An Attribute-Based Signcryption Scheme to Secure Attribute-Defined Multicast Communications

Chunqiang Hu[1], Xiuzhen Cheng[1], Zhi Tian[2], Jiguo Yu[3](✉), Kemal Akkaya[4], and Limin Sun[5]

[1] Department of Computer Science, George Washington University, Washington, DC, USA
{chu,cheng}@gwu.edu
[2] Electrical and Computer Engineering Department, George Mason University, Fairfax, USA
ztian1@gmu.edu
[3] School of Information Science and Engineering, Qufu Normal University, Qufu, China
jiguoyu@sina.com
[4] Electrical and Computer Engineering Department, Florida International University, Miami, USA
kakkaya@fiu.edu
[5] Beijing Key Laboratory of IOT Information Security Technology, Institute of Information Engineering, CAS, Beijing, China
sunlimin@iie.ac.cn

Abstract. We consider a special type of multicast communications existing in many emerging applications such as smart grids, social networks, and body area networks, in which the multicast destinations are specified by an access structure defined by the data source based on a set of attributes and carried by the multicast message. A challenging issue is to secure these multicast communications to address the prevalent security and privacy concerns, i.e., to provide access control, data encryption, and authentication to ensure message integrity and confidentiality. To achieve this objective, we present a signcryption scheme called CP_ABSC based on Ciphertext-Policy Attribute Based Encryption (CP_ABE) [2] in this paper. CP_ABSC provides algorithms for key management, signcryption, and designcryption. It can be used to signcrypt a message/data based on the access rights specified by the message/data itself. A multicast destination can designcrypt a ciphertext if and only if it possesses the attributes required by the access structure of the data. Thus CP_ABSC effectively defines a multicast group based on the access rights of the data. CP_ABSC provides collusion attack resistance, message authentication, forgery prevention, and confidentiality. It can be easily applied to secure push-based multicasts where the data is pushed from the source to multiple destinations and pull-based multicasts where the data is downloaded from a repository by multiple destinations. Compared to CP_ABE, CP_ABSC combines encryption with signature at a lower computational cost for signcryption and a slightly higher cost in designcryption for signature verification.

B. Thuraisingham et al. (Eds.): SecureComm 2015, LNICST 164, pp. 418–437, 2015.
DOI: 10.1007/978-3-319-28865-9_23

Keywords: Ciphertext-Policy Attribute Based Signcryption · Secure multicast communications · Push-based multicast · Pull-based multicast

1 Introduction

We consider a special type of multicast communications existing in emerging applications such as smart grids, social networks, and body area networks: a multicast message carries an access structure specified by the data source based on a set of attributes to define the right set of destinations - a recipient of the message can read the data only if it possesses the set of attributes required by the data source. Such multicasts can be either *push-based* or *pull-based*. For examples, a service provider in smart metering can employ push-based multicast to deliver a software update command to the smart meters of model A or B located at a certain area manufactured by company X after the year Y and the message carries an access structure defined by attributes {*location, time, company, model*} based on the AND and OR relations; a smart meter reading together with its access policy (e.g., only the service providers in Washington DC or Bethesda MD can access this data), again defined by AND and OR relations, can be stored in a data repository for future downloads (being pulled) by the service providers designated by the attributes (e.g., service providers in Washington DC or Bethesda MD).

Push-based multicasts under our consideration are very similar to the traditional ones except that no identities of the destinations are carried by the message; pull-based multicasts require the data to be stored in a repository and then downloaded by multiple users on-demand. Both multicast scenarios require the data to be protected for confidentiality, integrity, authentication, and access control. Specifically,

- All the multicast messages must be protected from adversaries as the data may disclose private information of the data source. For example, the electricity usage data could reveal the activities of the residents in a household [6], which places a significant privacy concern.
- The data source should provide access control and intelligently determine who should or should not have access to its data. An access structure should be defined based on the attributes required by the data source. The data should be accessible only by the destinations specified by the data source; no third party including the data repository should be able to read the data.
- The authenticity of the data source and the integrity of the data should be verifiable.

To achieve these objectives, we propose a signcryption scheme termed CP_ABSC based on Ciphertext-Policy Attribute-based Encryption (CP_ABE) [2] to address the secure multicast problem and provide the required security services mentioned above. CP_ABSC combines signature and encryption, and provides a new mechanism for data encryption, access control, and authentication to ensure security and privacy. The basic idea of CP_ABSC is to signcrypt a data item based on its access policy (represented by an access tree and specified by the data (data source) itself) and designcrypt the corresponding ciphertext with a secret key computed from a set of attributes. The access tree defines the access rights of the data based on the attributes and is carried by the ciphertext. This implies that any user possessing the set of attributes that satisfy the access policy defined by the data itself can access the data. Because a multicast group is uniquely defined by the data itself via the access policy, secure multicasts are effectively achieved. Moreover, other than supporting the traditional push-based multicast that "pushes" the data to all destinations, CP_ABSC can also support pull-based multicast, in which the data is stored in a repository and delivered to a multicast destination only when the destination needs the data and actively "pulls" the data.

The contributions of this paper can be summarized as follows:

– We develop a novel scheme called Ciphertext-Policy Attribute Based Signcryption (CP_ABSC) based on CP_ABE, which ensures security and privacy of the data by combining signature and encryption without requiring a certificate for verification.
– We prove the correctness of the proposed scheme and analyze its efficiency and feasibility. In particular, we discuss the security of the proposed scheme under four major attack scenarios: collusion, message authentication, forgery, and confidentiality. We also conduct a quantitative performance analysis, and our results indicate that the proposed CP_ABSC is efficient and feasible.
– We demonstrate how to apply the proposed signcryption scheme to secure different multicast communications in smart grids. Particularly, we develop a protocol to secure the instructions sent from utility companies to smart meters (push-based multicast); we also develop a procedure for the smart meter data to be securely stored and accessed by different service providers based on CP_ABSC (pull-based multicast).

The remainder of this paper is structured as follows: In Section 2, we present the motivations, our system model, and the most related work. Section 3 proposes our signcryption scheme CP_ABSC and illustrates how to use it to secure multicast communications. Section 4 proves the correctness of CP_ABSC and analyzes its security strength and computational cost. Conclusions and future research are presented in Section 5.

2 Motivations, System Model, and Related Work

In this section, we describe a few real world applications to motivate our problem formulation, present our system model, and then summarize the most related research.

