Privileged Data Reclamation for Distributed Disruption - Tolerant Networks

J. Sandeep Kumar Reddy 1 & T. Kantha Raju 2
1 M.Tech (DSCE) Student, Dept. of ECE, BIT INSTITUTE OF TECHNOLOGY, Affiliated to JNTUA, Andhra Pradesh, India
2 Associate prof., Dept. of ECE, BIT INSTITUTE OF TECHNOLOGY, Affiliated to JNTUA, Andhra Pradesh, India

Abstract— Cipher text-policy attribute based encryption (CPABE) provides an encrypted access control instrument for propagation messages. Basically, a sender encrypts a message with an access control policy tree which is rationally composed of attributes; receivers are able to decrypt the message when their attributes satisfy the policy tree. A user’s attributes stand for the properties that he current has. It is necessary for a user to keep his attributes up-to-date. However, this is not easy in CP-ABE because whenever one attribute changes, the entire private key, which is based on all the attributes, must be changed. In this paper, we introduce fading function, which renders attributes “dynamic” and allows us to update each one of them separately. We study how choosing fading rate for fading function affects the efficiency and security. We also compare our design with CP-ABE and find our scheme performs significantly better under certain circumstance.

Keywords: Cipher Text Policy; Attribute Based Encryption

I. INTRODUCTION

Cautious landlords replace the house locks after tenants leave because they worry those tenants might keep copies of the keys. The same concept applies to protecting confidential information. Whenever a user leaves a communication group that has been exchanging and sharing confidential information, the remaining group members will replace the key used to encrypt the messages with a new one. However, given the high cost of key redistribution, this can impact performance especially when the group is made of thousands of users and the group members are possible to move in and out frequently. Sahai et al. [2], [3]’s recent Attribute Based Encryption (ABE) scheme makes it possible to dynamically reassign group keys when supplies and conditions change. To introduce the concept of ABE, consider the following example: There are often several limitations to redeem a coupon, say, California resident, UC or CSU students, plus AAA or UHaul membership, etc. One must illustrate resident ID, student ID and AAA or UHaul ID etc. to get the coupon. In the ABE context, the coupon is the object or information that we have to protect, and the IDs are so-called attributes. The secret message (the coupon) is encrypted with an access control policy tree that contains the logical grouping of the dissimilar attributes. The policy tree for the above coupon example would be “CA resident AND (UC student OR CSU student) AND (AAA membership OR UHaul membership)”. Each qualified user can apply and obtain a private key from certifying authority (Key Master). The key is connected with the a variety of qualifications (i.e., attributes) of the applicant. The users can decrypt only if the attributes gratify the policy tree. Attributes can be expanded to represent all kinds of properties correlated to applicants, e.g., skin color, car brand, size, occupation and time window when these properties are valid, etc.. A policy tree defines a target multicast group to which a secret must be delivered - for example, a group key to be used for future communications. ABE saves the trouble to issue a group key in advance to every foreseen multicast collection (thus avoiding combinatorial explosion). Or, equally, it avoids the problem (and latency) of judgment and certify all the qualified members on the spot whenever the need arises. ABE requires the customers to pre-qualify (off line) for the attributes that may communicate to multicast group they will be asked to join. Thus, the work is done ahead of time; and, it does not require combinatorial complexity. To save CPU resources, bandwidth and time, we avoid updating those attributes that stay unmoved. To accomplish this, we

