8-2018

Server-aided attribute-based signature with revocation for resource-constrained Industrial-Internet-of-Things devices

Hui CUI
Royal Melbourne Institute of Technology

Robert H. DENG
Singapore Management University, robertdeng@smu.edu.sg

Joseph K. LIU
Monash University

Xun YI
Royal Melbourne Institute of Technology

Yingjiu LI
Singapore Management University, yjli@smu.edu.sg

DOI: https://doi.org/10.1109/TII.2018.2813304

Published in IEEE Transactions on Industrial Informatics, Aug. 2018, Volume: 14 , Issue: 8 , pp. 3724-3732.
https://doi-org.libproxy.smu.edu.sg/10.1109/TII.2018.2813304

Server-Aided Attribute-Based Signature With Revocation for Resource-Constrained Industrial-Internet-of-Things Devices

Hui Cui, Robert H. Deng, *Fellow, IEEE*, Joseph K. Liu, Xun Yi, and Yingjiu Li, *Member, IEEE*

***Abstract*—The industrial Internet-of-things (IIoT) can be seen as the usage of Internet-of-things technologies in industries, which provides a way to improve the operational efficiency. An attribute-based signature (ABS) has been a very useful technique for services requiring anonymous authentication in practice, where a signer can sign a message over a set of attributes without disclosing any information about his/her identity, and a signature only attests to the fact that it is created by a signer with several attributes satisfying some claim predicate. However, an ABS scheme requires exponentiation and/or pairing operations in the signature generation and verification algorithms, and hence, it is quite expensive for resource-constrained devices like a sensor in the IIoT network to run an ABS scheme. To reduce the computational overheads for both signers and verifiers, it has been suggested to introduce a server to help with signature generation and verification, but existing results on the ABS with "server-aided computation" either suffer from the security issues or are not sufficiently efficient. In this paper, we consider server-aided ABS one step further, and propose a notion called server-aided ABS with revocation (SA-ABSR), which not only securely mitigates the workloads of users in generating and verifying signatures, but also enables user revocation by having the server immediately stop signature generations for revoked signers. We formally define the security model for SA-ABSR, present a concrete construction of SA-ABSR based on a standard ABS scheme, and prove its security under the defined security model. Also, we implement the proposed SA-ABSR scheme and the underlying standard ABS scheme to evaluate the performance, from which it is easy to see that the proposed SA-ABSR scheme is more efficient than its underlying ABS scheme.**

Manuscript received October 26, 2017; revised January 2, 2018 and February 21, 2018; accepted March 2, 2018. Date of publication March 8, 2018; date of current version August 1, 2018. This work was supported in part by the Singapore National Research Foundation under the NCR Award NRF2014NCR-NCR001-012 and in part by the AXA Research Fund. Paper no. TII-17-2476. (*Corresponding author: Hui Cui.*)

H. Cui and X. Yi are with the School of Science, Royal Melbourne Institute of Technology (RMIT) University, Melbourne, VIC 3000, Australia (e-mail: hui.cui@rmit.edu.au; xun.yi@rmit.edu.au).

R. H. Deng and Y. Li are with the School of Information Systems, Singapore Management University, Singapore 188065 (e-mail: robertdeng@smu.edu.sg; yli@smu.edu.sg).

J. K. Liu is with the Faculty of Information Technology, Monash University, Melbourne, VIC 3800, Australia (e-mail: joseph.liu@monash.edu).

Digital Object Identifier 10.1109/TII.2018.2813304

***Index Terms*—Attribute-based signature (ABS), server-aided signing, server-aided verification, user revocation.**

I. Introduction

THE concept of industrial Internet-of-things (IIoT) [1] arms the internet with a capability to directly control the physical world including devices, factories, and infrastructures, but it also raises challenges in the security and privacy protection in the IIoT network.

The attribute-based signature (ABS) [2], [3] is a cryptographic primitive that provides not only unforgeability as that in the standard digital signature but also signer anonymity [4]. Therefore, the ABS has a wide range of applications such as privacy-preserving access control, anonymous credentials, and so on [4]. Unfortunately, due to the involvement of many exponentiation and/or pairing calculations, ABS schemes incur heavy computations in the signature generation and verification stages, which grow linearly to the complexity of the specified claim predicate. This is challenging for resource-constrained devices (with limited computational capabilities) such as tablets in the IIoT scenarios. To address this issue, one approach is to use the offline/online cryptographic approach (e.g., [5] and [6]), which allows a signer to perform heavy calculations (e.g., exponentiation and pairing) during the offline phase and generates a signature by performing light-weight computations in the online phase. But this approach does not reduce the computational overhead in the verification stage.

A promising solution to this problem is to delegate a third party (i.e., a server) to perform heavy calculations and leave the signer/verifier with a small amount of computations. This is known as "server-aided computation" [7], which has been a very popular method in ameliorating computational overheads for cryptographic primitives including attribute-based encryption [8], ABS [9], [10], identity-based encryption [11], [12], digital signature [13], public-key encryption [14], and authentication [15]. In terms of an ABS, several server-aided ABS schemes have been proposed, where a server is assigned to assist with either signature generation or signature verification or both. However, the server-aided ABS scheme (e.g., [10]), which applies a server to help with the workloads in both signing and verifying algorithms, is not secure [9], because the server and the signer can collude together to make the verifier accept an invalid signature. Though this security flaw has been fixed in

[9], the resulting ABS scheme with server-aided verification is not satisfactory in performance, which still requires the verifier to conduct heavy calculations. Due to these observations, we think about designing a secure server-aided ABS scheme with improved efficiency, in which the server helps to significantly reduce workloads in both signature generation and verification phases. On the other hand, as in an attribute-based cryptographic system, the status of users may change (e.g., users may leave the system) or their signing keys may be leaked or compromised, it is necessary to equip an ABS scheme with an efficient user revocation mechanism [8]. Interestingly, we find that we can subtly make use of the server in a server-aided ABS scheme to achieve efficient user revocation.

In this paper, we propose a notion of a server-aided ABS with revocation (SA-ABSR), which can be applied in scenarios such as an IIoT network to achieve anonymous authentication among mobile devices with limited computation power. In an SA-ABSR scheme, most of the computational overheads in the signature generation and verification algorithms are delegated to a server to alleviate the workloads of signers/verifiers, and user revocation can be easily accomplished by requiring the server to directly stop the partial signature generation operation for any revoked signer.

A. Our Contributions

Compared to a standard ABS scheme, an SA-ABSR scheme introduces an additional server with three main functions. First, the server helps a signer in the signature generation process such that the signer's workload in generating the signature is greatly reduced. Second, it assists a verifier to check the validity of a signature such that the verifier does not need to perform a large amount of computations. Third, it facilitates user revocation by terminating the signature generation operation for any revoked signer.

