Resilient Identity Based Encryption for Cloud Storage by using Aggregate Keys

L.MohamedIrfan, S.Muthurangasamy, T.Yogananth

Abstract— **Data sharing is an important feature in cloud storage. In this paper, we show how to securely, efficiently, share data with others in cloud storage. We describe novel public-key cryptography which produce constant-size ciphertexts such that it gives the efficient decryption rights for any set of ciphertexts are possible. The novelty is that one can combine any set of secret keys and make them as portable as a single key. In other words, the secret key holder can release a constant-size combined key for flexible choices of ciphertext set in cloud storage. This compact aggregate key can be conveniently sent to others or be stored in a smart card with very limited secure storage. On the other hand, when one carries the secret keys around in a mobile device without using special trusted hardware, the key is prompt to leakage, we provide an efficient key leakage cryptosystem which reduce desirable amount of key leakage.**

Index Terms—**Cloud storage, data sharing, key-aggregate encryption, resilient identity based encryption.**

I. INTRODUCTION

Recent days cloud storage is gaining popularity. In enterprise settings, we see the rise in demand for data outsourcing, which assists in the strategic management of corporate data. It is also used as a core technology behind many online services for personal applications. Now a days, it is easy to apply for free accounts for email, photo album, file sharing and/or remote access, with storage size more than 25GB (or a few dollars for more than 1TB). Together with the current wireless technology, users can access almost all of their files and emails by a mobile phone in any corner of the world.

A traditional way to ensure data privacy is to rely on the server to provide the access control after authentication, which means any unexpected privilege escalation will expose all data. In a shared-tenancy cloud computing environment, things become even worse. Data from different clients can be stored on separate virtual machines. but reside on a single physical machine. Data in a target VM could be stolen by instantiating another VM co-resident with the target one. Regarding availability of files, there are a series of cryptographic schemes which go as far as allowing a third-party auditor to check the availability of files on behalf of the data owner without leaking anything about the data, or without compromising the data owners anonymity. Likewise, cloud users probably will not hold the strong belief that the cloud server is doing a good job in terms of confidentiality. A cryptographic solution, which proven security relied on number-theoretic assumptions is more desirable, whenever the user is not perfectly happy with trusting the security of the VM or the honesty of the technical staff. These users are motivated to encrypt their data with their own keys before uploading them to the server.

Data sharing is an important benefits in cloud storage. For example, bloggers can let their friends view a subset of their private pictures; an enterprise may grant her employees access to a portion of sensitive data. The challenging problem is how to effectively share encrypted data. Of course users can download the encrypted data from the storage, decrypt them, then send them to others for sharing, but it loses the value of cloud storage. Users should be able to give the access rights of the sharing data to others so that they can access these data from the server directly. However, finding an efficient and secure way to share *partial* data in cloud storage is not trivial.

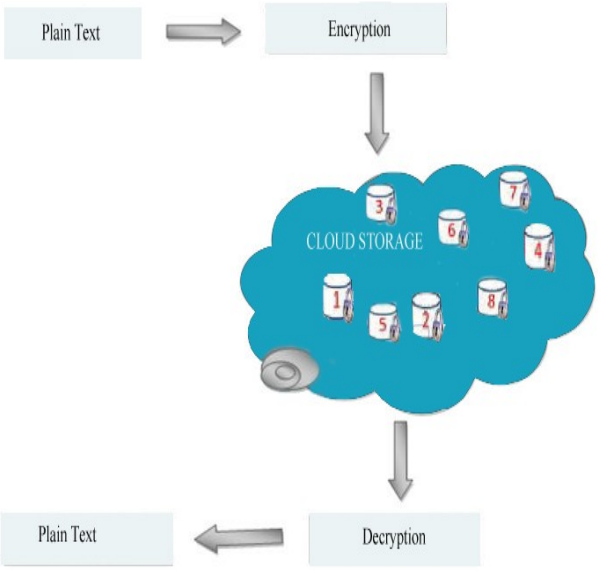

Figure1. process of data stored in cloud storage

Assume that Alice puts all her private data on cloud storage, and she does not want to expose her photos to everyone. Due to various data leakage possibility Alice cannot feel relieved by just relying on the privacy protection mechanisms provided by service provider, so she encrypts all the data's using her own keys before uploading. One day, Alice's friend, Bob, asks her to share the photos taken over all these years which Bob appeared in.

