2010

Constructing an authentication token to access external services in service aggregation

Peishun Wang
University of Wollongong, peishun@uow.edu.au

Yi Mu
University of Wollongong, ymu@uow.edu.au

Willy Susilo
University of Wollongong, wsusilo@uow.edu.au

Jun Yan
University of Wollongong, jyan@uow.edu.au

Constructing an authentication token to access external services in service aggregation

Abstract

Service aggregation is becoming a cost-effective and time-efficient way for a business to develop new applications and services. While it creates tremendous opportunities in various industry sectors, its cross-organization nature raises serious challenges in the security domains for authentication. In this paper we formulate a formal definition of authentication in service aggregation and a security model for it, and propose two authentication protocols. One is a one-way protocol and another is an interactive one. In particular, the constructed authentication tokens are anonymous to verifiers. We prove their security, show how to choose optimal system parameters, and analyse the efficiency.

Publication Details

Wang, P., Mu, Y., Susilo, W. & Yan, J. (2010). Constructing an authentication token to access external services in service aggregation. 2010 IEEE 7th International Conference on Services Computing, SCC 2010 (pp. 321-328). Piscataway, New Jersey, USA: IEEE.

2010 IEEE International Conference on Services Computing

Constructing an Authentication Token to Access External Services in Service Aggregation

Peishun Wang*, Yi Mu*, Willy Susilo* and Jun Yan†
*School of Computer Science and Software Engineering
†School of Information Systems and Technology
University of Wollongong
Sydney, NSW 2522, Australia
Email: peishun,ymu,wsusilo,jyan@uow.edu.au

***Abstract*—Service aggregation is becoming a cost-effective and time-efficient way for a business to develop new applications and services. While it creates tremendous opportunities in various industry sectors, its cross-organization nature raises serious challenges in the security domains for authentication. In this paper we formulate a formal definition of authentication in service aggregation and a security model for it, and propose two authentication protocols. One is a one-way protocol and another is an interactive one. In particular, the constructed authentication tokens are anonymous to verifiers. We prove their security, show how to choose optimal system parameters, and analyse the efficiency.**

***Keywords*-service aggregation; authentication; token.**

I. Introduction

Instead of selling its own services only, an organization is increasingly aggregating various services offered by other service providers in different domains with some of its own services to form more complex and powerful services to serve consumers. As a cost-effective and time-efficient way to develop new applications and services, service aggregation can be seen as the process of combining a set of typically similar or complementing services to achieve a common goal [11]. It supports on-the-fly discovery, aggregation, deployment and provision of services, and realizes the on-demand interaction among consumers, service aggregators and service providers. Therefore, it empowers service providers and consumers, and creates tremendous opportunities in various industry sectors. On the other hand, elementary services involved in service aggregation are distributed in different organizations all over the world. They are exposed to external attacks from the Internet, not to speak about internal attacks against internal services. Even though the security assurance of each elementary service, which is located in an independent security domain, is given, new security challenges still arise. For example, the authentication requirement defined for an organisation cannot be applied to another organisation. It is obvious that for a password-based authentication system, a user password cannot be applied to a different organisation. Similarly, a credential for a consumer might not be applicable to multiple organisations.

As a critical security issue, authentication is always one of the main concerns of organizations. Formally, authentication is the process of determining whether someone or something is, in fact, who or what it is declared to be. Without authentication, any user can access the services. In some extreme situations, an individual could pose as a willing user, accept the services, and then repudiate the transaction. In the past few years, there have been rapid development in the field of authentication, and a lot of authentication protocols were proposed [1], [7], [10], [12], [13], [16], [18]. According to implementation technologies, authentication protocols are categorized into different types, such as one-way protocols in which all communication is one way, and interactive protocols that work for both pairwise interaction and group formation. There have been several survey papers on authentication [14], [15], which gave comparative analyses. We observe that there is no existing authentication protocol in the literature that could be applied in service aggregation efficiently, and this point is discussed in Section VII. Motivated by it, in this paper, we present a formal definition of authentication in service aggregation and a security model for it, and propose a one-way authentication protocol, which is from Bloom filter. Then we improve it to generate an interactive authentication protocol by applying ElGamal cryptosystem. We prove their security, and give a way to optimize the system parameters. To our knowledge, this is the first work on authentication in service aggregation, and there is no existing protocol to compare. So we analyse their efficiency by comparing with possible solutions and show that the proposed protocols are the most efficient. In particular, the proposed protocols use the binary check to test if an authentication token is valid, which speeds up verification. Additionally, the constructed authentication tokens have a property of anonymity, $i.e.$, every service provider can verify the validity of the authentication token, but no one can know whom the token belongs to.

