A Generic Construction of Dynamic Single Sign-on with Strong Security

Jinguang Han[1,3], Yi Mu[1], Willy Susilo[1], and Jun Yan[2]

[1] Centre for Computer and Information Security Research
School of Computer Science and Software Engineering
[2] School of Information Systems and Technology
University of Wollongong, NSW2522, Australia
[3] College of Science, Hohai University, Nanjing 210098, China
{jh843,ymu,wsusilo,jyan}@uow.edu.au

Abstract. Single Sign-On (SSO) is a core component in a federated identity management (FIM). Dynamic Single Sign-on (DSSO) is a more flexible SSO where users can change their service requirements dynamically. However, the security in the current SSO and DSSO systems remain questionable. As an example, personal credentials could be illegally used to allow illegal users to access the services. It is indeed a challenging task to achieve strong security in SSO and DSSO. In this paper, we propose a generic construction of DSSO with strong security. We propose the formal definitions and security models for SSO and DSSO, which enable one to achieve the security of SSO and DSSO with the underlying (standard) security assumptions. We also provide a formal security proof on our generic DSSO scheme.

Keywords: Single Sign-on, Authentication, Security.

1 Introduction

With an increasing use of personalized/protected services, users need to maintain more and more usernames and the corresponding passwords in order to access the entitled services. This imposes a burden on users. Single Sign-on (SSO) provides a good remedy to this problem, as it allows a single password to be used to access multiple services. A traditional SSO system comprises three entities: an identity provider (IdP), a group of users (Us) and a group of service providers (SPs). The IdP manages the user's personally identifiable information (PII), authenticates network users and issues credentials to them. SPs provide services to users once they are authenticated by the IdP. SSO is a system where a user authenticates himself to the IdP and can access the designated SPs without the need for further authentication [24]. SSO can shift the great administrative burden of the numerous users profiles from SPs to the IdP. Hence, SSO plays a core role in the federated identity management (FIM) where the exchange of the user's identity-related information can be optimized [5].

Unfortunately, current SSO systems have some obvious flaws. For instance, they are fragile to resist single point of failure [16,22,23]. The main reason for this

S. Jajodia and J. Zhou (Eds.): SecureComm 2010, LNICST 50, pp. 181–198, 2010.

is that the IdP must always be online, otherwise users cannot be granted services from SPs. They are not well protected from illegally using a personal credential, where a credential could be used for an illegitimate user to gain the services which should not be accessed by him. These systems are subject to impersonation attacks. When a password is compromised, the attacker can impersonate the user and log in using the compromised account. This is mainly due to the missing of individual participation principle provided in the thirteenth principle of Organization for Economic CO-Operation and Development (OECD) [20] and the missing of the user control and consent principle for the laws of identity [6]. All these flaws stem from the lack of active/dynamic control over the process by the user, after the user has entered the correct password. In the following, we review some existing SSO systems. Although those systems provide elegant solutions to SSO, they suffer from various attacks.

Released 1999, Microsoft .NET Passport is one of the most widely deployed SSO systems, where a passport server acts as the IdP [22]. It uses cookies to store and convey user's PII. When a user access to an SP, the SP redirects the user to the passport server for authentication. After authentication, the passport server creates three cookies: ticket cookie, profile cookie and visited sits cookie. The ticket cookie contains the unique identifier and a timestamp. The profile cookie consists of the user's profile information. The visited sits cookie contains the lists sites the user has accessed. All cookies created by the passport server are encrypted with the triple DES encryption algorithm under the shared key between the passport server and all SPs. The passport server sends these cookies to the user. The user redirects them to the SP. The SP decrypts the cookies and obtains the user's authentication information. .Net Passport incurs some attacks, such as single point of failure, key management failure, misuse of cookies, etc. [16,22,23].

In September 2001, the Liberty Alliance Project was launched [17]. This project was aimed to create an open, federated, SSO solution for the digital economy via any device connected to the Internet. The Liberty project does not use cookies to transfer information between IdPs and SPs. Instead, it transfers information through HTTP redirects and URL encodings. In Liberty Alliance, an SSO Service (SSOS) provides users an Identity Web Services Framework (ID-WSF)-based means to obtain Liberty authentication assertions enabling them to interact with SPs [18]. In this system, the user only shows his credentials to the SP, without proving the ownership of them. Therefore, it is unable to prevent credential transfer, namely the user can share his credentials with other illegal users.

Proposed in 2005, OpenID is an open, decentralized standard for authenticating users. In OpenId, users are allowed to access to different services with the same digital identity where the SPs trust the IdP. OpenID solves the problem without relying on any centralized IdP to confirm digital identity. There are more than one IdP in OpenID system, users can get their OpenID from any IdP in the system. OpenID can be used as an effective mean for cross company authentication as well as for SSO. OpenID has two major modes of operation: Dumb

mode and Smart mode [21,28]. In the Dumb mode, the SP needs to compare the authentication assertion received from the user with the initial one stored in the IdP to prevent the malicious attackers. While in the Smart mode, the IdP encrypts the authentication assertion under the shared key between the SP and the IdP. Therefore, in both modes, the IdP must always be online to enable users authentication.

In 2003, Pahalidis and Mitchell presented a taxonomy of SSO system [24]. They divided SSO systems into four categories: local pseudo-SSO systems, proxy-based pseudo-SSO systems, local true SSO systems and proxy-based true SSO systems. They designed two SSO systems based on trusted platforms and GSM/UMTS, respectively [25,26]. In order to resolve the single point of failure, two distributed SSO systems Cornell SSO (CorSSO) and Threshold Passport (Thres-Passport) were proposed by Josephson and Chen in 2004 and 2005, respectively [7,15]. In these systems, the authentication key is split into n different shares, and each share is sent to an authentication server. Only authenticated by at least t authentication servers, can the user get services from an SP. Recently, user-centric federated identity management systems have been proposed to protect user's PII [30,27]. In 2009, based on a private credential mechanism, Suriadi and Foo proposed a user-centric federated SSO system (UFed SSO), in which the user can minimize the release of his PII [31]. Although, every system mentioned above has its merits, they did not provide a security proof.

In 2006, Bhargav-Spantzel and Camenisch [30] proposed a taxonomy and raised some open issues on user centric federated identity management systems. They classified the existing systems into two predominant variants: credential-focused systems and relation-focused systems. In credential-focused systems, the IdPs must be offline and issue long-term credentials. While in relationship-focused systems, users need to maintain the relationship with the online IdPs that create short-term credentials for them during transactions. They defined an universal user centric FIM which should have long-term as well as short-term credentials, online and offline IdP. However, this scheme has not been investigated thoroughly.

Our Contribution

In this paper, we propose a novel dynamic SSO scheme, which resists against all the previously described attacks. We formalize the definitions and the security models for SSO and DSSO. It is the first time that the formal definitions and security models for SSO and DSSO are formally defined. We give a generic construction of DSSO systems based on three building blocks: (1) CCA-secure broadcast encryption, (2) strongly existentially unforgeable signature, and (3) zero knowledge proof. We provide a formal security proof for our generic construction.

Paper Organization

The rest of this paper is organized as follows. In Section 2, we propose the formal definitions and security models for SSO and DSSO. We review the three building blocks which are used to construct DSSO in Section 3. In Section 4, a generic

construction for DSSO is described. In Section 5, we reduce the security of our construction to the underlying assumptions. Section 6 concludes this paper.