The key challenges in building an SA-ABSR scheme are twofold. On the one hand, regarding the server-aided signature generation, since the server is not fully trusted and the revocation function is achieved by the server who will immediately stop generating the partial signature for any revoked signer, it is essential to guarantee that neither the server nor the signer can independently create a valid signature. Our solution to address this challenge is motivated by the key splitting technique in [10]. The attribute authority (AA) splits the master private key into two parts, it embeds one part in the partial signing key, which is given to the server and the other part in the signing key, which is given to a signer. Thus, neither the server nor the signer can create a signature by itself, and they must collaborate to output a valid signature. On the other hand, concerning the server-aided signature verification, it is crucial to confirm the validity of the signature such that the verifier will not accept an invalid signature even if the server colludes with the signer. Though the server will not help a revoked signer generate a signature, it is possible that the server colludes with the signer such that a verifier might accept an invalid signature. To prevent such collusion attacks between the server and the signer, we improve the transformation technique introduced in [7] to accomplish secure and efficient server-aided verification. Note that it has been claimed in [9] that the transformation technique in [7] does not work for an ABS scheme and they introduce an ABS scheme following a different server-aided approach. In this paper, we show that it is possible to utilize the transformation technique in [7] to achieve secure server-aided verification via some subtle modifications and the resulting scheme can have better efficiency than the one in [7].

We summarize the contributions in this paper as follows.

1) We introduce a notion of SA-ABSR, which delegates most of the computational operations in the signature generation and verification algorithms to a server, and accomplishes user revocation by making the server stop the partial signature generation for any revoked signer.
2) We present a concrete SA-ABSR scheme, formally prove its security and assess its performance theoretically and experimentally. Compared to the existing results on server-aided ABS schemes, the proposed SA-ABSR scheme overcomes their security vulnerability and attains better efficiency.

B. Related Works

1) ABS: Li *et al.* [2] gave an ABS scheme that is secure in the standard model, but the claim predicates can only be the threshold ones. Maji *et al.* [3] first defined the formal definition of the ABS, and presented an efficient construction on the ABS supporting any claim-predicate expressed in any monotone span programme, but the security of the scheme can only be proved in the generic group model. Herranz *et al.* [16] built two ABS schemes with signatures of constant sizes, but their schemes are less efficient than the scheme in [2] and [3]. Okamoto and Takashima [17] put forward the first fully secure ABS scheme enabling nonmonotone claim predicates in the standard model under the decentralized setting, but it also suffers the efficiency issue.

There are ABS schemes presented to solve different issues raised in ABS including revocation (e.g., [18]), accountability (e.g., [19]), and key escrow (e.g., [4]), and ABS schemes proposed to meet the requirements in different scenarios such as an attribute-based group signature [20] and the forward secure ABS [21].

Motivated by that most of the existing ABS schemes require many exponentiation calculations in the signature generation phase, Chen *et al.* [10] introduced the notion of outsourced ABS (OABS) where the majority computational overhead from the signature generation algorithm is delegated to a server. Chen *et al.* [10] also extended their OABS scheme to accomplish server-aided verification, but the resulting scheme was found insecure [9], where the signer and the server may collude together such that an invalid signature will be accepted by the verifier. To address this security flaw, Wang *et al.* [9] proposed a notion of attribute-based server-aided verification (ABSAV), which enables a server to securely assist the verifier with signature verification, but the scheme could not considerably mitigate the workload of the verifier.

2) Server-Aided Technology: "Server-aided computation" was first introduced by Matsumoto *et al.* [22] to speed up the computation, where a powerful server is applied to help the

low-power devices execute heavy cryptographic operations. As a desirable solution to reduce computational overheads, server-aided computation has been widely used in various schemes to help with heavy calculations in the algorithms including key computation, signature generation, signature verification, encryption, decryption, and so on [11]. For example, server-aided signature [7], [23] was proposed to reduce the exponentiation and pairing calculations.

In addition to ameliorate the computational overheads, the server-aided technique has also been utilized for efficient user revocation (e.g., [24]), where a semitrusted server immediately terminates partial decryption operations for revoked users. Compared to the traditional revocation methodology, such an approach does not require the private key of users to be updated regularly, and greatly simplifies the revocation operation, but it does not allow the server to collude with the users.

C. Organization

The remainder of this paper is organized as follows. In Section II, we briefly review the notions and definitions that are relevant to this paper. In Section III, after depicting the framework of SA-ABSR), we formally describe its security model. In Section IV, we give a concrete construction on SA-ABSR based on a standard ABS scheme, sketch its security under the defined model, and analyze its performance complexity. Finally, we conclude this paper in Section V.

II. Preliminaries

In this section, we review some basic cryptographic notions and definitions that are to be used in this paper.

A. Bilinear Pairings and Complexity Assumptions

Let G be a group of a prime order p with a generator g. A function $\hat{e} : G \times G \rightarrow G_1$ is a bilinear map if it satisfies the following two properties [25]: 1) bilinear such that for all $g \in G$, and $a, b \in Z_p$, it holds $\hat{e}(g^a, g^b) = \hat{e}(g, g)^{ab}$; and 2) nondegenerate such that $\hat{e}(g, g) \neq 1$.

A group G is a bilinear group if the group operation in G is efficiently computable and there is a group G_1 and an efficiently computable bilinear map $\hat{e} : G \times G \rightarrow G_1$ as above.

Diffie–Hellman Exponent Problem [16]: The l-Diffie–Hellman exponent (l-DHE) problem is that given a tuple (g, g^a, ..., g^{a^l}, $g^{a^{l+2}}$, ..., $g^{a^{2l}}$), where $a \in Z_p$, it is hard to compute $g^{a^{l+1}}$.

B. Threshold ABS

Denote $\mathbb{A}$ by be the universe of possible attributes. Let $\mathbb{M}$ be the message space. In a threshold ABS scheme [16], every message $m \in \mathbb{M}$ is signed over a claim predicate $\Gamma_{k,S}$, where S is a subset over $\mathbb{A}$ and k is a threshold such that $1 \leq k \leq |S|$. We say that an attribute set $\mathbf{A} \in \mathbb{A}$ satisfies $\Gamma_{k,S}$ if $|\mathbf{A} \cap S| \geq k$.

1) Setup(1^λ) $\rightarrow$ ($pars$, msk): On input the security parameter λ, this setup algorithm outputs the public parameter $pars$ and a master private key msk for the AA.
2) Extract($pars$, msk, $\mathbf{A}$) $\rightarrow$ $sk_{\mathbf{A}}$: On input the public parameter $pars$, the master private key msk and an attribute set $\mathbf{A} \in \mathbb{A}$, this signing key extraction algorithm outputs the attribute-based signing key $sk_{\mathbf{A}}$ (if the signer is eligible to be issued with these attributes).
3) Sign($pars$, $sk_{\mathbf{A}}$, $\Gamma_{k,S}$, m) $\rightarrow$ σ: On input the public parameter $pars$, the signing key $sk_{\mathbf{A}}$, a claim-predicate $\Gamma_{k,S}$, where $S \subset \mathbb{A}$, $1 \leq k \leq |S|$ and a message $m \in \mathbb{M}$, this signing algorithm outputs a signature σ on the message m and the claim-predicate $\Gamma_{k,S}$.
4) Verify($pars$, σ) $\rightarrow$ *true/false*: On input the public parameter $pars$ and a signature σ on a message m and a claim-predicate $\Gamma_{k,S}$, this verification algorithm outputs *true* or *false* to denote whether the signature is valid or invalid.

We require that a threshold ABS scheme is correct, meaning that for any message $m \in \mathbb{M}$, any attribute set $\mathbf{A} \in \mathbb{A}$, any claim-predicate $\Gamma_{k,S}$ ($1 \leq k \leq |S|$) such that $|\mathbf{A} \cap S| \geq k$, if ($pars$, msk) $\leftarrow$ Setup(1^λ), $sk_{\mathbf{A}} \leftarrow$ Extract($pars$, msk, $\mathbf{A}$), $\sigma \leftarrow$ Sign($pars$, $sk_{\mathbf{A}}$, $\Gamma_{k,S}$, m), then Verify($pars$, σ) $\rightarrow$ *true*.

