sensors

MDPI

Article

Analysis of an ABE Scheme with Verifiable Outsourced Decryption

Yongjian Liao [1,*], Yichuan He [1], Fagen Li [2], Shaoquan Jiang [3] and Shijie Zhou [1]

1 School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China; 201622060427@std.uestc.edu.cn (Y.H.); sjzhou@uestc.edu.cn (S.Z.)
2 School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China; fagenli@uestc.edu.cn
3 Institute of Information Security, Mianyang Normal University, Mianyang 621000, China; shaoquan.jiang@gmail.com
* Correspondence: liaoyj@uestc.edu.cn

Received: 25 December 2017; Accepted: 5 January 2018; Published: 10 January 2018

Abstract: Attribute-based encryption (ABE) is a popular cryptographic technology to protect the security of users' data in cloud computing. In order to reduce its decryption cost, outsourcing the decryption of ciphertexts is an available method, which enables users to outsource a large number of decryption operations to the cloud service provider. To guarantee the correctness of transformed ciphertexts computed by the cloud server via the outsourced decryption, it is necessary to check the correctness of the outsourced decryption to ensure security for the data of users. Recently, Li et al. proposed a full verifiability of the outsourced decryption of ABE scheme (ABE-VOD) for the authorized users and unauthorized users, which can simultaneously check the correctness of the transformed ciphertext for both them. However, in this paper we show that their ABE-VOD scheme cannot obtain the results which they had shown, such as finding out all invalid ciphertexts, and checking the correctness of the transformed ciphertext for the authorized user via checking it for the unauthorized user. We first construct some invalid ciphertexts which can pass the validity checking in the decryption algorithm. That means their "verify-then-decrypt" skill is unavailable. Next, we show that the method to check the validity of the outsourced decryption for the authorized users via checking it for the unauthorized users is not always correct. That is to say, there exist some invalid ciphertexts which can pass the validity checking for the unauthorized user, but cannot pass the validity checking for the authorized user.

Keywords: attribute-based encryption; outsourced decryption; verifiable; cloud computing; authorized client; wireless sensor

1. Introduction

Recently, cloud computing has become a very fascinating computing paradigm, in which storage and computation have moved away from terminal devices to the remote side. There are many novel applications in this area, such as outsourcing computation [1,2] and outsourcing verification [3]. This new and popular method brings important revolutions for the management, distribution and sharing data of enterprises and individuals, especially for some constrained devices, such as mobile phone, wireless sensors. Cloud clients (or sensors) are able to achieve significant cost savings by outsourcing their data storage and computation to some cloud service providers. Since the data of cloud clients (or sensors) are out of control by themselves, how to ensure the data security of cloud clients (sensors) is a significant problem in academia and industrial. Utilizing all kinds of cryptographic schemes is an essential method to achieve this goal. While attribute-based encryption (ABE) [4] is one

of the most popular notions to study and utilize in cloud computing since it has the property of the flexible and fine-grained access control.

The notion of ABE was first introduced by Sahai and Waters [4]. There are two different types of ABE schemes according to the manner to deploy the access control policy, key-policy attribute-based encryption (KP-ABE) [5] and ciphertext-policy attribute-based encryption (CP-ABE) [6]. The ciphertexts are labeled with sets of attributes and access policies over these attributes are associated with clients' private keys in the KP-ABE scheme. While every ciphertext is associated with an access policy, and every client's private key is associated with a set of attributes in the CP-ABE scheme. However, decryption operations of most requirement that the set of attributes should satisfy the access policy in any ABE system and in most existing ABE schemes, one of the main drawbacks is that the length of the ciphertext and the decryption computational cost grow with the complexity of the access policy. This becomes critical obstacle in various applications, especially the applications on resource-limited devices.

In order to reduce the decryption time and the computation cost, Green et al. [7] proposed an ABE scheme with outsourced decryption (ABE-OD). In their scheme, an authorized client first delegated an untrusted cloud server to convert the original ciphertext into a transformed ciphertext with a transformation key, and then the client obtained the plaintext from the transformed ciphertext by spending a small overhead. The ABE-OD scheme would not leak any information about the encrypted data. However, the ABE-OD proposed by Green et al. cannot ensure the correctness of the transformed ciphertext since the cloud server is public and untrusted. The untrusted cloud server may send a wrong transformed ciphertext to the clients for saving computing cost or suffering from malicious attack which also causes to generate the incorrectly transformed ciphertext. In order to ensure the correctness of the ciphertext, Lai et al. [8] put forth an ABE-OD scheme that can check the correctness of the transformed ciphertext generated by the cloud server, which was called ABE with verifiable outsourced decryption (ABE-VOD). In their ABE-VOD scheme, the data owner encrypted a plaintext and a random message to the ciphertext respectively, and generated a commitment of an actual plaintext and the random message. And in the decryption algorithm of their ABE-VOD scheme, the client should compute the plaintext and the random message to use the commitment to verify whether the transformed ciphertext is generated correctly. A client was able to verify the correctness of the transformed ciphertext if and only if his/her attributes set satisfies the access structure associated with the ciphertext. Subsequently, several ABE-VOD schemes were proposed according to different methods and distinct scenarios in [9–13]. And Qiu et al. [14] used an ontology-based approach to achieve attribute-based access controls as well.

Recently, Li et al. [15] proposed a full verifiability for outsourced decryption in ABE, which could simultaneously check the correctness of transformed ciphertext for the authorized clients and unauthorized clients. In their scheme, a data owner constructed two access policies for the authorized clients and unauthorized clients, respectively. And then the data owner uses a short "signature" for each ciphertext to ensure that the client could verify the validity of the transformed ciphertext. In order to avoid first computing the plaintext and then verifying the validity of the ciphertext, Li et al. used "verify-then-decrypt" skill rather than "decrypt-then-verify" paradigm. That is to say, the client first verified the validity of the ciphertext or the transformed ciphertext, and then decrypted the ciphertext and obtains the corresponding plaintext or the random message if the ciphertext or the transformed ciphertext passed the verification of its validation.

1.1. Motivation and Contribution

In cloud computing, the ABE-OD scheme cannot ensure the correctness of the ciphertext or the transformed ciphertext for cloud server being untrusted. The untrusted server may send a wrong transformed ciphertext to the users for saving computing cost or it may have suffered from malicious attack which also produces the incorrect ciphertext or transformed ciphertext. In order to

ensure the correctness of the ciphertext or the transformed ciphertext, the ABE-VOD schemes were proposed in [9–13,15].

