A Lightweight Outsourced Attribute-Based Encryption Algorithm for Fog Computing Postprint

Authors: Zeng Ping, Qian Jin, Mu Chengxin, plateau, Hu Ronglei

Date: 2018-12-13T00:00:00+00:00

Abstract

Attribute-based encryption algorithms possess the characteristics of flexibility, fine-grained access control, and high security. To reduce the resource consumption of attribute-based encryption algorithms, an improved attribute-based outsourced encryption algorithm is proposed based on the attribute-based encryption algorithm for secure data access control. The improved algorithm outsources the complex bilinear pairing computations in the encryption process to fog nodes, thereby reducing the computational overhead for users. Simultaneously, by simplifying system parameters and reducing the random factors generated by the attribute authority for attributes, the lengths of ciphertexts and keys are shortened, which reduces the storage and communication overhead for both users and fog nodes. Additionally, a security proof for the proposed improved algorithm is provided, demonstrating that the improved algorithm is secure.

Full Text

Preamble

Vol. 37 No. 2
Application Research of Computers
ChinaXiv Cooperative Journal

A Lightweight Attribute-Based Outsourced Encryption Algorithm for Fog Computing

Zeng Ping[1], Qian Jin[1],[2], Mu Chenxin[3], Gao Yuan[1], Hu Ronglei[1]
(1. Beijing Electronics Science & Technology Institute, Beijing 100000, China;
2. School of Communication & Engineering, Xidian University, Xi' an 710000, China;

3. Dept. of Theoretical Training, People' s Liberation Army Air Force Harbin Flight Academy, Harbin 150001, China)

Abstract: Attribute-based encryption algorithms offer flexibility, fine-grained access control, and high security. To reduce the resource consumption of attribute-based encryption, this paper proposes an improved attribute-based outsourced encryption algorithm based on secure data access control. The improved algorithm outsources complex bilinear pairing computations in the encryption process to fog nodes, thereby reducing computational overhead for users. Simultaneously, by simplifying system parameters and reducing the random factors generated by the attribute authority for each attribute, the algorithm shortens ciphertext and key lengths, lowering storage and communication overhead for both users and fog nodes. The paper also provides a security proof for the proposed algorithm, demonstrating its security.

Keywords: attribute-based encryption; outsourcing encryption; fog computing; security proof; threshold segmentation

---

0 Introduction

Cloud computing is a new computing paradigm built on resource sharing, where users transmit data requiring hosting and analysis to cloud platforms for management [**?**]. As an extension of cloud computing, fog computing migrates computation to fog nodes near the user end, effectively saving network overhead for cloud platforms and avoiding network performance bottlenecks [**?**]. However, this introduces management challenges for fog computing devices. Due to their deployment at the edge of cloud platforms, fog nodes lack continuous and effective management, making them vulnerable to attacks. Additionally, there is no effective trust mechanism between user devices and fog nodes, and fog nodes must handle access from various heterogeneous devices. The massive number of user devices accessing the network makes user management exceptionally difficult [**?**].

Ciphertext-policy attribute-based encryption (CP-ABE) schemes enable fine-grained data access control by labeling ciphertexts with attributes and specifying access policies for accessors. While this approach overcomes granularity control issues in user access, it simultaneously introduces attribute management challenges. The key generation center generates corresponding private keys based on users'attribute sets, but hiding the characteristics of user attribute sets within private keys to prevent information leakage remains a problem. Furthermore, when users update, add, or revoke attributes, the key generation center must update keys according to the new attribute set. Ensuring that changed attributes are not exposed, revoked attributes are thoroughly destroyed, and maintaining both forward and backward security for key updates are currently difficult problems requiring urgent solutions.

