2013

Secure single sign-on schemes constructed from nominative signatures

Jingquan Wang
University of Wollongong, jw815@uowmail.edu.au

Guilin Wang
University of Wollongong, guilin@uow.edu.au

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

Secure single sign-on schemes constructed from nominative signatures

Abstract

Single Sign-on (SSO) allows users to only log on once and then access different services via automatic authentication by using the same credential. However, most existing SSO schemes do not satisfy security notions or require a high trust level on a trusted third party (TTP), even though SSO has become popular in new distributed systems and computer networks. Motivated by this fact, we formalise a new security model of single sign-on, which not only satisfies strong security notions but also has a low trust level on TTP. We then propose a generic construction of SSO from nominative signatures, and present concrete initialisation. We also provide formal proofs to show that the proposed SSO scheme is secure according to our new formal model, if the underlying nominative signature is secure. We note that this is the first study that investigates the link between SSO and nominative signatures, which also be of an independent interest.

Keywords

secure, sign, schemes, constructed, nominative, signatures, single

Disciplines

Engineering | Science and Technology Studies

Publication Details

Wang, J., Wang, G. & Susilo, W. (2013). Secure single sign-on schemes constructed from nominative signatures. 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom) (pp. 620-627). United States: IEEE.

Secure Single Sign-on Schemes Constructed from Nominative Signatures

Jingquan Wang, Guilin Wang, and Willy Susilo
Center for Computer and Information Security Research
School of Computer Science and Software Engineering
University of Wollongong, Australia
Email: jw815@uowmail.edu.au, {guilin,wsusilo}@uow.edu.au

***Abstract*—Single Sign-on (SSO) allows users to only log on once and then access different services via automatic authentication by using the same credential. However, most existing SSO schemes do not satisfy security notions or require a high trust level on a trusted third party (TTP), even though SSO has become popular in new distributed systems and computer networks. Motivated by this fact, we formalise a new security model of single sign-on, which not only satisfies strong security notions but also has a low trust level on TTP. We then propose a generic construction of SSO from nominative signatures, and present concrete initialisation. We also provide formal proofs to show that the proposed SSO scheme is secure according to our new formal model, if the underlying nominative signature is secure. We note that this is the first study that investigates the link between SSO and nominative signatures, which also be of an independent interest.**

***Keywords*-Single sign-on, Authentication, Nominative signature, Security.**

I. Introduction

The growing number of current internet based services leads more burden for users, because they may have to maintain more and more username/password pairs. Meanwhile, this is also insecure and inefficient with the growth in the number of services. Single sign-on (SSO) is an authentication mechanism that allows users to only log on one time and then access different services via automatic authentication by using the same single credential. This means that SSO may be an effective way to make users relax at their daily work.

Kerberos is one of the earliest single sign-on solutions, proposed by Steiner et al. [16] in 1988, though it is called a network authentication system. The system consists of an Authentication Server, a Ticket Granting Server and a set of service providers. To acquire the Ticket Granting Ticket from a Ticket Granting Server, a user should first go through the authentication with Authentication Server. Actually, Authentication Server and Ticket Granting Server act together as a TTP (trusted third party), or called trusted identity provider [5]), in an SSO scheme. However, not only the process of authentication but also the infrastructure management is very complex. Moreover, unproven symmetric mechanisms are used in Kerberos to authenticate users, which may lead to potential security weaknesses.

Released in 2005, OpenID [12] is one of the most successful single sign-on solutions, which is an open and decentralised approach for authenticating users. In OpenID, the user can freely select the identity providers from any web based application where he has registered with. Before signing on to a given web based application that supports OpenID, the user first signs on to the identity provider and OpenID exchanges the necessary authentication data between the identity provider and the application. However, the mechanism of exchanging data is complex and can be attacked through network based techniques [2].

In 2010, Han et al. [5] proposed a novel dynamic SSO model together with a generic scheme. This scheme employs a digital signature scheme to guarantee both the unforgeability and the public verification of credential. In addition, a broadcast encryption is used to protect the privacy of credential, which means that except the authorised service providers nobody can check the validity of a credential. After the credential verification the user should run zero-knowledge proofs to prove that he/she is legal to use the valid credential, for resisting against impersonation attacks. However, the broadcast encryption is a complex and rather inefficient. In 2012, Yu et al. [20] proposed a single sign-on model with key exchange, and the advantage is that each user does not need hold a public/private key pair, while this is required in Han et al.'s model. However, the trust level of TTP is higher than that in Han et al.'s model, because in Yu et al.'s model the TTP is assumed to not impersonate any user, which is rather unrealistic.

Unfortunately, most of existing SSO systems have shortcomings. For example, in some SSO systems-like [3], when the user wants to access a service by using a credential, the service provider (SP) has to directly communicate with the TTP because SP cannot verify the validity of credential. Another drawback is that some systems [9], [13], [14] are fragile to resist single point of failure, as the TTP is required to be always online. Moreover, SSO scheme proposed in [16] does not prevent illegal usage of a personal credential, since an illegal user can access services if he obtains a legal user's valid credential. Furthermore, SSO systems given in [5], [20] require high trust level on the TTP. Based on these shortcomings, we are motivated to study single sign-on (SSO) in this paper by proposing a formal model and

giving a construction from nominative signatures.

Classical digital signatures have been widely used to provide integrity, authenticity, and non-repudiation for authenticating electronic messages. However, sometimes the users may not want their signatures being publicly verifiable. Nominative signature is such a primitive supporting no public verification. It consists of three parties: nominator, nominee and verifier. The nominator and nominee jointly generate a signature but only the nominee can verify it. Moreover, the validity of a nominative signature can only be determined by the help of the nominee, while any body else (including the nominator) will not be able to show the validity of the nominative signature to a verifier.

In 1996, Kim et al. [7] proposed the first notion and construction of nominative signature. However, it was found that the nominator can also verify the signature [6]. Then, Huang and Wang [6] proposed a new convertible nominative scheme, in which the nominee can convert a nominative signature into a publicly verifiable one when necessary. Unfortunately, in [18], [4], [17], it was found that the nominator in Huang-Wang scheme can generate valid signatures alone and show the validity of the signature to anyone without the consent of the nominee. In 2007, Liu et al. [10] proposed a formal definition and a rigorous set of adversarial models for nominative signature, which is based on Chaums undeniable signature [1] and a strongly unforgeable signature scheme. In 2011, Schuldt and Hanaoka [15] proposed a stronger security model of nominative signature with a provably secure scheme.

