2013 5th International Conference on Intelligent Networking and Collaborative Systems

Anonymous Single Sign-on Schemes Transformed from Group 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

***Abstract*—Single Sign-on (SSO) allows a user to obtain a single credential from a Trusted Third Party (TTP) once and then authenticates himself/herself to different service providers by using the same credential. Though different SSO schemes have been obtained from various primitives, user anonymity *has not* yet been studied formally. Motivated by the fact that anonymity is a very essential security requirement in certain scenarios, in this paper we first formalize a security model of anonymous single-sign on (ASSO). Subsequently, we present a generic ASSO scheme which is transformed from group signatures. Formal proofs are provided to show that the proposed ASSO is secure under the assumption that the underlying group signature is secure according to Bellare et al.'s model introduced at CT-RSA 2005. Compared to existing SSO schemes, our transformation not only implements the user's anonymity, but also reduces the trust level in TTP.**

Keywords: Single Sign-On, Anonymity, Authentication, Group Signatures

I. Introduction

With the extensive usage of the Internet, users usually are required to access multiple services on a daily basis, and therefore they may have to maintain a lot of username/password pairs. Nevertheless, with the growth in the number of service providers this approach becomes either inefficient if each login should be unique for each service, or insecure if the same login is used for multiple services. In reality, as many as one third of users [1], [17] tend to use the same or similar passwords to access their services. Moreover, it is also a considerable burden for service providers if they have to manage credentials for dealing with credential issuing, updating, revocation, etc. Fortunately, in a single sign-on (SSO) model, during a given period (eg. one day) a user may perform just one single sign-on to a trusted third party (TTP), which is trusted by the applications he/she needs to access. Later on, each time the user wants to access an application, he/she will be automatically authenticated by the interaction between his client and the TTP, without requiring direct involvement from the user.

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 (or called trusted identity provider in [14]) 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 [2] is one of single sign-on solutions proposed in industry, which is an open and decentralized standard 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 the necessary authentication data between the identity provider and the application is complex and can be attacked through network based techniques[10].

In 2010, Han et al. [14] proposed a novel dynamic SSO model together with a generic scheme. This scheme employs a digital signature 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 inefficient process. In 2012, Yu et al. [22] 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.

Anonymity in electronic communication is very important for users in many scenarios, as they may not prefer to provide personal information to service providers though their membership for belonging to a group or association should be verified in the first place. For example, if a web site or online forum does not support user anonymity, it may be hard to attract a good number of comments and discussions. The main reason is that users may be afraid of talking freely to prevent unexpected trouble, as some sensitive topics may be involved. In addition, anonymity usually can prompt users to

978-0-7695-4988-0/13 $26.00 © 2013 IEEE
DOI 10.1109/INCoS.2013.104

express themselves while they may not like to behave the same in real life.

Unfortunately, most of existing SSO systems have shortcomings. For example, in some SSO systems like [11], 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 [15], [19], [20] 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. Moreover, SSO systems given in [14], [22] require high trust level in the TTP and anonymity is not implemented. To the best of our knowledge, anonymity in SSO has only been informally introduced in the scenario of Global System for Mobile communication by Elmufti et al [12]. In this system, to access each SP a user will use a different one-time ID as a temporary identity to authenticate himself to a TTP (the GSM network operator), and TTP then forward users request to the SP. So, the user is anonymous for the SP as the SP only knows the user's temporary ID, not real identity. In this system, single point of failure may be an issue and users are NOT anonymous to the TTP.

Group signatures, introduced by Chaum and and Van Heyst [9], implement the anonymity of signers, in contrast to the classical digital signature. With in-depth study, more security requirements were proposed in [3], [7]. After that, Bellare et al. [4], [5] proposed the formal definitions of group signatures.

Our Contributions. In this work, we aim to formalize the notion of anonymous single sign-on (ASSO). This notion is motivated by the essential need of anonymity in single sign-on systems. For example, in the scenarios of subscription of online magazines, news, and digital library or digital resource control in organisations. At the same time, we observe that anonymity has not been formally studied in SSO and that almost all SSO schemes do not support anonymity. Therefore, we are motivated to formally study anonymous single sign-on (ASSO) in this paper by proposing the first formal model and presenting a generic ASSO construction transformed from any group signature scheme, which satisfies the security notions proposed in [5]. Formal proofs are also provided to show that this ASSO is secure according to the proposed formal definitions. Though the transformation from group signature to ASSO is straightforward, the novelty of our work is threefold: (a) Our formal model is carefully formalised to capture security requirements in ASSO, in which group signature may be not necessary the only implementation tool; (b) Based on our formal security analysis, we can conclude that group signature (under proper model) is a strong enough building block to implement ASSO; and (c) ASSO forms one more potential application of group signatures, which have been extensively investigated in literature but lack real applications, except a variant technique called direct anonymous attestation (DAA) has been employed as the core mechanism in trusted computing [6].

Paper Organization. In Section 2, we formalize the security model of anonymous single sign-on (ASSO) scheme. Then, a very brief review of Bellare et al.'s formal definition for group signature is given Section 3, while more details are deferred to Appendix A. In Section 4, a generic construction of ASSO from group signature is described. Compared to previous SSO solutions given in [14], [22], there are two main advantages in our generic ASSO: users are anonymous, and the trust level of TTP is reduced as only the user knows his/her credential so that nobody can impersonate him. In Section 5, we provide formal proofs to show the security of our scheme. Finally, we give a short conclusion and discuss our future work in Section 6.