The remainder of the paper is organised as follows. Section II describes the architecture of service aggregation and the requirement of authentication. In Section III the models

978-0-7695-4126-6/10 $26.00 © 2010 IEEE
DOI 10.1109/SCC.2010.30

of authentication and security are demonstrated. Section IV and Section V present a one-way and an interactive authentication protocols, respectively, and show their security. Section VI gives a way to choose optimal system parameters. In Section VII the efficiency is discussed. Finally, Section VIII concludes this paper and outlines our future work.

II. Service Aggregation

There are three parties in service aggregation, users, service providers and a service aggregator. They collaborate in highly distributed environments and establish on-demand, short-term and dynamic business relationships for purposes such as maximizing profitability [17]. In order to understand the requirement of authentication in service aggregation, we first explain some terms as follows.

User :
: it is a fancy name for an application trying to access services, which are offered by service providers and the service aggregator. It can be a website, a desktop program, a mobile device, or anything else connected to the Internet.

Service Provider :
: it provides various public and professional services ranging from simple information retrieval services to more complex transaction oriented services. It is the term used to describe the website or web service where professional services are located. It can be an airline corporation, a picture shop, an online bank, an express delivery company, or any others where users could get their desired services.

Service Aggregator :
: it works as an agency to combine various services offered by service providers along with some of its own services and resell the aggregated services to users. It could be a travel agency, an e-Health service center, etc.

Authentication Token :
: it is used instead of user's credential to access services offered by service providers. A Token is generally a random string of letters and numbers (but not limited to) that is unique, hard to guess, and verifiable. Sometime it is paired with a secret to protect the token from being abused.

A general diagram that illustrates a simple architecture of Service Aggregation in a high level is shown in Figure 1.

In order to provide aggregated services to users, service providers register with the service aggregator and supply the links of their services to the service aggregator. Similarly, in order to access aggregated services, every user needs to register with the service aggregator and becomes its member. Usually different members of the service aggregator have different rights to access different services that the service aggregator provides. For example, golden members could access more services than normal members. Notice that users do not need to register themselves with service providers. When a user would like to access some services, she sends a service request to the service aggregator. The service aggregator searches for all services associated with the request and displays public services that are free to the user and protected services that need the user to sign on. To obtain the protected services, the user has to authenticate her to the service aggregator. If the protected services come from some external service providers ($i.e.$, the services are external), even though the user is not a member of these service providers, she, as a member of the service aggregator, can still access these external services. However, the user has to authenticate her to these service providers. To do so, the service aggregator will construct an authentication token for the user. The user shows the authentication token to these service providers, who can verify the validity of the authentication token. Because the user is just a member of the service aggregator, considering her privacy, she does not want other parties, including the service providers, to know her identity. This means, the authentication token must allow the user to remain anonymous to the verifiers. If the token is correct, the service providers give permission to the user for accessing the external services.

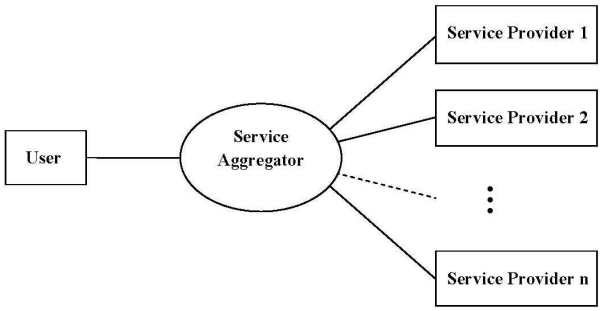

Figure 1. Architecture of Service Aggregation

III. Models

Throughout this paper, we use the following notation.

Let $a \xleftarrow{R} A$ denote choosing an element a uniformly at random from the set A. The set of natural numbers is denoted by $\mathbf{N}$. For an integer n, $[n]$ denotes the set of integers $\{1, 2, \ldots, n\}$. We write $x||y$ to denote the concatenation of the bit-strings x, y. By default, let e be the base of the natural logarithm, $\lg k \stackrel{def}{=} \log_2 k$, and $\ln k \stackrel{def}{=} \log_e k$ for a number k.

A. Authentication in Service Aggregation

We give a formal definition of Authentication in Service Aggregation (AuthSA) as follows.

Definition III.1 *An* **AuthSA** *protocol consists of two parties as follows.*

Setup *: The service aggregator chooses a security parameter s and generates a system parameter $PARA$ that includes a public key PUK and a private key PRK. Every user and service provider register with the service aggregator by supplying their identities ID_u and ID_{sp} and corresponding credentials C_u and C_{sp}, respectively.*

Sign-On *: A user logs on the service aggregator to obtain an authentication token and then uses the authentication token to access her accessible service providers. It consists of the following three algorithms.*

Logon(ID_u, C_u) :
The algorithm takes as input user's identity ID_u and credential C_u, and outputs 1 if the user is legitimate, or 0 otherwise.