However, we firstly show that the validity verification method in decryption algorithm of the ABE-VOD scheme put forth by Li et al. [15] cannot always check the validity of all ciphertexts in this paper. That is to say, there exist some invalid ciphertexts which can pass the validity checking and output the "corresponding" plaintexts. Furthermore, even if the untrusted server honestly performs the outsourced decryption for these invalid ciphertexts, the decryption algorithm cannot check them (the decryption algorithm cannot output $\perp$). Thus, the "verify-then-decrypt" skill used in [15] is unavailable. Then, we show that the method to check the validity of the outsourced decryption for the authorized user via checking it for the unauthorized user is not always correct. That is to say, there exist some invalid ciphertexts which can pass the validity checking for the unauthorized user, but cannot pass the correctness of the ciphertexts checking for the authorized user.

1.2. Organization of the Paper

The rest of this paper is organized as follows. The system model of the ABE-VOD and some basic mathematic knowledge are introduced in Section 2. In Section 3, we review the ABE-VOD scheme proposed by Li et al., and analyze their scheme. Finally, the conclusions are given in Section 4.

2. Premilinary

In the section, we will recall the definition of ABE-VOD and some basic mathematic knowledge in [15].

2.1. System Model

The ABE-VOD Scheme consists of seven algorithms: Setup, KeyGen, Encrypt, Decrypt, GenTK$_{out}$, Transform$_{out}$ and Decrypt$_{out}$. The detailed is described as follows.

- Setup$(1^\lambda, U)$. Take as input a security parameter 1^λ and attribute universe description U, generate a master secret key msk and public parameters PK.
- KeyGen(msk, PK, S). Take as input the master secret key msk, the public parameters PK and an attribute set S, generate the client's private key SK. If a client is an authorized one, use SK_{DS} to represent the private key of the authorized client, where DS represents an attribute set of the authorized client. If a client is an unauthorized one, the client uses SK_{VS} to represent the private key of the unauthorized client, where VS represents an attribute set of the unauthorized client.
- Encrypt$(PK, M, \mathbb{A}, \bar{\mathbb{A}})$. Take as input the public parameters PK, the plaintext M and two access structures $\mathbb{A}, \bar{\mathbb{A}}$, and output a ciphertext CT.
- Decrypt(SK, CT). Take as input a private key SK and a ciphertext CT. If the client's attribute set S satisfies the access policy $\mathbb{A}$, then the client utilizes the private key SK_{DS} to decrypt the ciphertext; otherwise, the client utilizes the private key SK_{VS} to decrypt the ciphertext. After the client checks the correctness of the ciphertext, he/she outputs the plaintext M if the ciphertext is valid; otherwise, the client outputs $\perp$.
- GenTK$_{out}(PK, SK)$. Take as input the public parameters PK and the private key SK, genetate a transformation key TK and a retrieving key RK. If a client is an authorized one, let $SK = SK_{DS}$ and set $TK = TK_{DS}$, $RK = RK_{DS}$; otherwise, let $SK = SK_{VS}$ and set $TK = TK_{VS}$, $RK = RK_{VS}$.
- Transform$_{out}(TK, CT)$. Take as input the transformation key TK and the ciphertext CT, generate the transformed ciphertext TCT.
- Decrypt$_{out}(CT, TCT, RK)$. Take as input a ciphertext CT, a transformed ciphertext TCT and a retrieving key RK. If the client's attribute set S satisfies the access policy $\mathbb{A}$, the client is an authorized one and then he/she utilizes CT, TCT and RK_{DS} to decrypt the ciphertext; otherwise, the client utilizes the private key CT, TCT and RK_{VS} to decrypt the ciphertext. After the client

checks the correctness of the ciphertext, outputs the plaintext M if the ciphertext is valid; otherwise, outputs $\perp$.

2.2. Bilinear Pairing

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two multiplicative groups which have the same prime order q, $\mathbb{Z}_q^*$ be the multiplicative group of the finite field $\mathbb{F}_q$. A bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ [16], which satisfies the followings three properties:

- Bilinearity: For any $\alpha, \beta, \gamma \in \mathbb{G}_1$,

$$e(\alpha, \beta\gamma) = e(\alpha, \beta)e(\alpha, \gamma), \text{and}$$

$$e(\alpha\beta, \gamma) = e(\alpha, \gamma)e(\beta, \gamma).$$

- Non-degeneracy: There are elements $\alpha, \beta \in \mathbb{G}_1$, such that $e(\alpha, \beta) \neq 1$, where 1 is the identity element of $\mathbb{G}_2$.
- Computability: For any elements $\alpha, \beta \in \mathbb{G}_1$, there is an efficient algorithm to compute $e(\alpha, \gamma)$.

The concrete bilinear pairings e will be using the modified Weil [17] or Tate pairings [18] on some elliptic curves. We will define two hard problems used in our paper below: Decisional Diffie-Hellman (DDH) problem and Computational Diffie-Hellman (CDH) problem. Let α be a generator of the group $\mathbb{G}_1$.

Definition 1. *(CDH problem in $\mathbb{G}_1$). Given $\alpha, \alpha^x, \alpha^y \in \mathbb{G}_1$, to compute α^{xy}.*

Definition 2. *(DDH problem in $\mathbb{G}_1$). Given α, α^x, α^y, $\alpha^z \in \mathbb{G}_1$, to decide whether $xy \equiv z \bmod q$ holds or not.*

It is obvious that the DDH problem in $\mathbb{G}_1$ is easy since it can verify above congruence by using the bilinear pairing e. However, as far there is no polynomial-time algorithm to solve CDH problem in $\mathbb{G}_1$, we assume that CDH problem in $\mathbb{G}_1$ is hard.

2.3. Linear Secret Sharing Schemes

We recall a description for LSSS in [19]. Let $\mathcal{P}$ be a set of parties. A secret sharing scheme Π is called linear (over $\mathbb{Z}_p$) if it satifies the following conditions.

