Key-Aggregate Cryptosystem for Scalable Data Sharing in Cloud Storage

Anumol A
Student,Mtech CSE
Younus College of Engineering and Technology,
Vadakkevila, Kollam-691010

Jiji. N
Associate.Professor in CSE
Younus College of Engineering and Technology,
Vadakkevila, Kollam-691010

Abstract- Data sharing is an important functionality in cloud storage. In this article, we show how to securely, efficiently, and flexibly share the data with others in cloud storage. We describe new public-key cryptosystems which produce constant-size cipher texts such that efficient delegation of decryption rights for any set of cipher texts are possible. The novelty is that one can aggregate any set of secret keys and make them as compact as a single key, but encompassing the power of all the keys being aggregated. In other words, the secret key holder can release a constant-size aggregate key for flexible choices of cipher text set in cloud storage, but the other encrypted files outside the set remain confidential. This compact aggregate key can be conveniently sent to others or be stored in a smart card with very limited secure storage. We provide formal security analysis of our schemes in the standard model. We also describe other application of our schemes.In particular our schemes give the first public-key patient controlled encryption for flexible hierarchy, which was yet to be known.

1. INTRODUCTION

Computers have become an indivisible part of our life. As the use of computers in our day to day life increases the computer resources that we need also increases. For enterprises affordability becomes a huge factor. They have to face problems like the huge cost of hardware, deployment and maintenance of software's, software bugs, machine failures, hardware crashes etc. and this might cost a headache to such a community.

Cloud computing comes in rescue and provide solutions for these problems. Cloud computing is an internet based computing in which large group of remote servers is networked to allow the centralized storage of data and online access to computer services or resources rather than saving or installing them on your personal or office computers.

While enjoying the convenience brought by this new technology, users also start worrying about losing control of their own data. Security of stored data and data in transit may be a great concern when storing sensitive data at cloud storage provider, since cloud storage is a rich resource for hackers and national security agencies. As cloud is gaining more popularity more and more organization are waiting to move towards cloud but the key concern about moving towards cloud has been security. Information officer of an organization while deciding to move to cloud he would have lot of questions.

1. Is my data secure on cloud?

2.Can others access my confidential data's? For e.g. if a competitor is also using the same cloud infrastructure how safe is my data, how confidential is my data?

3.What if an attacker brings down my application hosted on cloud?

Data in cloud should be stored in a secure manner i.e. stored in an encrypted form. cryptography plays an important role, to restrict client from direct accessing of shared data.

Key Aggregate Cryptosystem for scalable data sharing in cloud storage is an efficient public key encryption scheme which supports flexible delegation in the sense that any subset of the cipher text (produced by encryption scheme) is decryptable by a constant size decryption key(generated by the data owner). Data owner can simply send a single aggregate key to the delegate to decrypt the key.

Challenging problem in sharing data in cloud is Ofcourse, 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 delegate the access rights of sharing the data to others so that they can access these data from the server directly. Finding an efficient and secure way to share the partial data in cloud storage is not trivial.

Assume that Alice puts all her private photos on Dropbox, and she does not want to expose her photos to everyone. Due to various data leakage possibility Alice encrypts all her photos 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. Alice can then use the share function of Dropbox, but the problem now is how to delegate the decryption rights for these photos to Bob. Naturally there are two extreme ways for her under the traditional encryption paradigm.

- 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.

The first method is inadequate since all unchosen data may also leaked to Bob. For the second method, there are practical concerns on efficiency. The number of such keys is as many as the number of shared photos, say thousand. Transferring these keys inherently requires a secure channel, and storing these keys requires rather expensive secure storage. The costs and complexities involved generally increases with the number of decryption keys to be shared. In short, it is very heavy and costly to do that. Encryption keys also come with two flavors — symmetric key or asymmetric (public) 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 public-key encryption. The use of public-key encryption gives more flexibility for our applications. For example, in enterprise settings, every employee can up- load encrypted data on the cloud storage server without the knowledge of the company's master-secret key.

Key Aggregate Cryptosystem is the best solution for the above problem. Alice encrypts files with distinct public keys, but only sends Bob a single (constant-size) decryption key. Since the decryption key should be send through 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. Using KAC we can minimize the communication requirements such as bandwidth, rounds of communication.