Our Contributions. In this paper, we aim to formalise the notion of single sign-on (SSO). This notion is motivated by the need of strong security requirements and low trust level of TTP in a single sign-on system. Then, we construct a generic SSO scheme from nominative signatures and prove that this SSO is secure under the proposed security model, by assuming that the underlying nominative signature is secure according to the model introduced by Schuldt and Hanaoka [15]. This means that we establish a relation between a cryptographic primitive and a security application: nominative signatures are a sufficient tool to construct SSO with low trust level of TTP. In addition, a concrete intialisation of the generic SSO is also presented.

Paper Organization. In Section 2, we propose the formal definition and security notions for SSO. We review the formal definition and security notions of nominative signature in Section 3. Section 4 proposes a generic construction of SSO transformed from nominative signatures. In Section 5, a concrete initialisation is described. In Section 6, we provide formal security analysis of the proposed SSO scheme. In Section 7, we give some discussions and a short conclusion.

II. Formal Model of Single Sign-On

In this section, we provide a formal security model for single sign-on (SSO) model, which specifies both the syntax and security properties of SSO.

A. Syntax of Single Sign-on

In a SSO model, a user (U) obtains a single credential from a trusted third party (TTP) once and then authenticates himself to different service providers (SPs) by using the same credential. Now we formalise the syntax of SSO as follows by specifying its components:

Definition 1. *A Single Sign-on scheme comprises a trusted third party TTP, a group of service providers SPs and a group of users U. It consists of four algorithms and two protocols: system setup algorithm $Setup$, user key generation algorithm $UKGen$, enrollment algorithm $Enrol$, credential generation protocol $CreGen$, credential verification algorithm $CreVer$, and confirmation/disavowal protocol Con/Dis.*

- *$Setup(k)$: On input 1^k where $k \in \mathbb{N}$ is a security parameter, it returns a list of public parameters par and a public-private key pair (pk, sk) of TTP.*
- *$UKGen(par)$: On input public parameters par, this probabilistic algorithm returns a public-private key pair $(pk_i, sk_i) \leftarrow g(1^k)$ for a user.*
- *$Enrol(par, RI)$: On input public parameters par, a service provider $SP_j's$ registration information RI_j which includes $SP_j's$ username and some other information, or a user $U_i's$ registration information RI_i which includes $U_i's$ username, his public key pk_i and some other information, TTP executes Enrol (par, RI) and returns ID_j to SP_j, or $ID_i = (U_i, pk_i, AU_i)$ to U_i, where ID_j and ID_i are the identities of SP_j and U_i, AU_i is $U_i's$ access right which is a set consisting of the identities of the service providers that the user can access to.*
- *$CreGen(pk, pk_i, ID_i, par)[TTP(sk), U_i(sk_i)]$: The public input are TTP's public key pk, a user U_i's public key pk_i, U_i's identity ID_i, and public parameters par. TTP holds its private key sk, and U_i holds his private key sk_i. This algorithm outputs a credential Cre_i for user U_i.*
- *$CreVer(Cre_i, sk_i, pk, par)$: On input a user U_i's credential Cre_i, U_i's private key sk_i, TTP's public key pk and public parameters par, it returns if Cre_i is valid or invalid.*
- *$Con/Dis(Cre_i, pk_i, pk, par)[U_i(sk_i), SP_j]$: The public input are a user U_i's credential Cre_i and his public key pk_i, TTP's public key pk and public parameters par, while U_i's private input is his private key sk_i. If valid $\leftarrow CreVer(Cre_i, sk_i, pk, par)$, the confirmation protocol is carried out; otherwise, the disavowal protocol is carried out. If the confirmation protocol runs successfully and $SP_j's$ identity $ID_j \in AU_i$, SP_j accepts U_i's authentication request; otherwise it rejects.*

Remark 1. *Compared to the formal models given in [5], [20], our model not only satisfies the strong security notions of SSO, but also reduces the trust level of TTP, because TTP can neither generate the credential without the user's involvement nor impersonate the user by running Con protocol. Moreover, different from [5] no broadcast encryption is required here, so our model is simpler and more efficient.*

B. Security Definitions of Single Sign-On

Security requirements are crucial for a security system. In an SSO scheme, the TTP most concerns about whether others can generate the credential without its involvement. Users mainly worries about whether others who obtains his credential can access to SPs by impersonating him. The service providers most concern about whether their sevices can be accessed without holding a valid credential. Therefore, in the following we propose and formalize three security notions which should be considered. Namely, unforgeability, security against impersonation, and soundness. In order to formalise each security notion for SSO, we will define a series of games between two Turing machines: Challenger $\mathcal{C}$ and Adversary $\mathcal{A}$.

1) Unforgeability: Intuitively, unforgeability means that any adversary $\mathcal{A}$ should not be able to forge a credential of a user U_i if either the TTP's private key sk or the $U_i's$ private key sk_i is unknown. Formally, we say that a SSO scheme is unforgeable, if no polynomially bounded adversary $\mathcal{A}$ has a non-negligible advantage against the challenger $\mathcal{C}$ in the following two games:

Game 1. ***Unforgeability I***

- *Setup. $\mathcal{C}$ runs $Setup(k)$ to generate the public parameter par and the public-private key pair (pk, sk) of TTP. It sends $\mathcal{A}$ the public parameter par and the public key pk of TTP.*
- *Create user oracle. $\mathcal{A}$ can adaptively enroll user U_i. $\mathcal{A}$ generates the public-private key pair (pk_i, sk_i) of U_i by running $UKGen$ algorithm and sends the registration information RI_i to $\mathcal{C}$. $\mathcal{C}$ returns $ID_i = (U_i, pk_i, AU_i)$ to $\mathcal{A}$.*
- *Credential generation oracle. $\mathcal{A}$ can adaptively run the protocol $CreGen$ by using ID_i with $\mathcal{C}$. If the protocol runs successfully, $\mathcal{A}$ will receive a valid credential Cre_i.*

Finally, $\mathcal{A}$ outputs (ID^*, Cre^*), where $ID^* = (U^*, pk^*, AU^*)$. $\mathcal{A}$ wins the game if

1) $Valid \leftarrow CreVer(Cre^*, sk^*, pk, par)$.
2) *ID^* has not been queried to Credential generation oracle.*

$\mathcal{A}$'s advantage is defined to be the probability that $\mathcal{A}$ wins.

