PLOS ONE

RESEARCH ARTICLE

Improving the proof of "Privacy-preserving attribute-keyword based data publish-subscribe service on cloud platforms"

Shangping Wang[1], Qian Zhang[1]*, Yaling Zhang[2], Jin Sun[1], Juanjuan Chen[1], Xiaoqing Sun[1]

1 School of Science, Xi'an University of Technology, Xi'an, Shaanxi, China, **2** School of Computer Science and Engineering, Xi'an University of Technology, Xi'an, Shaanxi, China

* qianz95119@126.com

Citation: Wang S, Zhang Q, Zhang Y, Sun J, Chen J, Sun X (2019) Improving the proof of "Privacy-preserving attribute-keyword based data publish-subscribe service on cloud platforms". PLoS ONE 14(2): e0212761. https://doi.org/10.1371/journal.pone.0212761

Editor: Mehmet Hadi Gunes, University of Nevada, UNITED STATES

Received: June 19, 2018

Accepted: February 7, 2019

Published: February 25, 2019

Data Availability Statement: All relevant data can be found within the paper and its Supporting Information files.

Funding: This research is supported by the National Natural Science Foundation of China (No. 61572019 http://www.nsfc.gov.cn), the Key Project of Research Foundation of Natural Science Foundation of Shaanxi Province of China (NO.2016JZ001. http://www.sninfo.gov.cn/).

Competing interests: The authors have declared that no competing interests exist.

Abstract

Most recently, Kan Yang et al. proposed an attribute-keyword based encryption scheme for data publish-subscribe service(AKPS), which is highly useful for cloud storage scenario. Unfortunately, we discover that there is a flaw in the security proof of indistinguishability of the tag and trapdoor against chosen keyword attack under the Bilinear Diffie-Hellman (BDH) assumption. As the security proof is a key component for a cryptographic scheme, based on the Decisional Diffie-Hellman (DDH) assumption, we improve the security proof method and give a new security proof of the AKPS scheme for indistinguishability of the tag and trapdoor in our proposal, which is more rigorous than the original one. Furthermore, we also demonstrate that the AKPS scheme is secure against data Replayable Chosen Ciphertext Attack (RCCA).

I. Introduction

Data publish-subscribe system [1, 2] is an appropriate mode for data users to receive data for interest. Cloud server, owing to considerable resources on storage and calculation, has been proven to be the most applicable platform for this service [3–5].

To realize fine-grained access control of data on cloud storage, the concept of the attribute-based encryption (ABE) was proposed. Generally ABE can be divided into two categories: Ciphertext- Policy Attribute-based Encryption (CP-ABE) [6, 7] and Key-Policy Attribute-based Encryption (KP-ABE) [8], both are intended for one-to-many access mode. Attribute-based encryption is an extension of public-key cryptography and identity-based cryptography. Compared with traditional cryptography, attribute-based encryption provides a more flexible encryption and decryption relationship. For example, in an attribute-based encryption mechanism, both the ciphertext and the secret key are associated with a set of attributes, and the data owner can specify an encryption policy consisting of attributes, and the resulting ciphertext can be decrypted only by the data user whose attributes satisfies the encryption policy. The non-interactive access control with fine-grained can be realized effectively by attribute-based encryption, which greatly enriches the flexibility of encryption policy and the description of

user's rights. Due to its high efficiency, dynamic, flexibility and privacy, attribute-based encryption has a good application foreground in distributed file management, third party data storage and pay-TV system. With the development and popularization of cloud computing technology, more and more enterprises and users are outsourcing their data to cloud service providers, which providing a good way to protect data security and privacy by applying attribute-based encryption.

In [9, 10], a Dual-policy ABE was created to achieve CP-ABE and KP-ABE simultaneously, and it was appropriate for data publish-subscribe system as it enables the publishers and the subscribers to define individual policy according to the feature. A publish-subscribe system should meet the following requirements. (i) The publisher publishes data and specifies which subscribers can access his/her data and the subscriber may specify the data he/she is interested in; (ii) Allow subscribers to perform multiple keyword search; (iii) Search queries that allow multiple users to retrieve data. Considering data privacy, the keyword privacy in the tag and trapdoor and the decryption overhead from the subscriber, Kan Yang proposed a privacy-preserving attribute-keyword based data publish-subscribe (AKPS) scheme [1]. Firstly, the AKPS scheme implemented the access policy and subscription policy respectively by using dual policy attribute-based encryption, which could make the subscriber's attributes satisfy the publisher's access policy and the data released by the publisher also satisfied subscription policy specified by the subscriber. In this way, the AKPS scheme could achieve the fine-grained two-way access control. Secondly, by using the concept of attribute-keyword, the subscriber's subscription policy was constructed, which was designed to avoid the leaking of keywords information and realized the multi-keyword search and the expression of subscription policy. Thirdly, attribute-keyword based data publish-subscribe scheme supported multiple publishers and multiple subscribers, and could be used to data sharing with access control in publish-subscribe system on cloud platforms. Finally, the decryption overhead was transferred from the subscribers' devices to the cloud supporter to reduce the subscriber's computational burden by using outsourcing decryption technology [11–13], which was a practical tool for lightening the computational load on the subscriber side in reality result from that mobile devices have become primary computing device for many users and enterprises.

Unfortunately, we discovered that the security proof of the AKPS scheme [1] was not enough rigorous and adequate after carefully researching it. According to the security proof of the AKPS scheme[1], we find with random guessing, the adversary can solve the Bilinear Diffie-Hellman (BDH) problem with a probability of 1/2, and their security proof actually has nothing to do with the adversary's attacking of the scheme (the detail is refer to section IV). Through analyzing the security of the original AKPS scheme [1], we also find it is hard to apply the BDH assumption to the security proof of the AKPS scheme. Therefore in order to prove the security of the chosen keyword attack for indistinguishability of the tag and the trapdoor for the AKPS scheme, we carry out a detailed analysis of the AKPS scheme and study the basic steps of the provable security method. Through careful analysis and research, we discover that the Decisional Diffie-Hellman (DDH) assumption can be used to prove the security of the AKPS scheme.

In view of this, we improve the proof method and design a new security proof of the AKPS scheme for indistinguishability of the tag and trapdoor based on the DDH assumption, our new security proof is more rigorous than their original proof. In addition, Chosen Ciphertext Attack (CCA) security [12, 14] was regarded as the appropriate security notion for encryption schemes used as components in general protocols and applications. Whereas, there exists a weaker secure notion called Replayable Chosen Ciphertext Attack (RCCA) [15] security than the CCA security, and has been proven to be sufficient for most actual intention. In this paper we prove that the data security of the AKPS scheme is of RCCA security, which is not presented in [1].

Our Contributions

(1) We show that the security proof of the AKPS scheme [1] is not enough rigorous and adequate, and we give a detail analysis about it in section IV.

(2) We improve the security proof method and present a new security proof of the AKPS scheme for indistinguishability of the tag and trapdoor based on the DDH assumption.

(3) Using the conclusion that the Waters's scheme in [6] is the selectively CPA-secure, we can prove that the AKPS scheme realizes data security against replayable chosen ciphertext attack (RCCA).

In order to make the overall layout of the system clearer, the flaw-chart of the system is shown in Fig 1.

II. Preliminaries

In this section, we present the basics of mathematics and cryptography required in the scheme, including the deterministic assumptions used in the proof, system model of the AKPS scheme and the security definition of the AKPS scheme.