II. RELATED WORKS

This section we compare our basic KAC schemes with other possible solutions on sharing in secure cloud storage.

Cryptographic keys for a predefined Hierarchy

Cryptographic key assignment schemes aim to minimize the expense in storing and managing secret keys for general cryptographic use. Utilizing a tree structure a key for a given branch can be used to derive the keys for descendant nodes (but not the other way round). Just granting the parent key implicitly grants all the keys of its descendant nodes. Alice can first classify the cipher text classes according to their subjects.

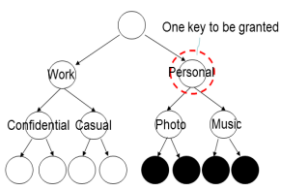

Figure:1

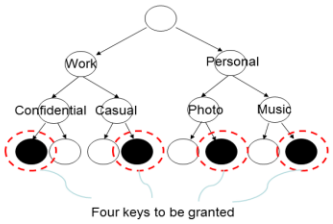

Figure:2

Each node in the trees represents a secret key. The leaf node represents keys for individual cipher text classes. Filled circles represent the keys for the classes to be delegated and circles circumvented by dotted lines represent the keys to be granted. Note that every key of the non leaf node can derive the keys of its descendant nodes.

if Alice wants to share all the files in the "personal" category, she only needs to grant the key for the node "personal", which automatically grants the delegatee the keys of all the descendant nodes ("Photo", "Music"). This is ideal case, where most classes to be shared belong to the same branch and thus a parent key of them is sufficient.
However it is still difficult for general cases. if Alice shares her demo music at work ("work"-> "casual"-> "demo" and "work"-> "confidential"-> "demo") with a colleague who also has the rights to see some of her personal data, what she can do is to give more keys, which leads to an increase in the total key size. One can see that tis approach is not flexible when the classifications are more complex and she wants to share different sets of files to different peoples. For this delegatee in our example, the number of granted keys becomes the same as the number of classes. In general, hierarchical approaches can solve the problem only partially if one tends to share all files under certain branch in the hierarchy. On, average, the number of keys increases with the number of branches. It is unlikely to come up with a hierarchy that can save the number of total keys to be granted for all individuals (which can access a different set of leaf-nodes) simultaneously.

Compact Key In Identity-Based Encryption

Identity Based-Encryption (IBE) is a type of public key encryption in which the public key of a user can be set as an identity string of the user (e.g. an email address). There is a trusted party called private key generator (PKG) in IBE which holds a master-secret key and issues a secret key to each user with respect to the user identity. The encryptor can take the public key parameter and a user identity to encrypt a message. The recipient can decrypt this cipher text by his secret key. Guo, tried to build IBE with key aggregation. In their schemes, key aggregation is constrained in the sense that all keys to be aggregated must come from different identity divisions. While there are an exponential number of identities and thus secret keys, only a polynomial number of them can be aggregated. Most importantly, their key-aggregation comes at the expense of O(n) sizes for both cipher texts and public parameter, where n is the number of secret keys which can be aggregated into a constant size one. This greatly increases

the costs of storing and transmitting ciphertexts, which is impractical I many situations such as shared cloud storage. As we mentioned, our schemes feature constant ciphertext size, and their security holds in the standard model.

Attribute-Based Encryption

Attribute based encryption (ABE) allows each ciphertext class 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 cipher text 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 ciphertext tagged with class 2,3,6,8. But, the size of the key often increases with the number of attributes it encompasses, or the ciphertext-size is not constant.

III.THE PROPOSED SYSTEM

In modern cryptography, a fundamental problem we often study is about leveraging the secrecy of a small piece of knowledge into the ability to perform cryptographic functions (e.g. encryption, authentication) multiple times. In this paper, we study how to make a decryption key more powerful in the sense that it allows decryption of multiple cipher texts, without increasing its size.

Specifically the problem statement is-"To design an efficient public key encryption scheme which supports flexible delegation in the sense that any subset of the cipher texts (produced by encryption scheme) is decryptable by a constant size decryption key (generated by the owner of master secret key)."

