Supervised Multi-Authority Scheme with Blind Signature for IoT with Attribute Based Encryption

O V Nissenbaum, K Y Ponomarov and A A Zaharov

Department of Information Security, Institute of Mathematics and Computer Science, University of Tyumen, Perekopskaya st., 15a, Tyumen, 625003, Russia

E-mail: o.v.nissenbaum@utmn.ru, k.y.ponomaryov@utmn.ru, azaharov@utmn.ru

Abstract. This article proposes a three-side cryptographic scheme for verifying device attributes with a Supervisor and a Certification Authority (CA) for attribute-based encryption. Two options are suggested: using a message authentication code and using a digital signature. The first version is suitable for networks with one CA, and the second one for networks with several CAs, including dynamic systems. Also, the addition of this scheme with a blind signature is proposed to preserve the confidentiality of the device attributes from the CA. The introduction gives a definition and a brief historical overview of attribute-based encryption (ABE), addresses the use of ABE in the Internet of Things.

1. Introduction

Attribute-based Encryption is a public key cryptographic scheme for realizing scalable and fine grained access control systems, where flexible rights can be assigned to individual users. This conception originates from the idea of Identity-based Encryption (IBE) that was introduced by A. Shamir [1] in order to ease public-key encryption and certificate management. IBE does not have necessity to authenticate user’s certificates, as user’s public key in fact their identities, such as e-mail or passport data. In this case sender can create a ciphertext under the receiver’s identity without any prior information. Afterward, A. Sahai and B. Waters [2] introduced the concept of Attribute-based Encryption (ABE) by replacing the identity with an attribute set. It turned out that ABE is quite efficient for realizing fine grained access control systems.

There are two approaches for creating ABE schemes namely Key-Policy (KP-ABE) and Ciphertext-Policy (CP-ABE). They differ by position of access policy: by the key or by the ciphertext. In a KP-ABE ciphertext is associated with attribute set, and each user private key is associated with an access structure that defines ciphertexts which could be decrypted by this key. In contrast, in CP-ABE each user’s key is associated with attribute set, and ciphertext determines access policy over attributes. More specifically, encrypted message is associated with an access structure, and user is able to decrypt this message if and only if his key attributes satisfy this access structure. The main difference between these approaches relies on who determines the access policy: sender or receiver, ciphertext or user key.
There is the specific entity named Attribute Authority (AA) in ABE, that is responsible for key management and generating private keys for users by theirs attributes. In many situations single authority cannot be responsible for the whole attributes universe. So we need to share its responsibilities between different authorities, each of them has their own attributes subset, generate and distribute private keys on its basis. Such ABE schemes are called Multi-authority ABE (MA-ABE), one of them was proposed by M. Chase in [3]. Usually MA-ABE contains main Attribute Authority that distributes system keys and acts like single point of failure. Decentralized multi-authority Attribute-based encryption system was proposed by A. Lewko and B. Walters in [4]. In such systems any party can become an authority and there is no any requirement for any coordination, only create public key and issue private keys to users that reflect new attributes set.

Attribute-based encryption schemes also classifying by its access structure: threshold [2,5], monotonic [6] or AND-gate [7]. At a threshold model users are able to decrypt only and only if they have at least t attributes among a certain universe of attributes, for some threshold t chosen by sender. Monotonic structures developed the idea of threshold parameter. They are using logical conditions to define access rules through access trees. Each non-leaf node of the three obtains threshold gate value, and if it’s equal to the number of children of this node, then node is realizing «AND» condition. When threshold value equals one, the threshold gate is «OR». Each leaf node describes certain attribute value. AND-gate access structure describes by attributes list, and secret key looks like array with specific values: positive – user has attribute, negative – user hasn’t attribute, wildcard – this attribute does not matter for message decryption.

Next, we describe the main problems that make it difficult to use ABE in practice. At many realizations question about revoking attributes is still opening, because private key does not expire. Publication [8] provides the solution to attributes revoking problem, authors introduced concept of semi-trusted authority (STA) and proxy re-encryption servers for CP-ABE. STA should store the data and on revocation event generate several proxy re-encryption keys. Proxy servers will update secret keys for all users but the one to be revoked.