A. Decisional Diffie-Hellman (DDH) assumption

Definition 1 (DDH [16]).Let $x,y,z \in Z_p$ be chosen at random and g be a generator of G. The Decisional DH assumption is that there is no probabilistic polynomial time algorithm P can distinguish the tuple $(A = g^x, B = g^y, C = g^{xy})$ from the tuple $(A = g^x, B = g^y, C = g^z)$ with more

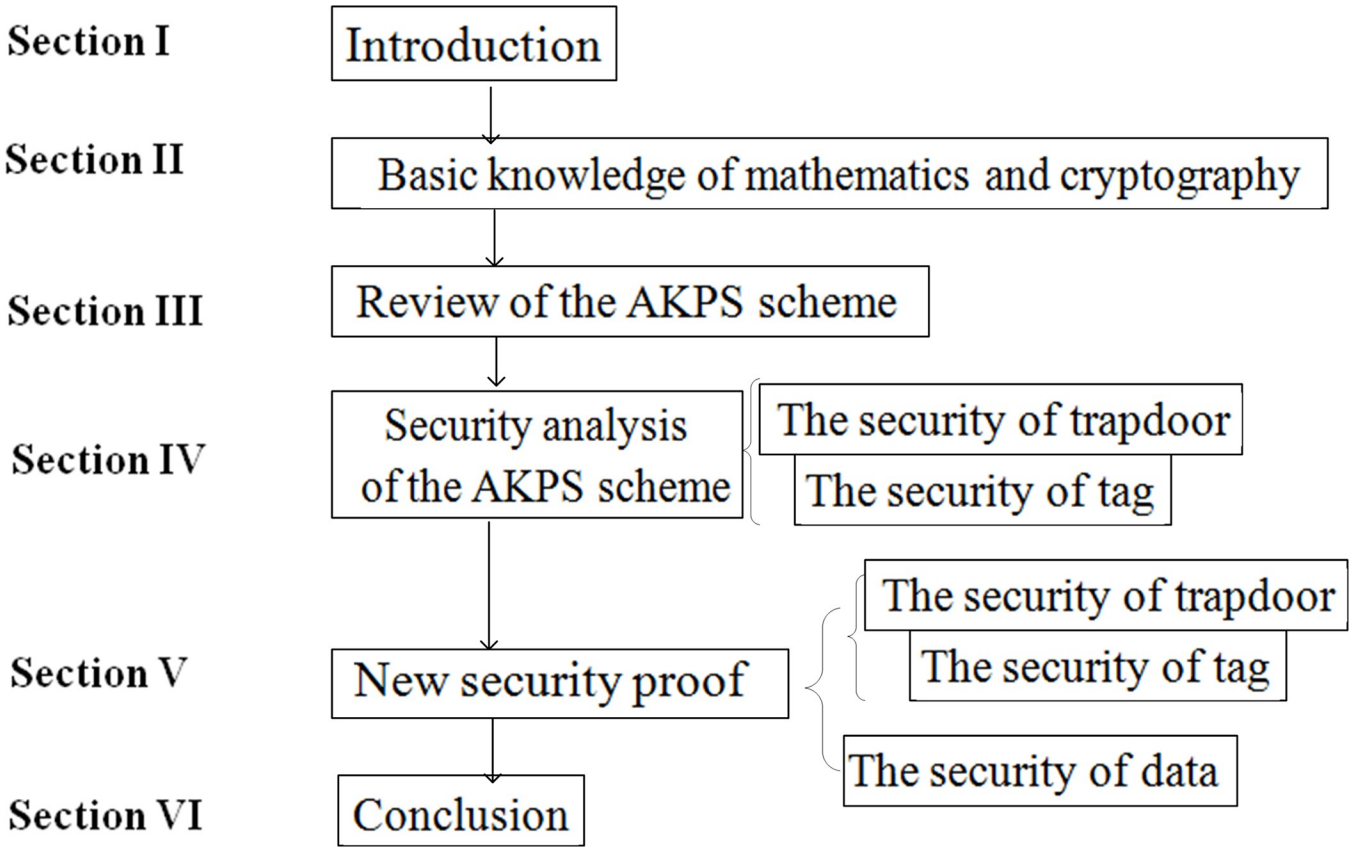

Fig 1. Flaw-chart in the system.

https://doi.org/10.1371/journal.pone.0212761.g001

than a negligible advantage ε. The advantage of P is defined as $|Pr[(A,B,g^{xy}) = 0]-Pr[A,B,g^{z}] = 0| = \varepsilon$.

B. Decision q-parallel Bilinear Diffie-Hellman Exponent (BDHE) assumption

Definition 2 (Decision q-parallel BDHE [11]). Let $a,s,b_1,\cdots,b_q \in Z_p$ be chosen randomly and g be a generator of G. If an adversary is given by

$$\overrightarrow{y} = \left(g, g^{s}, g^{\frac{1}{z}}, g^{\frac{a}{z}}, \cdots, g^{\left(\frac{a^q}{z}\right)}, g^{a}, \cdots, g^{(a^q)}, , g^{(a^{q+2})}, \cdots, g^{(a^{2q})},\right.$$

$$\forall 1 \le j \le q: \; g^{s\cdot b_j}, g^{\frac{a}{b_j}}, \cdots, g^{\left(\frac{a^q}{b_j}\right)}, \cdots, g^{(a^q)}, , g^{\left(\frac{a^{q+2}}{b_j}\right)}, \cdots, g^{\left(\frac{a^{2q}}{b_j}\right)},$$

$$\forall 1 \le j,k \le q, k \ne j: \; g^{a\cdot s\cdot b_k / b_j}, \cdots, g^{\left(a^q\cdot s\cdot b_k / b_j\right)},$$

It must be hard to distinguish a valid tuple $e(g,g)^{a^{q+1}s} \in G_T$ from a random element R in G_T. An algorithm $\mathcal{B}$ has advantage ε in solving q-parallel BDHE in G if $|\Pr[\mathcal{B}(\overrightarrow{y}, T = e(g,g)^{a^{q+1}s}] = 0 - \Pr[\mathcal{B}(\overrightarrow{y}, T = R] = 0| \ge \varepsilon$.

C. System model of AKPS

At the beginning, we give some notations that will appear in the AKPS scheme as shown in Table 1.

The system model of the data publish-subscribe service on cloud platforms as shown in Fig 2. It consists mainly of four entities: Authority, data publishers, data subscribers, and cloud server. The authority is responsible for establishing the system and generating private keys for data publishers and data subscribers, respectively. The publisher, on the one hand, encrypts data under an access policy about attributes and obtains data ciphertext; on the other hand, encrypts a set of keywords with his/her private key to generate the data tags. The data ciphertext and data tags are then uploaded to the cloud server. The subscriber defines a subscription policy for a set of keywords and uses it to generate search trapdoor, and uses his/her private

Table 1. Notations.

Notation	Description
msk	master key of the authority
pk	public parameter of the system
sk_{sub}	private key of subscriber
sk_{pub}	private key of publisher
Td_{sub}	keyword trapdoor of subscriber
pdk_{sub}	pre-decryption key of subscriber
dk_{sub}	decryption key of subscriber
S_{sub}	attribute set of subscriber
m	plaintext data
S_m	Keyword set associated with data m
C_m	encrypted data
T_m	tags associated with published data
C'_m	pre-decryption ciphertext of data m

https://doi.org/10.1371/journal.pone.0212761.t001

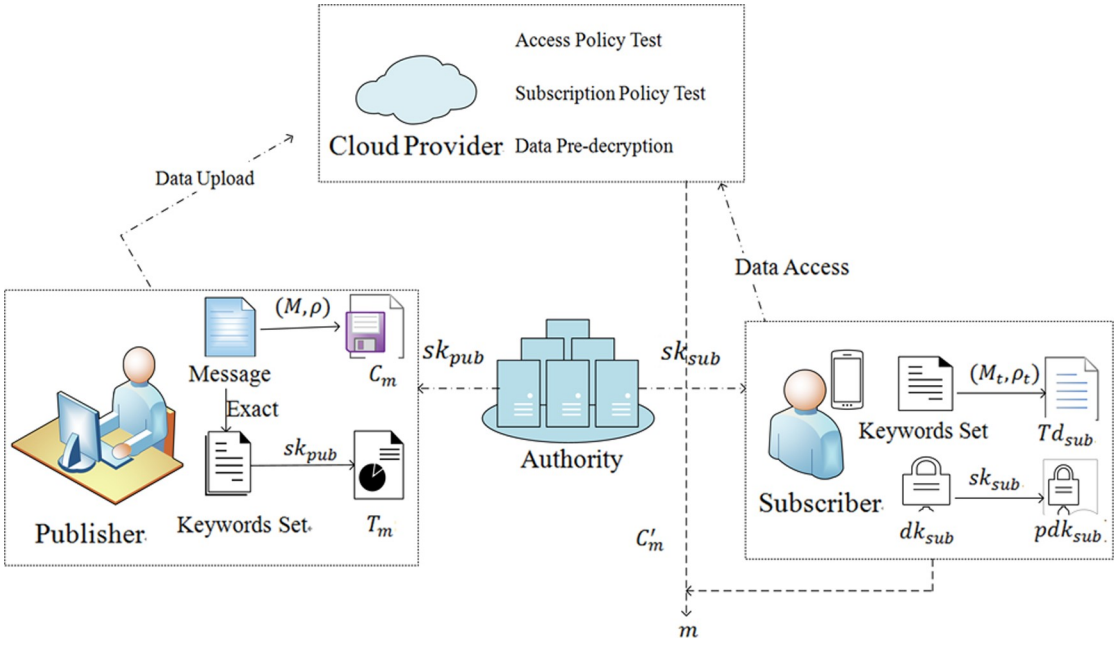

Fig 2. System model of data publish-subscribe service on cloud platforms.