II. Formal Model of Anonymous Single Sign-On

In this section, we provide a security model to formally define anonymous single sign-on (ASSO). This model specifies the functions of ASSO and the security requirements that an ASSO should satisfy.

A. Syntax of Anonymous Single Sign-on

In an ASSO scheme, a user (U) obtains a credential from a trusted third party (TTP) once and then authenticates himself to different service providers (SPs) by generating a user proof via using the same credential. SPs can confirm the validity of each user but should not be able to trace the user's identity. Now we formalize the components of ASSO as follows.

Definition 1. *An anonymous single sign-on (ASSO) scheme involves a trusted third party TTP, a group of service providers SPs and a group of users U. It consists of five algorithms and one protocol: system setup algorithm $Setup(\cdot)$, user proof generation algorithm $UPGen(\cdot)$, user proof verification algorithm $UPVer(\cdot)$, user tracing algorithm $Trace(\cdot)$, user tracing verification algorithm $Notary(\cdot)$ and user enrollment protocol $Enrol$.*

- *$Setup$: By taking a security parameter 1^k as an input, it outputs a tuple (tpk, tik, tok), where tpk is TTP's public key, tik is TTP's private issuing key, and tok is TTP's private opening key.*
- *$Enrol$: A user can enrol in the system by running $Enrol$ protocol with TTP. Firstly, a user U_i generates his personal public/private key pair (upk_i, usk_i). Then U_i sends a request with his/her upk_i to TTP. If TTP accepts according to some registration policy (such as charging subscription fee), it uses the private issuing key tik to generate a registration certificate for U_i, denoted as reg_i. Then, TTP will send reg_i to user U_i, and store a copy of this certificate in its registration table **reg** as well. Finally, U_i can get his credential $Cre_i = (reg_i, sk_i)$, where sk_i is U_i's signing key, which is generated from reg_i and usk_i.*
- *$UPGen$: By taking the inputs of TTP's public key tpk, U_i's credential Cre_i and a message m, it outputs a*

user proof up_i showing user U_i's knowledge of credential Cre_i.

- *$UPVer$: By taking the inputs of TTP's public key tpk and a message/user proof pair (m, up_i), it outputs 1 or 0 for accepting or rejecting up_i as a valid user proof.*
- *$Trace$: By taking the inputs of registration table **reg**, TTP's private opening key tok and a valid message/user proof pair (m, up_i), it outputs an integer i ($i \geq 0$) and a proof-string τ. If $i \geq 1$, i denotes an identity; otherwise it means no group member produced this up_i. Output τ is the associated evidence, which will be used in $Notary$ algorithm.*
- *$Notary$: By taking inputs of the TTP's public key tpk, an integer $i \geq 1$, the public key upk_i of U_i, a valid message/user proof pair (m, up_i), and a proof-string τ, it checks whether τ is a valid proof showing that U_i has generated up_i for message m.*

Remark 1. *Each user U_i's personal private key usk_i is only used to generate his credential when he enrols into system, while the corresponding public key upk_i can be published through a PKI online or some approach off-line.*

Remark 2. *Compared to Han et al.'s formal model [14] and Yu et al.'s formal model [22], the most obvious advantage in our model is to achieve anonymity. In addition, the trust level of TTP in our model is lowest, because TTP can neither generate the credential without the user's involvement nor impersonate the user. Moreover, there is no broadcast encryption and key exchange in our model, thus our model is simpler.*

Remark 3. *Note that the Trace algorithm is used to identify the identity of a malicious user to prevent misusing a system, though the enforcement of such a procedure should follow a carefully specified policy and may be monitored by an additional authority, which is outside of our ASSO system. So, in this case our ASSO scheme is actually conditionally anonymous as it is traceable. In some scenarios, however, strong anonymity may be required such that a user's identity is unconditionally untraceable. To define such SSO schemes with strong anonymity, one simple way is to 'turn off' the function of traceability by generating an unknown private opening key tok. Namely, nobody (including the TTP) knows the value of tok. For a Diffie-Hellman key based system, this can be simply done by selecting a random element y from a group G with a generator g so that the corresponding private key x satisfying $y = g^x$ is unknown to anybody. For an RSA based system, this can also be done but the mechanism will be more complex.*

B. Security Definitions of Anonymous Single Sign-On

In this section, we will formally define the security notions of ASSO, including correctness, anonymity, traceability and non-frameability. Correctness is the basic requirement which ensures that a scheme works if all parties honestly follow the algorithms and protocols as specified. Anonymity protects users' privacy by requiring that SPs can only confirm the validity of each user but are not able to know the user's identity. The TTP and SPs concern traceability because it can ensure that a user can neither generate a valid user proof without a credential nor get a valid credential without TTP's endorsement. The users also concern non-frameability as it means that any adversary, including malicious TTP and/or SPs, cannot impersonate a user.

1) Correctness: Correctness of ASSO requires that (a) a user proof generated by an honest user should be valid; (b) the $Trace$ algorithm, given a valid pair of message and user proof, should correctly identify the user who generated this proof; and (c) the proof τ returned by the $Trace$ algorithm should be accepted by the $Notary$ algorithm.