Remark 2. *In this game, $\mathcal{A}$ can verify the credential by himself after he corrupted the pk_i. Therefore, verification oracle is not necessary.*

Remark 3. *In this game, Con/Dis execution oracles are also not provided. The reason is that as the service provider SP does not hold any private parameter and $\mathcal{A}$ can corrupt all the users, $\mathcal{A}$ can run the protocols Con or Dis alone by playing both roles of SP_j and U_i.*

Game 2. ***Unforgeability II***

- *Init. $\mathcal{C}$ sends a target $ID^* = (U^*, pk^*, AU^*)$ to $\mathcal{A}$.*
- *Setup. $\mathcal{C}$ runs $Setup(k)$ to generate the public parameter par and then sends par to $\mathcal{A}$. $\mathcal{A}$ generates the public-private key pair (pk, sk) of TTP.*
- *Create user oracle. $\mathcal{A}$ can adaptively enroll user U_i. $\mathcal{A}$ generates the public-private key pair (pk_i, sk_i) of U_i by running $UKGen$ algorithm and sends the registration information RI_i to $\mathcal{C}$. $\mathcal{C}$ returns $ID_i = (U_i, pk_i, AU_i)$ to $\mathcal{A}$.*

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

1) $Valid \leftarrow CreVer(Cre^*, sk^*, pk, par)$.
2) *$\mathcal{A}$ has not obtain sk^*.*

$\mathcal{A}$'s advantage is defined to be the probability that $\mathcal{A}$ wins.

Definition 2. *A Single Sign-on system is said to be unforgeable if no PPT (Probabilistic Polynomial Time) adversary has a non-negligible advantage in either **Game 1** or **Game 2**.*

2) Security against impersonation: Security against impersonation means that even though the adversary $\mathcal{A}$ has obtained the credential of a user U_i, $\mathcal{A}$ should not be able to impersonate U_i to access to SPs by executing SSO protocols.

Formally, we say that a SSO scheme is secure against impersonation, if no polynomially bounded adversary $\mathcal{A}$ has a non-negligible advantage against the challenger $\mathcal{C}$ in the following game:

Game 3. ***Security against impersonation***

- *Init. $\mathcal{C}$ sends a target U^* with ID^* and corresponding credential Cre^* to $\mathcal{A}$.*
- *Setup. $\mathcal{C}$ runs $Setup(k)$ to generate the public parameter par and sends to $\mathcal{A}$. $\mathcal{A}$ generates the public-private key pair (pk, sk) of TTP.*
- *Create user oracle. $\mathcal{A}$ can adaptively enroll user U_i. $\mathcal{A}$ generates the public-private key pair (pk_i, sk_i) of U_i by running $UKGen$ algorithm and sends the registration information RI_i to $\mathcal{C}$. $\mathcal{C}$ returns $ID_i = (U_i, pk_i, AU_i)$ to $\mathcal{A}$.*
- *Confirmation oracle. There are two cases:*
  1) *$\mathcal{A}$ (as SP_j) can adaptively run the protocol Con with $\mathcal{C}$ (as U^*) using Cre^*. It returns accept or reject.*
  2) *$\mathcal{A}$ (as eavesdropper) can adaptively eavesdrop the conversions when U^* using Cre^* runs protocol Con or Dis with $\mathcal{C}$ (as SP_j). This oracle returns the accept or reject and the transcript of the protocol.*

Finally, $\mathcal{A}$ has to run confirmation protocol Con with a SP_j. $\mathcal{A}$ wins the game if

1) *SP_j returns accept.*
2) *$\mathcal{A}$ has not obtain sk^*.*

$\mathcal{A}$'s advantage is defined to be the probability that $\mathcal{A}$ wins the game.

Definition 3. *A Single Sign-on system is said to be secure against impersonation if no PPT adversary has a non-negligible advantage in **Game 3**.*

3) Soundness: Soundness means that any adversary $\mathcal{A}$ should not be able to access to SPs if the TTP does not generate a corresponding credential and sends it to $\mathcal{A}$.

Formally, we say that a SSO scheme is soundness, if no polynomially bounded adversary $\mathcal{A}$ has a non-negligible advantage against the challenger $\mathcal{C}$ in the following game:

Game 4. ***Soundness***

- *Setup. $\mathcal{C}$ runs $Setup(k)$ to generate the public parameter par and the public-private key pair (pk, sk) of TTP. It sends $\mathcal{A}$ the public parameter par and the public key pk of TTP.*
- *Create user oracle. $\mathcal{A}$ can adaptively enroll user U_i. $\mathcal{A}$ generates the public-private key pair (pk_i, sk_i) of U_i by running $UKGen$ algorithm and sends the registration information RI_i to $\mathcal{C}$. $\mathcal{C}$ returns $ID_i = (U_i, pk_i, AU_i)$ to $\mathcal{A}$.*
- *Credential generation oracle. $\mathcal{A}$ can adaptively run the protocol $CreGen$ by using ID_i with $\mathcal{C}$. If the protocol runs successfully, $\mathcal{A}$ outputs a Cre_i.*

Finally, $\mathcal{A}$ has to select a target U^* with ID^* and then runs protocol Con with SP_j. $\mathcal{A}$ wins the game if

1) *SP_j returns accept.*
2) *ID^* has not been queried to Credential generation oracle.*

$\mathcal{A}$'s advantage is defined to be the probability that $\mathcal{A}$ wins.

