THE PUBLISHING HOUSE
OF THE ROMANIAN ACADEMY

PROCEEDINGS OF THE ROMANIAN ACADEMY, Series A,
Volume 14, Number 1/2013, pp. 72–77

NEW CRYPTOGRAPHIC CHALLENGES IN CLOUD COMPUTING ERA

Laurenţiu BURDUŞEL

"Politehnica" University of Bucharest
E-mail: laurentiu.burdusel@gmail.com

New trends in IT business are about outsourcing computing from on-premises resources to remote servers for cost savings and many other performance benefits. Helped by the new virtualization technologies, the Cloud can handle the requirements of the market. With the migration of the data and the applications into the cloud, new security issues appear, making the trust in Cloud not to be enough. The researchers found new cryptographic ways to protect the data in the Cloud. This paper makes a review of the new cryptographic techniques used for protecting the data from a remote storage, and not only protect, but to be able to process remote encrypted data.

Key words: cloud security, homomorphic encryption, predicate encryption

1. INTRODUCTION

With the growing of the technology performance, an important step to optimized hardware resources uses was virtualization, creating many virtual machines that run on the same physical machine. The next step to this evolution was the construction of large data centers that can be used by many users in common, based on VMs, called Clouds. The main benefit of Cloud Computing is the high-availability storage of data, but also the high parallel computing resources. For enterprises, the Cloud offers an alternative to the on-premises infrastructure to an off-premises one which means costs savings and accelerate adaptation for new applications and resources requirements.

The Cloud represents an evolution in distributed computing that takes advantage of technology advances. Cloud Computing include several types of services as:

– IaaS (infrastructure as a service) – the consumer has the capability to provision processing, storage, networks and other computing resources and to deploy and run arbitrary software,

– PaaS (platform as a service) – the consumer can deploy applications onto the Cloud using programming languages supported by the Cloud,

SaaS (software as a service) – the consumer use the applications that already running in the Cloud.

The Cloud infrastructures can be categorized as private or public. The private cloud infrastructure is managed and owned by the customer and resides on-premises, and the public cloud infrastructure is owned and managed by the cloud service provider and is located off-premises.

Thanks to virtualization technology, the resources offered to the users have the following attributes:

– Multi-tenancy,
– Highly scalable and elastic,
– Self-provisioned,
– Pay-per-use price model.

But, the migrating from on-premises to off-premises infrastructures introduces significant security issues:

a. Multi-tenancy: data and application in a VM share the same physical resources with other VM which can have a malicious behavior.
b. Data mobility and control: data that reside on the cloud's servers can be in any geographical location, and this fact can break some international laws and regulations.
c. Data remanence: the recycle of the storage resources can give data leakage in the case if the data are not safety erased.

d. Data privacy: the public nature of the cloud has significant implications in data confidentiality and privacy, especially when the data are stored in plain text. A Cloud Security Alliance report put the data losses and leakage in the top of the Cloud Security issues.

The requirements for data security in the Cloud are: confidentiality, verifiability (user can verify that data and computation was processed in an expected way) and integrity.

This paper is organized as follows: in Section 2 is described the main techniques for data security in the cloud, Section 3 presents the *predicative encryption* and its applications, Section 4 in detail the *homomorphic encryption*, Section 5 presents some data regarding the performance of the homomorphic encryption and the new challenges facing homomorphic encryption, and Section 6 contains the conclusions.

2. DATA SECURITY ISSUES IN THE CLOUD

To protect data on the cloud storage can be used methods for access control or to encrypt the data. Access control consists of both authentication and authorization. Cloud Service Provider usually uses weak authentication mechanisms, and the authorization controls are not very granular, and in some cases the access control does not provide enough protection. Even with these mechanisms, because data is in plain text, it is vulnerable to different kinds of attacks. Cryptographic mechanisms applied to data offer the best solution for data protection.

This means the customer must encrypt the data locally prior to uploading to cloud. But the searching and indexing the data could not be done, and neither processing. To search the data, the user must download all data locally, decrypt it and does the search or the cloud decrypts the data before the search, but it must have the decryption key which is not desirable.

In June 2009 has discovered first fully homomorphic encryption scheme that allows data to be processed without being decrypted. The only issue regarding this homomorphic encryption scheme is the immense computational effort. Other solution is to limit the amount of data that would need to be decrypted for processing using predicate encryption.