- The secret shares of each party form a vector in $\mathbb{Z}_p$.
- Let A is a matrix with l rows and n columns. Let the function ρ represent the party labeling row i as $\rho(i)$, where is the ith row of A. Suppose a vector $\vec{v}_i = (s, r_2, \ldots, r_n)^T$ is the column vector and $r_2, \ldots, r_n$ are random value in $\mathbb{Z}_p$, where $s \in \mathbb{Z}_p$ is the secret to be shared. $A\vec{v}$ is the vectors of l shares for the the secret s with respect to Π. The share $(A\vec{v})_i$ belongs to party $\rho(i)$. Suppose that Π is an LSSS of the access policy $\mathbb{A}$ and $S \in \mathbb{A}$ is any authorized set. Let $I = \{i : \rho(i) \in S\} \subset [l] = \{1, 2, \ldots, l\}$. If $\{\lambda_i\}$ are valid shares for any secret s with respect to Π, then we can compute constants $\{\omega_i \in \mathbb{Z}_p\}_{i \in I}$ such that $\sum_{i \in I} \omega_i \lambda_i = s$, where $\lambda_i = (A\vec{v})_i$.

Notations. The vector $(1, 0, \ldots, 0)$ is the "target" vector of any LSSS. For any unauthorized set of rows I in A, the target vector is not in the span of the rows of set I. For any authorized set of rows I in A, the target vector is in the span of I.

3. Analysis of Li et al.'s Abe-Vod Scheme

Since ABE-VOD scheme proposed by Li et al. is much complex, we recall it in Appendix B and the security model in Appendix A.

3.1. The Excepted Functionalities of the ABE-VOD Scheme

In the subsection, we analyze the construction of the ABE-VOD scheme proposed by Li et al. The scheme wanted to get the following results at least.

- First, any ABE-VOD should have the decryption functionality. The decryption algorithm of the ABE-VOD can correctly check the valid ciphertext and invalid ciphertext (any encryption scheme must satisfy this condition). That is to say, the Decrypt algorithm outputs a corresponding plaintext of some ciphertext if and only if the ciphertext is valid, or the Decrypt_{out} algorithm outputs the corresponding plaintext of a transformed ciphertext if and only if the transformed ciphertext is correct.
- Then, the ABE-VOD scheme can simultaneously check the correctness of the transformed ciphertext for the authorized users and unauthorized users by using "verifying-then-decrypt" method to guarantee the correctness of the transformed ciphertext.

3.2. The ABE-VOD Scheme Cannot Verify the Validity of All Ciphertexts

In general, the goal of the verification formulas of the decryption algorithm are to check the correctness of ciphertext. However, the decryption algorithm of ABE-VOD scheme proposed by Li et al. only checks validity of a part of ciphertext, but not checks whether the output of the decryption algorithm for some ciphertext is the original plaintext . In the subsection, we show that there exist some ciphertexts which are verified by the decryption algorithm, but its output isn't the original plaintext.

As analysis in [15], the ciphertext stored in cloud server maybe be tampered by some malicious attackers or the transformed ciphertext could be generated via using incorrect one by the untrusted cloud server. We will view these activities as attacks of an adversary and describe how an adversary constructs an invalid ciphertext below, which the decryption algorithm will view as a valid ciphertext and output the "corresponding" plaintext.

The adversary takes as input a random message $M \in \{0,1\}^m$ and the two LSSS access structures $\mathbb{A} = (A, \rho)$, $\bar{\mathbb{A}} = (\bar{A}, \bar{\rho})$.

The adversary first picks up a random string $R \in \{0,1\}^m$, two random vectors

$$\vec{v} = (s_1, v_{12}, \cdots, v_{1n}) \in (\mathbb{Z}_p^*)^n$$

and

$$\vec{v}' = (s_2, v_{22}, \cdots, v_{2n}) \in (\mathbb{Z}_p^*)^n$$

and two random elements $s_1', s_2' \in \mathbb{Z}_p^*$ such that $s_1 \neq s_1'$ and $s_2 \neq s_2'$. For each row A_i of A, $\bar{A}_i$ of $\bar{A}$, it picks $r_{1,i}, r_{2,i} \in \mathbb{Z}_p^*$ uniformly at random. Then, it calculates:

$$C_M = M \oplus H_3(e(g,g)^{\alpha s_1'}),$$

$$\eta_1 = H_1(e(g,g)^{\alpha s_1}),$$

$$C_1 = g^{s_1},$$

$$C_{1,i} = g^{aA_i \cdot \vec{v}} T_{\rho(i)}^{-r_{1,i}}, D_{1,i} = g^{r_{1,i}} \forall i \in \{1, 2, \cdots, l\}.$$

Set $CT_M = (C_M, C_1, \{C_{1,i}\}_{i\in[l]}, \{D_{1,i}\}_{i\in[l]})$, and compute:

$$C_R = R \oplus H_3(e(g,g)^{\alpha s_2'}),$$

$$\eta_2 = H_1(e(g,g)^{\alpha s_2}),$$

$$C_2 = g^{s_2},$$

$$C_{2,i} = g^{a\bar{A}_i \cdot \vec{v}'} T_{\bar{\rho}(i)}^{-r_{2,i}}, D_{2,i} = g^{r_{2,i}}, \ \forall i \in [l].$$

Set $CT_R = (C_R, C_2, \{C_{2,i}\}_{i\in[l]}, \{D_{2,i}\}_{i\in[l]})$.

$$\sigma_1 = H_2(C_M, C_R)^{\eta_1}, \sigma_M = \{\sigma_1, H_2(C_M, C_R)\},$$

$$\sigma_2 = H_2(C_M, C_R)^{\eta_2}, \sigma_R = \{\sigma_2, H_2(C_M, C_R)\}.$$

The ciphertext $CT = (\mathbb{A}, \bar{\mathbb{A}}, CT_M, \sigma_M, CT_R, \sigma_R)$.

Obviously, the ciphertext CT is not a valid ciphertext of the message M since the adversary picks two distinct random numbers s_1 and s_1' to produce the ciphertext CT_M, and picks two distinct random numbers s_2 and s_2' to produce the ciphertext CT_R. However, the decryption algorithm will view it as a valid ciphertext and output the "corresponding" plaintext. When the decryption algorithm takes as input CT and SK, it runs as follows.