BuildAuthToken($PARA, ID_u, C_u, \{ID_{sp_i}, C_{sp_i}\}_{i=1,\cdots,m}$) :
The algorithm takes as input the system parameter $PARA$, user's identity ID_u and credential C_u, and service providers' identities $\{ID_{sp_i}\}_{i=1,\cdots,m}$ and credentials $\{C_{sp_i}\}_{i=1,\cdots,m}$ ($m \in \mathbf{N}$), and outputs an authentication token AT for the user identified by ID_u.

Verify($PUK, AT, ID_{sp_i}, C_{sp_i}$) :
The algorithm takes as its input the public key PUK, an authentication token AT, a service provider's identity ID_{sp_i} and corresponding credential C_{sp_i}, and outputs 1 if the user are authenticated to access the services of the service provider identified by ID_{sp_i}, or 0 otherwise.

B. Security Model

Now we introduce a security model called **Semantic Security against Adaptive Chosen Identity Attack (IND-CIA)**. Intuitively, it aims to capture the notion that an adversary $\mathcal{A}$ cannot deduce a user's identity from her authentication tokens. Roughly speaking, the game works as follows. Suppose the challenger $\mathcal{C}$ gives the adversary $\mathcal{A}$ two identities ID_0 and ID_1, together with an access token. Here, $\mathcal{A}$'s challenge is to determine which identity is the owner of the access token. If the problem of distinguishing between the access token for ID_0 and ID_1 is hard, then deducing the identity from the access token must also be hard. If $\mathcal{A}$ cannot determine which identity is the owner of the access token with probability non-negligibly different from 1/2, then the access token reveals nothing about the identity. We use this formulation of access token indistinguishability to prove the semantic security of access tokens

In particular, an adversary could be a user, a coalition of users, a service provider, a coalition of service providers, or a coalition of users and service providers. We use the following game between a challenger $\mathcal{C}$ and an attacker $\mathcal{A}$ to define IND-CIA from a coalition of users and service providers.

Definition III.2 IND-CIA *is a security game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ as follows. To describe conveniently, we omit the parts concerning passwords.*

Setup *: $\mathcal{A}$ adaptively selects a set of user identities $\{ID_{u_i}\}_{i=1,\cdots,n}$ ($n \in \mathbf{N}$) and a set of service provider identities $\{ID_{sp_j}\}_{j=1,\cdots,m}$ ($m \in \mathbf{N}$).*

Query *: $\mathcal{A}$ may query $\mathcal{C}$ for the authentication token AT for a user identified by $ID_{u'}$ ($u' \in \{u_i\}_{i=1,\cdots,n}$) and a subset of service providers identified by $\{ID_{sp_{j_i}}\}_{i=1,\cdots,m'}$ (i.e., $\{ID_{sp_{j_i}}\}_{i=1,\cdots,m'} \subseteq \{ID_{sp_j}\}_{j=1,\cdots,m}$). On receiving the identities of the user and the service providers, $\mathcal{C}$ runs the algorithm* **BuildAuthToken** *to generate the authentication token to $\mathcal{A}$, who can invoke the algorithm* **Verify** *to check if the user $ID_{u'}$ is permitted to access the services of the set of service providers $\{ID_{sp_{j_i}}\}_{i=1,\cdots,m'}$.*

Challenge *: After making a polynomial number of queries, $\mathcal{A}$ decides on challenge by picking two users identified by $ID_{u'_0}$ and $ID_{u'_1}$ and a subset of service providers identified by $\{ID_{sp_{j_i}}\}_{i=1,\cdots,m''}$ ($\{j_i\}_{i=1,\cdots,m''} \subseteq [m]$) and sending them to $\mathcal{C}$. Then $\mathcal{C}$ chooses $b \xleftarrow{R} \{0,1\}$, runs the algorithm* **BuildAuthToken** *with the identity $ID_{u'_b}$ and the set of service providers' identities $\{ID_{sp_{j_i}}\}_{i=1,\cdots,m''}$ to generate the authentication token AT_b, and returns it to $\mathcal{A}$. After the challenge of determining b for $\mathcal{A}$ is issued, $\mathcal{A}$ is allowed again to query $\mathcal{C}$ another polynomial number times.*

Response *: Finally $\mathcal{A}$ outputs a bit $b_{\mathcal{A}}$, and is successful if $b_{\mathcal{A}} = b$. The advantage of $\mathcal{A}$ in winning this game is defined as $Adv_{\mathcal{A}} = |Pr[b = b_{\mathcal{A}}] - 1/2|$, and the adversary is said to have an ϵ-advantage if $Adv_{\mathcal{A}} \geq \epsilon$.*

IV. One-way AuthSA Protocol

In the literature, there are many protocols for making a secure authentication between two parties – a user and a service aggregator. This is not the main concern of this paper. We just use the simplest method – identity and password – to make such an authentication. Our concentration is on the

construction of the authentication token. In this section, we use Bloom filter to construct a one-way AuthSA protocol and prove its security according to the game IND-CIA.