III. Framework and Security Definition

We describe the system framework of SA-ABSR, and define its security definitions in this section.

A. Framework

An SA-ABSR scheme consists of the following algorithms: system setup algorithm Setup, key generation algorithm KeyGen, partial signature generation algorithm SSign, signature generation algorithm USign, signature transformation algorithm Transform, server verification algorithm SVerify, user verification algorithm UVerify, and revocation algorithm Revoke.

1) Setup(1^λ) $\rightarrow$ (par, msk): Taking a security parameter λ as the input, this algorithm outputs the public parameter par and the master private key msk. This algorithm is run by the AA.
2) KeyGen(par, msk, $\mathbf{A}$) $\rightarrow$ ($tk_{id,\mathbf{A}}$, $sk_{id,\mathbf{A}}$): Taking the public parameter par, the master private key msk and an attribute set $\mathbf{A}$ of a user id as the input, this algorithm outputs an attribute-based signing key $sk_{id,\mathbf{A}}$ for the user id and a partial signing key $tk_{id,\mathbf{A}}$ for the server. This algorithm is run by the AA.
3) SSign(par, $tk_{id,\mathbf{A}}$, $\Gamma_{k,S}$, m) $\rightarrow$ σ': Taking the pubic parameter par, the partial signing key $tk_{id,\mathbf{A}}$ of a user id, a claim-predicate $\Gamma_{k,S}$ and a message m as the input, this algorithm outputs a partial signature σ' on the message m and the claim-predicate $\Gamma_{k,S}$. This algorithm is run by the server.
4) USign(par, $sk_{id,\mathbf{A}}$, σ') $\rightarrow$ σ: Taking the public parameter par, the attribute-based signing key $sk_{id,\mathbf{A}}$ and a partial signature σ' on a message m and a claim-predicate $\Gamma_{k,S}$ as the input, this algorithm outputs a signature σ. This algorithm is run by the singer.
5) Transform(par, σ) $\rightarrow$ (tk, $\tilde{\sigma}$): Taking the public parameter par and a signature σ on a message m and a claim-predicate $\Gamma_{k,S}$ as the input, this algorithm outputs a secret transformation key tk and a transformed signature $\tilde{\sigma}$. This algorithm is run by the verifier.

6) SVerify$(par, \tilde{\sigma}) \rightarrow B$: Taking the public parameter par and a transformed signature $\tilde{\sigma}$ on a message m and a claim-predicate $\Gamma_{k,S}$ as the input, this algorithm outputs an intermediate signature B. This algorithm is run by the server.
7) UVerify$(par, tk, B) \rightarrow true/false$: Taking the public parameter par, the transformation key tk and an intermediate signature B as the input, this algorithm outputs $true$ for a valid signature and $false$ otherwise. This algorithm is run by the verifier.
8) Revoke$(id, ul) \rightarrow \{id\} \cup ul$: On input an identity id and an initially empty revocation list ul, this algorithm adds id to the revocation list ul. For any user id in the list ul, the server immediately terminates generating partial signature for this user id. This algorithm is run by the AA.

We say that an SA-ABSR scheme is correct, meaning that for any message m, any (nonrevoked) user id with any attribute set $\mathbf{A}$, any claim-predicate $\Gamma_{k,S}$ ($1 \leq k \leq |S|$) such that $|\mathbf{A} \cap S| \geq k$, if $(par, msk) \leftarrow$ Setup(1^λ), $(tk_{id,\mathbf{A}}, sk_{id,\mathbf{A}}) \leftarrow$ KeyGen$(par, msk, \mathbf{A})$, $\sigma' \leftarrow$ SSign$(par, tk_{id,\mathbf{A}}, \Gamma_{k,S}, m)$, $\sigma \leftarrow$ USign$(par, sk_{id,\mathbf{A}}, \sigma')$, $(tk, \tilde{\sigma}) \leftarrow$ Transform(par, σ), $B \leftarrow$ SVerify$(par, \tilde{\sigma})$, then UVerify$(pars, tk, B) \rightarrow$ *true*.

B. Security Definitions

Similar to the security definitions for an ABS scheme, we require the SA-ABSR scheme to be unforgeable and anonymous.

1) Unforgeability: Unforgeability for an SA-ABSR scheme is defined by the following security game between a challenger Algorithm $\mathcal{B}$ and an adversary Algorithm $\mathcal{F}$, which is similar to that defined in [10].

1) *Setup:* Algorithm $\mathcal{B}$ generates the public parameter par and the master private key msk. Algorithm $\mathcal{B}$ gives Algorithm $\mathcal{F}$ the public parameter par, and keeps a list L storing $(id, \mathbf{A}, tk_{id,\mathbf{A}}, sk_{id,\mathbf{A}})$.
2) *Phase 1:* Algorithm $\mathcal{F}$ issues a sequence of queries to the following oracles.
   a) *Partial-signing-key oracle:* Algorithm $\mathcal{F}$ issues a partial signing key query on an identity id with an attribute set $\mathbf{A}$. If a tuple $(id, \mathbf{A}, tk_{id,\mathbf{A}}, sk_{id,\mathbf{A}})$ exists in the list L, Algorithm $\mathcal{B}$ returns the corresponding partial signing key $tk_{id,\mathbf{A}}$. Otherwise, Algorithm $\mathcal{B}$ runs the KeyGen algorithm to generate $(tk_{id,\mathbf{A}}, sk_{id,\mathbf{A}})$. Algorithm $\mathcal{B}$ returns the partial signing key $tk_{id,\mathbf{A}}$ to Algorithm $\mathcal{F}$. Also, Algorithm $\mathcal{B}$ adds $(id, \mathbf{A}, tk_{id,\mathbf{A}}, sk_{id,\mathbf{A}})$ to the list L.
   b) *Signing-key oracle:* Algorithm $\mathcal{F}$ issues a signing key query on an identity id and an attribute set $\mathbf{A}$. If a tuple $(id, \mathbf{A}, tk_{id,\mathbf{A}}, sk_{id,\mathbf{A}})$ exists in the list L, Algorithm $\mathcal{B}$ returns the corresponding signing key $sk_{id,\mathbf{A}}$. Otherwise, Algorithm $\mathcal{B}$ runs the KeyGen algorithm to generate $(tk_{id,\mathbf{A}}, sk_{id,\mathbf{A}})$, and returns the signing key $sk_{id,\mathbf{A}}$ to Algorithm $\mathcal{F}$. Also, Algorithm $\mathcal{B}$ adds $(id, \mathbf{A}, tk_{id,\mathbf{A}}, sk_{id,\mathbf{A}})$ to the list L.
   c) *Sign oracle:* Algorithm $\mathcal{F}$ issues a signing query on a message m and a claim-predicate $\Gamma_{k,S}$. Algorithm $\mathcal{B}$ runs the SSign and USign algorithms to generate a valid signature σ on the message m and the claim-predicate $\Gamma_{k,S}$, and returns the signature σ to Algorithm $\mathcal{F}$.
   d) *UVerify oracle:* Algorithm $\mathcal{F}$ issues a user verification query on a signature σ for a message m and a claim-predicate $\Gamma_{k,S}$. Algorithm $\mathcal{B}$ runs the Transform algorithm and sends a transformed signature to Algorithm $\mathcal{A}$. Algorithm $\mathcal{F}$ then forwards to Algorithm $\mathcal{B}$ an intermediate signature B, and Algorithm $\mathcal{B}$ sends the result of the UVerify algorithm to Algorithm $\mathcal{F}$.
3) *Output:* Algorithm $\mathcal{F}$ outputs a claim-predicate $\Gamma^*_{k,S}$, a message m^* and a signature σ^*. Algorithm $\mathcal{F}$ wins the game, if 1) $(\Gamma^*_{k,S}, m^*)$ has never been queried to the Sign oracle; 2) any id whose attributes $\mathbf{A}$ satisfy $\Gamma^*_{k,S}$ has never been queried to both the signing-key oracle and the partial-signing-key oracle; and 3) when Algorithm $\mathcal{B}$ returns a transformed signature $\tilde{\sigma}^*$ (by running the Transform algorithm) to Algorithm $\mathcal{F}$ and Algorithm $\mathcal{F}$ returns an intermediate signature B^* to Algorithm $\mathcal{B}$, the UVerify algorithm outputs *true*.

