Accountable Attribute-Based Broadcast

Shucheng Yu[†], Kui Ren[⋆], and Wenjing Lou[‡]

{yscheng,wjlou}@wpi.edu, kren@ece.iit.edu

[†]Student, Department of ECE, Worcester Polytechnic Institute, MA 01609
[⋆]Faculty, Department of ECE, Illinois Institute of Technology, IL 60616
[‡]Faculty, Department of ECE, Worcester Polytechnic Institute, MA 01609

I. Introduction

In many broadcast applications, fine-grained access control over contents is required to provide differentiated services to users. For this purpose, the content provider may assign sets of attributes to the contents, and user access privileges are defined as logic expressions over these attributes. For example, in a digital video recorder (DVR) system, the content provider might broadcast episodes of TV shows and each of episode may be assigned a set of attributes such as *name*, *season number*, *genre*, so on and so forth. User access privileges can be encoded as policies such as ("*name=friends*" AND ("*season 2*" OR "*season 3*")). To enforce these access polices, the content provider needs to encrypt the media products using some cryptographic primitives since the contents might be distributed across third party content delivery networks (CDNs). Key-policy attribute-based encryption (KP-ABE) [1] is a cryptographic primitive that was proposed to resolve the exact problem. In KP-ABE, a ciphertext is associated with a set of attributes, and each user secrete key is embedded with an access structure defined over attributes. Users can decrypt a ciphertext if and only if the attributes associated with the ciphertext satisfy the access structures embedded in their secret keys.

However, in the current KP-ABE construction [1], it is possible that a paid user "shares" his secret key and abuses his access privilege without being identified. More seriously, pirates may take this advantage to make profits. In conventional broadcast encryption, this issue is addressed by using a technique called *traitor tracing* [2]–[5]. The key idea of traitor tracing is to enable the content provider to trace any suspicious pirate device and thus discover illegal key distributors' identities and collect evidences of the key abuse. Then the content provider can sue the illegal key distributors by presenting these evidences to law authorities. At a high level view, we can play the same trick in KP-ABE to defend against key abuse attacks. However, underlying techniques adopted by existing traitor tracing systems can not be directly applied to KP-ABE because receivers are represented individually in conventional broadcast encryption while not in KP-ABE. Therefore, it is desirable to propose a novel solution for defending against key abuse attacks in KP-ABE.

II. Our Method

Our method of tracing is to enable the content provider to trick pirate decoders into decrypting tracing ciphertexts which are designed in the way that only the suspected user is able to correctly decrypt. The content provider obtains the evidence of piracy if the pirate decoder correctly decrypts certain tracing ciphertext. To be able to trick pirate decoders into decrypting tracing ciphertexts, it requires that tracing ciphertexts are indistinguishable from normal (non-tracing) cihpertexts. Otherwise, the pirate decoder is able to detect the tracing activity and stop outputting anything. Keeping this in mind, we describe our construction as follows.

A. Background

To help understand our method, we first briefly introduce KP-ABE. In KP-ABE, attributes are defined as public key components. To encrypt a message with a certain set of attributes, the encryptor just picks out the corresponding public key components of these attributes and use them to encrypt the message. A user secret key is associated with an access structure which is a logic expression over attributes. A user is able to decrypt a ciphertext if and only if the set of attributes associated with the ciphertext satisfy the access structure embedded in his secret key. For example, in the aforementioned DVR case, the content provider can define public key components for attributes such as name, season number, etc. Then, a ciphertext encrypted with the attribute set {"name=hero", "season=2"} can not be decrypted by the user whose access policy is ("*name=friends*" AND ("*season 2*" OR "*season 3*")). Fig. 1 illustrates this example. Note that, in KP-ABE attributes associated with the ciphertext should be revealed so that decryptors are able to correctly combine them with their secret key components.

B. Main Idea

Recall that, to enable tracing the main task of our construction is to generate tracing ciphertexts that is indistinguishable from normal (non-tracing) ciphertexts. The intuition of our method can be summarized as the follows. We define a n-bit user identity space and each bit of them is defined as an attribute with two occurrences, one for bit value 0 and the other for bit value 1. We call these attributes by "*ID-related attributes*" and other attributes used by