Manuscript received March, 2014.

L.MOHAMEDIRFAN *, Department of Computer Science Engineering, AnnaUniversity, Jay Shriram Group Of Institutions Tiruppur, India, 7418839800*

S.MUTHURANGASAMY*, Department of Computer Science Engineering,AnnaUniversity,JayShriramGroupOfInstitutions,Tiruppur,India,9843127229*

T.YOGANANTH*, Department of Computer Science Engineering AnnaUniversity, Jay Shriram Group Of Institutions, Tiruppur, India, 9003069962,*

Alice can then use the share function of cloud storage, but the problem now is how to give the *decryption rights* for these data's to Bob. A possible option Alice can choose is to securely send Bob the secret keys involved. Naturally, there are two ways for the traditional encryption:

- Alice encrypts all files with a single encryption key and gives Bob the corresponding secret key directly.
- Alice encrypts files with distinct keys and sends Bob the corresponding secret keys.

For the second method, there are practical The number of such keys is as many as the number of the shared data's. Say, a thousand.

Obviously, the first method is inadequate since all un chosen data may be also leaked to Bob. Transferring these secret keys inherently requires a secure channel, and storing these keys requires rather expensive secure storage. The costs and complexities involved generally increase with the number of the decryption keys to be shared. it is very heavy and costly.

Encryption keys also come with two flavors symmetric key or asymmetric key. Using symmetric encryption, when Alice wants the data to be originated from a third party, she has to give the encryptor her secret key; obviously, this is not always desirable. By contrast, the encryption key and decryption key are different in asymmetric-key encryption. The use of asymmetric-key encryption provides more flexibility for our applications. For example, in enterprise settings, every employee can upload encrypted data on the cloud storage server without the knowledge of the company's master-secret key.

Therefore, the best solution for the above problem is that Alice encrypts files with distinct public-keys, but only sends Bob a single decryption key. Since the decryption key should be sent via a secure channel and kept secret, small key size is always desirable. For example, we cannot expect large storage for decryption keys in the resource-constraint devices like smart phones, smart cards or wireless sensor nodes.

In traditional cryptography, we think the secret keys are completely hidden from the attackers. Many attacks such as timing attacks, power dissipation, cold-boot attacks, can extract some bits of information from the secret keys or the compromise encrypting system. Our goal is to design a resilient cryptographic system it will reduce leakage with comparable efficiency with previous system.

II. IDENTITY-BASED ENCRYPTION FROM BILINEAR PAIRING

In order to introduce identity-based encryption scheme and explain how a pair of nodes in a communication system can pre-shared a secret key in a non-interactive fashion, we need some concepts about paring and bilinear maps. A paring is a map which maps a pair of points in an elliptic curve (or it could be two different curves) to an element in a finite field, which are the building block and all pairing-based IBC schemes.

A. Bilinear Maps

Let G1, G2 be two groups of the same prime order q. G1 is as an additive group, (a group
of points on an elliptic curve), whereas G2 is a multiplicative subgroup of a finite field. Let P be an arbitrary generator of G1. Assume that discrete logarithm problem (DLP) is hard in both G1 andG2. A bilinear mapping is given byˆe : G1 × G1 7! G2which satisfy the following properties:

• Bilinearity: ˆe(aP, bQ) = ˆe(P,Q)ab for all P, Q 2 G1 and a, b 2 Z_q .

• Non-degeneracy: If P is a generator of G1, then ˆe(P, P) is a generator of G2. In other words, ˆe(P, P) 6= 1.

• Computable: There exists an efficient algorithm to compute ˆe(P,Q) for all P, Q 2 G1.

Weil pairings and Tate pairings are examples of such cryptographic bilinear mappings, where Tate pairings are computationally more efficient.