An SA-ABSR scheme $\mathcal{ABS}$ is unforgeable if the advantage function referring to the aforementioned security game

$$\mathbf{Adv}^{\mathrm{UNF}}_{\mathcal{F},\mathcal{ABS}}(\lambda) \stackrel{\mathrm{def}}{=} \Pr[\mathcal{F}\,\mathrm{wins}]$$

is negligible in the security parameter λ for any probabilistic polynomial-time (PPT) adversary Algorithm $\mathcal{F}$. In addition, an SA-ABSR scheme $\mathcal{ABS}$ is selectively unforgeable if an Init phase is added before the setup phase where Algorithm $\mathcal{F}$ chooses the claim-predicate $\Gamma^*_{k^*,S^*}$ that it aims to attack.

2) Anonymity: Similar to that defined in [10], anonymity for an SA-ABSR scheme is defined by the following security game between a challenger Algorithm $\mathcal{C}$ and an adversary Algorithm $\mathcal{A}$.

1) *Setup:* The same as that in the Unforgeability game.
2) *Phase 1:* The same as that in the Unforgeability game.
3) *Challenge:* Algorithm $\mathcal{A}$ sends a message m^* and two identities id^*_0 and id^*_1 with attributes $\mathbf{A}^*_0$ and $\mathbf{A}^*_1$ satisfying the same claim-predicate $\Gamma^*_{k,S}$ to Algorithm $\mathcal{C}$. Algorithm $\mathcal{C}$ randomly chooses a bit $b \in \{0, 1\}$, generates a signature σ^* on the message m^* under the claim-predicate $\Gamma^*_{k,S}$ by running the SSign algorithm using the partial signing key $tk_{id^*_b,\mathbf{A}^*_b}$ and the USign algorithm using the signing key $sk_{id^*_b,\mathbf{A}^*_b}$, and then, sends σ^* to Algorithm $\mathcal{A}$.
4) *Phase 2:* The same as that in Phase 1.
5) *Output:* Algorithm $\mathcal{A}$ outputs a guess b'. If $b' = b$, Algorithm $\mathcal{A}$ wins the game.

An SA-ABSR scheme $\mathcal{ABS}$ achieves anonymity if the advantage function referring to the aforementioned security game

$$\mathbf{Adv}^{\mathrm{ANON}}_{\mathcal{A},\mathcal{ABS}}(\lambda) \stackrel{\mathrm{def}}{=} |\Pr[b = b'] - 1/2|$$

is negligible in the security parameter λ for any PPT adversary Algorithm $\mathcal{A}$.

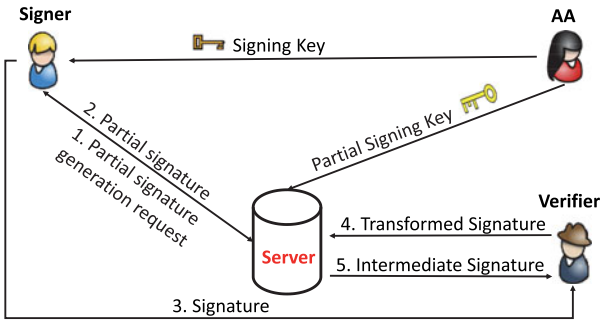


Fig. 1. System architecture for SA-ABSR.

IV. SA-ABSR

In this section, we propose a SA-ABSR scheme based on a standard ABS scheme, and analyze its security and performance.

A. System Overview

We describe the system architecture for an SA-ABSR scheme in Fig. 1. Specifically, a trusted AA, who generates the public parameter and keeps the master private key, issues a signing key to each signer (i.e., a resourced-constrained IIoT device) associated with his/her attributes; meanwhile, it generates a partial signing key for each signer and gives it to the server who keeps a list of all singer's partial signing keys. When a signer intends to generate a signature on a message under a claim predicate, he/she sends a partial signature generation request to the server, who then forwards to the signer a partial signature on the specified message and claim predicate. After receiving the partial signature from the server, the signer computes the signature using his/her signing key. In terms of the signature verification, the verifier first transforms a signature using a short-term transformation key (which is kept secret by the verifier), and then, sends the resulting transformed signature to the server. The server runs a verification algorithm on the transformed signature, and obtains an intermediate signature. Finally, the verifier checks the validity of the signature via the intermediate signature and the transformation key. When a signer is revoked, the server will not respond to any partial signature generation request from this signer.

B. Construction

Define the Lagrange coefficient as $\Delta_i^{\Omega}(x) = \prod_{j\in\Upsilon}^{j\neq i} \frac{x-j}{i-j}$, where $i \in Z_p$, Ω is a set of elements in Z_p. A polynomial $q(x)$ over Z_p with an order $d-1$ can be evaluated by using Lagrange interpolation as $q(x) = \sum_{i\in\Omega} q(i)\Delta_{i,\Omega}(x)$ where $|\Omega| = d$. On the basis of the ABS scheme proposed in [16], we describe the proposed SA-ABSR scheme as follows (notice that the techniques applied here can be applied to other ABS schemes (e.g., the scheme given in [2]) to achieve the SA-ABSR schemes).

1) *Setup:* On input the security parameter λ, this algorithm runs as follows. Let d be the upper bounded size of the threshold claim predicate allowed in the scheme. Denote $\mathbb{A}$ by the attribute space, and Ω for $|\Omega| = d$ by a default attribute set. Assume that each attribute in $\mathbb{A} \cup \Omega$ is an element from Z_p.
   a) It defines a bilinear pairing $\hat{e} : G \times G \rightarrow G_1$, where G is a group of a prime order p with a generator g. It randomly chooses $\alpha \in Z_p$, and computes $Z = \hat{e}(g,g)^{\alpha}$.
   b) It randomly chooses a vector $\vec{v} = (v_0, ..., v_l) \in {Z_p}^{l+1}$ where $l = 2d+1$, and computes $h_i = g^{v_i}$ for $i \in [0, l]$. In addition, it randomly chooses w_0, ..., $w_{n_m} \in G$, and defines a function F as $F(m) = w_0 \prod_{i=1}^{n_m} {w_i}^{m_i}$, where m_i are the ith bit of m.

