof of storage are that (1) they can be executed an arbitrary number of times; and (2) the amount of information exchanged between the client and the server is extremely small and independent of the size of the data.

Proofs of storage can be either privately or publicly verifiable. Privately verifiable proofs of storage only allow the client (i.e., the party that encoded the file) to verify the integrity of the data. With a publicly verifiable proof of storage, on the other hand, anyone that possesses the client's public key can verify the data's integrity.

Proofs of storage were introduced in [2] and [26]. Improved definitions and constructions were given in [30, 14, 20, 3] and dynamic proofs of storage (where the data can be updated) are considered in [4, 21, 33].

5 Cloud Services

Secure extranet. In addition to simple storage, many enterprise customers will have a need for some associated services. These services can include any number of business processes including sharing of data among trusted partners, litigation support, monitoring and compliance, back-up,

archive and audit logs. We refer to a cryptographic storage service together with an appropriate set of enterprise services as a secure extranet and believe this could provide a valuable service to enterprise customers.

Electronic health records. In February 2009, 19 billion dollars were provisioned by the U.S. government to digitize health records. This move towards electronic health records promises to reduce medical errors, save lives and decrease the cost of healthcare. Given the importance and sensitivity of health-related data, it is clear that any storage platform for health records will need to provide strong confidentiality and integrity guarantees to patients and care givers (see [9] for more regarding these issues).

Interactive scientific publishing. As scientists continue to produce large data sets which have broad value for the scientific community, demand will increase for a storage infrastructure to make such data accessible and sharable. To incent scientists to share their data, scientific societies such as the Optical Society of America are considering establishing a publication forum for data sets in partnership with industry. Such an interactive publication forum will need to provide strong guarantees to authors on how their data sets may be accessed and used by others, and could be built on a cryptographic cloud storage system like the one proposed here.

References

[1] M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. M. Lee, G. Neven, P. Paillier, and H. Shi. Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. In V. Shoup, editor, *Advances in Cryptology – CRYPTO '05*, volume 3621 of *Lecture Notes in Computer Science*, pages 205–222. Springer, 2005.

[2] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song. Provable data possession at untrusted stores. In P. Ning, S. De Capitani di Vimercati, and P. Syverson, editors, *ACM Conference on Computer and Communication Security (CCS '07)*. ACM Press, 2007.

[3] G. Ateniese, S. Kamara, and J. Katz. Proofs of storage from homomorphic identification protocols. In *To appear in Advances in Cryptology - ASIACRYPT '09*, Lecture Notes in Computer Science. Springer, 2009.

[4] G. Ateniese, R. Di Pietro, L. V. Mancini, and G. Tsudik. Scalable and efficient provable data possession. In *Proceedings of the 4th international conference on Security and privacy in communication netowrks (SecureComm '08)*, pages 1–10, New York, NY, USA, 2008. ACM.

[5] J. Baek, R. Safavi-Naini, and W. Susilo. On the integration of public key data encryption and public key encryption with keyword search. In *International Conference on Information Security (ISC '06)*, volume 4176 of *Lecture Notes in Computer Science*. Springer, 2006.

[6] J. Baek, R. Safavi-Naini, and W. Susilo. Public key encryption with keyword search revisited. In *International conference on Computational Science and Its Applications*, pages 1249–1259. Springer-Verlag, 2008.

[7] J. Bardin, J. Callas, S. Chaput, P. Fusco, F. Gilbert, C. Hoff, D. Hurst, S. Kumaraswamy, L. Lynch, S. Matsumoto, B. O'Higgins, J. Pawluk, G. Reese, J. Reich, J. Ritter, J. Spivey, and J. Viega. Security guidance for critical areas of focus in cloud computing. Technical report, Cloud Security Alliance, April 2009.

[8] M. Bellare, A. Boldyreva, and A. O'Neill. Deterministic and efficiently searchable encryption. In A. Menezes, editor, *Advances in Cryptology – CRYPTO '07*, Lecture Notes in Computer Science, pages 535–552. Springer, 2007.

[9] J. Benaloh, M. Chase, E. Horvitz, and K. Lauter. Patient controlled encryption: ensuring privacy of electronic medical records. In *ACM workshop on Cloud computing security (CCSW '09)*, pages 103–114. ACM, 2009.

[10] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *IEEE Symposium on Security and Privacy*, pages 321–334. IEEE Computer Society, 2007.

[11] D. Boneh, G. di Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology – EUROCRYPT '04*, volume 3027 of *Lecture Notes in Computer Science*, pages 506–522. Springer, 2004.

[12] D. Boneh, E. Kushilevitz, R. Ostrovsky, and W. Skeith. Public-key encryption that allows PIR queries. In A. Menezes, editor, *Advances in Cryptology – CRYPTO '07*, volume 4622 of *Lecture Notes in Computer Science*, pages 50–67. Springer, 2007.

[13] D. Boneh and B. Waters. Conjunctive, subset, and range queries on encrypted data. In *Theory of Cryptography Conference (TCC '07)*, volume 4392 of *Lecture Notes in Computer Science*, pages 535–554. Springer, 2007.