Definition 2. *Formally, an anonymous single sign-on system is* **correct** *if for any $k \in \mathbb{N}$, any $(tpk, tik, tok) \leftarrow Setup(1^k)$, a valid credential Cre_i for each identity i, and any $m \in \{0,1\}^*$, there is no probabilistic polynomial time (PPT) adversary $\mathcal{A}$ which can break any of three requirements listed below:*

- $UPVer(tpk, m, UPGen(tpk, Cre_i, m)) = 1$
- $Trace(\mathbf{reg}, tok, m, UPGen(tpk, Cre_i, m)) = (i, \tau)$
- $Notary(tpk, i, upk_i, m, up_i, \tau) = true$, *where (i, τ) is output by $Trace$ algorithm w.r.t. (m, up_i).*

2) Anonymity: Anonymity means that for any adversary $\mathcal{A}$, even if he knows the issuing key tik of TTP and the private key usk_i and credential Cre_i of every user, he should be still unable to deduce who has generated a given user proof.

Definition 3. *Formally, an anonymous single sign-on scheme is said to be* **anonymous***, if no PPT adversary $\mathcal{A}$ has a non-negligible advantage against the challenger $\mathcal{C}$ in **Game 1** defined below.*

Game 1. ***Anonymity***

- ***Setup***. *$\mathcal{C}$ runs the algorithm $Setup(1^k)$ to generate the tuple (tpk, tik, tok) and then sends (k, tpk) to adversary $\mathcal{A}$. $\mathcal{C}$ now initialises a number of users denoted as a list L_U, together with the corresponding registration table **reg**.*
- ***Running Queries***. *$\mathcal{C}$ will run three oracles for adversary $\mathcal{A}$ in this phase: Corrupt TTP Oracle I, Corrupt User Oracle and Tracing Oracle.*

  *$\mathcal{A}$ could query the corrupt TTP oracle I to obtain the private issuing key tik and the registration table **reg**. This oracle simulates the compromise of issuing key and registration table.*

  $\mathcal{A}$ could query the corrupt user oracle by sending an index i, where $U_i \in L_U$, to obtain the private key usk_i and the credential $Cre_i = (reg_i, sk_i)$ of U_i from $\mathcal{C}$. A list L_{CU} is assumed initially empty and updated by $L_{CU} \leftarrow L_{CU} \cup \{U_i\}$. This oracle simulates $\mathcal{A}$'s ability to collude or compromise as many as users.

  $\mathcal{A}$ could query the tracing oracle by sending a valid message-user proof pair (m_i, up_i) to obtain corresponding identity i and a proof τ, which is the output returned by challenger $\mathcal{C}$ after execution of $Trace$ algorithm w.r.t. $(\mathbf{reg}, tok, m_i, up_i)$. If $i > 0$, it means that U_i is traced

as the user who generated (m_i, up_i); while $i = 0$ means that there is no group member being traced as the issuer of (m_i, up_i) w.r.t. the current ***reg****. An opened message-user proof list L_O is assumed initially empty and updated by $L_O \leftarrow L_O \cup \{(m_i, up_i)\}$.*

- ***Challenge****: $\mathcal{A}$ selects a message m^* and two indices i_0, i_1 of its choice, where both U_{i_0}, $U_{i_1} \in L_U$, and sends them to $\mathcal{C}$. $\mathcal{C}$ responds back with a user proof up_{i_b}, where $b \in \{0, 1\}$ is a random bit and $up_{i_b} = UPGen(tpk, Cre_{i_b}, m^*)$.*
- ***Output****: $\mathcal{A}$ outputs a bit $b^* \in \{0, 1\}$ for guessing b. During this phase, $\mathcal{A}$ can access all the above three oracles with the restrictions that (m^*, up_{i_b}) has not been asked via opening oracle. If $b^* = b$, $\mathcal{A}$ wins the game. Formally, the advantage of adversary $\mathcal{A}$ in this game is defined as*

$$\textbf{\textit{Adv}}^{anon}_{\mathcal{ASSO},\mathcal{A}}(k) = |\Pr[b^* \leftarrow \mathcal{A}^{\mathcal{O}}(k, m^*, up_{i_b}) b^* = b| \wedge (m^*, up_{i_b}) \notin L_O] - 1/2|.$$

Remark 4. *In this game, $\mathcal{A}$ is allowed to obtain the private issuing key tik, so he can enrol new users by himself and change the content of registration table* ***reg****, which is the copy held by $\mathcal{A}$ itself. Note that according to above definition, $\mathcal{A}$ can corrupt the credential of any user, including the target users U_{i_0} and U_{i_1}. Due to this reason, it is not necessary to offer $\mathcal{A}$ the user proof generating oracle.*

3) Traceability: Traceability means that for any PPT adversary $\mathcal{A}$, who may know the opening key tok of TTP and corrupt the private key/credential (usk_i, Cre_i) of as many users as he likes, he should still be unable to generate a message/user proof pair (m, up) that cannot be traced to a user or that can be traced to a user U_i but $Notary(tpk, i, m, up_i, \tau)$ = false.

