Towards Key-Privacy in a Fuzzy Identity-Based Encryption Scheme

Jonathan Kirsch
Advanced Cryptographic Protocols (600.642)
Professor: Giuseppe Ateniese
Johns Hopkins University

Spring 2005

1 Introduction

As the use of computers in commercial and medical transactions increases, there is a growing need to employ cryptographic mechanisms for achieving data privacy and security. The Fuzzy Identity-based Encryption scheme, recently proposed by Sahai and Waters in [3], is one such mechanism. The Fuzzy IBE scheme has several advantages over traditional public key encryption schemes, such as RSA or El Gamal, because it supports the use of biometric input as a user's public key. In particular, it allows one user to encrypt for another without the latter possessing *a priori* keying information: the latter's biometric data *is* its public key, resulting in a scheme that is both convenient and practical in situations where the receiver is "unprepared."

One shortcoming of the Fuzzy IBE scheme as originally presented, however, is that it does not achieve *key privacy* [2]; namely, it reveals the identity (public key) used in the encryption. Moreover, the decryption operation relies on the fact that the ciphertext contains this identity. In many applications, revealing the receiver's identity – or a variant of the receiver's identity – may be unacceptable.

This is especially true in medical applications. Suppose, for example, that the Fuzzy IBE scheme is used by a hospital to encrypt the results of sensitive data, such as the results of an HIV or pregnancy test. While the security of the scheme protects the contents of the results, it does not prevent a passive attacker from learning that the encryption took place and was intended for a certain patient. If it is known that the hospital places all test results on a particular server, then the attacker can imply private information about the patient just by examining the ciphertext.

This essay explores the challenges in achieving key privacy in the Fuzzy IBE setting, and it presents one possible solution to the problem. The remainder of this essay is presented as follows. In Section 2, we provide background on the protocols used in constructing the solution. Section 3 presents the integrated protocol. Section 4 provides a security analysis, while Section 5 provides discussion and concludes.

2 Related Work

2.1 Fuzzy Identity-Based Encryption

The Fuzzy Identity-based encryption scheme presented in [3] supports the use of biometric input as keying information by using threshold cryptography to overcome the noise inherent in biometric readings. A user's *identity* consists of a set of biometric attributes; the user receives a private key for each attribute in its identity. The system ensures that a user with identity ω can decrypt a message encrypted for identity ω' only if $|\omega \cap \omega'| \geq d$, where d is a security parameter. Since the private keys are tied to unique, randomly-generated polynomials, two users cannot collude to decrypt a message neither could decrypt on its own. We review the Fuzzy IBE scheme below (see [3] for a complete description).

2.1.1 System Setup

The system defines a bilinear group $\mathbb{G}_1$ of order p with generator g, and a bilinear mapping: $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$.

The initial system setup consists of choosing a public attribute key, $T_i = g^{t_i}$, for randomly chosen t_i, for each attribute in the universe of elements. The public system key is then defined as $Y = e(g, g)^y$ for a randomly chosen y.

2.1.2 Key Generation

Each user is associated with a randomly chosen polynomial q of degree $d-1$ where $q(0) = y$. A user with identity ω is then assigned a private attribute key for each attribute in $\omega : D_i = g^{\frac{q(i)}{t_i}}$

2.1.3 Encryption

To encrypt a message $m \in \mathbb{G}_2$ for identity ω', the encrypter chooses a random value $s \in \mathbb{Z}_p$ and computes:

$$C = (\omega', E' = mY^s, E = \{E_i = T_i^s\} \forall_i \in \omega')$$

Thus, the ciphertext includes the identity ω', an El Gamal-like encryption of the message, and a set of randomized public attribute keys, one for each attribute in the identity of the intended receiver.

2.1.4 Decryption

To decrypt, a receiver with identity ω chooses a d-element subset S from $\omega \cap \omega'$, and applies the bilinear mapping to pairs consisting of its own private attribute key and the randomized public attribute key, E_i, for each element of S:

$$
\begin{aligned}
E' & / \prod_{i \in S} (e(D_i, E_i))^{\Delta_{i,S}(0)} \\
= & \; me(g,g)^{ys} / \prod_{i \in \omega \cap \omega'} e(g^{\frac{q(i)}{t_i}}, g^{st_i}))^{\Delta_{i,S}(0)} \\
= & \; me(g,g)^{ys} / \prod_{i \in \omega \cap \omega'} (e(g,g)^{sq(i)})^{\Delta_{i,S}(0)} \\
= & \; m
\end{aligned}
$$

We observe the following key point regarding decryption:

- Let $\omega' = a_1 a_2 \ldots a_k$ and let $E = \{E_1, E_2 \ldots E_k\}$. Then the randomized public attribute key corresponding to a_j will be E_j for $1 \leq j \leq k$.

