2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications

Provably Secure Single Sign-on Scheme in Distributed Systems and Networks

Jiangshan Yu, Guilin Wang, and Yi Mu
Center for Computer and Information Security Research
School of Computer Science and Software Engineering
University of Wollongong, Australia
Email: {jy898,guilin,ymu}@uow.edu.au

***Abstract*—Distributed systems and networks have been adopted by telecommunications, remote educations, businesses, armies and governments. A widely applied technique for distributed systems and networks is the single sign-on (SSO) which enables a user to use a unitary secure credential (or token) to access multiple computers and systems where he/she has access permissions. However, most existing SSO schemes have not been formally proved to satisfy credential privacy and soundness of credential based authentication. To overcome this drawback, we formalise the security model of single sign-on scheme with authenticated key exchange. Specially, we point out the difference between soundness and credential privacy, and define them together in one definition. Also, we propose a provably secure single sign-on authentication scheme, which satisfies soundness, preserves credential privacy, meets user anonymity, and supports session key exchange. The proposed scheme is very efficient so that it suits for mobile devices in distributed systems and networks.**

***Index Terms*—Single sign-on, Distributed systems and networks, Soundness, Authentication, Information security.**

I. Introduction

With the wide spreading of distributed computer networks, various network services have gained importance and popularity in recent few years [1][2]. Consequently, user authentication [3] has been widely used in distributed computer networks to identify a legal user who requires accessing network services. To prevent bogus servers, mutual authentication should be considered, and also, a session key establishment is normally required. In addition, user privacy may be desired in distributed computing environments since the information exchanged might be abused by some organizations for marketing purposes [4]. However, designing efficient and secure mutual authentication protocols is challenging in computer networks.

Moreover, with the increasing usage of network services, a user may need to maintain more and more ID/password pairs for accessing different distributed service providers, which impose a burden on users and service providers as well as the communication overhead of computer networks. Single sign-on (SSO) mechanism [5] provides a good remedy to this problem, as it allows a user with a single credential to access multiple service providers. Intuitively, there are three basic security requirements for SSO schemes, namely completeness, soundness and credential privacy [16], [6]. However, to the best of our knowledge soundness has not been formally studied yet and how to preserve both soundness and credential privacy is still a challenge [6].

In 2000, Lee and Chang [7] first proposed an SSO scheme with user anonymity. Later, Wu and Hsu [8] pointed out that Lee-Chang scheme suffers from masquerading attack and identity disclosure attack. Meanwhile, Yang et al. [9] showed that Wu-Hsu scheme can not preserve credential privacy either since a malicious service provider can recover users' credentials, and then proposed an improvement to overcome this limitation. In 2006, however, Mangipudi and Katti [10] pointed out that Yang et al.'s scheme is insecure against DoS (Deniable of Service) attack and presented a new scheme. In 2009, Hsu and Chuang [11] demonstrated that both Yang et al. and Mangipudi-Katti schemes have not provided user anonymity since their schemes are vulnerable to identity disclosure attacks. To prevent such attacks, Hsu and Chuang proposed an RSA-based user identification scheme.

Recently, Chang and Lee [12] pointed out that Hsu-Chuang scheme is vulnerable to impersonation attacks and the scheme requires additional time-synchronized mechanisms which has unstable latency in distributed networks. Then, they proposed a user anonymity preserving improvement with high efficiency. The scheme uses random nonce to replace additional time-synchronized mechanism, does not need PKI (Public key infrastructure) for users, and suits for mobile device users. However, the security analysis [6] shows that Chang-Lee scheme fails to provide proper user authentication and to preserve credential privacy since the knowledge proof of user authentication guarantees neither soundness nor credential privacy.

As promoted in [6], it is worthy to overcome the flaws in Chang-Lee scheme to obtain an efficient and provably secure scheme for mobile device users in distributed systems and networks. Moreover, the soundness of credential based authentication should be formalised and the credential privacy should be preserved. Motivated to solve these issues, in this paper we first specify a formal model for SSO with a unified definition to formally specify soundness and credential privacy (Section II). Then, after reviewing Chang-Lee SSO scheme in Section III and Schnorr signature [13] in Section IV, we improve Chang-Lee scheme by exploiting Schnorr signature in Section V due to its simplicity and unforgeability [14], [15], while keep Chang-Lee's session key establishment part

978-0-7695-4745-9/12 $26.00 © 2012 IEEE
DOI 10.1109/TrustCom.2012.228

IEEE computer society

unchanged. The security of the proposed protocol is discussed in Section VI. Finally, section VII concludes this paper.

II. Formal Model

In this section we present a formal model to define authenticated key exchange single sign-on ($AKESSO$) scheme and its security requirements. Specially, we list the components (e.g. syntax) of $AKESSO$, define correctness, describe an adversary model, and formally specify three security properties, including secure credential based user authentication, secure credential based service provider authentication, and session key security.

Definition 1. *An **authenticated key exchange single sign-on (AKESSO)** scheme comprises a trusted credential provider TCP, a group of service providers P and a group of users U. It consists of eight algorithms and one protocol: initialization algorithm $Init(\cdot)$, identity generation algorithm $IdGen(\cdot)$, credential generation algorithm $CGen(\cdot)$, credential verification algorithm $CVer(\cdot)$, user proof generation algorithm $UPGen(\cdot)$, user proof verification algorithm $UPVer(\cdot)$, service provider proof generation algorithm $SPPGen(\cdot)$, and service provider proof verification algorithm $SPPVer(\cdot)$, and key exchange protocol $\prod$.*