B. Parameter Generation

The security of pairing-based IBC schemes is based on the Bilinear Diffie-Hellman Problem (BDHP) which is defined as follows. Given two groups G1 and G2 of the same prime order q, a bilinear map ˆe : G1 × G1 7! G2 and a generator P of G1, the BDHP is to computeˆe(P, P)abc 2 G2 for any a, b, c 2 Z_q given (P, aP, bP, cP).In any IBC scheme, a key generation center (PKG) is needed which is responsible to select the system parameters and set up the system such that the BDH problem is hard. In the following paragraphs, we introduce the set up and extract algorithms of the original BFscheme.

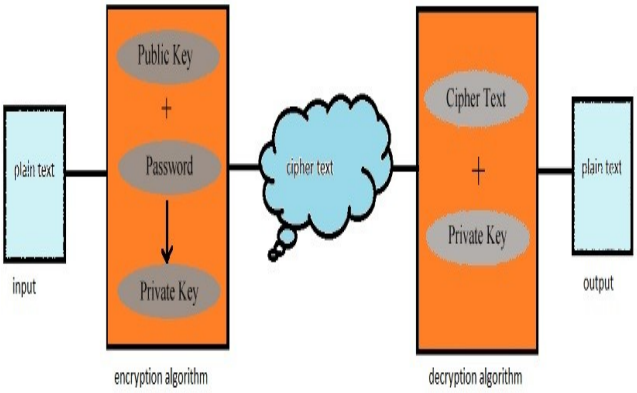

Figure2. Data Encryption Based on Aggregate Keys

which describe the secure set up of a pairing-based IBC scheme and secure derivation methods for ID based keys.

C. System Setup

1. On input of a security parameter k, the PKG uses a BDH parameter to generate a prime q, two groups G1 and G2 of order q, and a bilinear map ˆe : G1 × G1 7! G2.the PKG chooses a random generator P 2 G1.

2. The PKG picks a random s 2 Z_q , called a master private key, and sets Ppub = sP, called a master public key.

3. The PKG chooses a hash function that maps an arbitrarily long binary string to an element in group G1, i.e. H1:{0, 1}_ 7! G_1 the PKG publishes param =< G1,G2, q, P, ˆe, Ppub,H1 > as public parameters and keeps s secret. The PKG's private and public key pair is (s, Ppub).

D. Extract

The public key Qi 2 G_1 of a node i with identity IDi is asQi = H1(IDi), where IDi is an arbitrarily long binary string 2 {0, 1}_.The PKG derives the private keys di for every node i as di = sQi.

We can observe that instead of directly using identities as public keys, as done in Shamir's scheme here the identity string is first mapped to a point on an elliptic curve using hash function H1.Note that public keys can be computed from publicly available information, whereas private keys can only be computed by the PKG because the computation requires the PKG's private key s as input. The PKG generates the private keys and securely distributes them to the nodes. The key distribution channel between PKG and nodes needs to be authentic and confidential. Extension of ID-based Public Key.

E. KeyGen

In addition to pre-shared public keys, each pair of users i and j in a pairing-based IBC scheme is able to compute a pairwise pre-shared secret key Kij with Kij = ˆe(di,Qj) = ˆe(Qj , di) = ˆe(Qi,Qj)s this key is shared between users i and j in a non-interactive fashion, since both parties compute the bilinear mapping ˆe(·) over their own private key di and the public key Qj of the desired communication partner.

F. System Parameters

In order to perform encryption and decryption, PKG selects additional three functions H2,H3 and H4 which are defined as follows.

1. H2 : G2 ! {0, 1}n
2. H3 : {0, 1}_ × {0, 1}n ! Z_q
3. H4 : {0, 1}n ! {0, 1}n

where {0, 1}_ is the set consisting of any bit strings or any binary vectors with length n. H2 maps an element in the finite field to an n-bit vector, H3 maps a binary string and an n-bit string to a number < q, H4 maps an n-bit vector param =< G1,G2, q, P, ˆe,Ppub,H1,H2,H3,H4 >,which is known to each node in a system.