We solve this problem by introducing a special type of public-key encryption which we call key-aggregate cryptosystem (KAC). In KAC cipher texts are categorized into different classes. No special relation is required between classes. Users, encrypt a message not only under a public key, but also under the identifier of these cipher text classes. The key owner holds a master secret key, which can be used to extract secret keys for different classes. The extracted key can be an aggregate key which is as compact as a secret key for a single class, but aggregate the power of many such keys. i.e., the decryption power for any subset of cipher text classes.

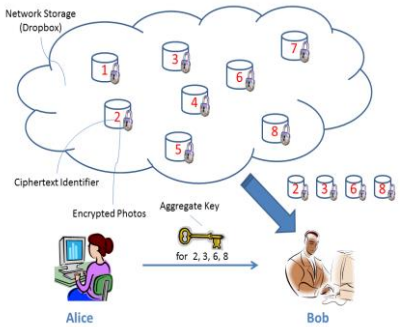

Figure:3 KAC system Architecture

With our solution Alice can simply send Bob a single aggregate key via a secure e-mail. Bob can download the encrypted photos from Alice's dropbox space and then use this aggregate key to decrypt these encrypted photos.

The sizes of cipher text, public key, master secret key and aggregate key in our KAC schemes are all of constant size. A canonical application of KAC is data sharing. The key aggregation property is especially useful when we expect the delegation to be efficient and flexible. The schemes enable a content provider to share her data in a confidential and selective way, with a fixed and small cipher text expansion, by distributing to each authorized user a single and small aggregate key.

IV.FIVE ALGORITHMIC STEPS IN KAC

Framework.

A key-Aggregate encryption scheme consists of five polynomial-time algorithms as follows:-

The data owner establishes the public system parameter via setup and generates a public/master secret key pair via KeyGen. Messages can be encrypted via Encrypt by anyone who also decides what cipher text class is associated with plain text message to be encrypted. The data owner can use the master secret to generate an aggregate decryption key for a set of cipher text classes via Extract. The generated keys can be passed to delegatees securely (through secure emails or secure devices). Finally, any user with an aggregate key can decrypt any cipher text provided that cipher text's class is contained in the aggregated key via Decrypt.

- Setup (λ, n): executed by the data owner to setup an account on a server. On input a security level parameter, λ and the number of cipher text classes n (ie, class index should be an integer bounded by 1 and n), which outputs the public system parameter param.
- KeyGen: executed by data owner to randomly generate a public/master-secret key pair (pk, msk).
- Encrypt (pk, i, m): executed by anyone who wants to encrypt the data. On input a public key pk, an index I denoting cipher text class, and a message m, it outputs a ciphertext 'C'.
- Extract (msk, S): executed by the data owner for delegating the decrypting power for a certain text of ciphertext classes to a delegatee. On input the master-secret key msk and a set S of indices corresponding to different classes, it outputs the aggregate key for set S denoted by Ks.
- Decrypt (Ks, S, i, C): executed by the delegatee who received an aggregate key Ks generated by Extract. On input Ks, the set S, an index I denoting the cipher text class, the cipher text C

belongs to, and C, it outputs the decrypted result m if i € S.

Sharing Encrypted data.

A canonical application of KAC is data sharing. The key aggregation property is especially useful when we expect the delegation to be efficient and flexible. The schemes enable a content provider to share her data in a confidential and selective way, with a fixed and small cipher text expansion, by distributing to each authorized user a single and small aggregate key.

Here we describe the main idea of data sharing in cloud storage using KAC.