2.1 Push-Based Multicast Communications

Traditional multicast communications are usually *push-based*, in which the data source pushes the data to all recipients (the multicast destinations) whose

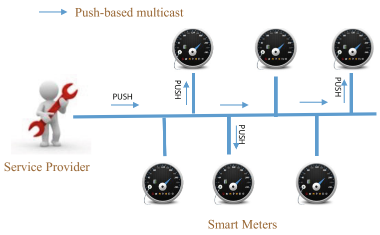

Fig. 1. Commands broadcast in smart metering

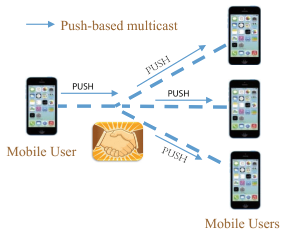

Fig. 2. Friend discovery in Social networks

identities are unique and known to the source ahead of time via one or more simultaneous transmissions. In this study, we consider a variation of the traditional multicast, in which the destinations are defined based on a set of attributes, i.e., the destinations must possess certain attributes in order to receive a multicast message. Such multicasts are popular in emerging applications such as smart grids and social networks.

Fig.1 illustrates a push-based multicast in smart metering, in which a service provider sends instructions or commands to a group of smart meters specified by their locations, models, the connected smart devices, and other attributes. For example, a service provider may broadcast a critical software update message to all smart meters at the Inverness Village whose connected devices include the smart fridges with model number 00000 or 11111 manufactured by XYZ company. This multicast message does not need to specify the identities of the smart meters (and smart devices); instead, it carries the following access structure defined by AND and OR relations: *Inverness Village AND smart fridges AND manufactured by XYZ company AND (model 00000 OR model 11111)*. Such an access structure clearly specifies the set of destinations that should receive the multicast message - it may not be practical to include a unique identity for each device in the multicast message. A similar scenario is observed in friend discovery in mobile social networks (see Fig. 2), in which a user who wants to make friends who share similar interests (reading certain types of novels, traveling to the east coast, enjoying sea food, etc.) broadcasts a query message carrying an access structure that specifies the type of friends the user is looking for.

These applications require a secure push-based multicast that can provide *access control* (not every recipient should be able to access the content of the message), *data encryption* (the query or the instruction should be kept confidential), and *authentication* (the data source should be verifiable and the data integrity should be protected) to ensure message integrity and confidentiality. But unfortunatley push-based multicast authentication schemes such as TELSA, Biba, HORS, and OTS [8,10,13–16,20] focus on authentication while ignoring access control and confidentiality. Moreover, the multicast destinations in our problem are defined by an access structure specified by the data source, which renders many popular secure multicast protocols inapplicable.

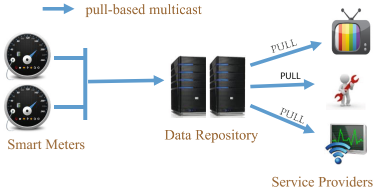

Fig. 3. Pull-based Multicast Communications in Smart Grid

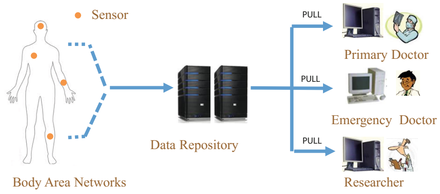

Fig. 4. Pull-based Multicast Communications in BANs

2.2 Pull-Based Multicast Communication

A *pull-based* secure multicast in which the data is stored after being generated and later is pulled by multiple authorized users may be as desirable for some cases in applications such as smart grids and body area networks. For example, multiple service providers may need to retrieve the electricity usage data of a smart meter for different purposes at different times; thus the smart meter should store its data at a data repository for future downloads. This poses significant security and privacy concerns because the access of the data in a data repository is completely out of the control of the smart meter who generated the data but it should be the smart meter's decision whether or not to disclose its electricity usage of certain smart devices to certain service providers – a service provider in California may not need the utility usage data of a microwave in a house at Washington DC. Moreover, not all service providers need the same data. Thus smart meters should have the right to decide who should have the access right to their data. Fig. 3 illustrates such a pull-based multicast scenario in smart metering. Fig. 4 demonstrates a similar example in body area networks (BANs), in which the data collected by the body sensors is stored in a data repository and later accessed by different people for different purposes: the primary doctor has the full access rights to pull the patient's medical information while a nurse is able to read only the meta data.

These applications require the data source to specify the set of users that can access the data: different users should have different access right to different data stored in the repository. Similar to the push-based multicast mentioned in Section 2.1, we resort to an access structure defined by the data source: only the user who possesses certain attributes can access the data stored in the data repository. This implies that the data source should store the access structure defining the access right in the repository as well. Note that pull-based multicast allows the destinations to actively and asynchronously pull the data from the repository while push-based multicast feeds the data to all destinations at one time.

2.3 System Model

We make the following observations from the application scenarios described in Sections 2.1 and 2.2: The multicast destinations are defined by a set of attributes forming an access structure specified by AND and OR relations. The message caring the data does not carry the identity of the destinations but carry an access structure: any user receiving the data is able to access the data only if it possesses the attributes specified in the access structure. Such multicast should provide access control, data encryption, confidentiality, and authentication to protect the data and the data source. These observations motivate us to consider a communication system depicted in Figure 5.

There are four entities in our system model: Key Generation Center (KGC), Data Source, Destinations, and Data Repository. The KGC generates and distributes keys for all entities. A data source produces the data to be broadcasted and defines the access structure of the data; it is assumed to have sufficient computational capacity to signcrypt the data. Destinations are defined by an access structure carried by the data; they are able to designcrypt a message and verify the authenticity of the source and the integrity of the data. A data repository stores signcrypted data generated by a data source.

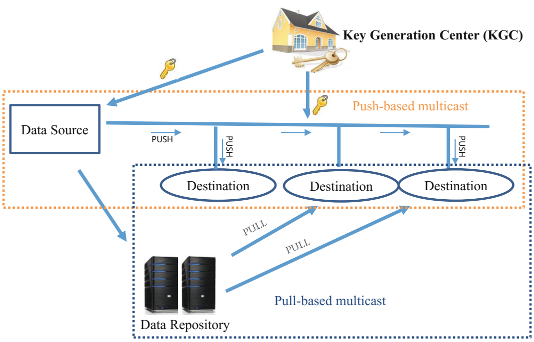

Fig. 5. A generical communication architecture.

This system model involves two types of multicasts: the multicast from a data source to all the destinations defined by an access structure (push-based multicast), and the retrieval of the data from a repository by multiple destinations (pull-based multicast).