G. Encryption

When a user is transmitting a message m to user i confidentially, he encrypts musing user i's public-key and the system parameters in the following way:

Encrypt(m) ! C

where C is computed by

1. Randomly choose _ 2 {0, 1}_, and compute r = H3(_,m) and gi = ˆe(Qi, Ppub).
2. The ciphertext is a triplets given by C = (U, V,W) = (rP, _ _ H2(gri),m _ H4(_)).

H. Decryption

When user i receives C = (U, V,W), using his private key di, he decrypts

Decrypt(C) ! m or declare "invalid" by performing the following computing process:

1. Compute _ = V _ H2(ˆe(di,U)),m = W _ H4(_), and r = H3(_,m).2. If U = rP, return m; otherwise return "invalid".

III. FEATURES OF IBC

- The main feature of IBC schemes is the use of self-authenticating public keys. Identities are used as public keys and hence, identities Idi of network nodes and their corresponding public keys Qi do not need to be bound by certificates or any other means. All private keys di in IBC schemes are derived from the corresponding pre-determined public keys.. The PKG distributes the private keys di over a secure channel during the initialization of the network nodes.

(A) IBC schemes provide implicit and non-interactive pre-authentication among all network nodes. is due to the use of identities as public keys which entails many desirable properties. IBC schemes do not require any secure channel for pre-authentication because public keys are self-authentication to prior communicate.

(B) IBC schemes provide implicit public key validity checks. When verifying a signature in an ID-based signature scheme, we check the validity of the keys at the same time. In case of an IBE scheme, only users with valid keys are able to decrypt.

(C) Paring-based IBC schemes provide a pairwise secret key Kij. the paring-based schemes offers all the benefits of symmetric key schemes without the need of a secure channel during pre-authentication.

IV. REST OF ENCRYPTION TECHNIQUES

In Attribute-based encryption (ABE) each encrypted message to be associated with an attribute, and the master-secret key holder can extract a secret key for a policy of these attributes so that a encrypted message can be decrypted by this key if its associated attribute conforms to the policy. For example, with the secret key for the policy (2 ∨3 ∨6 ∨8), one can decrypt encrypted message tagged with class 2, 3, 6 or 8. However, the major concern in ABE is collusion-resistance but not the compactness of secret keys. Indeed, the size of the key often increases linearly with the number of attributes it encompasses, or the size of encrypted message not constant.

To give the decryption power of some ciphertexts without sending the secret key to the give a useful primitive is proxy re-encryption (PRE). A PRE scheme allows Alice to give the server (proxy) the ability to convert the ciphertexts encrypted under her public-key into ones for Bob. PRE is well known to have numerous applications including cryptographic file system . Nevertheless, Alice has to trust the proxy that it only converts ciphertexts according to her instruction, which is what we want to avoid at the first place. Even worse, if the proxy colludes with Bob, some form of Alice's secret key can be recovered which can decrypt Alice's (convertible) ciphertexts without Bob's further help. That also means that the transformation key of proxy should be well protected. Using PRE just moves the secure key storage requirement from the give to the proxy. It is thus undesirable to let the proxy reside in the storage server. That will also be inconvenient since every decryption requires separate interaction with the proxy.

V. PERFORMANCE ANALYSIS

Our approaches allow the compression factor F ($F = n$ in our schemes) to be a tunable parameter, at the cost of $O(n)$-sized system parameter. Encryption can be done in constant time, while decryption can be done in $O(|S|)$ group multiplications (or point addition on elliptic curves) with 2 pairing operations, where S is the set of ciphertext classes decryptable by the granted aggregate key and $|S| \leq n$. As expected, key extraction requires $O(|S|)$ group multiplications as well, which seems unavoidable. However, as demonstrated by the experiment results, we do not need to set a very high n to have better compression than the