- If S satisfies the access policy $\mathbb{A}$, the private key SK of an authorized client is

$$(DS, K = g^{\alpha} y^{t_1}, K_0 = g^{t_1}, \{K_i = T_i^{t_1}\}_{att_i \in DS}).$$

Let $I = \{i : \rho(i) \in S\} \subset [l] = \{1, 2, \cdots, l\}$. Then it calculates $\omega_i \in \mathbb{Z}_p^*$ for $i \in I$ such that $\Sigma_{i\in I}\omega_i A_i = (1, 0, \cdots, 0)$, and computes:

$$X_M = \frac{e(C_1, K)}{\prod_{i\in I}(e(C_{1,i}, K_0)e(D_{1,i}, K_{\rho(i)}))^{\omega_i}},$$

which equals $e(g, g)^{\alpha s_1}$.

It is clear that the equality

$$e(\sigma_1, g) = e(H_2(C_M||C_R), g^{\eta_1})$$

holds, where $\eta_1 = H_1(X_M)$. Then it computes

$$M' = C_M \oplus H_3(X_M) = M \oplus e(g, g)^{\alpha s_1'} \oplus e(g, g)^{\alpha s_1}.$$

However, M' does not equal M since $s_1' \neq s_1$. That is to say, the decryption algorithm cannot refuse the plaintext of the ciphertext which is produced by other "encryption" algorithm.

- If S satisfies the access policy $\bar{\mathbb{A}}$, the private key SK of an unauthorized client is

$$(VS, K = g^{\alpha} y^{t_2}, K_0 = g^{t_2}, \{K_i = T_i^{t_2}\}_{att_i \in VS}).$$

Let $I = \{i : \bar{\rho}(i) \in S\} \subset \{1, 2, \cdots, l\}$. Then it calculates $\omega_i \in \mathbb{Z}_p^*$ for $i \in I$ such that $\Sigma_{i\in I}\omega_i \bar{A}_i = (1, 0, \cdots, 0)$, and computes:

$$X_R = \frac{e(C_2, KP)}{\prod_{i\in I}(e(C_{2,i}, KP_0)e(D_{2,i}, KP_{\bar{\rho}(i)}))^{\omega_i}},$$

which equals $e(g, g)^{\alpha s_2}$.

For the same reason above, the equality

$$e(\sigma_2, g) = e(H_2(C_M||C_R), g^{\eta_2})$$

holds, where $\eta_2 = H_1(X_R)$. Then it computes

$$R' = C_R \oplus H_3(X_R) = R \oplus e(g, g)^{\alpha s_2'} \oplus e(g, g)^{\alpha s_2}.$$

However, R' does not equal R since $s_2' \neq s_2$.

Thus, the decryption algorithm of the ABE-VOD scheme proposed by Li et al. for both the authorized client and the unauthorized client cannot check the validity of all ciphertexts. I.e., there exist some invalid ciphertexts which can pass the validity checking. Furthermore, their ABE-VOD scheme cannot check the validity of the outsourcing computation by checking the correctness of the corresponding ciphertext since the output of both the Decrypt algorithm and Decrypt_{out} algorithm is not always correct.

3.3. The ABE-VOD Scheme Is Not Full Verifiable

Since verifying the correctness of the outsourced decryption for unauthorized clients is very important, Li et al. considered the following scenario. The authorized user wants to, but is not able to, process some pending businesses when the time or position of the authorized client is limited. He/she needs someone to help him/her to verify whether a pending business is correctly processed and does not want the latter to know anything about the content of the business. Thus Li et al. proposed the ABE-VOD scheme which could utilize an unauthorized client to help him/her to verify the correctness of the transformed ciphertext. We construct the following ciphertext which can pass the correctness checking for an unauthorized client but it is not a valid ciphertext for the authorized client.

The adversary takes as input a random message $M \in \{0,1\}^m$ and the two LSSS access structures $\mathbb{A} = (A, \rho)$, $\bar{\mathbb{A}} = (\bar{A}, \bar{\rho})$.

The adversary first picks a random string $R \in \{0,1\}^m$, two random vectors

$$\vec{v} = (s_1, v_{12}, \cdots, v_{1n}) \in (\mathbb{Z}_p^*)^n$$

and

$$\vec{v}' = (s_2, v_{22}, \cdots, v_{2n}) \in (\mathbb{Z}_p^*)^n.$$

For each row $\bar{A}_i$ of $\bar{A}$, it picks $r_{2,i} \in \mathbb{Z}_p^*$ uniformly at random. And it uniformly picks

$$C_M \in \{0,1\}^m,\ \eta_1 \in \mathbb{Z}_p^*,\ C_1,\ \{C_{1,i},\ D_{1,i}\}_{[l]} \in \mathbb{G}_1$$

at random.

Set $CT_M = (C_M, C_1, \{C_{1,i}\}_{i\in[l]}, \{D_{1,i}\}_{i\in[l]})$, then it calculates:

$$C_R = R \oplus H_3(e(g,g)^{\alpha s_2}),$$

$$\eta_2 = H_1(e(g,g)^{\alpha s_2}),$$

$$C_2 = g^{s_2},$$

$$C_{2,i} = g^{a\bar{A}_i \cdot \vec{v}'} T_{\bar{\rho}(i)}^{-r_{2,i}}, D_{2,i} = g^{r_{2,i}},\ \forall\ i \in [l].$$

Set $CT_R = (C_R, C_2, \{C_{2,i}\}_{i\in[l]}, \{D_{2,i}\}_{i\in[l]})$.

$$\sigma_1 = H_2(C_M, C_R)^{\eta_1}, \sigma_M = \{\sigma_1, H_2(C_M, C_R)\},$$

$$\sigma_2 = H_2(C_M, C_R)^{\eta_2}, \sigma_R = \{\sigma_2, H_2(C_M, C_R)\}.$$

The ciphertext $CT = (\mathbb{A}, \bar{\mathbb{A}}, CT_M, \sigma_M, CT_R, \sigma_R)$.

It is clear that if S satisfies the access policy $\mathbb{A}$, the authorized client cannot pass the checking of the correctness of the ciphertext. Because the elements $C_M, C_1, \{C_{1,i}\}_{i\in[l]}, \{D_{1,i}\}_{i\in[l]}$ are random elements, which is a valid ciphertext with a negligible probability. That is to say, since the equation $\eta_1 = H_1(\frac{e(C_1,K)}{\prod_{i\in I}(e(C_{1,i},K_0)e(D_{1,i},K_{\rho(i)}))^{\omega_i}})$ with negligible probability for random elements $C_M, C_1, \eta_1, \{C_{1,i}\}_{i\in[l]}$,

$\{D_{1,i}\}_{i\in[l]}$, σ_1 is a valid signature of $H_2(C_M, C_R)$ with negligible probability. We use the decryption algorithm to check the equality

$$e(\sigma_1, g) = e(H_2(C_M||C_R), g^{\eta_1}),$$

which holds with negligible probability for random elements $C_M, C_1, \eta_1, \{C_{1,i}\}_{i\in[l]}, \{D_{1,i}\}_{i\in[l]}$.