https://doi.org/10.1371/journal.pone.0212761.g002

key to generate the pre-decryption key. The search trapdoor and pre-decryption key are then uploaded to the cloud server. The cloud servers can provide storage and computing services for users because of their considerable storage and computing resources. After receiving the data ciphertext and data tags from the publisher and the search trapdoor and pre-decryption key from the subscriber, the cloud server first conducts the access policy test and the subscription policy test, if and only if the subscriber's attributes satisfy the publisher's access policy and the publisher's tags satisfies the subscriber's subscription policy, the cloud server uses the subscriber's pre-decryption key to pre-decrypt the ciphertext, and sends the pre-decrypted data to the subscriber. The subscriber then decrypts the pre-decrypted data using his/her private key to get the plaintext data.

D. Security definition of AKPS

Definition 3 (Td-IND-CKA-Game).Trapdoor indistinguishability security against chosen keyword attacks (Td-IND-CKA) is described in detail in Definition 6 of [1].

Definition 4 (Tag-IND-CKA-Game). Tag indistinguishability security against chosen keyword attacks (Tag-IND-CKA) is described in detail in Definition 7 of [1].

Definition 5 (Data-RCCA [11]). The AKPS scheme is Data-RCCA secure if there is no probabilistic polynomial-time adversary who can win in Data-RCCA-Game.

Definition 6 (Data-RCCA-Game [11]).The Data-RCCA-Game involves a challenger C_Λ and an adversary A as follows.

Setup: The challenger C_Λ runs the setup algorithm and gives the public key *pk* to the adversary *A*, but does not divulge master secret key *msk*.

Phase 1: The challenger C_Λ initializes an empty set *D* and an empty table *T* respectively, setting an integer $j = 0$. The adversary *A* makes the following adaptive query to C_Λ.

Creat(S_{sub_j}) : The challenger C_Λ sets $j{:=}j{+}1$. It runs the key generation algorithm and pre-decryption algorithm on S_{sub_j} to obtain the pair $(dk_{sub_j}, pdk_{sub_j})$ and stores the entry

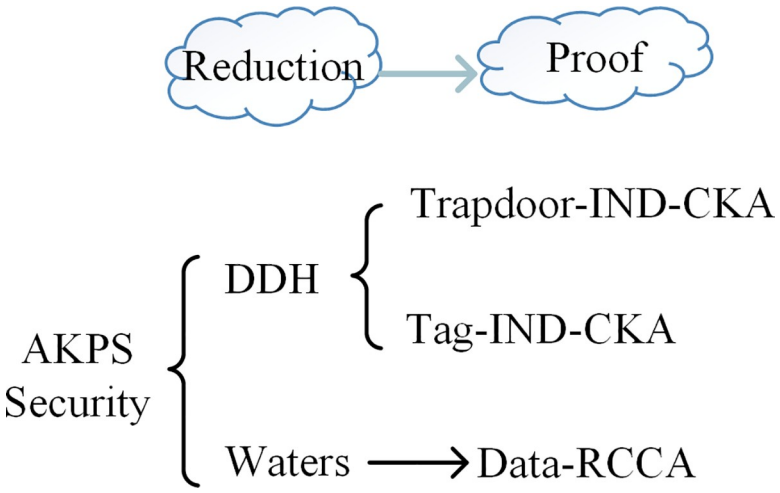

Fig 3. Security proof of the AKPS.

https://doi.org/10.1371/journal.pone.0212761.g003

$(j, S_{sub_j}, dk_{sub_j}, pdk_{sub_j})$ in table T, it then returns to the adversary A the pre-decryption key pdk_{sub_j}. Note that dk_{sub_j} is decrypt key and pdk_{sub_j} is pre-decrypt key about attribute set sub_j.

Corrupt(i): If there is an i^{th} entry $(i, S_{sub_i}, dk_{sub_i}, pdk_{sub_i})$ in T, then C_Λ sets $D := D \cup \{S_{sub_i}\}$ and returns to the adversary A the dk_{sub_i}. Otherwise, it returns the symbol $\perp$ meaning that there is no such entry.

Decrypt(i, C'_m) : If there is an i^{th} entry $(i, S_{sub_i}, dk_{sub_i}, pdk_{sub_i})$ in T, then C_Λ returns the outcome of decryption to the adversary A on takes the tuple (dk_{sub_i}, C'_m) as input. If no such entry exists, then it returns $\perp$.

Challenge: The adversary A submits two equal-length messages m_0 and m_1. In addition A submits $(\mathbf{M}^*, \rho^*)$ as the challenge access structure, where no queried $S_{sub_i} \in D$ from phase 1 fulfill it. The challenger C_Λ then flips a random coin b, and encrypts m_b under $\mathbf{M}^*$, gives the adversary with $C^*_{m_b}$.

Phase 2: The response is the same as **Phase 1** with the following restrictions:

- A cannot conduct a corrupt query that S_{sub_i} satisfies $(\mathbf{M}^*, \rho^*)$.
- no decryption queries on the message m_0 or m_1.

Guess: The adversary A outputs a guess b' of b.

We define $A's$ advantage in Data-RCCA-Game as $\mathbf{Adv}_{data} = |\Pr[b' = b] - 1/2|$.

Definition 7 (AKPS Security) The AKPS scheme is secure if it is Td-IND-CKA secure, Tag-IND-CKA secure and Data-RCCA secure.

The security of the AKPS scheme (Fig 3) encompasses the following two facets. On the one hand, Td-IND-CKA security and Tag-IND-CKA security can be reduced to the DDH assumption. On the other hand, regarding Data-RCCA security, this can be reduced to the security of Waters scheme [6].

III. Brief Review of AKPS

We briefly review the AKPS scheme below, which mainly contains five phases: System initialization, Trapdoor generation, Data publication, Policy checking and Pre-decryption, and Data decryption. For a detailed introduction to the scheme, please refer to [1].

Setup$(1^k)\rightarrow(msk,pk)$. The authority initializes the system by running the setup algorithm. It chooses two multiplicative groups G and G_T. Let g is a generator of G. Chooses two hash functions $H_1,H_2:\{0,1\}^*\rightarrow G$ and $\alpha, \beta, \gamma, a \in Z_p^*$. Then, it sets respectively the master key msk and public key pk as $msk = (g^a,\alpha,\beta,\gamma)$ and $pk = (p,g,G, G_T,e,e(g,g)^a,g^\alpha,g^\beta,g^\gamma,H_1,H_2)$.