A. Background

A Bloom Filter [3] can efficiently store information about the existence of a record in a database. Although it yields false positives, the probability of a false positive can be made as small as requested.

A Bloom filter represents a set of $Y = \{y_1, y_2, \ldots, y_t\}$ of t elements by an array BF of m bits. All array bits are initially set to 0. The filter uses s independent hash functions $H_1, \ldots, H_s$, where $H_j : \{0,1\}^* \rightarrow [m]$ for $j \in [s]$, *i.e.*, it requires a set of s independent hash functions that produce uniformly distributed output in $[m]$ over all possible inputs. For each element y_j $(j = 1, \ldots, t)$, the array bits at positions $H_1(y_j), \ldots, H_s(y_j)$ are set to 1. A location can be set to 1 multiple times, but only the first is noted.

To add an entry y to the filter, compute $p_1 = H_1(y), p_2 = H_2(y), \ldots, p_s = H_s(y)$, and set $BF[p_j] = 1 \; for \; j = 1, \ldots, s$.

To check whether or not a value y is in the database, the same positions p_j are calculated and bits $BF[p_j]$ are examined. If all the checked bits are 1, the record y is considered a member of the set. There is, however, some probability of a false positive, in which y appears to be in Y but actually is not. False positives occur because each location may have also been set by some element other than y. On the other hand, if any checked bits are 0, then y is definitely not a member of Y.

B. Construction

Setup :

1) The service aggregator chooses a security parameter s, $k_i \xleftarrow{R} \{0,1\}^s$, $i = 0, 1, \cdots, r$, $r \in \mathbf{N}$, and a pseudo-random function $H : \{0,1\}^* \times \{0,1\}^s \longrightarrow \{0,1\}^s$ as the system parameter $PARA = \{s, k_0, \cdots, k_r, H\}$. Then the service aggregator publishes the system public keys $PUK = \{H, k_1, \cdots, k_r\}$ and keeps the private key $PRK = \{k_0\}$.
2) A user u registers with the service aggregator by supplying her identity and password (as credential) (ID_u, PW_u).
3) A service provider sp registers with the service aggregator by supplying its identity and password (as credential) (ID_{sp}, PW_{sp}).

Sign-On :

Logon(ID_u, C_u) :
User u logs on the service aggregator with her identity and password (ID_u, PW_u), and the service aggregator verifies the user's identity and password with the data in its credential database. If the identity and password are correct, it outputs 1; Otherwise, 0.

BuildAuthToken($PARA, ID_u, C_u, \{ID_{sp_i}, C_{sp_i}\}_{i=1,\cdots,m}$) :
For a legitimate user (*i.e.*, the output of the algorithm **Logon**(ID_u, C_u) is 1), the service aggregator runs this algorithm as follows.

1) Finds all service providers sp_j $(j = 1, \cdots, n)$ accessible for the user u, where $n \in \mathbf{N}$.
2) Creates a timestamp ts.
3) Computes $M = H(ts||ID_u||PW_u, k_0)$.
4) Computes $p_{i,0} = H(M||k_i, k_0)$ for $i \in [r]$.
5) Computes $p_{i,j} = H(ts||M||ID_{sp_j}||PW_{sp_j}, k_i)$ for $i \in [r]$ and $j \in [n]$.
6) Generates a Bloom filter BF by setting $BF[p_{i,j}] = 1$ $(i = 1, \cdots, r$ and $j = 0, 1, \ldots, n)$.
7) Outputs an authentication token $AT = \{ts, M, BF\}$ for the user u.

Verify($PUK, AT, ID_{sp_i}, C_{sp_i}$) :
In order to access services provided by a service provider sp_i $(i \in [m])$, the user shows the authentication token $AT = \{ts, M, BF\}$ to the service provider, who runs this algorithm as follows.

1) Computes $p_i = H(ts||M||ID_{sp}||PW_{sp}, k_i)$ for $i \in [r]$.
2) Tests if BF contains 1's in all r locations denoted by $p_1, \cdots, p_r$.
3) If yes, outputs 1; Otherwise, 0.

C. Security

Theorem 1 *The proposed one-way AuthSA protocol is semantically secure under the security game IND-CIA.*

Proof: Suppose the proposed protocol is not semantically secure against chosen keyword-attacks, *i.e.*, there exists an adversary $\mathcal{A}$ who has an ϵ-advantage to win the **IND-CIA** game, even though $\mathcal{A}$ does not have the value of the private key k_0. Then we build an algorithm $\mathcal{A}'$ that uses $\mathcal{A}$ as a subroutine to determine with an ϵ-advantage if H is a pseudo-random function or a random function. Let $\mathcal{A}'$ query an oracle $\mathcal{O}_H$ for the unknown function $H : \{0,1\}^* \rightarrow \{0,1\}^s$. When running the algorithm **BuildAuthToken**, $\mathcal{A}'$ substitutes evaluations of H with queries to the oracle $\mathcal{O}_H$.