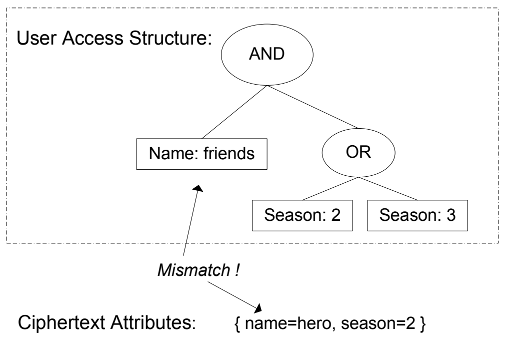

Fig. 1. Example of the DVR case

KP-ABE by "*normal attributes*". Each user is then assigned a unique ID from the identity space. In addition to normal attributes, the encryptor will also associate these ID-related attributes to the ciphertext in the following way: for normal (non-tracing) operations, all these n attributes are set as "don't care"; for tracing operations, they are set to represent the suspicious user's identity. In tracing operations, a user is able to decrypt the ciphertext only if his identity equals the suspicious one. To make tracing ciphertexts indistinguishable from normal ciphertexts, we hide these ID-related attributes in the way so that any user is not able to tell which and how many of them are set as "interested". In this way, we are able to make tracing ciphertexts indistinguishable from normal ciphertexts since the only difference between the two is on the usage of these ID-related attributes. We can hide these attributes by adopting similar techniques from the area of anonymous ciphertext-policy attribute-based encryption (CP-ABE) [6]. In addition, we also hide some normal attributes (we can use dummy attributes instead) so that upon a fail decryption the user can not tell if it is caused by the mismatch of his ID or by his access privilege (without considering his ID). Thus, he is not able to distinguish a tracing activity from a normal (non-tracing) one. Fig. 2 illustrates the previous DVR example using our method.

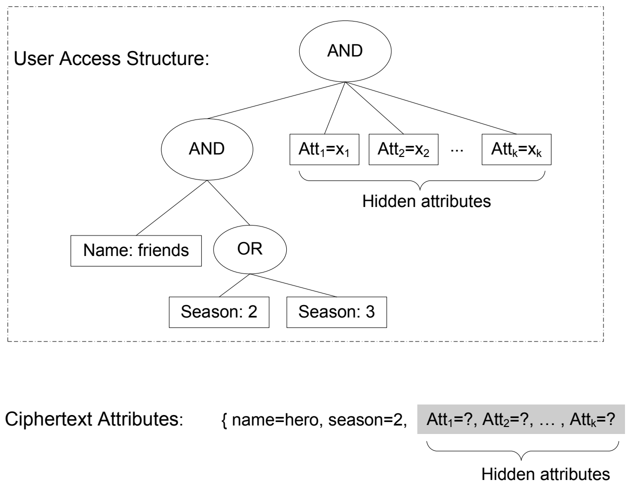

Fig. 2. Example of the DVR case using hidden attributes

Tracing To trace a pirate device and identify the guilty users, the content provider checks the user identity list of the system and generates tracing ciphertexts for each identity in the list. Then, he feeds these ciphertexts into the pirate device one by one. Because the pirate device does not know whether the ciphertexts are for tracing or just for normal content distribution, it decrypts them and outputs whatever it gets. The content provider just compares the pirate device's output with the original message. Once the pirate device outputs a correct message, the content provider adds the user's identity to the guilty user list. Finally, the content provider will obtain the list of all the guilty users.

C. Discussion

Efficiency In our design, both the ciphertext size and the secret key size are linear to n, where n is the number of bits in the identity space. As the maximum number of users it can represent is $N = 2^n$, the complexity can be written as $\mathcal{O}(logN)$, where N is the total number of users. To trace a pirate, the content provider needs to try with every user's identity in the system list. When the number of users in a system is large, the tracing algorithm would be inefficient. To resolve this issue, we can first test with some normal ciphertexts using combinations of normal attributes. For example, we can use different combinations of attributes like location, age, etc. In practice, this process will hopefully rule out a significant portion of users. Our tracing algorithm can just test over the remaining set of users.

Application Scenarios and Future Work In general, our proposed scheme is applicable to systems where 1) data can be categorized by their attributes and a user access privilege should be defined in the way that just allows the user to access certain intended subset of resources; 2) abuse of the access privilege should be prohibited. Applications of this kind can be found in "targeted broadcast", audit log, and etc. One important future work is to provide formal security proof to our construction. In addition, our current construction can just defend against partially colluding users. In the future, we will work on the case of arbitrary colluding users.

REFERENCES

[1] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *CCS '06*, 2006, pp. 89–98.

[2] B. Chor, A. Fiat, and M. Naor, "Tracing traitors," in *CRYPTO'94*. London, UK: Springer-Verlag, 1994, pp. 257–270.

[3] D. Boneh and M. K. Franklin, "An efficient public key traitor tracing scheme," in *CRYPTO'99*. London, UK: Springer-Verlag, 1999, pp. 338–353.

[4] A. Kiayias and M. Yung, "Traitor tracing with constant transmission rate," in *EUROCRYPT'02*. London, UK: Springer-Verlag, 2002, pp. 450–465.

[5] D. Boneh, A. Sahai, and BrentWaters, "Fully collusion resistant traitor tracing with short ciphertexts and private keys," in *EUROCRYPT'06*. London, UK: Springer-Verlag, 2006.

[6] T. Nishide, K. Yoneyama, and K. Ohta, "Attribute-based encryption with partially hidden encryptor-specified access structures," in *ACNS'08*. LNCS 5037, 2008, pp. 111–129.