Comment on “An Efficient ABE Scheme With Verifiable Outsourced Encryption and Decryption”

ZHEN GUO, BIN LIU, KAIQIANG ZHAO, AND CHAOSHENG FENG, (Associate Member, IEEE)
School of Computer Science, Sichuan Normal University, Chengdu 610101, China
Corresponding author: Chaosheng Feng (csfenggy@126.com)

ABSTRACT Recently in IEEE Access (DOI: 10.1109/ACCESS.2018.2890565), Li et al. proposed a secure outsourcing algorithm for modular exponentiation in one single untrusted server model and a new method of generating transformation keys. They claimed that their solution can securely outsource encryption and decryption to untrusted ESP (encryption service providers) and DSP (decryption service providers), leaving only a constant number of simple operations for the DO (data owner) and eligible users to perform locally. In addition, both DO and qualified users can check the correctness of the results returned from ESP and DSP, respectively. Although the authors provide security proofs for their scheme, unfortunately, after carefully observing their scheme, we find that the scheme has security vulnerabilities. These vulnerabilities allow the adversary to generate the sub-key for any attribute and replace ciphertext sub-item, which result in the adversary to be able to break their scheme. In response to this problem, we propose an improved solution and proved its security.

INDEX TERMS Attribute-based encryption, security vulnerability, key generation, ciphertext replacement.

I. INTRODUCTION
In 2019, Li et al. [1] proposed an ABE scheme with verifiable encryption and decryption outsourcing which can securely outsource encryption and decryption to untrusted ESP and DSP, respectively. Moreover, qualified users only need to perform a constant number of simple operations locally to complete the encryption and decryption work. In addition, DO and qualified users in this scheme can verify the correctness of the results returned from ESP and DSP. The scheme provides a certain security analysis, which proves the data confidentiality and outsourcing security of the proposed access control system.

However, this solution has two security vulnerabilities. One vulnerability is that any user is able to construct the sub-key corresponding to any attribute with his own user private key. The other is that any ciphertext sub-item related to an attribute can be replaced. The two vulnerabilities result in that any user is able to make his user private key satisfy any access policy by modifying his user private key or the ciphertext related to the access policy, and then can decrypt the ciphertext of any file which he has no right of access. For specific proofs, see Chapter 3.

II. REVIEW OF THE LI ET AL.’S SCHEME
We recapitulate Li et al.’s scheme [1]. The scheme consists of the following seven algorithms.

1) Setup(U): Given U as the universe attributes, this algorithm first chooses a group $G$ of prime order $p$, a generator $g$ of $G$ and a hash function $H : \{0, 1\}^* \rightarrow \mathbb{Z}_p^*$ which will map any attribute described as binary string to a random group element in $\mathbb{Z}_p^*$. It randomly picks several exponents $\alpha, a \in \mathbb{Z}_p^*$ and assigns $(g, e(g, g)^a, g^\alpha, H)$ as the public parameter $PK$ and $g^\alpha$ as the master secret key $MSK$.

2) Keygen(PK, MSK, $I_{key}$): Given the public parameter PK, the master secret key MSK and the $I_{key} \in U$, where $I_{key}$ is an attribute set $S$. Firstly, it randomly chooses $t \in \mathbb{Z}_p^*$. Then, it generates the private secret key $SK$ as $(K = g^{\alpha} g^t, L = g^t, \forall x \in S, K_x = g^{H(\text{att}(x))t})$.

3) Encryption(PK, $M$, (M, $\rho$)): Given the public parameter PK, this algorithm is executed by DO under the cooperation of ESP. It describes a DO who wants to encrypt message
M with an LSSS access structure \((M, \rho)\), the function \(\rho\) associates rows of \(M\) to the attributes, \(M\) is an \(l \times n\) matrix. The DO chooses a random \(s \in \mathbb{Z}_p^*\) and a random vector \(v = (s, y_2, y_3, ..., y_n) \in \mathbb{Z}_p^n\), then calculates \(\{c_i = s \cdot \rho_i\}^{i=1}_{i=1}\). In addition, the DO chooses random \(r_1, ..., r_l \in \mathbb{Z}_p\), and then obtains the ciphertext \(CT = \{C = M \cdot e(g, g)^\lambda, C' = g^{v_i}, (C_i = g^{\alpha_i} g^{-r_i H(\cdot)}(i)), D_i = g^{y_i} r_i^{i=1}\}\). The first two items \((C, C')\) of \(CT\) are computed by \(DO\), and the rest of the items \((C_i, D_i)^{i=1}_{i=1}\) are computed with Algorithm 1 and Algorithm 2 under the cooperation of \(ESP\).

