A Unlinkable Delegation-based Authentication Protocol with Users' Non-repudiation for Portable Communication Systems

Shin-Jia Hwang[1] Cheng-Han You[1]

[1] Department of Computer Science and Information Engineering, Tamkang University

Tamsui, New Taipei City, 251, Taiwan, R.O.C.

{sjhwang@mail, 699420435@s99}.tku.edu.tw

Abstract. For portable communication systems, the delegation-based authentication protocol provides efficient subsequent login authentication, data confidentiality, user privacy protection, and non-repudiation. However, in all proposed protocols, the non-repudiation of mobile users is based on an unreasonable assumption that home location registers are always trusted. To weaken this assumption and enhance the non-repudiation of mobile users, a new delegation-based authentication protocol is proposed. The new protocol also removes the exhaustive search problem of the subsequent login authentication to improve the subsequent login authentication performance. Moreover, the user unlinkability in the subsequent login authentication is also provided to enhance the user identity privacy protection.

Keywords: Delegation-based authentication, concurrent signatures, concurrent signcryption scheme, portable communication systems

Received 23 August 2013; Revised 10 November 2013; Accepted 16 December 2013

1 Introduction

Portable communication systems (PCSs) provide roaming services among wireless communication networks. A mobile user (MU) first registers his/her legality in some home location register (HLR). Before roaming, MU logins some visiting location register (VLR) and VLR validates the user's legality with the help of the HLR. If MU is legal of some HLR, VLR provides services and charges the roaming fee.

The global system for mobile communications (GSM) has some drawbacks [1]: No non-repudiation property, no users' identity privacy, and no mutual authentication between users and VLR. Some symmetric-cryptosystem-based protocols [2, 3] try to improve the GSM protocol, but they cannot achieve non-repudiation property. Some public-key-cryptosystem-based protocols are proposed [4, 5] to provide both non-repudiation and mutual authentication services by paying heavy computation cost.

For PCSs, Lee and Yeh [1] first proposed the delegation-based authentication protocol that exhibits off-line authentication processes to reduce the communication load between VLR and HLR; and mobile users' computation cost. Though their protocol satisfies the mutual authentication, data secrecy, and identity privacy, the protocol does not satisfy the non-repudiation property in off-line authentication processes [6] during roaming services. To overcome this flaw, Lee proposed an enhanced protocol. Unfortunately, Lee's protocol suffers the linkable problem [7-8] that two distinct roaming instances belonging to the same MU can be linked by the same MU's identity. To remove the linkable problem, Youn and Lim's [7], and Wang et al.'s [8] protocols are proposed. However, Wang et al.'s protocol suffers forgery attack [9]. Both those two protocols also suffer the exhausted search problem in off-line authentication [10]. To overcome the exhaustive search problem, Chen et al. [10] proposed their improvement.

In the proposed delegation-based authentication protocols [1, 6-8, 10], HLR knows any authorized MU's proxy public key and proxy private key. MU's proxy private key is used to generate the proxy signatures for non-repudiation property. To avoid misusing of the proxy public and private key pair and to provide MUs' non-repudiation, HLR must be always trustworthy.

However, it is hard to guarantee that the HLR is always trustworthy, because some staffs of the HLR may be malicious. After stealing the proxy private keys, the malicious staffs easily forge the proxy signatures for some mobile user for roaming. Then the mobile user owning the stolen proxy private key cannot deny the roaming records caused by the malicious staffs. So the trust HLR assumption weakens the mobile users' non-repudiation.

1.1 Our Contribution

The mobile users' non-repudiation is enhanced by reducing the trust HLR assumption to semi-trust HLR assumption that HLR does not know the proxy private key of any MU. So our new delegation-based authentication protocol for PCSs is proposed by adopting the concept of Hwang and Sung confidential deniable authentication protocol [10] to design our concurrent signcryption scheme for mobile users' anonymity. In our concurrent signcryption scheme, an initial signer and a matching signer exchange their signatures in a fair and confidential way. So the concurrent signcryption scheme satisfies the following security properties [11, 12].

Unlinkability: It is computationally infeasible to find the link between the initial signer's and the matching signer's signatures belonging to the same exchange, even though many initial signer's and match signer's signatures/signcryptexts are collected.