1) *$Init(\lambda)$: Taking security parameter λ_0 (or λ_1) as input, outputs the public/private key pair (PK, SK) for TCP (or (PK_j, SK_j) for $P_j \in P$).*
2) *$IdGen(RI_i)$: Taking registration information RI_i as input, outputs an unique identity ID_i for a user $U_i \in U$.*
3) *$CGen(ID_i, SK)$: Taking an identity ID_i and TCP's private key SK as input, outputs a credential C_i for user U_i.*
4) *$CVer(C_i, ID_i, PK)$: Taking credential C_i, an identity ID_i, and TCP's public key PK as input, outputs "1" or "0" for accepting or rejecting credential C_i respectively.*
5) *$UPGen(C_i, ID_i, PK, M)$: Taking a credential C_i, an identity ID_i, TCP's public key PK and a temporal message M generated in a session as input, outputs a user proof up_i showing user U_i's knowledge of credential C_i.*
6) *$UPVer(up_i, ID_i, PK, M)$: Taking a user proof up_i, an identity ID_i, TCP's public key PK, and a temporal message M generated in a session as input, outputs "1" or "0" for accepting or rejecting up_i as a valid credential proof w.r.t. identity ID_i respectively.*
7) *$SPPGen(SK_j, M')$: Taking service provider P_j's private key SK_j and a temporal message M' generated in a session as input, outputs a service provider proof spp_j showing P_j's knowledge of SK_j.*
8) *$SPPVer(spp_j, PK_j, M')$: Taking a service provider proof spp_j, P_j's public key PK_j, and a temporal message M' generated in a session as input, outputs "1" or "0" for accepting or rejecting spp_j as a valid service provider proof w.r.t. public key PK_j respectively.*
9) *$\prod$: This is a key exchange protocol run by a user U_i with private input C_i and a service provider P_j with private input SK_j. After the completion of each protocol instance, U_i will output a session key K_{ij} if he/she accepts P_j. Similarly, after the completion of each protocol instance P_j will output a session key K_{ji} if it accepts U_i. (Ideally, K_{ij} and K_{ji} are expected to be the same value.)*

Remark 1. *The above definition focuses on public key based AKESSO with non-interactive proofs. It could be extended to support interactive proofs, where sp_i and ssp_j are generated by interactive protocols run by user U_i and service provider P_j. However, defining symmetric key based AKESSO will be another story, which is out the scope of this paper.*

Remark 2. *Compared to Han et al.'s formal model given in [16], we require key exchange in AKESSO, and each user does not need to hold a public/private key pair. However, in Han et al.'s definition TCP (called IdP in their paper) is less trusted as it will not be able to impersonate any user: Each user will run a zero knowledge protocol to show that he/she knows the private key corresponding to the public key embedded in his/her credential.*

Before formally defining security properties, we naturally require an AKESSO should be *correct*. Namely, a credential C_i generated by the trusted credential provider TCP will be valid, a user proof up_i issued properly by user u_i who holds a valid credential C_i will be accepted by a service provider P_j according to $UPVer$ algorithm, a service provider proof spp_j issued properly by P_j will be accepted by user U_i according to $SPPVer$ algorithm, and U_i and P_j will accept each other and output the same session key if they honestly run the key exchange protocol $\prod$. Formally, we define correctness as below.

Definition 2. ***(Correctness)** An AKESSO scheme is called correct if it satisfies all the following conditions:*

1) *For any RI_i and any key pair (PK, SK), if $ID_i \leftarrow IdGen(RI_i)$ and $C_i \leftarrow CGen(ID_i, SK)$, then $CVer(C_i, ID_i, PK) = 1$.*
2) *For any ID_i, any key pair (PK, SK) and any M, if $C_i \leftarrow CGen(ID_i, SK)$ and $up_i \leftarrow UPGen(C_i, ID_i, PK, M)$, then $UPVer(up_i, ID_i, PK, M) = 1$.*
3) *For any key pair (PK_j, SK_j) and any M', if $spp_j \leftarrow SPPGen(SK_j, M')$, then $SPPVer(spp_j, PK_j, M') = 1$.*
4) *For any user U_i with valid credential C_i and service provider P_j with private key SK_j, if both of them run the key exchange protocol $\prod$ honestly, then they will accept each other and output the same session key, i.e., $K_{ij} = K_{ji}$.*

Informally, an AKESSO scheme is secure if all the desired functionalities given in the above definition can be carried out only by the proper entities, i.e., not by attackers who

are allowed to access all possible resources in a rigorously specified adversary model. In fact, we shall define *security of SSO authentication* which corresponds to items 1) to 3), and *session key privacy* which corresponds to item 4).

To further define these security properties, we specify the **adversary model** as follows: Let $\prod_{TCP}$ be the trusted authority oracle with its key pair (SK, PK), $\prod_{U,P}^{i}$ be the user oracle simulating a set of all registered users, interacting with the service provider oracle in session i, and $\prod_{P,U}^{j}$ be the service provider oracle simulating a set of all registered service providers, interacting with the user oracle in the session j. A probabilistic polynomial time (PPT) adversary A can ask the following oracle queries.

1) $\mathcal{O}_1$: $Register(\prod, U)$— Upon receiving this query, the $\prod_{TCP}$ will run $IdGen(RI_{A_i})$ and $CGen(ID_{A_i}, SK)$ algorithms, and output a new user identity ID_{A_i} with corresponding credential C_{A_i} to A who can verify the credential by running $CVer(\cdot)$.
2) $\mathcal{O}_2$: $Register(\prod, P)$— Upon receiving this query, the system will run $Init(\lambda_1)$ and output P_{A_j}'s private/public key pair (SK_{A_j}, PK_{A_j}) together with identity SID_{A_j} to A.
3) $\mathcal{O}_3$: $Execute(U_i, P_j)$— Upon receiving this query, $\prod_{U,P}^{i}$ and $\prod_{P,U}^{j}$ will execute protocol as U_i and P_j in $\prod$, respectively. The exchanged messages between them will be recorded and sent to A. Here, we require that both U_i's credential and P_j's private key are not been corrupted by A via $\mathcal{O}_1$ and $\mathcal{O}_2$ oracles.
4) $\mathcal{O}_4$: $Send(U_i, m, f)$—This query sends the message m as message flow $f \in \{0, 1, \cdots, n\}$ to the user oracle $\prod_{U,P}^{i}$ which simulates a user U_i, and then, the oracle computes message honestly in $\prod$, and sends responses back to A, where n is the total number of messages transmitted in protocol $\prod$. If a user is the protocol initiator by default, A can also start a new session by asking $Send(U_i, \emptyset, 0)$, where $\emptyset$ denotes an empty set.
5) $\mathcal{O}_5$: $Send(P_j, m, f)$—This query sends the message m as message flow $f \in \{0, 1, \cdots, n\}$ to the user oracle $\prod_{P,U}^{i}$ which simulates a service provider P_j, and then, the oracle computes message honestly in $\prod$, and sends responses back to A. If a service provider is the protocol initiator by default, A can also start a new session by asking $Send(P_j, \emptyset, 0)$.
6) $\mathcal{O}_6$: $Reveal(\prod, i)$—This query models the leakage of session key in session i. This query only can be asked when a session key has been shared between a service provider and a user in session i.

