Robust Multi-Factor Authentication for Fragile Communications

Xinyi Huang, Yang Xiang, *Senior Member, IEEE*, Elisa Bertino, Jianying Zhou, and Li Xu, *Member, IEEE*

Abstract—In large-scale systems, user authentication usually needs the assistance from a remote central authentication server via networks. The authentication service however could be slow or unavailable due to natural disasters or various cyber attacks on communication channels. This has raised serious concerns in systems which need robust authentication in emergency situations. The contribution of this paper is two-fold. In a slow connection situation, we present a secure generic multi-factor authentication protocol to speed up the whole authentication process. Compared with another generic protocol in the literature, the new proposal provides the same function with significant improvements in computation and communication. Another authentication mechanism, which we name stand-alone authentication, can authenticate users when the connection to the central server is down. We investigate several issues in stand-alone authentication and show how to add it on multi-factor authentication protocols in an efficient and generic way.

Index Terms—Authentication, efficiency, privacy, stand-alone, multi-factor

1 Introduction

Information systems are vulnerable to many kinds of cyber attacks, one of which is unauthorized access. As an indispensable component for building secure information systems, authentication can prevent devices and services from unauthorized access by validating user identity. Authentication is an interactive process between a user and an authentication server. A simple but representative run of authentication is as follows: (1) The user first sends out an authentication request; (2) The authentication server responds with a challenge; and (3) The user proves his/her identity by calculating a response which is validated by the server. Complicated designs of authentication involve multi-round message exchanges to satisfy specific security requirements.

It is evident that any communication delays or failures between the user and the authentication server will have significant impacts on authentication. Thus a reliable communication environment is critical for completing authentication. However, it would not be an easy task to ensure reliable communications in large-scale information systems, not only because of cyber attacks but also natural disasters (which could destroy communication infrastructure).

- *X. Huang and L. Xu are with Fujian Provincial Key Laboratory of Network Security and Cryptology, School of Mathematics and Computer Science, Fujian Normal University, Fuzhou 350000, Fujian, China. E-mail: {xyhuang, xuli}@fjnu.edu.cn.*
- *Y. Xiang is with the School of Information Technology, Deakin University, Burwood, Victoria, Australia. E-mail: yang@deakin.edu.au.*
- *E. Bertino is with the Department of Computer Science, Purdue University, 305 N. University Street, West Lafayette, IN. E-mail: bertino@purdue.edu.*
- *J. Zhou is with the Infocomm Security Department, Institute for Infocomm Research (I^2R), 1 Fusionopolis Way, Singapore. E-mail: jyzhou@i2r.a-star.edu.sg.*

Manuscript received 14 Apr. 2013; revised 21 Sept. 2013; accepted 22 Dec. 2013. Date of publication 1 Jan. 2014; date of current version 12 Nov. 2014. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TDSC.2013.2297110

1.1 Motivations and Goals

This paper aims to the design of robust multi-factor authentication in large-scale information systems with fragile communication environments. It is motivated by a specific situation depicted in *Guidelines for Smart Grid Cyber Security* from the US National Institute of Standards and Technology (NIST) [1].

As an example of large-scale information systems, Smart Grid is a form of electricity network utilizing modern digital communication technologies. Such technologies will turn the current centralized power system into a more distributed system with improved efficiency, reliability and safety of power delivery and use. But the same advances will also bring an array of new security challenges, one of which is "Authenticating and Authorizing Users to Substation Intelligent Electronic Devices (IEDs)" (Section 7.2.1 in [1]).

An IED is a microprocessor-based controller of power system equipment and designed to accommodate critical infrastructure protection programs. Substation IEDs, which review data from sensors and issue commands accordingly, are of great importance to the stability of the whole grid. A circuit breaker, for example, can immediately discontinue electrical flow once a fault condition is detected.

Due to its importance, authenticating and authorizing users to substation IEDs must address the following issues:

1. Efficient Authentication: *Slow Connection*. A substation IED may be accessed locally and may also be accessed remotely from a different physical location. Substations generally have some sort of connectivity to the control center, and this makes it possible to authenticate users with the assistance from central authentication servers by acting as a relay (as shown in Fig. 1). But the connections between substations and central servers can be very slow, and an efficient protocol is a necessity.

2. Stand-Alone Authentication (SAA): *No Connection.* Reliance on central authentication servers is not practical. Even if there is a connection between substations and center servers, authentication should continue to apply for personnel

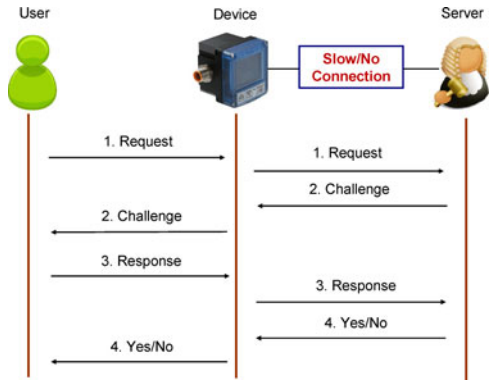


Fig. 1. Authentication to substation IEDs.

accessing devices locally in the substation when control center communications are down (as shown in Fig. 1). Such a feature is called stand-alone in this paper. Stand-alone authentication is particularly necessary in emergency situations when one must login to IEDs and take necessary actions, but the center server is unavailable due to natural disasters or cyber attacks. A local audit trail must be created every time stand-alone authentication is activated.

3. *Multi-Factor Authentication*: *High-Level Security.* Authenticating users by multiple factors, e.g., password, smart-card and biometrics, can provide a high level of security. This is necessary when authenticating users to key devices in information systems. As an example, substation is one of core building blocks on Smart Grid and plays a critical role in maintaining the stability of the whole system. Thus authenticating users to substation IEDs requires a higher security tier than the authentication on other less-crucial devices. Authentication based on multiple factors is recommended as a high-level security requirement in "SG. IA-4: User Identification and Authentication" [1].

1.2 Contributions

Our aim is to design authentication protocols satisfying aforementioned goals.

We first present an improved generic three-factor authentication protocol to speed up the whole authentication process in a slow connection environment. Our protocol shares many desirable properties with another generic three-factor authentication framework recently proposed in [2]: Both use smart-card-based password authentication and fuzzy extractor as building blocks, and provide privacy protection on biometrics, error-tolerance on biometrics and flexible authentication. The improvement is in the efficiency: Our protocol has significant advantages in terms of computation and communication over the protocol in [2]. We believe the newly proposed protocol provides an optimized tradeoff among efficiency, security and privacy, and thus is a promising solution in information systems with low bandwidth communication environments.

We then focus our attention on stand-alone authentication so that users can be authenticated correctly even the connection to the authentication server is down. We give a generic design of stand-alone authentication based on any existing multi-factor authentication protocols:

1) The protocol supports multi-factor authentication with formal security proofs;
2) Stand-alone authentication is under the control of the authentication server: only eligible users and devices can carry out stand-alone authentication;
3) Stand-alone authentication does not introduce any significant additional computation burden at the user side: the computation cost at the user side is almost the same as that in a normal authentication; and
4) It is applicable in a dynamic environment: user/device can carry out stand-alone authentication immediately once the registration is complete, and there is no need of system-wide update due to newly joined users/devices. Furthermore, the added computation and storage cost is independent of the number of users/devices in the system.

1.3 Related Work

As a fundamental security solution, authentication has been studied both extensively and intensively. A number of authentication protocols have been proposed. In the following, we only review some results which are most relevant to this paper, namely authentication involving biometrics and stand-alone authenticaiton.

Authentication using a cryptographic key extracted from biometrics can be traced back to [3]. The authentication server in [3] has a database of biometric templates of all registered users, and this could put user privacy at risk. A privacy preserving multi-factor authentication protocol was proposed in [4], where one's biometric information is kept secret from the authentication server. Their design is based on zero-knowledge proof [5], and either the authentication server must maintain a database of all users' commitments, or each user must store the commitment and the server's signature on his/her devices (e.g., mobile phones). Fan and Lin [6] proposed a more efficient three-factor authentication protocol with privacy protection on biometrics. In their protocol, user first chooses a random string and encrypts it using his/her biometric template during the registration. The result, called sketch, is stored in the smart-card. During the authentication, user must convince the server that he/she can decrypt the sketch, which needs correct biometrics.[1] Another biometric-based remote client authentication scheme using smart-card and password was given by Li and Hwang [7]. Their solution, however, does not support error-tolerance (which is essential in biometric authentication) and needs a fully-trusted remote device to verify biometric credential. Very recently, a generic three-factor authentication framework has been proposed in [2]. The framework provides a secure upgrade from smart-card-based password authentication to three-factor authentication and protects the privacy of biometrics. Additionally, the framework retains several practice-friendly properties of the underlying two-factor authentication, such as mutual authentication, security against replay attacks, key agreement, etc.

1. Biometric characteristics are prone to various noises during data collecting. Here, "correct biometrics" could be different from the biometric template extracted at the registration phase within a pre-defined range.

TABLE 1
Notations

$\mathcal{AS}$:	Authentication Server	SKE.KGen:	Key Generation in Symmetric Key Cipher
$PK_{\mathcal{AS}}$:	Public system parameters	SKE.ENC:	Encryption in Symmetric Key Cipher
$SK_{\mathcal{AS}}$:	$\mathcal{AS}$'s private information	SKE.DEC:	Decryption in Symmetric Key Cipher
κ:	Security Parameter	HASH:	A Cryptographic Hash Function
$\mathcal{C}$:	User	MAC:	A Message Authentication Code Function
PW:	A Password Chosen by $\mathcal{C}$	$\|\|$:	String Concatenation
Bio:	Biometric Information of $\mathcal{C}$	PKS.KGen:	Key Generation of Digital Signatures
SC:	Smart Card of $\mathcal{C}$	PKS.Sig:	Signing Algorithm of Digital Signatures
$D_{\mathcal{C}}$:	Data stored in SC	PKS.Ver:	Verification Algorithm of Digital Signatures
$D_{\mathcal{AS}}$:	Data maintained by $\mathcal{AS}$		

Stand-alone authentication allows a device to authenticate users when the connection with the central authentication server is down. While the notion has never been formally studied by academics, several authentication approaches are naturally stand-alone. One of these approaches is digital signature. To produce digital signatures, one must have a pair of keys: a public key and a private mathematically related key. There is a third party producing public key certificates (if needed) or (partial) private key for the user. One uses his/her private key to produce digital signatures which can be validated using the corresponding public key. A valid digital signature under a public key requires the corresponding private key and thus can prove the identity of a remote user during authentication. Using digital signatures, anyone with public information (e.g., public key, system parameter, etc.) can complete the authentication even the third party is offline.[2] More generally, such an approach falls into the category of smart-card-based authentication, where a smart-card is used to store private key and other relevant data. However, to the best of our knowledge, little attention has been put on stand-alone authentication involving multiple authentication factors.