Attribute-based encryption was first proposed in [?]. To overcome the inflexibility of access in this scheme, [?] introduced a ciphertext-policy encryption scheme that associates attributes with ciphertexts, allowing recipients to determine whether they can decrypt based on the attribute set specified in the ciphertext. The attribute-based signature (ABS) scheme [?] can achieve authentication functions but cannot enable secure communication. While [?] proved the security of the signature algorithm in [?] under the standard model, it cannot be applied to fog computing environments with outsourced encryption and decryption. [?] improved the signature algorithm to make it applicable to outsourced encryption/decryption environments, but it cannot guarantee data security. [?] proposed an attribute-based encryption scheme evolved from a hierarchical identity-based encryption scheme that employs hybrid dual-system encryption, significantly enhancing flexibility and allowing users to specify access policies of arbitrary depth. However, the scheme involves bilinear transformations over multiple multiplicative groups, resulting in excessive computational overhead. [?] proposed a hybrid attribute-and-role encryption algorithm combining fine-grained access with flexibility, but its security is compromised. [?] presented an outsourced encryption scheme with attribute update functionality. [?] introduced an improved attribute-based encryption scheme that reduces user computational overhead by decreasing the number of computations on multiplicative cyclic groups while enabling ciphertext updates according to changes in user attribute sets. [?] proposed a more secure attribute-based encryption scheme, but as a multi-authority scheme, it involves excessive authentication interactions and is unsuitable for fog computing environments. [?] proposed an encryption scheme for fog computing environments, but its use of multiple authentication centers creates excessive communication overhead for users, making it inappropriate for mobile devices.

The main contribution of this paper is the design of an improved attribute-based encryption algorithm suitable for fog computing. The algorithm simplifies system parameters, reduces the random factors generated by the attribute authority for each attribute to shorten ciphertext and key lengths, and lowers storage and communication overhead for users and fog nodes. Additionally, the paper provides a security proof for the improved algorithm under the standard model, demonstrating its security.

---

1 Technical Foundations

a) Bilinear Transformation [?] is a mapping defined on multiplicative groups. Let p be a large prime, and define G_0 and G_1 as two multiplicative cyclic groups of order p. A mapping $e : G_0 \times G_0 \to G_1$ running on group G_0 is called bilinear when it satisfies the following three properties:

(a) **Bilinearity:** For any elements $g, h \in G_0$ and any $a, b \in Z_p$, we have $e(g^a, h^b) = e(g, h)^{ab}$.

(b) **Non-degeneracy:** When g is a generator of G_0, $e(g,g)$ is the identity element of G_1, and e does not map all element pairs of G_0 to the identity element of G_1.

(c) **Computability:** For any $g, h \in G_0$, there exists an efficient algorithm to compute $e(g,h)$.

b) Decisional Bilinear Diffie-Hellman (DBDH) Problem: Assume there exist cyclic groups G_0 and G_1 of order p, elements $g, g^a, g^b, g^c \in G_0$, and a bilinear transformation e running on the groups. Randomly select $a, b, c, z \in Z_p$. Given two tuples $(g, g^a, g^b, g^c, e(g,g)^{abc})$ and $(g, g^a, g^b, g^c, e(g,g)^z)$, where z is a uniformly distributed random bit string, the problem is to distinguish which of the two tuples is the random tuple.

---

2 Security Model

This paper employs the standard model to prove the security of the encryption scheme. The model assumes the cloud platform is secure and describes the security through a game between a challenger and an attacker. The specific process is as follows:

System Initialization: The system inputs public parameters.

Query Phase 1: The attacker $\mathcal{A}$ can query the system with arbitrary attribute sets, and the system algorithm generates corresponding private keys SK and returns them to the attacker.

Challenge Phase: The attacker provides two equal-length plaintexts m_0 and m_1 to the system and proposes an access structure T that cannot have been queried in Query Phase 1 and cannot be a subtree of any structure queried in the query phase. The system randomly selects a bit $b \in \{0,1\}$, generates a private key for structure T, encrypts plaintext m_b to obtain ciphertext CT, and returns it to the attacker.