Remark 4. *In this game, $\mathcal{A}$ may have two ways to win the game: Forging a credential or successfully running protocol Con without a valid credential. The case 1 has been discussed in **Game 1** and **Game 2**. Therefore, this game focuses on the case 2.*

Definition 4. *A Single Sign-on system is said to be sound if no PPT adversary has a non-negligible advantage in **Game 4**.*

III. Nominative Signatures

A. Syntax of Nominative Signatures

In this section, we describe one nominative signature scheme which was proposed in [15]. A nominative signature scheme involves a signer S and a nominee N, and is specified by the algorithms described below.

- $Setup$: The input is a security parameter 1^k, it outputs a set of public parameters par.
- $KeyGen_S$, $KeyGen_N$: The input is the public parameters par, it outputs a public-private signer and nominee key pair, (pk_S, sk_S) and (pk_N, sk_N), respectively.
- $Sign$: The inputs are par, pk_N, a message m, and sk_S, it outputs a signature generation message δ.
- $Receive$: The inputs are par, pk_S, m, a signature generation message δ, and sk_N, it outputs a nominative signature σ on m.
- $Convert$: The inputs are par, pk_S, m, σ, and sk_N, it outputs a verification token tk_σ.
- $TkVerify$: The inputs are par, pk_S, pk_N, m, σ, and tk_σ, it outputs either accept or reject.
- (Confirm, V_C): this is a pair of interactive algorithms with common input (par, pk_S, pk_N, m, σ). The algorithm Confirm is furthermore given sk_N as private input. At the end of the interaction, V_C will either output accept or reject.
- (Disavow, V_D): like in the confirm protocol, this is a pair of interactive algorithms with the common input (par, pk_S, pk_N, m, σ), and Disavow is given sk_N as private input. At the end of the interaction, V_D will either output accept or reject.
- $Valid$: The inputs are par, pk_S, pk_N, m, σ, sk_N, this algorithm computes the verification token $tk_\sigma \leftarrow Convert(par, pk_S, \text{m}, \sigma, sk_N)$ and it outputs the result of $TkVerify(par, pk_S, pk_N, \text{m}, \sigma, tk_\sigma)$.

B. Security Notions for Nominative Signatures

For a nominative signature scheme to be secure [15], the scheme should be unforgeable, invisible, and secure against malicious signers. Additionally, the confirm and disavow protocols are required to be zero-knowledge proofs.

1) Unforgeability: Unforgeability requires that it can not generate a valid signatures without the signer i.e. a malicious nominee should not be able to produce a signature on a message m, and then convince a verifier that the signature is valid, either by running the confirm protocol or by presenting a verification token, without having requested a signature on m from the signer.

Formally, we define unforgeability against a chosen message attack (uf-cma) of a nominative signature scheme $\mathcal{NS}$ via the experiment $Exp^{uf-cma}_{\mathcal{NS},\mathcal{A}}$ shown in Figure 1. By x $\leftarrow \mathcal{A}^{\mathcal{O}}$ (y) it means that the algorithm $\mathcal{A}$ is executed on input y while being allowed to make queries to the oracle $\mathcal{O}$, and that the output of $\mathcal{A}$ is assigned to x. For a pair of interactive algorithms, $\mathcal{A}$ and V, it writes z $\leftarrow_2$ { $\mathcal{A}$ (x_1) $\leftrightarrow$ V (x_2)} (y) to mean that $\mathcal{A}$ and V interact with common input y and private inputs x_1 and x_2 to $\mathcal{A}$ and V, respectively, and that the output of V, upon the completion of the interaction, is assigned to the variable z. Lastly, it will use the abbreviation PPT algorithm to mean a probabilistic

$\mathbf{Exp}^{uf-cma}_{\mathcal{NS},\mathcal{A}}(1^k)$
$L_S \leftarrow \{\}$
$par \leftarrow Setup(1^k)$
$(pk_S, sk_S) \leftarrow KenGen_S(par)$
$(pk^*_N, m^*, \sigma^*, tk^*_\delta, st) \leftarrow \mathcal{A}^{\mathcal{O}}(par, pk_S)$
$z \leftarrow_2 \{ \mathcal{A}\text{ (st)} \leftrightarrow V_C(par, pk_S, pk^*_N, m^*, \sigma^*)\}$
$z' \leftarrow TkVerify(par, pk_S, pk^*_N, m^*, \sigma^*, tk^*_\delta)$
if $(pk^*_N, m^*) \notin L_S \wedge$ (z = accept $\vee$ z' = accept)
output 1
else output 0

Figure 1. Unforgeability security experiment

$\mathbf{Exp}^{mal-sig}_{\mathcal{NS},\mathcal{A}}(1^k)$
$L_R \leftarrow \{\}$; $L_C \leftarrow \{\}$
$par \leftarrow Setup\ (1^k)$
$(pk_N, sk_N) \leftarrow KenGen_N(par)$
$(pk^*_S, m^*, \sigma^*, tk^*_\delta, st) \leftarrow \mathcal{A}^{\mathcal{O}}(par, pk_N)$
$z \leftarrow Valid(par, pk^*_S, pk_N, m^*, \sigma^*, sk_N)$
$z' \leftarrow TkVerify(par, pk^*_S, pk_N, m^*, \sigma^*, tk^*_\delta)$
if z = accept
$z'' \leftarrow_2 \{\mathcal{A}\text{(st)} \leftrightarrow V_C(par, pk^*_S, pk_N, m^*, \sigma^*)\}$
else
$z'' \leftarrow_2 \{\mathcal{A}\text{(st)} \leftrightarrow V_D(par, pk^*_S, pk_N, m^*, \sigma^*)\}$
if (z = accept $\wedge (pk^*_S, m^*, \sigma^*) \notin L_R$) $\vee$
(z' = accept $\wedge (pk^*_S, m^*, \sigma^*) \notin L_C$) $\vee$
(z'' = accept)
output 1
else output 0

Figure 2. Malicious signer security experiment

polynomial time algorithm. In the experiment, $\mathcal{A}$ has access to the oracle $\mathcal{O} = \{\mathcal{O}_{Sign}\}$ defined as follows:

- $\mathcal{O}_{Sign}$: The inputs are pk_N and m, this oracle computes $\delta \leftarrow Sign(par,\ pk_N,\ \text{m},\ sk_S)$, adds ($pk_N$, m) to L_S, and returns the signature generation message δ to $\mathcal{A}$.