Organization. The remainder of this paper is organized as follows. Section 2 reviews general definitions of authentication and fuzzy extractor. A new generic framework for three-factor authentication and its performance analysis are given in Section 3. Our design of stand-alone authentication is presented in Section 4. Section 6 concludes this paper.

2 Preliminaries

This section reviews general definitions of authentication and fuzzy extractor.

2.1 Outline of Authentication Protocols

We first give a high-level description of an authentication protocol, which is made up of four sub-protocols: Initial, Reg, Login-Auth and Credential-Update. Table 1 summarizes the notations used in this paper.

Initial: The authentication server $\mathcal{AS}$, taking a security parameter κ as input, generates two system parameters $PK_{\mathcal{AS}}$ and $SK_{\mathcal{AS}}$. $PK_{\mathcal{AS}}$ is published in the system, and $SK_{\mathcal{AS}}$ is kept secret by $\mathcal{AS}$. The security parameter κ determines the system's security level, such as the size of $PK_{\mathcal{AS}}$ and $SK_{\mathcal{AS}}$.

User-Reg: An interactive protocol between $\mathcal{C}$ and $\mathcal{AS}$, where $\mathcal{C}$ registers his/her initial authentication credential (e.g., PW and/or Bio) with $\mathcal{AS}$ in a secure environment.

The output of User-Reg protocol contains two pieces of data $(D_{SC}, D_{\mathcal{AS}})$: (1) D_{SC} is stored in a smart-card SC kept by $\mathcal{C}$; and (2) $D_{\mathcal{AS}}$ contains the registration information (e.g., expiration date) and is put in a database maintained by $\mathcal{AS}$. Note that $D_{\mathcal{AS}}$ is an optional output and there are designs of authentication protocols where $\mathcal{AS}$ does not need to maintain such information.

An execution of this protocol is denoted by

$$\mathcal{C}[Initial\ Credential] \overset{\text{User-Reg}}{\Longleftrightarrow} \mathcal{AS}[SK_{\mathcal{AS}}] \rightarrow \{D_{SC}, D_{\mathcal{AS}}\}.$$

Login-Auth: This is another interactive protocol between $\mathcal{C}$ and $\mathcal{AS}$, with which $\mathcal{AS}$ authenticates $\mathcal{C}$. An execution of this protocol is denoted by

$$\mathcal{C}[Credential] \overset{\text{Login-Auth}}{\Longleftrightarrow} \mathcal{AS}[SK_{\mathcal{AS}}, D_{\mathcal{AS}}] \rightarrow \{1, 0\}.$$

Here, the authentication factor set $Credential$ is a subset of $\{PW, Bio, SC\}$. As an example, $Credential$ is $\{PW, SC\}$ in smart-card-based password authentication and $\{PW, SC, Bio\}$ in three-factor authentication. The output of this protocol is "1" (if the authentication is successful) or "0" (otherwise).

Credential-Update: This protocol allows $\mathcal{C}$ to change authentication credential after a successful authentication, i.e., Login-Auth outputs "1". D_{SC} and $D_{\mathcal{AS}}$ will be updated accordingly.

Two-Factor/Three-Factor Authentication Protocols. To make notations easy to follow, we put a prefix 2F/3F before each sub-protocol to specify the number of authentication factors:

- A two-factor authentication protocol is made up of {2F-Initial, 2F-User-Reg, 2F-Login-Auth and 2F-Credential-Update}, and
- A three-factor authentication protocol is made up of {3F-Initial, 3F-User-Reg, 3F-Login-Auth and 3F-Credential-Update}.

Security Requirements. Intuitively, the security of an authentication protocol requires that without all authentication factors, no attackers can impersonate the client and have a successful run of Login-Auth protocol with the authentication server. In Section 5.1, we employ the

2. In stand-alone authentication, it is still necessary that devices must obtain user revocation information from central server on a regular basis.

approach in [8] and define the security of multi-factor authentication.

2.2 Fuzzy Extractor

Following the approach in [2], we use fuzzy extractor to extract a secret from biometrics. In this case, there is no need to maintain a database of biometrics at the server side, which helps protect user privacy. Below is a brief review of the fuzzy extractor introduced in [9].

Metric Space. A metric space is a set $\mathcal{M}$ with a distance function $\mathsf{dis} : \mathcal{M} \times \mathcal{M} \rightarrow \mathbb{R}^+ = [0, \infty)$ which obeys various natural properties. One example of metric space is *Hamming metric*: $\mathcal{M} = \mathcal{F}^n$ is over some alphabet $\mathcal{F}$ (e.g., $\mathcal{F} = \{0, 1\}$) and $\mathsf{dis}(w, w')$ is the number of positions in which they differ.

Statistic Distance. The statistical distance between two probability distributions A and B is denoted by $\mathbf{SD}(A, B) = \frac{1}{2}\sum_v |\Pr(A = v) - \Pr(B = v)|$.

Entropy. The min-entropy $\mathbf{H}_\infty(A)$ of a random variable A is $-\log(\max_a \Pr[A = a])$.

Fuzzy Extractor. A fuzzy extractor extracts a nearly random string R from its biometric input w in an error-tolerant way. If the input changes but remains close, the extracted R remains the same. To assist in recovering R from a biometric input w', a fuzzy extractor outputs an auxiliary string P. However, R remains uniformly random even given P. The fuzzy extractor is formally defined as below.

Definition 1 (Fuzzy Extractor). *An $(\mathcal{M}, m, \ell, t, \epsilon)$ fuzzy extractor is given by two procedures* $(\mathsf{Gen}, \mathsf{Rep})$.

1.

$$\xrightarrow{BioData:w} \boxed{\mathsf{Gen}} \rightarrow \begin{cases} R & : \text{ Random String,} \\ P & : \text{ Auxiliary String.} \end{cases}$$

Gen is a probabilistic generation procedure, which on (biometric) input $w \in \mathcal{M}$ outputs an "extracted" string $R \in \{0,1\}^\ell$ and an auxiliary string P. For any distribution W on $\mathcal{M}$ of min-entropy m, if $< R, P > \leftarrow \mathsf{Gen}(W)$, then we have $\mathbf{SD}(< R, P >, < U_\ell, P >) \leq \epsilon$, which should be negligible. Here, U_ℓ denotes the uniform distribution on ℓ-bit binary strings.

2.

$$\xrightarrow[P]{BioData:w'} \boxed{\mathsf{Rep}} \rightarrow R \text{ if } \mathsf{dis}(w, w') \leq t.$$

Rep is a deterministic reproduction procedure allowing to recover R from the corresponding auxiliary string P and any vector w' close to w: for all $w, w' \in \mathcal{M}$ satisfying $\mathsf{dis}(w, w') \leq t$, if $< R, P > \leftarrow \mathsf{Gen}(w)$, then we have $\mathsf{Rep}(w', P) = R$.

3 A New Generic Design of Multi-Factor Authentication

This section describes a generic and efficient design of three-factor authentication for slow communication situations. The protocol is based on two-factor (password and smart-card) authentication and fuzzy extractor. The details of our design, together with the performance and security analysis, are given in the following sections.

3.1 Our Design

3F-Initial: This phase generates a public parameter and a secret parameter for our three-factor authentication. Given a security parameter κ,

1. $\mathcal{AS}$ runs $\mathsf{2F\text{-}Initial}(\kappa)$ to obtain a pair (PK_{2F}, SK_{2F}).
2. $\mathcal{AS}$ runs $\mathsf{SKE.KGen}(\kappa)$ to obtain a secret SK_E.
3. The public parameter $PK_{\mathcal{AS}}$ is PK_{2F}, and the corresponding secret parameter $SK_{\mathcal{AS}}$ is (SK_{2F}, SK_E).

3F-User-Reg: As in the existing authentication protocols, we assume the registration phase is performed in a secure and reliable environment. This phase is made up of the following steps:

1. An initial password PW is chosen by $\mathcal{C}$.
2. $\mathcal{C}[PW] \overset{\mathsf{2F-User-Reg}}{\Longleftrightarrow} \mathcal{AS}[SK_{2F}] \rightarrow \{D_{\mathcal{C}}, D_{\mathcal{AS}}\}$. $\mathcal{C}$ registers PW with $\mathcal{AS}$ using $\mathsf{2F\text{-}User\text{-}Reg}$.
3. $\mathsf{Gen}(Bio) \rightarrow (R, P)$. A pair (R, P) is produced by a trusted device using $\mathcal{C}$'s biometric template Bio and algorithm Gen in the fuzzy extractor (defined in Section 2.2).
4. $\mathcal{C}$ extracts a MAC key $K_B = \mathsf{HASH}(R)$ and sends (K_B, P) to $\mathcal{AS}$.
5. Upon receiving (K_B, P), $\mathcal{AS}$ uses SK_E to encrypt K_B:

$$D_K = \mathsf{SKE.Enc}(\mathcal{C} \| K_B \| \mathsf{HASH}(\mathcal{C} \| K_B \| SK_E), SK_E).$$

The tag $\mathsf{HASH}(\mathcal{C} \| K_B \| SK_E)$ is for integrity check.

6. Let $D_{Bio} = (P, D_K, \mathsf{HASH}, \mathsf{Rep})$. Here, Rep is the reproduction algorithm in the fuzzy extractor.
7. At the end of this protocol, $\mathcal{C}$ is given a smart-card SC, which contains $D_{\mathcal{C}}$ (data associated with the underlying two-factor authentication) and D_{Bio}.

Remark. In our protocol, $\mathcal{AS}$ will need to maintain a local database of $D_{\mathcal{AS}}$, if this is required in the underlying smart-card-based password authentication. □

This completes the description of registration.

3F-Login-Auth: At the beginning, $\mathcal{C}$ inserts the smart-card $SC = \{D_{\mathcal{C}}, D_{Bio} = (P, D_K, \mathsf{HASH}, \mathsf{Rep})\}$ into a card reader and inputs PW and Bio.[3] Let Bio' be the biometric template extracted at this phase.

1. A random string R is calculated using Bio', P and Rep, based on which the MAC key K_B is retrieved

$$R = \mathsf{Rep}(Bio', P) \text{ and } K_B = \mathsf{HASH}(R).$$

2. $\mathcal{C}$ and $\mathcal{AS}$ perform a two-factor authentication based on PW and SC:

$$\mathcal{C}[PW, SC(D_{\mathcal{C}})] \overset{\mathsf{2\text{-}Login\text{-}Auth}}{\Longleftrightarrow} \mathcal{AS}[SK_{2F}, D_{\mathcal{AS}}].$$

- 3F-Login-Auth outputs "0" if the above authentication fails.