Query Phase 2: The attacker continues to make queries to the system. In this phase, queries on structure T and any structure with T as a subtree are prohibited. The system continues to provide query information to the attacker.

Guess Phase: Based on the query results, the attacker guesses which of the plaintexts m_0 and m_1 corresponds to the ciphertext obtained in the challenge phase. The attacker outputs a corresponding bit b', and the attacker's advantage is defined as $Adv = |Pr[b' = b] - 1/2|$.

3.1 System Overview

As shown in Figure 1, the system consists of four components: the cloud platform, fog nodes, data owners, and other access users. The cloud platform is responsible for generating system parameters, determining the security level, serving as the attribute authority, scheduling fog computing node resources, processing user access authentication requests, and generating user keys. Fog nodes are responsible for communicating with users, transmitting user authentication information, completing outsourced data encryption on behalf of users, and ensuring that data can only be decrypted by users meeting certain attribute requirements. Data owners use their private keys to perform user encryption on plaintext, ensuring that ciphertext cannot be decrypted by unauthorized users. When an access user requests data access, the system first checks whether the ciphertext can be correctly decrypted. If it can, the designated fog node performs outsourced decryption. After receiving the partially decrypted ciphertext from the fog node, the access user performs user decryption to obtain the plaintext.

[Figure 1: see original paper]

The encryption scheme design process is as follows:

SysSetup(k): The attribute authority inputs a security parameter k to generate system parameters. The algorithm generates multiplicative groups G_0 and G_1 of order q, where q is a large prime and $G_0 \xrightarrow{e} G_1$. The system parameters are $\{g, h, g^{\alpha}, e(g,h)^{\alpha}, H\}$, and it generates a bilinear mapping $e : G_0 \times G_0 \to G_1$. The attribute authority selects random numbers $\alpha, \beta \in Z_q$. The platform master key is $MSK = \alpha$, and it computes the cloud platform public key $PK = g^{\alpha}$ and hash function $H : \{0,1\}^* \to G_0$.

AttrSetup($params, A$): After obtaining system parameters, the attribute authority generates key components for each attribute, where A is the attribute set. Assume A contains N attributes $\{attr_1, attr_2, ..., attr_N\}$, where $attr_i$ is a specific attribute. The attribute authority treats all attributes as distinct bit strings and computes $D_i = g^{\alpha \cdot H(attr_i)}$ for all attributes.

KeyGen($params, A_i, U_i$): The attribute authority inputs a user U_i' s attribute set A_i to generate the corresponding key. Assume the attribute set A_i is $\{U_{a1}, U_{a2}, ..., U_{an}\}$, where U_{ai} represents attributes owned by the user. The attribute authority selects a unique random number $\omega \in Z_q$ for the user and computes the user key $SK = g^{\alpha+\beta+\omega}$.

FogEncrypt(SK', T): After receiving the outsourced encryption key SK' and the user' s attribute policy T, the fog node selects a random number $m \in Z_q$. The fog node uses m as the secret splitting number and employs threshold secret sharing based on the user' s attribute policy to compute the split components $\{y_0, y_1, ..., y_n\}$. It then computes the partial ciphertext $CT' = \{g^m, g^{m \cdot D_1}, g^{m \cdot D_2}, ..., g^{m \cdot D_n}\}$.

UserEncrypt(C, CT', T): After obtaining C, CT', and T, the user performs

user encryption by selecting a random number $t \in Z_q$ and using t to symmetrically encrypt the plaintext. The ciphertext is $CT = \{C, T, T_0, T_1\}$ where $T_0 = g^{t \cdot \alpha}$ and $T_1 = h^{t \cdot \alpha}$.