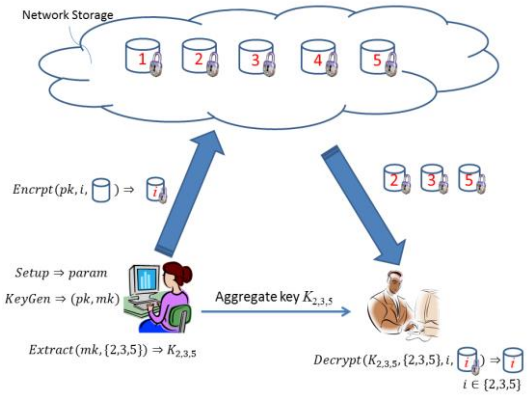

Figure:4

Suppose Alice wants to share her data m_1, m_2, ….m_v on the server. She first performs Setup (λ,n) to establish connection with server to get param and execute KeyGen to get the public/master secret key pair (pk, msk). The system parameter param and public-key pk can be made public and master secret key msk should be kept secret by Alice. Anyone (including Alice herself) can then encrypt each message m_i by C_i = Encrypt (pk, i, m_i). The encrypted data are uploaded to server. Once Alice is willing to share a set S of her data with a friend Bob, she can compute the aggregate key Ks for Bob by performing Extract (msk, S). Since Ks is just a constant size key, it is easy to be sent to Bob via a secure e-mail. After obtaining the aggregate key, Bob can download the data he is authorized to access. That is, for each i € S, Bob downloads C_i (and some needed values in param) from the server. With the aggregate key Ks, Bob can decrypt each C_i by Decrypt (Ks, S, i, C_i) for each i € S.

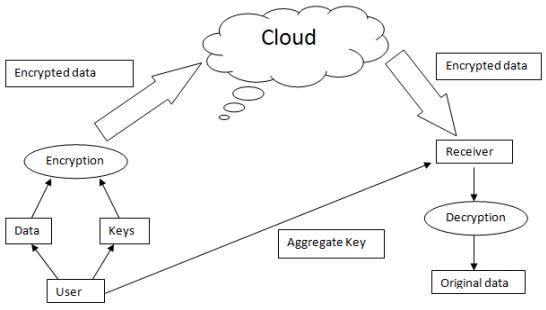

Figure:5 DATA FLOW ARCHITECTURE

V.BASIC CONSTRUCTION OF KAC

The design of our basic scheme is inspired from the collusion-resistant broadcast encryption scheme proposed by Boneh. Although their scheme supports constant-size secret keys, every key only has the power for decrypting ciphertexts associated to a particular index. We thus need to devise a new Extract algorithm and the corresponding Decrypt algorithm.

- Setup(λ; n): Randomly pick a bilinear group G of prime order p where $2^{\lambda} \le p \le 2^{\lambda+1}$, a generator g £ G and α £ Z_p. Compute $g_i = g^{\alpha^i}$ £ G for i=1,…,n,n+2,….,2n. Output the system parameter as param=‹g, g_1,…g_n,g_{n+2},….g_{2n}> (_ can be safely deleted after Setup).
  Note that each ciphertext class is represented by an index in the integer set{1, 2, _ _ _ , n} where n is the total number of ciphertext classes.
- KeyGen( ): Pick ϒ £ Z_p, Output the public and master-secret key pair: (pk = v = g^{Υ}, msk=ϒ).
- Encrypt(pk,i,m): For a message m £ G_T and an index i £ {1, 2, _ _ _ , n}, randomly pick t£ Z_p and compute the ciphertext as C=< g^t, $(vg_i)^t$, $m.e(g_1,g_n)^t$ >.
- Extract(msk= ϒ,S): For the set S of indices j's the aggregate key is computed as Ks= $\Pi_{j£S} g^{\Upsilon_{n+1-j}}$.
- Decrypt(Ks,S,I,C=<c_1,c_2,c_3>): If i € S, Output ⊥ .Otherwise, return the message:m=c_3.e(Ks. $\Pi_{j£S,j\neq i} g^{\Upsilon_{n+1-j+I}}, C_1)/e(\Pi_{j£S} g^{\Upsilon_{n+1-j}}, C_2)$.

V.PERFORMANCEANALYSIS

Comparison of KAC with other schemes

A comparison of the number of granted keys between three methods is depicted in

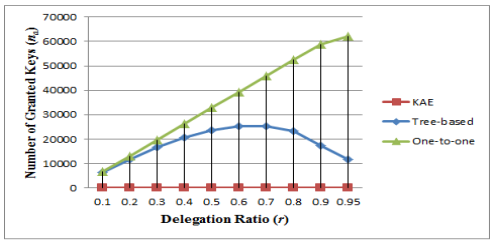

Figure:6

We can see that if we grant the key one by one, the number of granted keys would be equal to the number of the delegated ciphertext classes. With the tree-based structure, we can save a number of granted keys according to the delegation ratio. On the contrary, in our proposed approach, the delegation of decryption can be efficiently implemented with the aggregate key, which is only of fixed size.