3. The biometric extractor is trusted to extract biometrics properly and never divulges the biometric information.

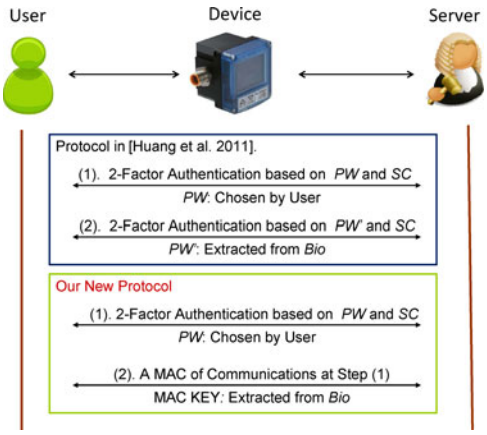


Fig. 2. Comparison.

- Otherwise, $\mathcal{AS}$ is convinced that the remote user has a correct password of $\mathcal{C}$. The authentication then goes to the next step.

3. A MAC value is calculated as

$$\mathsf{Tag} = \mathsf{MAC}_{K_B}(\mathsf{TRANSCRIPTS}).$$

Here, TRANSCRIPTS are messages exchanged during Step 2. The pair (Tag, D_K) is sent to $\mathcal{AS}$.

4. Upon receiving (Tag, D_K), $\mathcal{AS}$ performs the following operations:

- $\mathsf{SEC.Dec}(D_K, SK_E) \rightarrow$ "$\mathcal{C}' \| K'_B \| \mathsf{HashValue}$",
- 3F-Login-Auth outputs "1" if

$$\mathcal{C}' = \mathcal{C}, \mathsf{HashValue} = \mathsf{HASH}(\mathcal{C}' \| K'_B \| SK_E), \text{ and}$$

$$\mathsf{Tag} = \mathsf{MAC}_{K'_B}(\mathsf{TRANSCRIPTS}).$$

  The first two checks ensure that the MAC key retrieved from D_K has been registered for $\mathcal{C}$. The last operation verifies that the remote user possess a correct MAC key and thus the Bio.
- Otherwise, 3F-Login-Auth outputs "0".

Remark. Step 3-4 are only needed when authentication requires biometrics. □

3F-Credential-Update: After a successful login (i.e., 3F-Login-Auth outputs "1"), user can update credentials as follows.

- 2F-Credential-Update will be executed to change PW, D_{SC} and $D_{\mathcal{AS}}$.
- Step 3-6 in 3F-User-Reg will be executed to change D_{Bio}. Recall that fuzzy extractor (defined in Section 2.2) can extract different pairs of Random Strings and Auxiliary Strings from the same biometrics.

This completes the description of our protocol.

3.2 Comparison

We compare our proposal with another (and the only known) generic design of multi-factor authentication in [2]. Both protocols employ smart-card-based password authentication and fuzzy extractor as the building block to realize multi-factor authentication, but our design has significant improvements in computation and communication. As shown in Fig. 2, 3F-Login-Auth in [2] runs 2F-Login-Auth twice, but ours is made up of one 2F-Login-Auth and one MAC generation/verification. In the term of communication, the second run of 2F-Login-Auth in [2] (which may incur multi-round message exchanges) is replaced by MAC (only one message exchange) in our protocol. The saving in computation is also obvious: most specific designs of smart-card-based password authentication (such as [10], [11]) involve costly public-key operations (e.g., module exponentiations), while ours only needs lightweight symmetric-key operations.

3.3 Security Analysis

We now show that the newly proposed protocol is a secure three-factor authentication, namely attackers with any two authentication factors will not be able to have a successful authentication.

We first briefly describe the security intuition for our protocol, and the detailed analysis is given in Section 5.2.

There are three types of attackers considered in this paper:

Attackers with Bio and SC. According to the protocol specification, a successful three-factor authentication requires a successful execution of 2F-Login-Auth which requires SC and PW. Thus, such an attacker cannot pass the authentication if 2F-Login-Auth is a secure two-factor authentication based on SC and PW (Theorem 5.1).

Attackers with Bio and PW. Similarly, such an attacker (without SC) cannot pass the authentication if 2F-Login-Auth is a secure two-factor authentication based on SC and PW (Theorem 5.2).

Attackers with SC and PW. Our framework requires a correct MAC value in the authentication. For a well designed MAC function, the calculation of a correct MAC value, i.e., the Tag, requires the MAC key K_B. In our protocol, there are three ways to obtain K_B, none of which will be feasible for attackers with SC and PW (Theorem 5.3).:

1. $Bio + \mathsf{Rep}$: K_B can be retrieved from a correct Bio and the reproduction algorithm Rep. This will not work if one does not have Bio (if Bio is an high-entropy input).
2. $SK_E + D_K$: Alternatively, one can use $\mathcal{AS}$'s private key SK_E to decrypt D_K and obtain K_B. Without SK_E, it will be infeasible to do the decryption.
3. $\mathcal{AS}$ uses a MAC key K'_B chosen by the attacker during the authentication. This however requires a valid hash $H(C \| K'_B \| SK_E)$. The success probability is negligible for an attacker without SK_E (which is $\mathcal{AS}$'s private key).

This completes the security intuition of the proposed protocol. Formal proofs of each claim can be found in Section 5.2.

4 A Generic Approach of Stand-Alone Authentication

Our design in the previous section provides efficient multi-factor authentication in the situation of a slow connection to the central authentication server. This section presents a

solution of user authentication when there is no connection between an access device and the central authentication server. This kind of authentication is called stand-alone authentication (hereinafter referred to as SAA) in this paper.

We first recall a normal authentication between $\mathcal{C}$ and $\mathcal{AS}$

$$\mathcal{C}[Credential] \overset{\text{Login-Auth}}{\Longleftrightarrow} \mathcal{AS}[SK_{\mathcal{AS}}, D_{\mathcal{AS}}] \rightarrow \{1, 0\}.$$

At the server side, there are two kinds of information that may be necessary to authenticate a user: (1) The master secret key $SK_{\mathcal{AS}}$, and (2) $D_{\mathcal{AS}}$ from the local database. It is evident that with $SK_{\mathcal{AS}}$ and $D_{\mathcal{AS}}$, anyone would be able to authenticate $\mathcal{C}$. Our design of SAA follows this idea but has the following differences:

First, the master key $SK_{\mathcal{AS}}$ should be known only by $\mathcal{AS}$, due to its critical role in the whole system. Thus in our design $\mathcal{AS}$ uses a randomly chosen key TK in each user registration, and accordingly TK will be used in authentication.

Second, taking the Smart Grid for example, it would not be practical to store $D_{\mathcal{AS}}$ on each device, due to the large number of users and the limited storage space on intelligent electronic devices. Even in situations where storage is not an issue, maintaining and updating that information will be both time and effort consuming. Thus in our design $D_{\mathcal{AS}}$ is stored on smart-cards and sent to the device during SAA.

Last, it would be certainly desirable if $\mathcal{AS}$ were able to determine which kind of devices in the system can support SAA. In other words, TK and $D_{\mathcal{AS}}$ must be protected such that only the designated devices, i.e., those that indeed need SAA, can obtain them. How to efficiently achieve such a kind of protection is the most challenging task in our design. As explained earlier, an extreme case we are concerned about is a newly joined user that must be authenticated by a device which however has lost the connection with $\mathcal{AS}$ since that user joined the system. The next two sections give an insight on how to achieve such a control in a large-scale information system.

4.1 Hybrid-Encryption

A common approach to share data with a designated party is encryption: One can encrypt the data into a ciphertext such that only the intended party can decrypt and obtain the data. The process can use a secret key shared between two parties (known as symmetric-key encryption) or a public/private key pair of the designated party (known as public key encryption).

From the operational point of view, public-key encryption is a desirable solution: Each device has a public key, a private key and a public key certificate (if needed), and once installed, the private key never leaves the device. It does not require the level of coordination in symmetric-key settings, and the public key and its certificate can be publicly known. But public key operations have a much higher computation requirement, and power-constrained devices cannot afford public-key operations very frequently.

In order to achieve a tradeoff between key management and efficiency, a commonly used approach is using hybrid-encryption: First establish a secret key between communicating parties using public-key encryption, and then all following communications are secured using symmetric-key encryption with the established key. Following this approach, $\mathcal{AS}$ can delegate the authentication right to a device as follows (high-level description):

1) Choose a random AES key key for each registration;
2) Encrypt $(TK, D_{\mathcal{AS}})$ using AES and key: let C_1 be the output;
3) Encrypt key using the device's public key: let C_2 be the output; and
4) Store (C_1, C_2) on the smart card.

The designated device can retrieve key from C_2 (using its private key), and then decrypt C_1 (using key) to obtain $(TK, D_{\mathcal{AS}})$, which enables the device to authenticate the user when the connection to $\mathcal{AS}$ is down.

4.2 Attribute-Based Encryption (ABE)

In an information system as large and dynamic as the Smart Grid, there are a large number of users and devices that need SAA. As an example, due to different roles, Alice can have SAA only with "Substation IEDs in Area A OR Area B" but another user Bob can have stand-alone authentication with "All IEDs in Area A". A naïve way to address this issue is extending the hybrid-encryption into a one-to-many setting.

In the approach described in Section 4.1, one can calculate C_2 as $\{C_2^1, C_2^2, \ldots, C_2^N\}$ where C_2^i is the encryption of key using the public key of the ith eligible device $\mathcal{D}_i$, i.e., devices designated by the central authentication server. However, this naïve approach has two inherent drawbacks: (1) The size of C_2 increases linearly with the number of eligible devices; and (2) Whenever there is a newly equipped device that needs stand-alone authentication, the server must calculate a new C_2^i and store it on each user's smart-card. This apparently is not an easy task in a system with a large number of users. Overcoming these two drawbacks reminds us a recently introduced cryptographical primitive: attribute-based encryption [12].

ABE is a versatile tool for data provider, without prior knowledge of who exactly will be receiving the data, to share data with others in a more flexible way than traditional end-to-end encryption. A typical example of ABE works as follows. Each potential data recipient is associated with an attribute set S and given a private key (generated by a third party) accordingly. The data provider can encrypt data with an embedded predicate function $f(\cdot)$, a description of which kind of recipients can decrypt the ciphertext correctly. Anyone with an attribute set S can successfully decrypt the ciphertext if and only if $f(S) = 1$.