FogDecrypt(CT): Fog node 2 receives CT and performs outsourced decryption. Fog node 2 uses the user' s outsourced key SK' to compute $DK = e(g,g)^{t \cdot (\alpha+\beta)}$. Starting from the leaf nodes of the access structure, it computes the parent node values:

$$F_{parent} = \prod_{x \in children} F_x^{q_{parent}(index(x))}$$

where q_{parent} is the polynomial associated with the parent node. This process recursively computes up to the root node to reconstruct the secret.

UserDecrypt(DK, CT): The user computes the symmetric key $DK' = e(g,g)^{t \cdot (\alpha+\beta+\omega)}$ and finally decrypts the ciphertext to obtain the plaintext.

The above encryption and decryption processes are illustrated in Figures 2 and 3.

[Figure 2: see original paper]
Figure 2. Schematic diagram of encryption process

[Figure 3: see original paper]
Figure 3. Schematic diagram of decryption process

3.2 Design Philosophy

To simplify encryption complexity and save overhead, encryption and decryption computations can be outsourced to fog nodes. Under the premise of ensuring security, to further simplify the encryption and decryption process outsourced to fog nodes, the original ciphertext length can be shortened by half by reducing the random factors that fog nodes add to the partial ciphertext. Simultaneously, having fog nodes specify secret components in the partial ciphertext ensures forward security.

---

4 Security Proof

Assume there exists an attacker $\mathcal{A}$ that can break the system with non-negligible advantage ε. Then there exists an algorithm $\mathcal{B}$ that can solve the DBDH problem by invoking $\mathcal{A}$. Assume the system has a total of N attribute structures.

Initialization: The system generates groups of order q, selects generators $g, h \in G_0$, and a bilinear mapping $e : G_0 \times G_0 \to G_1$. It computes g^α and g^β. There are N attributes $\{attr_1, attr_2, ..., attr_N\}$, where $N \leq q$. The challenger $\mathcal{B}$ selects

a subset $\{a_1, a_2, ..., a_M\}$ from Z_q as the corrupted attribute set, where $M \leq N$. The challenger computes $D_i = g^{a_i \cdot H(attr_i)}$, where a_i is the attribute set for structure A_i.

Query Phase 1: The attacker $\mathcal{A}$ queries the challenger $\mathcal{B}$ with an attribute structure A. The challenger maintains a table recording attribute structure A and the corresponding private key SK. This table is completely transparent to $\mathcal{A}$, who can query it at any time. The challenger processes the attacker' s queries based on different cases:

Case 1: If $A \cap A_{Corrupted} = \emptyset$, the challenger selects random numbers $\omega, \varepsilon \in Z_q$, computes the key $SK = g^{\alpha+\beta+\omega}$, and sends it to the attacker. The challenger outputs a random bit $\{0, 1\}$ and stops the attack.

Case 2: If $A \cap A_{Corrupted} \neq \emptyset$ and $A \not\subset A_{Corrupted}$, the challenger selects random numbers $\omega, \varepsilon \in Z_q$, computes the key $SK = g^{\alpha+\beta+\omega}$, and sends it to the attacker. The challenger outputs a random bit $\{0, 1\}$ and stops the attack.

Case 3: If $A \subset A_{Corrupted}$, the challenger selects random numbers $\omega, \varepsilon \in Z_q$, computes the outsourced key $SK' = g^{\alpha+\omega}$ with $C_0 = g^{a \cdot \omega}$, and sets $\{A_j\}_{j \in A} = \{g^{a_j \cdot \omega}\}$. The challenger sends the key $\{SK', C_0, \{A_j\}_{j \in A}\}$ to the attacker and records SK, SK', and C_0 in the table.

Assume this phase involves E_1 queries.