3. PREDICATE ENCRYPTION

Predicate encryption is a new paradigm, generalizing Identity-Based Encryption. In this paradigm secret keys correspond to predicates and the encrypted text to attributes. A secret key SK_f, corresponding to a predicate f, can decrypt a ciphertext with attribute I if and only if $f(I)=1$.

Predicate encryption (PE) is used for searching over encrypted data, evaluating certain encrypted attributes. Such encryption schemes work only for a few classes of predicates. PE has two important properties: does not reveal any information about the ciphertext and no information about the query predicate [9].

In [8] authors show how to construct a scheme for evaluation of inner product over $\mathbb{Z}_N$.

Definition. Key K with attribute y can decrypt ciphertext C_t with attribute x if and only if $P(x, y) = 1$.

Also, one of the most useful properties of the PE is that the owner of the data can delegate the decryption.

1. Owner generates Master Secret Key (MSK) and Public Key (PK);
2. Owner publishes PK
3. Other person (B) has a message M being attributed "private", and wants to send it to the owner. B computes $E(PK, M, private)$
4. Another person (C) has a message M' being attributed "work", and wants to send it to the owner. C computes $E(PK, M', work)$
5. The owner delegates the decryption of the M' with the attribute "work" to other entity and keep the "private" message for himself.

To achieve this goal it can be used Elliptic Curves Techniques, as follows:

a. Define group G: multiplicative of prime order p, and a bilinear map $e : G \times G \to G_T$, that satisfies the following properties:

$$e(g^a, g^b) = e(g,g)^{ab}$$
$$e(g,g) \neq 1, \forall a, b \in Z_p, g \in G. \quad (1)$$

b. Setup:
- Choose random $a, b \in Z_p$
- Set $MSK = a$, and $PK = g, g^b, e(g,g)^a$ and $H : \{0,1\} \to G$.

c. KeyGen:
- Choose random $t \in Z_p$, different for each entity. Entity has the attributes $x = \langle x_1, ..., x_n \rangle$
- Set $SK = ga + bt, gt, H(x_1)t, ..., H(x_n)t$.

d. Encryption:
- Choose random $s, r_1, ..., r_n$
- Ciphertext: $CT = Me(g,g)^{as}, g^s, (g^{b\lambda_1} H(y_1)^{r_1}, g^{r_1}), ..., (g^{b\lambda_n} H(y_1)^{r_n}, g^{r_n})$.

e. Decryption:
- Using g^s from CT and g^{a+bt} from SK, compute:

$$e(g^s, g^{a+bt}) = e(g,g)^{s(a+bt)} = e(g,g)^{sa} e(g,g)^{sbt}$$

- If the entity has all attributes $H(x_i)^t$,then it can obtain

$$e(g,g)^{bt(r)} e(g,g)^{bt(s-r)} = e(g,g)^{bts}$$, so it can compute M.

Hidden Vector Encryption (HVE) is a case of predicate encryption. In this case, the ciphertext attributes are represented as a vector $x = \langle x_1, x_2, ..., x_n \rangle$ and the keys are represented as $y = \langle y_1, y_2, ..., y_l \rangle$ and the $Match(x, y) = 1$ if and only if for all $x_i, y_i \neq *, x_i = y_i$.

1. $Setup(1^n, 1^l)$ => public key PK and secret key SK;
2. $Encrypt(PK, x)$ => encrypted attribute vector X;
3. $GenToken(SK, y)$ => key K_y;
4. $Test(X, T_y)$ => $Match(x, y)$.

A construction of this kind of algorithm is presented in [8] and in [9].

Applications

1. *Searchable Encrypted Databases.* PE can be used for secure searching on outsourced encrypted database in Cloud. There are efficient solutions for equality searches, but the issue is that the application in the Cloud can learn the access pattern of a user.
2. *Identit-Based Encryption and Attribute-Based Encryption.* IBE is a special case of PE for equality tests. In an ABE a user can have capabilities in senses of access control policy over the attributes of a ciphertext. In both schemes, the attributes and the identities are not hidden.