In other words, for a given element $v \in \omega \cap \omega'$, the receiver uses the *index* of v, as it appears in ω', to extract the corresponding public attribute key from E. As we will see, determining this index is one of the challenges in achieving a key-private version of the protocol, since ω' cannot be sent in the clear.

2.2 Key Privacy

As defined in [2], *key-privacy* is the property by which an adversary, in possession of a given ciphertext and a set of known public keys, is unable to determine the key with which the ciphertext was created. Put another way, the receiver is *anonymous* from the adversary's point of view.

Although El Gamal encryption (on which the Fuzzy IBE encryption is based) has been shown to achieve key-privacy under chosen plaintext attack, the Fuzzy IBE encryption itself does not. There are two reasons the Fuzzy IBE ciphertext is not key-private:

1. The ciphertext includes ω', the identity used in the encryption, which must (to allow decryption) match the intended receiver's identity on at least d attributes.

2. The size of E, the set of randomized public attribute keys, varies according to the number of attributes in ω'. Given two public keys of unequal length, it is easy for the adversary to determine which key was used to create a particular ciphertext by examining the size of E.

We will overcome the second problem by padding the set E with "dummy" values that cannot be used in the decryption process; these values are used solely to standardize the necessary fields of the ciphertext.

Once the length of the ciphertext is standardized, we argue that key privacy can be achieved by removing ω'. Unfortunately, this modification makes decryption inefficient, since the receiver uses ω' to construct the d-element overlap subset S, and to match the elements in S to their corresponding public attribute keys. Therefore, some modifications are needed to allow the receiver to securely and efficiently perform these tasks.

The Equijoin protocol of [1], which we presently describe, will allow us to achieve these goals.

2.3 Secure Set Intersection

We present the protocol from a high-level perspective; see [1] for a more formal presentation.

The Equijoin protocol is run interactively by two parties, Server and Client, each with its own data set (VS and VC, respectively). The protocol serves two main functions. First, it allows the Client to determine the intersection between the two data sets, while learning nothing of those elements $v \in VS \setminus VC$. Second, the protocol allows the Server to encrypt extra information for each element such that the Client can only decrypt the information for those elements in $VS \cap VC$.

The protocol makes use of several building blocks. The first is a commutative encryption function F. Informally, this function has the property that that we can compose two encryptions (with two different keys) and obtain the same ciphertext, regardless of the order in which the encryptions were applied. The second building block is a cryptographic hash function h. Finally, the protocol requires a second encryption function K which can be efficiently inverted and the output of which appears random.

The basic idea behind the protocol is that the Server and the Client can securely compare their sets by comparing encrypted hashes of the elements in the sets. Due to the nature of the commutative encryption function, the following property holds for each element v in the intersection:

$$F_S(F_C(h(v))) = F_C(F_S(h(v)))$$

The Server encrypts the extra information associated with each element, *ext(v)*, by encrypting it with the second encryption function K. The trick is that the Server generates a different encryption key K_i for each element such that the key can be recovered by the Client only for those elements in the intersection. The Client then uses these keys to obtain $ext(v)$.

3 An Integrated Protocol

3.1 Intuition

The integrated protocol combines the Equijoin protocol with the original Fuzzy IBE scheme to achieve a key-private, fuzzy identity-based encryption scheme. Upon completing the protocol, the Client is able to answer two key questions:

1. Which attributes from my own identity ω should be used in the decryption?
2. Which of the randomized public attribute keys E_i from the ciphertext correspond to these attributes?

We integrate the Equijoin protocol with the Fuzzy IBE scheme in a straightforward way. We first provide a padding mechanism to standardize the size of the sets (identities) used in the protocol. The encrypter (the Server) then chooses its set VS to be ω', the (padded) identity used in the encryption, while the receiver (the Client) chooses its set VC to be ω,

a (padded) fresh biometric reading. The Server encodes the mappings between attributes in ω' and their corresponding public attribute keys as *ext(v)* for each $v \in VS$. We modify the Equijoin protocol slightly, leveraging the ability to securely compute the size of the set intersection, to ensure that the protocol only proceeds for clients whose sets overlap with the Server's set on at least d attributes.

3.2 Phase 1: Padding the Identities

We distinguish between *valid* attributes and *padding* attributes. Valid attributes are those attributes for which a client may be given a private key component, D_i. Padding attributes, on the other hand, are those used solely to achieve key privacy; they mask the true number of valid attributes in a given identity.