2.4 Related Work

The most related works are IBE and ABE, which have received a significant amount of attention in recent years. There exists two different and complementary notions of ABE: Key-Policy ABE (KP_ABE) [5] and Ciphertext-Policy ABE (CP_ABE) [2]. In KP_ABE, encryption is completely determined by the full set of descriptive attributes possessed by the data source while the decryption key is computed by a Key Generation Center (KGC) from an access policy defined by the KGC. In order to decrypt a ciphertext, a user must go to KGC to get a decryption key. In CP_ABE, encryption is completely determined by an access tree defined from the set of attributes possessed by the data source, and the ciphertext carries the access policy; the decryption key is computed by KGC and is associated with a user possessing a certain set of descriptive attributes. In other words, KGC helps a user compute a deception key based on the user's attributes. A user can decrypt a ciphertext if and only if its attributes satisfy the access tree carried by the ciphertext. Therefore in CP_ABE, a data source is able to intelligently decide who should or should not have access to its data. A new construction of CP_ABE, named Constant-sized CP_ABE (denoted as CCP_ABE), was presented in [21], which reduces the ciphertext length to a constant size for an AND gate access policy with any given number of attributes at the cost of long secret keys and complicated access structures.

A scheme that employs IBE to provide a zero-configuration encryption and authentication solution for end-to-end secure communications was proposed in [19]. The concept of IBE was utilized by [11] to construct a signature and later verify the signature. KP_ABE was adopted by [3] to broadcast a single encrypted message to a specific group of users. The Lewko-Waters ABE scheme [9], was used by [17] to ensure access control. The above schemes can not ensure message integrity and confidentiality. A signcryption scheme based on KP_ABE was proposed in [4], which does not meet the requirements of many practical applications as the data source can not intelligently decide who should or should not have access to its data.

In this paper, we present a signcryption scheme termed Ciphertext-Policy Attribute-Based SignCryption (CP_ABSC) to provide the security services required by the multicast communications mentioned above. Compared to CP_ABE, CP_ABSC provides both encryption and signature without significantly increasing the computational cost (actually only the computational cost of designcryption is slightly increased compared to CP_ABE due to signature verification in CP_ABSC). CP_ABSC has strong security strength in terms of collusion resistance, message authentication, forgery prevention, and confidentiality.

3 CP_ABSC: A Ciphertext-Policy Attribute Based Signcryption Scheme

3.1 Preliminary Knowledge for CP_ABSC

Bilinear Mapping and the Bilinear Diffie-Hellman Problem. Let $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$ be three bilinear groups of prime order p, and let g_1 be a generator of $\mathbb{G}_1$ and g_2 be a generator of $\mathbb{G}_2$. Our proposed scheme makes use of a bilinear mapping: $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_3$ with the following properties:

1. *Bilinear:* A mapping $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_3$ is bilinear if and only if for $\forall P \in \mathbb{G}_1, \forall Q \in \mathbb{G}_2$, and $\forall a, b \in \mathbb{Z}_p$, $e(P^a, Q^b) = e(P, Q)^{ab}$ holds. Here $\mathbb{Z}_p = \{0, 1, \dots, p-1\}$ is a Galois field of order p.
2. *Non-degeneracy:* The generators g_1 and g_2 satisfy $e(g_1, g_2) \neq 1$.
3. *Computability:* There is an efficient algorithm to compute $e(P, Q)$ for $\forall Q \in \mathbb{G}_2$.

With a bilinear mapping, one can get the following **Bilinear Diffie-Hellman problem (BDH)**: Given three groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$ of the same prime order p. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_3$ be a bilinear mapping and g_1, g_2 be respectively the generators of $\mathbb{G}_1$ and $\mathbb{G}_2$. The objective of BDH is to compute $e(g_1, g_2)^{abc}$, where $a, b, c \in \mathbb{Z}_p$, from the given $(g_1, g_1^a, g_1^c, g_2, g_2^a, g_2^b)$.

Note that the hardness of the CBDH - i.e., the Computational Bilinear Diffie-Hellman problem (CBDH) - forms the basis for the security of our scheme.

Secret Sharing. Another important cryptographic primitive used by our CP_ABSC is secret sharing [7,18]. In the context of a *dealer* sharing a secret with n *participants* $u_1, \ldots, u_n$, a participant learns the secret if and only if it can cooperate with at least $t-1$ other participants (on sharing what they learn from the dealer), where $t \leq n$ is a pre-determined parameter. The secret to be shared by the dealer is $s \in \mathbb{Z}_p$, where $p > n$. Before secret sharing, each participant u_i holds a pairwise secret key $k_i \in \mathbb{Z}_p$, which is only known by u_i and the dealer.

The dealer follows a two-step process. First, it constructs a polynomial function $f(z)$ of degree $t-1$, i.e., $f(z) = s + \sum_{j=1}^{t-1} a_j z^j$, by randomly choosing $t-1$ i.i.d. coefficients (the a_j's) from $\mathbb{Z}_p$. Note that all (additive and multiplicative) operations used in (3.1) and throughout the rest of the paper are modular arithmetic (defined over $\mathbb{Z}_p$) as opposed to real arithmetic. Also note that s forms the constant component of $f(z)$ - i.e., $s = f(0)$. Then, in the second step, the dealer transmits to each u_i a secret share $s_i = f(k_i)$ computed from k_i, the secret key known only by u_i and the dealer.

We now show how t or more users can cooperate to recover s by sharing the secret shares received from the dealer. Without loss of generality, let $u_1, \ldots, u_t$ be the cooperating users. These t users can reconstruct the secret $s = f(0)$ from $s_1 = f(k_1), \ldots, s_t = f(k_t)$ by computing

$$s = f(0) = \sum_{j=1}^{t} \left(s_j \prod_{i \in [1,t], i \neq j} \frac{0 - k_i}{k_i - k_j} \right). \tag{1}$$

Note that the cumulative product in (1) is essentially a Lagrange coefficient. The correctness of (1) can be easily verified based on the definition of $f(z)$.