However, if S satisfies the access policy $\bar{\mathbb{A}}$, the unauthorized client can pass the correctness checking of the ciphertext. Because the adversary uses the Encrypt algorithm to encrypt the message R for the unauthorized client. The equations

$$\eta_2 = H_1(e(g, g)^{\alpha s_2}),$$

$$\text{and } \sigma_2 = H_2(C_M, C_R)^{\eta_2}$$

hold. That means

$$e(\sigma_2, g) = e(H_2(C_M||C_R), g^{\eta_2})$$

always holds. Thus, the decryption algorithm can output plaintext R correctly. Especially, when the untrusted server honestly runs the Transform$_{out}$ algorithm, the unauthorized client can always pass the correctness checking of the transformed ciphertext.

Thus, the ABE-VOD scheme cannot verify the correctness of the ciphertext or the transformed ciphertext for the authorized user via verifying it for the unauthorized user.

3.4. Furthermore Analysis

We have showed that the decryption algorithm cannot satisfy two functionalities, checking the correctness of all ciphertexts and "full verifiable" above. Next, we will explain the reason and possibly reasonable method.

On one hand, the construction of the above ABE-VOD scheme utilized ABE-OD scheme proposed by Green et al. [7] and short signature scheme proposed by Boneh et al. [16]. The one-time signature σ_1 of a "message" $H_2(C_M, C_R)$ (or σ_2 of a "message" $H_2(C_M, C_R)$) is unforgeable and it also ensures that

$$e(\sigma_1, g) = e(H_2(C_M||C_R), g^{\eta_1})$$

or

$$e(\sigma_2, g) = e(H_2(C_M||C_R), g^{\eta_2})$$

holds if and only if σ_1 and σ_2 are valid signatures of $H_2(C_M||C_R)$ (or C_M and C_R) under public key g^{η_1} and g^{η_2}, respectively. However, there is no condition that guarantees the validity of C_M and C_R. That is to say, we can choose any random element as C_M (or C_R). Thus, the above adversary can construct an invalid C_M or C_R but the ciphertext CT can be verified as a valid ciphertext. It seems that the method to sign a part of the ciphertext cannot guarantee all invalid ciphertexts to be refused. It needs another secure mechanism to guarantee the part of the ciphertext is valid.

On the other hand, from the unauthorized client's view, C_M is a random element in $\{0,1\}^m$, which is independent of $C_R, \bar{\mathbb{A}}$ and σ_R. Thus, the unauthoized client has no capability to verify the validity of C_M, and the construction in [15] cannot check the correctness of the ciphertext and the transformed ciphertext for the authorized users by checking the validity of the ciphertext and the correctness of the transformed ciphertext for the unauthorized clients.

4. Conclusions

In this paper, we showed that the validity verification method in decryption algorithm of the ABE-VOD scheme put forth by Li et al. cannot always check the validity of all ciphertexts. There exist some invalid ciphertexts which can pass the validity checking and the "verify-then-decrypt" skill

used in [15] is unavailable. Then, we showed that the method to check the validity of the outsourced decryption for the authorized client via checking it for the unauthorized client was not always correct. There exist some invalid ciphertexts which can pass the validity checking for the unauthorized client but cannot pass the validity checking for the authorized client. Finally, we pointed out that although the scheme used signature skill to guarantee the ciphertext cannot be tampered, the signing key of the "signature scheme" used in the encryption scheme was not fixed and anyone can generated it. That caused our constructions.

Acknowledgments: Our work was supported by the Sichuan Key Technology Support Program (No. 18ZDYF2907).

Author Contributions: The five authors of the paper have extensively participated in all of the paper analysis and manuscript revised. Fagen Li, Shaoquan Jiang and Shijie Zhou added to and revised the related works. Yongjian Liao and Yichuan He mainly wrote the manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Security Model

We recall the security model in [15]. We first consider the selective chosen plaintext attack (CPA) security model for ABE with fully verifiable outsourcing decryption is described by the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

- **Init**. The adversary $\mathcal{A}$ sets a challenge access policy $\mathbb{A}^*$ that it wishes to challenge.
- **Setup**. The challenger $\mathcal{C}$ executes the algorithm Setup to generate the public parameters PK and the master secret key msk. $\mathcal{C}$ sends PK to $\mathcal{A}$, and keeps msk secret.
- **Phase 1**. The challenger $\mathcal{C}$ sets a set D and a table T initially empty. The adversary $\mathcal{A}$ makes the following queries:
  - (1) *Private key* query. The adversary $\mathcal{A}$ makes private key queries on an attribute set S, the challenger $\mathcal{C}$ runs KeyGen algorithm to generate a private key SK_S, and sets $D = D \cup S$. Then it returns the private key to the adversary $\mathcal{A}$. The only restriction is that the attribute set cannot satisfy the access policy $\mathbb{A}^*$.
  - (2) *Transformation key* query. $\mathcal{A}$ makes transformation key queries on an attribute set S, and $\mathcal{C}$ searches the tuple (S, SK_S, TK_S, RK_S) in the table T. If such tuple exists, it returns TK_S as response. Otherwise, it executes KeyGen(PK, msk, S) to generate SK_S and GenTK$_{out}(PK, SK_S)$ to generate (TK_S, RK_S). Then the adversary $\mathcal{A}$ stores the tuple (S, SK_S, TK_S, RK_S) in table T. It returns the transformation key TK_S to $\mathcal{A}$.
- **Challenge**. The adversary $\mathcal{A}$ submits two messages M_0 and M_1 with the same size. Then $\mathcal{C}$ randomly picks a bit $b \in \{0,1\}$ and R with the same length as M_0 and M_1, and computes $CT^* =$ Encrypt$(PK, M_b, \mathbb{A}^*)$. Finally, the challenger $\mathcal{C}$ sends to CT^* to $\mathcal{A}$ as a challenge ciphertext.
- **Phase 2**. $\mathcal{A}$ proceeds to make *Private key* queries and *Transformation key* queries as Phase 1, however the only restriction is that the attribute set does not satisfy the access policy $\mathbb{A}^*$.
- **Guess**. $\mathcal{A}$ outputs its guess $b' \in \{0,1\}$ with respect to b and wins the game if $b' = b$.