   The master private key is $msk = \alpha$, and the public parameter is $par = (g, G, G_1, \hat{e}, p, Z, h_0, ..., h_l, w_0, ..., w_{n_m})$.
2) *KeyGen:* On input the public parameter par, the master private key msk and a user id with an attribute set $\mathbf{A}$, this algorithm randomly chooses $\beta_{id}, a_1, ..., a_{d-1} \in Z_p$, and defines a polynomial $q(w) = \sum_{i=1}^{d-1} a_i w^i + \beta_{id}$. Then, for each attribute $w \in \mathbf{A} \cup \Omega$, it randomly chooses r_w, $r_u \in Z_p$, and computes

$$P_w = g^{q(w)} \cdot {h_0}^{r_w},\ P_{w,0} = g^{r_w}$$
$$P_{w,i} = ({h_1}^{-w^i} \cdot h_{i+1})^{r_w}\ \forall\, i \in [i, l-1]$$
$$Q_w = g^{\alpha-\beta_{id}} \cdot {h_0}^{r_u},\ Q_{w,0} = g^{r_u}$$
$$Q_{w,i} = ({h_1}^{-w^i} \cdot h_{i+1})^{r_u}\ \forall\, i \in [i, l-1].$$

   It sends $tk_{id,\mathbf{A}} = (\{P_w,\ P_{w,0},\ \{P_{w,i}\}_{i\in[1,l-1]}\}_{w\in\mathbf{A}\cup\Omega})$ as the partial signing key for the user id to the server, and the signing key $sk_{id,\mathbf{A}} = (\{Q_w,\ Q_{w,0},\ \{Q_{w,i}\}_{i\in[1,l-1]}\}_{w\in\mathbf{A}\cup\Omega})$ to the user id.
3) *SSign:* On input the public parameter par, the partial signing key $tk_{id,\mathbf{A}}$ and a claim-predicate $\Gamma_{k,S}$ and a message m, this algorithm runs as follows.
   a) It randomly chooses a subset $S' \in \mathbf{A} \cap S$ and a default attribute subset $\Omega' \in \Omega$ such that $|S'| = k$ and $|\Omega'| = d-k$. Then, it defines a coefficient vector $\vec{b} = (b_1, ..., b_l) \in {Z_p}^l$ from the polynomial

$$\varphi(y) = \prod_{w\in S\cup\Omega'} (y-w) = \sum_{i=1}^{l} b_i y^{i-1}$$

   where b_i is set to 0 for $|S \cup \Omega'| + 2 \leq i \leq l$.
   b) Denote Υ by $S' \cup \Omega'$. It computes

$$P'_w = P_w \cdot \prod_{i=1}^{l-1} {P_{w,i}}^{b_{i+1}}$$
$$= g^{q(w)} \cdot \left(h_0 \prod_{i=1}^{l} {h_i}^{b_i}\right)^{r_w}\ \forall\, w \in \Upsilon$$
$$P'_0 = \prod_{w\in\Upsilon} {P'_w}^{\Delta_w^{\Upsilon}(0)} = g^{\beta_{id}} \cdot \left(h_0 \prod_{i=1}^{l} {h_i}^{b_i}\right)^{r}$$
$$P'_1 = \prod_{w\in\Upsilon} P_{w,0}^{\Delta_w^{\Upsilon}(0)} = g^{r}$$

   where $r = \sum_{w\in\Upsilon} \Delta_w^{\Upsilon}(0) \cdot r_w$.

c) It randomly chooses s_0, $s_1 \in Z_p$, and computes

$$\sigma'_0 = P'_0 \cdot \left(h_0 \prod_{i=1}^{l} {h_i}^{b_i} \right)^{s_0} \cdot F(m||\Omega'||S'||\Gamma_{k,S})^{s_1}$$

$$\sigma'_1 = P'_1 \cdot g^{s_0},\ \sigma'_2 = g^{s_1}.$$

d) It outputs the partial signature $\sigma' = (\Omega', S', \Gamma_{k,S}, m, \sigma'_0, \sigma'_1, \sigma'_2)$.

4) *USign:* On input the public parameter par, the signing key $sk_{id,\mathbf{A}}$ and a partial signature σ' under a claim predicate $\Gamma_{k,S}$ on a message m, this algorithm runs as follows.
   a) It defines a coefficient vector $\vec{b} = (b_1, ..., b_l) \in {Z_p}^l$ as that in the SSign algorithm.
   b) It randomly chooses $s \in Z_p$, and computes

$$\sigma_0 = \sigma'_0 \cdot Q_w \cdot \prod_{i=1}^{l-1} {Q_{w,i}}^{b_{i+1}} \cdot F(m||\Omega'||S'||\Gamma_{k,S})^s$$

$$\sigma_1 = \sigma'_1 \cdot Q_{w,0},\ \ \sigma_2 = \sigma'_2 \cdot g^s.$$

   c) It outputs the signature $\sigma = (\Omega', S', \Gamma_{k,S}, m, \sigma_0, \sigma_1, \sigma_2)$.
5) *Transform:* On input the public parameter par and a signature σ on a claim-predicate $\Gamma_{k,S}$ and a message m, this algorithm randomly chooses $t \in Z_p$, and computes

$$\tilde{\sigma}_0 = {\sigma_0}^t,\ \ \tilde{\sigma}_1 = {\sigma_1}^t,\ \ \tilde{\sigma}_2 = {\sigma_2}^t.$$

   It outputs the transformed signature $\tilde{\sigma} = (\Omega', S', \Gamma_{k,S}, m, \tilde{\sigma}_0, \tilde{\sigma}_1, \tilde{\sigma}_2)$, and the transformation key $tk = t$.
6) *SVerify:* On input the public parameter par and a transformed signature $\tilde{\sigma}$ on a message m under a claim-predicate $\Gamma(k, S)$, this algorithm runs as follows. It uses the default attribute subset Ω' to obtain the vector $\vec{b} = (b_1, ..., b_l)$ from the polynomial $\varphi(y)$ as defined in the SSign algorithm, and computes

$$\frac{\hat{e}(g, \tilde{\sigma}_0)}{\hat{e}(h_0 \prod_{i=1}^{l} {h_i}^{b_i}, \tilde{\sigma}_1) \cdot \hat{e}(F_2(m||\Omega'||S'||\Gamma_{k,S}), \tilde{\sigma}_2)} = B.$$

   It outputs the intermediate signature B.
7) *UVerify:* On input the public parameter par, an intermediate signature B and the transformation key tk, this algorithm checks whether $Z^t = B$. If so, it outputs $true$ indicating that the signature is valid. Otherwise, it outputs $false$.
8) *Revoke:* On input an identity id, this algorithm adds id to the revocation list ul. For any user id in the list ul, the server immediately terminates generating partial signature for this user id.

Notes: In the proposed SA-ABSR scheme, the server is semitrusted in the sense that it will not collude with any signer, i.e., it immediately terminates helping a signer with signature generation if this signer is revoked. It would be desirable to build an SA-ABSR scheme with an untrusted server who does not own any secret information and can collude with the signers. As a matter of fact, this can be achieved via the revocation mechanism based on the binary tree data structure proposed in [8], where the AA issues a long-term public signing key to the server for each signer and publicly broadcasts key updates at the beginning of each time period. The server keeps a list of public signing keys for all signers, but the server can only generate partial signing keys for nonrevoked signers from their corresponding long-term public signing keys and the key updates. Please refer to [8] for the details.