Let n be the maximum number of valid attributes that may appear in an identity, and let d be the threshold security parameter used by the Fuzzy IBE scheme. It should be clear that all identities must have at least d valid attributes for decryption to be possible. The following padding mechanism forces those identities of size k, where $d \leq k < n$, to "behave" as if they were of size n:

- We reserve $(n-d)$ padding attributes, $pad_1, pad_2 \ldots pad_{n-d}$, and their associated public attribute keys. For simplicity, we choose the first $(n-d)$ integers from $\mathbb{Z}_p$ as the padding attributes, with associated public keys T_1 through T_{n-d}.
- Let $\omega = a_1 a_2 \ldots a_k$. Then in the protocols below, append the first $(n-k)$ padding attributes to ω to obtain $\omega' = a_1 a_2 \ldots a_k pad_1 pad_2 \ldots pad_{n-k}$

3.3 Phase 2: Setup

3.3.1 Server Key Generation

When the Server takes a Client's biometric reading, it pads it, if necessary, to obtain ω', a reading of size n. The Server then generates two keys: a main key, ES, and an Index Encryption key, ES', and derives n *index keys* K_i as follows:

$$\begin{aligned} VS &= \omega' \\ K_i &= F_{ES'}(h(VS_i)), 1 \leq i \leq n \end{aligned}$$

These index keys will be used to encrypt $ext(VS_i)$, the index of the public attribute key in the ciphertext corresponding to VS_i.

3.3.2 Client Key Generation

The Client generates one main key, EC. It takes a fresh biometric reading and pads it, if necessary, to obtain a reading of size n, ω. The Client constructs its encrypted set YC as follows:

$$\begin{aligned} VC &= \omega \\ YC_i &= F_{EC}(h(VC_i)), 1 \leq i \leq n \end{aligned}$$

We state explicitly that we do *not* modify the key generation phase of the Fuzzy IBE scheme. In particular, Clients do not receive private key components for the padding attributes so that these attributes cannot be used during decryption.

3.3.3 Ciphertext Publication

The Server encrypts the hashed elements of its set to construct the set YS as follows:

$$YS_i = F_{ES}(h(VS_i)), 1 \leq i \leq n$$

The Server reorders both the set YS and the set of public attribute keys, E, lexicographically, to avoid leaking information about them [1]. The Server then publishes the ciphertext as:

$$C = (YS, mY^s, E = \{E_i = T_i^s\} \forall_i \in \omega')$$

Notice that the ciphertext includes an encrypted representation of ω', rather than the identity itself. Note also that the size of the set E has been standardized to n, and that the randomization of the public attribute keys protects the attributes used via the discrete logarithm.

The Server stores pairs of the following form:

$$<(YS_1, index\_1), (YS_2, index\_2), ...(YS_n, index\_n)>$$

where $index\_i$ is the index in the ciphertext of the randomized public attribute key corresponding to YS_i.

3.4 Phase 3: Overlap Verification

In the third phase of the protocol, the Server verifies that the Client's identity, ω, intersects with ω', the identity used in the encryption, on at least d *valid* attributes. The Server learns the intersection size, and proceeds with the protocol only if it is sufficiently large.

First, the Client retrieves the Server's set YS from the ciphertext. The Client then encrypts each element in this set with its own key, to obtain:

$$Z_{CS} = F_{EC}(YS) = F_{EC}(F_{ES}(h(VS)))$$

The Client sends this set, along with its own encrypted set YC, back to the Server, reordering both lexicographically:

$$C \rightarrow S : < YC, Z_{CS} >$$

Upon receiving this message, the Server encrypts each element in the Client's set YC with its own key, to obtain:

$$Z_{SC} = F_{ES}(YC) = F_{ES}(F_{EC}(h(VC)))$$

Due to the nature of the commutative encryption function, the Server can determine the intersection set size by tallying the number of elements that appear in both Z_{SC} and Z_{CS}. However, to avoid "giving credit" to the Client for any padding attributes that may overlap, the Server computes the size of the overlap as:

$$overlap = (num\_matches) - (num\_padding\_attributes\_used)$$

We assume the fuzziness in biometric readings changes the values of particular attributes, not the number of attributes, for a particular Client. Without this assumption, a Client's reading could contain fewer than d attributes, making decryption impossible. We note, however, that this assumption can be removed if we modify the protocol to allow the Server to compute the intersection itself, instead of just the size of the intersection. The Server would then make sure that there are enough valid attributes in $\omega \cap \omega'$ to proceed.