VI.CONCLUSION

Thus data privacy and security is maintained by designing a public key cryptosystem called as Key Aggregate Cryptosystem (KAC). This KAC helps user to share their data partially over cloud with constant size key pair of public-master keys and also receiver can decrypt this data with single constant size aggregate key. No matter which one among the power set of classes, the delegatee can always get an aggregate key of constant size. Our approach is more flexible than hierarchical key assignment which can only save spaces if all key-holders share a similar set of privileges.

VII. FUTURE ENHANCEMENT

A limitation in our work is the predefined bound of the number of maximum ciphertext classes. In cloud storage, the number of ciphertexts usually grows rapidly. So we have to reserve enough ciphertext classes for the future extension. Otherwise, we need to expand the public-key. Another limitation in our work is that the aggregate key has been send to delegatee through email without any security so as a future extension we can encrypt the aggregate key and send it to delegatee

REFERENCES

[1] S. S. M. Chow, Y. J. He, L. C. K. Hui, and S.-M. Yiu, "SPICE - Simple Privacy-Preserving Identity-Management for Cloud Environment," in Applied Cryptography and Network Security – ACNS 2012, ser. LNCS, vol. 7341. Springer, 2012, pp. 526–543.

[2] L. Hardesty, "Secure computers aren't so secure," MIT press, 2009, http://www.physorg.com/news176107396.html.

[3] C. Wang, S. S. M. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy-Preserving Public Auditing for Secure Cloud Storage," IEEE Trans. Computers, vol. 62, no. 2, pp. 362–375, 2013.

[4] B. Wang, S. S. M. Chow, M. Li, and H. Li, "Storing Shared Data on the Cloud via Security-Mediator," in International Conference on Distributed Computing Systems - ICDCS 2013. IEEE, 2013.

[5] S. S. M. Chow, C.-K. Chu, X. Huang, J. Zhou, and R. H. Deng, "Dynamic Secure Cloud Storage with Provenance," in Cryptography and Security: From Theory to Applications - Essays Dedicated to Jean-Jacques Quisquater on the Occasion of His 65th Birthday, ser. LNCS, vol. 6805. Springer, 2012, pp. 442–464.

[6] 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.

[7] M. J. Atallah, M. Blanton, N. Fazio, and K. B. Frikken, "Dynamic and Efficient Key Management for Access Hierarchies," ACM Transactions on Information and System Security (TISSEC), vol. 12, no. 3, 2009.

[8] J. Benaloh, M. Chase, E. Horvitz, and K. Lauter, "Patient Controlled Encryption: Ensuring Privacy of Electronic Medical Records," in Proceedings of ACM Workshop on Cloud Computing
Security (CCSW '09). ACM, 2009, pp. 103–114.

[9] F. Guo, Y. Mu, Z. Chen, and L. Xu, "Multi-Identity Single-Key Decryption without Random Oracles," in Proceedings of Information Security and Cryptology (Inscrypt '07), ser. LNCS, vol. 4990. Springer, 2007, pp. 384–398.

[10] V. Goyal, O. Pandey, A. Sahai, and B. 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 '06). ACM, 2006, pp. 89–98.

[11] S. G. Akl and P. D. Taylor, "Cryptographic Solution to a Problem of Access Control in a Hierarchy," ACM Transactions on Computer Systems (TOCS), vol. 1, no. 3, pp. 239–248, 1983.

[12] G. C. Chick and S. E. Tavares, "Flexible Access Control with Master Keys," in Proceedings of Advances in Cryptology – CRYPTO '89, ser. LNCS, vol. 435. Springer, 1989, pp. 316–322.

[13] W.-G. Tzeng, "A Time-Bound Cryptographic Key Assignment Scheme for Access Control in a Hierarchy," IEEE Transactions on Knowledge and Data Engineering (TKDE), vol. 14, no. 1, pp. 182–188, 2002.

[14] G. Ateniese, A. D. Santis, A. L. Ferrara, and B. Masucci, "Provably-Secure Time-Bound Hierarchical Key Assignment Schemes," J. Cryptology, vol. 25, no. 2, pp. 243–270, 2012.

[15] R. S. Sandhu, "Cryptographic Implementation of a Tree Hierarchyfor Access Control," Information Processing Letters, vol. 27, no. 2, pp. 95–98, 1988.