Remark 3. *$\mathcal{O}_3$ simulates the real environment for a passive attacker A who can eavesdrop all messages exchanged between U_i and P_j when executing protocol $\prod$. If A knows U_i's credential C_i and P_j's private key SK_j, oracle $\mathcal{O}_3$ is not necessary as A can run protocol $\prod$ by itself on behalf of them. If A knows one of these two secrets but not both, A can run protocol $\prod$ with U_i (P_j) whose secret is not released via executing oracle $\mathcal{O}_4$ ($\mathcal{O}_5$).*

Remark 4. *$\mathcal{O}_4$ simulates the real environment for an active attacker A who may obtain a service provider P_j's private key SK_j, send message m as message flow $f \in \{0, 1, \cdots, n\}$ to a target user U_i and then get the corresponding response. To answer this oracle, U_i will generate his/her response according to the specification of protocol $\prod$ and sends it to A. Notes that if U_i did not receive all necessary previous messages that match this message with message flow f, this oracle request will be rejected, since it is meaningless in the view point of U_i. Actually, $\mathcal{O}_4$ also provides adversary A oracle access on algorithm $UPGen(\cdot)$ since $\prod_{U,P}^{i}$ will run $UPGen(\cdot)$ somehow in executing $\prod$. In our construction, $UPGen(\cdot)$ is Schnorr signature generation algorithm. In this case, on the one hand, oracle $\mathcal{O}_4$ may be not stronger than the signing oracle in Game-UFCMA reviewed in section IV, since the temporal message M, one input of algorithm $UPGen(\cdot)$, may be jointly decided by U_i and A (playing the role of one P_j), rather than just by A. So, it may be hard for A to get U_i's user proof for any arbitrary message M. On the other hand, adversary A may be not weaker than the forger in Game-UFCMA since besides $\mathcal{O}_4$ we also offer other oracle queries, which may increase A's ability. We omit a similar remark which applies to $\mathcal{O}_5$.*

To formally define the soundness and credential privacy, we first discuss the difference between soundness and credential privacy since the majority of existing schemes only consider the credential privacy. The credential privacy requires unforgeability and irrecoverableness. The former guarantees that any PPT adversary A has only a negligible probability for successfully forging a valid credential C_t of a target user U_t in the credential generation phase, while the latter requires that in user authentication phase, any A can only recover C_t with a negligible probability. Soundness is also critical in the user authentication phase as it ensures that any A without a valid credential can only generate a user proof up that passes through user authentication with a negligible probability. The existing studies [16], [12] only focus on if a valid credential can be forged or recovered by attackers, but do not consider if a valid credential is definitely necessary for generating a valid user proof. We shall define these three properties as a single definition (but one for users and one for service providers).

Let $A^{\mathcal{O}}$ denotes an adversary A who has access to all oracle queries in $\mathcal{O} = \{\mathcal{O}_i | i = 1, 2, \cdots, 6\}$ in adversary model; let the credential holder U_i with identity ID_i and credential C_i, and the service provider P_j with identity SID_j and key pair (SK_j, PK_j) are two polynomial-time Turing machines. Let U_i and P_j interact with each other, and place A between U_i and P_j. ϵ denotes a negligible function. We define secure credential based user authentication as follows:

Definition 3. ***(Secure credential based user authentication ($SCUA$))*** *An AKESSO scheme achieves secure credential based user authentication, if any PPT adversary A has a negligible advantage $Adv^{SCUA}(A^{\mathcal{O}})$ for creating a valid user proof without holding the corresponding credential. Formally,*

for any PPT A, $Adv^{SCUA}(A^{\mathcal{O}}) \triangleq \Pr[(ID_t, up_t, M) \leftarrow A^{\mathcal{O}} | UPVer(up_t, ID_t, PK, M) = 1] \leq \epsilon$ *with the following restrictions:*

- *A has not obtained the credential* C_t *corresponding to* ID_t *via* $\mathcal{O}_1$ *-* $Register(\prod, U)$ *oracle; and*
- *A has not obtained any valid user proof* up'_t *for message* M *by asking any oracle in* $\mathcal{O}$*, in particular* $\mathcal{O}_3$ *and* $\mathcal{O}_4$*.*

Similarly, the definition of secure service provider authentication is given as below:

Definition 4. ***(Secure service provider authentication (*** $SSPA$***))*** *An AKESSO scheme achieves secure service provider authentication, if any PPT adversary A has a negligible advantage* $Adv^{SSPA}(A^{\mathcal{O}})$ *for forging a valid service provider proof without holding the corresponding service provider's private key . Formally, for any PPT A,* $Adv^{SSPA}(A^{\mathcal{O}}) \triangleq \Pr[(PK_t, M', spp_t) \leftarrow A^{\mathcal{O}} | SPPVer(PK_t, M', spp_t) = 1] \leq \epsilon$ *with the following restrictions:*

- *A has not obtained the private key* SK_t *corresponding to* SID_t *via* $\mathcal{O}_2$ *-* $Register(\prod, P)$ *oracle;*
- *A has not obtained any valid service provider proof* spp_t *for message* M' *by asking any oracle in* $\mathcal{O}$*, in particular* $\mathcal{O}_3$ *and* $\mathcal{O}_5$*.*