Available online: http://internationaljournalofresearch.org/
introduce in this paper the concept of attribute fading function, making attributes “independent” and “dynamic”. With fading function, an attribute associated with a private key has its own expiration time. When an underlying property changes, the user requests a new attribute from the authority to represent his new property and the out-of-date attribute expires after a certain period of time. By this mean, a user can update partial attributes, rather than all of them, in one update. Our simulation results show that this approach significantly reduces the overhead comparing with traditional ABE especially when there are a number of “dynamic” attributes associated with users’ private keys. Our Contribution. We develop a much richer type of attribute-based encryption cryptosystem and demonstrate its applications. In our system each ciphertext is labeled by the encryptor with a set of descriptive attributes. Each private key is associated with an access structure that specifies which type of ciphertexts the key can decrypt. We call such a scheme a Key-Policy Attribute-Based Encryption (KP-ABE), since the access structure is specified in the private key, while the ciphertexts are simply labeled with a set of descriptive attributes. We note that this setting is reminiscent of secret sharing schemes (see, e.g., [4]). Using known techniques one can build a secret-sharing scheme that specifies that a set of parties must cooperate in order to reconstruct a secret. For example, one can specify a tree access structure where the interior nodes consist of AND OR gates and the leaves consist of different parties. Any set of parties that satisfy the tree can reconstruct the secret. In our construction each user’s key is associated with a tree-access structure where the leaves are associated with attributes. A user is able to decrypt a cipher text if the attributes associated with a cipher text satisfy the key’s access structure. The primary difference between our setting and secret-sharing schemes is that while secret-sharing schemes allow for cooperation between different parties, in our setting, this is expressly forbidden. For instance, if Alice has the key associated with the access structure “X AND Y”, and Bob has the key associated with the access structure “Y AND Z”, we would not want them to be able to decrypt a cipher text whose only attribute is Y by colluding. To do this, we adapt and generalize the techniques introduced by [34] to deal with more complex settings. We will show that this cryptosystem gives us a powerful tool for encryption with fine-grained access control for applications such as sharing audit log information

II.RELATED WORK

Fine-grained access control systems facilitate granting discrepancy access rights to a set of users and allow elasticity in specifying the admission rights of personality users. Several techniques are recognized for implementing fine grained access control. Common to the offered techniques are the fact that they employ a trusted server that stores the data in clear. Access control relies on software check to ensure that a user can access a piece of data only if he is authorized to do so. This location is not particularly appealing from a security standpoint. In the event of server compromise, for example, as a result of a software vulnerability exploit, the potential for information theft is immense. Furthermore, there is always a danger of “insider attacks” wherein a person having access to the server steals and leaks the information, for example, for economic gains. Some techniques create user hierarchies and require the users to share a common secret key if they are in a common set in the hierarchy. The data is then classified according to the hierarchy and encrypted under the public key of the set it is meant for. Clearly, such methods have several limitations. If a third party must access the data for a set, a user of that set either needs to act as an intermediary and decrypt all relevant entries for the party or must give the party its private decryption key, and thus let it have access to all entries. In many cases, by using the user hierarchies it is not even possible to realize an access control equivalent to monotone access trees. In this paper, we introduce new techniques to implement fine grained access control. In our techniques, the data is stored on the server in an encrypted form while different users are still allowed to decrypt different pieces of data per the security policy. This effectively eliminates the need to rely on the storage server for preventing unauthorized data access. Secret-Sharing Schemes. Secret-sharing schemes (SSS) are used to divide a secret among a number of parties. The information given to a party is called the share (of the secret) for that party. Every SSS realizes some access structure that defines the sets of parties who should be able to reconstruct the secret by using their shares. Shamir [35] and Blakley [7] were the first to propose a construction for secret-sharing schemes where the
access structure is a threshold gate. That is, if any \( t \) or more parties come together, they can reconstruct the secret by using their shares; however, any lesser number of parties do not get any information about the secret. Benaloh [6] extended Shamir’s idea to realize any access structure that can be represented as a tree consisting of threshold gates. Other notable secret-sharing schemes are [25, 15]. In SSS, one can specify a tree-access structure where the interior nodes consist of AND and OR gates and the leaves consist of different parties. Any set of parties that satisfy the tree can come together and reconstruct the secret. Therefore in SSS, collusion among different users (or parties) is not only allowed but required. In our construction each user’s key is associated with a tree-access structure where the leaves are associated with attributes. A user is able to decrypt a ciphertext if the attributes associated with a ciphertext satisfy the key’s access structure. In our scheme, contrary to SSS, users should be unable to collude in any meaningful way.