Challenge Phase: The attacker generates two equal-length plaintexts m_0 and m_1 and the attribute structure A_T it wants to challenge, sending them to the challenger. The challenger computes $D_1 = g^{a \cdot \lambda_1}$, $D_2 = g^{a \cdot \lambda_2}$, $\cdots$, $D_N = g^{a \cdot \lambda_N}$. It selects random numbers $\omega, \varepsilon \in Z_q$, computes $SK' = g^{\alpha+\omega}$, and selects a random bit $v \in \{0, 1\}$. It uses v as the secret splitting number to obtain $\{y_0, y_1, ..., y_m\}$, encrypts plaintext m_b with this key, and uses constant a as t to compute $C_0 = g^{a \cdot c}$, $C_1 = h^{a \cdot c}$. The partial ciphertext is $CT' = \{g^{a \cdot c}, g^{a \cdot c \cdot D_1}, ..., g^{a \cdot c \cdot D_N}\}$. The complete ciphertext is $CT = \{C_0, C_1, CT', SEM\}$, which is a valid ciphertext.

Query Phase 2: The attacker continues to query the challenger with structures. The challenger checks the table, and the queried structures cannot be subtrees of any structure in the table. Assume this phase involves E_2 queries.

Guess Phase: Based on the query results, the attacker guesses which plaintext the ciphertext corresponds to and outputs a bit $b' \in \{0, 1\}$.

Proof: For the attacker to successfully attack, the attributes in Query Phase 1 must be contained in the attribute set $A_{Corrupted}$. The system has N attributes total, while the challenger controls M corrupted attributes. The challenger can embed the DBDH problem into these attributes, provided the attacker' s proposed attributes do not exceed $A_{Corrupted}$. There are $\binom{N}{M}$ ways to select M attributes from N attributes, with $\binom{M}{M}$ correct cases. The probability that all queries are correct is:

$$\frac{\binom{M}{M}}{\binom{N}{M}} = \frac{1}{\binom{N}{M}}$$

Since this value is no greater than 1/2, $\mathcal{B}$' s advantage is no greater than $\varepsilon/2$.

---

5 Performance Comparison

This section compares our scheme with those in [**?**, **?**] using time and space metrics. First, we analyze the time metrics for all schemes. To hide user attribute information, each attribute must be concealed, requiring bilinear transformations for each attribute. Our scheme has similar computational complexity to [**?**] during decryption. Due to the use of outsourced encryption algorithms, most encryption and decryption computations do not need to be performed by users.

Table 1 Comparison of encryption computational cost

Scheme	Outsourced encryption	User encryption
Scheme in [11]	$(\|S\|+2)C$	$5C+2C_T$
Scheme in [12]	$(2\|S\|+2)C$	$2C+2C_T$
Our scheme	$(\|S\|+2)C$	$3C+2C_T$

Table 2 Comparison of decryption computational cost

Scheme	Outsourced decryption	User decryption
Scheme in [11]	$(2\|S\|+2)E+(T_a+2)C_T$	$(2\|S\|+1)E+2\|S\|C_T$
Scheme in [12]	$(2\|S\|+2)E+(T_a+1)C_T$	$E+2C_T$
Our scheme	$(2\|S\|+2)E+(T_a+2)C_T$	$E+2C_T$

In the scheme of [**?**], the key length is twice the number of attributes because the algorithm requires separate bilinear transformations for each attribute, with two different factors in each transformation. Consequently, both key and ciphertext lengths in this scheme double as attributes increase. In our scheme, the bilinear transformations share a common factor, saving half the storage space. In the tables, S represents the user' s attribute set, and $|S|$ denotes the number of elements in S. $|g|$ is the length of generator g in group G_0, and $|g_T|$ is the length of generator in group G_T. C and C_T represent computations on groups G_0 and G_1, respectively, while E denotes bilinear transformation computations. The results show that while maintaining computational complexity and security, our scheme significantly reduces ciphertext and key lengths to half of their original size.