3.5 Phase 4: Server Response

Assuming that the *overlap* computed in Phase 3 was at least d, the Server constructs n triples, one for each element in the Client's encrypted set YC:

$$< YC_i, F_{ES}(F_{EC}(YC_i)), F_{ES'}(YC_i) >$$

Thus the Server encrypts each element of the Client's encrypted set with both of its keys. Note that the middle element of each triple was already computed in Phase 3 of the protocol.

The Server next uses its index keys K_i to encrypt the index information – stored in the Setup phase – for each element in YS:

$$C_i = K_{K_i}(index\_i)$$

The Server sends these triples and pairs to the Client:

$$\begin{array}{lcl} S \rightarrow C & : & < YC_i, F_{ES}(F_{EC}(YC_i)), F_{ES'}(YC_i) > \\ S \rightarrow C & : & < (YS_i, C_i) > \end{array}$$

3.6 Phase 5: Client Calculation

Upon receiving these two messages from the Server, the Client now has everything it needs to compute the intersection $\omega \cap \omega'$ and the corresponding indices.

First, the Client decrypts the elements of each of the n triples to obtain new triples of the following form:

$$<h(VC_i), F_{ES}(h(VC_i)), F_{ES'}(h(VC_i))>$$

Any element that appears in $VS \cap VC$ will appear as the second element in one of the new triples and as the first element in one of the pairs received in Phase 4, because the corresponding elements VC_i and VS_i must have been equal.

For each such element, the Client replaces $h(VC_i)$ (the first element of the triple) with the element $v \in VC$ that hashes to that value. v is the actual attribute in the identity; the Client will use its private key D_j that corresponds to v as one half of the bilinear mapping. If v is one of the padding attributes, it is ignored, since it will not aid the Client in the decryption process.

The last thing the Client must do is determine the index of the public attribute key in the ciphertext that corresponds to attribute v. Recall that the Server encrypted the index with the following key:

$$K_i = F_{ES'}(h(VS_i))$$

Then the third element in each triple, $F_{ES'}(h(VC_i))$ will match the key K_i when $VC_i = VS_i$. Thus, the Client learns the index key for those elements appearing in the intersection. It then uses the index key to decrypt the corresponding C_i (which appears as the second element in the pair) to determine the correct index.

The Client uses the public attribute key with this index as the second half of the bilinear mapping. Once it has performed this calculation for d attributes, it can perform the Fuzzy IBE decryption operation in the normal way.

4 Analysis

The Equijoin protocol was shown to be secure in an "honest but curious" setting, and we therefore assume this model for the interactive portion of the protocol. The security follows directly from the security of the Equijoin protocol.

4.1 Ciphertext Key Privacy

We argue that the ciphertext published in the integrated protocol is key-private. Recall that the ciphertext is of the following form:

$$C = (YS, mY^s, E = \{E_i = T_i^s\} \forall_i \in \omega')$$

We first consider the set E. Due to the padding mechanism used, E will always contain n elements. Each element in E is a randomization of one of the public attribute keys, g^{t_i}. Since the randomization element s is changed each time, the adversary is unable to determine which attributes keys are being used. In particular, the keys corresponding to padding attributes will be indistinguishable from the keys corresponding to valid attributes. Note that this holds only when all attributes in a given identity are distinct (which must be the case), since otherwise identical elements would appear multiple times in E.

We next consider the set YS. Recall that YS is constructed as:

$$YS_i = F_{ES}(h(VS_i)), 1 \leq i \leq n$$

Then YS is a random encryption on $h(VS)$. The security of the encryption function prevents the adversary from learning which elements appear in VS. Therefore, YS does not reveal anything about the identity of the intended receiver.

5 Discussion

One of the drawbacks of the protocol described above is that it is interactive: it requires two rounds of additional communication overhead to perform decryption. While this cost may be prohibitive in some instances, we believe the added privacy outweighs a slight increase in transaction latency for sensitive applications, such as retrieving medical results from a hospital database.

There are several other attractive features of the integrated protocol. First, it fits the Fuzzy IBE model in that it does not require additional public key infrastructure. Second, the Server does not need to store the Client's identity after generating the ciphertext, limiting the window of vulnerability to compromise. Furthermore, the Client's communication is secured by its own private keys; it need not rely on the security of a shared key with the Server.

One interesting open question is whether or not it is possible to achieve a key-private version of the Fuzzy IBE scheme that is non-interactive. Such a scheme would result in increased efficiency and reduced latency for each transaction.

References

[1] R. Agrawal, A. Evfimievski, and R. Srikant. Information sharing across private databases, 2003.

[2] Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. Key-privacy in public-key encryption. In *ASIACRYPT '01: Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security*, pages 566–582, London, UK, 2001. Springer-Verlag.

[3] Amit Sahai and Brent Waters. Fuzzy identity based encryption.