TABLE I
COMPARISON OF PERFORMANCES AMONG THE PROPOSED SA-ABSR SCHEME, OABS SCHEME, AND THEIR UNDERLYING ABS SCHEME

	Size of Signature	Sign	Verify	Revoke	SAV Secure
ABS [16]	3	$(6d+4)$E	$(2d+1)$E +3P	NA	NA
OABS [10]	3	$(2d+2)$E	2E	NA	×
SA-ABSR	3	$(2d+2)$E	4E	√	√

C. Security

Theorem 1: The proposed SA-ABSR scheme is secure under the l-DHE assumption.

Proof: At a high level, to prove the security of the proposed SA-ABSR scheme, we need to prove that it is selectively unforgeable and anonymous. We sketch the proof in terms of unforgeability and anonymity, respectively, as follows.

Unforgeability: Assume that there exists a forger Algorithm $\mathcal{F}$ that breaks the selective unforgeability of the proposed SA-ABSR scheme. Then, we can build a challenger Algorithm $\mathcal{B}$ that solves the l-DHE problem.

1) *Init:* Algorithm $\mathcal{F}$ selects a claim-predicate $\Gamma^*_{k^*,S^*}$ under which it aims to output a signature.
2) *Setup:* The same as the "Setup" phase in [16].
3) *Phase 1:* Algorithm $\mathcal{F}$ adaptively issues the following queries.
   a) *Partial-signing-key oracle:* For a partial-signing-key generation query on an identity id with an attribute set $\mathbf{A}$ from Algorithm $\mathcal{F}$, Algorithm $\mathcal{B}$ checks whether there exists a tuple $(id, \mathbf{A}, tk_{id,\mathbf{A}}, sk_{id,\mathbf{A}})$ in the list L. If so, it returns the partial-signing-key $tk_{id,\mathbf{A}}$. Otherwise, it randomly chooses $\beta_{id} \in Z_p$, and generates the partial-signing-key $tk_{id,\mathbf{A}} = (\{P_w, P_{w,0}, \{P_{w,i}\}_{i\in[1,l-1]}\}_{w\in\mathbf{A}\cup\Omega})$ as required.
   b) *Signing-key oracle:* For a signing-key generation query on an identity id with an attribute set $\mathbf{A}$ from Algorithm $\mathcal{F}$, Algorithm $\mathcal{B}$ checks whether there exists a tuple $(id, \mathbf{A}, tk_{id,\mathbf{A}}, sk_{id,\mathbf{A}})$ in the list L. If so, it returns the signing key $sk_{id,\mathbf{A}}$. Otherwise, it randomly chooses $\beta_{id} \in Z_p$, and follows the "private key queries" in [16] except that α will be replaced by $\alpha - \beta_{id}$. Thus, Algorithm $\mathcal{B}$ generates a well-formed signing key $sk_{id,\mathbf{A}} = (\{Q_w, Q_{w,0}, \{Q_{w,i}\}_{i\in[1,l-1]}\}_{w\in\mathbf{A}\cup\Omega})$.
   c) *Sign oracle:* The same as the "signing queries" phase in [16].
   d) *UVerify oracle:* For each user verification query on a signature σ for a message m under a claim-predicate

TABLE II
COMPARISON OF PERFORMANCES BETWEEN THE ABSAV SCHEME AND ITS UNDERLYING ABS SCHEME

	Size of Signature	Sign	Verify	Revoke	SAV Secure
ABS [2]	$n+d-k+3$	$2(n-k+2d)$E	$(n+d-k+2)$P	NA	NA
ABSAV [9]	$n+d-k+3$	$2(n-k+2d)$E	$2(n+d-k+1)$E + 2P	NA	$\surd$

$\Gamma_{k,S}$ issued by Algorithm $\mathcal{A}$, Algorithm $\mathcal{B}$ runs the transform Algorithm to generate and send a transformed signature $\tilde{\sigma}$ to Algorithm $\mathcal{F}$. After receiving the intermediate signature B from Algorithm $\mathcal{F}$, Algorithm $\mathcal{B}$ returns the result of the UVerify algorithm to Algorithm $\mathcal{F}$.

4) *Output:* The same as the "Forgery" phase in [16].

Similar to the analysis in [16], if Algorithm $\mathcal{F}$ has nonnegligible probability in forging a valid signature, then Algorithm $\mathcal{B}$ has nonnegligible probability in solving the l-DHE problem.

Anonymity: Assume that σ is produced by using a partial signing key and a signing key corresponding to an attribute set $\mathbf{A}$ satisfying $\mathbf{A} \cap S \geq k$. It is obvious that σ_1 and σ_2 are independent of the choice of $\mathbf{A}$. In addition, according to the definition of b_i, $1 \leq i \leq l$, these values also do not depend on the choice of $\mathbf{A}$, so σ_0 reveals no information about $\mathbf{A}$. That is, $(\sigma_0, \sigma_1, \sigma_2)$ has a uniform distribution over $G \times G \times G$. Since the proof of anonymity is similar to that in [16], we omit the details here. ■

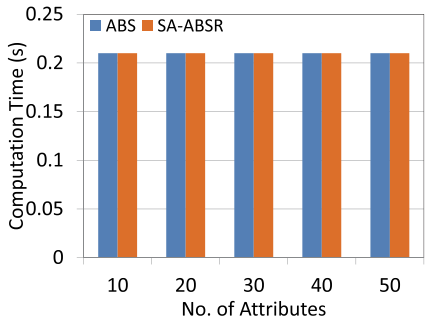
D. Performance Evaluation and Implementation

Denote d as the predefined size of the default attribute set and $\Gamma_{k,S}$ as the claim predicate with n being the size of S. Let "E" be exponentiation calculation, "P" be pairing calculation, "NA" be nonapplicable and "SAV secure" be server-aided verification secure. Considering building upon the same underlying ABS scheme, we compare the performance of the proposed SA-ABSR scheme with the secure outsourced ABS (OABS) scheme described in [10], and show their computational and storage overheads in Table I. Also, we compare the ABSAV scheme presented in [9] and its underlying ABS scheme [2] in Table II. It is straightforward to see that the proposed SA-ABSR scheme has an advantage over the existing solutions in that it reduces the workloads of both signing and verification phases yet supports the revocation function.

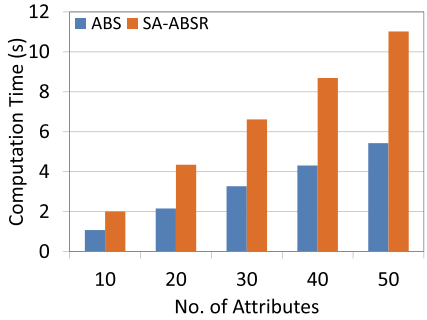
We implement the proposed SA-ABSR scheme and the underlying ABS scheme in the Charm [26][1] framework, which is developed to facilitate rapid implementation of cryptographic schemes and protocols. Since all the Charm routines are designed under the asymmetric groups, we transform the two constructions to the asymmetric setting before the implementation. That is, three groups G, $\hat{G}$, and G_1 are used and the pairing $\hat{e}$ is a function from $G \times \hat{G}$ to G_1. Note that it has been stated in [27] that the assumptions and the security proofs can be converted to the asymmetric setting in a generic transformation.