Correctness: The initial signer obtains the match signer's signature and the match signer obtains the initial signer's signature if they honestly perform the concurrent signcryption scheme.

Fairness: After finishing the concurrent signcryption scheme, only the two cases occur. One case is that both initial signer and matching signer obtain the match signer's signature and the initial signer's signature, respectively. One case is that no one obtains the signatures.

Unforgeability: Neither the initial signer's signatures nor matching signer's signatures can be forged, even though the forger is initial signer or matching signer.

Confidentiality: The exchanging signcryptexts are indistinguishably secure against adaptively chosen ciphertext attacks (IND-CCA2).

Our concurrent signcryption scheme is embedded in our delegation-based authentication protocol. The concurrent signcryption scheme is used to fairly exchange the mobile user's signature on the proxy public key application and the HLR's signature on the proxy public key delegation warrants. Our new delegation-based authentication protocol satisfies the following security properties [1, 6-8,10].

Non-repudiation: HLR cannot repudiate its authorization of MUs and the MU cannot repudiate its roaming services usage.

Mutual authentication: During roaming services, the delegation authentication protocol provides the mutual authentication between HLR and VLR, VLR and MU, and HLR and MU.

Session key security: During roaming services, the secret session keys between VLR and MU are secure.

User identity privacy: During roaming services, no one can find out MU's real identity except HLR.

User unlinkability: It is hard to classify the roaming requests into groups such that the roaming requests in the same group belong to the same MU, even though the roaming requests are collected from different VLRs. The user unlinkability is divided into the login and subsequent login user unlinkability. The login user unlinkability means that only the beginning roaming requests for the first roaming in the VLR are considered. The subsequent login user unlinkability means that not only the beginning roaming requests but also the continuing roaming requests are considered.

No exhausted search: After receiving the roaming request from some anonymous MU, the VLR easily find the session key for the anonymous MU during the subsequent login authentication without exhaust search.

In the next section, the confidential deniable authentication protocol [11] is reviewed. Our new delegation-based authentication protocol is described in Section 3. The security analysis of our protocol is given in Section 4. Section 5 gives the security property comparison and discussions among Youn and Lim, Chen et al., and our protocols. The final section is our conclusions.

2 Review of Hwang and Sung's Confidential Deniable Authentication Protocol

Hwang and Sung confidential deniable authentication protocol [11] is reviewed in Section 2.1. Then the decision Diffie-Hellman (DDH) problem and DDH assumption are described in Section 2.2.

2.1 Hwang and Sung's Confidential Deniable Authentication Protocol

The protocol consists of the setup, promised-signcryption, and promise-unsigncryption phases. In the setup phase, the system parameters and public functions are published. Public parameters p and q are two large primes with $q|(p\text{-}1)$ according to the security level parameter l. The public parameter g is an element in Z_p^* with order q. The public symmetric-key encryption is $E_k(m)$ and the decryption function is $D_k(m)$, where m is the message and k is the session key. Two public one-way hash functions are $H_1(.)$ mapping from $\{0,1\}^*$ to Z_q^* and $H_2(.)$ mapping from $\{0,1\}^*$ to $\{0,1\}^l$. The i's private key and public key pair is $(x_i, y_i = g^{x_i} \bmod p)$, where x_i is randomly chosen from Z_q^*. Here and after, the notations p and q denote two large primes with $q|(p-1)$ and the notation g denotes the element in Z_p^* with order q.

In the promised-signcryption phase, suppose that the sender A generates the promised signcryptext on the message m to the receiver B by the following steps.

Step 1: Choose two random numbers R and k in Z_q^*.

Step 2: Compute $V= H_1(g^k \bmod p||m||R||y_B)$, $s= k+ Vx_A \bmod q$, $S= g^s \bmod p$, and $K= H_2((y_B)^s \bmod p)$.

Step 3: Generate the ciphertext $C= E_K(m||R)$ and transmit the promise signcryptext (C, V, S) to the receiver B.

In the promise-unsigncryption phase, B decrypts and verifies the promised signcryptext.

Step 1: Compute $K= H_2(S^{x_B} \bmod p)$.

Step 2: Decrypt C to get the message m and the random number R.

Step 3: Verify whether or not $V= H_1(Sy_A^{-V} \bmod p||m||R||y_B)$. If the equation holds, B is convinced that the message m is sent from the sender A.