The advantage of the adversary $\mathcal{A}$ in the above game is

$$|\Pr[b' = b] - \frac{1}{2}|,$$

where the probability is taken over the random bits by the adversary $\mathcal{A}$ and the challenger $\mathcal{C}$.

Definition A1. *An ABE-VOD scheme is selective CPA-secure if every polynomial time adversary $\mathcal{A}$ has at most a negligible advantage in the above game.*

Next, we review the formal definition of *verifiability* for an ABE-VOD scheme through a game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ [15]. The definition is just considered the part of the authorized user here, which is the same as the definition of verifiability for the unauthorized user. The game is described as follows:

- **Init**. The adversary $\mathcal{A}$ sets an access policy $\mathbb{A}^*$ that it wishes to challenge.
- Setup. The challenger runs the Setup$(1^\lambda, U)$ to generate the public parameters PK and the master key msk, then keeps msk secret and sends PK to the adversary.
- **Phase 1**. The adversary $\mathcal{A}$ can execute the *private key* query and the *transformation key* query as in Phase 1 in the above security game.
  - (1) *Private key* query. The adversary $\mathcal{A}$ makes private key queries on an attribute set S, the challenger runs KeyGen(msk, PK, S) to generate SK and sets $D = D \bigcup \{S\}$ which is initially empty. It then returns the private key SK_S to the adversary. The only restriction is that the attribute set S cannot satisfy the access policy $\mathbb{A}^*$.
  - (2) *Transformation key* query. $\mathcal{A}$ makes transformation key queries on the attribute set S; $\mathcal{C}$ searches the tuple (S, SK_S, TK_S, RK_S) in the table T. If the tuple exists, $\mathcal{C}$ returns TK_S as a response. Otherwise, it executes KeyGen(msk, PK, S) to generate SK_S and GenTK$_{out}(PK, SK_S)$ to generate (TK_S, RK_S). Then $\mathcal{C}$ stores the tuple (S, SK_S, TK_S, RK_S) in table T and returns the transformation key TK_S to $\mathcal{A}$.
- **Challenge**. The adversary submits a message M^*. The challenger computes a challenge ciphertext CT^* = Encrypt$(PK, M^*, \mathbb{A}^*)$ and sends it to $\mathcal{A}$.
- **Phase 2**. The same as Phase 1.
- **Output**. The adversary outputs an attributes set S^* and a transformed ciphertext TCT^*. We assume that the adversary knows (S^*, SK^*, TK^*, PK^*). The adversary wins the game if

$$\text{Decrypt}_{out}(PK, CT^*, TCT^*, PK^*) \notin \{M^*, \perp\}.$$

The advantage of the adversary $\mathcal{A}$ is

$$Adv^{verify}_{ABE_{out}}(1^\lambda) = \Pr[\mathcal{A}\ wins].$$

Definition A2. *(Verifiability) An ABE-VOD scheme is verifiable, if for any polynomial time adversary $\mathcal{A}$, the advantage $Adv^{verify}_{ABE_{out}}(1^\lambda)$ is negligible in the security parameter.*

Appendix B. Review of Li et al.'s Abe-Vod Scheme

Here, we recall the ABE-VOD scheme proposed by Li et al.

- Setup $(1^\lambda, U)$. Take as input the security parameter 1^λ and the attribute set $U = \{att_1, att_2, \cdots, att_l\}$. Generate bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, e)$, where $\mathbb{G}_1$ and $\mathbb{G}_2$ are two multiplicative groups with a prime order p. Choose a random generator $g \in \mathbb{G}_1$, random elements $h_1, \cdots, h_l \in \mathbb{Z}_p^*$ and $a, \alpha \in \mathbb{Z}_p^*$, computes $y = g^a$. Then generate three collision resistance hash functions

$$H_1 : \mathbb{G}_1 \to \mathbb{Z}_p,$$

$$H_2 : \{0,1\}^* \to \mathbb{G}_1$$

and

$$H_3 : \mathbb{G}_2 \to \{0,1\}^m.$$

$PK = (\mathbb{G}_1, \mathbb{G}_2, g, y, e(g,g)^\alpha, H_1, H_2, H_3, \{T_i = g^{h_i}\}_{i \in U})$ are published as the public parameters. The master secret key msk is α.

- KeyGen(msk, PK, S). To generate private keys for two types of clients (the authorized client and the unauthorized client). If S is an attribute set of the authorized client, then the algorithm picks a random value $t_1 \in \mathbb{Z}_p^*$. The private key of the authorized client is

$$SK_{DS} = (DS, K = g^{\alpha} y^{t_1}, K_0 = g^{t_1}, \{K_i = T_i^{t_1}\}_{att_i \in DS}).$$

  If S is an attribute set of the unauthorized client, then the algorithm picks a random value $t_2 \in \mathbb{Z}_p^*$. The private key for the unauthorized client is

$$SK_{VS} = (VS, KP = g^{\alpha} y^{t_2}, KP_0 = g^{t_2}, \{KP_i = T_i^{t_2}\}_{att_i \in VS}).$$

- Encrypt$(M, \mathbb{A}, \bar{\mathbb{A}})$. Take as input a message $M \in \{0,1\}^m$ and two LSSS access structures $\mathbb{A} = (A, \rho)$, $\bar{\mathbb{A}} = (\bar{A}, \bar{\rho})$. A and $\bar{A}$ are two $l \times n$ matrixes. ρ is a map from each row A_i of A to an attribute $\rho(i)$ and $\bar{\rho}$ is a map from each row $\bar{A}_i$ of $\bar{A}$ to an attribute $\bar{\rho}(i)$. The encryption algorithm first picks a random string $R \in \{0,1\}^m$ and two random vectors

$$\vec{v} = (s_1, v_{12}, \cdots, v_{1n}) \in (\mathbb{Z}_p^*)^n$$

  and

$$\vec{v}' = (s_2, v_{22}, \cdots, v_{2n}) \in (\mathbb{Z}_p^*)^n.$$

  For each row A_i of A, $\bar{A}_i$ of $\bar{A}$, it picks $r_{1,i}, r_{2,i} \in \mathbb{Z}_p^*$ uniformly at random. Then it computes:

$$C_M = M \oplus H_3(e(g,g)^{\alpha s_1}),$$

$$\eta_1 = H_1(e(g,g)^{\alpha s_1}),$$

$$C_1 = g^{s_1},$$

$$C_{1,i} = g^{aA_i \cdot \vec{v}} T_{\rho(i)}^{-r_{1,i}}, D_{1,i} = g^{r_{1,i}} \forall i \in \{1, 2, \cdots, l\}.$$