SKeyGen$(msk,pk,\{S_{sub}\},\{pub\})\rightarrow(sk_{sub},sk_{pub})$. For each subscriber sub who has the attribute set S_{sub} and each publisher pub, the authority chooses random numbers $r_{sub}, r_{pub} \in Z_p^*$ respectively, and respectively generates the secret key sk_{sub} and sk_{pub} for each of subscribers and each of publishers as

$$sk_{sub} = (K_{1,sub} = g^{\alpha\beta r_{sub}}, K_{2,sub} = g^{\alpha r_{sub}} \cdot g^{\alpha\gamma}, K_{3,sub} = g^a \cdot g^{\gamma r_{sub}}, K_{4,sub} = g^{r_{sub}}, \forall att \in S_{sub} : K_{sub,att} = H_1(att)^{r_{sub}}).$$

$$sk_{pub} = (K_{1,pub} = g^{\alpha\beta r_{pub}}, K_{2,sub} = g^{\beta r_{pub}} \cdot g^{\beta\gamma}).$$

TdGen$(sk_{sub},pk,(\mathbf{M}_t,\rho_t))\rightarrow(Td_{sub},pkd_{sub},dk_{sub})$. To subscribe some interested data, the subscriber first defines an access structure as $(\mathbf{M}_t,\rho_t)$, where $\mathbf{M}_t$ is a $n_t\times l_t$ subscription matrix with ρ_t mapping its rows to keywords. The subscriber first generates a decryption key $dk_{sub} = z_t$ by selecting a random number $z_t \in Z_p^*$. It then selects $s_t \in Z_p^*$ and a random vector $\mathbf{v}_t = (s_t, y_{t,2}, \cdots, y_{t,l}) \in Z_p^{l_t}$, For $j = 1$ to n_t, it computes $\lambda_{t,j} = \mathbf{M}_{t,j}\cdot\mathbf{v}_t$, where $\mathbf{M}_{t,j}$ is the vector corresponding to the jth row of $\mathbf{M}_t$. It then computes $t_j = \lambda_{t,j}\cdot z_t$. Using it, subscriber designs the trapdoor as $Td_{sub} = (\mathbf{M}_t, \{j, Td_j\}_{j=1,\cdots,n_t})$

$$Td_j = (Td_{1,j} = (K_{1,sub} \cdot H_2(\rho_t(j)))^{t_j}, Td_{2,j} = (K_{2,sub})^{t_j}, Td_{3,j} = (g^\alpha)^{t_j}, Td_{4,j} = g^{t_j}).$$

In order to protect the keyword leakage from the subscription policy, only $\mathbf{M}_t$ of the subscription policy $(\mathbf{M}_t,\rho_t)$ will be sent to the cloud together with the trapdoor, while ρ_t is kept secret against the cloud server. The pre-decryption key pdk_{sub} is generated as $pdk_{sub} = (K'_{sub} = (K_{3,sub})^{z_t}, L'_{sub} = (K_{4,sub})^{z_t}, \forall att \in S_{sub} : K'_{sub,att} = (g^{\gamma s_t} \cdot K_{sub,att})^{z_t})$.

Encrypt$(m,S_m,pk,sk_{pub}, (\mathbf{M},\rho))\rightarrow(C_m,T_m)$. To publish data, the publisher first defines an access policy over attributes of subscribers. The access policy is also described by an LSSS structure as $(\mathbf{M},\rho)$, where $\mathbf{M}$ is an $n\times l$ access matrix and ρ maps the rows of $\mathbf{M}$ to attributes. The publisher then runs the following encryption algorithm to encrypt the data, which contains part of the two.

·**DataEnc**$(m,pk,(\mathbf{M},\rho))\rightarrow C_m$. First, the publisher chooses random values $s_1, s_2 \in Z_p^*$ with two vectors $\mathbf{v}_1 = (s_1, y'_2, \cdots, y'_l)$ and $\mathbf{v}_2 = (s_2, y''_2, \cdots, y''_l)$. For $i = 1$ to n, it computes $\lambda_i = \mathbf{M}_i\cdot\mathbf{v}_1$ and $\mu_i = \mathbf{M}_i\cdot\mathbf{v}_2$, where $\mathbf{M}_i$ is the vector corresponding to the ith row of $\mathbf{M}$. It outputs the ciphertext C_m as

$$C_m = ((\mathbf{M},\rho), C = m \cdot e(g,g)^{as_1}, C' = g^{s_1}, for\ i = 1\ to\ n : C_i = g^{\gamma\lambda_1} \cdot H_1(\rho(i))^{-\mu_i}, D_i = g^{\mu_i}).$$

·**TagGen**$(S_m,pk,sk_{pub},S_2)\rightarrow T_m$. The publisher takes the same random number S_2 as input, and outputs the tags as $T_m = \{W_i, T_i\}_{w_i\in S_m}$.

$$W_i = (W_{1,i} = (K_{1,pub} \cdot H_2(w_i))^{r_i}, W_{2,i} = (K_{2,pub})^{r_i}, W_{3,i} = (g^\beta)^{r_i}, W_{4,i} = g^{r_i})$$

$$T_i = (T_{1,i} = (K_{1,pub} \cdot H_2(w_i))^{r_i^*} \cdot g^{\gamma s_2}, T_{2,i} = (K_{2,pub})^{r_i^*}, T_{3,i} = (g^\beta)^{r_i^*}, T_{4,i} = g^{r_i^*})$$

PolicyTest$(C_m, T_m, Td_{sub}, pdk_{sub}) \rightarrow C'_m\ or\ \perp$. The policy test algorithm consists of both

access policy test and subscription policy test, and if and only if both policies are satisfied, the algorithm continues to pre-decrypt the data, otherwise it terminates.

·**Access policy test**. Access policy test is easier than the subscription policy test, because the attributes are not hidden in both the access policy and the pre-decryption key, while the keywords are hidden in both trapdoors and tags. Therefore, algorithm first evaluates whether the attributes of the subscriber can satisfy the access policy associated with data. If the access policy is not satisfied, the policy test algorithm will terminate.

·**Subscription policy test**. If the access policy is satisfied, it continues to test whether the tags can satisfy the subscription policy in the trapdoor by running the following subroutine:

−**KwdLocate**(T_m,Td_{sub})→I_t. Due to the obfuscation of keyword in both the trapdoor and the tags, the algorithm first locates the row number in M_t for each tag. When finished search for all the tags, it outputs an index set $I_t = \{j|\rho_t(j) = w_i, \forall w_i \in S_m\}$. To test whether the tag W_i and the trapdoor Td_j are corresponding to the same keyword, it checks whether the following equation is equal.

$$\frac{\mathrm{e}(W_{1,i}, Td_{4,j}) \cdot \mathrm{e}(Td_{2,j}, W_{3,i})}{\mathrm{e}(Td_{1,j}, W_{4,i}) \cdot \mathrm{e}(W_{2,i}, Td_{3,j})} = 1$$

·**Data pre−decryption.** If both access policy and subscription policy are satisfied, the algorithm pre-decrypts the data as follows.

−**PreDecrypt**$(C_m, pdk_{sub}, I_t) \rightarrow C'_m$ *or* $\perp$. The cloud server computes TK_1 from the trapdoor and the data tag as

$$TK_1 = \prod_{j \in I_t} \left(\frac{e(T_{1,\phi(j)}, Td_{4,j}) \cdot e(Td_{2,j}, T_{3,\phi(j)})}{e(Td_{1,j}, T_{4,\phi(j)}) \cdot e(T_{2,\phi(j)}, Td_{3,j})}\right)^{c_{t,j}} = e(g,g)^{\gamma s_2 s_t z_t}$$

Similarly, it further computes TK_2 from the ciphertext by using the pre-decryption key as

$$TK_2 = \frac{e(C', K'_{sub})}{\prod_{i \in I}(e(C_i, L'_{sub}) \cdot e(D_i, K'_{sub,\rho(i)}))^{c_i}} = \frac{e(g^{s_1}, g^a)^{z_t}}{e(g,g)^{\gamma s_2 s_t z_t}}$$

When obtaining both TK_1 and TK_2, it computes the token as $TK = TK_1 \cdot TK_2 = e(g,g)^{as_1 z_t}$. The pre-decrypted data C'_m is denoted in an Elgamal encryption form [17] as $C'_m = (C, TK) = (m \cdot e(g,g)^{as_1}, e(g,g)^{as_1})$.