4) \text{Gen\text{TK}(SK)}: This algorithm takes the private key \(SK = (K, L, K_s), x \in S\) as input. Then it randomly chooses two values \(z_1, z_2 \in \mathbb{Z}_p^*\) and \(a_j \in K_s\), note that the attribute \(j \in S\) must be a necessary one to fully decrypt the \(CT\). Then it generates two transformation keys \(TK_1\) and \(TK_2\), and the corresponding retrieving keys are \(RK_1\) and \(RK_2\), respectively. Finally, the user sends the \(TK_1\) and \(TK_2\) to \(DSP_1\) and \(DSP_2\).

5) \text{Transform\text{out}(CT, TK_1, TK_2, PK)}: In this algorithm, \(DSP_i\), for \(i = 1, 2\) takes the \(PK, CT\) and the corresponding key \(TK_i\) as inputs. Then it takes a partially decrypted ciphertext \(CT[i]\), for \(i = 1, 2\) as outputs. If the attribute set \(S = Ikey\) satisfies the access structure \((M, \rho)\), let \(I \subseteq \{1, 2, ..., l\}\) be defined as \(I = \{i: \rho(i) \in S\}\), let \(\{\omega_i \in \mathbb{Z}_p\}^{i=1}_{i=1}\) be a set of constants such that \(\{c_i = \omega_i\}^{i=1}_{i=1}\) are valid shares of any secret \(s\) according to \(M\), then \(\sum \omega_i c_i = s\). After the calulations by \(DSP_1\) and \(DSP_2\), the user gets the transformed ciphertext as \(CT' = \langle CT[1], CT[2]\rangle\).

6) \text{Check\text{Correctness(RK_1, RK_2, CT_1, CT_2, PK)}: Given the CT[i] and RK[i], then it checks the correctness of the computing by \(DSP_i\), for \(i = 1, 2\). If the outputs done by \(DSP_1\) and \(DSP_2\) are correct, then it outputs \(CT^*\).

7) \text{Decryption(PK, CT, CT^*)}: After the \text{Check\text{Correctness} algorithm outputs \(CT^*\), the decryption algorithm takes the \(CT^*\) and the \(CT\) as inputs, then it computes \(M = C/CT^*\), finally it takes the message \(M\) as output.

Detailed algorithms, see original paper [1].

III. CRYPTANALYSIS

There exist two evident vulnerabilities in Li et al.’s scheme.

The first is that it is easy to obtain the sub-key \(g^{H(x)y}\) related to the attribute \(x\) from known \(g^i\) and \(H(x)\) in a user private key. The second is that it is easy to obtain the sub-item \(g^{\alpha_i}\) from known \(g^i\) and \(H(A)\) in a ciphertext. The two vulnerabilities cause that Li et al.’s scheme is vulnerable to attack.

And then we show how to attack Li et al.’s scheme. In the guess phase of two security games for proving the security of Li et al.’s scheme, the adversary can break Li et al.’s scheme with probability 1 in the following two methods.

Method 1: Generating the sub-key of any attribute. Suppose that the user’s private key queried by the adversary is:

\[SK = (K = g^a g^{at}, L = g^i, \forall x \in S, K_s = g^{H(x)y})\]

where \(S\) does not satisfy the challenger’s access structure \((M^*, \rho^*)\). But using \(g^i\) or \(g^{H(x)y}\) of the private key, the adversary can obtain the sub-key \(g^{H(x)y}\) of any attribute \(y \notin S\) by the following way:

\[(g^i)^{H(x)} = g^{H(x)y}\]

or

\[(g^{H(x)} y^{(H(\cdot))-1}) = g^{y}\].

The adversary sets up a \(S'\), which satisfies the challenger’s access structure \((M^*, \rho^*)\), and with the above way, the private key corresponding to the attribute \(y\) is generated as:

\[SK' = (K = g^a g^{at}, L = g^i, \forall x \in S', K_s = g^{H(x)y})\].

Method 2: Replacing ciphertext.

Similar to the private key, the ciphertext is easily replaced, too. Given the following ciphertext sub-item:

\[C_i = g^{\alpha_i} g^{-r_i H(\cdot)}\]

We obtain:

\[(g^i)^{H(\cdot)} = g^{-r_i H(\cdot)}\]

\[g^{\alpha_i} g^{-r_i H(\cdot)} = g^{\alpha_i}\]

\[C_i = (g^{\alpha_i})(g^{r_i})^{-H(\cdot)} = g^{\alpha_i} g^{-r_i H(\cdot)}\]

Thus, the hash value of the attribute \(A\) is replaced by the hash value of the attribute \(B\).