  Set $CT_M = (C_M, C_1, \{C_{1,i}\}_{i \in [l]}, \{D_{1,i}\}_{i \in [l]})$.

$$C_R = R \oplus H_3(e(g,g)^{\alpha s_2}),$$

$$\eta_2 = H_1(e(g,g)^{\alpha s_2}),$$

$$C_2 = g^{s_2},$$

$$C_{2,i} = g^{a\bar{A}_i \cdot \vec{v}'} T_{\bar{\rho}(i)}^{-r_{2,i}}, D_{2,i} = g^{r_{2,i}} \forall i \in \{1, 2, \cdots, l\}.$$

  Set $CT_R = (C_R, C_2, \{C_{2,i}\}_{i \in [l]}, \{D_{2,i}\}_{i \in [l]})$.

$$\sigma_1 = H_2(C_M, C_R)^{\eta_1}, \sigma_M = \{\sigma_1, H_2(C_M, C_R)\},$$

$$\sigma_2 = H_2(C_M, C_R)^{\eta_2}, \sigma_R = \{\sigma_2, H_2(C_M, C_R)\}.$$

  The ciphertext $CT = (\mathbb{A}, \bar{\mathbb{A}}, CT_M, \sigma_M, CT_R, \sigma_R,)$.
- Decrypt (SK, S, CT). Take as input the private key SK, an attribute set S of the client and a ciphertext $CT = (\mathbb{A}, \bar{\mathbb{A}}, CT_M, \sigma_M, CT_R, \sigma_R)$.

  – (1) If S satisfies the access policy $\mathbb{A}$, then the client is an authorized one and the private key of the client is SK= $(DS, K = g^{\alpha} y^{t_1}, K_0 = g^{t_1}, \{K_i = T_i^{t_1}\}_{att_i \in DS})$. Let $I = \{i : \rho(i) \in S\} \subset \{1, 2, \cdots, l\}$.

Then the client is able to compute $\omega_i \in \mathbb{Z}_p^*$ for $i \in I$ such that $\Sigma_{i\in I}\omega_i A_i = (1, 0, \cdots, 0)$, and the client calculates:

$$X_M = \frac{e(C_1, K)}{\prod_{i\in I}(e(C_{1,i}, K_0)e(D_{1,i}, K_{\rho(i)}))^{\omega_i}} = e(g, g)^{\alpha s_1},$$

and $\eta_1 = H_1(X_M)$. After the client checks whether the following equality

$$e(\sigma_1, g) = e(H_2(C_M||C_R), g^{\eta_1})$$

holds or not. If it holds, the client calculates

$$M = C_M \oplus H_3(X_M);$$

otherwise, the client outputs $\perp$.

– (2) If S satisfies the access policy $\bar{\mathbb{A}}$, then the client is an unauthorized one and the private key of the client is $SK = (VS, K = g^{\alpha}y^{t_2}, K_0 = g^{t_2}, \{K_i = T_i^{t_2}\}_{att_i\in VS})$. Let $I = \{i : \rho(i) \in S\} \subset \{1, 2, \cdots, l\}$. Then the client is able to compute $\omega_i \in \mathbb{Z}_p^*$ for $i \in I$ such that $\Sigma_{i\in I}\omega_i \bar{A}_i = (1, 0, \cdots, 0)$, and the client calculates:

$$X_R = \frac{e(C_2, KP)}{\prod_{i\in I}(e(C_{2,i}, KP_0)e(D_{2,i}, KP_{\bar{\rho}(i)}))^{\omega_i}} = e(g, g)^{\alpha s_2},$$

and $\eta_2 = H_1(X_R)$. After the client checks whether the following equality

$$e(\sigma_2, g) = e(H_2(C_M||C_R), g^{\eta_2})$$

holds or not. If it holds, the client computes

$$R = C_R \oplus H_3(X_R);$$

otherwise, the client outputs $\perp$.

- GenTK$_{out}(SK)$. Take the private key SK as input. If the client is an authorized one, the private key is SK $(DS, K = g^{\alpha}y^{t_1}, K_0 = g^{t_1}, \{K_i = T_i^{t_1}\}_{att_i\in DS})$. If the client is an unauthorized one, the private key is $SK = (VS, K = g^{\alpha}y^{t_2}, K_0 = g^{t_2}, \{K_i = T_i^{t_2}\}_{att_i\in VS})$. Then the client picks two random values $z_1, z_2 \in \mathbb{Z}_p^*$, and the transformation keys are

$$TK_{DS} = (DS, K' = K^{1/z_1}, K_0' = K_0^{1/z_1}, \{K_i' = K_i^{1/z_1}\}_{att_i\in DS}),$$

and

$$TK_{VS} = (VS, KP' = KP^{1/z_2}, KP_0' = KP_0^{1/z_2}, \{KP_i' = KP_i^{1/z_2}\}_{att_i\in VS}),$$

respectively. The retrieving keys are $RK_{DS} = z_1$ and $RK_{VS} = z_2$, respectively.
- Transform$_{out}(TK, CT)$. Takes as input the ciphertext CT and the transformation key TK. For the authorized client, the transformation key is $TK = TK_{DS}$, and for the unauthorized client, the transformation key is $TK = TK_{VS}$. The transformed is described as follows.

$$T_1' = \frac{e(C_1, K')}{\prod_{i\in I}(e(C_{1,i}, K_0')e(D_{1,i}, K_{\rho(i)'}))^{\omega_i}} = e(g, g)^{\alpha s_1/z_1},$$

$$T_2' = \frac{e(C_2, KP')}{\prod_{i\in I}(e(C_{2,i}, KP_0')e(D_{2,i}, KP'_{\bar{\rho}(i)}))^{\omega_i}} = e(g, g)^{\alpha s_2/z_2}.$$

Finally, the transformed ciphertext

$$TCT' = (C_M, T_1', \sigma_M)$$

if the attribute set S of the user satisfies the access policy $\mathbb{A}$ or

$$TCT' = (C_R, T_2', \sigma_R)$$

if the attribute set S of the client satisfies the access policy $\bar{\mathbb{A}}$.