It’s necessary to pay attention on the problem of key cloning. User can break access policy, just gives away their private key to another user. This problem makes sense in all known ABE schemes. The solution was proposed in work [9], the idea is that secret keys should be personalized and directly linked to a user’s personal information. So, users will think about sharing the keys because it will
reveal their personal data. To achieve it authors introduced new entities: trusted key generator (TKG) and token server (TS). TGS generate private keys and store and contains some user-specific information to create token information. TS create tokens for every decryption message event in the system, this tokens directly connected with personal information. Similar problem is possibility for users to break private key on parts and transfer them to third parties [10].

At CP-ABE schemes for that receiver will be able to successfully decrypt the message, it’s necessary to transmit access structure with ciphertext. It can be dangerous for user’s privacy, if attributes are confidential [11]. Adversary can intercept the message, get knowledge about using attributes. Based on the user’s behavior adversary is able to discover message decrypt result. By this way adversary have chances to know users attributes.

ABE schemes are quite demanded in the Internet of Things (IoT) as they allow building decentralized dynamic systems that combine many sensors and servers [12,13]. ABE has advantages for IoT environment [14]:

- fine-grained access control based on context and device attributes;
- scales independent from the number of authorized users;
- does not require key sharing or key management algorithms between the participating parties (data owner does not need to identify the destination client).

The main aspect of using ABE in IoT is performance on resource-constrained devices: execution time, memory and energy consumption. In [14] these metrics were calculated for widely used board computers: Intel Galileo, Intel Edison and Rasberry Pi. Authors evaluated CP-ABE encryption and decryption operations with monotonic access trees. For example, encryption with 15 attributes and 112 bits security level (number of bits in AES symmetric key for plaintext encryption) shows average execution time of 9.68 sec. Authors of [15] calculated the above metrics with data and network overhead additionally for Android-based smartphone devices. They evaluated CP-ABE and KP-ABE with access trees too. Performance analysis noticed that classic ABE schemes have unacceptable metrics on smartphone devices.

Work [16] focused on CP-ABE lightweight security protocol for battery-limited devices, such as laptops and smartphones. CP-ABE schemes require long decryption keys, ciphertexts, so this publication presented RSA-based ABE-model. It contains constant size secret keys and constant size ciphertexts with AND gate access structure. Also this work provides comparative summary for many ABE schemes with description of their security models, access structures and size of keys and ciphertexts. Publication [17] provides comprehensive overview of ABE implementations and shows their libraries, mathematical foundations, programming languages, descriptions. Authors noticed the lack of real-world implementations but the variety of mathematical and theoretical models.

Many researchers have been inspired by ABE and started to use it in their projects. For example, in [18] WI-FAB was proposed - Attribute-based WLAN access control mechanism without pre-shared keys. The idea is that WPA2 secret encrypt by CP-ABE and divided into several chunks. Each of them broadcasted into the networks through WLAN beacons. On the users who can rebuild the information included in beacons and retrieve secret, are authorized to connect the network.

2. Supervised Attribute Authority Scheme

Despite the fact that ABE is useful in IoT schemes, there is the problems with ABE when changing the device attributes. Consider next example: an electronic bracelet is issued to the every patient. The attributes of which are the patient’s ID, his placement (ward, bed), his doctor, information for the access to his/her medical history etc. The patient can be transferred to another ward, the bracelet can be given to another patient or transferred to another ward, etc. Some attributes should be able to change only the attending doctor, and others - the senior nurse or medical registrar. Therefore we need so-called Multi-Authority Scheme, that should be simple, fast and realized by standard cryptographic primitives.