Therefore, the adversary can decrypt the challenger’s ciphertext to obtain the plaintext \(m_s\) by making the queried user’s private key matching the challenger’s access structure with the above two methods, and correctly give a guess \(\nu'\) of \(\nu\), i.e., \(\nu' = \nu\) with probability 1.

IV. OUR SCHEME

In response to the above-mentioned security loopholes, we revise the original plan and come up with a new plan. The key is to modify the key construction algorithm to avoid the security risks of the \(K_s\) part.

The specific improvements are as follows:

1) \text{Setup(U)}: It defines another hash function \(H_1 : \{0, 1\}^* \rightarrow G\), then it outputs the \(PK, MSK\) as follows:

\[PK = (g, e(g, g)^a, g^a, H, H_1)\]

\[MSK = g^a\]

2) \text{Keygen(PK, MSK, I_{key})}: The algorithm chooses a random \(t \in \mathbb{Z}_p^*\) and \(g^a g^{at}\). It generates the private key \(SK\) as:

\[K = g^a g^{at}, L = g^i, \forall x \in S, K_s = H_1(\rho(x))^{H(\cdot)}\]

3) \text{Encryption(PK, MSK, I_{key})}: This algorithm are executed by \(DO\) under the cooperation of \(ESP\), the \(DO\) chooses random \(r_1, ..., r_l \in \mathbb{Z}_p\), then the ciphertext \(CT = \{C = M \cdot e(g, g)^{at}, C' = g^{i}, (C_i = g^{\alpha_i} H_1(\cdot) - r_i H(\cdot)), D_i = g^{y_i} r_i^{i=1}\}\)

Note that Only the items \((D_i)^{i=1}_{i=1}\) are computed under the cooperation of \(ESP\), like the original scheme’s Algorithm 2.
4) GenTK(SK): There is no change in the form of GenTK(SK), we can still use this algorithm to perform corresponding operations on the private key SK to further complete the decryption outsourcing.

5) Transform_out(CT, TK₁, TK₂, PK): Upon receiving the CT and the TK₁, DSP₁ computes:

\[ CT_{[1]} = \sum\limits_{i \in 1, i \neq j} e(C', K) \]

\[ = \frac{e(g, H_1(j)H(j)_{\ast f_{\alpha,i}^*}, e(g, g)^{\ast s} e(g, g)^{\ast s})}{e(g, H_1(j)H(j)_{\ast f_{\alpha,i}^*}, e(g, g)^{\ast s} e(g, g)^{\ast s})} \]

\[ = \frac{e(g, H_1(j)H(j)_{\ast f_{\alpha,i}^*}, e(g, g)^{\ast s} e(g, g)^{\ast s})}{e(g, H_1(j)H(j)_{\ast f_{\alpha,i}^*}, e(g, g)^{\ast s} e(g, g)^{\ast s})} \]

in the same way, DSP₂ computes:

\[ CT_{[2]} = \sum\limits_{i \in 1, i \neq j} e(C', K) \]

\[ = \frac{e(g, H_1(j)H(j)_{\ast f_{\alpha,i}^*}, e(g, g)^{\ast s} e(g, g)^{\ast s})}{e(g, H_1(j)H(j)_{\ast f_{\alpha,i}^*}, e(g, g)^{\ast s} e(g, g)^{\ast s})} \]

After the calculations by DSP₁ and DSP₂, the user gets the transformed ciphertext as CT’ = (CT₁, CT₂).

6) CheckCorrectness(RK₁, RK₂, CT₁, CT₂, PK): After the relevant parameters change, the calculation is as follows:

\[ CT_{[1]}B_{[1]}C₀ = \sum\limits_{i \in 1, i \neq j} e(g, g)^{\ast s} e(g, H_1(j)H(j)_{\ast f_{\alpha,i}^*}, e(g, H_1(j)H(j)_{\ast f_{\alpha,i}^*}, e(g, g)^{\ast s} e(g, g)^{\ast s})} \]

\[ = \frac{e(g, H_1(j)H(j)_{\ast f_{\alpha,i}^*}, e(g, g)^{\ast s} e(g, g)^{\ast s})}{e(g, H_1(j)H(j)_{\ast f_{\alpha,i}^*}, e(g, g)^{\ast s} e(g, g)^{\ast s})} \]

\[ = \frac{e(g, H_1(j)H(j)_{\ast f_{\alpha,i}^*}, e(g, g)^{\ast s} e(g, g)^{\ast s})}{e(g, H_1(j)H(j)_{\ast f_{\alpha,i}^*}, e(g, g)^{\ast s} e(g, g)^{\ast s})} \]

7) Decryption(PK, CT, CT’): After the CheckCorrectness algorithm outputs CT’, DU calculates:

\[ \frac{C}{C_{\text{part}}} = \frac{M e(g, g)^{\ast s}}{e(g, g)^{\ast s}} = M \]

V. SECURITY PROOF
The proof of this article is based on the expansion of Waters’ scheme [2].

If Waters’ scheme can achieve targeted sCPA security, then the solution in this article can also achieve targeted sCPA security.