4. HOMOMORPHIC ENCRYPTION

Homomorphic describes a property of an encryption scheme. The homomorphic encryption permits processing of encrypted data on a remote storage without decrypting it. This method offers many advantages especially for the cloud paradigm because the users can benefit of the advantages offered by the Cloud and they also protect the data confidentiality and privacy which are one of the issues in the cases of storage and processing their data by an un-trusted third party. Practically, the owner delegates processing without giving away access.

A scheme to be homomorphic is enough to support AND and XOR Boolean operations, and to be *fully homomorphic* it must be able to perform these operations an unlimited number of times. Other Boolean operation can be obtained using these two operations according to De Morgan's laws. Also, it is enough to implement just NAND or just NOR.

A homomorphic encryption scheme can be named *"somewhat" homomorphic*. To be *fully homomorphic encryption* (FHE) it must be composed by a homomorphic encryption scheme and a bootstrappable function. In many real-world cases a "somewhat" homomorphic encryption could be enough, and has the main advantage to be much faster and more compact than FHE.

Gentry [2] proposed a FHE scheme that enables evaluation on arbitrary number of additions and multiplications on encrypted data. He first constructs a "somewhat homomorphic" encryption that uses "ideal lattice". This scheme can be used to handle only simple functions. He then shows how this scheme can be "bootstrapped" to create a FHE scheme. In the bootstrap scheme he uses the decryption function which must be simple enough to permit bootstrapping.

4.1. Homomorphic symmetric encryption

A homomorphic symmetric encryption can be constructed as follows:

1. Choose a secret key *p*, odd integer.
2. Encrypt a bit $m = \{0,1\}$:
   a. Choose a small integer *r*, and a large one *q*
   b. Ciphertext: $c = m + 2r + pq$
3. Decrypt *c*:
   a. $m = (c \bmod p) \bmod 2 = LSB(c) \oplus LSB([c/p])$, where $[c/p]$ is the nearest integer of c/p.

4.1.1. Why is homomorphic

Verifying the homomorphism of the scheme means to add and multiply two encrypted text and the decrypted result must be the same with the result of the same operation applied to plain text. Basically, the scheme is homomorphic because two near multiples of p are added or multiplied; the result will be another near multiple of p.

Consider two encrypted numbers:

$$\begin{aligned} c_1 &= q_1 p + 2r_1 + m_1 \\ c_2 &= q_2 p + 2r_2 + m_2. \end{aligned} \tag{2}$$

The result of the addition of the two numbers is:

$$c_1 + c_2 = (q_1 + q_2) p + 2(r_1 + r_2) + (m_1 + m_2), \tag{3}$$

where $2(r_1 + r_2) + (m_1 + m_2)$ is still smaller than *p*.

The result of the multiplication of the numbers is:

$$c_1 \times c_2 = (c_1 q_2 + q_1 c_2 - q_1 q_2) p + 2(2r_1 r_2 + r_1 m_2 + m_1 r_2) + m_1 m_2, \tag{4}$$

and $2(2r_1 r_2 + r_1 m_2 + m_1 r_2) + m_1 m_2$ is still smaller than *p*.

4.2. Homomorphic public key encryption

The components for a common public-key encryption are: key generation, encryption and decryption. In a *homomorphic public key encryption* another procedure appears the evaluation. The components of the scheme used to construct the somewhat homomorphic encryption are described below.

First, it is set a security parameter: λ and computed the number of bits for each number used in the scheme to preserve the security [2]:

$$\rho = \lambda, \eta = \lambda^2, \gamma = \lambda^5, \text{ and } t = \gamma + \lambda, \tag{5}$$

where: γ – number of bits of public key (pk); η – number of bits of secret key (sk); ρ – number of bits of noise r; t – number of integers in public key.

The cryptographic operations used in this scheme are:

1. $KeyGen(\lambda)$

   a. secret key: $p \leftarrow (2Z + 1) \cap [2^{\eta-1}, 2^{\eta})$, odd integer

   b. choose: $q_0, q_1, ..., q_t$; odds, integers, where q_0 is the biggest

   c. choose: $r_0, r_1, ..., r_t$; integers.

   Compute: $x_0 \leftarrow q_0 p + r_0, x_i \leftarrow [q_i + r_i]_{x_0} \in (-x_0/2, x_0/2)$.

   The public key: $pk = \langle x_0, x_1, ..., x_t \rangle$.