We use the Charm-0.43 version and the Python 3.4 in the implementation. Along with the Charm-0.43, the PBC library

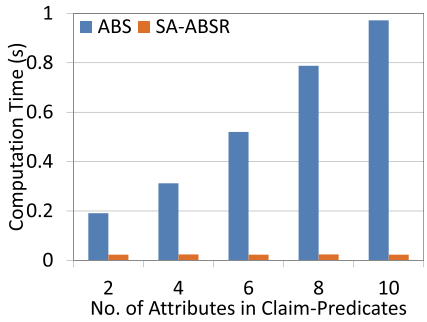
[1]For the explicit information on the Charm framework, please refer to [26].



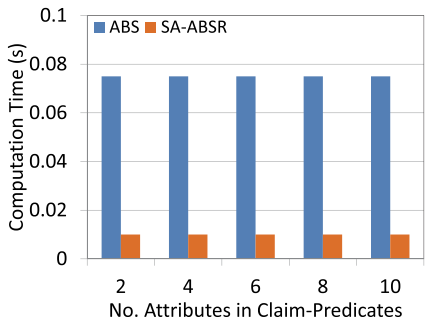
(a)



(b)



(c)



(d)

Fig. 2. Average computation time of the proposed SA-ABSR scheme and its underlying ABS scheme. (a) Setup. (b) Key generation. (c) User signing. (d) User verifying.

for the underlying cryptographic operations is installed. All experiments are run on a laptop with Intel Core i5-4210U CPU at 1.70 GHz and 8.00-GB RAM running 64-bit Ubuntu 16.04. All these experiments are conducted over the elliptic curve: MNT224, which is an asymmetric Type 3 pairings and provide a security level of 112-bit.

In the experiments, the size of the default attribute set is set to be $d = 10$. In this way, we first test the average computation time of the setup algorithm in the two schemes see Fig. 2(a)]. In both schemes, the setup time is immutable to the number of attributes, which is about 0.21 s. Then, we test the average computation time of running the key generation algorithms in terms of 10, 20, 30, 40, and 50 attributes in the two schemes see Fig. 2(b)]. As the key generation algorithm in the proposed SA-ABSR scheme is required to generate two keys: one for the signer and the other for the server, the computation time for the KeyGen algorithm in the SA-SBSR scheme almost doubles that in the original ABS scheme. Thereafter, we test the average computation time for a signer in generating a signature under a claim predicate for $d = 10$ and $k = 2, 4, 6, 8, 10$ in the two schemes see Fig. 2(c)]. It is not difficult to see that different from that in the underlying ABS scheme where the signer's computational cost in generating a signature raises linearly to the number of attributes, the signer's computational cost in generating a signature in the SA-ABSR scheme is around 0.03 s, which does not depend on the complexity of the claim-predicate involved. Finally, we test the average computation time for a verifier in checking the validity of a signature associated with a claim predicate with $d = 10$ and $k = 2, 4, 6, 8, 10$ in the two schemes see Fig. 2(d)]. It is straightforward to see that the computation time for a verifier in checking the validity of a signature does not change with the complexity of the claim predicate associated with the signature in both schemes, but the computation time for a verifier in checking the validity of a signature in the proposed SA-ABSR scheme is much less than that in the original ABS scheme. Nevertheless, the experimental results clearly show that the proposed SA-ABSR scheme is more efficient than the underlying ABS scheme in terms of users' workloads in both signing and verification algorithms.

V. Conclusion

An ABS [2], [3] is a promising solution to achieve anonymous authentication for many services, because an ABS only tells that "a signer whose attributes satisfy the claim-predicate endorses this message" [3] but nothing else. Unfortunately, because of the involvement of many exponentiation and/or pairing calculations, an ABS scheme incurs expensive computational costs in the signature generation and verification stages, which is undesirable for resource-constrained devices, e.g., a "smart" phone locates in an IIoT network. A desirable solution to this problem is to delegate a third party (e.g., a server) to perform the heavy calculations and leave the signer/verifier with a small amount of computations, which is known as "server-aided computation" [7]. However, existing solutions on the ABS with server-aided computation either are not secure or still require the verifier to perform a number of calculations. On the other hand, since a user's status in a system might change over time, user revocation has been a very challenging problem in the attribute-based cryptographic systems. With all these concerns in mind, in this paper, we proposed a notion called SA-ABSR to mitigate the workloads of signers/verifiers while achieving user revocation. After defining the security requirements for SA-ABSR, we presented a concrete construction of SA-ABSR on the basis of a standard ABS scheme, and proved the security of the proposed SA-ABSR scheme under the defined security model. Finally, we simulated the proposed SA-ABSR scheme and its underlying ABS scheme to evaluate the performance, and showed that the former considerably improves the efficiency of the latter.

References

[1] L. D. Xu, W. He, and S. Li, "Internet of things in industries: A survey," *IEEE Trans. Ind. Informat.*, vol. 10, no. 4, pp. 2233–2243, Nov. 2014.

[2] J. Li, M. H. Au, W. Susilo, D. Xie, and K. Ren, "Attribute-based signature and its applications," in *Proc. 5th ACM Symp. Inf., Comput. Commun. Security*, Beijing, China, Apr. 13–16, 2010, 2010, pp. 60–69.

[3] H. K. Maji, M. Prabhakaran, and M. Rosulek, "Attribute-based signatures," in *Topics in Cryptology—CT-RSA 2011—The Cryptographers' Track at the RSA Conference 2011, San Francisco, CA, USA, Feb. 14–18, 2011, Proceedings* (Lecture Notes in Computer Science), vol. 6558. New York, NY, USA: Springer, 2011, pp. 376–392.

[4] H. Cui, G. Wang, R. H. Deng, and B. Qin, "Escrow free attribute-based signature with self-revealability," *Inf. Sci.*, vol. 367-268, no. 6, pp. 660–672, 2016.

[5] S. Even, O. Goldreich, and S. Micali, "On-line/off-line digital schemes," in *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, Aug. 20–24, 1989, Proceedings*, (Lecture Notes in Computer Science), vol. 435. New York, NY, USA: Springer, 1989, pp. 263–275.

[6] A. C. Yao and Y. Zhao, "Online/offline signatures for low-power devices," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 2, pp. 283–294, Feb. 2013.

[7] W. Wu, Y. Mu, W. Susilo, and X. Huang, "Provably secure server-aided verification signatures," *Comput. Math. Appl.*, vol. 61, no. 7, pp. 1705–1723, 2011.

[8] H. Cui, R. H. Deng, Y. Li, and B. Qin, "Server-aided revocable attribute-based encryption," in *Computer Security—ESORICS 2016—21st European Symposium on Research in Computer Security, Heraklion, Greece, Sep. 26–30, 2016, Proceedings, Part II* (Lecture Notes in Computer Science), vol. 9879. New York, NY, USA: Springer, 2016, pp. 570–587.

[9] Z. Wang, R. Xie, and S. Wang, "Attribute-based server-aided verification signature," *Appl. Math. Inf. Sci.*, vol. 8, no. 6, pp. 3183–3190, 2014.

[10] X. Chen, J. Li, X. Huang, J. Li, Y. Xiang, and D. S. Wong, "Secure outsourced attribute-based signatures," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 12, pp. 3285–3294, Dec. 2014.