2 Formal Definitions and Security Models

In this section, we provide a formal definition and a security model for SSO and DSSO.

2.1 Single Sign-on

In SSO systems, a user needs to authenticate himself to the IdP once for access to multiple SPs without the need to re-authentication. In order to protect the PII of the user, an ideal SSO system should satisfy the basic requirement that only the intended SPs can check the user's PII. Now we formalise the definition of SSO as follows:

Definition 1. *A Single Sign-on system consists of five algorithms: system setup algorithm* $\mathsf{Setup}(\cdot)$*, enrollment algorithm* $\mathsf{Enrol}(\cdot)$*, credential generation algorithm* $\mathsf{CreGen}(\cdot)$*, credential verification algorithm* $\mathsf{CreVer}(\cdot)$*, and proof of knowledge algorithm* $\mathsf{PK}(\cdot)$*.*

- $\mathsf{Setup}(\lambda)$*: Taking as input a security parameter* $\lambda \in \mathbb{N}$*, it returns public parameters* PP *and a public-secret key pair* $(K_{IP}, K_{IS}) \leftarrow \mathcal{G}(1^\lambda)$ *for the IdP.*
- $\mathsf{Enrol}(PP, RI)$*: Taking as input public parameters* PP*,* SP_i*'s registration information* RI_{SPi} *or* U*'s registration information* RI_U*, it returns* (ID_{SP_i}, K_{SP_i}) *to* SP_i*, and* (ID_U, A_U) *to* U*, where* ID_{SP_i} *and* ID_U *are the identifiers of* SP_i *and* U *in the circle of trust (CoT)*[1]*,* K_{SP_i} *is* SP_i*'s verification key and* A_U *is* U*'s access right which is a set consisting of the identifiers of the service providers that the user has selected.* U *generates his public-secret key pair* $(K_{UP}, K_{US}) \leftarrow \mathcal{G}(1^\lambda)$*.*
- $\mathsf{CreGen}(K_{IS}, M_U, ID_U, T_U, K_{UP}, PP)$*: Taking as input the IdP's secret key* K_{IS}*, an authentication assertion* M_U*, user's identifier* ID_U*, user's public key* K_{UP}*, a timestamp* T_U *and public parameters* PP*, it returns a credential* Cre_U*.*
- $\mathsf{CreVer}(K_{SP_i}, Cre_U, M_U, ID_U, K_{UP}, T_U, K_{IP}, PP)$*: Taking as input the service provider* SP_i*'s verification key* K_{SPi}*, the user's credential* Cre_U*, the authentication assertion* M_U*, the user's identifier* ID_U*, user's public key* K_{UP}*, IdP's public key* K_{IP}*, the timestamp* T_U *and public parameters* PP*, it returns* $True$ *if and only if the service provider* $ID_{SP_i} \in A_U$ *and the credential* Cre_U *is created by the IdP. Otherwise it returns* $False$*.*
- $\mathsf{PK}((K_{US}, K_{UP}), \gamma)$*: Taking as input the user's public-secret key pair* (K_{UP}, K_{US}) *and a random number* γ*, it returns a number* s *such that* $Accept \leftarrow PKVer(s, K_{UP}, \gamma)$ *if and only if* K_{US} *is the user's secret key corresponding to the public key* K_{UP}*, where* $PKVer(\cdot)$ *is the verification algorithm in the proof of knowledge. Otherwise it returns* $Reject$*.*

[1] A circle of trust consists of the identity provider and service providers, where each service provider trusts the identity provider.

2.2 The Security of Single Sign-on

In SSO systems, three types of attacks should be considered: collusion credential forging attacks, collusion impersonation attacks and coalition credential forging attacks. In the collusion credential forging attacks, malicious users can collaboratively forge a credential for the target user. They can impersonate the target one to get services from the service providers whose identifiers are listed in the forged credential. In the collusion impersonation attacks, we assume that the malicious service providers have checked the credentials of a user and therefore obtained the corresponding proof information on the user. Hence, malicious service providers can collaboratively mimic the owner of the credentials. In the coalition credential forging attacks, malicious users and service providers can collaboratively forge a credential for the target user, in which the identifiers of the malicious service providers are not included.

In order to formalise our notions of security for SSO systems, we define a series of games between two Turing machines: Challenger and Adversary $\mathcal{A}$.

Game 1: Collusion Credential Forging Attacks.

Init. Let $\mathcal{A}$ be all malicious users. $\mathcal{A}$ begins by outputting the target user U^* for whom it wants to forge a credential.

Setup. The challenger runs $\mathsf{Setup}(\lambda)$ to generate the public parameter PP and the public-private key pair (K_{IP}, K_{IS}). It sends $\mathcal{A}$ the public parameter PP and the public key K_{IP}.

Enrollment queries. $\mathcal{A}$ can adaptively issue enrollment queries $\{RI_{U_1}, RI_{U_2}, \cdots, RI_{U_{qe}}\}$, where $RI_{U_i} \neq RI_U^*$, $qe \leq (|\mathbb{U}| - 1)$, $\mathbb{U}$ is the set consists of all users in the CoT. The challenger returns $\{(ID_{U_1}, A_{U_1}), (ID_{U_2}, A_{U_2}), \cdots, (ID_{U_{qe}}, A_{U_{qe}})\}$.

Credential generation queries. $\mathcal{A}$ can adaptively issue credential generation queries $\{(ID_{U_1}, K_{UP_1}), (ID_{U_2}, K_{UP_2}), \cdots, (ID_{U_{qc_1}}, K_{UP_{qc_1}})\}$, where $(ID_{U_i}, K_{UP_i}) \neq (ID_U^*, K_{UP}^*)$. The challenger returns $\{Cre_{U_1}, Cre_{U_2}, \cdots, Cre_{U_{qc_1}}\}$.

Credential verification queries. $\mathcal{A}$ can adaptively issue credential verification queries $\{(ID_{U_1}, K_{UP_1}, Cre_{U_1}), (ID_{U_2}, K_{UP_2}, Cre_{U_2}), \cdots, (ID_{U_{qc_2}}, K_{UP_{qc_2}}, Cre_{U_{qc_2}})\}$. Upon receiving a query, the challenger returns $True$ or $False$.

Output. $\mathcal{A}$ outputs a credential Cre_U^* for U^*. $\mathcal{A}$ wins the game if

1. $True \leftarrow \mathsf{CreVer}\ (K_{SP_i}, Cre_U^*, ID_U^*, K_{UP}^*, T_U^*, K_{IP}, PP)$ and
2. $(ID_U^*, Cre_U^*) \notin \{(ID_{U_1}, Cre_{U_1}), (ID_{U_2}, Cre_{U_2}), \cdots, (ID_{U_{qc_1}}, Cre_{U_{qc_1}})\}$.

Definition 2. *A Single Sign-on system is $(t, qe, qc_1, qc_2, \epsilon)$-secure against collusion credential forging attacks if no t-time adversary, who makes at most qe enrollment queries, qc_1 credential generation queries and qc_2 credential verification queries, has advantage at least ϵ in* Game 1.

Game 2. Collusion Impersonate Attacks.