3.2 Access Control Policy – The Access Tree

Our main idea is to design an attribute-based signcryption scheme that views an identity as a set of attributes, and enforces a lower bound on the number of common attributes between a user's identity and its access rights specified by the sensitive data. We use an access tree structure proposed by [2], which is illustrated in Figure 6, to control the user's access to the encrypted data. In Figure 6, each non-leaf node x is associated with two parameters, num_x and k_x, where num_x is the number of child nodes of node x, and $k_x \in [1, num_x]$ is its threshold value indicating that node x performs the *OR* operation over all

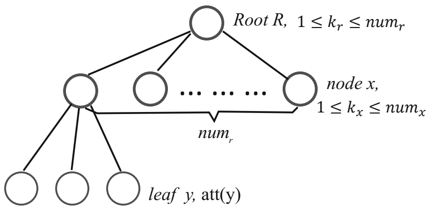

Fig. 6. An access control tree structure

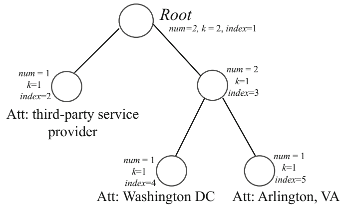

Fig. 7. An example access control structure

subsets of k_x child nodes of x, with each subset supporting an AND operation; each leaf node x is described by an attribute and a threshold value $k_x = 1$. We also associate an index with each node x in T, denoted by $index(x)$. Since a tree with $|S|$ number of attributes can have at most $2|S| - 1$ nodes, we can assign a unique number in $\{1, 2, \cdots, 2|S| - 1\}$ to each node in the tree based on pre-order tree traversal. Other tree traversal techniques such as in-order or post-order can also be applied. Let $parent(x)$ be the parent node of x in T.

Note that any attribute-based access structure can be represented by a tree T shown in Figure 6. For example, the following access structure may be specified for a data item: *Third-Party Service Provider* AND *Arlington, VA* OR *Washington, DC*, which indicates that only the third-party service providers in Arlington, VA or Washington, DC have the access to this data. Thus a user located in Washington DC with a set of attributes {*Third-Party Service Provider, Washington DC, Air-Conditioner*} has an access right to the data mentioned above. The corresponding access control tree for this example is illustrated in Figure 7. The indices of the root node and its two children are respectively 1, 2, and 3 based on pre-order tree traversal.

3.3 CP_ABSC: Ciphertext-Policy Attribute Based Signcryption

In this subsection, we propose our CP_ABSC, a Ciphertext-Policy Attribute-Based SignCryption scheme. CP_ABSC consists of four primary algorithms. Algorithm 1 is executed by KGC to provide system initialization. It generates and distributes to all the involved entities the public parameters of the system.

Algorithm 2 is also executed by KGC to generate three keys for an attribute set S: the key SK for ciphertext designcryption, the signing key K_{sign} for signing the ciphertext message, and the verification key K_{ver} for signature verification. For example, a utility company possessing the attribute set S can use its signing key K_{sign} to sign its commands or instructions sent to the smart meters, and use its designcryption key SK to designcrypt the smart meter data stored in ciphertext format (signcrypted data) at the data repositories; its verification key k_{ver} is published for others to verify the signature of its ciphertext.

Algorithm 3 details the signcryption procedure, which is the core of the proposed CP_ABSC. This algorithm is mainly performed by data sources to signcrypt its data before transmitting to the data repositories or to other receivers.

Algorithm 1 System Initialization

1: Select a prime p, the generators g_1 and g_2 for $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively, and a bilinear mapping $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_3$.
2: Choose two random exponents $\alpha, \beta \in \mathbb{Z}_p$.
3: Select a hash function $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_p$. This function H_1 is viewed as a random oracle.
4: Publish the public parameters given by

$$PK = (p, \mathbb{G}_1, \mathbb{G}_2, H_1, g_1, g_2, h = g_1^{\beta}, t = e(g_1, g_2)^{\alpha}) \tag{2}$$

5: Compute the master key $MSK = (\beta, g_2^{\alpha})$.

Algorithm 2 Key Generation (MSK, S)

Inputs: The master key MSK and a set of attributes S belonging to an entity.

1: Select random numbers $r_{en}, r_{sn} \in \mathbb{Z}_p$
2: Compute the secret key component $D_{en} = g_2^{\frac{(\alpha+r_{en})}{\beta}}$ and signing key $K_{sign} = g_2^{\frac{(\alpha+r_{sn})}{\beta}}$.
3: **for** each attribute $j \in S$ **do**
4: Select a random number $r_j \in \mathbb{Z}_p$
5: Compute the secret key components $D_j = g_2^{r_{en}} \cdot g_2^{(H_1(j) \cdot r_j)}$ and $D'_j = g_2^{r_j}$
6: **end for**
7: The secret key SK for designcryption is:

$$SK = (D_{en}, \forall j \in S : D_j, D'_j). \tag{3}$$

8: Compute the verification key: $K_{ver} = g_2^{r_{sn}}$
9: Send SK and K_{sign} to the owner of the attribute set S, and publish K_{ver} for others to verify the owner of S.

In a typical application, a data source encrypts a message/data whose access control is specified by an access tree T, and signs the message with its signing key. Note that Lines 1 to 7 is executed only once for all the data with the same access structure. Algorithm 3 is designed to provide confidentiality, access control, integrity, authentication, and non-repudiation to ensure the security and privacy of the data sources. Note that encryption is completely determined by the access policy of the data itself.

Algorithm 4 implements verification and decryption. The ciphertext receivers execute it to decrypt the ciphertext according to their attributes. Note that Algorithm 4 calls a function *DecryptNode* described in Algorithm 5, which was originally proposed by [2]. Here we include *DecryptNode* for completeness and to help the readers without the knowledge of CP_ABE to understand CP_ABSC.

Algorithm 3 SignCryption*(M, T, K_{sign})*

Inputs: The public parameter PK; plaintext message M; the tree T rooted at node R specifying the access control policy of message M; and the signing key K_{sign}.

1: Choose a polynomial q_x and sets its degree $d_x = k_x - 1$ for each node x in the tree T.
2: Choose a random number $s \in \mathbb{Z}_p$ and sets $q_R(0) = s$;
3: Choose d_R random numbers from $\mathbb{Z}_p$ to completely define the polynomial q_R.
4: **for** any other node x in T **do**
5: Set $q_x(0) = q_{parent(x)}(index(x))$.
6: Select d_x random numbers from $\mathbb{Z}_p$ to completely define q_x.
7: **end for**
8: Let Y be the set of leaf nodes in T. The ciphertext CT is constructed based on the access tree T as follows:

$$CT = (T, \tilde{C} = M \oplus t^s, C = h^s, \forall y \in Y : C_y = g_1^{q_y(0)}, C'_y = g_1^{(H_1(att(y)) \cdot q_y(0))}) \quad (4)$$

9: Choose a random $\zeta \in \mathbb{Z}_p$; compute $\delta = e(C, g_2)^\zeta$, $\pi = H_1(\delta|M)$, and $\psi = g_2^\zeta \cdot (K_{sign})^\pi$.
10: Output the message:

$$CT_{sign} = (T, \tilde{C}, C, \forall y \in Y : C_y, C'_y; W = g_1^s, \pi, \psi)$$

Algorithm 4 DeSignCryption (CT_{sign}, SK, S)

Inputs: The $CT_{sign} = (CT, W, \pi, \psi)$; the private key SK for designcryption; and the set of possessed attributes S.