To solve this problem we propose an attribute verification scheme with a supervisor. To verify the ABE-attributes, we enter two nodes: a Supervisor (SV) and a Certification Authority (CA). Supervisor is a person or a center with the authority to change the attributes of devices (D). There can be several
SV in the system, that corresponds with typical situation in IoT; each SV may have different authority rights. For example, it is possible to set limits on the ranges of devices, types, and attribute values that the Supervisor is able to modify. The Supervisor is registered with a Certification Authority, which stores information about the rights of this Supervisor towards Device attributes. Upon registration, the SV and CA generate a common secret key for the MAC (Message Authentication Code).

A set of Device attributes are stored and transmitted as an attr string. The process of changing device attributes occurs in the proposed protocol using digital signature algorithms (S), the authenticator (MAC), and the following notations are used:

- attr\textsubscript{old}, attr\textsubscript{new} – old and new device attributes, respectively;
- MAC(\textit{x}) – SV authenticator of the message \textit{x};
- S(\textit{x}) – CA digital signature of the message \textit{x};
- T – time stamp;
- * indicates optional fields in the transaction.

The protocol consists of the following steps (figure 3).

1. The context in which the device is located is changed - it is necessary to get a new secret key to read messages intended for new attributes. The Device sends to the Supervisor a signed old and (if the change is initiated by the Device itself) new attribute values:
   \[ D \rightarrow \text{attr}_{\text{old}}, S(\text{attr}_{\text{old}}), \text{attr}_{\text{new}}^* \rightarrow SV. \]

2. The Supervisor verifies the signature of old attributes to make sure they are correct. If not, it refuse to continue a transaction. If signature is correct and Supervisor is agree with changes in attributes and Supervisor is allowed to make these changes then S adds a time stamp and calculates authenticator of the whole resulting message. Then S sends to CA this message and authenticator:
   \[ SV \rightarrow \text{attr}_{\text{old}}, S(\text{attr}_{\text{old}}), \text{attr}_{\text{new}}, T, \text{MAC}(\text{attr}_{\text{old}}, S(\text{attr}_{\text{old}}), \text{attr}_{\text{new}}, T) \rightarrow CA. \]

3. Each Supervisor is limited by his rights to modify or confirm attribute changes. If the Supervisor has exceeded his authority, or the authenticator and (or) time stamp is incorrect, CA responds with a refusal. Otherwise, it signs a set of new attributes and passes them and a signature of attr\textsubscript{new} to the Device:
   \[ CA \rightarrow \text{attr}_{\text{new}}, S(\text{attr}_{\text{new}}) \rightarrow D. \]

In case the Device needs to change several attributes belonging to different supervisors, steps 1-3 are repeated sequentially for each of the supervisors.

![Figure 3. Supervised Attribute Authority Protocol](image-url)
4. Further, according to KP-ABE or CP-ABE paradigm, the Device presents attributes and digital signature of CA to the trusted center AA:

\[ D \rightarrow \text{attr}_{\text{new}}, S(\text{attr}_{\text{new}}) \rightarrow AA. \]

5. Finally, AA verifies the signature and, if it's correct, begins a pass of the secret key to Device using some appropriate protocol.

Note that scheme above can be implemented in decentralized environment with several CAs using cross-certificates to verify each other’s signatures. Also various trusted centers can respond for different sets of attributes, so the construction of MA-ABE (Multi-Authority ABE) schemes is in progress.

It should be noted that MAC is evaluated and verified using a secret key, which must be unique for each pair (SV, CA), so in the case of a distributed system with multiple SVs and CAs, it is advisable to replace it with the digital signature of the supervisor \( S_{SV} \). Then steps 1-3 of the protocol above will look like this:

1. \( D \rightarrow \text{attr}_{\text{old}}, S(\text{attr}_{\text{old}}), \text{attr}_{\text{new}} \rightarrow SV; \)
2. \( SV \rightarrow \text{attr}_{\text{old}}, S(\text{attr}_{\text{old}}), \text{attr}_{\text{new}}, T, S_{SV}(\text{attr}_{\text{old}}, S(\text{attr}_{\text{old}}), \text{attr}_{\text{new}}, T) \rightarrow CA; \)
3. \( CA \rightarrow \text{attr}_{\text{new}}, S(\text{attr}_{\text{new}}) \rightarrow D. \)

The digital signature, although calculated slower than MAC and has a larger size, is a public-key algorithm and therefore only one open-secret key is sufficient for each SV. Note that in proposed scheme it's possible to merge CA and AA, as well as the presence of several SV and CA.