Here, we review the freshness and test query $Test(\prod, i)$ for defining session key security [17]. An adversary can get session keys by asking $\mathcal{O}_6$. We say the session key is *fresh* if and only if the $\mathcal{O}_6$ query has not been asked w.r.t. this session. In other words, the fresh session key must be unknown to the adversary. For simplicity, we call the test query as $\mathcal{O}_7$, which is a game defined as follows:

- $\mathcal{O}_7$ — $Test(\prod, i)$: In protocol $\prod$, if $\prod^i_{U,P}$ and $\prod^i_{P,U}$ accept and share the same fresh session key in session i, upon receiving this query, by tossing a coin b the correct session key is returned if $b = 0$, otherwise, a random session key is returend. A only can ask this query one time and A needs to output one bit b' as the result of guessing b. A's advantage in attacking the session key security (SKS) of protocol $\prod$ is defined as $Adv^{SKS}_{\prod}(A^{\mathcal{O}'}) = |2\Pr[b' = b] - 1|$, where $\mathcal{O}' = \mathcal{O} \cup \{\mathcal{O}_7\}$.

Session key security [17] models adversary A's inability to distinguish the real session key and a random string, as formally defined below.

Definition 5. ***(Session Key Security)*** *We say an AKESSO satisfies session key security if for any PPT adversary A,* $Adv^{SKS}_{\prod}(A^{\mathcal{O}'}) \leq \epsilon$*, where* $\mathcal{O}' = \mathcal{O} \cup \{\mathcal{O}_7\}$*.*

Finally, we can give the definition of secure authenticated key exchange single sign-on scheme.

Definition 6. ***(Secure Authenticated Key Exchange Single Sign-On Scheme)****: An* $AKESSO$ *scheme is called secure if it is correct and satisfies* $SCUA$*,* $SSPA$*, and session key security.*

III. Review of Chang-Lee's scheme

In 2012, Chang and Lee [12] proposed an improved efficient remote user identification scheme for mobile device users, the scheme employs single sign-on technique, supports session key establishment, and preserves user anonymity. However, the scheme neither provides credential privacy nor soundness due to [6]. In this section, We briefly reviews the Chang-Lee scheme and its drawbacks.

A. Review of the Scheme

Chang-Lee's SSO scheme consists of three phases: system initialization, registration, and user identification. The details are as follows.

1) System Initialization Phase: The trusted authority TCP determines the RSA key pair (e, d) and a generator g, and publishes public parameters.

2) Registration Phase: In this phase, the trusted authority signs an RSA signature $S_i = (ID_i || h(ID_i))^d \mod N$ to user U_i as the credential. For each service provider P_j, he needs to maintain his own RSA public parameters (ID_j, e_j, N_j) and private parameter d_j similar as TCP.

3) User Identification Phase: In this phase, the session key is $K_{ij} = h(ID_i || k_{ij})$, where k_{ij} is the plain Diffie-Hellman session key. For identifying service providers, an RSA signature scheme has been used; for user authentication, the user need to provide a proof $z = S_i^{h(K_{ij}||k_2||n_2)} \mod N$ of credential S_i, where k_2 is user's session key material and n_2 is a random nonce selected by the user. For the purpose of anonymity, the random nonce n_3 and user identity which used for proof checking has been encrypted via symmetric key encryption scheme with session key K_{ij} (treated as encryption key). The user can pass authentication if $z^e \mod N = SID_i^{h(K_{ij}||k_2||n_2)} \mod N$ dose hold, and the user believes that they are share the same session key if the hashed n_3 has been received.

B. Review of Attacks

Two high risky attacks are identified in [6] on Chang-Lee scheme. The former allows a malicious P_j to recover user credential; the latter enables an adversary passing user authentication without a valid credential. They are briefly reviewed below.

1) Credential Recovering Attack: A user U_i can pass authentication if he provides the valid proof z of knowledge C_i. To simplify the discussion, we use h_2 to denote $h(K_{ij}||k_2||n_2)$. So proof $z = S_i^{h_2}$. It is easy to see that for different proofs in different session, the same credential S_i has been encrypted multiple times with different h_2 but the same modulo N. Thus, if a malicious P_j has been accessed twice with the same user U_i, then P_j is able to recover U_i's credential S_i by using extended Euclidean algorithm. Let us suppose that (z', z'') and (h'_2, h''_2), the proofs and hash values in two different sessions, satisfy $gcd(h'_2, h''_2) = 1$. Then we can find two integers a and b such that $a \cdot h'_2 + b \cdot h''_2 = 1$ (in $\mathbb{Z}$) due to the extended Euclidean algorithm. Finally, the P_j can recover user credential by computing $z'^a \cdot z''^b$

$\mod N = S_i^{h_2' \cdot a + h_2'' \cdot b} \mod N = S_i$. The success rate of this attack is about 60% [6].

2) Impersonation Attack without Credentials: A small RSA public key e has been assumed in this attack, where the "small" requires the binary length of e is much less than the output length of hash function h. The rationality of this assumption is given in [6]. In the conversation, if the h_2 is divisible by e, then the adversary computes an integer b such that $h_2 = e \cdot b$, and calculates proof z by $z = SID_i^b$, where $SID_i = ID_i || h(ID_i)$. The verification holds as $SID_i^{h_2} \mod N = SID_i^{b \cdot e} \mod N = z^e \mod N$. Thus, the adversary can pass user authentication without a valid credential. The success rate of the attack is about $1/e$ [6].

IV. Review of Schnorr Signature

As one of the simplest, shortest, and frequently used signature schemes, Schnorr signature scheme [18], [13] is provably secure in a random oracle model under the assumption that discrete logarithm problem is intractable [19], [20], [21], [15]. We now review Schnorr signature scheme as follows.

Initialisation: The scheme is defined in a cyclic group G of order q with a generator $g \in \mathbb{Z}_p^*$, were p and q are primes such that $q|p-1$, $q \geq 2^{160}$, and $p \geq 2^{1024}$. A secure hash function $h(\cdot)$ is also selected.