1: $A = DecryptNode(CT, SK, R)$
2: **if** $A \neq \perp$ **then**
3: $\tilde{A} = e(C, D_{en})/A$
4: **end if**
5: Compute

$$\delta' = \frac{e(C, \psi)}{(e(W, K_{ver}) \cdot \tilde{A})^\pi} \quad (5)$$

6: **if** $H_1(\delta'|M') = \pi$ **then**
7: return $M = M'$
8: **end if**
9: Return $\perp$

3.4 CP_ABSC v.s. CP_ABE

In this section, we compare CP_ABSC and CP_ABE[2] to illustrate their differences. The characteristics of CP_ABSC and CP_ABE are summarized in Table 1.

Algorithm 5 Function *DecryptNode* (CT, SK, x)

Inputs: A ciphertext $CT = (T, \tilde{C}, C, \forall y \in Y : C_y, C'_y)$; the secret key SK, which is associated with a set S of attributes, the node x from T.

if x is a leaf node of T **then**
  Let $i = att(x)$
  if $i \in S$ **then**

$$\text{Return } F_x = \frac{e(C_i, D_i)}{e(C'_i, D'_i)} = e(g_1, g_2)^{r_{en} q_x(0)} \tag{6}$$

  else Return $\perp$
  end if
else
  for Each child node z of x **do**
    $F_z = DecryptNode(CT, SK, z)$
  end for
end if
Let S_x be an arbitrary k_x-sized set of child nodes of x such that $F_z \neq \perp$ for $\forall z \in S_x$.
if S_x exists **then**
  for Each node $z \in S_x$ **do**
    $i_z = index(z)$
    $S'_z = \{index(z) \mid\mid z \in S_x\}$
    $\triangle_{i_z, S'_z}(y) = \prod_{j \in S'_z, j \neq i_z} \frac{y-j}{i_z-j}$
  end for
  Return

$$F_x = \prod_{z \in S_x} F_z^{\triangle_{i_z, S'_z}(0)} = \prod_{z \in S_x} \left(e(g_1, g_2)^{r_{en} \cdot q_z(0)}\right)^{\triangle_{i_z, S'_z}(0)}$$
$$= \prod_{z \in S_x} e(g_1, g_2)^{r_{en} \cdot q_x(i_z) \cdot \triangle_{i_z, S'_z}(0)} = e(g_1, g_2)^{r_{en} \cdot q_x(0)}$$

else
  Return $F_x = \perp$
end if

System Initialization. This procedure creates the groups, the group generators, and the bilinear mapping. The difference between CP_ABSC and CP_ABE is that the former uses asymmetric groups while the latter uses symmetric groups.

Key Generation. The Key Generation algorithm in our scheme CP_ABSC is different from the key generation in CP_ABE [2] in two aspects: i) since we are designing a signcryption scheme, we need to compute a signing key (which will be sent to the signcryptor) and a verification key (which will be public) while CP_ABE only needs one key for decryption; and ii) due to the fact that CP_ABSC utilizes asymmetric groups, its key generation is more computationally efficient than the one proposed in [2] according to our comparison study in Section 4.3.

Table 1. Comparison between CP_ABE and CP_ABSC

The scheme	System Initialization	Key Generation	Encryption	Decryption
CP_ABE [2]	symmetric groups	private(encrypt) key	encryption	decryption
CP_ABSC	asymmetric groups	private(encrypt+sign) key	signcryption	decryption& verification

Encryption (SignCryption). The SignCryption in CP_ABSC combines signature and encryption, while the one in [2] performs only encryption. The computational cost of our SignCryption algorithm is less than the sum of the two computations (encryption and signature), and is also less than that of the encryption algorithm in [2], according to our analysis in Section 4.3, which is attributed to the adopted asymmetric groups.

Decryption (DeSignCryption). The DeSignCryption in CP_ABSC includes decryption and verification, while the decrypt algorithm in [2] performs only decryption. The computational cost of DeSignCryption is only slightly higher than that of the decyption algorithm in [2], according to our analysis in Section 4.3.

3.5 Application of CP_ABSC in Smart Grids

In this section, we illustrate how to use CP_ABSC to secure the two typical multicast communications in a smart grid. Initially, KGC computes the public parameters PK according to Algorithm 1, and posts PK to all active entities (smart meters and service providers) in the system. Each entity also needs to register with KGC to get the corresponding keys computed from Algorithm 2. For example, a utility company needs a private key SK for designcryption based on its access attributes, a signing key K_{sign} to sign its commands, and a verification key K_{ver} for others to verify its signature.

Push-Based Multicast Communication in Smart Grid. When a service provider wants to send instructions or commands to one or more smart meters, the service provider constructs an access structure T that describes the set of smart meters satisfying the access policy. It then signcrypts an instruction I with a timestamp ts. The timestamp can be the current time or the current time with an expiration time. Generally speaking, the timestamp can help the receivers decide whether or not instruction I is valid and resist replay attacks. The following procedure implements a push-based multicast for a service provider to broadcast I to certain smart meters.

1. The service provider broadcasts the following signcrypted instruction to the smart meters according to Algorithm 3:

$$Service\ provider \rightarrow Smart\ meters : SignCryption(I||ts, T, K_{sign}).$$

2. When a smart meter receives the signcrypted instruction, it designcrypts and verifies the message according to Algorithm 4. If the verification is passed,

the smart meter executes the instruction and sends a response to the service provider to notify that it has received the instruction (proving that it has the required privilege).

3. When the service provider receives the feedback response, the communication is completed; otherwise, the service provider sends the instruction again.

Pull-Based Multicast Communication in Smart Grid. In order to protect the power usage data, a smart meter signcrypts the data of its household devices using Algorithm 3 based on the access policy specified by the data, and then sends the signcrypted data CT_{sign} to a data repository. When a service provider possessing an attribute set S wants to get the data for a particular household device, it contacts the data repository and gets the signcrypted data CT_{sign}. The following procedure details the process implementing a pull-based multicast.