Proof: The scheme proposed in this article is similar to the Waters’ scheme in both ciphertext and key form. Compared with the Waters’ scheme, the scheme proposed in this paper uses \( H_1(x)^{\ast H_2(x)} \) instead of \( H(x) \) in the original scheme.

Assuming that there is an adversary A who can break the scheme in this paper with a non-negligible advantage under the deterministic q-BDHE assumption, then the simulator B can be constructed to break the Waters scheme with a non-negligible advantage.

The simulation process of the simulator B is the same as the large-attribute space structure of Waters.

Initialization: The simulator B obtains the system public key from the challenger C.

\[ PK = (g, e(g, g)^{\ast s}, e(g, g)^{\ast s}, H_1, H_2, H) \]

Note that \( H(\rho(x)) = H_1(\rho(x)^{\ast H_2(\rho(x))}) \)

And the simulator sends it to the adversary A as its own public key.

Phase 1: The adversary A sends the attribute set \( S_A \) to the simulator B, and the simulator uses \( S_A \) as its own attribute set to obtain the private key \( SK_{S_A} \) from the challenger C, and finally sends it to \( A \).

\[ SK_{S_A} = \{ K = g^{\alpha}, g^{\ast s}, L = g', \forall x \in S, K_x = H_1(\rho(x)^{\ast H_2(\rho(x))}) \} \]

Challenge: The simulator B submits two challenge messages \( M_0 \) and \( M_1 \) to the challenger C. The challenger C chooses a random \( b \in \{0, 1\} \) and encrypts \( M_b \). The challenge ciphertext is then generated as \( CT^* = \{
\]

\[ C = M_b * e(g, g)^{\ast s}, C' = g^\ast \]

\[ (C_i = g^{\alpha}H_1(\rho(i))^{\ast -rH_2(\rho(i))}, D_i = g^{r_j}j_{i=1}^t) \]

The simulator B sends \( CT^* \) to the adversary A.

Phase 2: Repeat Phase 1 query.

Guess: If the conjecture that the adversary A outputs \( b \) is \( b \), then the guess that B outputs is also \( b \).

In summary, the simulator B completely simulates the challenge of the adversary A to the Waters’ solution. If the adversary A can break the solution in this article with a non-negligible advantage, it can also break the Waters’ solution with a non-negligible advantage.

VI. CONCLUSION
We find two vulnerabilities of Li et al.’s scheme, which make their scheme vulnerable to two attacks of generating the sub-key of any attribute and replacing ciphertext. Either of two attacks enables the adversary to decrypt any ciphertext by modifying his user private key to make it match the ciphertext or modifying the ciphertext to make it match his user private key. Therefore, Li et al.’s scheme is insecure and cannot defend against selected plaintext attack and selected ciphertext attack. Finally, in response to these problems, we propose a new algorithm to improve the security of the basic scheme.
REFERENCES

[1] Z. Li, W. Li, Z. Jin, H. Zhang, and Q. Wen, “An efficient ABE scheme with verifiable outsourced encryption and decryption,” IEEE Access, vol. 7, pp. 29023–29037, 2019.

[2] B. Waters, “Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization,” in Proc. Int. Workshop Public Key Cryptogr., Berlin, Germany: Springer, 2011, pp. 53–70.

ZHEN GUO was born in Chengdu, Sichuan, China, in 1997. She received the Bachelor of Science degree from Sichuan Normal University, where she is currently pursuing the master’s degree. Her main research interests include cloud computing and information security.

BIN LIU was born in Yibin, Sichuan, China, in 1996. He received the degree in engineering from Sichuan Normal University, where he is currently pursuing the master’s degree. His main research interests include cloud computing and information security based on blockchain.

KAIQIANG ZHAO was born in Bazhong, Sichuan, China, in 1996. He received the Bachelor of Science degree from the Chengdu University of Information Technology. He is currently pursuing the master’s degree with Sichuan Normal University. His major research interests include cloud computing and information security.

CHAOSHENG FENG (Associate Member, IEEE) was born in Guangyuan, Sichuan, China, in 1971. He received the Ph.D. degree in information and communication engineering from the University of Electronic Science and Technology, Chengdu, China, in 2010. He is currently a master’s Tutor with the School of Computer Science, Sichuan Normal University. His research interests include peer-to-peer network security, malicious code analysis and defense, and cloud computing security. He is a Senior Member of the China Computer Association.