Verifiable Attribute-Based Encryption

Qiang Tang Dongyao Ji
State Key Lab of Information Security
Graduation University of Chinese Academy of Science
Beijing 100049
qtang84@gmail.com dyji@gucas.ac.cn

Abstract: In this paper, we construct two verifiable attribute-based encryption schemes. One is for a single authority ABE, and the other is for a multi authority ABE. Not only our schemes are proved secure as the formal schemes, they also provide a verifiable property which allows the user to check the correctness of the keys immediately he got them without decrypting out a wrong message.

Keywords: Attribute-based encryption, verifiable, multi authority ABE

1 Introduction

Identity based encryption (IBE), introduced by Shamir [Sha85], is a novel encryption which allows users to use any string as their public key (for example, an ID card number or an email address). Encrypting messages without access to a public key certificate reduces the load of creating and storing certificates. The first provably secure and elegantly designed IBE scheme was given by Boneh and Franklin [BF01], after that, IBE has received a lot of attention.

To better express identity and allow for a certain amount of error-tolerance, Sahai and Waters proposed fuzzy IBE [SW05], in their scheme, identity is viewed as a set of descriptive attributes, and a user with the secret key for the identity ω is able to decrypt a ciphertext encrypted with the public key ω' if and only if ω and ω' are with a certain distance of each other as judged by some metric.

In the paper [GPSW06], Goyal et al. developed a much richer type of attribute-based encryption cryptosystem and demonstrated its applications. In their 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. The access policy in their work is described by an access tree, which is more general than simple t-out-of-n threshold, and thus well suits for fine-grained access control of encrypted data and some other kind of applications.

Both the schemes in [SW05] and [GPSW06] are with single authority, so Chase presented multi-authority ABE in [Cha07] to answer an open question in [SW05], in multi-authority scenario, more than one authority are responsible for maintaining one kind of attributes, they operate simultaneously, and handle out secret keys for different set of attributes.

OUR CONTRIBUTIONS: the ideas in fuzzy IBE and its extension GPSW ABE are from secret sharing, we could not be sure that all the shares sent by the key generation center are consistent (being consistent means the shares can be used to reconstruct the same secret), maybe the key generation center is not that impartial, or something wrong happens in the process of creating or the sending period of the information is some secret shares, and in this kind of encryption schemes, one key can be used to decrypt many pieces of ciphertext, so changing secret sharing with verifiable secret sharing adds a verification property allowing the user to verify whether the share is consistent with other shares and whether the key the user got is rightly shared from the true secret which means using the key can decrypt, if not, we require the key generation center to resend them, doing this can reduce the meaningless computation cost of decrypting with wrong shares. Our work makes the GPSW ABE be verifiable before reconstructing the secret and check the deciphered text, we also realize the same function in the multi-authority ABE which also needs to make sure all the shares are right, or else, the user could not decrypt because the sharing among the authorities is not a threshold one, all the shares are used in reconstruction. The security of these schemes have not been influenced, we only need to make some modifications that computing the values for verification to answer the new queries in the proof in [GPSW06] and [Cha07] to finish our proof.

CHLLENGES AND OUR METHOD: because the secrets in the sharing schemes in the GPSW ABE are shared more than one time, we should make sure that there is nothing wrong in every step of sharing, so it is very difficult to finish all the process of verifying the rightness of the shares in one step, unless we can compute the equation of the secret and the shares in the leaf nodes beforehand, thus, we adopt a compromised method that we verify the shares we will use in the next step, if it's affirmative, and then, we verify the result for the next step of computing, repeat until to the top node, if any verification returns "negative", we stop computing and require the key generation center to resend the keys.

2 Preliminaries

2.1 Bilinear maps

Let G_1 and G_2 be two multiplicative cyclic groups of prime order p. Let g be a generator of G_1 and e be a bilinear map, $e: G_1 \times G_1 \rightarrow G_2$. The bilinear map e has the following properties:

1. Bilinearity: for all u,v $\in G_1$ and $a,b \in Z_p$, we have $e(u^a, v^b) = e(u^b, v^a) = e(u,v)^{ab}$
2. Non-degeneracy: $e(g,g) \neq 1$.
3. Efficiently computable.