Decrypt$(C'_m, dk_{sub}) \rightarrow m$. Upon receiving the pre-decrypted data, the data can be easily decrypted by the subscriber as $m = C/TK^{(1/z_t)}$.

IV. Analysis on the security proof of AKPS

In this section, we give a detail analysis about the security proof of the AKPS scheme [1], and point out the flaw of their security proof in [1]. In order to make it clearer, we will name the theorems in the AKPS scheme as Theorem 1, Theorem 2, Theorem 3, and name the security proof of our theorems in section V as Theorem 1', Theorem 2', and Theorem 3'.

The security proof of the AKPS scheme [1] is based on the hardness of the BDH problem, and we find that the security proof of the AKPS scheme is not rigorous and adequate. In the AKPS scheme, Theorem 1 is about the Td-IND-CKA security of the AKPS scheme, it is proved in the random oracle model under the BDH assumption. The BDH assumption and assumption 1 in [1] are as follows.

BDH assumption: Let A be an attacker whose running time is polynomial in a security parameter k. Given a tuple (g,g^a,g^b,g^c), where $a, b, c \in Z_p^*$. The attacker A tries to compute the

answer of the BDH problem. We define $A's$ advantage in work out the BDH problem as

$$\mathbf{Adv}_{A}^{BDH}(k) = \Pr[A(g, g^{a}, g^{b}, g^{c}) = e(g,g)^{abc}].$$

Assumption 1: The BDH problem is said to be computationally intractable if $\mathbf{Adv}_{A}^{BDH}(k)$ is negligible in k.

In their proof, the challenger $\mathcal{C}$ interacts with the adversary A to conduct the security game. Let's recall the proof procedure of Theorem 1 about Td-IND-CKA.

Firstly, the challenger $\mathcal{C}$ sets master key $msk = (g^{a},\alpha,\beta,\gamma)$ and public key $pk = (g,G,G_T, g^{\alpha},g^{\beta}, g^{\gamma})$. Then it generates a private key $sk_{sub} = (K_{1,sub}^{*} = g^{\alpha\beta r_{sub}}, K_{2,sub}^{*} = g^{\alpha r_{sub}} \cdot g^{\alpha\gamma})$ for a subscriber by selecting a random number r_{sub}. Then adversary A submits two equal- length keyword vectors $\mathbf{w}_0^{*}$ and $\mathbf{w}_1^{*}$. In addition, A also submits a challenge access policy $(\boldsymbol{M}_t^{*}, \rho_t^{*})$, which can be satisfied by both of the two keyword vectors. $\mathcal{C}$ flips a random binary coin $b\in\{0,1\}$, then selects randomly $z_t, s_t \in Z_p^{*}$ and computes share component $\lambda_{t,j} = \boldsymbol{M}_{t,j}^{*} \cdot \mathbf{v}$ and $t_j = \lambda_{t,j} \cdot z_t$. Using generated private key sk_{sub}, $\mathcal{C}$ generates the challenge trapdoor $Td_{sub}^{(b)}$ as

$$Td_{j}^{(b)} = (Td_{1,j}^{(b)} = (K_{1,sub}^{*} \cdot H_2(w_{b,j}))^{t_j}, Td_{2,j}^{(b)} = (K_{2,sub}^{*})^{t_j}, Td_{3,j}^{(b)} = (g^{\alpha})^{t_j}, Td_{4,j}^{(b)} = (g)^{t_j}).$$

In the challenge phase, the challenger $\mathcal{C}$ conducts the simulation of trapdoor corresponding the keyword vectors are submitted by the adversary A. Subsequently, the adversary A needs to compute $e(g,g)^{\beta\gamma\alpha t_j}$ as the solution to the BDH problem, where given the BDH tuple as $g^{\beta}, g^{\gamma}, g^{c} = g^{\alpha t_j}$.

After challenger $\mathcal{C}$ gives the challenge trapdoor $Td_{sub}^{(b)}$ and the input of BDH problem $g^{\beta}, g^{\gamma}, g^{c} = g^{\alpha t_j}$ to adversary A, the adversary A needs to calculate the solution of the BDH problem $e(g,g)^{\beta\gamma\alpha t_j}$ as:

$$e(g,g)^{\beta\gamma\alpha t_j} = \frac{e(Td_{2,j}^{(b)},\ g^{\beta}) \cdot e(H(w_{b',j}), Td_{4,j}^{(b)})}{e(Td_{1,j}^{(b)}, g)} = e\left(g^{\alpha\gamma t_j}, g^{\beta}\right) \cdot \frac{e(H(w_{b',j}), g^{t_j})}{e(H(w_{b,j}), g^{t_j})}$$

The adversary A then outputs a random guess b' of b, if $b' = b$, it means that the adversary A calculates the solution of the BDH problem. Since it is a random guess, the guessing probability of this case is 1/2. If $b' \neq b$, it means that the adversary A does not calculate the solution of the BDH problem, and the probability of this case is also 1/2. Therefore, the adversary A can solve the BDH problem with a probability of 1/2, but did not attack the real AKPS scheme. In other words, the adversary A has no advantage in the attack of the scheme and it does not need to interact with $\mathcal{C}$. The only thing that needs the adversary A to do is just a random guess b' of b. From the above analysis, we can obtain the conclusion that the security proof of the AKPS scheme is not adequate.

Similarly, it's the same as theorem 1 in theorem 2, which is about the security of the Tag-IND-CKA.

The theorem 3 of the AKPS scheme [1] is that the AKPS is Data-CPA secure in the random oracle if the decision q-parallel BDHE assumption holds. In Theorem 3, the outsourcing decryption technology of the AKPS scheme is based on the technique in [11]. Specifically, to enable the cloud server to pre-decrypt the data, the pre-decryption key generation algorithm is constructed by employing the technique from [11], which is proven to be semantic security against chosen plaintext attacks by using literature [6]. Similarly, the AKPS scheme can be proven to be Data-CPA secure, nevertheless the original AKPS scheme [1] dose not gives specific proof. Through researching the literature [11], we conclude that the security proof in [11] is reduced to Waters scheme [6], which is proven to be Data-RCCA secure. Therefore, using

the proof method in [6], we demonstrate Data-RCCA security on the AKPS scheme, and the specific proof can be found in Theorem 3' of Part V.

V. Improving the security Proof of AKPS

In this section, we will give our new security proof of the AKPS scheme about the security of the trapdoor and tag. In addition, we give a new security proof of the Data-RCCA security for the AKPS scheme.

Loosely speaking, the security proof of a cryptographic scheme can be described as follows. A pre-defined difficult problem is deployed by the challenger, and the adversary is supposed to attack the constructed scheme in a security game. The challenger interacts with the adversary to conduct the security game, and inserts the difficulty problem into the scheme. Only if the adversary attacks the scheme successfully with a non-negligible advantage, then the challenger can figure out the difficulty problem correctly with a non-negligible advantage. So if the difficulty problem is really hard to solve then the constructed scheme is secure. Our security proof of the AKPS scheme is based on the DDH assumption in this section. The new security proofs are given as theorem 1' and theorem 2', where theorem 1' is about Td-IND-CKA secure, and theorem 2' is about Tag-IND-CKA secure. While theorem 3' is about Data-RCCA secure which is a complete new proof for the security of AKPS scheme [1]. Our security proof is different from the original one in [1], one of the main differences is that our proof is under the DDH assumption while their proof is BDH assumption, as we cannot apply the BDH assumption to the security proof of the AKPS scheme [1].

Td-IND-CKA Security

Theorem 1'. The AKPS scheme is Td-IND-CKA secure in the random oracle model if the DDH problem is intractable.

Proof. Suppose that there is an adversary A_1 with non-negligible advantage ε_1 in the Td-INK-CKA Game against the construction of AKPS scheme in [1]. We build a simulator S_1 that can figure out the DDH problem with advantage $\varepsilon_1/2$, the simulation proceeds as follows. Let the challenger C_1 generates public parameter (e,g,G,G_T) and $G = <g>$. C_1 flips a fair binary coin $\varphi \in \{0,1\}$ outside of $S_1's$ view. if $\varphi = 1$, C_1 sets $(A,B,Z) = (g^x,g^y,g^{xy})$, else it sets $(A,B,Z) = (g^x, g^y,g^z)$, where given the values x,y,z are chosen randomly from Z_p^*.