2.2 Underlying Security Assumptions

Our protocol adopts some underlying security assumptions. One is the symmetric encryption/decryption scheme is IND-CCA2. One is the DDH assumption that no probabilistic polynomial-time (PPT) algorithm solves the following DDH problem with non-negligible probability.

Decision Diffie-Hellman Problem

Given the DDH instance ($g^a \bmod p$, $g^b \bmod p$, $g^c \bmod p$), to determine whether $g^{ab} \equiv g^c \pmod p$, where a, b, and c are secret numbers in Z_q^*.

3 Our Delegation-Based Authentication Protocol with Unlinkability and Users' Non-repudiation for Portable Communication Systems

Some system parameters and public functions are constructed first. The parameter l is a security level parameter. The public parameters p and q are two large primes with $q|(p-1)$. The public parameter g is an element in Z_p^* with order q. Two one-way hash functions $h()$ and $H()$ are published, where $h(.)$ maps from $\{0, 1\}^*$ to Z_q^* and $H(.)$ maps from $\{0, 1\}^*$ to $\{0, 1\}^l$. Our protocol also publishes a symmetric cipher $[M]_K$ satisfying IND-CCA2, where M is a message and K is the symmetric secret key. In our scheme, some public key based signature scheme is also published for all legal mobile users. Notation $A= Sig[M]_x$ denotes the signature generation function and $Verify[A, M]_y$ is the signature verification function, where M is the message, A is the signature on the message M, x is the signer's private key, and y is the signer's public key. Notation K_{VH} denotes the shared secret key between VLR and HLR. Notations ID_V and ID_H denote the identities of VLR and HLR, respectively. Notation $m_1||m_2$ denotes the message m_1 is concatenated with the message m_2. Notation $A \rightarrow B$: M denotes that the user A sends the message M to the user B. Notations (x_M, $y_M= g^{x_M} \bmod p$) and (x_H, $y_H= g^{x_H} \bmod p$) denote the certificated private-public key pair of MU and HLR, respectively.

Our protocol contains three phases: Initialization, login authentication, and subsequent login authentication phases.

Initialization Phase

A mobile user MU submits his/her anonymous private-public key pair to the HLR for the first registration.

Step 1: MU constructs an anonymous proxy private-public key pair (x_K, $y_K= g^{x_K} \bmod p$), where the a proxy private key $x_K \in Z_q^*$ is a random integer.

Step 2: MU generates the promise of Schnorr signature σ_M on the registration for (x_K, y_K).

Step 2.1: Select a random number $r_M \in Z_q^*$.

Step 2.2: Compute $V_M= h(g^{r_M} \bmod p, m_M||ID_M||y_K)$, the keystone $s_M= r_M+ V_Mx_M \bmod q$, and $S_M= g^{s_M} \bmod p(= g^{r_M+V_Mx_M} \bmod p)$, where m_M is the registration document for the proxy private-public key pair (x_K, y_K).

Step 2.3: Store $\sigma_M= (S_M, V_M)$ and s_M in MU's database.

Step 3: MU transmits HLR (σ_M, $m_M||ID_M||y_K$) through secure channels.

Step 4: HLR verifies the promise σ_M by checking whether or not $V_M= h(S_My_M^{-V_M} \bmod p, m_M||ID_M||y_K)$. If the equation does not hold, then stop.

Step 5: HLR generates the promise of Schnorr-like signature σ_H.

Step 5.1: Select a random number $r_H \in Z_q^*$.

Step 5.2: Compute $S_H = S_M^{x_H}$ mod p, $V_H = h(g^{r_H} S_H$ mod p, $m_H||ID_H||y_K)$, $k = (r_H - V_H)x_H^{-1}$ mod q, where m_H is the anonymous authorization warrant on (x_K, y_K).

Step 5.3: Store $(\sigma_H = (S_H, k, V_H), \sigma_M, m_H, m_M, ID_M, y_K)$ in HLR's database sorted by y_K.

Step 6: HLR transmits (σ_H, m_H) to MU through secure channels.

Step 7: MU validates σ_H by checking whether or not $V_H = h(g^{V_H} y_H^k S_H$ mod p, $m_H||ID_H||y_K)$. If the equation does not hold, then stop.