Definition 5. *A nominative signature scheme $\mathcal{NS}$ is said to be uf-cma secure if all PPT adversaries $\mathcal{A}$ have advantage $Adv^{uf-cma}_{\mathcal{NS},\mathcal{A}} = Pr[Exp^{uf-cma}_{\mathcal{NS},\mathcal{A}}(1^k) = 1]$ which is negligible in k .*

2) Security against Malicious Signers: Our definition of security against malicious signers requires that any adversary with the knowledge of the private signer key should not be able to (1) produce a new valid nominative signature associated to the nominee, (2) convince a verifier about the validity or invalidity of a signature through the confirm or disavow protocols, regardless of the signature being valid or not, or (3) produce an accepting verification token for a signature he has not previously seen a verification token for.

Formally, Security against malicious signers of a nominative signature scheme $\mathcal{NS}$ is defined via the experiment $Exp^{mal-sig}_{\mathcal{NS},\mathcal{A}}$ shown in Figure 2. In the experiment, $\mathcal{A}$ has

$\mathbf{Exp}^{inv-cma}_{\mathcal{NS},\mathcal{A}}(1^k)$
$par \leftarrow Setup(1^k)$
$(pk^*_N, sk^*_N) \leftarrow KenGen_N(par)$
$(pk^*_S, m^*, \delta^*, st) \leftarrow \mathcal{A}^{\mathcal{O}}(par, pk^*_N)$
$b \leftarrow \{0, 1\}$
if b = 0
$\sigma^* \leftarrow Receive(par, pk^*_S, m^*, \delta^*, sk_N)$
else (b = 1)
$(pk_N, sk_N) \leftarrow KenGen_N(par)$
$(pk_S, sk_S) \leftarrow KenGen_S(par)$
$m \leftarrow \mathcal{M}(par)$
$\delta \leftarrow Sign(par, pk_N, m, sk_S)$
$\sigma^* \leftarrow Receive(par, pk_S, m, \delta, sk_N)$
$b' \leftarrow \mathcal{A}^{\mathcal{O}}(st, \sigma^*)$
if $b = b'$ output 1
else output 0

Figure 3. Invisibility security experiment

access to the oracles $\mathcal{O} = \{\mathcal{O}_{Receive},\ \mathcal{O}_{Convert},\ \mathcal{O}_{Con},\ \mathcal{O}_{Dis}\}$ defined as follows:

- $\mathcal{O}_{Receive}$: The inputs are pk_S, m and δ, this oracle computes $\sigma \leftarrow Receive(par,\ pk_S,\ \text{m},\ \delta,\ sk_N)$, adds the tuple ($pk_S$, m, σ) to the list L_R, and returns σ to $\mathcal{A}$.
- $\mathcal{O}_{Convert}$: The inputs are pk_S, m and σ, this oracle adds the tuple (pk_S, m, σ) to L_C and returns the verification token $tk_\sigma \leftarrow Convert(par,\ pk_S,\ \text{m},\ \sigma,\ sk_N)$.
- $\mathcal{O}_{Con}$: The inputs are pk_S, m and σ, this oracle interacts with $\mathcal{A}$ by running Confirm with the common input (par, pk_S, pk_N, m, σ) and the private input sk_N.
- $\mathcal{O}_{Dis}$: The inputs are pk_S, m and σ, this oracle interacts with $\mathcal{A}$ by running Disavow with the common input (par, pk_S, pk_N, m, σ) and private input sk_N.

Definition 6. *A nominative signature scheme $\mathcal{NS}$ is said to be mal-sig secure if all PPT adversaries $\mathcal{A}$ have advantage $Adv^{mal-sig}_{\mathcal{NS},\mathcal{A}} = Pr[Exp^{mal-sig}_{\mathcal{NS},\mathcal{A}}(1^k) = 1]$ which is negligible in k .*

3) Invisibility: To ensure that no information leaked from the signer will reveal the validity of a signature, invisibility requires that an adversary with the knowledge of the private signer key, cannot distinguish between a valid signature, and a random element of the signature space.

Formally, we define invisibility under a chosen message attack (inv-cma) of a nominative signature scheme $\mathcal{NS}$ via the experiment $Exp^{inv-cma}_{\mathcal{NS},\mathcal{A}}$ shown in Figure 3 where $\mathcal{M}$ (par) is the message space defined by par. In the experiment, A has access to the oracles $\mathcal{O} = \{\mathcal{O}_{Receive},\ \mathcal{O}_{Convert},\ \mathcal{O}_{Con},\ \mathcal{O}_{Dis}\}$ defined as above.

Definition 7. *A nominative signature scheme $\mathcal{NS}$ is said to be inv-cma secure if all PPT adversaries $\mathcal{A}$ have advantage*

$Adv^{inv-cma}_{\mathcal{NS},\mathcal{A}}$ = $Pr[Exp^{inv-cma}_{\mathcal{NS},\mathcal{A}}(1^k) = 1]$- *1/2 which is negligible in k .*

4) Protocol Security: Lastly, we require the confirm and disavow protocols to be zero-knowledge proofs. More specifically, consider the languages $L(par)$ and $\bar{L}(par)$ parameterized by par and defined by

$$L(par) = \{(pk_S, pk_N, m, \sigma) : \exists sk_N s.t.(pk_N, sk_N) \in \{KeyGen_N(par)\} \wedge Valid(par, pk_S, m, \sigma, sk_N) = accept\}$$

$$\bar{L}(par) = \{(pk_S, pk_N, m, \sigma) : \exists sk_N s.t.(pk_N, sk_N) \in \{KeyGen_N(par)\} \wedge Valid(par, pk_S, m, \sigma, sk_N) = reject\}$$

The confirm protocols is required to be a zero-knowledge proof of membership for L, whereas the disavow protocol is required to be a zero-knowledge proof of membership for $\bar{L}$.

IV. Generic Construction

This section describes how a nominative signature scheme can be directly transformed into an SSO scheme, where a nominative signature is employed as a user's credential.