2.2 The decisional bilinear Diffie-Hellman (BDH) assumption

Let a,b,c,z $\in Z_p$ be chosen at random and g be a generator of G_1. The decisional BDH assumption is that no probabilistic polynomial-time algorithm $\Im$ can distinguish the tuple (A=g^a, B=g^b, C=g^c, $e(g,g)^{abc}$) from the tuple (A=g^a, B=g^b, C=g^c, $e(g,g)^z$) with more than a negligible advantage. The advantage of $\Im$ is:

| Pr[$\Im$(A,B,C, $e(g,g)^{abc}$)=0]-Pr[$\Im$(A,B,C, $e(g,g)^z$)=0] |, where the probability is taken over the random choice of the generator g, the random choice of a,b,c,z in Z_p, and the random bits consumed by $\Im$.

2.3 definition of success in verification

We adopt the definition in [Ped91]and make a little modification, if both of the condition below are satisfied, then we say the verification is succeed.

1. If all the shares are right, the user could reconstruct the secret with probability 1, which means the user could decrypt.

2. For two authorized sub access structures $\Gamma_1 and \Gamma_2$ of the original access structure Γ, both satisfy $\Gamma_i(\gamma)=1$, then two messages reconstructed from each structure s_1 and s_2, we have $s_1 = s_2$.

[note]: the condition2 means in one round of sharing, every group of authorized shares can reconstruct the same secret, then further means, all shares come from the same polynomial, here, this means they also decrypt the right plaintext.

2.4 security model for verifiable ABE

Our security model only need make a little modification of the selective-set model in [GPSW06],

Init: The adversary declare the set of attributes, γ, that he wishes to be challenged upon.

Setup: The challenger runs the SETUP algorithm of GPSW ABE and gives the public parameters to the adversary.

Phase1: The adversary is allowed to issue queries for verification information and private keys for many access structure Γ_j, where $\Gamma_j(\gamma) \neq 1$ for all j, and the adversary checks the correctness of the keys.

Challenge: The adversary submits two equal length messages M_0 and M_1. The challenger flips a random coin b, and encrypts M_b with γ. The ciphertext is passed to the adversary.

Phase2: Phase1 is repeated.

Guess: The adversary outputs a guess b’ of b.

The advantage of an adversary $\Im$ in this game is defined as Pr[b’=b]- 1/2

This model can be easily extended to handle chosen-ciphertext attacks by allowing for decryption queries in Phase1 and Phase2, and a scheme secure in this model is also easily be extended to be secure in chosen-ciphertext model using simulation sound NIZK proofs which presented in [Sa99]. A GPSW ABE scheme is secure in the selective-set model of security if all polynomial time adversaries have at most a negligible advantage in the selective-set game.

2.5 Basic algorithms of GPSW ABE

The GPSW ABE scheme consists of four algorithms:

SETUP: This is a randomized algorithm that takes no input other than the implicit security parameter. It outputs the public parameters PK and a master key MK

ENCRYPTION: This is a randomized algorithm that takes as input a message m, a set of attributes γ, and the public parameters PK. It outputs the ciphertext E.

KEY GENERATION: This is a randomized algorithm that takes as input- an access structure Γ, the master key MK and the public parameter PK. It outputs a decryption key D.

DECRYPTION: This algorithm takes as input- the ciphertext E that was encrypted under the set γ of attributes, the decryption key D for access control structure Γ and the public parameter PK. It outputs the message M if $\gamma \in \Gamma$ (or $\Gamma(\gamma) = 1$).

2.6 The algorithms of Chase Multi Authority ABE and its security model

A Multi Authority ABE scheme is composed of K attribute authorities and one central authority, the scheme uses the following algorithms:

SETUP: A randomized algorithm which must be run by some trusted part (e.g CA). Takes as input the security parameter. Outputs a public key, secret key pair for each of the attribute authorities, and also outputs a system public key and master secret key which will be used by the central authority.

ATTRIBUTE KEY GENERATION: A randomized algorithm run by an attribute authority. Takes as input the authority secret key, the authority's value d_k, a user's ID, and an access structure Γ_C^k. Output secret key for the user.

CENTRAL KEY GENERATION: A randomized algorithm run by the central authority. Take as input the master secret key and a user's ID and outputs secret for the user.

ENCRYPTION: A randomized algorithm run by a sender. Takes as input a set of attributes for each authority, a message, and the system public key. Outputs the ciphertext.

DECRYPTION: A deterministic algorithm run by a user. Takes as input a ciphertext, which was encrypted under attribute set A_c. Output a message M if Γ_C^k (A_c)=1 for all authorities k.