ABE was first introduced in [12] under the name fuzzy identity-based encryption. Later on several variants and improvements have been proposed [13], [14], [15], [16]. Among them, the one fits our situation is ciphertext-policy ABE (CP-ABE) where ciphertexts are associated with access policies and keys are associated with sets of attributes. A CP-ABE consists of four algorithms: ABE.Setup, ABE.Enc, ABE.KeyGen, and ABE.Dec. Table 2 gives the description of each algorithm.

Now we have completed the description of the building blocks required by our SAA, the details of which are given in the next section.

TABLE 2
Description of Ciphertext-Policy ABE

ABE.Setup:	Produce System's Master Key.	ABE.Enc:	Produce Ciphertext.
Input:	κ (security parameter); and U (attribute universe description).	Input:	PK_{ABE}, M (a message), and $\mathbb{A}$ (a predicate).
Output:	PK_{ABE} (public parameters); and MK_{ABE} (master key).	Output:	CT (a ciphertext).
ABE.KeyGen:	Produce Private Key.	ABE.Dec:	Decrypt Ciphertext.
Input:	MK_{ABE} and S (a set of attributes).	Input:	PK_{ABE}, CT, $\mathbb{A}$ and sk.
Output:	sk (a private key).	Output:	A message M if and only if the attribute set S associates with sk satisfies $\mathbb{A}$.

4.3 Adding SAA to Normal Authentication

This section shows how to add SAA to a normal multi-factor authentication protocol. Below are the components used in our design.

Building Blocks:

- A normal multi-factor authentication protocol defined in Section 2.1: {Initial, User-Reg, Login-Auth, Credential-Update};
- An attribute-based encryption scheme: {ABE.Setup, ABE.Enc, ABE.KeyGen, ABE.Dec};
- A digital signature scheme: {PKS.KGen, PKS.Sig, PKS.Ver};
- A symmetric key encryption scheme: {SKE.KGen, SKE.Enc, SKE.Dec}.

Protocol Specification: An authentication protocol with SAA consists of five sub-protocols: {SAA-Initial, SAA-Device-Reg, SAA-User-Reg, SAA-Login-Auth, SAA-Credential-Update}. As one can see, SAA has an extra sub-protocol "SAA-Device-Reg" compared with a normal authentication protocol. It is used to register eligible devices for SAA.

SAA-Initial: Given a security parameter κ, $\mathcal{AS}$

1. Run Initial(κ) to obtain $(PK_{\mathcal{AS}}, SK_{\mathcal{AS}})$.
2. Run PKS.KGen(κ) to obtain (PK_S, SK_S).
3. Run ABE.Setup(κ) to obtain (PK_{ABE}, MK_{ABE}).

The public parameter is $(PK_{\mathcal{AS}}, PK_S, PK_{ABE})$, and the corresponding secret parameter is $(SK_{\mathcal{AS}}, SK_S, MK_{ABE})$.

SAA-Device-Reg: For each new device, $\mathcal{AS}$

1. Assign a set of attributes S to describe the new device, e.g., $S = \{$"IED", "Area A", "Security Level: Medium"$\}$.
2. Run ABE.KeyGen(MK_{ABE}, S) to generate a private key sk_D for the new device.
3. sk_D is stored in the device with tamper-resistance.

SAA-User-Reg: User registration is made up of two phases.

Phase 1. Registration in this phase is for a normal authentication between $\mathcal{C}$ and $\mathcal{AS}$:

$$\mathcal{C}[PW, Bio] \overset{\text{User-Reg}}{\Longleftrightarrow} \mathcal{AS}[SK_{\mathcal{AS}}] \rightarrow \{D_{\mathcal{C}}, D_{\mathcal{AS}}\}.$$

Phase 2. Registration in this phase is for SAA between $\mathcal{C}$ and devices satisfying a predicate $\mathbb{A}$. In this phase, $\mathcal{AS}$

1. Determine what kind of devices can have SAA with $\mathcal{C}$ by choosing a predicate $\mathbb{A}$. As an example, $\mathbb{A}$ = "IED" AND "Area A" implies that only IEDs in Area A are eligible devices.
2. Calculate $TK =$ SKE.KGen(κ) and calculate $\sigma_1 =$ PKS.Sig(TK, SK_S), and $D_1 =$ ABE.Enc$((TK, \sigma_1), \mathbb{A})$.
3. Execute the registration protocol with $\mathcal{C}$:

$$\mathcal{C}[PW',\ Bio] \overset{\text{User-Reg}}{\Longleftrightarrow} \mathcal{AS}[TK] \rightarrow \{D'_{\mathcal{C}}, D'_{\mathcal{AS}}\}.$$

   Note that (1) The password in SAA is different from the one in normal authentication, which will help $\mathcal{C}$ distinguish normal authentication from SAA; and (2) $\mathcal{AS}$ uses TK, instead of $SK_{\mathcal{AS}}$, during the registration.
4. Calculate $\sigma_2 =$ PKS.Sig$(D'_{\mathcal{AS}}, SK_S)$ and $D_2 =$ SKE.$Enc((D'_{\mathcal{AS}}, \sigma_2), TK)$.
5. Let $D_{\mathsf{SAA}} = (D'_{\mathcal{C}}, D_1, D_2, \mathbb{A})$.

At the end of the registration, $\mathcal{C}$ is given a smart-card SC containing $(D_{\mathcal{C}}, D_{\mathsf{SAA}})$.

SAA-Login-Auth: During the authentication, the accessing device first tries to establish a connection with $\mathcal{AS}$ and carry out a normal authentication.

Normal Authentication: The following protocol will be executed if the connection is successfully established:

$$\mathcal{C}[PW, Bio, D_{\mathcal{C}}] \overset{\text{Login-Auth}}{\Longleftrightarrow} \mathcal{AS}[SK_{\mathcal{AS}}, D_{\mathcal{AS}}].$$

The authentication is successful if and only if the protocol outputs "1".

Stand-Alone Authentication: SAA will be executed if the connection with $\mathcal{AS}$ is down:

1. Parse D_{SAA} as $(D'_{\mathcal{C}}, D_1, D_2, \mathbb{A})$ and send $(D_1, D_2, \mathbb{A})$ to the device;
2. Upon receiving $(D_1, D_2, \mathbb{A})$, the device does the following calculations:

$$(TK, \sigma_1) = \text{ABE.Dec}(PK_{ABE}, D_1, sk_D, \mathbb{A}), \text{ and}$$

$$(D'_{\mathcal{AS}}, \sigma_2) = \text{SKE.Dec}(D_2, TK).$$

   Note that only eligible devices, i.e., devices with attributes satisfying $\mathbb{A}$, can successfully retrieve (TK, σ_1) and $(D'_{\mathcal{AS}}, \sigma_2)$.
3. Output "1" if and only if PKS.Ver$(TK, \sigma_1, PK_S) = 1$, and PKS.Ver$(D'_{\mathcal{AS}}, \sigma_2, PK_S) = 1$, and

$$\mathcal{C}[PW', Bio, D'_{\mathcal{C}}] \overset{\text{Login-Auth}}{\Longleftrightarrow} Device[TK, D'_{\mathcal{AS}}] = 1.$$

4. An audit trail is created and stored in the device, regardless of the result of SAA.

SAA-Credential-Update: The followings are carried out between $\mathcal{C}$ and $\mathcal{AS}$ after a successful normal authentication:

Credential-Update for Normal Authentication: The protocol Credential-Update is executed.

Credential-Update for SAA: Phase 2 in SAA-User-Reg will be re-executed and the data D_{SAA} will be updated accordingly.

Performance of SAA: Compared with normal authentication, SAA introduces additional computation cost due to the decryption and the verification of (D_1, D_2). In our design, this is carried out by the accessing device rather than the user. An essential requirement is that accessing devices must be capable of carrying out decryption algorithms in ABE, and the most costly operation in ABE is pairing. The calculation of pairing in practice is not a big issue since there are efficient implementations of pairing on lightweight devices such as mobile sensors and mobile phones [17], [18], [19], [20], and IEDs are generally much more powerful. Actually the computation cost at the user side in our SAA is almost the same as in normal authentication. But adding SAA will introduce additional storage cost at the user side, since the size of D_1 grows proportionally to the complexity of the access policy. We believe such a design satisfies the applications of SAA where storage is not a big issue to personal devices, which are still much less powerful than accessing devices.

Additionally, the proposed SAA is scalable in the sense that users/devices can carry out stand-alone authentication immediately once the registration is complete, and there is no need of system-wide update due to newly joined users/devices. Furthermore, the computation cost and storage cost introduced by SAA is independent of the number of users/devices in the system, which is desirable in dynamic situations such as the Smart Grid.

Security of SAA: We provide detailed analysis in Section 5.3, which shows that the proposed design of SAA is a secure authentication protocol if the underlying building blocks satisfy certain security properties.

5 Security Model and Proofs

This section presents the security model of multi-factor authentication and the proofs of our two newly proposed generic protocols in previous sections.

5.1 Security Model of Multi-Factor Authentication

An attacker, without all authentication factors, has a full control of the communication channel by eavesdropping, injecting, modifying and deleting messages exchanged between $\mathcal{C}$ and $\mathcal{AS}$. During the execution there are many instances of an entity $\mathcal{E} \in \{\mathcal{C}, \mathcal{AS}\}$. As [8], we call instance i of $\mathcal{E}$ an oracle denoted as $\mathcal{O}^i_{\mathcal{E}}$. The aim of the attacker is to impersonate $\mathcal{C}$ and have a successful authentication in a new login request:

For an N-factor authentication protocol, the adversary is given $PK_{\mathcal{AS}}$ (system's public parameter) and $N-1$ authentication factors at the beginning of the game. (An attacker with SC is assumed to have the ability to read and modify the data in the smart-card.)

Queries:

- $EXECUTE(\mathcal{C}, i; \mathcal{AS}, j)$: Assuming that client oracle $\mathcal{O}^i_{\mathcal{C}}$ and server oracle $\mathcal{O}^j_{\mathcal{AS}}$ have not been used, this call carries out an honest execution of the protocol between these oracles, returning a transcript of that execution. These queries are essential for properly dealing with dictionary attacks by eavesdropping.
- $SEND(\mathcal{E}, i, M)$: This sends message M to oracle $\mathcal{O}^i_{\mathcal{E}}$. The oracle computes what the protocol says to and sends back the response. To initiate the protocol with client $\mathcal{C}$ trying to enter into an exchange with $\mathcal{AS}$, the adversary $\mathcal{A}$ should send message $M = \mathcal{AS}$ to an unused instance of $\mathcal{C}$. A Send-query models the real-world possibility of an adversary $\mathcal{A}$ causing an instance to come into existence, for that instance to receive communications fabricated by $\mathcal{A}$, and for that instance to respond in the manner prescribed by the protocol.