Definition 4. *Formally, an ASSO scheme is* **traceable***, if no PPT adversary $\mathcal{A}$ has a non-negligible advantage against the challenge $\mathcal{C}$ in the game defined below.*

Game 2. ***Traceability***

- ***Setup****. $\mathcal{C}$ runs the algorithm $Setup(1^k)$ to generate the tuple (tpk, tik, tok) and initialises a number of users, which is denoted as a list L_U, together with registration table* ***reg****. Then, k, tpk, L_U and* ***reg*** *are sent to adversary $\mathcal{A}$.*
- ***Running Queries****. $\mathcal{C}$ will run three oracles in this phase: Corrupt TTP Oracle II, Enrol Oracle and Corrupt User Oracle.*

  $\mathcal{A}$ could query the corrupt TTP oracle II to obtain the private opening key tok. This oracle simulates the compromise of opening key.

  $\mathcal{A}$ could query the enrol oracle to enrol a new user U_i with $\mathcal{C}$ by running $Enrol$ protocol. If $\mathcal{A}$ performs honestly, he knows the private key usk_i and is able to obtain credential $Cre_i = (reg_i, sk_i)$ of U_i, where reg_i is returned from the challenger $\mathcal{C}$. Correspondingly $\mathcal{C}$ will update the registration table by ***reg*** $\leftarrow$ ***reg*** $\cup \{reg_i\}$*. This oracle simulates $\mathcal{A}$'s ability to collude or compromise as many as new users.*

  $\mathcal{A}$ could query the corrupt user oracle by sending an index i, where $U_i \in L_U$, to obtain the private key usk_i and the credential $Cre_i = (reg_i, sk_i)$ of U_i from $\mathcal{C}$. A corrupted user list L_{CU} is assumed initially empty, and then updated by $L_{CU} \leftarrow L_{CU} \cup \{U_i\}$. This oracle simulates $\mathcal{A}$'s ability to collude or compromise as many as users initialised by challenger $\mathcal{C}$.
- ***Output****. Finally, $\mathcal{A}$ outputs a message m^* and a user proof up^*. If up^* turns up to be a valid user proof for m^*, $\mathcal{C}$ will try to trace it to a user. If it traces to nonmember or if it traces to a user U_j but $Notary(tpk, j, m^*, up^*, \tau)$ = false, then $\mathcal{A}$ wins the game. Formally, $\mathcal{A}$'s advantage in this game is defined by*

$$\textbf{\textit{Adv}}^{trac}_{\mathcal{ASSO},\mathcal{A}}(k) = \Pr[(m^*, up^*) \leftarrow \mathcal{A}^{\mathcal{O}}(k) | UPVer(tpk, m^*, up^*) = 1 \wedge (Trace(\textbf{\textit{reg}}, tok, m^*, up^*) = (0, \tau) \vee (Trace(tok, m^*, up^*) = (j, \tau) \wedge Notary(tpk, j, m^*, up^*, \tau) = false))].$$

Remark 5. *In this game, as $\mathcal{A}$ is allowed to obtain the private opening key tok he can open any message-user proof pair (m, up) by himself without needing opening oracle. Moreover, note that both unforgeability of credential [14], [22] and soundness of SSO [22] are implied by traceability defined above. On the one hand, according to Definition 4 without the knowledge of tik $\mathcal{A}$ is not able to forge a valid credential for a new user and then employ this credential to generate user proofs (unforgeability). On the other hand, in Definition 4 it is infeasible for $\mathcal{A}$ to generate a valid user proof without holding a valid credential (soundness).*

4) Non-frameability: Non-frameability means that for any PPT adversary $\mathcal{A}$, even he can corrupt TTP and all users except the target uncorrupted user U^*, he should be still unable to generate a valid message/user proof pair (m^*, up^*), which traces to user U^*. In other words, non-frameability implies that even a malicious TTP colluding with dishonest users is still unable to frame a honest user by forging a valid user proof.

Definition 5. *An anonymous single sign-on system satisfies* **non-frameability***, if no PPT adversary $\mathcal{A}$ has a non-negligible advantage against the challenger in in the game defined below.*

Game 3. ***Non-frameability***

- ***Setup****: $\mathcal{C}$ runs algorithm $Setup(1^k)$ to generate a tuple (tpk, tik, tok), and initialises a number of users, denoted as the list L_U, together with the corresponding registration table* ***reg****. Then, C sends (k, tpk, L_U) to adversary $\mathcal{A}$.*
- ***Running Queries****: $\mathcal{C}$ will run three oracles in this phase: Corrupt TTP Oracle, Corrupt User Oracle, and Generate User Proof Oracle.*

  $\mathcal{A}$ could query the corrupt TTP oracle to obtain the

*issuing key tik, the opening key tok and the registration table **reg**. This oracle simulates $\mathcal{A}$'s ability to compromise or collude TTP.*

$\mathcal{A}$ could query the corrupt user oracle by sending an index i, where $U_i \in L_U$, to obtain the private key usk_i and the credential Cre_i = (reg_i, sk_i) of U_i from $\mathcal{C}$. A corrupted user list L_{CU} is assumed initially empty, and then updated by $L_{CU} \leftarrow L_{CU} \cup \{U_i\}$. This oracle simulates $\mathcal{A}$'s ability to collude or compromise as many as users initialised by challenger $\mathcal{C}$.

$\mathcal{A}$ could query the generate user proof oracle by sending a message-user pair (m_i, U_i) to obtain a valid user proof up_i from challenger $\mathcal{C}$, where $U_i \in L_U$ and $up_i = UPGen(tpk, Cre_i, m_i)$. A generated user proof list, denoted as L_{UP}, which is initialised as empty, will be updated by $L_{UP} \leftarrow L_{UP} \cup \{(m_i, U_i, up_i)\}$.