The security model of Chase Multi Authority ABE is almost the same as the model mentioned before, only some little modifications in the Phase1 are needed. The requirements in Phase1 are that for each ID, there must be at least one authority k whose access structure denies giving decryption key for the challenge ciphertext, and the adversary never queries the same authority twice with the same ID. A multi authority ABE scheme is selective-set secure if there exists a negligible function ν such that, in the above game any adversary will succeed with probability at most 1/2+ν (t), t is a parameter.

3 Verifiable version of GPSW Complex Access Structure ABE

3.1 Our construction

We first describe the tree structure used in this scheme, the access tree Γ, each non-leaf node represents a threshold gate, described by its children and a threshold value, a node x, has num_x children and a k_x threshold value, changing k_x can make the node to represent both OR gate and AND gate. We also define the function parent(x) to return the parent node of x, and index(x) to return the index of x as a child of its parent. a leaf node x is defined by an attribute att(x).

Denote by Γ_x the subtree rooted at the node x, we compute Γ (γ) in a recursive manner:

If x is a leaf node, Γ_x (γ) returns 1 if and only if att(x)$\in \gamma$

If x is a non-leaf node, evaluate $\Gamma_{x'}(\gamma)$ for all children x' of node x, $\Gamma_x(\gamma)$ returns 1 if and only if at least k_x children returns 1.

Now we demonstrate the construction as follows:

Let G_1 be a bilinear group of prime order p, and let g be a generator of G_1. In addition, let $e: G_1 \times G_1 \to G_2$ denote the bilinear map. A security parameter, k, will determine the size of the groups. We also define the Lagarange coefficient $\Delta_{i,S}$ for $i \in Z_p$ and a set S, of elements in Z_p: $\Delta_{i,S}(x) = \prod_{j \in S, j \neq i} \frac{x-j}{i-j}$. We will associate each attribute with a unique element in Z_p^*.

Setup Define the universe of attributes U= {1, 2, … , n}. Randomly choose $t_1, ..., t_n, y$ from Z_p. The published public parameters PK are $T_1 = g^{t_1}, ..., T_{|U|} = g^{t_{|U|}}, Y = e(g,g)^y$. The master key MK is: $t_1, ..., t_n, y$.

Encryption (M, γ, PK) To encrypt a message M$\in G_2$ under a set of attributes γ, choose a random number s$\in Z_p$ and publish the ciphertext as : E= (γ, E'=MY^s, $\{E_i = T_i^s\}_{i \in \gamma}$).

Key Generation (Γ, MK) this process shares the secret y in a top-down manner with Shamir's threshold secret sharing scheme , for each non leaf node x, we choose a polynomial q_x with degree $d_x = k_x - 1$, make the polynomial satisfy $q_x(0) = q_{parent(x)}(index(x))$, and randomly fix other d_x points to completely define q_x, then compute $h_x = e(g,g)^{q_x(0)}$ and $C_x : \{e(g,g)^{a_i}\}_{i=1..k_x-1}$, { a_i } are the non constant coefficients of the polynomial used to share the secret of the node x. After all the polynomials are decided, for each leaf node x, we give the following set of secret values D to the user: $D_x = g^{q_x(0)/t_i}$, where i= att(x) and $h_x = e(g,g)^{q_x(0)}$. This process enables the user to decrypt a message encrypted under a set of attributes γ if and only if $\Gamma(\gamma)$ =1.

Remark: This process can be seen as the secret y is shared as $y_1, ..., y_k$, each y_i then is shared as $y_{i1}, .., y_{it}$, and this sharing repeats till the leaf nodes.

Verification （Γ， **PK,** {h_x}, {C_x}, **D**） for leaf node x, after getting { D_x }, the user firstly checks whether e (D_x , T_i) = h_x , and then, verifies

$$h_x = e(g,g)^{q_x(0)} = e(g,g)^{q_{parent(x)}(index(x))} = e(g,g)^{q_{parent(0)} + a_1(index(x)) + .. + a_{k-1}(index(x)^{k-1})}$$

$= h_{parent(x)} * \prod_{i=1}^{k-1} (e(g,g)^{a_i})^{index(x)^i}$, k=d ($q_{parent(x)}(\bullet)$), (1) if all the leaf nodes pass the verification, directly use h_x and equation (1) to verify the correctness of the parent nodes of leaf nodes, then repeats this procedure of checking for the upper nodes, until to the root node and at last checks whether $h_r = Y$. If in any step of verification fails, the user could stops, and requires the key generation center for right decryption keys.