The adversary wins the game if $\mathcal{A} \overset{\text{Login-Auth}}{\Longleftrightarrow} \mathcal{AS}$ outputs 1 in a new instance of $\mathcal{AS}$. We say an authentication protocol is (t, ϵ)-secure if no adversaries can win the game with probability at least ϵ in time t.

5.2 Security Analysis of Our New Generic Design

This section proves the security of the new generic design of multi-factor authentication protocol described in Section 3.

Security against Attackers with SC and Bio.

Theorem 5.1. *The proposed three-factor authentication protocol is secure against attackers with smart-card and biometrics, assuming that the underlying two-factor authentication protocol is secure against attackers with smart-card.*

Proof. In the proposed protocol, a successful login (i.e., 3F-Login-Auth protocol outputs "1") requires a successful execution of 2F-Login-Auth: $\mathcal{C}[PW, SC(D_{SC})] \overset{\text{2F-Login-Auth}}{\Longleftrightarrow} \mathcal{AS}[SK_{2F}, D_{\mathcal{AS}}]$. Without the password PW, it is infeasible to login successfully if 2F-Login-Auth is a secure two-factor authentication protocol. We will prove this claim by converting a successful attacker on the proposed three-factor authentication to a successful attacker with smart-card on the underlying two-factor authentication.

Let $\mathcal{A}$ be an attacker with smart-card that tries to have a successful run of 2F-Login-Auth. At the beginning, $\mathcal{A}$ is given the following information:

1. A security parameter κ and a public parameter PK_{2F} of the underlying two-factor authentication
2. A smart-card that contains the data $D_{\mathcal{C}}$.

Suppose there is another attacker $\mathcal{B}$ that, given smart-card and biometrics of a legitimate user, can make a successful login of the proposed three-factor authentication framework. In order to make use of $\mathcal{B}$ to achieve its goal, $\mathcal{A}$ must generate the following data for $\mathcal{B}$.

1. *Public parameter* in three-factor authentication.

$\mathcal{A}$ generates SK_E by running SKE.KGen(κ). $\mathcal{B}$ is given PK_{2F} as the public parameter of the three-factor authentication protocol. SK_E is kept secret by $\mathcal{A}$.

2. *Smart-card and biometrics*.

1) Let Bio be the biometric data and HASH be the hash function, both of which are chosen by $\mathcal{A}$.
2) Let $(\mathsf{Gen}, \mathsf{Rep})$ be the two algorithms in a fuzzy extractor. $\mathcal{A}$ runs the algorithm $\mathsf{Gen}(Bio)$ to generate a pair (R, P). Let $K_B = \mathsf{HASH}(R)$.
3) $\mathcal{A}$ calculates

$$D_K = \mathsf{SKE.Enc}(\mathcal{C}\|K_B\|\mathsf{HASH}(\mathcal{C}\|K_B\|SK_E), SK_E).$$

4) Let $D_{Bio} = (P, D_K, \mathsf{HASH}, \mathsf{Rep})$.
5) At the end of this protocol, $\mathcal{B}$ is given Bio and a smart-card SC, which contains $D_{\mathcal{C}}$ and D_{Bio}.

3. *Queries*.

1) $\mathcal{C}[PW, SC(D_{\mathcal{C}})] \overset{\mathsf{2F\text{-}Login\text{-}Auth}}{\Longleftrightarrow} \mathcal{AS}[SK_{2F}, D_{\mathcal{AS}}]$. $\mathcal{B}$ can make $\mathsf{SEND}(\mathcal{E}, i, M)$ and $\mathsf{EXECUTE}(\mathcal{C}, i; \mathcal{AS}, j)$ queries, but $\mathcal{A}$ is not able to generate all correct responses itself. However, $\mathcal{A}$ can forward $\mathcal{B}$'s queries and obtain correct responses from the user or the server in the underlying two-factor authentication as below:

$$\mathcal{B} \xrightarrow{1.\,Messages} \mathcal{A} \xrightarrow{2.\,Messages} \{\mathcal{C}/\mathcal{AS}\},$$

$$\mathcal{B} \xleftarrow{4.\,Responses} \mathcal{A} \xleftarrow{3.\,Responses} \{\mathcal{C}/\mathcal{AS}\}.$$

2) With K_B and SK_E, $\mathcal{A}$ can either calculate a correct MAC value or verify a MAC value. Thus, it can answers B's queries regarding the MAC function.

This completes the description of how $\mathcal{A}$ can provide $\mathcal{B}$ with the required information.

If $\mathcal{B}$ can make a successful login with the given information, it must have a successful execution of $\mathcal{C}[PW, SC(D_{\mathcal{C}})] \overset{\mathsf{2F-Login-Auth}}{\Longleftrightarrow} \mathcal{AS}[SK_{2F}, D_{\mathcal{AS}}]$, where $\mathcal{B}$ acts as the user $\mathcal{C}$ and $\mathcal{A}$ acts as the server $\mathcal{AS}$. During the execution, $\mathcal{A}$ will act as a relay by passing all messages from $\mathcal{B}$ to the authentication server in the two-factor authentication and vice versa. In this way, $\mathcal{A}$ can collect enough information to make a successful login if $\mathcal{B}$'s login request is successful. This contradicts the assumption that the two-factor authentication is secure against attackers with smart-card. Thus the proposed three-factor authentication protocol is secure against attackers with smart-card and biometrics.

This completes the security analysis of the proposed three-factor authentication framework against attackers with smart-card and biometrics. □

Security against Attackers with PW and Bio.

Theorem 5.2. *The proposed three-factor authentication protocol is secure against attackers with password and biometrics, assuming that the underlying two-factor authentication is secure against attackers with password.*

Proof. We will prove this claim by converting a successful attacker with password and biometrics on the proposed three-factor authentication protocol to a successful active attacker with password on the underlying two-factor authentication.

Let $\mathcal{A}$ be an attacker with password that tries to have a successful run of $\mathsf{2F\text{-}Login\text{-}Auth}$. At the beginning, $\mathcal{A}$ is given κ, PK_{2F} and PW of the underlying two-factor authentication. Suppose there is an attacker $\mathcal{B}$, given password and biometrics, can make a successful login of the proposed framework. In order to make use of $\mathcal{B}$ to achieve its goal, $\mathcal{A}$ needs to generate the following data for $\mathcal{B}$.

1. *Public parameter*. The generation of the public parameter is the same in the proof of Theorem 5.1. $\mathcal{B}$ is given PK_{2F}.

2. *Password*. $\mathcal{B}$ is given PW as the password in the proposed three-factor authentication protocol.

3. *Bio*. The generation of Bio, K_B, and D_{Bio} is the same in the proof of Theorem 5.1. $\mathcal{B}$ is given Bio.

4. *Queries*.

1) $\mathcal{C}[PW, SC(D_{\mathcal{C}})] \overset{\mathsf{2F\text{-}Login\text{-}Auth}}{\Longleftrightarrow} \mathcal{AS}[SK_{2F}]$. Similarly, $\mathcal{B}$ can make $\mathsf{SEND}(\mathcal{E}, i, M)$ and $\mathsf{EXECUTE}(\mathcal{C}, i; \mathcal{AS}, j)$ queries, but $\mathcal{A}$ is not able to generate all correct responses itself (since it does not have $(D_{\mathcal{C}}, SK_{2F})$). However, $\mathcal{A}$ can act what $\mathcal{B}$ does in the execution and obtain correct responses from the client or the server in two-factor authentication (as illustrated in the proof of Theorem 5.1).
2) With K_B and SK_E, $\mathcal{A}$ can either calculate a correct MAC value or verify a MAC value.

This completes the description of how $\mathcal{A}$ can provide $\mathcal{B}$ with the required information.

If $\mathcal{B}$ can make a successful login with the given information, it must have a successful execution of $\mathcal{C}[PW, SC(D_{\mathcal{C}})] \overset{\mathsf{2F\text{-}Login\text{-}Auth}}{\Longleftrightarrow} \mathcal{AS}[SK_{2F}]$, where $\mathcal{B}$ acts as the user $\mathcal{C}$ and $\mathcal{A}$ acts as the server $\mathcal{AS}$. Similarly, $\mathcal{A}$ can collect enough information to make a successful login of $\mathsf{2F\text{-}Login\text{-}Auth}$ with its own authentication server if $\mathcal{B}$'s login request is successful. This contradicts the assumption that the underlying two-factor authentication is secure against attackers with password. This completes the security analysis of the proposed three-factor authentication protocol against attackers with password and biometrics. □

Security against Attackers with SC and PW.

Theorem 5.3. *The proposed three-factor authentication protocol is secure (in the random oracle model) against attackers with smart-card and password, assuming that the underlying two-factor authentication is a secure authentication protocol, the fuzzy extractor satisfies Definition 5, the symmetric key encryption algorithm has ciphertext-indistinguishability [21], and the MAC algorithm is unforgeable [22].*

Proof. Let $\mathcal{A}$ be an attacker on the underlying MAC function. $\mathcal{A}$ is not given the MAC key K^* but can make two types of queries:

Q1: $\mathcal{A}$ can request MAC values of messages of its choices; and

Q2: $\mathcal{A}$ can request the validity of message-MAC pairs of its choices.

$\mathcal{A}$'s goal is to output a valid MAC tag of a new message, i.e., a message never been asked during Q1.

Let $\mathcal{B}$ be an attacker with smart-card and password on the proposed protocol with a success probability ε_B. In order to make use of $\mathcal{B}$ to achieve its goal, $\mathcal{A}$ must generate the following data for $\mathcal{B}$.

1. *Public parameter* in three-factor authentication.

$\mathcal{A}$ generates (PK_{2F}, SK_{2F}) and SK_E by running 3F-Initial of the proposed protocol. $\mathcal{B}$ is given PK_{2F}.

2. *Password and Smart-Card*.

1) $\mathcal{A}$ chooses an initial password PW for a client $\mathcal{C}$.
2) With PW and SK_{2F}, $\mathcal{A}$ can simulate $\mathcal{C}[PW] \overset{\text{2F-User-Reg}}{\Longleftrightarrow} \mathcal{AS}[SK_{2F}]$ and obtain $(D_{\mathcal{C}}, D_{\mathcal{AS}})$.
3) Let Bio be the biometric data chosen by $\mathcal{A}$. $\mathcal{A}$ runs the algorithm $\mathsf{Gen}(Bio)$ (in the fuzzy extractor) to obtain a pair (R, P).
4) A hash function HASH is also chosen by $\mathcal{A}$, and let $K_B = \mathsf{HASH}(R)$.
5) With K_B and SK_E, $\mathcal{A}$ is able to calculate

$$D_K = \mathsf{SKE.Enc}(\mathcal{C}\|K_B\|\mathsf{HASH}(\mathcal{C}\|K_B\|SK_E), SK_E).$$

6) Let $D_{Bio} = (P, D_K, \mathsf{HASH}, \mathsf{Rep})$.
7) $\mathcal{B}$ is given PW and $SC = \{D_{\mathcal{C}}, D_{Bio}\}$.