Setup: The simulator S_1 runs **Setup**(k) algorithm, sets $msk = (\alpha,\beta,\gamma)$ for value β had known by S_1 that chosen randomly from Z_p^*, where implicitly sets $\alpha = x,\gamma = y$.Then S_1 computes public key $pk = (g^\alpha,g^\beta,g^\gamma)$ and sends it to A_1. In addition, S_1 generates respectively private key sk_{pub} and sk_{sub} for publisher and subscriber as

$$sk_{pub} = (K^*_{1,pub} = g^{\alpha\beta r_{pub}} = A^{\beta r_{pub}}, K^*_{2,pub} = g^{\beta r_{pub}} \cdot g^{\beta\gamma} = g^{\beta r_{pub}} \cdot B^\beta)$$

$$sk_{sub} = (K^*_{1,sub} = g^{\alpha\beta r_{sub}} = A^{\beta r_{sub}}, K^*_{2,sub} = g^{\alpha r_{sub}} \cdot g^{\alpha\gamma} = A^{r_{sub}} \cdot Z)$$

Each of which does not be divulged to A_1, where $r_{pub}, r_{sub} \in Z_p^*$.

Phase 1: The A_1 can adaptively query, S_1 answers queries as following.

H_2 **query**. A_1 can query the random oracle H_2. To respond to H_2 queries, S_1 maintains a list of tuple (w_i, g^{τ_i}) called the H_2 *list*, which is initially empty. When A_1 queries H_2 at a specific point $w_i \in \{0,1\}^*$, S_1 responds as follows:

—If the query w_i already appears on the H_2 *list* in a tuple (w_i, g^{τ_i}), then S_1 responds A_1 with the tuple $H_2(w_i) = g^{\tau_i}$.

—Else, S_1 chooses a random value $\tau_i \in Z_p^*$, sets $H_2(w_i) = g^{\tau_i}$ and stores the tuple (w_i, g^{τ_i}) into H_2 *list*.

Tag query. A_1 is allowed to issue queries for the tag of a set of keywords $kw = (w_1, \cdots, w_n)$, S_1 chooses randomly $s_2, r_i, r_i^* \in Z_p^*$. For each keyword w_i, using the publisher's private key sk_{pub}, S_1 makes the corresponding response $T' = \{W_i, T_i\}_{w_i \in KW}$ as

$$W_i = (W_{1,i} = (K_{1,pub}^* \cdot H_2(w_i))^{r_i} = (A^{\beta r_{pub}} \cdot g^{\tau_i})^{r_i}, W_{2,i} = (K_{2,pub}^*)^{r_i} = (g^{\beta r_{pub}} \cdot B^{\beta})^{r_i}, W_{3,i} = (g^{\beta})^{r_i}, W_{4,i} = g^{r_i})$$

$$T_i = (T_{1,i} = (K_{1,pub}^* \cdot H_2(w_i))^{r_i^*} \cdot g^{\gamma s_2} = (A^{\beta r_{pub}} \cdot H_2(w_i))^{r_i^*} \cdot B^{s_2}, T_{2,i} = (K_{2,pub}^*)^{r_i^*} = (g^{\beta r_{pub}} \cdot B^{\beta})^{r_i^*}, T_{3,i} = (g^{\beta})^{r_i^*}, T_{4,i} = g^{r_i^*})$$

Challenge: Let two equal-length keyword vectors $\mathbf{w_0} = (w_{0,1}, \cdots, w_{0,n^*})$, $\mathbf{w_1} = (w_{1,1}, \cdots, w_{1,n^*})$ submitted by the adversary A_1, which have not been queried in above phase. In addition, A_1 submits a challenge access policy $(\mathbf{M}_t^*, \rho_t^*)$, which can be satisfied by both of the two vectors. S_1 flips a random binary coin $b \in \{0,1\}$, selects randomly $z_t, s_t \in Z_p^*$ and a vector $\mathbf{v} = (s_t, y_2''', \cdots, y_n''')$. Subsequently, it computes $\lambda_{t,j} = \boldsymbol{M}_{t,j}^* \cdot \mathbf{v}$ and $t_j = \lambda_{t,j} \cdot z_t$. Using subscriber's private key sk_{sub}, S_1 generates the challenge trapdoor $Td_{sub}^{(b)}$ as

$$Td_{sub}^{(b)} = (\mathbf{M}_t^*, \{j, Td_j^{(b)}\}_{j=1,\cdots,n^*})$$

$$Td_j^{(b)} = (Td_{1,j}^{(b)} = (K_{1,sub}^* \cdot H_2(w_{b,j}))^{t_j} = (A^{\beta r_{sub}} \cdot g^{\tau_i})^{t_j}, Td_{2,j}^{(b)} = (K_{2,sub}^*)^{t_j} = (A^{r_{sub}} \cdot Z)^{t_j}, Td_{3,j}^{(b)} = (g^{\alpha})^{t_j} = A^{t_j}, Td_{4,j}^{(b)} = g^{t_j}.$$

$Td_{2,j}^{(b)}$ is the correct trapdoor component only if $Z = g^{xy}$, else $Td_{2,j}^{(b)}$ is a random element.

Phase 2: It's the same as **Phase 1**.

Guess: A_1 outputs a guess b' of b. if $b' = b$, then S_1 outputs $\varphi = 1$ to indicate that it is given a valid DDH tuple, otherwise it outputs $\varphi = 0$ to indicate that it is a random element. The advantage of S_1 to solve DDH problem is

$$\frac{1}{2} \times Pr[\varphi' = \varphi | \varphi = 1] + \frac{1}{2} \times Pr[\varphi' = \varphi | \varphi = 0] - \frac{1}{2}$$

$$= \frac{1}{2} \times \left(\frac{1}{2} + \varepsilon_1\right) + \frac{1}{2} \cdot \frac{1}{2} - \frac{1}{2} = \frac{\varepsilon_1}{2}$$

Therefore, if the A_1 has a non-negligible advantage ε_1 in the above game, then we can build a simulator S_1 which can break the DDH problem with non-negligible advantage $\varepsilon_1/2$, which is an intractable problem. Hence, the theorem 1.

B.Tag-IND-CKA Security

Theorem 2'. The AKPS scheme is Tag-IND-CKA secure in the random oracle model if the DDH problem is intractable.

Proof. Suppose that the adversary A_2 with non-negligible advantage ε_2 in the Tag-INK-CKA Game against the construction of AKPS scheme in [1]. We build a simulator S_2 that can solve the DDH problem with advantage $\varepsilon_2/2$. The simulation proceeds as follows. Let the challenger C_2 generates the parameter (e,g,G,G_T) and $G = <g>$. Then it flips a fair binary coin

$\phi \in \{0,1\}$, outside of $S_2's$ view. If $\phi = 1$, C_2 sets $(A,B,Z) = (g^x, g^y, g^{xy})$, otherwise it sets $(A,B,Z) = (g^x, g^y, g^z)$ for values x,y,z chosen randomly from Z_p^*.

Setup: The simulator S_2 runs **Setup**(k) algorithm, sets $msk = (\alpha,\beta,\gamma)$ for value α had known by S_2 chosen randomly from Z_p^*, where implicitly sets $(\beta = x, \gamma = y)$. Then S_2 computes public key $pk = (g^\alpha, g^\beta, g^\gamma)$ and sends it to A_2. In addition, S_2 generates respectively private key sk_{pub} and sk_{sub} for publisher and subscriber as

$$sk_{pub} = (K_{1,pub}^* = g^{\alpha\beta r_{pub}} = A^{\alpha r_{pub}}, K_{2,pub}^* = g^{\beta r_{pub}} \cdot g^{\beta\gamma} = A^{r_{pub}} \cdot Z)$$

$$sk_{sub} = (K_{1,sub}^* = g^{\alpha\beta r_{sub}} = A^{\alpha r_{sub}}, K_{2,sub}^* = g^{\alpha r_{sub}} \cdot g^{\alpha\gamma} = g^{\alpha r_{sub}} \cdot B^\alpha)$$

Each of which does not be divulged to A_2, where $r_{pub}, r_{sub} \in Z_p^*$.