Decryption (E, D) we specify the decryption procedure in a bottom-up manner: Let i= att (x)

If x is a leaf node, then:

DecryptNode (E, D, x) returns $e(D_x, E_i) = e(g^{q_x(0)/t_i}, g^{st_i}) = e(g,g)^{sq_x(0)}$ if $i \in \gamma$, otherwise, returns _|_;

If x is a non-leaf node, we recursively compute the DecryptNode (E, D, x), the output of it is denoted as F_x , for all nodes z that are children of x, let S_x be an arbitrary k_x -sized set of child nodes z such that $F_z \neq$ _|_, if no such set exists, the function returns _|_, otherwise, we compute:

$$F_x = \prod_{z \in S_x} F_z^{\Delta_{i,S'_x}(0)} ,$$ where i=index(z), S'_x ={index(z): z$\in S_x$ }

$$= e(g,g)^{sq_x(0)}$$ using Lagarange polynomial interpolation

We can know that, at last when reaching the root node r, we get DecryptNode (E, D, r)= $e(g,g)^{sy} = Y^s$, so if the condition Γ (γ)=1 is satisfied, the user can decrypt.

3.2 The effectiveness and the security proof for the verifiable version of ABE scheme

Theorem1. If all the checks in the verification procedure pass, the verifiable version of GPSW ABE scheme satisfies the two conditions in 2.3, and the scheme is also secure in the selective-set model defined in 2.4 under the decisional BDH assumption.

Proof sketch: First, we observe the results about the conditions in2.3. It is easy to prove that the condition1 is satisfied. Let's have a look at condition2, the additional information of each leaf is not the same as the shares in the key generation, but the real secret is y, so in the first step of check whether e (D_x, T_i) equals h_x to ensure that the very $q_x(0)$ in D_x is the same as that in h_x . The sharing process is to share y, so the polynomial in each step to finally get D_x and h_x is the same, if $h_x = h_{parent(x)} * \prod_{i=1}^{k-1} (e(g,g)^{a_i})^{index(x)^i}$ passes, we can be sure that h_x is rightly computed from the polynomial of parent(x), namely, $q_x(0) = q_{parent(x)}(index(x))$ is rightly computed from the polynomial, thus, $q_{parent(x)}(0)$ is shared without mistakes, while for degree d polynomial $q_{parent(x)}(\bullet)$, every qualified structure in this level at least contains $k_{parent(x)}$ values, while $k_{parent(x)} = d+1$, so these values uniquely decide the polynomial. If from two qualified sub structure we get different secrets, the difference must be from some level of sharing, then, in this level of sharing, at least two qualified sets of values from one polynomial reconstruct different secret, that is a contradiction. And at last, we check $h_r = Y$ to ensure the initial secret is the same as the one in the public key. If all these checks are valid, the initial secret y is rightly shared in each step to the final pieces of sharing.

Next, we show the security of our scheme. As in [GPSW06], we use A the adversary of attacking the ABE scheme with advantage ε to build a simulator B for solving the DBDH problem with advantage ε /2. The challenger flips a fair binary coin μ, if μ =0, it sets the tuple (A,B,C,Z)= $(g^a, g^b, g^c, e(g,g)^{abc})$, else, it sets the tuple (A,B,C,Z)= $(g^a, g^b, g^c, e(g,g)^z)$, for random a,b,c,z. The simulation proceeds as follows:

Init B runs A, A chooses the attributes set γ he wishes to attack.

Setup B sets the parameter Y=e(A,B)= $e(g,g)^{ab}$, for all i in the universe, if i $\in \gamma$, set $T_i = g^{r_i}, r_i$ is randomly chosen from $\mathbb{Z}_p$, otherwise, set $T_i = B^{k_i}, k_i$ is randomly chosen from $\mathbb{Z}_p$, then B gives the public parameters to A

Phase1 A makes requests for keys corresponding to access structure Γ' that $\Gamma'(\gamma) \neq 1$. We define two functions Satpoly and Unsatpoly.

Satpoly($\Gamma_x, \gamma, \lambda_x$) constructs the polynomials for the sub tree Γ_x and $\Gamma_x(\gamma) = 1$. It first sets up a polynomial q_x of degree d_x for the root node x and satisfying $q_x(0) = \lambda_x$. For each child node x' of x, we defines its polynomial by calling Satpoly($\Gamma_{x'}, \gamma, q_x(index(x'))$).