[14] K. Bowers, A. Juels, and A. Oprea. Proofs of retrievability: Theory and implementation. Technical Report 2008/175, Cryptology ePrint Archive, 2008.

[15] J. W. Byun, H. S. Rhee, H.-A. Park, and D. H. Lee. Off-line keyword guessing attacks on recent keyword search schemes over encrypted data. In *Secure Data Management*, volume 4165 of *Lecture Notes in Computer Science*, pages 75–83. Springer, 2006.

[16] Y. Chang and M. Mitzenmacher. Privacy preserving keyword searches on remote encrypted data. In J. Ioannidis, A. Keromytis, and M. Yung, editors, *Applied Cryptography and Network Security (ACNS '05)*, volume 3531 of *Lecture Notes in Computer Science*, pages 442–455. Springer, 2005.

[17] M. Chase. Multi-authority attribute based encryption. In *Theory of Cryptography Conference (TCC '07)*, volume 4392 of *Lecture Notes in Computer Science*, pages 515–534. Springer, 2007.

[18] M. Chase and S.M. Chow. Improving privacy and security in multi-authority attribute-based encryption. In *ACM conference on Computer and communications security (CCS '09)*, pages 121–130, New York, NY, USA, 2009. ACM.

[19] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky. Searchable symmetric encryption: Improved definitions and efficient constructions. In A. Juels, R. Wright, and S. De Capitani di Vimercati, editors, *ACM Conference on Computer and Communications Security (CCS '06)*, pages 79–88. ACM, 2006.

[20] Y. Dodis, S. Vadhan, and D. Wichs. Proofs of retrievability via hardness amplification. In *Theory of Cryptography Conference*, volume 5444 of *Lecture Notes in Computer Science*, pages 109–127. Springer, 2009.

[21] C. Erway, A. Kupcu, C. Papamanthou, and R. Tamassia. Dynamic provable data possession. In *ACM conference on Computer and communications security (CCS '09)*, pages 213–222, New York, NY, USA, 2009. ACM.

[22] T. Fuhr and P. Paillier. Decryptable searchable encryption. In *International Conference on Provable Security*, volume 4784 of *Lecture Notes in Computer Science*, pages 228–236. Springer, 2007.

[23] E-J. Goh. Secure indexes. Technical Report 2003/216, IACR ePrint Cryptography Archive, 2003. See `http://eprint.iacr.org/2003/216`.

[24] P. Golle, J. Staddon, and B. Waters. Secure conjunctive keyword search over encrypted data. In M. Jakobsson, M. Yung, and J. Zhou, editors, *Applied Cryptography and Network Security Conference (ACNS '04)*, volume 3089 of *Lecture Notes in Computer Science*, pages 31–45. Springer, 2004.

[25] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM conference on Computer and communications security (CCS '06)*, pages 89–98, New York, NY, USA, 2006. ACM.

[26] A. Juels and B. Kaliski. PORs: Proofs of retrievability for large files. In P. Ning, S. De Capitani di Vimercati, and P. Syverson, editors, *ACM Conference on Computer and Communication Security (CCS '07)*. ACM, 2007.

[27] R. Ostrovsky, A. Sahai, and B. Waters. Attribute-based encryption with non-monotonic access structures. In *ACM conference on Computer and communications security (CCS '07)*, pages 195–203. ACM, 2007.

[28] D. Park, K. Kim, and P. Lee. Public key encryption with conjunctive field keyword search. In C. H. Lim and M. Yung, editors, *Workshop on Information Security Applications (WISA '04)*, volume 3325 of *Lecture Notes in Computer Science*, pages 73–86. Springer, 2004.

[29] A. Sahai and B. Waters. Fuzzy identity-based encryption. In R. Cramer, editor, *Advances in Cryptology – EUROCRYPT '05*, volume 3494 of *Lecture Notes in Computer Science*, pages 457–473. Springer, 2005.

[30] H. Shacham and B. Waters. Compact proofs of retrievability. In *Advances in Cryptology - ASIACRYPT '08*. Springer, 2008.

[31] E. Shi, J. Bethencourt, T. Chan, D. Song, and A. Perrig. Multi-dimensional range query over encrypted data. In *IEEE Symposium on Security and Privacy*, pages 350–364, Washington, DC, USA, 2007. IEEE Computer Society.

[32] D. Song, D. Wagner, and A. Perrig. Practical techniques for searching on encrypted data. In *IEEE Symposium on Research in Security and Privacy*, pages 44–55. IEEE Computer Society, 2000.

[33] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou. Enabling public verifiability and data dynamics for storage security in cloud computing. In *European Symposium on Research in Computer Security (ESORICS '09)*, volume 5789 of *Lecture Notes in Computer Science*, pages 355–370. Springer, 2009.

[34] K. Zetter. Compay caught in texas data center raid loses suit against FBI. *Wired Magazine*, April 2009.