[16] Y. Sun and K. J. R. Liu, "Scalable Hierarchical Access Control in Secure Group Communications," in Proceedings of the 23th IEEE International Conference on Computer Communications (INFOCOM'04). IEEE, 2004.

[17] Q. Zhang and Y. Wang, "A Centralized Key Management Scheme for Hierarchical Access Control," in Proceedings of IEEE Global Telecommunications Conference (GLOBECOM '04). IEEE, 2004, pp. 2067–2071.

[18] J. Benaloh, "Key Compression and Its Application to Digital Fingerprinting," Microsoft Research, Tech. Rep., 2009.

[19] B. Alomair and R. Poovendran, "Information Theoretically Secure Encryption with Almost Free Authentication," J. UCS, vol. 15, no. 15, pp. 2937–2956, 2009.

[20] 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.

[21] A. Sahai and B. Waters, "Fuzzy Identity-Based Encryption," in Proceedings of Advances in Cryptology - EUROCRYPT '05, ser. LNCS, vol. 3494. Springer, 2005, pp. 457–473.

[22] 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.

[23] F. Guo, Y. Mu, and Z. Chen, "Identity-Based Encryption: How to Decrypt Multiple Ciphertexts Using a Single Decryption Key," in Proceedings of Pairing-Based Cryptography (Pairing '07), ser. LNCS, vol. 4575. Springer, 2007, pp. 392–406.

[24] M. Chase and S. S. M. Chow, "Improving Privacy and Security in Multi-Authority Attribute-Based Encryption," in ACM Conference on Computer and Communications Security, 2009, pp. 121–130.

[25] T. Okamoto and K. Takashima, "Achieving Short Ciphertexts or Short Secret-Keys for Adaptively Secure General Inner-Product Encryption," in Cryptology and Network Security (CANS '11), 2011, pp. 138–159.

[26] R. Canetti and S. Hohenberger, "Chosen-Ciphertext Secure Proxy Re-Encryption," in Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS '07). ACM, 2007, pp. 185–194.

[27] C.-K. Chu and W.-G. Tzeng, "Identity-Based Proxy Re-encryption Without Random Oracles," in Information Security Conference (ISC '07), ser. LNCS, vol. 4779. Springer, 2007, pp. 189–202.

[28] C.-K. Chu, J. Weng, S. S. M. Chow, J. Zhou, and R. H. Deng, "Conditional Proxy Broadcast Re-Encryption," in Australasian Conference on Information Security and Privacy (ACISP '09), ser. LNCS, vol. 5594. Springer, 2009, pp. 327–342.

[29] S. S. M. Chow, J. Weng, Y. Yang, and R. H. Deng, "Efficient Unidirectional Proxy Re-Encryption," in Progress in Cryptology - AFRICACRYPT 2010, ser. LNCS, vol. 6055. Springer, 2010, pp. 316–332.

[30] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved Proxy Re-Encryption Schemes with Applications to Secure Distributed Storage," ACM Transactions on Information and System Security (TISSEC), vol. 9, no. 1, pp. 1–30, 2006.

[31] D. Boneh, C. Gentry, and B. Waters, "Collusion Resistant Broadcast Encryption with Short Ciphertexts and Private Keys," in Proceedings of Advances in Cryptology - CRYPTO '05, ser. LNCS, vol. 3621. Springer, 2005, pp. 258–275.

[32] L. B. Oliveira, D. Aranha, E. Morais, F. Daguano, J. Lopez, and R. Dahab, "TinyTate: Computing the Tate Pairing in Resource-Constrained Sensor Nodes," in Proceedings of 6th IEEE International Symposium on Network Computing and Applications (NCA '07).
IEEE, 2007, pp. 318–323.

[33] D. Naor, M. Naor, and J. Lotspiech, "Revocation and Tracing Schemes for Stateless Receivers," in Proceedings of Advances in Cryptology - CRYPTO '01, ser. LNCS. Springer, 2001, pp. 41–62.

[34] 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.

[35] D. Boneh, X. Boyen, and E.-J. Goh, "Hierarchical Identity Based Encryption with Constant Size Ciphertext," in Proceedings of Advances in Cryptology - EUROCRYPT '05, ser. LNCS, vol. 3494. Springer, 2005, pp. 440–456.

[36] 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.