In this scheme, as in most ABE schemes, the set of attributes becomes known to third parties: the Certification Authority, even if encrypted communication channels are used. To hide the set of attributes from CA, we propose to use a blind-based approach [19-21].

3. Blind Signature in Supervised Attribute Authority Scheme

Now we will describe the interaction between the Supervisor and the CA in the framework of the new approach, that is, we complete the scheme given above. Now we assume that attributes are confidential and should not be known to anyone other than the Device, AA and Supervisor, hence all communication channels are considered protected. The purpose of the proposed modification of the protocol is to hide the attribute values from the CA. We assume that there is \( K \) attributes in \( \text{attr}_{\text{string}} \), the supervisor has the right to confirm the changes in \( k \) of them, each of which has \( n_1, n_2, ..., n_k \) possible values, respectively.

1. The supervisor creates \( l \) identical sets of attributes, each set contains \( d \) different attribute values \( (d \leq n_1 \cdot n_2 \cdot ... \cdot n_k) \), and \( \text{attr}_{\text{new}} \) among them. All the changed attributes in the set should be available to the supervisor within the scope of his rights. In each set, Supervisor mixes the attributes randomly and encrypts them according to a blind signature scheme (for example, using a blinding factor if Chaum scheme is used [19]). The encryption keys are random, one-time and are different for each set.
2. By second step of the protocol above, instead of the \( \text{attr}_{\text{new}} \) the Supervisor sends all sets of encrypted attributes.
3. CA randomly selects the set to sign and asks secret keys of others.
4. SV sends to CA all the secret keys except one and the position of \( \text{attr}_{\text{new}} \) in the encrypted set.
5. After receiving the keys from the SV and decrypting the \( l-1 \) set, the CA makes sure that all the sets contain the same attribute values that are within the SV rights. Probability of correctness of encrypted set is \( (l-1)/l \).

If all the parameters are correct, CA signs blindly the specified value in the selected encrypted set and sends a signature to SV.
6. SV decrypts the received blind signature and in this way receives a CAs digital signature of attributes \( \text{attr}_{\text{new}} \). Afterwards instead of step 3 of the protocol above, transmits \( \text{attr}_{\text{new}} \) and signature to the device.
Thus, although CA is convinced with a probability \((l-1)/l\) that Supervisor has not exceeded his authority, he does not recognize these new attributes of the device. From his point of view, there are \(d\) possible values of attributes.

Note that in the above approach there should be a lot of attributes, they must be independent of each other and have a large set of valid values. This option is often not always achievable in practice. Also not only one attribute should be of particular importance for an attacker, since it is easy to guess, and a certain aggregate with the condition of independence of attributes from each other.

We just described a scheme in brief. Let describe a process in detail.

3.1. Blinding of Attributes (Steps 1-2 of Blind Signature Protocol)

Let \(\text{attr} = [\text{attr}_1, \text{attr}_2, \ldots, \text{attr}_d]\) is a set of random attributes, containing true attribute \(\text{attr}_{\text{new}}\), where \(\text{attr}_m = \text{attr}_{\text{new}}\), and \(m \in \{1, 2, \ldots, d\}\) selected randomly too. Every array element in \(\text{attr}\) is possible modification of \(\text{attr}_{old}\) that Supervisor allowed to perform.