Unsatpoly($\Gamma_x, \gamma, g^{\lambda_x}$) sets up polynomials for the sub tree Γ_x and $\Gamma_x(\gamma) = 0$. For unsatisfied root node x, it has num_x children, k_x as its gate threshold value, and the degree d_x of the polynomial $q_x(\bullet)$ for node x is $k_x - 1$, because x is unsatisfied, so only h_x child nodes of x are satisfied, B randomly choose λ_y and ensures $q_y(0) = q_x(index(y)) = \lambda_y$, as B does not know λ_x and B has to make sure that $q_x(0) = \lambda_x$, B can only share the known g^{λ_x}, so for another $d_x - h_x$ child nodes, B randomly choose a value λ_z for each, which satisfying $g^{q_x(index(z))} = g^{\lambda_z}$, then the polynomial $q_x(\bullet)$ is decided in this way: $g^{q_x(\bullet)} = g^{a+a_1x+..+a_kx^k}$, B knows k= d_x different value $g^{q_x(i)}$, so B could compute all g^{a_i} i=1,… d_x. For the rest unsatisfied child nodes, B fixes the value using $g^{\lambda_z} = g^{q_x(index(z))}$ or direct interpolation. Next, B defines the polynomials for the child nodes recursively as follows, if child node x' is satisfied, B calls Satpoly ($\Gamma_{x'}, \gamma, q_x(index(x'))$) If child node x' is unsatisfied, B calls Unsatpoly($\Gamma_{x'}, \gamma, g^{q_x(index(x'))}$).

Notice that, for each node x, if x is satisfied, then, $q_x(0)$ is known, if x is unsatisfied, at least $g^{q_x(0)}$ and $g^{q_x(\bullet)}$ are known. Furthermore, the construction satisfies $q_r(0) = a$. The final polynomial $Q_x(\bullet) = bq_x(\bullet)$, the simulator B then computes all the values needed to send to A: for leaf node x, i = att(x), if $i \in \gamma$, B computes D_x = $B^{q_x(0)/r_i} = g^{bq_x(0)/r_i} = g^{Q_x(0)/t_i}$, if $i \notin \gamma$, B computes D_x = $g^{q_x(0)/k_i} = g^{bq_x(0)/bk_i} = g^{Q_x(0)/t_i}$. Therefore, the simulator is able to construct the private key for Γ', and the distribution is identical to that in the original scheme. Further, for all every node x, B can compute h_x = $e(B, g^{q_x(0)}) = e(g,g)^{Q_x(0)}$, and $C_x : \{e(B, g^{a_i})\}_{i=1..k_x-1} = \{e(g,g)^{ba_i}\}_{i=1..k_x-1}$, so all values for verification are ready. A then checks the correctness.

Challenge A sends B two messages M_0, M_1. The simulator B flips a coin ν, returns the encryption of M_ν, the ciphertext is as: E= (γ, E'= M_ν Z, $\{E_i = C^{r_i}\}_{i\in\gamma}$), Z is from the DBDH challenger, if μ =0, Z= $e(g,g)^{abc}$. Then, here, Y= $e(g,g)^{ab}$, s=c,

$E_i = C^{r_i} = (g^{r_i})^c = T_i^s$, therefore, E is a valid encryption. If μ =1, Z=$e(g,g)^z$,E' will be a random number.

Phase2 the simulator repeats phase1

Guess A submits his guess ν' for ν, if $\nu'=\nu$, the simulator B outputs μ'=0, otherwise, it outputs μ'=1.

The overall advantage of the simulator in the DBDH game is:

Pr [μ '= μ] -1/2= Pr[μ '= μ | μ =0].Pr[μ =0]+Pr[μ '= μ | μ =1]Pr[μ =1]-1/2

=1/2(Pr[μ '= μ | μ =0]+ Pr[μ '= μ | μ =1])-1/2

=1/2(1/2+ ε)+1/2.1/2-1/2= ε /2.

4 Verifiable version of Chase Multi-Authority ABE

4.1 our construction

Chase gives out a Multi-Authority ABE scheme in [Cha07] that to solve the problem Sahai and Waters left in their paper that more than one authority manipulate user's attributes, for example, different apartment of a company to handle the attributes related to its own apartment, if all apartments' requirement are satisfied, the user could then decrypt. The method used in the fuzzy ABE can also be used in GPSW ABE, so we take an abstract form of sub function to represent single authority ABE.

The details are as follows:

Init Fix prime order group G, G_1, bilinear map $e: G_1 \times G_1 \rightarrow G_2$, and generator $g \in G$.