- ***Output**: Finally, $\mathcal{A}$ outputs a message m^* and a user proof up^*. If $UPVer(tpk, m^*, up^*) = 1$, $\mathcal{C}$ will try to trace this user proof. $\mathcal{A}$ wins the game, if a user $U_j \in L_U$ is traced, but U_j has never been corrupted by $\mathcal{A}$ in corrupt user oracle and $(m^*, U_j, *)$ has never been returned in the generate user proof oracle. $\mathcal{A}$'s advantage in this game is defined by*

$$\begin{aligned}\mathbf{Adv}^{nonf}_{\mathcal{ASSO},\mathcal{A}}(k) = \Pr[(m^*, up^*) \leftarrow \mathcal{A}^{\mathcal{O}}(k, L_U)| \\ UPVer(tpk, m^*, up^*) = 1 \wedge U_j \in \\ L_U \backslash L_{CU} \wedge (m^*, U_j, *) \notin L_{UP}, \\ where\ Trace(tpk, tok, m^*, up^*) = (j, \tau)].\end{aligned}$$

Remark 6. *The above definition of non-frameability also covers credential privacy [21], [14], [22], which is defined to guarantee that colluded dishonest service providers should not be able to fully recover a user's credential and then impersonate the user to log in to other service providers.*

III. Dynamic Group Signatures

In the formal model proposed by Bellare et al. [5], a dynamic group signature scheme GS comprises a trusted party for generating keys, two authorities called the issuer and opener, and a body of users, each with a unique identity $i \in \mathbb{N}$. A GS is specified as a tuple (**GKg**, **UKg**, **Join**, **Iss**, **GSig**, **GVf**, **Open**, **Judge**) of polynomial-time algorithms, whose intended usage and functionality are described in [5].

A correct and secure GS should satisfy four properties, namely, correctness, anonymity, traceability, and non-frameability, which are defined by experiments in which an adversary has access to certain oracles. Moreover, the oracles in each of these experiments are assumed to maintain and manipulate the global variables: a set HU of honest users; a set CU of corrupted users; a set GSet of message-signature pairs; a table **upk** such that **upk**[i] contains the public key of $i \in \mathbb{N}$; a table **reg** such that **reg**$[i]$ contains the registration information of group member i. The sets HU, CU, GSet, as well as all entries of the tables **upk** and **reg**, are assumed initially empty, denoted as ε. Formal definitions of these properties are specified in [5].

IV. Generic Construction of ASSO from Group Singatures

In this section, we give a straightforward construction of ASSO from dynamic group signatures. It means that any group signature $\mathcal{GS}$ satisfying the definition given in Section 3 can be used to derive an ASSO scheme by simply using group signature to issue a user proof.

- $Setup(1^k)$: It runs group signature $\mathcal{GS}$'s key generation algorithm **GKg**(1^k) to obtain the tuple (gpk, ik, ok), and then sets (tpk, tik, tok) = (gpk, ik, ok).
- $Enrol$: A user can enrol in the system by running $Enrol$ protocol with TTP. Firstly, user U_i runs $\mathcal{GS}$'s algorithm **UKg**(1^k) to generate a key pair $(upk[i], usk[i])$, and sets (upk_i, usk_i) = $(upk[i], usk[i])$. Then, U_i sends a request to TTP. If TTP accepts, it uses the issuing key tik to run $\mathcal{GS}$'s algorithm **Iss** to generate an certificate for U_i, denoted as reg_i, which will be stored in the registration table **reg** and sent to U_i as well. If U_i accepts, he runs $\mathcal{GS}$'s algorithm **Join** to generate his signing key $gsk[i]$. Finally, the credential of U_i is defined as $Cre_i = (reg_i, sk_i)$ where $sk_i = gsk[i]$.
- $UPGen(tpk, Cre_i, m)$: It runs $\mathcal{GS}$'s algorithm **GSig**(sk_i, m) to generate a signature σ, and outputs the user proof $up_i = \sigma$.
- $UPVer(tpk, m, up_i)$: It runs $\mathcal{GS}$'s algorithm **GVf**(tpk, m, up_i) to verify if up_i is a valid group signature on m, and correspondingly outputs 1 or 0 for accepting or rejecting up_i as a valid user proof for message m.
- $Trace($**reg**$, tok, m, up_i)$: It runs $\mathcal{GS}$'s algorithm **Open**$(tok,$ **reg**$, m,\ up_i)$ to output an integer i and a proof-string τ. If $i \geq 1$, i denotes U_i; otherwise it means no group member produced this up_i. Output τ is the associated evidence, which will be used in $Notary$ algorithm.
- $Notary(tpk, i, upk_i, m, up_i, \tau)$: It runs $\mathcal{GS}$'s algorithm **Judge**$(tpk, i, upk_i, m, up_i,\ \tau)$ to check whether τ is a proof that U_i generated up_i.

Instantiation. Our ASSO construction is a generic transformation from Bellare et al.'s dynamic group signature model [5]. This implies that any secure group signature follow Bellare et al.'s model, like [13], [18], can be used to instantiate our ASSO scheme. In particular, Nguyen and Safavi-Naini [18] proposed a group signature scheme without requiring trapdoor, in which both signature and public key are constant-size regardless of the group size. This scheme is efficient and practical because the lengths of signature, group manager's public key, private issuing key, private opening key and group member's signing key are 574 bytes, 363 bytes, 22 bytes, 44 bytes, and 192 bytes, respectively. Thus our transformation can be efficiently implemented in concrete instantiation. Due to space limit, detailed analysis has to be omitted here.