![Figure 4. Blinding of attributes in blind signature (steps 1-2).](image)

After creation of \(\text{attr}\) set Supervisor forms \(l\) random permutations \((i_{11}, i_{12}, \ldots, i_{1d}), (i_{21}, i_{22}, \ldots, i_{2d}), \ldots, (i_l, i_{l2}, \ldots, i_{ld})\) and makes \(l\) shuffled arrays

\[
P_1 = (\text{attr}_{i_{11}}, \text{attr}_{i_{12}}, \ldots, \text{attr}_{i_{1d}}),
\]

\[
P_2 = (\text{attr}_{i_{21}}, \text{attr}_{i_{22}}, \ldots, \text{attr}_{i_{2d}}),
\]

\[
\ldots
\]

\[
P_l = (\text{attr}_{i_{l1}}, \text{attr}_{i_{l2}}, \ldots, \text{attr}_{i_{ld}})
\]
through implementation them to attribute array.

Afterwards Supervisor creates \( l \) random secret keys \( key_1, key_2, ..., key_l \) and encrypts every single \( P_j \) (\( j=1, ..., l \)) by corresponding key \( key_j \) using a symmetric-key encryption algorithm \( E \) (that commutates with digital signature \( S \) according the blind signature principle: \( S(E(x)) = E(S(x)) \forall x \)). Note that every array element of \( P_j \) is encrypted separately:

\[
C_1 = E_{key_1}(P_1) = (E_{key_1}(attr_{11}), E_{key_1}(attr_{12}), ..., E_{key_1}(attr_{1d}))
\]

\[
C_2 = E_{key_2}(P_2) = (E_{key_2}(attr_{21}), E_{key_2}(attr_{22}), ..., E_{key_2}(attr_{2d}))
\]

\[
\vdots
\]

\[
C_l = E_{key_l}(P_l) = (E_{key_l}(attr_{l1}), E_{key_l}(attr_{l2}), ..., E_{key_l}(attr_{ld})).
\]

Set \( C_1, C_2, ..., C_l \) is entity Supervisor creates on Step 1 and sends to CA on Step 2 of protocol above.

3.2. Blind Selection of Signed Element (Steps 3-5 of Blind Signature Protocol)

CA receives a set \( C_1, C_2, ..., C_l \) and randomly selects index \( x \in \{1, 2, ..., l\} \) that indicates array CA going to sign. CA sends \( x \) to Supervisor who respond wit set of secret keys \( key_1, ..., key_{x-1}, key_{x+1}, ..., key_l \) (excluding \( key_x \)) and index of true attributes in selected array \( y: i_y = m \).

CA decrypts every single \( C_j \) (\( j=1,...,x-1,x+1,...,l \)) with corresponding \( key_j \) and every array element of \( C_j \) decrypted separately. CA make sure that every decrypted array is some permutation of each other and every array element is legal modification of \( attr_{old} \). If not, CA refuses to continue this protocol and sends to Supervisor message about error.

3.3. Blinding Signing and Unmask (Steps 5-6 of Blind Signature Protocol)

If all decrypted arrays are correct, CA picks true encrypted attributes \( E_{key_x}(attr_{new}) = E_{key_x}(attr_{ixy}) = C_x[y] \) and digitally signed it. Then CA sends a resulted digital signature \( S(E_{key_x}(attr_{new})) \) to Supervisor.

Finally Supervisor decrypts received value using decryption algorithm \( D \) with secret key \( key_x \):

\( D_{key_x}(S(E_{key_x}(attr_{new}))) = S(attr_{new}) \). Afterwards Supervisor verifies resulting signature with open key of CA to acknowledges signature is correct. If yes, Supervisor pass attributes and signature to Device.

3.4. Security

A proposed Blind Signature protocol has vulnerabilities typical for all Blind signature schemes and provide to Supervisor some possibilities for overwhelming his boundary.

First of possible attack is follows. If Supervisor has a possibility of multiple tries in case of discrepancy on Step 5 (if some of array is different with other and/or has attributes out of Supervisor's rights), it may purposely sent exceeding attribute inside one custom array in hope it will not be decrypted. To avoid this vulnerability there must be logging and limited tries for each supervisor.
Second vulnerability is relevant if Chaum scheme if used. Since RSA signature in Chaum scheme is used, Supervisor can attempt to find several legal attributes \( attr_{new}^*, \ attr_{new}^{**} \) so that their product modulo RSA will provide \( attr_{new} \) that transcends Supervisor's authority. Then after Supervisor receives CA's signatures of \( attr_{new}^* \) and \( attr_{new}^{**} \) it can calculate CA's signature of \( attr_{new} \) by multiplying them modulo RSA.