Phase 1: The adversary A_2 can adaptively query, S_2 answers queries as following.

H_2 **query**. A_2 can query the random oracle H_2. To respond to H_2 queries, S_2 maintains a list of tuple (w_i, g^{τ_i}) called the H_2 *list*, which is initially empty. When A_2 queries H_2 at a specific point $w_i \in \{0,1\}^*$, S_2 responds as follows:

—If the query w_i already appears on the H_2 *list* in a tuple (w_i, g^{τ_i}), then S_2 responds A_2 with the tuple $H_2(w_i) = g^{\tau_i}$.

—Else, S_2 chooses a random value $\tau_i \in Z_p^*$, sets $H_2(w_i) = g^{\tau_i}$ and stores the tuple (w_i, g^{τ_i}) into H_2 *list*.

Trapdoor query: A_2 is allowed to issue queries for the trapdoor of a set of keywords kw' and a subscription policy $SP_{kw'}$ (the subscription policy is described as $(\mathbf{M}_t', \rho_t')$) constructed over kw'.

S_2 chooses randomly $z_t', s_t' \in Z_p^*$, sets vector $\mathbf{v}' = (s_t', y_2^\circ, \cdots, y_n^\circ)$ and computes $\lambda_{t,j}' = \mathbf{M}_t' \cdot \mathbf{v}'$ and $t_j' = \lambda_{t,j}' \cdot z_t'$. Then, using subscriber's private key sk_{sub}, S_2 generates the corresponding challenge trapdoor Td_{sub}' as

$$Td_{sub}' = (\mathbf{M}_t', \{j, Td_j'\}_{j=1,\cdots,n^*})$$

$$\begin{aligned} Td_j' &= (Td_{1,j}' = (K_{1,sub}^* \cdot H_2(\rho_t(j)))^{t_j'} = (A^{\alpha r_{sub}} \cdot g^{\tau_{\rho_t(j)}})^{t_j'}, Td_{2,j}' = (K_{2,sub}^*)^{t_j'} \\ &= (g^{\alpha r_{sub}} \cdot B^\alpha)^{t_j'}, Td_{3,j}' = (g^\alpha)^{t_j'}, Td_{4,j}' = g^{t_j'}. \end{aligned}$$

Challenge: Let $\mathbf{kw_0} = (w_{0,1}, \cdots, w_{0,n})$, $\mathbf{kw_1} = (w_{1,1}, \cdots, w_{1,n})$ are two equal-length keyword vectors submitted by the adversary A_2,which have not been queried in above phase. Then S_2 flips a random coin $b \in \{0,1\}$ and selects randomly $s_2, r_i, r_i^* \in Z_p^*$. Using publisher's private key sk_{pub}, S_2 generates the challenge tag $T^{(b)} = \{W_i^{(b)}, T_i^{(b)}\}_{w_{b,i} \in KW_b}$ as

$$\begin{aligned} W_i^{(b)} &= (W_{1,i}^{(b)} = (K_{1,pub}^* \cdot H_2(w_{b,i}))^{r_i} = (A^{\alpha r_{pub}} \cdot g^{\tau_{b,i}})^{r_i}, W_{2,i}^{(b)} = (K_{2,pub}^*)^{r_i} = (A^{r_{pub}} \cdot Z)^{r_i}, W_{3,i}^{(b)} \\ &= (g^\beta)^{r_i} = A^{r_i}, W_{4,i}^{(b)} = g^{r_i}) \end{aligned}$$

$$\begin{aligned} T_i^{(b)} &= (T_{1,i}^{(b)} = (K_{1,pub}^* \cdot H_2(w_i))^{r_i^*} \cdot g^{\gamma s_2} = (A^{\alpha r_{pub}} \cdot g^{\tau_{b,i}})^{r_i^*} \cdot B^{s_2}, T_{2,i}^{(b)} = (K_{2,pub}^*)^{r_i^*} \\ &= (A^{r_{pub}} \cdot Z)^{r_i^*}, T_{3,i}^{(b)} = (g^\beta)^{r_i^*} = A^{r_i^*}, T_{4,i}^{(b)} = g^{r_i^*}) \end{aligned}$$

$W_{2,i}^{(b)}$ and $T_{2,i}^{(b)}$ are the correct trapdoor component only if $Z = g^{xy}$, else the component of both are random element.

Phase 2: It's the same as **Phase 1**.

Guess: A_2 outputs a guess b' of b. if $b' = b$,then S_2 outputs $\phi = 1$ to indicate that it is given a valid DDH tuple, else it outputs $\phi = 0$ to indicate that it is a random element. The advantage of S_2 to solve the DDH problem is

$$\frac{1}{2} \times Pr[\phi' = \phi|\phi = 1] + \frac{1}{2} \times Pr[\phi' = \phi|\phi = 0] - \frac{1}{2}$$

$$= \frac{1}{2} \times \left(\frac{1}{2} + \varepsilon_2\right) + \frac{1}{2} \cdot \frac{1}{2} - \frac{1}{2} = \frac{\varepsilon_2}{2}$$

Therefore, if the A_2 has a non-negligible advantage ε_2 in the above game then we can build a simulator S_2 which can break the DDH problem with non-negligible advantage $\varepsilon_2/2$, which is an intractable problem. Hence, the theorem 2.

C.Data-RCCA Security

Theorem 3'. The AKPS scheme is RCCA secure in random oracle model assuming that the Waters scheme [6] is a selectively CPA-secure scheme.

Proof. Suppose there is a polynomial-time adversary A_3 that can attack our scheme in the selective RCCA security model with advantage ε_3. we build a simulator S_3 that can attack the Waters scheme in the selective CPA-security model with advantage ε_3 minus a negligible amount. Let C_3 be the challenger of the Waters scheme.

Init: The simulator S_3 runs A_3. A_3 chooses the challenge access structure $(\mathbf{M}^*,\rho^*)$, which S_3 sends it to the Waters challenger C_3.

Setup: S_3 queries C_3 to obtain the Waters public key $pk = (g,e(g,g)^a,g^\alpha)$ and a hash function H_1. It sends these to A_3 as the public parameters.

Phase 1: The simulator S_3 initializes an empty table T, an empty set D and an integer $j = 0$. Then S_3 responses to A_3 as follows:

Creat(S_{sub_j}) : S_3 sets j:=j+1. It has the condition of the two.

—If $(\mathbf{M}^*,\rho^*)$ be satisfied by S_{sub_j}, then it choose a "fake" pre-decryption key as follows. It chooses $d \in Z_p^*$ randomly and sends S_{sub_j} to C_3 to query the corresponding user secret key sk_{sub_j}, then set $pdk_{sub_j} = sk_{sub_j}$ and implicitly set $dk_{sub_j} = d$. where secret key pairs $(dk_{sub_j}, pdk_{sub_j})$ is incomplete, but that pdk_{sub_j} is be fittingly distributed if d was substitute for the unknown value $z_t = d/a$.

—Otherwise, S_3 sends S_{sub_j} to C_3 to request the corresponding secret key, and C_3 replies with the secret key $sk_{sub_j} = (pk, K_{3,sub_j}, K_{4,sub_j}, \{K_{sub_j,att}\}_{att \in S_{sub_j}})$. Note that we substitute the symbol of Waters scheme for the symbol of AKPS scheme, which represent the same meaning with it. The algorithm chooses random value $z_t, s_t \in Z_p^*$ where $dk_{sub_j} = z_t$, and sets pre-decryption key pdk_{sub_j} as $pdk_{sub_j} = (K'_{sub_j} = (K_{3,sub_j})^{z_t}, L'_{sub_j} = (K_{4,sub_j})^{z_t}, K_{sub_j,att} = (g^{\gamma s_t} \cdot K_{sub,att})^{z_t} : \forall att \in S_{sub})$.