1. A smart meter signcrypts its reading M with a timestamp ts, $M||ts$, based on Algorithm 3 and then sends CT_{sign} to the data repository. This step can be performed whenever a new data item is generated.

$$Smart\ meter \rightarrow Data\ repository : CT_{sign}.$$

2. When a service provider holding an attribute set S needs to access the smart meter data, it contacts the data repository to obtain the signcrypted data CT_{sign}:

$$Data\ repository \rightarrow Service\ provider : CT_{sign}.$$

3. Upon receiving the signcrypted data CT_{sign}, the service provider designcrypts CT_{sign} and verifies the message according to Algorithm 4: it first recovers the plaintext M' based on its private key SK and then computes δ'; if $H_1(\delta'|M') = \pi$, which demonstrates the successful designcryption of the data, the service provider accepts M'; otherwise, the message is dropped.

4 Correctness and Performance Analysis

In this section, we prove the correctness of CP_ABSC and analyze its security strength. We also carry out a simulation based performance analysis to quantitatively study the efficiency and computational cost of CP_ABSC.

4.1 The Correctness of CP_ABSC

In this subsection, we show that CP_ABSC is indeed feasible and correct. First, from the decryption procedure we have

$$\begin{aligned} M' &= \tilde{C} \oplus \tilde{A} = \tilde{C} \oplus (\frac{e(C,D)}{A}) = \tilde{C} \oplus (\frac{e(C,D)}{A}) \\ &= \tilde{C} \oplus (\frac{e(h^s, g_2^{(\alpha+r_{en})/\beta})}{e(g_1,g_2)^{r_{en}s}}) = M \oplus e(g_1,g_2)^{\alpha s} \oplus (\frac{e(g_1^{\beta s}, g_2^{\alpha+r_{en}/\beta})}{e(g_1,g_2)^{r_{en}s}}) \end{aligned}$$

$$= M \oplus e(g_1, g_2)^{\alpha s} \oplus (\frac{e(g_1, g_2)^{\beta s \cdot (\alpha + r_{en})/\beta}}{e(g_1, g_2)^{r_{en} s}})$$
$$= M \oplus e(g_1, g_2)^{\alpha s} \oplus (\frac{e(g_1, g_2)^{(\alpha s + r_{en} s)}}{e(g_1, g_2)^{r_{en} s}})$$
$$= M \oplus e(g_1, g_2)^{\alpha s} \oplus e(g_1, g_2)^{\alpha s} = M.$$

which indicates that Algorithm 4 can correctly decrypt the ciphertext if the designcryptor satisfies the access policy (posessing the designcryption key SK).

Second, the receiver verifies whether the message M' has been forged or falsified, and whether the received message is indeed sent by the generator of the message. The designcryptor (the receiver) computes δ' by:

$$\delta' = \frac{e(C, \psi)}{(e(W, K_{ver}) \cdot \tilde{A})^{\pi}} = \frac{e(g_1^{\beta s}, g_2^{\zeta} \times g_2^{\frac{(\alpha + r_{sn})}{\beta}\pi})}{(e(g_1^{s}, g_2^{r_{sn}}) \cdot e(g_1, g_2)^{\alpha s})^{\pi}}$$
$$= e(g_1, g_2)^{\beta s(\zeta + \frac{(\alpha + r_{sn})}{\beta}\pi) - s r_{sn}\pi - \alpha s \pi} = e(g_1, g_2)^{\beta s \zeta + s(\alpha + r_{sn})\pi - s r_{sn}\pi - \alpha s \pi}$$
$$= e(g_1, g_2)^{\beta s \zeta} = e(C, g_2)^{\zeta} = \delta.$$

If $H_1(\delta'|M') = \pi$, M' is valid, i.e., $M = M'$, and the message is not modified and is indeed sent by the generator; otherwise, M' is invalid.

4.2 Security Strength

In this subsection, we analyze the security strength of the proposed scheme CP_ABSC by examining how it can counter four major attacks.

Collusion. In CP_ABSC, the set of attributes composes of the user's identity. In order to provide different types of users with different access rights, the scheme provides an access tree structure for each signcrypted data item, and requires only a subset of the attributes for designcryption. Since the secret key computation involves a unique random number for each attribute in the access policy, our scheme can defend against collusion attacks. For example, assume that neither user U_1 nor user U_2 possesses a sufficient number of attributes to successfully designcrypt the ciphertext CT_{sign} alone but the combined attribute set has sufficient number of attributes for the designcryption. Then U_1 and U_2 may collude by combining their attributes. However, they are not able to combine their secret keys (the SKs) to get a secret key for the combined set of attributes according to Algorithm 2 because the KGC generates different random numbers r_{en} for U_1 and U_2. Thus they could not designcrypt the message, and the proposed scheme is secure against collusion attacks.

Message Authentication. Assume that a user U wants to get a message M from the data repository. Before the data is stored in the data repository, the data generator has signcrypted it with Algorithm 3. When U plans to obtain the

Table 2. The details of Functions and Operations between CP_ABE and our scheme

	CP_ABE [2]	CP_ABSC
Key Generation	$n\mathbb{G}_1 + (n+2)\mathbb{G}_2 + nH_{\mathbb{G}_2}$	$(2n+5)\mathbb{G}_2$
Encryption	$(k+1)\mathbb{G}_1 + k\mathbb{G}_2 + 1\mathbb{G}_3 + kH_{\mathbb{G}_2}$	$(2k+2)\mathbb{G}_1 + 2\mathbb{G}_2 + 2\mathbb{G}_3 + 2$ (pairings)
Decryption	$(2k'+1)$ (pairings)	$(2k'+3)$ (pairings)

Notes: $\mathbb{G}_1$ in the table means an exponentiation operation in $\mathbb{G}_1$ group; $\mathbb{G}_2$ and $\mathbb{G}_3$ are defined similarly. $H_{\mathbb{G}_1}$ means hashing an attribute string or a message into an element in $\mathbb{G}_1$; $H_{\mathbb{G}_2}$ is defined similarly.

Table 3. The Computational Cost (Run Time) of Different Operations in Charm Library

Group	$\mathbb{G}_1$	$\mathbb{G}_2$	$\mathbb{G}_3$	(pairings)	$H_{\mathbb{G}_1}$	$H_{\mathbb{G}_2}$
SS512	3.73	3.70	0.48	3.92	8.34	8.39
MNT159	1.12	9.84	2.62	8.42	0.10	34.82

Notes: Time is in ms. The result in this table is the average of 1000 runs.

data from the data repository, it needs its private key $SK = (D = g_2^{\frac{(\alpha+r_{en})}{\beta}}, \forall j \in S : D_j = g_2^{r_{en}} \cdot g_2^{(H_1(j)\cdot r_j)}, D'_j = g_2^{r_j})$, which is computed by Algorithm 2. Meanwhile, U obtains the data source's verification key from KGC. It designcrypts the ciphertext to get the message M' by Algorithm 4: if $H_1(\delta'|M') = \pi$, the decrypted message M is valid; otherwise, it is discarded.