2. $Encrypt(pk, m \in \{0,1\})$

   Choose: $S \in \{1,2,..., t\}$

   Choose $r \in (2^{-\rho}, 2^{\rho})$

   Ciphertext: $c \leftarrow \left\lfloor m + 2r + \sum_{i \in S} x_i \right\rfloor_{x_0}$.

3. $Evaluate(pk, C, c_1, ..., c_t)$

   $D(sk, Eval(pk, C, \vec{c})) = C(m_1, ..., m_t)$.

4. $Decrypt(sk, c)$

   $m' \leftarrow \left\lfloor [c]_p \right\rfloor_2 = (c \bmod 2) \oplus ([c/p] \bmod 2)$, because $[c]_p = c - p[c/p]$.

4.3. Fully homomorphic encryption

The main problem is that the noise is growing with each operation. Gentry has a new idea to reduce the noise from time to time for preserving fresh numbers. This kind of operation is called *bootstrapping*. He proposed to encrypt the ciphertext before a specific number of operations, and before the noise is too big, the encrypted ciphertext will be decrypted. Both encryption/decryption operations are done through the homomorphic encryption scheme.

But this idea has only one issue, the encryption and the decryption function have too much operations and the homomorphic scheme cannot handle this. That is why the scheme must be changed to obtain a homomorphic one. Gentry use a technique that is named “squash the decryption circuit”. To be able to handle the encryption operation, it must be removed some of them. The idea was to add to public key a hint about the secret key, so anybody can post-process the ciphertext leaving less work for the decryption. This idea is used in server-aided cryptography.

5. CHALLENGES

Some cryptographers study the practical implications of these new revolutionary cryptographic techniques. The security relies on the hardness of the approximate GCD. Because the scheme is new, the attacks are still waiting to come, but the second test is about the performance.

Tests [1] shows that the homomorphic encryption works very well, but adding the bootstrappable scheme makes it unpractical. For instance [1] a SELECT command applied to a database of 10 records takes 7 days. That because the SELECT command has 619839 additions and multiplications.

But also can be used in some specific domains of e-government like e-health, e-voting, and finance.

The next step in this area is to reduce the computation complexity to obtain a practical fully homomorphic encryption. Some new papers try to remove the need of the bootstrap to increase the computational speed, as in [4], [5].

CONCLUSIONS

With the more and more interest of the IT community in the Cloud, new security problems are found. The protection of the data stored in the Cloud face new vulnerabilities. Best solutions for remote data protection remain cryptography. This article presents the new techniques that provide security to the private data, and also provide mechanisms for searching or processing encrypted data, like PE and FHE. The FHE represents a big step in modern cryptography and opens new challenges to cryptology researchers and also it helps the new IT technologies to be faster adopted.

REFERENCES

1. YOUSSEF GAHI, MOUHCINE GUENNOUN, KHALIL EL-KHATIB, *A Secure Database System using Homomorphic Encryption Schemes,* DBKDA, 2011.
2. Marten van Dijk, Craig Gentry, Shai Halevi, Vinod Vaikuntanathan, *Fully Homomorphic Encryption over Integers,* 2009.
3. KRISTIN LAUTER, MICHAEL NAEHRIG, VINOD VAIKUNTANATHAN, *Can Homomorphic Encryption be Practical?*, ACM CCSW, 2011.
4. CRAIG GENTRY AND SHAI HALEVI, *Fully Homomorphic Encryption without Squashing Using Depth-3 Arithmetic Circuits*, September 2011.
5. ZVIKA BRAKERSKI, CRAIG GENTRY, VINOD VAIKUNTANATHAN, *Fully Homomorphic Encryption without Bootstrapping*, 2011.
6. GIUSEPPE PERSIANO, *Predicate Encryption for Private and Searchable Remote Storage,* Workshop on Cryptography and Security in Clouds, 2011.
7. CRAIG GENTRY, *Manipulating Data while It Is Encrypted,* Lattice Crypto Day, 2010.
8. JONATHAN KATZ, AMIT SAHAI, BRENT WATERS, *Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Product*, 2007.
9. EMILY SHEN, ELAINE SHI, BRENT WATERS, *Predicate Privacy in Encryption Systems*, 2008.
10. SENY KAMARA, KRISTIN LAUTER, *Cryptographic Cloud Storage*, 2010.

Received March 13, 2012