3. *Queries*:

1. $\mathcal{C}[PW, SC(D_{\mathcal{C}})] \overset{\text{2F-Login-Auth}}{\Longleftrightarrow} \mathcal{AS}[SK_{2F}, D_{\mathcal{AS}}]$.
   $\mathcal{A}$ is able to generate correct responses since it has PW, $(D_{\mathcal{C}}, D_{\mathcal{AS}})$ and SK_{2F}.
2. Now we show how $\mathcal{A}$ deals with MAC operations, which includes two cases.
   Case 1. $\mathcal{A}$ needs to calculate a valid MAC tag of a transcript $\mathcal{T}_i$. To do that, $\mathcal{A}$ sends $\mathcal{T}_i$ to its own challenger as Q1 query and the response would be a valid MAC Tag_i calculated using K^*. $\mathcal{A}$ sends Tag_i to $\mathcal{B}$ as the response.
   Case 2. $\mathcal{A}$ needs to verify the validity of a triple $(D^*, \mathsf{Tag}_i, \mathcal{T}_i)$. If $D^* = D_K$ (Recall that D_K is generated by $\mathcal{A}$ at the registration phase), $\mathcal{A}$ sends its challenger $(\mathsf{Tag}_i, \mathcal{T}_i)$ as a Q2 query and sends the response to $\mathcal{B}$. Otherwise, $D^* \neq D_K$, $\mathcal{A}$ uses SK_E to decrypt D^* and generates the response as described in Step 4 of 3F-Login-Auth.

Note that the simulation given above is slightly different from the real world situation: A real world attacker is given an encryption of a MAC key K^* and a set of MAC values calculated with K^*, but in our simulation, the attacker is given D_K, which is an encryption of a different MAC key K_B. However, such a difference, i.e., given the encryption of K_B instead of K^*, will not have any significant impact on $\mathcal{B}$ in breaking the proposed three-factor authentication protocol if the encryption algorithm satisfies ciphertext indistinguishability [21].

Given an encryption of a message randomly chosen from two equal-length messages m_0 and m_1, the ciphertext indistinguishability requires that no adversary can tell which message is encrypted with success probability significantly more than 1/2. In other words, one cannot distinguish the encryption of m_0 from the encryption of m_1. In our case, this implies that $\mathcal{B}$ cannot tell if D_K is the encryption of K^*. Note that encrypting messages in our protocol are independent of the encrypting key if the hash function is viewed as the random oracle.

Another difference in our simulation is that MAC values are produced/verified based on K^* instead of K_B (the one extracted from biometrics). Recall that the fuzzy extractor extracts a nearly random key from biometrics. Thus, without Bio it is difficult to find the difference, i.e., whether or not the real key extracted from biometrics is used in authentication.

To summarize, the security properties of encryption and fuzzy extractor provide the guarantee that the attacker's view in our simulation is almost the same as an attacker in the real world.

If $\mathcal{B}$ can make a successful login with the given data, it must send $\mathcal{A}$ a valid pair (Tag^*, D_K^*). Let $\mathsf{TRANSCRIPTS}^*$ be the corresponding transcripts. $\mathcal{A}$ outputs $(\mathsf{TRANSCRIPTS}^*, \mathsf{Tag}^*)$ as the forgery of the underlying MAC function. $\mathcal{A}$ successes if all the following events hold true:

1) $\mathcal{A}$'s simulation does not fail. As we have explained, the simulation could only fail if $\mathcal{B}$ can tell the difference between the encryption of K^* and K_B. If the underlying encryption scheme satisfies ciphertext indistinguishability, this only happens with a negligible probability with the security parameter κ, denoted by ϵ_{EK}.
2) $D_K^* = D_K$. If $D_K^* \neq D_K$, the pair (Tag^*, D_K^*) would not be helpful to $\mathcal{A}$: Tag^* is a valid tag under a totally different key. However, this requires that $\mathcal{B}$ must calculate $\mathsf{HashValue} = \mathsf{HASH}(\mathcal{C}\|K'_B\|SK_E)$. Without SK_E, it only occurs with a negligible probability, denoted by ϵ_H, in the random oracle model.
3) The authentication is successful. By assumption, this happens with ε_B. Note that $\mathsf{TRANSCRIPTS}^*$ is not in $\{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_n\}$, since a replay of a previous transcript will not pass 2F-Login-Auth if it is a secure two-factor authentication.

In total, $\mathcal{A}$'s success probability in breaking the underlying MAC function is $\varepsilon_B(1-\epsilon_{EK})(1-\epsilon_H)$, where ϵ_{EK} and ϵ_H are negligible functions of the security parameter κ.

Therefore, with overwhelming probability, $\mathcal{A}$ can output $(\mathsf{Tag}^*, \mathsf{TRANSCRIPTS}^*)$ as a valid new message-MAC pair and break the security of the MAC algorithm. This contradicts the security assumption on MAC. Thus, the proposed protocol is secure against attackers with smart-card and password. □

5.3 SAA Security Analysis

In this section, we prove that the proposed SAA in Section 4.3 is a secure three-factor authentication.

Security against Attackers with SC and Bio.

Theorem 5.4. *The proposed SAA protocol is secure against attackers with smart-card and biometrics, assuming that the underlying authentication protocol is secure against attackers with smart-card and biometrics, the signature scheme is*

existentially unforgeable [5] and the ABE has ciphertext indistinguishability [13].

Proof. Let $\mathcal{A}$ be an attacker given the public parameter $PK_{\mathcal{AS}}$, Bio and SC on a three-factor authentication protocol. Let D_C be the data in SC and $D_{\mathcal{AS}}$ be the data at the server side. $D_{\mathcal{AS}}$ and the corresponding password are unknown to $\mathcal{A}$. $\mathcal{A}$'s aim is to have a successful login.

Suppose an attacker $\mathcal{B}$, given smart-card and biometrics, can make a successful login of SAA in the proposed framework. In order to make use of $\mathcal{B}$, $\mathcal{A}$ needs to first generate the following data for $\mathcal{B}$:

1) Run $\mathsf{PKS.KGen}(\kappa)$ to obtain (PK_S, SK_S).
2) Run $\mathsf{ABE.KGen}(\kappa)$ to obtain (PK_{ABE}, SK_{ABE}).
3) Choose a random predicate $\mathbb{A}$ and let $sk_D = \mathsf{ABE.ENC}(SK_{ABE}, \mathbb{A})$.
4) Calculate $TK = \mathsf{SKE.KGen}(\kappa)$ and calculate $\sigma_1 = \mathsf{PKS.Sig}(TK, SK_S)$, and $D_1 = \mathsf{ABE.Enc}((TK, \sigma_1), \mathbb{A})$.
5) Choose a random data D_R and calculate $\sigma_2 = \mathsf{PKS.Sig}(D_R, SK_S)$, and $D_2 = \mathsf{SKE.Enc}((D_R, \sigma_2), TK)$.
6) Let $D_{\mathsf{SAA}} = (D_{\mathcal{C}}, D_1, D_2, \mathbb{A})$.

The adversary $\mathcal{B}$ is given $(PK_{\mathcal{AS}}, PK_S, PK_{ABE})$, D_{SAA}, and Bio. Here, the generation of D_{SAA} is slightly different from the actual protocol, where σ_2 (resp. D_2) is the signature (resp. encryption) of $D_{\mathcal{AS}}$. In our simulation, σ_2 (resp. D_2) is the signature (resp. encryption) of a randomly chosen data D_R. However, $\mathcal{B}$'s view in our simulated protocol would be computationally indistinguishable from its view in an actual protocol, if the underlying encryption algorithm has ciphertext indistinguishability. In other words, such a difference will only have a negligible impact on $\mathcal{B}$'s success probability on attacking SAA.

$\mathcal{A}$ also needs to simulate the communication between $\mathcal{B}$ and the device. Like the analysis in previous proofs, $\mathcal{A}$ is not able to generate correct responses if $\mathcal{B}$ is actively involved in the execution. However, $\mathcal{A}$ can act what $\mathcal{B}$ does in the execution and obtain responses from its own $\mathcal{AS}/\mathcal{C}$. The responses that $\mathcal{B}$ receives in the simulation would be the same as those in an actual SAA, unless $\mathcal{B}$ can create a pair (D'_1, D'_2) such that $\mathsf{PKS.Ver}(TK', \sigma'_1, PK_S) = 1$, $\mathsf{PKS.Ver}(D'_{\mathcal{AS}}, \sigma'_2, PK_S) = 1$, and $D'_{\mathcal{AS}} \neq D_R$, where

$$(TK', \sigma'_1) = \mathsf{ABE.Dec}(PK_{ABE}, D'_1, sk_D, \mathbb{A}), \text{ and}$$
$$(D'_{\mathcal{AS}}, \sigma'_2) = \mathsf{SKE.Dec}(D_2, TK').$$

In this case, the responses obtained from $\mathcal{AS}$ (which are produced based on the unknown $SK_{\mathcal{AS}}$ and $D_{\mathcal{AS}}$) would not be correct. However, if this happens, $\mathcal{B}$ is able to forge valid signatures of $\mathcal{AS}$ on new messages. Thus, this again will occur only with negligible probability if the underlying signature scheme is existentially unforgeable.

This completes the description of how $\mathcal{A}$ can provide $\mathcal{B}$ with all necessary information.

If $\mathcal{B}$ can make a successful SAA login with the given information, it must have a successful execution of the SAA protocol with $\mathcal{A}$:

1) Let $D'_{\mathsf{SAA}} = (D'_{\mathcal{C}}, D'_1, D'_2)$ be the data $\mathcal{B}$ used in the authentication.
2) Let

$$(TK', \sigma'_1) = \mathsf{ABE.Dec}(PK_{ABE}, D'_1, sk_D, \mathbb{A}), \text{ and}$$
$$(D'_{\mathcal{AS}}, \sigma'_2) = \mathsf{SKE.Dec}(D'_2, TK').$$

3) Then, $\mathsf{PKS.Ver}(TK', \sigma'_1, PK_S) = 1$, $\mathsf{PKS.Ver}(D'_{\mathcal{AS}}, \sigma'_2, PK_S) = 1$, and $\mathcal{C}[PW', Bio, D'_{\mathcal{C}}] \overset{\mathsf{Login\text{-}Auth}}{\Longleftrightarrow} \mathcal{AS}[SK_{\mathcal{AS}}, D_{\mathcal{AS}}] = 1$.