tree-based approach. Note that group multiplication is a very fast operation .Again, we confirm empirically that our analysis is true. We implemented the basic KAC system in C with the Pairing-Based Cryptography (PBC) Library8 version0.4.18 for the underlying elliptic-curve group and pairing operations. Since the granted key can be as small as one G element, and the ciphertext only contains two G and one G*T* elements, we used (symmetric) pairings over Type-A (super singular) curves as defined in the PBC library which offers the highest efficiency among all types of curves, even though Type-A curves do not provide the shortest representation for group elements. In our implementation, *p* is a 160-bit So l in as prime, which offers 1024-bit of discrete-logarithm security. With this Type-A curves setting in PBC, elements of groups G and G*T* take 512 and 1024 bits to represent, respectively. The test machine is a Sun UltraSparc IIIi system with dual CPU (1002 MHz) running Solaris, each with2GB RAM. The timings reported below are averaged over 100 randomized runs. In our experiment, we take the *number of ciphertext classes n* = 216 = 65536. The Setup algorithm, while outputting (2*n* + 1) elements by doing (2*n* − 2) exponentiations, can be made efficient by preprocessing function offered by PBC, which saves time for exponentiating the same element (*g*) in the long run. This is the only "low-level" optimization trick we have used. All other operations are implemented in a straight forward manner.

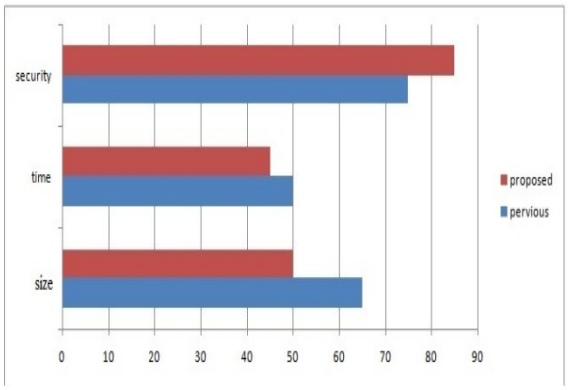

Figure3.Comparision With Existing Systems

In particular, we did not exploit the fact that ˆ*e*(*g*1, *gn*) will be exponentiated many times across different encryptions. However, we pre-computed its value in the setup stage, such that the encryption can be done without computing any pairing. Our experiment results are shown in Table 3. The execution times of Setup, KeyGen, Encrypt are independent of the delegation ratio *r*. In our experiments, KeyGen takes 3.3 milliseconds and Encrypt takes 6.8 milliseconds. As expected, the running time complexities of Extract and Decrypt increase linearly with the delegation ratio *r*(which determines the size of the delegated set *S*). Our timing results also conform to what can be seen from the equation in Extract and Decrypt — two pairing operation stake negligible time, the running time of Decrypt is roughly a double of Extract. Note that our experiments dealt with up to 65536 number of classes (which is also the compression factor), and should be large enough for fine-grained data sharing in most situations. Finally, we remark that for applications where then number of ciphertext classes is large but the *non-confidential l* storage is limited, one should deploy our schemes using the Type-D pairing bundled with the PBC, which only requires 170-bit to represent an element in G. For *n* = 216, the system parameter requires approximately2.6 megabytes, which is as large as a lowerqualityMP3 file or a higher-resolution JPEG file that a typical cellphone can store more than a dozen of them .But we saved expensive *secure* storage without the has self-managing a hierarchy of delegation classes.

VI. CONCLUSION

When one carries the secret keys around in a mobile device without using special trusted hardware, the key is prompt to leakage, so we design leakage resilient cryptosystem using identity based encryption which allows efficient and flexible key delegation. users' data privacy is a central question of cloud storage. With more mathematical tools, cryptographic schemes are getting more versatile and often involve multiple keys for a single application. In this paper, we consider how to "compress" secret keys in public-key cryptosystems in cloud storage. On the other hand, a promising direction is to improve the leakage allowed from each secret key as the fraction of its size. and all leakage-resilient IBE systems, including the ones presented in this paper, leakage is allowed from only one secret key per identity. although, this can be easily achieved by generating the randomness of the secret-key algorithm using a pseudo-random generator. And IBE schemes are affected by key escrow attacks. there are two kinds of key escrow attacks, (1) passive attacks by honest-but-curious PKGs such as eavesdropping and(2) active attacks by dishonest PKGs such as impersonation attacks. Passive attacks by curious-but-honest PKGs can be prevented by executing a Diffie-Hellman (DH)-like key agreement protocol. Unfortunately, active attacks by dishonest PKGs cannot be fully prevented.in future extension implement efficient mechanism to prevent key escrow attacks is an interesting direction.