Choose seeds $s_1,...,s_k$ for all authorities, also randomly choose $y_0, \{t_{k,i}\}_{k=1..K, i=1..n} \in Z_q$.

System public key $Y_0 = e(g,g)^{y_0}$.

Attribute Authority k

Authority Secret Key s_k , $t_{k,1}...t_{k,n}$

Authority Public Key $T_{k,1}...T_{k,n}$ where $T_{k,i} = g^{t_{k,i}}$

Secret Key for User u: Let $y_{k,u} = F_{s_k}(u)$.

To use a single authority verifiable ABE scheme as sub function with $y_{k,u}$ as its secret input to provide user with $\{D_x\}$.

Central Authority

Central Authority Secret Key s_k for all authorities k, y_0

Secret Key for User u: Let $y_{k,u} = F_{s_k}(u)$ for all k, Secret Key:

$D_{CA} = g^{(y_0 - \sum_{k=0}^{K} y_{k,u})}$, CA also constructs a table, storing information related to the secret of each authority, and publish the table, the table has K+1 columns and the row is labeled by user identification u, in each row, the CA put $Y_{k,u} = e(g,g)^{y_{k,u}}$ k is from 1 to K, the last one in a row is $Y_{CA} = e(g,g)^{(y_0 - \sum_{k=0}^{K} y_{k,u})}$, once a new user makes a query for decryption key, the CA add a new row to the table.

Encryption for Attribute set A_C Choose random s from Z_q . E= $Y_0^s m$, $E_{CA} = g^s$,

$$\{E_{k,i} = T_{k,i}^s\}_{i \in A_C^k, \forall k}$$

Verification: After getting the $\{D_x\}$ from each authority, the user verifies as in 3.2, and in the last step of verification, take the value in the table to compare, if passes for all authorities, check an equation in the row labeled by his user identification $Y_0 = Y_{CA} * \prod_{k=1}^{K} Y_k$, if this also passes, then the key the user got is a right one which could be used to decrypt.

Decryption: For each authority k, the authorized user could interpolate to reconstruct $Y_{k,u} = e(g,g)^{sy_{k,u}}$, compute $Y_{CA}{}^s = e(E_{CA}, D_{CA})$. Combine all these values to obtain $Y_0^s = Y_{CA}^s * \prod_{k=1}^{K} Y_k^s$. Then decrypt to get m.

4.2 The effectiveness and security proof

Theorem2. If all the checks pass, the verifiable version of Multi Authority ABE scheme satisfies the two conditions in 2.3, and based on the DBDH assumption, the scheme is secure in the selective-set model defined in 2.5.

Proof sketch: the proof of this theorem is very similar to the proof of theorem1,

First, checks the two conditions in 2.3, at each authority, the verification makes sure that the sharing process is correct, and the checks in the table ensures all the shares of the authorities are rightly shared by the CA from the top secret y_0 , the rest is the same as the proof of theorem1.

Next, the security proof only needs a few modifications of the original proof in [Cha07], and the modification method is like that in the proof in theorem1, just adding the verification information when answering the queries.

5 Conclusions and Open Problems

We present a verifiable version of ABE scheme which allows the user checks the correctness of the key, using to decrypt all qualified ciphertext, he got from the authority, doing this kind of verification reduces the trust of the authority, it is helpful when some error happens in creating or sending the secret, and it results in eliminates meaningless cost of decryption.

The verification algorithm of our scheme is not efficient enough, for the first scheme in our paper, if anyone can design a verification algorithm only computing in one step, it will be an elegant optimization.

References

[Sha85] Adi Shamir. Identity-based cryptosystems and signature schemes. In Proc. of CRYPTO 1984, volume 196, LNCS, 47-53. Springer

[BF01] Dan Boneh and Matthew Franklin. Identity-based encryption from the Weil pairing. In Proc. of CRYPTO 2001, volume 2139, LNCS, 213-229. Springer

[SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In Proc. of EUROCRYPT 2005, volume 3494, LNCS, 457-473. Springer

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Proc. of CCS06, 89-98, New York, ACM Press

[Cha07] Melissa Chase. Multi-Authority Attribute-Based Encryption In TCC07, volume 4392, LNCS, 515-534. Springer

[Ped91] Torben Pryds Pederson. Non-interactive and Information-theoretic Secure Verifiable Secret Sharing. In Proc. of CRYPTO 1991, volume 576, LNCS, 129-140, Springer