1) Setup. Runs the $Setup(k)$ to generate the public parameters par, which includes all public parameters in this scheme, and runs $KeyGen_S(par)$ to generate the public-private key pair (pk, sk) of TTP.
2) Enrollment.
   - Service providers enrollment. SP_j submits registration information RI_j which includes the $SP_j's$ username and some other information to TTP. TTP issues an identity ID_j to SP_j, and stores (SP_j, ID_j) for SP_j.
   - User enrollment. U_i generates his public-private key pair (pk_i, sk_i) by running $KeyGen_N(par)$. U_i sends his registration information RI_i which includes the $U_i's$ username, public key pk_i and some other information to the TTP. The TTP issues an identity $ID_i = (U_i, pk_i, AU_i)$ to U_i, where AU_i is a set that consists of the identities of the service providers that the user can access, and stores (U_i, ID_i) for U_i.
3) Single Sign-on.
   - Login. U_i uses his username and corresponding password to log in the system.
   - Credential generation. U_i submits his ID_i and TTP checks it. If ID_i has been registered, TTP executes $Sign(par, pk_i, ID_i, sk)$ and sends a signature generation message δ_i to U_i. U_i executes $Receive(par, pk, ID_i, \delta_i, sk_i)$ and outputs a nominative signature σ_i. The credential of U_i is $Cre_i = (ID_i, \sigma_i)$.
   - Service request. U_i sends a service request to the service provider $SP_j (ID_j \in AU_i)$.
   - Verification request. SP_j asks U_i to show his credential to him.
   - Credential verification. U_i verifies the Cre_i by executing $Valid(par, pk, pk_i, ID_i, \sigma_i, sk_i)$. If it returns accept, then do the next step.
   - Confirmation. U_i sends Cre_i to SP_j, the protocol $Confirm(par, pk_i, pk, ID_i, \sigma_i)[U(sk_i), SP_j]$ is carried out. If the confirmation protocol runs successfully, SP_j executes the next step. Otherwise SP_j aborts.
   - Service grant. If SP_j's identity $ID_j \in AU_i$, SP_j grants the services to the user. Otherwise, SP_j rejects the services.

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

V. Instantiation

Now, we present a concrete SSO scheme by instantiating the above generic SSO with the concrete nominative signature scheme proposed by Schuldt and Hanaoka in [15], while their scheme is inspired by [19], [11].

1) $Setup(k)$: given 1^k, choose a bilinear map e: $\mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ where $|\mathbb{G}_1|$ = p, and a generator $\langle g \rangle = \mathbb{G}_1$. Lastly pick a collision resistant hash function H: $\{0,1\}^* \rightarrow \mathbb{Z}_p$ and return $par \leftarrow (e, p, g, H)$. Given par, pick $\alpha_S, v_0, ..., v_n \leftarrow \mathbb{Z}_p$ and $h_S \leftarrow \mathbb{G}_1$, and compute $g_S \leftarrow g^{\alpha_S}$ and $u_i \leftarrow g^{v_i}$ for 1 $\leq$ i $\leq$ n. Furthermore, for m $\in \{0,1\}^n$, define $F_S(m)$ = $u_0 \Pi_{i=1}^n u_i^{m_i}$ where m_i is the i-th bit of m, and finally set TTP's public-private key pair (pk, sk) as pk $\leftarrow (g_S, h_S, u_0, ..., u_n)$ and sk $\leftarrow \alpha_S$.
2) $Enrol(par, RI)$:
   - Service providers enrollment: given par, $SP_j's$ registration information RI_j which includes $SP_j's$ username and some other information. TTP issues an identity ID_j to SP_j, and stores (SP_j, ID_j) for SP_j.
   - Users enrollment: given par, pick α_N, y_1, y_2, v_0,..., $v_n \leftarrow \mathbb{Z}_p$ and h_N, $k \leftarrow \mathbb{G}_1$, and compute $g_N \leftarrow g^{\alpha_N}$. Furthermore, for m $\in \{0,1\}^n$, define $F_N(m)$ = u_0, $\Pi_{i=1}^n u_i^{m_i}$ where m_i is the i-th bit of m. Lastly compute $x_1 \leftarrow g^{y_1^{-1}}$ and $x_2 \leftarrow g^{y_2^{-1}}$, and finally $U_i's$ generates his public-private key pair (pk_i, sk_i) as $pk_i \leftarrow (g_N, h_N, k, u_0, ..., u_n, x_1, x_2)$ and $sk_i \leftarrow (\alpha_S, v_0, ..., v_n, y_1, y_2)$. Given par, U_i registration information RI_i which includes $U_i's$ username, public key pk_i and some other information. TTP issues an identity $ID_i = (U_i, pk_i, AU_i)$ to U_i, where AU_i is a set that consists of the identities of the service providers that the user can access, and stores (U_i, ID_i) for U_i.