Signature Generation: To sign a message m with private key $x \in \mathbb{Z}_q^*$, a signer picks a randomness $r \in \mathbb{Z}_q^*$, and outputs the signature (a, e, s) by computing $a = g^r \mod p$, $e = h(a, m)$, and $s = r + x \cdot e \mod q$.

Signature Verification: Given a signature (a, e, s) for message m w.r.t. public key $y = g^x \mod p$, the verifier accepts this signature iff $e \equiv h(a, m)$ and $g^s \equiv ay^e \mod p$.

Let us denote $Init(\lambda)$, $SGen(\cdot)$ and $SVer(\cdot)$ the initialisation algorithm, signing algorithm and verification algorithm, respectively. Formally, a signature scheme is called **existentially unforgeable** if for any PPT forgery algorithm A, it can only win the following game, called **Game-UFCMA**, with a negligible probability [22][23].

- **Setup:** $(pk, sk) \leftarrow Init(\lambda)$. Given a security parameter λ, a public/private key pair is generated by the initialisation algorithm and adversary A is given the public key pk.
- **Query:** $\sigma_i \leftarrow SGen(sk, m_i)$. A runs up to q times to ask the signature signing oracle in an adaptive manner. Each time, the signing oracle will reply a signature σ_i for each message m_i chosen by A, where $1 \leq i \leq q$.
- **Forge:** A outputs a new message and signature pair (m_j, σ_j). A wins if
  1) $SVer(pk, m_j, \sigma_j) = 1$, i.e., σ_j is a valid signature for message m_j under the public key pk.
  2) $m_j \neq m_i$, for any $i \in \{1, \cdots, q\}$.

V. Proposed scheme

This section presents a secure single sign-on scheme with user anonymity for remote user authentication in distributed systems and networks. We use Schnorr signature [18][13] to overcome the drawbacks in Chang-Lee scheme as their user proof cannot provide soundness and credential privacy while Schnorr signature can. As a proveably unforgeable signature scheme [21], Schnorr signature allows a signer to authenticate him/herself by signing a message without releasing any other useful information about his/her private signing key. In the proposed scheme, the TCP first issues the credential for each user by signing the user's identity ID_i according to Schnorr signature. Then, by treating his/her credential as another public/private key pair the user can authenticate him/herself by signing a Schnorr signature on a temporal message generated in the protocol. In contrast, each service provider maintains its own public/private key pair in any secure signature scheme so that it can authenticate itself to users by simply issuing a normal signature. Finally, as does in Chang-Lee scheme [12], the session key is established by running a variant of Diffie-Hellman key exchange protocol, and the user anonymity is guaranteed by symmetric key encryption. The notations used in the scheme are summarised in Table I.

TABLE I
Notations used in the scheme

TCP	The trusted credential provider
P_j	A service provider
U_i	A user
SID_j	The unique identity of P_j
ID_i	The unique identity of U_i
C_i	The credential of U_i
x	The long term private key of TCP
y	The public key of TCP
$E_k(M)$	Symmetric encryption of message M using key k
$D_k(C)$	Symmetric decryption of ciphertext C using key k
$h(\cdot)$	A secure hash function

System Setup Phase: In this phase, TCP initializes his/her public and private parameters as Schnorr signature scheme. Firstly, TCP picks large primes p and q such that $q|p-1$, chooses a generator g of large safe prime order q in cyclic group G. Then, TCP sets its private key $SK = x$, where $x \in \mathbb{Z}_q^*$ is a random number, and publishes its public key $PK = y$, where $y = g^x \mod p$.

Registration Phase: In this phase, user asks TCP for registration, then TCP issues a unique identity ID_i via $IdGen(RI_i)$ and signs a Schnorr signature (a, e, C) for user's identity as credential generation algorithm $CGen(ID_i, SK)$. C is kept secret by user, while (a, e) will be made public. The details are given below.

- User Registration: When a user U_i asks for registration, TCP selects a unique identity ID_i and generates a credential $C_i = (a, e, C)$ for U_i by selecting a randomness $r \in \mathbb{Z}_q^*$ and computing $a = g^r \mod p$, $e = h(a, ID_i)$, and $C = r + xe \mod q$. Then, TCP sends identity ID_i and credential C_i which is Schnorr signature for ID_i to user U_i, where C should be kept as a secret.
- Service Provider Registration: Each P_j maintains a public/private key pair (PK_j, SK_j) of any secure signature scheme. Here, algorithms $SPPGen(\cdot)$ and $SPPVer(\cdot)$ are identical to the signature generation and verification algorithms respectively.

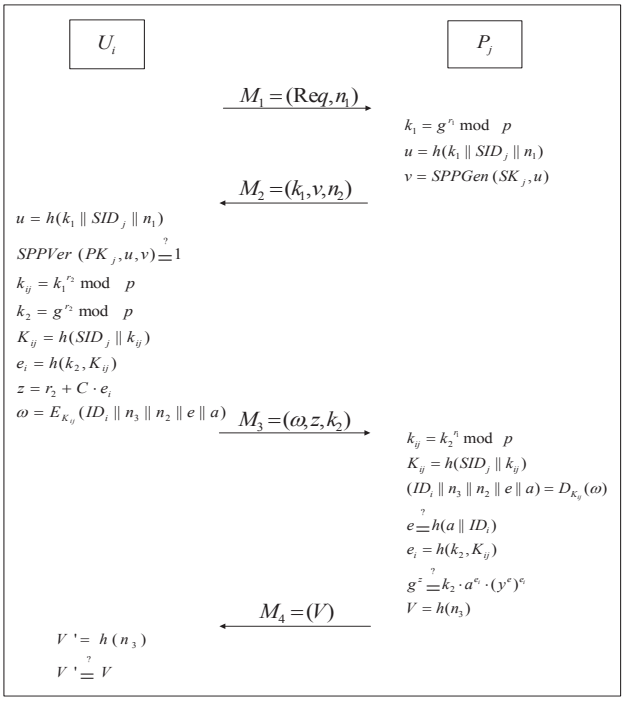

Fig. 1. Participant Identification Phase

Authentication Phase: In this phase, to authenticate him/herself user U_i signs a Schnorr signature the newly established session key K_{ij} using credential C the signing key, while U_i's session key material k_2 is used as the commitment. Note that the corresponding verification key of C is g^C, which can be recovered by computing $g^C = a \cdot y^e \mod p$. For service provider authentication, any provably secure signature scheme can be used to authenticate a service provider in proposed scheme. The session key is established by using modified Diffie-Hellman key exchange scheme which has been formally proved in [12], and the user anonymity and unlinkability are preserved by using symmetric key encryption to encrypt a, e, and user's identity ID_i. The details of this phase are illustrated in Figure 1 and further explained below.