Identity-Based Encryption and Extensions. The concept of Attribute-Based Encryption was introduced by Sahai and Waters [34], who also presented a particular scheme that they called Fuzzy Identity-Based Encryption (FIBE). The Fuzzy-IBE scheme builds upon several ideas from Identity-Based Encryption [10, 36, 18]. In FIBE, an identity is viewed as a set of attributes. FIBE allows for a private key for an identity, \( \omega \), to decrypt to a ciphertext encrypted with an identity, \( \omega' \), if and only if the identities \( \omega \) and \( \omega' \) are close to each other as measured by the “set overlap” distance metric. We develop a much richer type of attribute-based encryption. The private keys of different users might be associated with different access structures. Our constructions support a wide variety of access structures (indeed, in its most general form, every LSSS realizable access structure), including a tree of threshold gates. Yao et. al. [19] show how an IBE system that encrypts to multiple hierarchical identities in a collusion-resistant manner implies a forward secure Hierarchical IBE scheme. They also note how their techniques for resisting collusion attacks are useful in attribute-based encryption. However, the cost of their scheme in terms of computation, private key size, and ciphertext size increases exponentially with the number of attributes. We also note that there has been other work that applied IBE techniques to access control, but did not address our central concern of resisting attacks from colluding users [37, 14].

III. BACKGROUND

A. Attribute Based Encryption Sahai and Waters et al. [2], [3] introduced Attribute Based Encryption (ABE) as a new mechanism for encrypted access control. There are several versions of ABE; the one discussed in this paper is so-called Cipher-Policy Attribute Based Encryption (CP-ABE) [1]. CP-ABE utilizes identity based encryption [7], [8] and threshold secret sharing scheme [6]. To some extent, CP-ABE is an extension of conventional PKI to groups: An authority generates public and private keys. Public key is for encryption while users keep their own private keys to decrypt. In CP-ABE, public and private keys are a not one-to-one pair, instead, there is only one public key and a potentially large number of private keys, one for each user in the target group. A user’s private key is associated with an arbitrary number of attributes. One attribute corresponds to one property. Properties such as name, ages and employers are different from person to person, so users have different attributes associated with their private keys. The publisher uses a policy tree, i.e., the logical combination of various attributes, and the public key to encrypt a message. Only clients with those attributes associated with their private keys satisfy the policy tree can decrypt the message. In Fig. 1, the policy tree is logically composed of five different attributes. Two users try to decrypt. Kevin has attributes CA resident, UC student and AAA member so that he can decrypt the cipher while Sarah cannot.

CCA SECURITY CONSTRUCTION

We now proceed to discuss the construction of the chosen ciphertext secure scheme. For IBE schemes, a common practice of constructing a CCA secure scheme from a CPA secure one is to generate one-time signature keys (K and sign the ciphertext with K with a strongly existentially unforgeable signature scheme, while K is viewed as the message receiver’s identity. This technique was proposed by Canetti, Halvi, and Katz [7]. In [9], Cheung and Newport applied the similar technique to CP-ABE and constructed a CCA secure CP-ABE scheme from the CPA secure one. Thereafter, these attributes are treated similarly as other normal attributes. For encryption, the encryptor chooses a pair (K) and encrypts the message with the attributes for K in addition to other normal attributes. The whole ciphertext is then signed with Ksv . The ciphertext along with the signature is sent to

Available online: http://internationaljournalofresearch.org/
receiver(s), who will verify the signature before decryption. In our work, it seems to be a contradiction to construct a CCA secure scheme since we on one hand require the ciphertext to be non-malleable, and on the other hand give the proxy re-key’s to proxy servers and allow them to reencrypt ciphertexts. However, in our scheme ciphertext re-encryption is just limited to updating partial ciphertext components to the latest version. Modification of the underlying message or the access structure is not permitted.

In terms of non-malleability, we just need to prevent adversaries from modifying the message or the access structure. Based on this observation, we adopt the same technique as [9] but just sign on partial ciphertext components.

B. Situation Aware Trust Xiaoyan Hong et al. developed Situation Aware Trust (SAT) to provide adaptive and proactive security in mobile scenarios like VANET. Attributes in SAT identify a group of entities (e.g., taxis associated with a company, police cars in a city), a type of events (e.g., accidents, congestions), or the property of

![Policy Tree and Private Keys](image)