For example, when $\mathcal{A}'$ needs to compute the value of M, $H(ts||ID_u||PW_u, k_0)$, it queries $\mathcal{O}_H$ for the value with input $ts||ID_u||PW_u||k_0$. $\mathcal{A}'$ uses $\mathcal{A}$ in **IND-CIA** game as follows:

Setup : $\mathcal{A}$ adaptively selects a set of user identities $\{ID_{u_i}\}_{i=1,\cdots,n}$ ($n \in \mathbf{N}$) and a set of service provider identities $\{ID_{sp_j}\}_{j=1,\cdots,m}$ ($m \in \mathbf{N}$).

Query : $\mathcal{A}$ may query $\mathcal{A}'$ for the authentication token AT for a user identified by $ID_{u'}$ ($u' \in \{u_i\}_{i=1,\cdots,n}$) and a subset of service providers identified by $\{ID_{sp_{j_i}}\}_{i=1,\cdots,m'}$ (*i.e.*, $\{ID_{sp_{j_i}}\}_{i=1,\cdots,m'} \subseteq \{ID_{sp_j}\}_{j=1,\cdots,m}$). On receiving the identities of the user and the service providers, $\mathcal{A}'$ runs the algorithm **BuildAuthToken** to generate the authentication token and sends it to $\mathcal{A}$.

Challenge: After making a polynomial number of queries, $\mathcal{A}$ decides on challenge by picking two users identified by $ID_{u'_0}$ and $ID_{u'_1}$ and a subset of service providers identified by $\{ID_{sp_{j_i}}\}_{i=1,\cdots,m''}$ ($\{j_i\}_{i=1,\cdots,m''} \subseteq [m]$) and sending them to $\mathcal{A}'$. Then $\mathcal{A}'$ chooses $b \xleftarrow{R} \{0,1\}$, runs the algorithm **BuildAuthToken** with the identity $ID_{u'_b}$ and the set of service providers' identities $\{ID_{sp_{j_i}}\}_{i=1,\cdots,m''}$ to generate the authentication token AT_b, and returns it to $\mathcal{A}$. After the challenge of determining b for $\mathcal{A}$ is issued, $\mathcal{A}$ is allowed again to query $\mathcal{A}'$ another polynomial number times.

Response : Finally $\mathcal{A}$ outputs a bit $b_{\mathcal{A}}$, which represents its guess for b. If $b_{\mathcal{A}} = b$, then $\mathcal{A}'$ outputs 1, indicating that it guesses that H is a pseudo-random function. Otherwise, $\mathcal{A}'$ outputs 0.

When H is a pseudo-random function. Because $\mathcal{A}$ has an ϵ-advantage to win the **IND-CIA** game, and $\mathcal{A}'$ simulates the challenger $\mathcal{C}$ perfectly in the **IND-CIA** game. We have

$$|Pr[\mathcal{A}'_{H(\cdot,k)} = 1 | k \xleftarrow{R} \{0,1\}^s] - \frac{1}{2}| \geq \epsilon. \quad (1)$$

When H is a random function. The hash value is treated as a random value, it follows that the values of M, $p_{i,0}$ and $p_{i,j}$ ($i \in [r]$, $j \in [n]$) can be viewed as random values. Because the timestamp ts is never repeated, for any two queries, no matter whether the user identities are different or identical, the values of M that $\mathcal{A}'$ queries $\mathcal{O}_H$ twice with the input $ts||ID_u||PW_u||k_0$, as two random values, are independent to each other. In the same way, the values of $p_{i,0}$ and $p_{i,j}$ ($i \in [r]$, $j \in [n]$) are also independent to each other. This means, the authentication token reveals nothing about the user identity. Therefore, for the challenge, it is infeasible for $\mathcal{A}$ to correlate the authentication token with the user identity. Based on the above analysis, $\mathcal{A}$ at best guesses b correctly with probability 1/2. Thus we have

$$Pr[\mathcal{A}'_H = 1 | H \xleftarrow{R} \{\mathcal{H} : \{0,1\}^* \to \{0,1\}^s\}] = 1/2. \quad (2)$$

From (1) and (2), we get

$$|Pr[\mathcal{A}'_{H(\cdot,k)} = 1 | k \xleftarrow{R} \{0,1\}^s]$$

$$- Pr[\mathcal{A}'_H = 1 | H \xleftarrow{R} \{\mathcal{H} : \{0,1\}^* \to \{0,1\}^s\}]| \geq \epsilon.$$

Therefore, we have proven the desired conclusion. ■

V. Interactive AuthSA Protocol

For the above protocol, if an adversary could capture an authentication token that a user is using to access a service provider, then he can use it to gain unauthorized access to other service providers that are accessible for the user in the meantime. To overcome this vulnerability, we improve the protocol to construct an interactive authentication protocol by applying ElGamal cryptosystem.