Init. Let $\mathcal{A}$ be all malicious service providers in the CoT. $\mathcal{A}$ begins by outputting a user U^* whom it wants to impersonate.

Setup. The challenger runs $\mathsf{Setup}(\lambda)$ to obtain the public parameters PP and the public-secret key pair (K_{IP}, K_{IS}). It sends $\mathcal{A}$ the public parameter PP and the public key K_{IP}.

Proof of Knowledge queries. $\mathcal{A}$ can adaptively issues proof of knowledge queries $\{(K_{UP_1}, \gamma_1), (K_{UP_2}, \gamma_2), \cdots, (K_{UP_{qp}}, \gamma_{qp})\}$, where $K_{UP_i} \neq K^*_{UP}$. The challenger returns $\{s_1, s_2, \cdots, s_{qp}\}$.

Challenge. The challenger sends a challenge (K^*_{UP}, γ^*) to $\mathcal{A}$.

Output. $\mathcal{A}$ outputs s^*. $\mathcal{A}$ wins the game if

1. $Accept \leftarrow PKVer(s^*, K^*_{UP}, \gamma^*)$ and
2. $K^*_{UP} \notin \{K_{UP_1}, K_{UP_2}, \cdots, K_{UP_{qp}}\}$.

Definition 3. *A Single Sign-on system is (t, qp, ϵ)-secure against collusion impersonation attacks if no t-time adversary, who makes at most qp proof of knowledge queries, has advantage at least ϵ in* Game 2.

Game 3: Coalition Credential Forging Attacks.

Init. Let $\mathcal{A}$ be the coalition, which consists of all malicious users and service providers. $\mathcal{A}$ begins by outputting a user U^* that it wants to impersonate and a service provider SP^* that it wants to attack.

Setup. The challenger runs $\mathsf{Setup}(\lambda)$ to generate the public parameters PP and the public-secret key pair (K_{IP}, K_{IS}). It sends $\mathcal{A}$ the public parameter PP and the public key K_{IP}.

Enrollment queries. $\mathcal{A}$ can adaptively issue enrollment queries $\{RI_{U_1}, RI_{U_2}, \cdots, RI_{U_{qe_1}}\}$ and $\{RI_{SP_1}, RI_{SP_2}, \cdots, RI_{SP_{qe_2}}\}$, where $RI_{U_i} \neq RI^*_U$, $RI_{SP_i} \neq RI^*_{SP}$, $qe_1 \leq (|\mathbb{U}| - 1)$, $qe_2 \leq (|\mathbb{SP}| - 1)$, $\mathbb{SP}$ is the set consists of all service providers in the CoT, and $qe_1 + qe_2 = qe$. The challenger returns $\{(ID_{U_1}, A_{U_1}), (ID_{U_2}, A_{U_2}) \cdots, (ID_{U_{qe_1}}, A_{U_{qe_1}})\}$ and $\{(ID_{SP_1}, K_{SP_1}), (ID_{SP_2}, K_{SP_2}), \cdots, (ID_{SP_{qe_2}}, K_{SP_{qe_2}})\}$.

Credential generation queries. $\mathcal{A}$ can adaptively issue credential generation queries $\{(ID_{U_1}, K_{UP_1}), (ID_{U_2}, K_{UP_2}), \cdots, (ID_{U_{qc_1}}, K_{UP_{qc_1}})\}$, where $(ID_{U_i}, K_{UP_i}) \neq (ID^*_U, K^*_{UP})$. The challenger returns $\{Cre_{U_1}, Cre_{U_2}, \cdots, Cre_{U_{qc_1}}\}$.

Credential verification queries. $\mathcal{A}$ can adaptively issue credential verification queries $\{(ID_{U_1}, K_{UP_1}, Cre_{U_1}), (ID_{U_2}, K_{UP_2}, Cre_{U_2}), \cdots, (ID_{U_{qc_2}}, K_{UP_{qc_2}}, Cre_{U_{qc_2}})\}$. The challenger returns $True$ or $False$.

Output. $\mathcal{A}$ outputs a credential Cre^*_U. $\mathcal{A}$ wins the game if

1. $True \leftarrow CreVer(K^*_{SP}, Cre^*_U, M^*_U, ID^*_U, K^*_{UP}, T^*_U, K_{IP}, PP)$.

2. $(ID_U^*, Cre_U^*) \notin \{(ID_{U_1}, Cre_{U_1}), (ID_{U_2}, Cre_{U_2}), \cdots, (ID_{U_{qc_1}}, Cre_{U_{qc_1}})\}$.

3. $ID_{SP}^* \in A_U^*$ and $ID_{SP} \notin A_U^*$ if $SP \in \mathcal{A}$.

Definition 4. *A Single Sign-on system is* $(t, qe, qc_1, qc_2, \epsilon)$*-secure against coalition credential forging attacks if no* t*-time adversary, who makes at most* qe *enrollment queries,* qc_1 *credential generation queries, and* qc_2 *credential verification queries, has advantage at least* ϵ *in the* Game 3.

2.3 Dynamic Single Sign-on

DSSO is an SSO system in which the user can change his choice dynamically. We formalise the definition of DSSO as follows:

Definition 5. *A dynamic single sign-on system (DSSO) consists of seven algorithms:* $\mathsf{Setup}(\cdot)$, $\mathsf{Enrol}(\cdot)$, $\mathsf{CreGen}(\cdot)$, $\mathsf{CreVer}(\cdot)$, $\mathsf{PK}(\cdot)$, *an addition algorithm* $\mathsf{A}(\cdot)$ *and a deletion algorithm* $\mathsf{D}(\cdot)$. *Where* $\mathsf{Setup}(\cdot)$, $\mathsf{Enrol}(\cdot)$, $\mathsf{CreGen}(\cdot)$, $\mathsf{CreVer}(\cdot)$ *and* $\mathsf{PK}(\cdot)$ *are the same as in definition 1.*

- A (ID_{SP}) : *Taking as input the service provider* SP*'s identifier* ID_{SP}*, it returns* $A_U \leftarrow A_U \bigcup \{ID_{SP}\}$.
- D (ID_{SP}) : *Taking as input the service provider* SP*'s identifier* ID_{SP}*, it returns* $A_U \leftarrow A_U \backslash \{ID_{SP}\}$.

2.4 The Security of Dynamic Single Sign-on

In multiple parties communication and dynamic schemes, because the participants can join or leave frequently, two special attacks should be addressed, namely forward security and backward security. In DSSO, users can be added to or revoked from a service dynamically; therefore a secure DSSO system can resist these attacks. By forward security, we mean that the SP can not validate the credentials, which were issued before he is added to the user's access right A_U. By backward security, we mean that the service providers can not validate the credentials, which are issued after he has been removed from the the user's access right A_U. We formalise these two attacks by the following games.

Game 4: Forward Security.

Setup. Let $\mathcal{A}$ be malicious service providers. The challenger runs $\mathsf{Setup}(\lambda)$ to generate the public parameters PP and the public-secret key pair (K_{IP}, K_{IS}). It sends $\mathcal{A}$ the public parameter PP and the public key K_{IP}.