1) User U_i chooses a random nonce n_1 and sends $M_1 = (Req, n_1)$ to P_j, where Req is a service request.
2) Upon receiving (Req, n_1), P_j picks random number $r_1 \in \mathbb{Z}_q^*$, computes its session key material $k_1 = g^{r_1} \mod p$, $u = h(k_1||SID_j||n_1)$ and signs u to get a signature $v = SPPGen(SK_j, u)$, and sends $M_2 = (k_1, v, n_2)$ to the user.
3) User U_i first computes $u = h(k_1||SID_j||n_1)$ and verifies the signature v by checking if $SPPVer(PK_j, u, v) = 1$. If the output is "0", U_i terminates the protocol. Otherwise, U_i accepts the service provider P_j's authentication, and then selects a random number $r_2 \in \mathbb{Z}_q^*$ to compute $k_2 = g^{r_2} \mod p$, $k_{ij} = k_1^{r_2} \mod p$, and the session key $K_{ij} = h(SID_j||k_{ij})$. After that, U_i signs K_{ij} using his/her credential secret C by calculating $e_i = h(k_2, K_{ij})$, $z = r_2 + Ce_i \mod q$ and $\omega = E_K(ID_i||n_3||n_2||e||a)$, where n_3 is a nonce chosen by U_i. Finally, U_i sends $M_3 = (\omega, z, k_2)$ to service provider P_j.
4) To verify z, P_j first calculates $k_{ij} = k_2^{r_1} \mod p$, derives session key $K_{ij} = h(SID_j||k_{ij})$ and decrypt ω with K_{ij} to recover $ID_i||n_3||n_2||e||a$. Then, P_j checks if $e = h(a||ID_i)$. If this does not hold, P_j aborts the protocol. Otherwise, the service provider computes $e_i = h(k_2, K_{ij})$ and verifies z by checking if $g^z = k_2 \cdot a^{e_i} \cdot (y^e)^{e_i} \mod p$. If this holds, P_j accepts U_i's authentication, believes that they have shared the same session key K_{ij}, and sends $V = h(n_3)$ as M_4 to U_i.
5) User U_i computes $V' = h(n_3)$ and checks if $V' = V$. If this holds, U_i believes that he/she has shared the same session key K_{ij} with P_j.

VI. Security Analysis

The proposed scheme employs Schnorr signature scheme [18][13] to generate credentials for users, uses modified Diffie-Hellman key exchange scheme to establish the session key, signs a Schnorr signature on the hashed session key for user authentication, uses any secure signature scheme for server authentication, and takes symmetric key encryption to ensure user anonymity. The secure authenticated key exchange single sign-on (AKESSO) scheme requires secure credential based user authentication ($SCUA$), secure service provider authentication ($SSPA$), and secure session key. To prove the security of proposed AKESSO, we will just prove $SCUA$ and $SSPA$ because (1) the proposed scheme only improves parts of key generation, user authentication and service provider authentication in Chang-Lee scheme [12], while the parts of user anonymity and session key establishment have not been modified; and the user anonymity and session key security have been proved in [12] and discussed in [6] without revealing any problems. Now, we start to formally analyse the security of the proposed AKESSO scheme.

Theorem 1. *(Correctness) The proposed construction is a correct AKESSO scheme according to Definition 2.*

Proof: This can be straightforwardly verified according to Definition 2 given in Section II. ■

Informally, the proposed AKESSO scheme guarantees $SSPA$ as each service provider employs a secure signature scheme. To prove $SCUA$, we need to show that Definition 3 holds for the proposed AKESSO scheme by assuming the unforgeability of Schnorr signature scheme.

Theorem 2. *(Secure Credential based User Authentication) In proposed AKESSO scheme, if there is an PPT adversary A who has a non-negligible advantage $Adv^{SCUA}(A^{\mathcal{O}})$ as specified in Definition 3, then Schnorr signature scheme is existentially forgeable under $UFCMA$ attacks as defined in Section IV.*

Proof: As adversary A, with access to all oracles in $\mathcal{O} = \{\mathcal{O}_1, \cdots, \mathcal{O}_6\}$, has a non-negligible advantage

$Adv^{SCUA}(A^{\mathcal{O}})$, according to Definition 3 this implies that at least one of the following two cases is true:

- Case (1): With a non-negligible probability ϵ_1, $A^{\mathcal{O}}$ is able to derive a credential C_t corresponding to an *unregistered* target identity ID_t.
- Case (2): With a non-negligible probability ϵ_2, $A^{\mathcal{O}}$ is able to forge a valid user proof for a new message M w.r.t. a *registered* target identity ID_i.

Now, we will prove that if either Case (1) or Case (2) is true, we can construct an algorithm B that is able to break the unforgeability of Schnorr signature, where B runs $A^{\mathcal{O}}$ as a sub-program for fulfilling its purpose.

Case (1). Suppose that B is given a target Schnorr signature scheme with parameter $(p, q, h(\cdot))$ and public key $y = g^x \mod p$, where the private key x is not known to B. B's strategy for wining Game-UFCMA with non-negligible probability is to set up an AKESSO scheme for A and to simulate oracles in $\mathcal{O}$ such that A cannot distinguish the difference between this simulated environment and a real AKESSO scheme. Therefore, A will be able to successfully derive a credential C_t for an unregistered identity ID_t with probability ϵ_1. After that, B can adapt this credential into a forged Schnorr signature for a new message and thus break the unforgeability of Schnorr signature scheme.