[11] H. Cui, Y. Mu, and F. Guo, "Server-aided identity-based anonymous broadcast encryption," *IJSN*, vol. 8, no. 1, pp. 29–39, 2013.

[12] B. Qin, R. H. Deng, Y. Li, and S. Liu, "Server-aided revocable identity-based encryption," in *Computer Security—ESORICS 2015-20th European Symposium on Research in Computer Security, Vienna, Austria, Sep. 21–25, 2015, Proceedings, Part I* (Lecture Notes in Computer Science), vol. 9326. New York, NY, USA: Springer, 2015, pp. 286–304.

[13] S. S. M. Chow, M. H. Au, and W. Susilo, "Server-aided signatures verification secure against collusion attack," *Inf. Security* Tech. Rep., vol. 17, no. 3, pp. 46–57, 2013.

[14] R. Chen, Y. Mu, G. Yang, F. Guo, X. Huang, X. Wang, and Y. Wang, "Server-aided public key encryption with keyword search," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 12, pp. 2833–2842, Dec. 2016.

[15] Z. Liu, H. Yan, and Z. Li, "Server-aided anonymous attribute-based authentication in cloud computing," *Future Gener. Comput. Syst.*, vol. 52, pp. 61–66, 2015.

[16] J. Herranz, F. Laguillaumie, B. Libert, and C. Ràfols, "Short attribute-based signatures for threshold predicates," in *Topics in Cryptology—CT-RSA 2012—The Cryptographers' Track at the RSA Conference 2012, San Francisco, CA, USA, Feb. 27–Mar. 2, 2012, Proceedings* (Lecture Notes in Computer Science), vol. 7178. New York, NY, USA: Springer, 2012, pp. 51–67.

[17] T. Okamoto and K. Takashima, "Efficient attribute-based signatures for non-monotone predicates in the standard model," in *Public Key Cryptography—PKC 2011—14th International Conference on Practice and Theory in Public Key Cryptography, Taormina, Italy, Mar. 6–9, 2011, Proceedings* (Lecture Notes in Computer Science), vol. 6571. New York, NY, USA: Springer, 2011, pp. 35–52.
[18] J. Wei, X. Huang, X. Hu, and W. Liu, "Revocable threshold attribute-based signature against signing key exposure," in *Information Security Practice and Experience—11th International Conference, ISPEC 2015, Beijing, China, May 5–8, 2015, Proceedings* (Lecture Notes in Computer Science), vol. 9065. New York, NY, USA: Springer, 2015, pp. 316–330.
[19] Y. Ren, C. Tang, G. Wang, and D. S. Wong, "Attribute-based signature schemes with accountability," *Int. J. Inf. Commun. Technol.*, vol. 7, no. 2/3, pp. 141–158, 2015.
[20] S. T. Ali and B. B. Amberker, "Attribute-based group signature without random oracles with attribute anonymity," *Int. J. Inf. Commun. Security*, vol. 6, no. 2, pp. 109–132, 2014.
[21] J. Wei, W. Liu, and X. Hu, "Forward-secure threshold attribute-based signature scheme," *Comput. J.*, vol. 58, no. 10, pp. 2492–2506, 2015.
[22] T. Matsumoto, K. Kato, and H. Imai, "Speeding up secret computations with insecure auxiliary devices," in *Advances in Cryptology—CRYPTO '88, 8th Annual International Cryptology Conference, Santa Barbara, California, USA, Aug. 21–25, 1988, Proceedings*, (Lecture Notes in Computer Science), vol. 403. New York, NY, USA: Springer, 1988, pp. 497–506.
[23] F. Guo, Y. Mu, W. Susilo, and V. Varadharajan, "Server-aided signature verification for lightweight devices," *Comput. J.*, vol. 57, no. 4, pp. 481–493, 2014.
[24] Y. Yang, J. K. Liu, K. Liang, K. R. Choo, and J. Zhou, "Extended proxy-assisted approach: Achieving revocable fine-grained encryption of cloud data," in *Computer Security—ESORICS 2015—20th European Symposium on Research in Computer Security, Vienna, Austria, Sep. 21–25, 2015, Proceedings, Part II* (Lecture Notes in Computer Science), vol. 9327. New York, NY, USA: Springer, 2015, pp. 146–166.
[25] D. Boneh and M. K. Franklin, "Identity-based encryption from the weil pairing," *SIAM J. Comput.*, vol. 32, no. 3, pp. 586–615, 2003.
[26] J. A. Akinyele, C. Garman, I. Miers, M. W. Pagano, M. Rushanan, M. Green, and A. D. Rubin, "Charm: a framework for rapidly prototyping cryptosystems," *J. Cryptographic Eng.*, vol. 3, no. 2, pp. 111–128, 2013.
[27] Y. Rouselakis and B. Waters, "Practical constructions and new proof methods for large universe attribute-based encryption," in *Proc. 2013 ACM SIGSAC Conf. Comput. Commun. Security*, 2013, pp. 463–474.

Hui Cui received the Ph.D. degree from the School of Computing and Information Technology, University of Wollongong, Wollongong, NSW, Australia, in 2015.

She is currently a Research Fellow with the Royal Melbourne Institute of Technology University, Melbourne, Australia.

Robert H. Deng (SM'04-F'16) received the Ph.D. degree in electrical and computer engineering from the Illinois Institute of Technology, Chicago, IL, USA, in 1985.

He has been a Professor with the School of Information Systems, Singapore Management University, Singapore, since 2004. His research interests include data security and privacy, multimedia security, and network and system security.

Prof. Deng has served/is serving on the editorial boards of many international journals in security, such as IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, the *International Journal of Information Security*, and *IEEE Security and Privacy Magazine*. He is the Chair of the Steering Committee of the ACM Asia Conference on Computer and Communications Security.

Joseph K. Liu received the Ph.D. degree in information engineering from the Chinese University of Hong Kong, Hong Kong, in July 2004, specializing in cyber security, protocols for securing wireless networks, privacy, authentication, and provable security.

He is a Senior Lecturer with the Faculty of Information Technology, Monash University, Melbourne, Australia. His current research interests include cyber security in the cloud computing paradigm, smart city, lightweight security, and privacy enhanced technology.

Xun Yi received the Ph.D. degree in electronic engineering from Xidian University, Xi'an, China, in 1995.

He is currently a Professor with the School of Science, Royal Melbourne Institute of Technology University, Melbourne, Australia. His research interests include data privacy, cloud security, cybersecurity, wireless and mobile security, and applied cryptography. He has published more than 150 research papers in international journals, such as IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, IEEE TRANSACTIONS ON WIRELESS COMMUNICATION, IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, IEEE TRANSACTIONS ON CIRCUIT AND SYSTEMS, and conference proceedings.

Prof. Yi has been an Associate Editor for the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING since 2014.

Yingjiu Li (M'03) received the Ph.D. degree in information technology from the Center for Secure Information Systems, Department of Information and Software Engineering, George Mason University, Fairfax, VA, USA, in 2003.

He is currently an Associate Professor with the School of Information Systems, Singapore Management University, Singapore. His research interests include RFID security and privacy, mobile and system security, applied cryptography and cloud security, and data application security and privacy. He has published more than 130 technical papers in international conferences and journals, and served in the program committees for more than 80 international conferences and workshops.

Prof. Li is a Senior Member of the ACM and a Member of the IEEE Computer Society.