Credential verification queries. $\mathcal{A}$ can adaptively issue credential verification queries $\{(ID_{U_1}, K_{UP_1}, Cre_{U_1}), (ID_{U_2}, K_{UP_2}, Cre_{U_2}), \cdots, (ID_{U_{qc}}, K_{UP_{qc}}, Cre_{U_{qc}})\}$, which were issued after $\mathcal{A}$ has been joined to A_U. The challenger returns $True$ or $False$.

Challenge. The challenger sends to $\mathcal{A}$ an old credential Cre_U^O, which was issued before he is joined to A_U.

Output. $\mathcal{A}$ outputs $True$ or $False$. $\mathcal{A}$ wins the game if his answer on Cre_U^O is correct.

Definition 6. *A Dynamic Single Sign-on system is* (t, qc, ϵ)*-forward secure if no t-time adversary, who makes at most qc credential verification queries, has advantage at least* ϵ *in the* Game 4*.*

Game 5: Backward Security.

Setup. Let $\mathcal{A}$ be malicious service providers. The challenger runs $\mathsf{Setup}(\lambda)$ to generate the public parameters PP and the public-secret key pair (K_{IP}, K_{IS}). It sends $\mathcal{A}$ the public parameter PP and the public key K_{IP}.

Credential verification queries. $\mathcal{A}$ can adaptively issue credential verification queries $\{(ID_{U_1}, K_{UP_1}, Cre_{U_1}),\ (ID_{U_2}, K_{UP_2}, Cre_{U_2}),\ \cdots,\ (ID_{U_{qc}}, K_{UP_{qc}}, Cre_{U_{qc}})\}$, which were issued before $\mathcal{A}$ is deleted from A_U. The challenger returns $True$ or $False$.

Challenge. The challenger sends $\mathcal{A}$ a new credential Cre_U^N which was issued after he has been deleted from A_U.

Output. $\mathcal{A}$ outputs $True$ or $False$. $\mathcal{A}$ wins the game if his answer on Cre_U^N is correct.

Definition 7. *A Dynamic Single Sign-on system is* (t, qc, ϵ)*-backward secure if no t-time adversary, who makes at most qc credential verification queries, has advantage at least* ϵ *in the* Game 5*.*

3 Building Blocks

In this section, we provide three building blocks, which are used to construct DSSO systems.

3.1 Broadcast Encryption System

The notion of broadcast encryption was proposed by Fiat and Naor in 1993 [10]. A broadcast encryption system consists of three randomized algorithms:

- $\mathsf{Setup}(n, \lambda)$: Taking as input the number of receivers n and security parameter λ, it outputs n secret keys $K_{R1}, K_{R2}, \cdots, K_{Rn}$ and public parameters PP_B.
- $\mathsf{Encrypt}(S, PP_B)$: Taking as input a subset $S \subseteq \{ID_1, ID_2, \cdots, ID_n\}$ and public parameters PP_B, it outputs a pair (Hdr, K), where Hdr is called the header and $K \in \mathcal{K}$ is a message encryption key. (S, Hdr) is often called the full header.
- $\mathsf{Decrypt}(Hdr, K_{Ri}, PP_B)$: Taking as input the header Hdr, the secret key K_{Ri} for the receiver $ID_i \in S$ and the public parameter PP_B, it outputs the message encryption key $K \in \mathcal{K}$.

3.2 Chosen Ciphertext Security of Broadcast Encryption System

The chosen ciphertext security of broadcast encryption system is defined using the following game between a challenger and an adversary $\mathcal{A}$ [1,9].

Init. The adversary $\mathcal{A}$ outputs a receivers set $S^* \subseteq \{ID_1, ID_2, \cdots, ID_n\}$ which he wants to attack.

Setup. The challenger runs $\mathsf{Setup}(n, \lambda)$ to generate secret keys $K_{R1}, K_{R2}, \cdots, K_{Rn}$ and public parameters PP_B. It sends $\mathcal{A}$ all secret key K_{Ri} for $ID_i \notin S^*$.

Query phase 1. $\mathcal{A}$ issues decryption queries $q_1, q_2, \cdots, q_t$, where $q_i = (Hdr, ID_l)$, $ID_l \in S^*$. The challenger responds with $\mathsf{Decrypt}(Hdr, K_{R_l}, PP_B)$.

Challenge. The challenger runs algorithm $\mathsf{Encrypt}(S^*, PP_B)$ to obtain (Hdr^*, K), where $K \in \mathcal{K}$. The challenger chooses a random $b \in \{0, 1\}$. It sets $K_b = K$ and chooses a random $K_{1-b} \in \mathcal{K}$. It sends (Hdr^*, K_0, K_1) to $\mathcal{A}$.

Query phase 2. $\mathcal{A}$ can adaptively issue decryption queries $q_{t+1}, q_{t+2}, \cdots q_d$, where $q_j = (Hdr, ID_l)$, $ID_l \in S^*$. The only constraint is that $Hdr \neq Hdr^*$. The challenger returns $\mathsf{Decrypt}(Hdr, K_{R_l}, PP_B)$.

Guess. $\mathcal{A}$ outputs its guess $b' \in \{0, 1\}$ for b. $\mathcal{A}$ wins the game if $b = b'$.

Definition 8. *A broadcast encryption system is (t, n, q_d, ϵ) CCA-secure if no t-time adversary $\mathcal{A}$, who makes at most q_d decryption queries, has advantage at least ϵ in the above game.*

3.3 Signature Scheme

Digital signature scheme was proposed by Diffie and Hellman [8]. A signature scheme consists of four algorithms:

- $\mathsf{Setup}(\gamma)$: Taking as input the security parameter γ, it outputs the public parameters PP_S.
- $\mathsf{KeyGen}(\gamma, PP_S)$: Taking as input the security parameter γ and the public parameters PP_S, it outputs a public-secret key pair (K_S, K_P).
- $\mathsf{Sign}(K_S, m, PP_S)$: Taking as input the secret key K_S, a message m and the public parameters PP_S, it outputs a publicly verifiable signature σ.
- $\mathsf{Ver}(m, \sigma, K_P, PP_S)$: Taking as input the message m, the signature σ, the public key K_P and the public parameters PP_S, it outputs $True$ if the signature is correct. Otherwise it outputs $False$.

3.4 Strong Unforgeability of Signature

A digital signature system is said to be secure if it is existentially unforgeable under a chosen-message attack [2,13]. The strong unforgeability of signature is defined using the following game between a challenger and an adversary $\mathcal{A}$.

Setup. The challenger runs $\mathsf{Setup}(\gamma)$ and $\mathsf{KeyGen}(\gamma, PP_S)$ to generate the public parameters PP_S and a public-secret key pair (K_P, K_S). It sends public parameters PP_S and public key K_P to $\mathcal{A}$.

Signature queries. $\mathcal{A}$ can adaptively issue up to qs signature queries $\{m_1, m_2, \cdots, m_{qs}\}$. To each query m_i, the challenger runs algorithm $\mathsf{Sign}(K_S, m_i, PP_S)$ to produce the corresponding signature σ_i. The challenger responds with message-signature pairs $\{(m_1, \sigma_1), (m_2, \sigma_2), \cdots, (m_{qs}, \sigma_{qs})\}$.