Now we describe how B sets up such a simulated AKESSO scheme for A. First, B sets y as the public key of TCP and gives y to B. Then, each oracle in $\mathcal{O}_i$ $(i = 1, \cdots, 6)$ can be simulated as follows. To simulate $\mathcal{O}_1$ query B can ask its own signing oracle to get a Schnorr signature C_i for each identity ID_i and then reply (ID_i, C_i) to A. To simulate $\mathcal{O}_2$ query B can simply run $Init(\lambda_1)$ to get a public/private key pair (SK_j, PK_j) for an identity SID_j, and then forwards (SID_j, SK_j, PK_j) to A. As B knows all users' credentials and all service providers's private keys, it can simulate oracles $\mathcal{O}_3$, $\mathcal{O}_4$, $\mathcal{O}_5$ and $\mathcal{O}_6$ by trivially executing the whole protocol $\prod$, running one move on behalf of a user, running one move on behalf of a service provider, and revealing a session, respectively. Note that as ID_t is an unregistered identity in this case, the corresponding user U_t will not be involved in any oracle $\mathcal{O}_i$ $(i = 1, \cdots, 6)$.

It is not difficult to see that the above simulated system is indistinguishable from a real system in the view point of A. Hence, A will be able to output a credential C_t for target identity ID_t with non-negligible probability ϵ_1, where ID_t is not asked in $\mathcal{O}_1$ queries. Therefore, B will simply forward C_t as a forged Schnorr signature for message ID_t. Since ID_t is not asked in $\mathcal{O}_1$ queries, A does not ask ID_t in its signing oracle, i.e., ID_t is a new message for B. So, B's forged message-signature pair (ID_t, C_t) is valid according to the definition of Game-UFCMA (refer to Section IV). Moreover, B's success rate is exactly the same as A's, i.e., ϵ_1, which is non-negligible. Consequently, this means that B successfully breaks the unforgeablity of Schnorr signature scheme.

Case (2). This can be proved similarly as Case (1) but B will embed its target Schnorr signature scheme in the user proof generation algorithm for a registered target user U_t with identity ID_t. Details are given as follows.

Suppose that B is given a target Schnorr signature scheme with parameter $(p, q, h(\cdot))$ and public key $y' = g^{x'} \mod p$, where the private key x' is not known to B. First, B sets $y = g^x \mod p$ as the public key of TCP by selecting a random number x as TCP's private key. For any identity ID_i except target identity ID_t, to answer an $\mathcal{O}_1$ query B can directly issue a credential C_i for ID_i by generating a Schnorr signature for ID_i as B knows TCP's private key x. In contrast, B will take (a', e', x') as the credential C_t for target identity ID_t, where $e' \in \{0, 1, \cdots, q-1\}$ is a random number, $a' \in \mathbb{Z}_p^*$ is set as $a' = y' \cdot y^{-e'} \mod p$, and $h(a', ID_t)$ is set as e'. So, we have $g^{x'} = a' y^{h(e', ID_t)} \mod p$. Note that B does not know the value of x' and it will be not required to reveal C_t to A because ID_t is the target identity. In addition, here we can artificially fix the hash value for such a special input (a', ID_t) because Schnorr signature is secure in random oracle where hash function can be viewed as an random function [21]. All other oracles in $\mathcal{O}$ can be simulated as in Case (1), except A asks $\mathcal{O}_3$ and $\mathcal{O}_4$ queries in which U_t with identity ID_t is involved. In such scenarios, B can simulate U_t to output a valid user proof up_t w.r.t. credential C_t by executing the whole protocol $\prod$ or running one move with necessary help from its own signing oracle w.r.t. public key y'.

Again, it is not difficult to see that the above simulated system is indistinguishable from a real system in the view point of A. Hence, with probability ϵ_2 A will be able to output a valid user proof up_t for a message M w.r.t. target identity ID_t, where M is not asked in $\mathcal{O}_3$ and $\mathcal{O}_4$ queries. Therefore, B can simply forward up_t as a forged Schnorr signature for message M. Since M is not asked in $\mathcal{O}_3$ and $\mathcal{O}_4$ queries, A does not ask M in its signing oracle, i.e., M is a new message for B. So, B's forged message-signature pair (up_t, M) is valid according to the definition of Game-UFCMA (refer to Section IV). Moreover, B's success rate is exactly the same as A's, i.e., ϵ_2, which is non-negligible. Consequently, this means that B successfully breaks the unforgeablity of Schnorr signature scheme. ■

Remark 5. *In Case (1), $A^{\mathcal{O}}$ could directly forge C_t, recover C_t after executing protocol $\prod$ with user U_t or eavesdropping the transcripts between U_t and some service providers, or derive C_t in any other possible way, though $A^{\mathcal{O}}$ is not allowed to obtain C_t by trivially asking $\mathcal{O}_1$ oracle w.r.t. ID_t. Hence, this means that if our AKESSO fails to satisfy the unforgeablity or unrecoverableness of credential, then Schnorr signature is forgeable. Similarly, in Case (2) $A^{\mathcal{O}}$ could directly forge a user proof up_t without credential C_t, observe and adapts existing user proofs generated by U_t into a user proof up_t for a message M, or compute up_t in any other way, though $A^{\mathcal{O}}$ is not allowed to obtain any user proof for the same message M by trivially asking $\mathcal{O}_3$ and $\mathcal{O}_4$ oracles w.r.t. ID_t. Hence, this implies that if our AKESSO fails to satisfy soundness of credential based authentication [6], then Schnorr signature is forgeable.*

As Schnorr signature scheme is proved to be secure under the discrete logarithm assumption [21], Theorem 2 assures that the proposed AKESSO scheme achieves secure credential based user authentication under the discrete logarithm assumption.