Forgery. An adversary who wishes to forge the signcryption of a legal user must possess the user's signing key. An adversary cannot infer the signing key K_{sign} or the root node of the access tree T because the random number r for each attribute in S (In Algorithm 2) and the s for the root of T (in Algorithm 3) are chosen randomly and secretly. An adversary cannot create a new, valid ciphertext from other user's ciphertexts. If the adversary changes the ciphertext of a message, the receiver can verify that the ciphertext is illegal by Algorithm 4. Moreover, colluding users can not forge a ciphertext, as analyzed before. Thus we claim that our proposed scheme is unforgeable.

Confidentiality. Decryption requires the knowledge of $e(g_1, g_2)^{\alpha s}$. The decryption procedure takes the same idea as that of CP_ABE [2], and thus CP_ABSC has the same security strength as that of the CP_ABE. The designcryption requires the knowledge of $\delta = e(C, g_2)^{\zeta}$. For a passive adversary, the available information is CT_{sign}. It is difficult to get s from the W in CT_{sign} since it is difficult to compute the discrete logarithm problem. Even if the adversary constructs the bilinear mapping e via C and the public parameter g_2 to obtain $e(C, g_2)$, it can not get ζ, which is randomly chosen by the signcryptor. The adversary may try to get ζ from ψ, but it has to get the K_{sign} first. Even if the K_{sign} is compromised, the adversary still can't get ζ from ψ due to the difficulty of computing the discrete logarithm problem. Given the discussion above and the

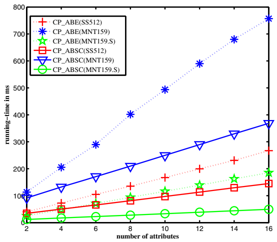

Fig. 8. Key generation time

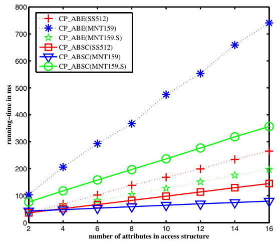

Fig. 9. Encryption time

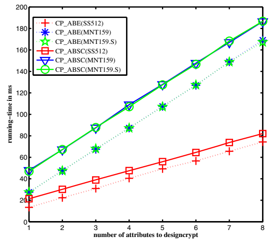

Fig. 10. Decryption time

fact that CP_ABE is proven secure under chosen-ciphertext attacks, our scheme is secure under chosen-ciphertext attacks too.

4.3 Efficiency and Cost Analysis

In this subsection, we present a quantitative performance study on CP_ABSC.

Our scheme CP_ABSC does not incur a high computational cost in Key Generation, SignCryption, and DeSignCryption compared to CP_ABE. Table 2 reports the amount of operations performed by CP_ABE and CP_ABSC. The notations are explained as follows: n is the number of attributes a user holds, k is the number of leaf nodes in the access tree T, and k' is the number of attributes a user possesses. $\mathbb{G}_1$ denotes an exponent operation in $\mathbb{G}_1$ group, and the same definitions hold for $\mathbb{G}_2$ and $\mathbb{G}_3$. $H_{\mathbb{G}_1}$ means hashing an attribute or message into an element in $\mathbb{G}_1$, and $H_{\mathbb{G}_2}$ is defined similarly.

Starting with Key Generation, as described in Algorithm 2, there is $2n + 5$ exponent operations in $\mathbb{G}_2$, which includes 5 exponent operations $\{g_2^{r_{en}}, g_2^{\beta}, g_2^{r_{sn}}, D_{en}, K_{sign}\}$, and $2n$ exponent operations $\{D_j, D'_j\}$. In CP_ABE[2], the total operations is $n\mathbb{G}_1 + (n+2)\mathbb{G}_2 + nH_{\mathbb{G}_2}$.

Moving next to the Signcryption in Algorithm 3, there are $2k + 2$ exponent operations in group $\mathbb{G}_1$ and 2 exponent operations in group $\mathbb{G}_2$. Additionally, there are 2 map operations and 2 pairing. The combined overhead is thus $(2k + 2)\mathbb{G}_1 + 2\mathbb{G}_2 + 2\mathbb{G}_3 + 2$ (pairings). Similarly, in CP_ABE, the total operation is $(k+1)\mathbb{G}_1 + k\mathbb{G}_2 + 1\mathbb{G}_3 + kH_{\mathbb{G}_2}$.

For Designcryption (in Algorithm 4), there are $(2k'+3)$ (pairings) operations. In CP_ABE, there are $(2k' + 1)$ (pairings) operations.

We run the experiment with Ubuntu 12.04 running as a VM on a MAC-Book Air with one 1.8GHz core and 1GB memory. The implementation uses a Python library called Charm-crypto [1], which is a framework used to prototype advanced cryptosystems such as IBE and IBS (Identity-Based Signature). The core mathematical functions behind Charm are from the Stanford Pairing-Based Cryptography (PBC) library [12], which is an open source C library that performs mathematical operations underlying pairing-based cryptosystems.

We execute the implementation under both symmetric (SS512) and asymmetric groups (MNT159 and MNT159.S), both with 80 bits of security, to compare CP_ABE and CP_ABSC. In SS512, the map is $\mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_3$, where $\mathbb{G}_1$ and $\mathbb{G}_2$ are the same group. In MNT159, the map is $\mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_3$, where $\mathbb{G}_1$ and $\mathbb{G}_2$ are different groups, and $\mathbb{G}_2$ and $\mathbb{G}_3$ are extension groups of $\mathbb{G}_1$. The elements in $\mathbb{G}_2$ and $\mathbb{G}_3$ are longer than those in $\mathbb{G}_1$. The longer the element, the larger the computational cost in exponential operations. In MNT159.S, we swapped the $\mathbb{G}_1$ and $\mathbb{G}_2$ group so that most of the key generation operations are in $\mathbb{G}_1$ instead of $\mathbb{G}_2$.

Table 3 lists the run time of each operation and function in SS512 and MNT159. One can see that some operations are more efficient in SS512 than in MNT159 while others are the opposite. For example, the operations $H_{\mathbb{G}_1}$ and $\mathbb{G}_1$ have less run time in MNT159 than in SS512 but the operations of $\mathbb{G}_2$ and $H_{\mathbb{G}_2}$ have less runtime in SS512 than in MNT159.