Finally, store $(j, S_{sub_j}, dk_{sub_j}, pdk_{sub_j})$ in table T and return pdk_{sub_j} to A_3.

Corrupt(i): A_3 can adaptively queries any secret corresponding to the access structure expect $(\mathbf{M}^*,\rho^*)$. If there is an i^{th} entry $(i, S_{sub_i}, dk_{sub_i}, pdk_{sub_i})$ in table T, thent S_3 sets D as D:=$D \cup \{S_{sub_i}\}$. It then returns to the adversary A_3 with the dk_{sub_i}, or $\perp$ otherwise.

Decrypt(i, C'_m) : Thinking that S_3 and A_3 can obtain the pdk_{sub} values for all keys created, either can realize the pre-decryption algorithm. Therefore, we assume that ciphertexts which we obtain are already partially decrypted. Let $CT = (C,TK)$ be associated with structure $(\mathbf{M}_\chi,$

ρ_χ). Extract the entry $(i, S_{sub_i}, dk_{sub_i}, pdk_{sub_i})$ from table T. If it is not exist there or S_{sub_i} unsatisfied $(\mathbf{M}_\chi, \rho_\chi)$, return $\perp$ to A_3.

—If i^{th} does not satisfy the $(\mathbf{M}^*, \rho^*)$, obtain the records (z_t, pdk_{sub_j}) from table T, then parse it to output $m = C/TK^{(1/z_t)}$ in response of A_3 queries. If none exist, return $\perp$ to A_3.

—If i^{th} satisfy the policy $(\mathbf{M}^*, \rho^*)$, obtain the records (d, pdk_{sub_j}) from table T, then parse it to output $m = C/TK^{(1/z_t)}$ in response of A_3 queries. If none exist, return $\perp$ to A_3.

Challenge: A_3 submits two equal-length messages m_0 and m_1, S_3 sends it to challenger C_3, then the challenger C_3 flips a random coin $b \in \{0,1\}$ to obtain the challenge ciphertext $C^*_{m_b} = (C, C', \{C_i, D_i\}_{i\in[1,n]})$ under the access structure $(\mathbf{M}^*, \rho^*)$, then S_3 sends the $C^*_{m_b}$ to A_3.

Phase 2: The response is the same as **Phase 1**, expect that no decryption queries would be either m_0 or m_1.

Guess: Eventually, A_3 outputs a guess $b' \in \{0,1\}$, then S_3 outputs b'.

Hence, if the adversary A_3 can break the AKPS scheme with the given advantage ε_3, then S_3 can break the Waters scheme with the same advantage. Hence, the theorem 3.

VI. Conclusion

We analyze the security proof of AKPS scheme [1] for indistinguishability of tag and trapdoor and show that the security proof of the AKPS scheme is not rigorous and adequate, although the construction of AKPS scheme is remarkable. Based on it, we give an improving security proof of AKPS scheme for its Tag-IND-CKA security and Td-IND-CKA security based on the DDH assumption. Furthermore, by using of the conclusion that the Waters scheme in [6] is selectively CPA-secure, we manifest that the AKPS scheme realizes data replayable secure against replayable chosen ciphertext attack (RCCA), which has a higher level of security than the security of the indistinguishability of the Data-CPA in original AKPS scheme, which is mentioned but not demonstrated. Moreover, there are a number of issues that need to be studied and solved for the attribute-keyword based data publish-subscribe scheme on the cloud platforms. Firstly, new AKPS scheme should be designed to cope with the situation that a subscription policy is spelled with mistakes of interesting words, for example, 'compute' may be spelled as 'compote' or 'compue'. Secondly, for the situations where the subscriber's attributes may have been changed, such as revoke, update, increase etc, how to design efficient attribute revocation and update algorithm to realize the dynamic management of attributes, and protect the forward and backward security of the algorithm is a promising study topics. Thirdly, in publish-subscribe system, how to add the concept of time into the access policy to avoid illegal data access in the case of private key is leaked. These three aspects will be the focus of our future work.

Supporting information

S1 File. Computational cost in the AKPS scheme.
(DOCX)

S2 File. The runtime of cryptographic operations.
(DOCX)

Acknowledgments

Thanks for useful comments from anonymous reviewers.

Author Contributions

Formal analysis: Shangping Wang.

Methodology: Shangping Wang, Qian Zhang.

Validation: Yaling Zhang, Jin Sun.

Writing – original draft: Qian Zhang.

Writing – review & editing: Yaling Zhang, Juanjuan Chen, Xiaoqing Sun.

References

1. Yang K, Zhang K, Jia X, Hasan MA, Shen X. Privacy-preserving attribute-keyword based data publish-subscribe service on cloud platforms. Inf Sci. 2017; 387(C):116–31. https://doi.org/10.1016/j.ins.2016.09.020
2. Nabeel M, Appel S, Bertino E, Buchmann A. Privacy Preserving Context Aware Publish Subscribe Systems2013.
3. Muhammad Agus T, Hindersah H, Yolanda D, Hadiatna F, editors. Internet of things using publish and subscribe method cloud-based application to NFT-based hydroponic system. 2016 6th International Conference on System Engineering and Technology (ICSET); 2016 3–4 Oct. 2016.
4. Nasim R, Kassler AJ, Žarko IP, Antonic A, Antonic A. Mobile Publish/Subscribe System for Intelligent Transport Systems over a Cloud Environment. Proceedings of the 2014 International Conference on Cloud and Autonomic Computing. 2760992: IEEE Computer Society; 2014. p. 187–95.
5. Hassan MM, Hossain MA, Abdullah-Al-Wadud M, Al-Mudaihesh T, Alyahya S, Alghamdi A. A scalable and elastic cloud-assisted publish/subscribe model for IPTV video surveillance system. Cluster Computing. 2015; 18(4):1539–48. https://doi.org/10.1007/s10586-015-0476-2
6. Waters B. Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization. Proceedings of the 14th international conference on Practice and theory in public key cryptography conference on Public key cryptography; Taormina, Italy. 1964664: Springer-Verlag; 2011. p. 53–70.
7. Waters B. Ciphertext-policy attribute-based encryption. 2011.
8. Goyal V, Pandey O, Sahai A, Waters B. Attribute-based encryption for fine-grained access control of encrypted data. Proceedings of the 13th ACM conference on Computer and communications security; Alexandria, Virginia, USA. 1180418: ACM; 2006. p. 89–98.
9. Waters B. Dual System Encryption: Realizing Fully Secure IBE and HIBE under Simple Assumptions2009. 619–36 p.
10. Attrapadung N, Imai H. Dual-Policy Attribute Based Encryption2009. 168–85 p.
11. Green M, Hohenberger S, Waters B. Outsourcing the decryption of ABE ciphertexts. Proceedings of the 20th USENIX conference on Security; San Francisco, CA. 2028101: USENIX Association; 2011. p. 34-.
12. Zuo C, Shao J, Wei G, Xie M, Ji M. CCA-secure ABE with outsourced decryption for fog computing. Future Generation Computer Systems. 2018; 78:730–8. https://doi.org/10.1016/j.future.2016.10.028.
13. Wang Z, Liu W. CP-ABE with outsourced decryption and directionally hidden policy. Sec and Commun Netw. 2016; 9(14):2387–96. https://doi.org/10.1002/sec.1507
14. Li D, Chen J, Liu J, Wu Q, Liu W, editors. Efficient CCA2 Secure Revocable Multi-authority Large-Universe Attribute-Based Encryption2017; Cham: Springer International Publishing.
15. Canetti R, Krawczyk H, Nielsen J. Relaxing Chosen-Ciphertext Security2003. 565–82 p.
16. Escala A, Herold G, Kiltz E, Ràfols C, Villar J, editors. An Algebraic Framework for Diffie-Hellman Assumptions2013; Berlin, Heidelberg: Springer Berlin Heidelberg.
17. Elgamal T. A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory. 1985; 31(4):469–72. https://doi.org/10.1109/TIT.1985.1057074