Output. $\mathcal{A}$ outputs a message-signature pair (m^*, σ^*). $\mathcal{A}$ wins the game if

1. $True \leftarrow Ver(m^*, \sigma^*, K_P, PP_S)$ and
2. $(m^*, \sigma^*) \notin \{(m_1, \sigma_1), (m_2, \sigma_2), \cdots, (m_{qs}, \sigma_{qs})\}$.

Definition 9. *A signature is (t, qs, ϵ)-strongly existentially unforgeable under adaptive chosen-message attacks if no t-time adversary, who makes at most qs signature queries, has advantage at least ϵ in the above game.*

3.5 Zero Knowledge Proof

Zero knowledge proof (ZKP) was introduced by Goldwasser, Micali and Rackoff in 1985 [14]. It is an interactive protocol by which a prover P (Peggy) can convince a verifier V (Victor) that he knows a secret without revealing any information about it to V. The formal definition of zero knowledge proof is as follows:

Definition 10. *Let (P, V) be a pair of Turing machines and V is polynomially bounded. P and V share the same input and can interact with each other. Let L be a language. We say that a pair (P, V) is zero knowledge proof system, if P and V satisfy the following properties:*

- Completeness: *For any input $x \in L$ to (P, V), $Pr[s \leftarrow (P, V)(x), V(x, s) = 1] \geq 1 - \frac{1}{n^k}$, for each k and sufficiently large n which denotes the length of the input.*
- Soundness: *For any $x \notin L$, and any prover P', $Pr[s \leftarrow (P', V)(x), V(x, s) = 1] < \frac{1}{n^k}$.*
- Zero-knowledge: *For any $x \in L$, and any verifier V', there exists a simulator S such that two distribution $S_{V'}(x)$ and $Vew_{V'}(x)$ are computationally indistinguishable.*

Any language in $\mathcal{NP}$ has an interactive zero knowledge proof system [11,12]. Let (P, V) be an interactive zero knowledge proof system. By $(P, V)(x)$ we denote that the prover P executes an interactive zero knowledge proof protocol with the verifier V to prove that he knows the secret corresponding to x.

4 Generic Construction for Dynamic Single Sign-on

Our generic construction for DSSO consists of three building blocks: a CCA-secure broadcast encryption scheme $\mathsf{BroEnc}(\cdot)$, a strongly unforgeable signature

scheme Sign$(\cdot)$ and a zero knowledge proof scheme (P,V)$(\cdot)$. In our construction, users can change their choices dynamically, while other participants (users and SPs) in the system do not need to change their credentials. When the user logs in, the IdP creates a credential for him. The user can then use this credential to access all designated SPs, instead of sending different credentials to different SPs. For each logging request, the IdP creates a new credential for the user. At this point of time, a user can also be revoked due to expiry of his membership, for instance. Our construction can prevent illegal credential sharing, which is defined as all-or-nothing non-transferability. By all-or-nothing non-transferability, we mean that all the credentials of a user are shared, once he shares one of them with others [30,3,4,19]. Figure 1 provides the architecture of our construction.

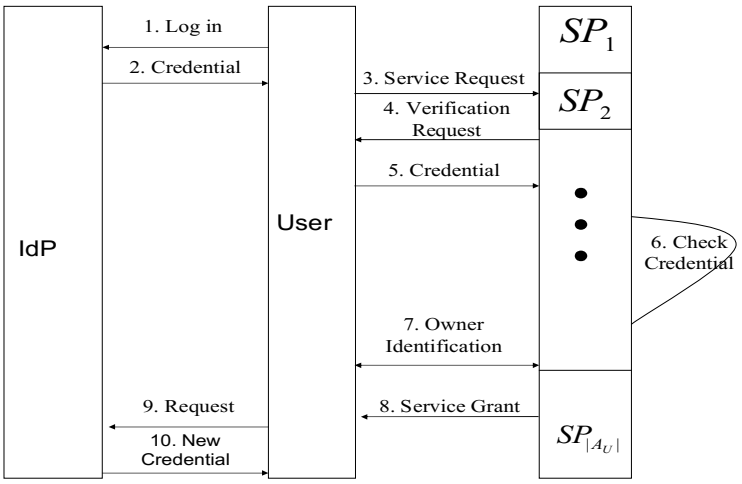

Fig. 1. DSSO Architecture

1. System Set-up. Runs the Setup(λ) to generate the public parameters PP, which includes all public parameters in the three underlying building blocks, and a public-secret key pair $(K_{IP}, K_{IS}) \leftarrow \mathcal{G}(1^\lambda)$ for the IdP, where λ is the security parameter.
2. Enrollment.
   (a) Service providers enrollment. SP_i submits his necessary registration information RI_{SPi} to the IdP. The IdP issues an identifier ID_{SPi} for SP_i, sends a verification key K_{SPi} to him, which is regarded as the receiver key in the broadcast encryption scheme, and stores $(SP_i, ID_{SPi}, K_{SPi})$ for him.
   (b) User enrollment. U sends his necessary registration information RI_U to the IdP. The IdP issues an identifier ID_U for him. The user generates his public-secret key pair $(K_{UP}, K_{US}) \leftarrow \mathcal{G}(1^\lambda)$ and sends the public key K_{UP} to the IdP. The IdP decides the user's access right A_U, which is a set that consists of the identifiers of the service providers that the user can access, and stores (ID_U, K_{UP}, A_U) for the user. Note that A_U will be regarded as the receiver set S in broadcast encryption.

3. Single Sign-on.
   (a) Log in. U uses his username and corresponding password to log in the system.
   (b) Credential generation. The IdP runs $\mathsf{BroEnc}(|A_U|)$ to generate the broadcast encryption key K which can only be computed by the service providers whose identifiers are listed in A_U, and encapsulates it in (A_U, Hdr). IdP generates a signature $\delta_U = \mathsf{Sign}(K_{IS}, M_U, ID_U, K_{UP}, T_U)$, where K_{IS} is the secret key of IdP, M_U is an authentication assertion, ID_U is the user's identifier, K_{UP} is the user's public key and T_U is a timestamp. Then, IdP encrypts the signature δ_U under K. The credential for the user is
   $$Cre_U = (A_U, Hdr, D), \quad \text{where} \quad D = E_K(\delta_U, M_U, ID_U, K_{UP}, T_U).$$
   (c) Service request. U sends a service request to the service provider SP_i $(ID_{SPi} \in A_U)$.
   (d) Verification request. SP_i asks U to show his credential to him.
   (e) Credential verification. U sends Cre_U to SP_i. SP_i computes the broadcast encryption key K from (A_U, Hdr) using his verification key K_{SPi}, decrypts $D = E_K(\delta_U, M_U, ID_U, K_{UP}, T_U)$ and verifies the signature δ_U. If δ_U is a valid signature on (M_U, ID_U, K_{UP}, T_U), SP_i executes the next step. Otherwise SP_i aborts.
   (f) Owner identification. U executes zero knowledge proof protocol (U, SP_i) (K_{UP}) with SP_i to prove that he knows the secret key K_{US} corresponding the public key K_{UP} included in Cre_U.
   (g) Service grant. If the zero knowledge proof is successful, SP_i grants the services to the user. Otherwise, SP_i rejects the services.

   If the user wants to access to other SPs whose identifiers are listed in A_U, he can send Cre_U to them directly, without having to request the IdP to issue a new credential for him, namely step (a) and (b) can be omitted.