Table 3 Comparison of storage overhead

Scheme	Plaintext length	Key length
Scheme in [11]	$(3\|S\|+1)\|g\|+\|g_T\|$	$(\|S\|+3)\|g\|+\|g_T\|$
Scheme in [12]	$(2\|S\|+3)\|g\|+\|g_T\|$	$(2\|S\|+3)\|g\|$
Our scheme	$(\|S\|+3)\|g\|+\|g_T\|$	$(\|S\|+3)\|g\|$

6 Conclusion

This paper improves existing ciphertext-policy attribute-based encryption schemes by simplifying the generation process of outsourced private keys for data owners and reducing the number of attribute components generated by the attribute authority. This shortens ciphertext and key lengths, saving storage and computational overhead for encryption and decryption while maintaining system security. Based on the DBDH problem, we proved the security of our scheme under the standard model. Performance analysis and security proof demonstrate that the proposed scheme achieves high efficiency while maintaining strong security.

References

[1] Cui Yong, Song Jian, Miao Congcong, et al. Research progress and trend of mobile cloud computing [J]. Journal of Computer Science, 2017, 40(2): 273-295.

[2] Fang Wei. Paradigm shift from cloud computing to fog computing [J]. Journal of Nanjing University of Information Science and Technology, 2016, 8(5): 404-414.

[3] Barrett D, Kipper G. Cloud Computing and the Forensic Challenges [J]. Virtualization and Forensics, 2010 (1): 197-209.

[4] Pirretti M, Traynor P, McDaniel P, et al. Secure attribute-based systems [J]. Journal of Computer Security, 2010, 18 (5): 799-837.

[5] Bethencourt J, Sahai A, Waters Brent. Ciphertext-Policy attribute-based encryption [C]// Proc of the 29th IEEE Symposium on Security and Privacy. Washington DC: IEEE Computer Society, 2007: 321-334.

[6] Maji H K, Prabhakaran M, Rosulek M. Attribute-based signatures [C]// Proc of the 11th International Conference on Topics in Cryptology. Berlin: Springer-Verlag, 2011: 376-392.

[7] Alex Escala, Javier Herranz, Paz Morillo. Revocable attribute-based signatures with adaptive security in the standard model [C]// Proc of the 4th

International Conference on Progress in Cryptology in Africa. Berlin: Springer-Verlag, 2011: 224-241.

[8] Li Jin, Chen Xiaofen, Li Jingwei, et al. Secure outsourced attribute-based signatures [J]. IEEE Trans on Parallel & Distributed Systems, 2014, 25 (12): 3285-3294.

[9] Lewko A, Waters B. Unbounded HIBE and attribute-based encryption [C]// Proc of the 30th International Conference on Theory and Applications of Cryptographic Techniques. Berlin: Springer-Verlag, 2011: 547-567.

[10] Jin Xin, Krishnan R, Sandhu R. A unified attribute-based access control model covering DAC, MAC and RBAC [C]// Proc of the 26th of Data and Applications Security and Privacy. Berlin: Springer, 2012: 41-55.

[11] Zhang Peng, Chen Zehong, Joseph K. Liu, et al. An efficient access control scheme with outsourcing capability and attribute update for fog computing [J]. Future Generation Computer Systems, 2018(78).

[12] Huang Qinlong, Yang Yixian, Wang Licheng. Secure data access control with ciphertext update and computation outsourcing in fog computing for Internet of things [J]. IEEE Access, 2017, 5 (99): 12941-12950.

[13] Chase M, Chow S S M. Improving privacy and security in multi-authority attribute-based encryption [C]// Proc of the 16th ACM Conference on Computer and Communications Security. New York: ACM Press, 2009: 121-130.

[14] Li Fei, Yogachandran Rahulamathavan, Muttukrishnan Rajarajan, et al. Low complexity multi-authority attribute based encryption scheme for mobile cloud computing [C]// Proc of the 7th IEEE International Symposium on Service-Oriented System Engineering. Piscataway: IEEE Press, 2013: 573-577.

[15] Dan B, Franklin M. Identity-based encryption from the weil pairing [J]. Grypto, 2003, 32(3): 213-229.

Note: Figure translations are in progress. See original paper for figures.