3) $CreGen(pk, pk_i, ID_i, par)[TTP(sk), U_i(sk_i)]$: Round 1: given par, pk_i, ID_i, and sk, pick r $\leftarrow \mathbb{Z}_p$, and compute $\delta_1 \leftarrow g^r$ and $\delta_2 \leftarrow h_S^{\alpha_S} F_S(pk_i||ID_i)^r$. Lastly TTP returns $\delta \leftarrow (\delta_1, \delta_2)$ to U_i. Round 2: given par, pk, ID_i, δ and sk_i, firstly check that $e(g_S, h_S)e(\delta_1, F_S(pk_i||ID_i)) = e(g, \delta_2)$ holds. If this is not the case, return $\perp$. Otherwise, pick r, r', s $\leftarrow \mathbb{Z}_p$ and re-randomize δ by computing $\delta_1' \leftarrow \delta_1 g^{r'}$ and $\delta_2' \leftarrow \delta_2 F_S(pk_i||ID_i)^{r'}$.Then set $\sigma_1 \leftarrow (\delta_1'/g^r)^{y_1^{-1}}$ and $\sigma_2 \leftarrow (g^r)^{y_2^{-1}}$, and compute $t \leftarrow H(\sigma_1||\sigma_2||pk||ID_i)$ and $M \leftarrow g^t k^s$. Finally set $\sigma_3 \leftarrow \delta_2' h_N^{\alpha_N} (\delta_1')^{v_0 + \Sigma_{i=1}^n v_i M_i}$, where M_i is the i-th bit of M, and U_i outputs the signature $\sigma_i \leftarrow (\sigma_1, \sigma_2, \sigma_3, s)$ and the credential $Cre_i = (ID_i, \sigma_i)$.
4) $CreVer(Cre_i, sk_i, pk, par)$: given par, pk, Cre_i = (ID_i, σ_i) and sk_i, first check that the equation $e(g, \sigma_3) = e(g_S, h_S)e(g_N, h_N)e(\sigma_1^{y_1}, \sigma_2^{y_2}, F_S(pk_i||ID_i)F_N(M))$ hold, where M = $g^t k^s$, and t = H($\sigma_1||\sigma_2||pk||ID_i$), and if this is not the case, output $\perp$. Otherwise, return the verification token $tk_\sigma \leftarrow (\sigma_1^{y_1}, \sigma_2^{y_2})$. Given par, pk, pk_i, Cre_i = (ID_i, σ_i) and tk_σ = (tk_1, tk_2), output valid if the equations $e(\sigma_1, g) = e(tk_1, x_1)$, $e(\sigma_2, g) = e(tk_2, x_2)$, and $e(g, \sigma_3) = e(g_S, h_S)e(g_N, h_N)e(\sigma_1^{y_1}, \sigma_2^{y_2}, F_S(pk_i||ID_i)F_N(M))$ hold, where M = $g^t k^s$, and t = H($\sigma_1||\sigma_2||pk||ID_i$). Otherwise, output invalid.
5) Con $(Cre_i, pk_i, pk, par)[U_i(sk_i), SP_j]$: given ($par$, pk, pk_i, Cre_i) as common input and sk_i as the private input to U_i, let $e_1 = e(g, \sigma_3)$, $e_2 = e(g_S, h_S)e(g_N, h_N)$, $e_3 = e(\sigma_1, F_S(pk_i||ID_i)F_N(M))$ and finally let $e_4 = e(\sigma_2, F_S(pk_i||ID_i)F_N(M))$ where M = $g^t k^s$, and t = H ($\sigma_1||\sigma_2||pk||ID_i$). Then U_i and SP_j interacts in the protocol

$$ZKPK\{(y_1, y_2) : x_1^{y_1} = g \wedge x_2^{y_2} = g \wedge e_1 = e_2 e_3^{y_1} e_4^{y_2}\}$$

If the protocol runs successfully, SP_j returns accept. Otherwise, SP_j returns reject.

VI. Security Analysis

Theorem 1. *The SSO scheme proposed above is unforgeable if the nominative signature scheme used above is unforgeable and secure against malicious signers.*

Proof: Suppose there exists an adversary $\mathcal{A}$ which can break the unforgeability of our generic construction for SSO. We will show that there exists an adversary $\mathcal{B}$ which can break the unforgeability or the security against malicious signers of above nominative signature scheme. There are two cases for unforgeability.

Unforgeability I

- Init. $\mathcal{B}$ receives the public parameter par and a public key pk_S, and $\mathcal{B}$ runs $\mathcal{A}$. $\mathcal{B}$ simulates an SSO scheme by setting pk_S as the public key of TTP and par as the public parameter. Then $\mathcal{B}$ sends (par, pk) to $\mathcal{A}$.
- Create user oracle. $\mathcal{A}$ can adaptively enroll user U_i. $\mathcal{A}$ generates the public-private key pair (pk_i, sk_i) of U_i by running $UKGen$ algorithm and sends the registration information RI_i to $\mathcal{B}$. $\mathcal{B}$ returns $ID_i = (U_i, pk_i, AU_i)$ to $\mathcal{A}$.
- Credential generation oracle. $\mathcal{A}$ can adaptively run the protocol $CreGen$ by using ID_i with $\mathcal{B}$. $\mathcal{B}$ redirects these queries to $\mathcal{NS}$ model's oracle $\mathcal{O}_{Sign}$. If the protocol runs successfully, $\mathcal{A}$ outputs a Cre_i = (ID_i, σ_i).

Finally, $\mathcal{A}$ outputs (ID^*, Cre^*), where $ID^* = (U^*, pk^*, AU^*)$ and Cre^* = (ID^*, σ^*). According to the Game 1, ID^* has not been queried to Credential generation oracle.

If $Valid \leftarrow CreVer(Cre^*, pk_S, sk_*, par)$, namely $\mathcal{A}$ outputs a valid σ^*. So $(pk^*, ID^*) \notin L_S \wedge$ accept $\leftarrow$ TkVerify (par, pk_S, pk^*, ID^*, σ^*, tk_δ^*). This means $\mathcal{B}$ can use $\mathcal{A}$ to break the unforgeability of the underlying nominative signature according to Definition 5.

Unforgeability II

- Init. $\mathcal{B}$ receives the public parameter par and a public key pk_N, and $\mathcal{B}$ runs $\mathcal{A}$. $\mathcal{B}$ sends a target $ID^* = (U^*, pk_N, AU^*)$ to $\mathcal{A}$.
- Setup. $\mathcal{B}$ sends the public parameter par to $\mathcal{A}$. $\mathcal{A}$ executes $\mathcal{NS}$ model's algorithm $KeyGen_S$ to generate TTP's public-private key pair (pk_S, sk_S).
- Create user oracle. $\mathcal{A}$ can adaptively enroll user U_i. $\mathcal{A}$ generates the public-private key pair (pk_i, sk_i) of U_i by running $UKGen$ algorithm and sends the registration information RI_i to $\mathcal{B}$. $\mathcal{B}$ returns $ID_i = (U_i, pk_i, AU_i)$ to $\mathcal{A}$.

Finally, $\mathcal{A}$ outputs a credential $Cre^* = (ID^*, \sigma^*)$ of U^*. According to the Game 2, $\mathcal{A}$ has not obtain sk_N.

If $Valid \leftarrow CreVer(Cre^*, pk_S, sk_N, par)$, namely $\mathcal{A}$ outputs a valid σ^*. So accept $\leftarrow$ Valid(par, pk_S, pk_N, ID^*, σ^*, sk_N) $\wedge$ (pk_S, ID^*, σ^*) $\notin L_R$. This means $\mathcal{B}$ can use $\mathcal{A}$ to break the security against malicious signers of the underlying nominative signature specified in Definition 6.