4. Dynamic Change.

   If the user needs to change his access right, when he logs in, he must submit a request to the IdP. After checking it, the IdP creates a new credential for the user, according to his current status.
   (a) Request. U must submit a request for changing A_U to the IdP. After checking the request, the IdP does the following two changes on A_U.
   (b) Add. The IdP adds a service provider SP_j to the user's access right A_U by setting $A_U \leftarrow A_U \bigcup \{ID_{SP_j}\}$, and updates the broadcast encryption key K.
   (c) Delete. The IdP deletes a service provider SP_j from A_U by setting $A_U \leftarrow A_U \backslash \{ID_{SP_j}\}$, and updates the broadcast encryption key K.
   (d) New credential generation. The IdP uses the updated broadcast encryption key K to generate a new credential for U.

5 Security Analysis

In this section, we prove that our construction for DSSO is secure against collusion credential forging attacks, collusion impersonate attacks and coalition credential forging attacks, and provides forward security and backward security.

Theorem 1. *Our generic construction for DSSO is* $(t, qe, qc_1, qc_2, \epsilon)$ *-secure against collusion credential forging attacks if the broadcast encryption scheme is* (t, n, qc_1, ϵ_1) *CCA-secure and the signature scheme is* $(t, qc_2, \epsilon.(1-\epsilon_1)^{qc_2})$*-strongly existentially unforgeable.*

Proof. Suppose there exists t-time malicious users $\mathcal{A}$ that $(t, qe, qc_1, qc_2, \epsilon)$ breaks the collusion credential unforgeability of our generic construction for DSSO. We will show that there exists an algorithm $\mathcal{B}$ who can $(t, qc_2, \epsilon.(1-\epsilon_1)^{qc_2})$ breaks the strongly existential unforgeability of the underlying signature scheme.

Init. Algorithm $\mathcal{B}$ runs $\mathcal{A}$ and receives a user U^* for whom $\mathcal{A}$ wants to forge a credential.

Setup. $\mathcal{B}$ sends the public parameters PP and the IdP's public key K_{IP} to $\mathcal{A}$.

Enrollment queries. $\mathcal{A}$ can adaptively issues at most qe enrollment queries $\{RI_{U_1}, RI_{U_2}, \cdots, RI_{U_{qe}}\}$, where $RI_{Ui} \neq RI_U^*, qe \leq (|\mathbb{U}|-1)$. $\mathcal{B}$ returns $\{(ID_{U_1}, A_{U_1}), (ID_{U_2}, A_{U_2}), \cdots, (ID_{Uqe}, A_{U_{qe}})\}$.

Credential generation queries. $\mathcal{A}$ can adaptively issue credential generation queries $\{(ID_{U_1}, K_{UP_1}), (ID_{U_2}, K_{UP_2}), \cdots, (ID_{U_{qc_1}}, K_{UP_{qc_1}})\}$, where $(ID_{U_i}, K_{UP_i}) \neq (ID_U^*, K_{UP}^*)$. $\mathcal{B}$ redirects these queries to the challenger. The challenger returns $\{Cre_{U_1}, Cre_{U_2}, \cdots, Cre_{U_{qc_1}}\}$, where $Cre_{U_i} = (A_{U_i}, Hdr_i, D_i)$, $D_i = E_{K_i}(\delta_i, M_{U_i}, ID_{U_i}, K_{UP_i}, T_{U_i})$ and $\delta_i = \mathsf{Sign}(K_{IS}, M_{U_i}, ID_{U_i}, K_{UP_i}, T_{U_i})$.

Credential verification queries. $\mathcal{A}$ can adaptively issue credential verification queries $\{(ID_{U_1}, K_{UP_1}, Cre_{U_1}), (ID_{U_2}, K_{UP_2}, Cre_{U_2}), \cdots, (ID_{U_{qc_2}}, K_{UP_{qc_2}}, Cre_{U_{qc_2}})\}$. $\mathcal{B}$ redirects these queries to the challenger. The challenger returns $True$ or $False$.

Output. $\mathcal{A}$ outputs a credential $Cre_U^* = (A_U^*, Hdr^*, D^*)$, where $(ID_U^*, Cre_U^*) \notin \{(ID_{U_1}, Cre_{U_1}), (ID_{U_2}, Cre_{U_2}), \cdots, (ID_{U_{qc_1}}, Cre_{U_{qc_1}})\}$

$\mathcal{B}$ sends Cre_U^* to SP_l ($ID_{SP_l} \in A_U^*$). SP_l returns the corresponding plaintext $(M_U^*, ID_U^*, K_{UP}^*, T_U^*)$ or $\perp$ for meaningless ciphertext.

1. If $\perp$ is responded, namely D^* is not the corresponding ciphertext of $(M_U^*, ID_U^*, K_{UP}^*, T_U^*)$ under the encryption key K^* encapsulated in Hdr^*, $\mathcal{B}$ aborts. The simulation fails.
2. If the corresponding plaintext $(M_U^*, ID_U^*, K_{UP}^*, T_U^*)$ is responded, namely $\mathcal{A}$ can get the broadcast encryption key K^* from Hdr^*, $\mathcal{B}$ will abort. $\mathcal{B}$ can use $\mathcal{A}$ to break the broadcast encryption scheme. Due to the broadcast encryption is (t, n, qc_1, ϵ_1) CCA-secure, the probability that $(M_U^*, ID_U^*, K_{UP}^*, T_U^*)$ is received is at most ϵ_1.

3. If $\mathcal{B}$ does not abort, he can obtain a valid signature δ^* on $(M_U^*, ID_U^*, K_{UP}^*, T^*)$ at the same advantage ϵ.

Now we compute the probability that $\mathcal{B}$ does not abort. If the broadcast encryption scheme is (t, n, qc_1, ϵ_1) CCA-secure, then $\mathcal{B}$ can abort at most ϵ_1. Therefore, the probability that $\mathcal{B}$ dose not abort during the qc_2 credential verification queries is at least $(1-\epsilon_1)^{qc_2}$. Thus, the advantage that $\mathcal{B}$ can break the strongly existential unforgeability of the underlying signature scheme is at least $\epsilon.(1-\epsilon_1)^{qc_2}$ which contradicts the assumption that the underlying signature is $(t, qc_2, \epsilon.(1-\epsilon_1)^{qc_2})$-strongly existentially unforgeable.

Theorem 2. *Our generic construction for DSSO is secure against collusion impersonation attacks if the zero knowledge proofs scheme is secure.*

Proof. Let $\mathcal{A}$ be malicious service providers to whom U has showed credentials and proved the ownership of these credentials. If $\mathcal{A}$ can impersonate U, we will show that there exists an algorithm $\mathcal{B}$ (knowledge extractor) can break the security of the underlying zero knowledge scheme.

If $\mathcal{A}$ can impersonate U to prove that he is the owner of the credentials which U has showed to him, he must execute the ZKP protocol with some service providers to prove that he knows the secret key K_{US} corresponding to the public key K_{UP}. If $\mathcal{A}$ can do this, $\mathcal{B}$ (knowledge extractor) can use the rewinding techniques to obtain the user's secret key K_{US} from two different challenges sent to U and $\mathcal{A}$. So, $\mathcal{B}$ can use $\mathcal{A}$ to break the security of the underlying zero knowledge proofs scheme.