V. Security Proofs

It is easy to see that correctness of a dynamic group signature (Section 3.2.1) implies the correctness of our generic ASSO described above according to Definition 2. Now, we prove that other properties can be guaranteed as well.

Theorem 1. *The ASSO scheme proposed above is anonymous if the dynamic group signature scheme used above is anonymous.*

Proof. Suppose there exists a PPT adversary $\mathcal{A}$ which can break the anonymity of our generic construction of ASSO. We will show that there exists an adversary $\mathcal{B}$ which can break the anonymity of the dynamic group signature scheme $\mathcal{GS}$, which is used to construct ASSO.

- Init. $\mathcal{B}$ receives the security parameter $k \in \mathbb{N}$, a public key gpk and a private key ik. $\mathcal{B}$ sets gpk as the public key tpk of TTP and ik as the private issuing key tik of TTP. As $\mathcal{B}$ knows ik, it initialises a number of users in the list L_U and generates the corresponding registration table **reg** by running **Join** and **Iss** algorithms. Then, $\mathcal{B}$ runs $\mathcal{A}$ by simulating oracles for $\mathcal{A}$ as follows.
- Corrupt TTP Oracle I: If $\mathcal{A}$ queries the partially corrupt TTP oracle I, $\mathcal{A}$ returns back $\mathcal{B}$ the private issuer key tik and the registration table **reg**.
- Corrupt User Oracle: If $\mathcal{A}$ queries the corrupt user oracle by sending an index i, where $U_i \in L_U$, $\mathcal{B}$ retrieves reg_i from **reg**, redirects this query to $\mathcal{GS}$'s oracle $USK(i)$ to get usk_i, and calculates sk_i from reg_i and usk_i. Then, $\mathcal{B}$ will respond $\mathcal{A}$ the private key usk_i and the credential Cre_i = (reg_i, sk_i) of U_i. In addition, a corrupted user list L_{CU}, assumed initially empty, will be updated by $L_{CU} \leftarrow L_{CU} \cup \{U_i\}$.
- Opening Oracle: If $\mathcal{A}$ queries the opening oracle by sending a message m_i and a user proof up_i. $\mathcal{B}$ redirects this query to $\mathcal{GS}$'s oracle $Open(\cdot,\cdot)$ to get a pair (i, τ). Then, (i, τ) will be forwarded to $\mathcal{A}$. List L_O, assumed initially empty, will be updated by $L_O \leftarrow L_O \cup \{(m_i, up_i)\}$.

Once $\mathcal{A}$ selects a message m^* and two indices i_0 and i_1, $\mathcal{B}$ will forward (m^*, i_0, i_1) to his challenger, who will respond back with a signature σ_b to $\mathcal{B}$. Then, $\mathcal{B}$ returns the user proof $up_b = \sigma_b$ to $\mathcal{A}$.

Finally, $\mathcal{A}$ will output a guess $b^* \in \{0, 1\}$. If $\mathcal{A}$ can break the anonymity of our generic construction for ASSO, $b^* = b$ will be true with non-negligible probability and $(m^*, up_b) \notin L_O$. This means that $\mathcal{B}$ can just forward b^* to his challenger to break the anonymity of the dynamic group signature scheme with the same non-negligible probability. It is also easy to see that $\mathcal{B}$'s running time is polynomial if $\mathcal{A}$'s is.

Therefore, the ASSO scheme proposed above is anonymous if the dynamic group signature scheme used above is anonymous. □

Theorem 2. *The ASSO scheme proposed above is traceable if the dynamic group signature scheme used above is traceable.*

Proof. Suppose there exists a PPT adversary $\mathcal{A}$ which can break the traceability of our generic construction for ASSO. We will show that there exists an adversary $\mathcal{B}$ which can break the traceability of above dynamic group signature scheme $\mathcal{GS}$. $\mathcal{C}$ is the challenger of $\mathcal{GS}$.