A. ElGamal Cryptosystem

ElGamal cryptosystem consists of the following three algorithms: the key generation, the encryption, and the decryption. For the details, refer to [8].

Key Generation :

The algorithm works as follows:

1) Generates a multiplicative cyclic group G with the order of a large prime p and the generator g.
2) Selects $x \xleftarrow{R} \mathbf{Z}_p$ and compute $y = g^x$.
3) Publishes $\{p, g, y\}$ as the public key, and retains x as the private key.

Encryption :

The algorithm works as follows: to encrypt a message m with the public key,

1) Chooses $t \xleftarrow{R} \mathbf{Z}_p$.
2) Computes $c_1 = g^t$ and $c_2 = my^t$.
3) Outputs the ciphertext $c = \{c_1, c_2\}$.

Decryption :

The algorithm works as follows: to decrypt a ciphertext $c = \{c_1, c_2\}$ with the private key.

1) Computes $h = c_1^x$.
2) Recovers the plaintext message $m = c_2 h^{-1}$.

B. Construction

Setup :

1) Service aggregator chooses a security parameter s, creates a multiplicative cyclic group G with the order of a large prime p and the generator g, and selects $k_i \xleftarrow{R} \{0,1\}^s$, $i = 0, 1, \cdots, r$, $r \in \mathbf{N}$, and a pseudo-random function $H : \{0,1\}^* \times \{0,1\}^s \longrightarrow \{0,1\}^s$. Then the

service aggregator sets the system parameter $PARA = \{s, k_0, \cdots, k_r, H, g, p\}$, publishes the system public keys $PUK = \{k_1, \cdots, k_r, H, g, p\}$, and keeps the private key $PRK = \{k_0\}$.

2) A user u chooses $x \xleftarrow{R} \mathbf{Z}_p$ and computes $y = g^x$. She takes $Prk_u = \{x\}$ and $Pub_u = \{y\}$ as her private key and public key, respectively. Then she registers with the service aggregator by supplying her identity and password (as credential) (ID_u, PW_u).
3) This step is the same as in the above one-way AuthSA protocol.

Sign-On :

Logon(ID_u, C_u) :

This algorithm is the same as in the above one-way AuthSA protocol.

BuildAuthToken$(PARA, ID_u, C_u, Puk_u, \{ID_{sp_i}, C_{sp_i}\}_{i=1,\cdots,m})$:

For a legitimate user ($i.e.$, the output of the algorithm **Logon**(ID_u, C_u) is 1), the service aggregator runs this algorithm as follows.

1) The first 6 steps are the same as in the above one-way AuthSA protocol.
2) Chooses $t \xleftarrow{R} \mathbf{Z}_p$, computes $c_1 = g^t$ and $c_2 = My^t$, and sets $M' = \{c_1, c_2\}$.
3) Outputs an authentication token $AT = \{ts, M', BF\}$ for the user u.

Verify$(PUK, AT, Prk_u, ID_{sp_i}, C_{sp_i})$:

In order to access services provided by a service provider sp_i ($i \in [m]$), the user u shows the authentication token $AT = \{ts, M', BF\}$ to the service provider sp_i, who interacts with the user u as follows.

1) sp_i chooses $v \xleftarrow{R} \mathbf{Z}_p$, computes $c_2' = c_2 g^v$, and gives c_2' to u.
2) u computes $\tilde{c}_2 = c_2' c_1^{-x}$ and gives $\tilde{c}_2$ to sp_i.
3) sp_i computes $M = \tilde{c}_2 g^{-v}$.
4) The next 3 steps are the same as in the above one-way AuthSA protocol.

As stated in [2], ElGamal cryptosystem has a property of anonymity. This means that no body can guess the identity of the user from the public key and the ciphertext. Based on Theorem 1, we have the following theorem. The detailed proof is omitted.