- $\text{Decrypt}_{out}(CT, TCT', RK)$. Takes as input the ciphertext $CT = (\mathbb{A}, \bar{\mathbb{A}}, CT_M, \sigma_M, CT_R, \sigma_R)$, the transformed ciphertext TCT' and the retrieving key RK. The retrieving key of the authorized client $RK = RK_{DS} = z_1$ and the retrieving key of the unauthorized client $RK = RK_{VS} = z_2$.
  - (1) If the attribute set S of the client satisfies the access policy $\mathbb{A}$, the client verifies whether

    $$e(\sigma_1, g) = e(H_2(C_M||C_R), g^{H_1(T_1^{'z_1})})$$

    holds, if it does, then the client outputs

    $$M = C_M \oplus H_3(T_1^{'z_1});$$

    otherwise, the client outputs $\perp$.
  - (2) If the attribute set S of the client satisfies the access policy $\bar{\mathbb{A}}$, the client verifies whether

    $$e(\sigma_2, g) = e(H_2(C_M||C_R), g^{H_1(T_2^{'z_2})})$$

    holds, if it does, then the client outputs

    $$R = C_R \oplus H_3(T_2^{'z_2});$$

    otherwise, the client outputs $\perp$.

References

1. Yu, J.; Ren, K.; Wang, C. Enabling Cloud Storage Auditing with Verifiable Outsourcing of Key Updates. *IEEE Trans. Inf. Forensics Secur.* **2016**, *11*, 1362–1375.
2. Su, Q.; Yu, J.; Tian, C.; Zhang, H.; Hao, R. How to Securely Outsource the Inversion Modulo a Large Composite Number. *J. Syst. Softw.* **2017**, *127*, 26–34.
3. Liao, Y.; He, Y.; Li, F.; Zhou, S. Analysis of a Mobile Payment Protocol with Outsourced Verification in Cloud Server and the Improvement. *Comput. Stand. Interfaces* **2018**, *56*, 101–106.
4. Sahai, A.; Waters, B. Fuzzy Identity-Based Encryption. In Proceedings of the 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, 22–26 May 2005; pp. 457–473.
5. Goyal, V.; Pandey, O.; Sahai, A.; Waters, B. Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data. In Proceedings of the 13th ACM Conference on Computer and Communications Security, Alexandria, VA, USA, 30 October–3 November 2006; pp. 89–98.
6. Bethencourt, J.; Sahai, A.; Waters, B. Ciphertext-Policy Attribute-Based Encryption. In Proceedings of the IEEE Symposium on Security Privacy, Berkeley, CA, USA, 20–23 May 2007; pp. 321–334.
7. Green, M.; Hohenberger, S.; Waters, B. Outsourcing the Decryption of Abe Ciphertexts. In Proceedings of the 20th USENIX Conference on Security Symposium, San Francisco, CA, USA, 8–12 August 2011; p. 34.
8. Lai, J.; Deng, R.; Guan, C.; Weng, J. Attribute-Based Encryption with Verifiable Outsourced Decryption. *IEEE Trans. Inf. Forensics Secur.* **2013**, *8*, 1343–1354.

9. Qin, B.; Deng, R.H.; Liu, S.; Ma, S. Attribute-Based Encryption with Efficient Verifiable Outsourced Decryption. *IEEE Trans. Inf. Forensics Secur.* **2015**, *10*, 1384–1393.
10. Mao, X.; Lai, J.; Mei, Q.; Chen, K.; Weng, J. Generic and Efficient Constructions of Attribute-Based Encryption with Verifiable Outsourced Decryption. *IEEE Trans. Dependable Secure Comput.* **2015**, doi:10.1109/TDSC.2015.2423669.
11. Lin, S.; Zhang, R.; Ma, H.; Wang, M. Revisiting Attribute-Based Encryption with Efficient Verifiable Outsourced Decryption. *IEEE Trans. Inf. Forensics Secur.* **2015**, *10*, 2119–2130.
12. Li, J.; Huang, X.; Li, J.W.; Chen, X.; Xiang, Y. Securely Outsourcing Attribute-Based Encryption with Checkability. *IEEE Trans. Parallel Distribu. Syst.* **2014**, *25*, 2201–2210.
13. Li, J.; Sha, F.; Zhang, Y.; Huang, X.; Shen, J. Verifiable out-Sourced Decryption of Attribute-Based Encryption with Con-Stant Ciphertext Length. *Secur. Commun. Netw.* **2017**, doi:10.1155/2017/3596205.
14. Qiu, M.; Gai, K.; Thuraisingham, B.; Tao, L.; Zhao, H. Proactive User-Centric Secure Data Scheme Using Attribute-Based Semantic Access Controls for Mobile Clouds in Financial Industry. *Future Gener. Comput. Syst.* **2018**, *80*, 421–429.
15. Li, J.; Wang, Y.; Zhang, Y.; Han, J. Full Verifiability for Outsourced Decryption in Attribute Based Encryption. *IEEE Trans. Serv. Comput.* **2017**, doi:10.1109/TSC.2017.2710190.
16. Boneh, D.; Lynn, B.; Shacham, H. Short Signatures from the Weil Pairing. In Proceedings of the ASIACRYPT 7th International Conference on the Theory and Application of Cryptology and Information Security, Gold Coast, Australia, 9–13 December 2001; Volume 2248, pp. 514–532.
17. Boneh, D.; Franklin, M. Identity-Based Encryption from the Weil Pairing. In Proceedings of the CRYPTO 21st Annual International Cryptology Conference, Santa Barbara, CA, USA, 19–23 August 2001; Volume 2139, pp. 213–229.
18. Miyaji, A.; Nakabayashi, M.; Takano, S. New Explicit Conditions of Elliptic Curve Traces for FR-Reduction. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* **2001**, *84*, 1234–1243.
19. Beimel, A. Secure Schemes for Secret Sharing and Key Distribution. Ph.D. Dissertation, Israel Institute of Technology, Technion City, Haifa, Israel, 1996.