REFERENCES

[1] D. Boneh, R. Canetti, S. Halevi, and J. Katz, "Chosen-Ciphertext Security from Identity-Based Encryption," *SIAM Journal on Computing (SIAMCOMP)*, vol. 36, no. 5, pp. 1301–1328, 2007.

[2] T. H. Yuen, S. S. M. Chow, Y. Zhang, and S. M. Yiu, "Identity-Based Encryption Resilient to Continual Auxiliary Leakage," in *Proceedings of Advances in Cryptology - EUROCRYPT '12*, ser. LNCS, vol. 7237, 2012, pp. 117–134.

[3] S. S. M. Chow, Y. Dodis, Y. Rouselakis, and B. Waters, "Practical Leakage-Resilient Identity-Based Encryption from Simple Assumptions, "in *ACM Conference on Computer and Communications Security*, 2010, pp. 152–161.

[4] D. Boneh and M. K. Franklin, "Identity-Based Encryption from the Weil Pairing," in *Proceedings of Advances in Cryptology CRYPTO'01*, ser. LNCS, vol. 2139.Springer, 2001, pp. 213–229.

[5] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and Verifiably Encrypted Signatures from Bilinear Maps," in *Proceedings of Advances in Cryptology - EUROCRYPT '03*, ser. LNCS, vol. 2656. Springer, 2003, pp. 416–432.

[6] Stefan Dziembowski and Krzysztof Pietrzak. Leakage-Resilient Cryptography. In *FOCS*, pages293–302, 2008.

[7] Adi Shamir. Identity-Based Cryptosystems andSignature Schemes. In *CRYPTO*, pages 47–53, 1984.

[8] Moni Naor and Gil Segev. Public-Key CryptosystemsResilient to Key Leakage. In *CRYPTO*, pages 18–35,2009.

[9] Paul C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *CRYPTO*, pages 104–113, 1996.

[10] Sebastian Faust, Eike Kiltz, Krzysztof Pietrzak, andGuy Rothblum. Leakage Resilient Signatures. In *TCC*, pages 455–479, 2010.
[11] Craig Gentry. Practical Identity-Based EncryptionWithout Random Oracles. In *EUROCRYPT*, pages445–464, 2006.
[12] Moni Naor and Gil Segev. Public-key cryptosystems resilient to key leakage. In CRYPTO, pages 18–35, 2009. Also available at http://eprint.iacr.org/2009/105.
[13] Jonathan Katz and Vinod Vaikuntanathan. Signature schemes with bounded leakage resilience. In ASIACRYPT 2009. Springer, 2009. http://www.mit.edu/ ˜vinodv/papers /asiacrypt09/KV-Sigs.pdf.
[14] Yevgeniy Dodis and Daniel Wichs. Non-malleable extractors and symmetric keycryptography from weak secrets. In STOC, 2009. Full version at http://eprint.iacr.org /2008/503.
[15] Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. SIAM J. Comput.,33(1):167–226, 2004

L.MOHAMED IRFAN Pursuing Final Year Computer Science Engineering in Jay Shriram Group Of Institutions, Tirupur. and doing project on cloud computing domain.

S.MUTHURANGASAMY Pursuing Final Year Computer Science Engineering in Jay Shriram Group Of Institutions, Tirupur. and doing project on cloud computing domain. .

T.YOGANANTH, Received his B.E degree in Computer Science and Engineering from Coimbatore Institute of Technology, Coimbatore in 2008 and M.E degree in Computer Science and Engineering from Hindustan University, Coimbatore in 2012

A.M.RAVISHANKAR Received his B.E degree in Computer Science and Engineering from Kongu College of Engineering And Technology,Erode in 2008 and M.E degree in Computer Science and Engineering from kumarasmay engineering collage Coimbatore in 2012