Theorem 2 *The proposed interactive AuthSA protocol is semantically secure under the security game IND-CIA.*

VI. Optimizing Parameters

The proposed protocol uses the Bloom filter to build authentication token for users to access external services anonymously, so it induces false positives inevitably. Now we compute the probability of a false positive occurring in a Bloom filter. Let's assume that the size of the Bloom filter is m bits, the number of hash keys is r, the maximum of service providers accessible for a user is n, and the hash function H behaves as a random oracle. The probability of a false positive is

$$\begin{aligned} Pr &= (1 - (1 - \tfrac{1}{m})^{rn})^r \\ &\approx (1 - e^{-\frac{rn}{m}})^r \end{aligned}$$

This means that three parameters r, n and m affect the probability of a false positive in different (positive or negative) ways. In order to minimize probability of a false positive, we compute the derivative of Pr with respect to r, $\frac{\mathrm{d}Pr}{\mathrm{d}r}$, and let it equal to 0 to obtain the optimal parameter

$$r = \tfrac{m \ln 2}{n}.$$

Then, the minimum probability of a false positive is

$$Pr = 2^{-r}.$$

Therefore, we can control the probability of false positive by adjusting the related parameter r, $i.e.$, adjusting the ratio of m to n. On the other hand, we point out that the false negatives never happen.

Now we describe the procedure showing how to select the suitable parameters as follows.

Step 1 Choose the desired probability of a false positives rate Pr and compute the number of pseudo-random function keys required,

$$r = \lceil -\lg Pr \rceil .$$

Step 2 Estimate the upper bound of the number of service providers accessible for a user, n, which should satisfy the future update.

Step 3 Once the parameters r and n are determined, the size of Bloom filter can be obtained by computing

$$m = \left\lceil \frac{nr}{\ln 2} \right\rceil .$$

VII. Discussion

First we describe other two possible methods of constructing AuthSA protocols. Let's recall the mechanism of classic (non-interactive) signature-based authentication protocols. Usually a message M of Sender A is accompanied by the signature $Sigh_{prk_A}(M)$. Receiver B can know that M really

is from A by confirming that the signature is signed on the message M by A with her private key prk_A.

We can apply any classic (non-interactive) signature-based authentication protocol to construct an AuthSA protocol by simply replacing the Bloom filter BF in the proposed one-way AuthSA protocol with the signature set $\{Sigh_{prk_{sp_1}}(M), \cdots, Sigh_{prk_{sp_n}}(M)\}$. When a service provider SP_j ($j \in [n]$) receives the authentication token, she can decide to or not to authenticate the user by verifying whether the j-th signature $Sigh_{prk_{sp_j}}(M)$ is generated with her private key prk_{sp_j}. We call such a straightway application of signature-based authentication protocol a naive AuthSA protocol.

Another way to construct an AuthSA protocol might be based on multi-user signature schemes, such as group, ring, and aggregate signatures. We just give a general idea to the construction as follows. Let $MSign_{(prk_{sp_1},\cdots,prk_{sp_1})}(M)$ denote a signature on the message M with the private keys of the service providers $\{sp_1, \cdots, sp_1\}$. The Bloom filter BF in the proposed one-way AuthSA protocol is replaced with the signature $MSign_{(prk_{sp_1},\cdots,prk_{sp_1})}(M)$. When a service provider receives the authentication token, she authenticates the user if she confirms that the signature $MSign_{(prk_{sp_1},\cdots,prk_{sp_1})}(M)$ is signed on the message M with her private key prk_{sp_j}. We did not try to modify any multi-user signature scheme to construct an AuthSA protocol, but we believe it should be possible. We call such a scheme a MS AuthSA protocol.

Now we consider an example of the proposed one-way AuthSA protocol to compute the size of the Bloom filter. Assume that the probability of a false positive is a millionth (10^{-6}), and the maximum number of service providers accessible to a user is 20. We use the method in Section VI to compute the size of Bloom filter $m = 578(bits)$. To evaluate the sizes of signatures in naive and MS AuthSA protocols, we use the data in [9] about the short signature sizes in BLS (Boneh, Lynn, and Shacham's) regular signature scheme [5], BBS (Boneh, Boyen, and Shacham's) group signature scheme [4], and CYH (Chow, Yiu, and Hui's) ring signature scheme [6]. Then the comparison is as follows.

Table 1. Comparison of the Token

Scheme	Naive (BLS)	MS (BBS)	MS (CYH)	Ours
Size	3200 $bits$	2400 $bits$	3360 $bits$	578 $bits$

Note that the size is just of the third part of the authentication ($i.e.$, the Bloom filter or signature), not of the whole token.

From the above table, we know that our schemes are the best on the size of the authentication token. As stated in [9], the size of ring signature grows linearly with the number of service providers. For the computation complexity, different algorithms and hash functions have different computation expenses. Therefore, without the detailed constructions, we cannot give the detailed evaluation on them. However, all short signature schemes require expensive pairing operations. According to the experiment of http://www.shamus.ie, the calculation of one pairing takes at least 4 times or more than that of RSA. Instead, the proposed protocols use the binary check to test if an authentication token is valid, which speeds up service providers' verification more efficiently.

VIII. Conclusions and Open Problem

In this paper, we described the requirement of authentication in service aggregation, and formulated a formal definition and a security game for it. Based on Bloom filter and ElGamal cryptosystem, we proposed a one-way authentication protocol and improved it to get an interactive one. Their security was proven. We also discussed how to choose the optimal system parameters, and gave efficiency analysis showing that the proposed protocols are the most efficient.