- Init. $\mathcal{B}$ receives the public parameter $k \in \mathbb{N}$, a public key gpk and a private opening key ok from the challenger $\mathcal{C}$. $\mathcal{B}$ simulates an ASSO system by setting the tuple (gpk, ik, ok) as the TTP's key tuple (tpk, tik, tok), where the issuing key ik of $\mathcal{GS}$ is not known to $\mathcal{B}$ but it can be implicitly set as the private issuing key of $ASSO$. Then, $\mathcal{B}$ initialises a number of users in the list L_U by enquiring oracle $AddU(\cdot)$, which is provided by challenger $\mathcal{C}$ of $\mathcal{GS}$. After that, $\mathcal{B}$ can get the registration table **reg** by enquiring $RReg(\cdot)$ oracle. Now, $\mathcal{B}$ sends $(k, tpk, L_U, \textbf{reg})$ to $\mathcal{A}$, and then runs sub-routine $\mathcal{A}$ as follows.
- Corrupt TTP Oracle II: If $\mathcal{A}$ queries the corrupt TTP oracle II, $\mathcal{B}$ simply returns the private opening key tok to $\mathcal{A}$.
- Enrol Oracle: If $\mathcal{A}$ queries the enrol oracle with $\mathcal{B}$ by running $Enrol$ protocol to enrol a new user U_i with user public key upk_i. If $\mathcal{A}$ behaves honestly on behalf of U_i, $\mathcal{B}$ can obtain reg_i by enquiring oracle $SndToI(\cdot,\cdot)$, which is provided by challenger $\mathcal{C}$ of $\mathcal{GS}$. Then, $\mathcal{B}$ will update **reg** by **reg** $\leftarrow$ **reg**$\cup\{reg_i\}$ and forwards reg_i to $\mathcal{A}$ so that $\mathcal{A}$ can obtain both usk_i which is selected by $\mathcal{A}$ on behalf of U_i and U_i's credential $Cre_i = (reg_i, sk_i)$, where sk_i is calculated from usk_i and reg_i.
- Corrupt User Oracle: If $\mathcal{A}$ queries the corrupt user oracle by sending an index i, where $U_i \in L_U$. To get (usk_i, reg_i, cre_i) $\mathcal{B}$ redirects this query to $USK(\cdot)$ oracle and $RReg(\cdot)$ oracle which are provided by $\mathcal{C}$. Then, $\mathcal{B}$ forwards $\mathcal{A}$ the private key usk_i and the credential Cre_i = (reg_i, sk_i) of U_i. The corrupted user list L_{CU}, assumed initially empty, is updated by L_{CU} = $L_{CU} \cup \{U_i\}$.

Finally, $\mathcal{A}$ outputs a message-user proof pair (m^*, up^*). As $\mathcal{A}$ can break the traceability of our generic ASSO construction, with a non-negligible probability (m^*, up^*) should be valid and it will trace to nonmember or to a user U^* but $Notary(tpk, *, m^*, up^*, \tau)$ = false. So, by setting $\sigma^* = up^*$ $\mathcal{B}$ can trivially forward a message-signature pair (m^*, σ^*) as its forgery to break the traceability of the dynamic group signature scheme $\mathcal{GS}$ used to construct our ASSO. According the definition given in Section 3.2.3, it is not difficult to see that attacker B breaks the traceability of the underlying group signature $\mathcal{GS}$ with the same non-negligible probability in polynomial time.

Therefore, the ASSO scheme proposed above is traceable if the dynamic group signature scheme used above is traceable. □

Theorem 3. *The ASSO scheme proposed above is non-frameable if the dynamic group signature scheme used above is non-frameable.*

Proof. Suppose that there exist a PPT adversary $\mathcal{A}$ which can break the non-frameability of our generic ASSO construction.

We will show that there exists a PPT adversary $\mathcal{B}$ which can break the non-frameability of the dynamic group signature scheme $\mathcal{GS}$ used in our ASSO. Let $\mathcal{C}$ be the challenger of $\mathcal{GS}$.

- Init. $\mathcal{B}$ receives the public parameter $k \in \mathbb{N}$, a public key gpk, a private opening key ok and a private issuing key ik from $\mathcal{C}$. $\mathcal{B}$ simulates an ASSO system by setting the tuple (gpk, ik, ok) as the TTP's key tuple (tpk, tik, tok). $\mathcal{B}$ initialises a number of users in the list L_U by honestly running **Join** and **Iss** algorithms. During this process, $\mathcal{B}$ obtains L_U and **reg**. Now, $\mathcal{B}$ sends (k, tpk, L_U) to $\mathcal{A}$, and then runs sub-routine $\mathcal{A}$ as follows.
- Corrupt TTP Oracle: If $\mathcal{A}$ queries the corrupt TTP oracle, $\mathcal{B}$ sends the private issuer key tik, the private opening key tok and the registration table **reg** to $\mathcal{A}$.
- Corrupt User Oracle: If $\mathcal{A}$ queries the corrupt user oracle by sending an index i,where $U_i \in L_U$, $\mathcal{B}$ will retrieve $\mathbf{reg}_i$ from **reg** and obtain $(\mathbf{gsk}[i], \mathbf{usk}[i])$ from challenger $\mathcal{C}$ by asking $USK(i)$ oracle. Then, $\mathcal{B}$ sends the private key usk_i and the credential Cre_i = (reg_i, sk_i) of U_i to $\mathcal{A}$, where $usk_i = \mathbf{usk}[i]$, $reg_i = \mathbf{reg}_i$, and $sk_i = \mathbf{gsk}[i]$. The corrupted user list L_{CU}, assumed initially empty, will be updated by L_{CU} = $L_{CU} \cup \{U_i\}$.
- Generate User Proof Oracle: If $\mathcal{A}$ queries the generate user proof oracle by a message m_i and a user ID U_i, where $U_i \in L_U$, $\mathcal{B}$ redirects this query to oracle $Gsig(i, m_i)$ which is provided by $\mathcal{C}$. After obtaining $\sigma_i \leftarrow \mathbf{GSig}(\mathbf{gsk}_i, m_i)$, by setting $up_i = \sigma_i$ B sends up_i w.r.t (m_i, U_i) to $\mathcal{A}$. The generated user proof list L_{UG}, assumed initially empty, is then updated by L_{UP} = $L_{UP} \cup \{m_i, U_i, up_i\}$.