According to the parameter generation, $D'_{\mathcal{AS}}$ must be the same as D_R generated by $\mathcal{A}$ since the signature scheme is existentially unforgeable. In this case, $\mathcal{A}$ can collect enough information to make a successful login of the underlying authentication protocol if $\mathcal{B}$'s login request is successful. This contradicts the assumption that the underlying authentication protocol is secure against attackers with smart-card and biometrics. □

Security against Attackers with PW and Bio

Theorem 5.5. *The proposed SAA protocol is secure against attackers with password and biometrics, assuming that the underlying authentication protocol is secure against attackers with password and biometrics.*

Proof. We will show that a successful attacker with password and biometrics on the proposed SAA can be converted into a successful attacker with password and biometrics on the underlying authentication protocol.

Let $\mathcal{A}$ be an attacker given the public parameter $PK_{\mathcal{AS}}$, Bio and PW on a three-factor authentication protocol. $\mathcal{A}$'s aim is to have a successful login.

Suppose an attacker $\mathcal{B}$, given password and biometrics, can make a successful login of SAA in the proposed framework. In order to make use of $\mathcal{B}$, $\mathcal{A}$

1) Run $\mathsf{PKS.KGen}(\kappa)$ to obtain (PK_S, SK_S).
2) Run $\mathsf{ABE.KGen}(\kappa)$ to obtain (PK_{ABE}, SK_{ABE}).

$\mathcal{B}$ is given $(PK_{\mathcal{AS}}, PK_S, PK_{ABE})$, PW and Bio.

$\mathcal{A}$ also needs to simulate the communication between $\mathcal{B}$ and the device. Like the analysis in previous proofs, $\mathcal{A}$ is not able to generate correct responses if $\mathcal{B}$ is actively involved in the execution. However, $\mathcal{A}$ can act what $\mathcal{B}$ does in the execution and obtain responses from its own $\mathcal{AS}/\mathcal{C}$. This completes the description of how $\mathcal{A}$ can provide $\mathcal{B}$ with all necessary information.

If $\mathcal{B}$ can make a successful SAA login with the given information, it must have a successful execution of the SAA protocol with $\mathcal{A}$. According to the parameter generation and simulation, $\mathcal{A}$ can collect enough information to make a successful login of the underlying authentication protocol, if $\mathcal{B}$'s login request is successful. This contradicts the assumption that the underlying authentication protocol is secure against attackers with password and biometrics. □

Security against Attackers with SC and PW.

Theorem 5.6. *The proposed SAA protocol is secure against attackers with smart-card and password, assuming that the underlying authentication protocol is secure against attackers*

with smart-card and password, the signature scheme is existentially unforgeable [5] and the ABE has ciphertext indistinguishability [13].

Proof. Let $\mathcal{A}$ be an attacker given the public parameter $PK_{\mathcal{AS}}$, SC and PW on a three-factor authentication protocol. Let D_C be the data in SC and $D_{\mathcal{AS}}$ be the data at the server side (which is unknown to $\mathcal{A}$). $\mathcal{A}$'s aim is to have a successful login.

Suppose an attacker $\mathcal{B}$, given smart-card and password, can make a successful login of SAA in the proposed framework. In order to make use of $\mathcal{B}$, $\mathcal{A}$ needs to first generate the following data for $\mathcal{B}$.

1) Run $\mathsf{PKS.KGen}(\kappa)$ to obtain (PK_S, SK_S).
2) Run $\mathsf{ABE.KGen}(\kappa)$ to obtain (PK_{ABE}, SK_{ABE}).
3) Choose a random predicate $\mathbb{A}$.
4) Calculate $TK = \mathsf{SKE.KGen}(\kappa)$, $\sigma_1 = \mathsf{PKS.Sig}(TK, SK_S)$ and $D_1 = \mathsf{ABE.Enc}((TK, \sigma_1), \mathbb{A})$.
5) Choose a random data D_R and calculate $\sigma_2 = \mathsf{PKS.Sig}(D_R, SK_S)$, and $D_2 = \mathsf{SKE.Enc}((D_R, \sigma_2), TK)$.
6) Let $D_{\mathsf{SAA}} = (D_{\mathcal{C}}, D_1, D_2, \mathbb{A})$.

The adversary $\mathcal{B}$ is given $(PK_{\mathcal{AS}}, PK_S, PK_{ABE})$, D_{SAA}, and PW. Here, the generation of D_{SAA} is slightly different from the actual protocol, where σ_2 (resp. D_2) is the signature (resp. encryption) of $D_{\mathcal{AS}}$. In our simulation, σ_2 (resp. D_2) is the signature (resp. encryption) of a randomly chosen data D_R. However, $\mathcal{B}$'s view in our simulated protocol will be computationally indistinguishable from its view in an actual protocol, if the underlying encryption algorithm has ciphertext indistinguishability. In other words, such a difference will only have a negligible impact on $\mathcal{B}$'s success probability on attacking SAA.

$\mathcal{A}$ also needs to simulate the communication between $\mathcal{B}$ and the device. Like the analysis in previous proofs, $\mathcal{A}$ is not able to generate correct responses if $\mathcal{B}$ is actively involved in the execution. However, $\mathcal{A}$ can act what $\mathcal{B}$ does in the execution and obtain responses from its own $\mathcal{AS}/\mathcal{C}$. The responses $\mathcal{B}$ receives in the simulation would be the same as those in an actual SAA, unless $\mathcal{B}$ can create a new pair $(D'_1, D'_2) \neq (D_1, D_2)$ such that $\mathsf{PKS.Ver}(TK', \sigma'_1, PK_S) = 1$, $\mathsf{PKS.Ver}(D'_{\mathcal{AS}}, \sigma'_2, PK_S) = 1$, and $D'_{\mathcal{AS}} \neq D_R$, where $(TK', \sigma'_1) = \mathsf{ABE.Dec}(PK_{ABE}, D'_1, sk_D, \mathbb{A})$ and $(D'_{\mathcal{AS}}, \sigma'_2) = \mathsf{SKE.Dec}(D_2, TK')$. In this case, the responses obtained from $\mathcal{AS}$ (which are produced based on the unknown $SK_{\mathcal{AS}}$ and $D_{\mathcal{AS}}$) would not be correct. However, if this happens, $\mathcal{B}$ is able to forge valid signatures of $\mathcal{AS}$ on new messages. Thus, this again will occur only with negligible probability if the underlying signature scheme is existentially unforgeable.

This completes the description of how $\mathcal{A}$ can provide $\mathcal{B}$ with all necessary information.

If $\mathcal{B}$ can make a successful SAA login with the given information, it must have a successful execution of the SAA protocol with $\mathcal{A}$:

1) Let $D'_{\mathsf{SAA}} = (D'_{\mathcal{C}}, D'_1, D'_2)$ be the data $\mathcal{B}$ used in the authentication.
2) Let $(TK', \sigma'_1) = \mathsf{ABE.Dec}(PK_{ABE}, D'_1, sk_D, \mathbb{A})$ and $(D'_{\mathcal{AS}}, \sigma'_2) = \mathsf{SKE.Dec}(D'_2, TK')$.
3) Then, $\mathsf{PKS.Ver}(TK', \sigma'_1, PK_S) = 1$, $\mathsf{PKS.Ver}(D'_{\mathcal{AS}}, \sigma'_2, PK_S) = 1$, and

$$\mathcal{C}[PW', Bio, D'_{\mathcal{C}}] \overset{\text{Login-Auth}}{\Longleftrightarrow} \mathcal{AS}[SK_{\mathcal{AS}}, D_{\mathcal{AS}}] = 1.$$

According to the parameter generation, $D'_{\mathcal{AS}} = D_R$ since the signature scheme is existentially unforgeable. In this case, $\mathcal{A}$ can collect enough information to make a successful login of the underlying authentication protocol, if $\mathcal{B}$'s login request is successful. This contradicts the assumption that the underlying authentication protocol is secure against attackers with password and smart-card. □

6 Conclusion

In large-scale information systems, communications may be slow or unavailable due to natural disasters or various cyber attacks. This has raised serious concerns in user authentication which usually needs the aid from a remote authentication server. To address this issue, we proposed two solutions of provably secure robust authentication in fragile communication environments. Our first solution is a new generic multi-factor authentication protocol which authenticates users by password, smart-card and biometrics. Our proposal has significant advantages in terms of computation and communication over another generic design in [2]. We believe the new protocol provides a promising authentication solution in slow connection situations. The other contribution of this paper is stand-alone authentication, with which users can be authenticated correctly even the connection to the remote authentication server is down. We gave an efficient and controllable design of stand-alone authentication on any multi-factor authentication protocols.

6.1 Future Work

Stand-alone authentication has two issues worth further investigation.

6.1.1 User Revocation

There are two cases on user revocation, i.e., revocation due to expiration and revocation before expiration. For the first case, $\mathcal{AS}$ can add expiration information on $D'_{\mathcal{AS}}$ which will allow eligible devices to verify whether or not an account has expired. To deal with revocation before expiration, $\mathcal{AS}$ must maintain a revocation list and distribute it among all devices in the system. In this case, devices can obtain the revocation list from $\mathcal{AS}$ whenever there is a connection available. However, there is always a chance that a user has been revoked but the device does not have the latest revocation list due to communication failures. In this case, a revoked user will have a successful SAA with that device. This explains why our SAA uses audit trails: The device will create an audit trail for each SAA request, which will facilitate forensics analysis and find the responsible user. Audit trails alone however cannot prevent a revoked user from having a successful SAA. We leave this as one of the future work directions.

6.1.2 Distributed On-Line Dictionary Attacks

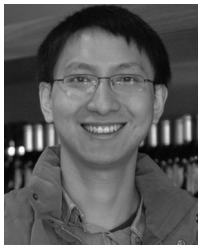
Another future work direction is effective mechanisms to defeat more powerful attacks which are not studied in this paper, one of which is distributed on-line dictionary attacks in SAA. An online-dictionary attacker makes authentication requests by trying every possible password for a specific user. In normal authentication, such attacks can be prevented using lockout mechanisms to lock out the user account after a certain number of invalid login attempts. However, the same approach does not apply to SAA in information systems with a large number of devices: An attacker can run SAA with device D_1 using its guess PW_1, make another login request at device D_2 using another guess PW_2, and so on. This is like amounting on-line dictionary attacks on the same user in a distributed way. A naïve solution requires all devices sharing a common user list with failed login requests, but such a coordination would not be easily achievable in the situations need SAA (i.e., fragile communication environments). In our design of SAA, authenticating users to a device must first try the normal authentication with the authentication server $\mathcal{AS}$, and SAA is invoked only if there is no connection to $\mathcal{AS}$. This will significantly reduce the successful chance of distributed on-line dictionary attacks. Another mitigating method, as shown in our design, is using multiple authentication factors: In case the password has been compromised, the security will still remain in an acceptable level due to other authentication factors.