Theorem 3. *(Secure Service Provider Authentication) In proposed AKESSO, if there is an PPT adversary A who has a non-negligible advantage $Adv^{SSPA}(A^{\mathcal{O}})$ as specified in Definition 4, then signature signature scheme employed by service providers is existentially forgeable under $UFCMA$ attacks as defined in Section IV.*

Proof: Since a service provider proof is directly generated as a normal signature by the corresponding service provider, Theorem 3 can be formally proved as we did for Case (2) in Theorem 1. Note that here we do not need to discuss Case (1) as in Theorem 1, because each service provider is required to register its public/private key pair. Due to space limit, the full proof is omitted. ■

Theorem 4. *According to Definition 6, the proposed AKESSO scheme is secure under the assumption that all digital signatures employed in the scheme are existentially unforgeable against $UFCMA$ attacks as specified in Section IV.*

Proof: By Theorem 1, Theorem 2, Theorem 3 and session key security proved in [12], Theorem 4 holds according to Definition 6. ■

VII. Conclusions

Most existing single sign-on schemes suffer from various security issues and are vulnerable to different attacks. In this paper, we first formalized authenticated key exchange single sign-on scheme. Specially, we formally defined secure authentication for both users and service providers as such a treatment has not been studied yet [6]. Moreover, a Schnorr mechanism based SSO scheme has been proposed to overcome the drawbacks of Chang-Lee scheme [12] but keep the same advantages. In this new scheme, to preserve credential generation privacy, the TCP signs a Schnorr signature [18][13] on user identity; and to protect credential privacy and soundness, the user exploits his/her credential as a signing key to sign a Schnorr signature on the hashed session key. In fact, Schnorr signature mechanism [18][13] is more efficient than RSA mechanism which has been employed by Chang-Lee scheme. Thus, the proposed scheme reduces the computation cost, enhances the confidentiality, and preserves soundness and credential privacy.

References

[1] A. C. Weaver and M. W. Condtry, "Distributing Internet Services to The Networks Edge", *IEEE Trans. Ind. Electron.*, vol. 50, no. 3, pp. 404-411, Jun. 2003.

[2] L. Barolli and F. Xhafa, "JXTA-OVERLAY: A P2P Platform for Distributed, Collaborative and Ubiquitous Computing", *IEEE Trans. Ind. Electron.*, vol. 58, no. 6, pp. 2163-2172, Oct. 2010.

[3] L. Lamport, "Password Authentication with Insecure Communication", *Commun. ACM*, vol. 24, no. 11, pp. 770-772, Nov. 1981.

[4] F. Bao, R. H. Deng, "Privacy Protection for Transactions of Digital Goods", *Proceedings of the Third International Conference on Information and Communications Security (ICICS '01)*, Springer-Verlag, London, UK, pp. 202-213.

[5] The Open Group, "Security Forum on Single Sign-on", http://www.opengroup.org/security/l2-sso.htm.

[6] G. Wang, J. Yu, and Q. Xie, "Security Analysis of A Single Sign-On Mechanism for Distributed Computer Networks", IACR Cryptology ePrint Archive, Report 2012/107, http://eprint.iacr.org/2012/107.

[7] W. B. Lee and C. C. Chang, "User Identification and Key Distribution Maintaining Anonymity for Distributed Computer Networks", *Computer Systems Science and Engineering*, vol. 15, no. 4, pp. 113-116, 2000.

[8] T.-S. Wu and C.-L. Hsu, "Efficient User Identification Scheme with Key Distribution Preserving Anonymity for Distributed Computer Networks", *Computers and Security*, vol. 23, no. 2, pp. 120-125, 2004.

[9] Y. Yang, S.Wang, F. Bao, J.Wang, and R. H. Deng, "New Efficient User Identification and Key Distribution Scheme Providing Enhanced Security", *Computers and Security*, vol. 23, no. 8, pp. 697-704, 2004.

[10] K. V. Mangipudi and R. S. Katti, "A Secure Identification and Key Agreement Protocol with User Anonymity (sika)", *Computers and Security*, vol. 25, no. 6, pp. 420-425, 2006.

[11] C.-L. Hsu and Y.-H. Chuang, "A Novel User Identification Scheme with Key Distribution Preserving User Anonymity for Distributed Computer Networks", *Inf. Sci.*, vol. 179, no. 4, pp. 422-429, 2009.

[12] C.-C. Chang and C.-Y. Lee, "A Secure Single Sign-on Mechanism for Distributed Computer Networks", *IEEE Transactions on Industrial Electronics*, vol. 59, no. 1, pp. 629-637, 2012.

[13] C.P. Schnorr, "Efficient Signature Generation by Smart Cards", *J. Cryptology*, vol. 4, no. 3, pp. 161-174, 1991.

[14] S. Goldwasser, S. Micali, and C. Rackoff, "The Knowledge Complexity of Interactive Proof-Systems", *SIAM J. Computing,* vol. 18, no. 1, pp. 186-208, Feb. 1989.

[15] W. Mao, *Modern Cryptography: Theory and Practice*, Prentice Hall PTR, 2004.

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

[17] M. Bellare and P. Rogaway, "Entity Authentication and Key Distribution", *CRYPTO*, pp. 232-249, 1993.

[18] C.P. Schnorr, "Efficient Identification and Signatures for Smart Cards", *CRYPTO* ,pp. 239-252, 1989.

[19] M. Bellare and A. Palacio, "GQ and Schnorr Identification Schemes: Proofs of Security against Impersonation under Active and Concurrent Attacks", *CRYPTO*, pp. 162-177, 2002.

[20] D. Pointcheval, J. Stern, "Security Proofs for Signature Schemes", *EUROCRYPT*, pp. 387-398, 1996.

[21] D. Pointcheval, J. Stern, "Security Arguments for Digital Signatures and Blind Signatures", *J.Cryptology*, vol.13, no.3, pp. 361-369, 2000.

[22] S. Goldwasser, S. Micali, and L. Ronald, "A "Paradoxical" Solution to the Signature Problem (Extended Abstract)", *FOCS*, pp. 441-448, 1984.

[23] S. Goldwasser, S. Micali, and R. L. Rivest, "A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks", *SIAM J. Comput.,* vol. 17, no. 2, pp. 281-308, 1988.