We cannot avoid this vulnerability because in Chaum blind signature scheme the signer signs the message directly. By contrast, in an usual signature scheme the signer would typically use a cryptographic hash function applied to the message, instead of signing the message itself. But according blind signatures, implementation of hash-function would produce an incorrect value when unblinded. Probability of this attack is definitely small if attribute notation is redundant and therefore it's difficult to find appropriate false attribute values. Anyway, we do not recommend use of RSA signature. Much better are ElGamal-based schemes like [20,21].

4. Conclusion

A supervised attribute-authority protocol with optional blind signatures for verification of device attributes for use in the Internet of Things is proposed. A blind signature scheme for attributes is intended to hide new attribute values from the Center Authority which may be useful if the device attributes contain confidential information, such as personal or medical data. Brief security analysis of proposed protocols is done.

5. References

[1] Shamir A 1984 Proc. Of CRYPTO 84. LNSC 196 47–53 Springer
[2] Sahai A and Walters B 2005 Eurocrypt 3494 457–73 Springer
[3] Chase M 2007 Proc. of the 4th Theory of Cryptography Conference 515–34 Springer, Berlin, Heidelberg
[4] Lewko A and Walters B 2011 Proc. of the 30th Annual international conference on Theory and application of cryptographic techniques: advances in cryptology 568–88. Springer-Verlag
[5] Herranz J, Laguillaumie F and Rafols C 2010 Public Key Cryptography - PKC 19–34. Springer
[6] Goyal V, Pandey O, Sahai A and Waters B 2006 Proc. of the 13th ACM conference on Computer and Communications Security 89–98 ACM
[7] Cheung L and Newport C 2007 Proc. ACM Conference on Computer and Communications Security (CCS) 456–65 ACM
[8] Shusheng Y, Wang C and Ren K 2010 Proc. of the 5th ACM Symposium on Information, Computer and Communication Security 261-70 ACM
[9] Jiang S, Safavi-Naini R and Shahandashti S.F. 2012 International Journal of Applied Cryptography 2(3) 250–70
[10] Jiang Y, Susilo W, Mu Y and Guo F 2017 Future Generation Computer Systems. Elsevier
[11] Hishide T, Yoneyama K and Ohta K 2008 Attribute-Based Encryption with Partially Hidden Encryptor-Specified Access Structures ACNS 111–29 Springer, Berlin, Heidelberg
[12] Internet of Things Global Standards Initiative Available at: http://www.itu.int/en/ITU-T/oui/iot/Pages/default.aspx
[13] Lee J, Oh S and Wook Jang J 2015 The 10th International Conference on Future Networks and Communications, Procedia Computer Science 56 271–75. Elsevier
[14] Ambrosin M, Anzanpour A, Conti M, Dargahi T, Moosavi S R, Rahmani A M and Liljeberg P 2016 IEEE Micro 36(6) 25–35
[15] Wang X, Zhang J, Schooker E M and Ion M 2014 IEEE International Conference on Communications
[16] Odelu V, Das A K, Khan M K, Choo K R and Jo M 2017 IEEE Access 5 3273–83
[17] Zickau S, Thatmann D, Butyrtschik A, Denisow I and Kupper A 2016 *19th International ICIN Conference*

[18] Pisa C, Caponi A, Dargahi T, Bianchi G and Blefari-Melazzi N 2016 *HotPOST16* 31–36, Paderborn, Germany

[19] Chaum D 1983 *Advances in cryptology* 199–203 Springer US

[20] Camenisch J, Piveteau J M and Stadler M 1995 *Advances in Cryptology - EUROCRYPT'94* 428–32 Springer Berlin/Heidelberg

[21] Chakraborty K and Mehta J 2012 *IJ Network Security* **14**(6) 316–19