The performance analysis compares the efficiency and computational cost between CP_ABSC and CP_ABE for Key Generation, Signcryption/Encryption, and Designcryption/Decryption. The results are reported in Figures 8-10. Figure 8 shows the run times of Key Generation. MNT159.S has the best performance since we swapped $\mathbb{G}_1$ and $\mathbb{G}_2$ and most of the operations are in $\mathbb{G}_1$ after the swap. Figure 9 reports the encryption run times. The run time in CP_ABE and that in our scheme CP_ABSC is almost linear with respect to the number of leaf nodes in the access policy. The polynomial operation at leaf nodes does not significantly contribute to the run time. Comparing the run time between CP_ABE encryption and CP_ABSC signcryption, one can see that our scheme costs less time than CP_ABE because we don't need to compute $H_{\mathbb{G}_2}$. Figure 10 illustrates the run times of decryption. Our scheme is slightly higher than that of CP_ABE due to the fact that we add the signature verification process. However, because the computational cost of ABE is more expensive as the number of attributes increases, the cost of signature verification is relatively trivial in practice.

Considering all three processes of KeyGeneration, SignCryption, and DeSignCryption, MNT159.S has considerably better performance than MNT159. We recommend executing the schemes in asymmetric groups and swapping $\mathbb{G}_1$ and $\mathbb{G}_2$ to gain a better performance.

Due to space limitation, we omit the part of comparison between the proposed scheme and Attribute based signature, which will be included in the extended version.

In summary, the run time is predictable for key generation and encryption in our scheme and is correlated with the number of attributes. Comparing the run times of key generation, encryption, and decryption between CP_ABE and our scheme CP_ABSC, the run times of our scheme is a little higher than CP_ABE for some cases. However, considering that our scheme combines encryption and signature, CP_ABSC is feasible and more desirable than the encryption-only CP_ABE.

5 Conclusion and Future Work

In this paper, we present a signcryption scheme called CP_ABSC that can provide access control, data confidentiality, and authentication based on an access structure specified by the to-be-protected data itself. We analyze the computational cost and security strength of CP_ABSC, and illustrate how to apply CP_ABSC to protect the multicast communications in smart grids. Particularly, we employ CP_ABSC to secure two types of multicasts: the push-based multicast of instructions/commands from service providers to smart meters and the pull-based data retrieval from data repositories to service providers.

Our future research lies in the following directions: design more efficient signcryption approaches with less computational and storage requirements; and develop a dynamic scheme that could dynamically add attributes to adapt to the changing requirements of applications.

Acknowledgement. This research is supported by National Natural Science Foundation of China under grant 61373027, and the US National Science Foundation under grants CCF-1442642, IIS-1343976, CNS-1318872, and CNS-1550313.

References

1. Akinyele, J.A., Garman, C., Miers, I., Pagano, M.W., Rushanan, M., Green, M., Rubin, A.D.: Charm: A framework for rapidly prototyping cryptosystems. Journal of Cryptographic Engineering **3**(2), 111–128 (2013)
2. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE Symposium on Security and Privacy, SP 2007, pp. 321–334. IEEE (2007)
3. Fadlullah, Z.M., Kato, N., Lu, R., Shen, X., Nozaki, Y.: Toward secure targeted broadcast in smart grid. IEEE Communications Magazine **50**(5), 150–156 (2012)
4. Gagné, M., Narayan, S., Safavi-Naini, R.: Threshold attribute-based signcryption. In: Garay, J.A., De Prisco, R. (eds.) SCN 2010. LNCS, vol. 6280, pp. 154–171. Springer, Heidelberg (2010)
5. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Proceedings of the 13th ACM Conference on Computer and Communications Security, pp. 89–98. ACM (2006)
6. Hart, G.: Nonintrusive appliance load monitoring. Proc. IEEE **80**(12), 1870–1891 (1992)
7. Chunqiang, H., Liao, X., Cheng, X.: Verifiable multi-secret sharing based on lrsr sequences. Theoret. Comput. Sci. **445**, 52–62 (2012)
8. Kgwadi, M., Kunz, T.: Securing rds broadcast messages for smart grid applications. International Journal of Autonomous and Adaptive Communications Systems **4**(4), 412–426 (2011)
9. Lewko, A., Waters, B.: Decentralizing attribute-based encryption. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 568–588. Springer, Heidelberg (2011)
10. Li, Q., Cao, G.: Multicast authentication in the smart grid with one-time signature. IEEE Transactions on Smart Grid **2**(4), 686–696 (2011)

11. Lu, R., Liang, X., Li, X., Lin, X., Shen, X., et al.: Eppa: An efficient and privacy-preserving aggregation scheme for secure smart grid communications. IEEE Trans. on Parallel and Distributed Systems (2012)
12. Lynn, B.: On the implementation of pairing-based cryptosystems. PhD thesis, Stanford University (2007)
13. Neumann, W.D.: Horse: an extension of an r-time signature scheme with fast signing and verification. In: International Conference on Information Technology: Coding and Computing, Proceedings. ITCC 2004, vol. 1, pp. 129–134. IEEE (2004)
14. Perrig, A.: The biba one-time signature and broadcast authentication protocol. In: Proceedings of the 8th ACM conference on Computer and Communications Security, pp. 28–37. ACM (2001)
15. Perrig, A., Canetti, R., Tygar, J.D., Song, D.: The tesla broadcast authentication protocol. CryptoBytes **5**(2), 2–13 (2002)
16. Reyzin, L., Reyzin, N.: Better than BiBa: short one-time signatures with fast signing and verifying. In: Batten, L.M., Seberry, J. (eds.) ACISP 2002. LNCS, vol. 2384, pp. 144–153. Springer, Heidelberg (2002)
17. Ruj, S., Nayak, A., Stojmenovic, I.: A security architecture for data aggregation, access control in smart grids. Arxiv preprint arXiv: 1111.2619 (2011)
18. Shamir, A.: How to share a secret. Commun. ACM **22**(11), 612–613 (1979)
19. So, H.K.H., Kwok, S.H.M., Lam, E.Y., Lui, K.S.: Zero-configuration identity-based signcryption scheme for smart grid. In: IEEE International Conference on Smart Grid Communications, pp. 321–326. IEEE (2010)
20. Wang, Q., Khurana, H., Huang, Y., Nahrstedt, K.: Time valid one-time signature for time-critical multicast data authentication. In: IEEE INFOCOM 2009, pp. 1233–1241. IEEE (2009)
21. Zhou, Z., Huang, D.: On efficient ciphertext-policy attribute based encryption and broadcast encryption. In: Proceedings of the 17th ACM Conference on Computer and Communications Security, pp. 753–755. ACM (2010)