We are working on security issues in service aggregation in the cloud environment. How to authenticate a user for such a situation is an open problem.

Acknowledgment

The authors would like to thank Smart Services CRC Australia for the support in making this work and the industry partners who provided valuable input on the direction of this work.

References

[1] D. Balfanz, D.K. Smetters, P. Stewart, and H.C. Wong, Talking to Strangers: Authentication in Ad-Hoc Wireless Networks. In: *Proceedings of Network and Distributed System Security Symposium* 2002 (NDSS '02), 2002.

[2] M. Bellare, A. Boldyreva, A. Desai, and D. Pointcheval, Key-Privacy in Public-Key Encryption. In: *ASIACRYPT 2001*. LNCS, vol. 2248, pp. 566-582. Springer, Heidelberg, 2001

[3] B.H. Bloom, Space/time trade-offs in hash coding with allowable errors. In *Communications of ACM*, vol. 13, no. 7, pp. 422-426, July 1970.

[4] D. Boneh, X. Boyen, and H. Shacham, Short group signatures. In: Franklin, M. (ed.) *CRYPTO 2004*. LNCS, vol. 3152, pp. 41-55. Springer, 2004.

[5] D. Boneh, B. Lynn, and H. Shacham, Short signatures from the Weil pairing. In: Boyd, C. (ed.) *ASIACRYPT 2001*. LNCS, vol. 2248, pp. 514-532. Springer, 2001.

[6] S.S.M. Chow, S.-M. Yiu, and L.C.K. Hui, Efficient identity based ring signature. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) *ACNS 2005*. LNCS, vol. 3531, pp. 499-512. Springer, 2005.

[7] S.J. Creese, M.H. Goldsmith, R. Harrison, A.W. Roscoe, P. Whittaker, and I. Zakiuddin, Exploiting empirical engagement in authentication protocol design. In D. Hutter and M. Ullmann, editors, *Proceedings of 2nd International Conference on Security in Pervasive Computing (SPC 2005)*, LNCS, vol. 3450, Springer, 2005.

[8] T. ElGamal, A Public Key Cryptosystem and a Signature Scheme based on Discrete Logarithms. In *IEEE Trans. Inf. Theory*, vol. 31, pp. 469-472, 1985.

[9] A.L. Ferrara1, M. Green, S. Hohenberger, and M.Φ. Pederse, Practical Short Signature Batch Verification. In M. Fischlin (Ed.), *CT-RSA 2009*, LNCS 5473, pp. 309-324, Springer, 2009.

[10] C. Gehrmann, C. Mitchell, and K. Nyberg, Manual Authentication for Wireless Devices. *RSA Cryptobytes*, vol. 7, no. 1, pp. 29-37, 2004.

[11] R. Khalaf and F. Leymann, On web services aggregation. In: *Technologies for E-Services* 2003. LNCS, vol. 2819, pp. 1-13. Springer, Heidelberg, 2003.

[12] S. Laur and K. Nyberg, Efficient mutual data authentication using manually authenticated strings. In: *Proceedings of the 5th International Conference on Cryptology and Network Security*. LNCS 4301, pp. 90-107, 2006.

[13] L.H. Nguyen and A.W. Roscoe, Efficient group authentication protocol based on human interaction. *Proceedings of the Workshop on Foundations of Computer Security and Automated Reasoning for Security Protocol Analysis (FCS-ARSPA'06)*, pp. 9-31, August 2006.

[14] L.H. Nguyen and A.W. Roscoe, Authentication protocols based on low-bandwidth unspoofable channels: a comparative survey In: *Journal of Computer Security*, 2010.

[15] J. Suomalainen, J. Valkonen, and N. Asokan, Security Associations in Personal Networks: A Comparative Analysis. In: *Proceedings of the Fourth European Workshop on Security and Privacy in Ad hoc and Sensor Networks (ESAS)*, Cambridge, UK, July 2007. LNCS, Vol. 4572, pp. 43-56, Springer, 2007.

[16] S. Vaudenay, Secure Communications over Insecure Channels Based on Short Authenticated Strings. *Advances in Cryptology - Crypto 2005*, LNCS vol. 3621, Springer-Verlag, pp. 309-326, 2005.

[17] P. Wang, L. Dong, Y. Mu, W. Susilo, and J. Yan, A Framework for Privacy Policy Management in Service Aggregation. In: *14th International Conference on Computer Supported Cooperative Work in Design (CSCWD 2010)*, pp. 166-171, 2010.

[18] F.L. Wong and F. Stajano, Multi-channel Protocols. *Proceedings of the 13th International Workshop on Security Protocols*, Cambridge, England, 20-22 Apr 2005, LNCS.