Finally, $\mathcal{A}$ will output a message m^* and a user proof up^*. If $\mathcal{A}$ can break the non-frameability of our generic ASSO construction, with non-negligible probability (m^*, up^*) should be valid and it will be traced to a user U^*, where $U^* \in L_U \backslash L_{CU} \wedge (m^*, U^*, *) \notin L_{UP}$. So, $\mathcal{B}$ can trivially set $\sigma^* = up^*$ and output its forgery (m^*, σ^*) to break the non-frameability of the dynamic group signature scheme $\mathcal{GS}$ used in our ASSO. According the definition given in Section 3.2.4, it is not difficult to see that attacker B breaks the the non-frameability of the underlying group signature $\mathcal{GS}$ with the same non-negligible probability in polynomial time.

Hence, the ASSO scheme proposed above is non-frameable if the dynamic group signature scheme $\mathcal{GS}$ used in our construction is non-frameable. □

VI. Conclusion and Future Work

In this paper, we proposed the first formal model to capture an anonymous single sign-on (ASSO). In this model, a user is anonymous and the TTP's trust level is reduced, compared to existing SSO solutions. We also demonstrated how any dynamic group signature scheme, which follows Bellare et al.'s model, can be straightforwardly transformed to an ASSO. We also proved the security of this generic transformation by assuming the security of the underlying group signature. This formally confirms the relationship between a cryptographic primitive and a security application. Namely, group signatures are a strong enough tool to implement ASSO and ASSO forms a good potential application of group signatures.

As our future work, several issues can be investigated. Firstly, new ASSO solutions could be constructed from primitives other than group signatures. Secondly, mutual authentication and/or key agreement could be introduced into ASSO as existing solutions for SSO given in [21], [22]. Finally, how to implement fine-grained access control can be considered.

References

[1] Security at risk as one third of surfers admit they use the same password for all websites. March 2009. Available at http://www.sophos.com/pressoffice/news/articles/2009/03/password-security.html

[2] OpenID: www.openid.net.

[3] G. Ateniese and G. Tsudik. Some open issues and directions in group signature. *Financial Cryptography'99*, Lecture Notes in Computer Science Vol.1648, M. Franklin ed., Springer-Verlag, 1999.

[4] M. Bellare, D. Micciancio, and B. Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In E. Biham, editor, *Proceedings of Eurocrypt 2003*, volume 2656 of *LNCS*, pages 614-629. Springer-Verlag, May 2003.

[5] M. Bellare, H. Shi, and C. Zhang. Foundations of group signatures: the case of dynamic groups. *Proceedings of CT-RSA'05*, volume 3376 of *LNCS*, pages 136-153. Springer-Verlag, 2005.

[6] E. F. Brickell, J. Camenisch, and L. Chen. Direct anonymous attestation. *Proc. of the 11th ACM Conf. on Computer and Communications Security*, pp. 132-145, ACM, 2004.

[7] L. Chen and T. P. Pedersen. New group signature schemes. *Advances in Cryptology-EUROCRYPT'94*, Lecture Notes in Computer Science Vol.950, A. DeSantis ed., Springer-Verlag, 1994.

[8] C. C. Chang and C. Y. Lee. A secure single sign-on mechanism for distributed computer networks, *IEEE Trans. Ind. Electron.*, Vol. 59, No. 1, pp. 629-637, Jan. 2012.

[9] D. Chaum and E. vanHeyst. Group signatures. *Advances in Cryptology EUROCRYPT'91*, Lecture Notes in Computer Science Vol.547, D. Davies ed., Springer-Verlag, 1991.

[10] B. M. David, A. C. A. Nascimento, and R. Tonicelli. A framework for secure single sign-on.", *IACR Cryptology ePrint Archive*, report 246, 2011.

[11] B. Dodson, D. Sengupta, D. Boneh, and M. S. Lam. Secure, consumer-friendly web authentication and payments with a phone. *In: Proc. of the 2nd International ICST Conference on Mobile Computing, Applications, and Services (MobiCASE)*, LNICS 76, pp. 17-38, 2012.

[12] K. Elmufti, D. Weerasinghe, M. Rajarajan, and V. Rakocevic. Anonymous authentication for mobile Single Sign-On to protect user privacy. *International Journal of Mobile Communications*, 6(6): 760-769, August 2008.

[13] J. Groth. Simulation-Sound NIZK Proofs for a Practical Language and Constant Size Group Signatures. *in Proc. of ASIACRYPT 2006*, pp. 444-459, 2006.

[14] 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*, LNICS 50, pp. 181-198, Springer, 2010.

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

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

[17] B. Stone-Gross, M. Cova, L. Cavallaro, et al. Your botnet is my botnet: Analysis of a botnet takeover, *in Proc. of the 16th ACM conference on Computer and communications security*, pp. 635-647, ACM, 2009.

[18] L. Nguyen, R and Safavi-Naini. Efficient and provably secure trapdoor-free group signature schemes from bilinear pairings. In *Proc. of ASIACRYPT 2004*, pp.372-386, 2004.

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

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

[21] G. Wang, J. Yu, and Q. Xie. Security analysis of a single sign-on mechanism for distributed computer networks", *IEEE Trans. on Industrial Informatics*, 9(1): 294-302, Feb. 2013.

[22] J. Yu, G. Wang, and Y. Mu. Provably Secure Single Sign-on Scheme in Distributed Systems and Networks. *TrustCom 2012*, pp. 271-278, 2012.