Note that, in our generic construction, the user can not share his credentials with others. Because, if he wants to share one credential with others, he must reveal his secret key K_{US} to them and all credentials of the user will be shared with others. This is the so-called all-or-nothing non-transferability property mentioned at section 4.

Theorem 3. *Our generic construction for DSSO is* $(t, qe, qc_1, qc_2, \epsilon)$ *-secure against coalition credential forging attacks if the broadcast encryption scheme is* (t, n, qc_1, ϵ_1) *CCA-secure, the signature scheme is* $(t, qc_2, \epsilon.(1-\epsilon_1)^{qc_2})$*-strongly existentially unforgeable.*

Proof. Suppose there exists t-time coalition $\mathcal{A}$ that $(t, qe, qc_1, qc_2, \epsilon)$ can forge a credential for the target user, in which the identifiers of the malicious service providers are not included. We will show that there exists an algorithm $\mathcal{B}$ who can $(t, qc_2, \epsilon.(1-\epsilon_1)^{qc_2})$- break the strongly existential unforgeability of the underlying signature scheme.

Init. Algorithm $\mathcal{B}$ runs $\mathcal{A}$ and receives a target user U^* for whom $\mathcal{A}$ wants to forge a credential and a target service provider SP^* that $\mathcal{A}$ wants to attack.

Setup. $\mathcal{B}$ sends the public parameters PP and IdP's public key K_{IP} to $\mathcal{A}$.

Enrollment queries. $\mathcal{A}$ can adaptively issue enrollment queries $\{RI_{U_1}, RI_{U_2}, \cdots, RI_{U_{qe_1}}\}$, where $RI_{U_i} \neq RI_U^*$, $qe_1 \leq (|\mathbb{U}| - 1)$, and $\{RI_{SP_1}, RI_{SP_2}, \cdots, RI_{SP_{qe_2}}\}$, where $RI_{SP_i} \neq RI_{SP}^*$, $qe_2 \leq (|\mathbb{SP}| - 1)$, and $qe_1 + qe_2 = qe$. $\mathcal{B}$ redirects these queries to the challenger. The challenger returns $\{(ID_{U_1}, A_{U_1}), (ID_{U_2}, A_{U_2}) \cdots, (ID_{U_{qe_1}}, A_{U_{qe_1}})\}$ and $\{(ID_{SP_1}, K_{SP_1}), (ID_{SP_2}, K_{SP_2}), \cdots, (ID_{SP_{qe_2}}, K_{SP_{qe_2}})\}$ respectively.

Credential generation queries. $\mathcal{A}$ can adaptively issue credential generation queries $\{(ID_{U_1}, K_{UP_1}), (ID_{U_2}, K_{UP_2}), \cdots, (ID_{U_{qc_1}}, K_{UP_{qc_1}})\}$. $\mathcal{B}$ redirects these queries to the challenger. The challenger returns $\{Cre_{U_1}, Cre_{U_2}, \cdots, Cre_{U_{qc_1}}\}$, where $Cre_{U_i} = (A_{U_i}, Hdr_i, D_i)$, $D_i = E_K(\delta_i, MU_i, ID_{U_i}, K_{UP_i}, T_i)$ and $\delta_i = \mathsf{Sign}(K_{IS}, M_{U_i}, ID_{U_i}, K_{UP_i}, T_{U_i})$.

Credential verification queries. $\mathcal{A}$ can adaptively issue credential verification queries $\{(ID_{U_1}, K_{UP_1}, Cre_{U_1}), (ID_{U_2}, K_{UP_2}, Cre_{U_2}), \cdots, (ID_{U_{qc_2}}, K_{UP_{qc_2}}, Cre_{U_{qc_2}})\}$. $\mathcal{B}$ redirects these queries to the challenger. The challenger returns $True$ or $False$ by decrypting D_i and verifying δ_i.

Output. $\mathcal{A}$ outputs a credential $Cre_U^* = (A_U^*, Hdr^*, D^*)$, where $(ID_U^*, Cre_U^*) \notin \{(ID_{U_1}, Cre_{U_1}), (ID_{U_2}, Cre_{U_2}), \cdots, (ID_{U_{qc_1}}, Cre_{U_{qc_1}})\}$, $ID_{SP}^* \in A_U^*$ and $ID_{SP} \notin A_U^*$ if $SP \in \mathcal{A}$.

$\mathcal{B}$ sends Cre_U^* to SP^*. SP^* returns the corresponding plaintext $(M_U^*, ID_U^*, K_{UP}^*, T^*)$ or $\perp$ for meaningless ciphertext.

1. If $\perp$ is responded, namely D^* is not the corresponding ciphertext of $(\delta^*, M_U^*, ID_U^*, K_{UP}^*, T^*)$ under encryption key K^* encapsulated in Hdr^*, $\mathcal{B}$ aborts. The simulation fails.
2. If the corresponding plaintext $(M_U^*, ID_U^*, K_{UP}^*, T_U^*)$ is responded, namely $\mathcal{A}$ can compute the broadcast encryption key K^* from Hdr^*, $\mathcal{B}$ will abort. $\mathcal{B}$ can use $\mathcal{A}$ to break the broadcast encryption scheme. Due to the assumption that broadcast encryption is (t, n, qc_1, ϵ_1)CCA-secure, the advantage that $(M_U^*, ID_U^*, K_{UP}^*, T_U^*)$ is responded is at most ϵ_1.
3. If $\mathcal{B}$ does not abort, he can obtain a valid signature δ^* on $(M_U^*, ID_U^*, K_{UP}^*, T_U^*)$ at the same advantage ϵ.

Now, we compute the probability that $\mathcal{B}$ does not abort at the qc_2 credential verification queries. Due to the broadcast encryption scheme is (t, n, qc_1, ϵ_1)CCA-secure, the probability of $\mathcal{B}$ aborts when he gets the corresponding ciphertext is at most ϵ_1 for each decryption query. Therefore the probability of $\mathcal{B}$ dose not abort at the qc_2 decryption queries is at least $(1-\epsilon_1)^{qc_2}$. So, the probability that $\mathcal{B}$ can break the strongly existential unforgeability of the underlying signature scheme is at least $\epsilon.(1-\epsilon_1)^{qc_2}$ which contradicts the assumption that the underlying signature scheme is $(t, qc_2, \epsilon.(1-\epsilon_1)^{qc_2})$-strongly existentially unforgeable.

Theorem 4. *Our generic construction for DSSO is (t, qc, ϵ)-forward secure if the broadcast encryption scheme is (t, n, qc, ϵ) CCA-secure.*

Proof. Suppose there exists a t-time malicious service provider $\mathcal{A}$ that (t, qc, ϵ) breaks the forward security of our generic construction for DSSO. We will show there exists an algorithm $\mathcal{B}$ who can (t, n, qc, ϵ) breaks the CCA security of the broadcast encryption scheme. By A_U^O, we denote the access right of U before $\mathcal{A}$'s identifier is listed in it.

Setup. $\mathcal{B}$ sends public parameter PP, and the IdP's public key K_{IP} to $\mathcal{A}$.