Acknowledgments

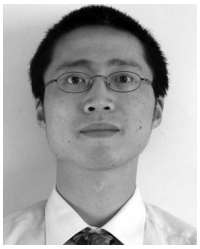
This work is supported by the Energy Market Authority (EMA) of Singapore, through grant SecSG-EPD090005RFP (D) under the Smart Grid Challenge Program, National Natural Science Foundation of China (Grant No. 61202450 and No. 61072080), Ph.D. Programs Foundation of Ministry of Education of China (Grant No. 20123503120001), Distinguished Young Scholars Fund of Department of Education, Fujian Province, China (JA13062), Fujian Normal University Innovative Research Team (No. IRTL1207), and Natural Science Foundation of Fujian Province (No. 2013J01222).

References

[1] NIST, "Guidelines for Smart Grid Cyber Security (INIST 7628)," http://csrc.nist.gov/publications/PubsNISTIRs.html, 2010.
[2] X. Huang, Y. Xiang, A. Chonka, J. Zhou, and R.H. Deng, "A Generic Framework for Three-Factor Authentication: Preserving Security and Privacy in Distributed Systems," *IEEE Trans. Parallel and Distributed Systems*, vol. 22, no. 8, pp. 1390-1397, Aug. 2011.
[3] U. Uludag, S. Pankanti, S. Prabhakar, and A.K. Jain, "Biometric Cryptosystems: Issues and Challenges," *Proc. IEEE*, vol. 92, no. 6, pp. 948-960, June 2004.
[4] A. Bhargav-Spantzel, A.C. Squicciarini, S.K. Modi, M. Young, E. Bertino, and S.J. Elliott, "Privacy Preserving Multi-Factor Authentication with Biometrics," *J. Computer Security*, vol. 15, no. 5, pp. 529-560, 2007.
[5] S. Goldwasser, S. Micali, and C. Rackoff, "The Knowledge Complexity of Interactive Proof Systems," *SIAM J. Computing*, vol. 18, no. 1, pp. 186-208, 1989.
[6] C.I. Fan and Y.H. Lin, "Provably Secure Remote Truly Three-Factor Authentication Scheme with Privacy Protection on Biometrics," *IEEE Trans. Information Forensics and Security*, vol. 4, no. 4, pp. 933-945, Dec. 2009.
[7] C.T. Li and M.S. Hwang, "An Efficient Biometrics-Based Remote User Authentication Scheme Using Smart Cards," *J. Network and Computer Applications*, vol. 33, no. 1, pp. 1-5, 2010.
[8] M. Bellare, D. Pointcheval, and P. Rogaway, "Authenticated Key Exchange Secure Against Dictionary Attacks," *Proc. 19th Int'l Conf. Theory and Application of Cryptographic Techniques (EUROCRYPT '00)*, vol. 1807, pp. 139-155, 2000.
[9] Y. Dodis, L. Reyzin, and A. Smith, "Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data," *Proc. Advances in Cryptology (EUROCRYPT '04)*, vol. 3027, pp. 523-540, 2004.
[10] G. Yang, D.S. Wong, H. Wang, and X. Deng, "Two-Factor Mutual Authentication Based on Smart Cards and Passwords," *J. Computer and System Sciences*, vol. 74, no. 7, pp. 1160-1172, 2008.
[11] J. Xu, W.T. Zhu, and D. Feng, "An Improved Smart Card Based Password Authentication Scheme with Provable Security," *Computer Standards & Interfaces*, vol. 31, no. 4, pp. 723-728, 2009.
[12] A. Sahai and B. Waters, "Fuzzy Identity-Based Encryption," *Proc. 24th Int'l Conf. Theory and Application of Cryptographic Techniques (EUROCRYPT '05)*, vol. 3494, pp. 457-473, 2005.
[13] A.B. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters, "Fully Secure Functional Encryption: Attribute-Based Encryption and (Hierarchical) Inner Product Encryption," *Proc. 29th Int'l Conf. Theory and Application of Cryptographic Techniques (EUROCRYPT '10)*, vol. 6110, pp. 62-91, 2010.
[14] A.B. Lewko and B. Waters, "Unbounded HIBE and Attribute-Based Encryption," *Proc. Advances in Cryptology (EUROCRYPT '11)*, vol. 6632, pp. 547-567, 2011.
[15] B. Waters, "Ciphertext-Policy Attribute-Based Encryption: An Expressive, Efficient, and Provably Secure Realization," *Proc. Public Key Cryptography (PKC '11)*, vol. 6571, pp. 53-70, 2011.
[16] R. Ostrovsky, A. Sahai, and B. Waters, "Attribute-Based Encryption with Non-Monotonic Access Structures," *Proc. 14th ACM Conf. Computer and Comm. Security*, pp. 195-203, 2007.
[17] T. Ishiguro, M. Shirase, T. Takagi, "Efficient Implementation of Pairing on Sensor Nodes," http://csrc.nist.gov/groups/ST/IBE/documents/June08/Takagi.pdf 2008.
[18] X. Xiong, D.S. Wong, and X. Deng, "Tinypairing: A Fast and Lightweight Pairing-Based Cryptographic Library for Wireless Sensor Networks," *Proc. IEEE Wireless Comm. and Networking Conf. (WCNC)*, pp. 1-6, 2010.
[19] A.D. Caro, "JPBC library: Benchmark," http://gas.dia.unisa.it/projects/jpbc/benchmark.html, 2014.
[20] T. Iyama, S. Kiyomoto, K. Fukushima, T. Tanaka, and T. Takagi, "Efficient Implementation of Pairing on Brew Mobile Phones," *Proc. Fifth Int'l Conf. Advances in Information and Computer Security (IWSEC '10)*, vol. 6434, pp. 326-336, 2010.
[21] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway, "A Concrete Security Treatment of Symmetric Encryption," *Proc. 38th Ann. Symp. Foundations of Computer Science (FOCS '97)*, pp. 394-403, 1997.
[22] M. Bellare, R. Guérin, and P. Rogaway, "XOR MACs: New Methods for Message Authentication Using Finite Pseudorandom Functions," *Proc. Advances in Cryptology (CRYPTO '95)*, vol. 963, pp. 15-28, 1995.



Xinyi Huang received the PhD degree from the School of Computer Science and Software Engineering, University of Wollongong, Australia, in 2009. He is currently a professor at the Fujian Provincial Key Laboratory of Network Security and Cryptology, School of Mathematics and Computer Science, Fujian Normal University, China. His research interests include cryptography and information security. He has published more than 60 research papers in refereed international conferences and journals. His work has been cited more than 1,000 times at Google Scholar. He is in the Editorial Board of International Journal of Information Security (IJIS, Springer) and has served as the program/general chair or program committee member in more 40 international conferences.



Yang Xiang received the PhD degree in computer science from Deakin University, Australia. He is currently a full professor at School of Information Technology, Deakin University. He is the director of the Network Security and Computing Lab (NSCLab). His research interests include network and system security, distributed systems, and networking. In particular, he is currently leading his team developing active defense systems against large-scale distributed network attacks. He is the chief investigator of several projects in network and system security, funded by the Australian Research Council (ARC). He has published more than 130 research papers in many international journals and conferences, such as *IEEE Transactions on Computers, IEEE Transactions on Parallel and Distributed Systems, IEEE Transactions on Information Security and Forensics*, and *IEEE Journal on Selected Areas in Communications*. Two of his papers were selected as the featured articles in the April 2009 and the July 2013 issues of *IEEE Transactions on Parallel and Distributed Systems*. He has published two books, Software Similarity and Classification (Springer) and Dynamic and Advanced Data Mining for Progressing Technological Development (IGI-Global). He has served as the program/general chair for many international conferences such as ICA3PP 12/11, IEEE/IFIP EUC 11, IEEE TrustCom 13/11, IEEE HPCC 10/09, IEEE ICPADS 08, NSS 11/10/09/08/07. He has been the PC member for more than 60 international conferences in distributed systems, networking, and security. He serves as the associate editor of *IEEE Transactions on Computers, IEEE Transactions on Parallel and Distributed Systems, Security and Communication Networks* (Wiley), and the editor of *Journal of Network and Computer Applications*. He is the coordinator, Asia for *IEEE Computer Society Technical Committee on Distributed Processing* (TCDP). He is a senior member of the IEEE.

Elisa Bertino is a professor of computer science at Purdue University, and director of the Purdue Cyber Center (Discovery Park). She also serves as research director of the Center for Information and Research in Information Assurance and Security (CERIAS). Prior to joining Purdue, she was a professor and department head at the Department of Computer Science and Communication of the University of Milan. She has been a visiting researcher at the IBM Research Laboratory (now Almaden) in San Jose, at the Microelectronics and Computer Technology Corporation, at Rutgers University, at Telcordia Technologies. Her recent research focuses on database security, digital identity management, policy systems, and security for web services. She is a fellow of the ACM and the IEEE. She received the IEEE Computer Society 2002 Technical Achievement Award and the IEEE Computer Society 2005 Kanai Award. She a member of the editorial board of *IEEE Transactions on Dependable and Secure Computing*, and *IEEE Security & Privacy*. She served as chair of the ACM Special Interest Group on Security, Audit and Control (ACM SIGSAC).

Jianying Zhou received the PhD degree in information security from University of London. He is a senior scientist at Institute for Infocomm Research, and heads the Infocomm Security Department. His research interests are in computer and network security, mobile and wireless communications security. He was an adjunct professor in Kyushu University, Shanghai Jiaotong University and University of Science and Technology of China. He is a founder and steering committee member of International Conference on Applied Cryptography and Network Security (ACNS).

Li Xu received the BS and MS degrees from Fujian Normal University in 1992 and 2001 and the PhD degree from Nanjing University of Posts and Telecommunications in 2004. He is a professor and doctoral supervisor at the School of Mathematics and Computer Science at Fujian Normal University. He is currently the vice dean of the School of Mathematics and Computer Science and the director of the Key Lab of Network Security and Cryptography in Fujian Province. His interests include wireless networks and communication, network and information security, complex networks and systems, intelligent information in communication networks, etc. He has been invited to act as PC chair or member at more than 30 international conferences. He is a member of the IEEE and ACM, and a senior member of the CCF and CIE in China. He has published more than 100 papers in refereed journals and conferences.