Fig. 1. An example of Policy Tree and Private Keys associated with attributes

IV. DYNAMIC ATTRIBUTES

A. Key Insight

Here is another story from our acquaintance, landlord. The landlord owns a house with 5 rooms. At first, advertises the house for rent on the Craigslist. 5 people (they are friends to each other) come together and rent the house. They sign a one-year contract and each one has a copy. However, a month later, one tenant leaves and another friend of theirs moves in. The landlord renews the house renting contract with them to incorporate the new tenant’s name. Unfortunately, this group of people really like changing their home and several people move in and out in the following several months. Every time when a new tenant comes in, the landlord has to meet all the tenants and update their contracts. The tiring moving-in and out stuff finally drives the landlord crazy. He decides to adapt a new strategy - make the rooms for rent and sign room renting contract with each tenant. In this way, he gets rid of the trouble of meeting all the tenants when new tenants move in. This life case inspires us. Why do we need a separate expiration timestamp attribute (a house renting contract) for the private key (house)? Why not make each attribute (room) have its own expiration time (room renting contract)? If that works, whenever we need to update an attribute, we make the old attribute expired, and then request a new attribute from authority and add it into private key. The second step (i.e., adding a new attribute to the user’s private key) is so not difficult. But it is a challenge to remove (or revoke) an attribute that is already associated with a private key. This is because the party encrypting the message does not obtain the receiver’s certificate (attribute) online, and he is not able to check whether it is expired or not [1]. We do not want to assume that there is a tamper-proof “hardware”, so no one can force a user to drop his old attributes. The only way to make an old attribute useless is never to post it again in policy tree (when it is expired). Our scheme is designed under this direction and we will focus on how to expire an attribute in the following sections.

B. Design

Key Master is the authority in our system. It is responsible to formalize and maintain the set of attributes that can be used in policy trees. It publishes all the attribute values, that can be used to encrypt a message, and the corresponding descriptions.

Key Master is also responsible to generate keys, including public and private keys, as well as the attributes associated with private keys. Usually, for a given application, there is only one public key. Key Master generates private keys and attributes for users according to their requests. Key Master verifies each request for compliance before issuing the key to the user.

Sender is the individual that encrypts a message and sends it to others. Before encrypting a message, the sender looks up the desired attributes in the Key Master’s attribute list and downloads the corresponding attribute values. He downloads the public key as well. Then he encrypts a message as instructed in Section II-A and sends out the cipher.
Receiver receives ciphers and tries to decrypt. A receiver can get a private key before or after receiving ciphers. Generally, to enable “proactive trust” [5], a receiver is supposed to obtain a private key at the start of the day (or mission) and keep his private key up-to-date proactively. Lifetime of an attribute is a fixed time window during which an attribute’s value remains unchanged. We assume that all the parties in the system, including Key Master, Senders and Receivers, have coarsely synchronized time clocks using, for example, GPS or equivalent techniques.

V. CONCLUSION

This work is a follow-up of previous research on Situation Aware Trust. SAT uses descriptive attributes to efficiently perform security policy control. It also builds mechanisms for predicting future trust situations, and; it transforms trust from Internet social communities to VANET trust in order to enhance and promote VANET applications. ABE was used as the basis to integrate the policy control and security services for SAT.

One of SAT main contributions was to divide attributes into two categories, static and dynamic, in order to improve efficiency. However, this does not help much since updating is still in terms of (private) keys, instead of attributes. In this paper, we introduce the concept of a fading function to CP-ABE, allowing attributes “dynamic”. DABE provides an efficient mechanism for attribute revocation that does not require the reissuing of the entire key. Namely, attributes can be updated independently from entire private keys, making key management in presence of dynamic attributes much more efficient and scalable.

We are now in the way to finding out the DABE’s performance in the practical scenarios. Besides, we plan to investigate the feasibility of incorporating value compare predicates in policy tree [1] in the future so that the sender can control the lifetime of attributes.

VI. REFERENCES


Dr. T Kantharaju is currently working as an Associate Professor in ECE department, BIT Institute of Technology, Hindupur. He received B.Tech degree from Sri Venkateswara University, Tirupati, A.P. and M.E degree from University Visvesvaraya College of Engineering, Bangalore University, Bangalore, India. He received Ph.D in Wireless Networks from NWMD-International University. His research areas in Wireless Communication & Networking, Satellite Communication, Microwave Engineering, Embedded Systems, Image & Signal Processing, MANET’s and VANET’s. He is a life time member of ISTE, New Delhi, India. He is a life time member of The Institution of Engineers (INDIA), Kolkata, India. He has published many research papers in various National and International Journal and Conferences across the globe.