Credential verification queries. $\mathcal{A}$ can adaptively issue credential verification queries $\{(ID_{U_1}, K_{UP_1}, Cre_{U_1}),\ (ID_{U_2}, K_{UP_2}, Cre_{U_2}),\ \cdots,\ (ID_{U_{qc}}, K_{UP_{qc}}, Cre_{U_{qc}})$, which are issued after his identifier has been added to A_U. $\mathcal{B}$ redirects these quires to the challenger. The challenger returns $True$ or $False$.

Challenge. $\mathcal{B}$ sends $\mathcal{A}$ an old credential Cre_U^O, where $Cre_U^O = (A_U^O, Hdr_O, D_O)$, $D_O = E_{K_O}(\delta_O, m_U, ID_U, K_{UP}, T_O)$ and $ID_{\mathcal{A}} \notin A_U^O$.

Output. $\mathcal{A}$ outputs a correct verification result $True$ or $Fals$ on credential Cre_U^O at least ϵ.

If it is, $\mathcal{A}$ can decrypt $D_O = E_{K_O}(\delta_O, m_U, ID_U, K_{UP}, T_O)$ at least ϵ. Namely, $\mathcal{A}$ is not a receiver in the broadcast encryption scheme, but can compute the broadcast encryption key K_O from Hdr_O at least ϵ. So, $\mathcal{B}$ can use $\mathcal{A}$ to break the CCA security of the broadcast encryption scheme at least ϵ.

Theorem 5. *Our generic construction for DSSO is (t, qc, ϵ)-backward secure if the broadcast encryption scheme is (t, n, qc, ϵ)CCA-secure.*

The proof is similar to that of theorem 4. We omit the proof due to the page constraint.

6 Conclusion

The current SSO systems suffer from various security issues such as illegally sharing credentials and difficulties in user revocation. In this paper, we formalised the definitions and security models for SSO and DSSO, and proposed a generic scheme of DSSO. Our generic scheme provides a sound solution to these problems. We also provided a formal security proof of our scheme.

Acknowledgement

The first author was supported by PhD scholarships of Smart Services Cooperative Research Centre (CRC) and University of Wollongong.

References

1. Boneh, D., Gentry, C., Waters, B.: Collusion resistant broadcast ecryption with short ciphertexts and private keys. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 258–275. Springer, Heidelberg (2005)

2. Boneh, D., Shen, E., Waters, B.: Strongly unforgeable signatures based on computational diffie-hellman. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 229–240. Springer, Heidelberg (2006)
3. Camenisch, J., Herreweghen, E.V.: Design and Implementation of the idemix Anonymous Credential System. In: Atluri, V. (ed.) ACM CCS 2001, pp. 93–118. ACM, Innsbruck (2001)
4. Camenisch, J., Lysyanskaya, A.: An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 93–118. Springer, Heidelberg (2001)
5. Camenisch, J. and Pfitzmann, B.: Federated identity management. In: Petkovic, M. and Jonker, W. (eds.), Preceedings: Security, Privacy, and Trust in Modern Data Management. Data-Centric Systems and Applications, vol. 2851, pp 213–238. Springer, Heidelberg (2007)
6. Cameron, K.: The laws of identity. Architect of Identity. Microsoft Corporation (2005)
7. Chen, T., Zhu, B.B., Li, S., Cheng, X.: Threspassport-A distributed single sign-on service. In: Huang, D.-S., Zhang, X.-P., Huang, G.-B. (eds.) ICIC 2005. LNCS, vol. 3645, pp. 772–780. Springer, Heidelberg (2005)
8. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Transactions on Information Theory 22(6), 644–654 (1976)
9. Dodis, Y., Fazio, N.: Public key trace and revoke scheme secure against adaptive chosen ciphertext attack. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 100–115. Springer, Heidelberg (2002)
10. Fiat, A., Naor, M.: Broadcast encryption. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 480–491. Springer, Heidelberg (1994)
11. Fiege, U., Fiat, A., Shamir, A.: Zero knowledge proofs of identity. In: ACM STOC 1987, pp. 210–217. ACM, New York (1987)
12. Goldreich, O., Micali, S., Wigderson, A.: Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. Journal of the Association for Comptuing Machinery 38(1), 691–729 (1991)
13. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM Journal on Computing 17(2), 281–308 (1988)
14. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems. In: ACM STOC 1985, pp. 291–304. ACM, Providence (1985)
15. Josephson, W.K., Sirer, E.G., Schneider, F.B.: Peer-to-peer authentication with a distributed single sign-on service. In: Voelker, G.M., Shenker, S. (eds.) IPTPS 2004. LNCS, vol. 3279, pp. 250–258. Springer, Heidelberg (2005)
16. Kormann, D.P., Rubin, A.D.: Risks of the passport single signon protocol. Computer Networks 33(1), 51–58 (2000)
17. Liberty Alliance, `http://www.projectliberty.org`
18. Liberty Alliance. Liberty ID-WSF Authentication Service and Single Sign-On Service Specification Version: v2.0, `http://www.projectliberty.org/liberty/content/download/871/6189/file/liberty-idwsf-authn-svc-v2.0.pdf`
19. Lysyanskaya, A., Rivest, R.L., Sahai, A., Wolf, S.: Pseudonym systems. In: Heys, H.M., Adams, C.M. (eds.) SAC 1999. LNCS, vol. 1758, pp. 184–199. Springer, Heidelberg (2000)
20. OECD. OECD Guidelines on the Protection of Privacy and Transborder Flows of Personal Datal (1980), `http://it.ojp.gov/documents/OECD-FIPs.pdf`

21. OpenID, http://openid.net
22. Oppliger, R.: Microsoft .Net passport: a security analysis. Computer 36(7), 29–35 (2003)
23. Oppliger, R.: Microsoft. Net passport and identity managemen. Information Security Technical Report 9(1), 26–34 (2004)
24. Pashalidis, A., Mitchell, C.J.: A taxonomy of single sign-on systems. In: Safavi-Naini, R., Seberry, J. (eds.) ACISP 2003. LNCS, vol. 2727, pp. 249–265. Springer, Heidelberg (2003)
25. Pashalidis, A., Mitchell, C.J.: Single sign-on using trusted platforms. In: Safavi-Naini, R., Seberry, J. (eds.) ISC 2003. LNCS, vol. 2851, pp. 54–68. Springer, Heidelberg (2003)
26. Pashalidis, A., Mitchell, C.J.: Using GSM/UMTS for single sign-on. In: IEEE SympoTIC 2003, pp. 138–145. IEEE, Bratislava (2003)
27. Perlman, R., Kaufman, C.: User-centric PKI. In: Seamons, K., McBurnett, N., Polk, T. (eds.) IDtrust 2008, pp. 59–71. ACM, Gaithersburg (2008)
28. Rehmant, R.U.: Get Ready for OpenID. Conformix Technologies Inc. (2008)
29. Sahai, A., Waters, B.: Fuzzy Identity-Based Encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
30. Spantzely, A.B., Camenisch, J., Gross, T., Dieter Sommer, D.: User centricity: a taxonomy and open issues. In: ACM DIM 2006, pp. 1–10. ACM, Alexandria (2006)
31. Suriadi, S., Foo, E., Jsang, A.: A user-centric federated single sign-on system. Journal of Network and Computer Applications 32(2), 388–401 (2009)