Therefore, by combining Unforgeability I and Unforgeability II, we can conclude that the SSO scheme proposed above is unforgeable if the underlying nominative signature scheme is unforgeable and secure against malicious signers. ■

Theorem 2. *The SSO scheme proposed above is secure against impersonation if the nominative signature scheme used above is secure against malicious signers.*

Theorem 3. *The SSO scheme proposed above is sound if the nominative signature scheme used above is protocol secure.*

The details of proofs for Theorems 1 and 2 are omitted due to space limit.

VII. Discussion and Conclusion

In this section, we give a comparison between our scheme and Han et al.'s [5] scheme. In terms of security, both schemes can resist forging attack and impersonation attack. Moreover, both schemes can satisfy the requirement of soundness because of a zero-knowledge proof. However, the dependence for TTP of our scheme is lower, because TTP can not generate a credential only by himself. In terms of efficiency, Han's scheme uses a broadcast encryption, which reduces the efficiency of whole scheme. In our scheme, it is guaranteed that even though the attacker obtains the content of a credential, the attacker can not break the security of our scheme. Therefore, our new SSO scheme meets stronger security and is more efficient.

Security notions	our scheme	Han et al.'s scheme [5]
unforgeability	yes, and stronger	yes
impersonation	yes	yes
soundness	yes	yes, but not mention
trust level of TTP	low	high

Table I
The comparison about security notions

In conclusion, based on the observation that the current SSO systems suffer from various security issues, we formalised a security model of single sign-on scheme and implemented a generic SSO by using any secure nominative signature scheme in this paper. In our model, SP is not required to communicate with TTP when it verifies the credential; an illegal person can not access to the SPs even though he/she obtains a user's valid credential; the soundness is satisfied and furthermore, the dependence of trusted third party is reduced.

References

[1] D. Chaum. Zero-knowledge undeniable signatures. In: *Proc. of EUROCRYPT'90*. LNCS, vol. 473, pp. 458-464. Springer, 1991.

[2] B. M. David, A. C. A. Nascimento, and R. Tonicelli. "A Framework for Secure Single Sign-On". *IACR Cryptology ePrint Archive*, 2011/246, 2011.

[3] B. Dodson, D. Sengupta, D. Boneh and L. M. S. Secure. Consumer-friendly web authentication and payments with a phone. In: *Pro. of the Second International ICST Conference on Mobile Computing, Applications, and Services (MobiCASE)*, 2010.

[4] L. Guo, G. Wang, and D. Wong. "Further Discussions on the Security of a Nominative Signature Scheme". *IACR Cryptology ePrint Archive*, 2006/7, 2006.

[5] J. Han, Y. Mu, W. Susilo and J. Yan. A generic construction of dynamic single sign-on with strong security. In: *Proc. of SecureComm'10*, LNICST 50, pp. 181-198, Springer, 2010.

[6] Z. Huang and Y. Wang. Convertible Nominative Signatures. In: *Proc. of Information Security and Privacy (ACISP'04)*, LNCS, vol. 3108, pp. 348-357, Springer, 2004.

[7] S. J. Kim, S. J. Park, and D. H. Won. Zero-Knowledge Nominative Signatures. In: *Proc. of International Conference on the Theory and Applications of Cryptology (PragoCrypt'96)*, pp. 380-392, 1996.

[8] J. Knapp. Overview of Zero-Knowledge Protocols. 2009.

[9] D. P. Korman and A. D. Rubin. "Risks of the passport single sign-on protocol". *Computer Networks*. vol. 33, pp. 51-58, 2000.

[10] D. Y. W. Liu, D. S. Wong, X. Huang, G. Wang, Q. Huang, Y. Mu and W. Suslio. Formal Definition and Construction of Nominative Signature, In: *Proc. of ICICS'07*, LNCS, vol. 4861, pp. 57-68, Springer, 2007.

[11] S. Lu, R. Ostrovsky, A. Sahai, H. Shacham, and B. Waters. Sequential aggregate signatures and multisignatures without random oracles. In: *Proc. of EUROCRYPT'06*, LNCS, vol. 4004, pp. 465-485. Springer, 2006.

[12] OpenID: www.openid.net.

[13] R. Oppliger. "Microsoft .Net passport: a security analysis". *IEEE computer*. vol. 36, pp. 29-35, 2003.

[14] R. Oppliger. "Microsoft .Net passport and identity management". *Information Security Technical Report*. vol. 9, pp. 26-34, 2004.

[15] J. C. N. Schuldt and G. Hanaoka. Non-transferable User Certification Secure against Authority Information Leaks and Impersonation Attacks. In: *Proc. of ACNS'11*, LNCS 6715, pp. 413-430, 2011.

[16] J. G. Steiner, C. Neuman, and J. I. Schiller. Kerberos: An authentication service for open network systems. In: *Proc. Usenix Conference*, pp. 191-202, 1988.

[17] W. Susilo and Y. Mu. On the Security of Nominative Signatures. In: *Proc. of Information Security and Privacy (ACISP'05)*, LNCS 3547, pp. 329-335. Springer, 2005.

[18] G. Wang and F. Bao. Security Remarks on Convertible Nominative Signature Scheme. In: *Proc. of the 22nd IFIP TC-11 International Information Security Conference (SEC'07)*, IFIP 232, pp. 265-275, 2007.

[19] B. Waters. Efficient identity-based encryption without random oracles. In: *Proc. EUROCRYPT'05*, LNCS 3494, pp. 114-127. Springer, 2005.

[20] J. Yu, G. Wang, Y. Mu. Provably Secure Single Sign-on Scheme in Distributed Systems and Networks. In: *Proc. 11th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom'12)*, pp. 271-278, IEEE Computer Society, 2012.

[21] W. Zhao, C. Lin and D. Ye. Provably Secure Convertible Nominative Signature Scheme, In: *Proc. of INSCRYPT'08*, LNCS 5487, pp. 23-40, Springer, 2009.