Step 8: MU sets his/her proxy private key as x_K and his/her proxy public key as y_K.

Step 9: MU computes HLR's Schnorr-like signature $\rho_H = (s_H = s_M + k$ mod q, $V_H)$.

When $\rho_H = (s_H, V_H)$ is used, HLR easily recovers MU's Schnorr signature $\rho_M = (s_M = s_H - k$ mod q, $V_M)$.

Login Authentication Phase

MU contracts VLR to obtain roaming services, and VLR checks the MU's legality by the following login authentication protocol. Suppose that MU's current unused proxy public key is y_K.

Step 1: MU randomly selects an integer n_1 and computes a one-way hash chain $h^1(n_1) = h(n_1)$, $h^2(n_1)$, $h^3(n_1)$, …, $h^{(n+1)}(n_1)(= N_1)$, where $h^{i+1}(n_1) = h(h^i(n_1))$ for $n \geq i \geq 1$.

Step 2: MU constructs a new anonymous private-public key $(x_{k,new}, y_{k,new})$ for the next round by selecting a random number $x_{K,new} \in Z_q^*$ and computing $y_{K,new} = g^{x_{K,new}}$ mod p.

Step 3: MU generates the promise of Schnorr signature for $(x_{k,new}, y_{k,new})$.

Step 3.1: Select a random number $r_M' \in Z_q^*$.

Step 3.2: Compute $V_M' = h(g^{r_M'}$ mod p, $m_M||ID_M||y_{K,new})$, the keystone $s_M' = r_M' + V_M' x_M$ mod q, and $S_M' = g^{s_M'}$ mod $p = g^{r_M' + V_M' x_M}$ mod p, where m_M is the registration document for $(x_{K,new}, y_{K,new})$.

Step 3.3: Store $\sigma_M' = (S_M', V_M')$ and s_M' in its database

Step 4: MU generates the promise of signcrytext.

Step 4.1: Select a random number $x \in Z_q^*$.

Step 4.2: Compute $Y = g^x$ mod p, $SK = H(S_M' ||(y_H)^{s_M'}$ mod $p)$, and $MK = H(SK)$.

Step 4.3: Generate a $MAC = H(MK, m_M||Y||ID_V||ID_M||V_M'||y_{K,new})$ and ciphertext $C_M = [m_M||Y||ID_V||ID_M||V_M'||y_{K,new}||MAC]_{SK}$.

Step 5: MU transmits $(\rho_H = (s_H, V_H) = (s_M + k$ mod q, $V_H)$, m_H, ID_H, $y_K)$ to VLR.

Step 6: VLR verifies ρ_H by checking whether or not $V_H = h(g^{V_H} y_H^{s_H}$ mod p, $m_H||ID_H||y_K)$. If the equation does not hold, then stop.

Step 7: VLR selects a random number n_2 and transmits MS the n_2, the period of validity Per, and ID_V.

Step 8: MU generates the signature $A = Sig[N_1||n_2||Per||ID_V]_{x_K}$ and sends $(A, (C_M, S_M'), ID_V, N_1)$ to VLR.

Step 9: VLR validates the signature A on $N_1||n_2||Per||ID_V$.

Step 9.1: If $Verify[A, N_1||n_2||Per||ID_V]_{y_K}$ is "valid", generate the ciphertext $[A||Per||N_1||n_2||y_K||(C_M,S_M')||\rho_H||Dig]_{K_{HV}}$, where $Dig = H(A||Per||N_1||n_2||y_K||(C_M,S_M')||\rho_H)$.

Step 9.2: Transmit $([A||Per||N_1||n_2||y_K||(C_M,S_M')||\rho_H||Dig]_{K_{HV}}, ID_H, ID_V)$ to HLR.

Step 10: HLR decrypts the ciphertext $[A||Per||N_1||n_2||y_K||(C_M,S_M')||\rho_H||Dig]_{K_{HV}}$.

Step 10.1: Obtain $(A||Per||N_1||n_2||y_K||(C_M,S_M')||\rho_H||Dig)$ by decrypting $[A||Per||N_1||n_2||y_K||(C_M,S_M')||\rho_H||Dig]_{K_{HV}}$.

Step 10.2: Validate the recovered message by checking whether or not $Dig = H(A||Per||N_1||n_2||y_K||(C_M,S_M')||\rho_H)$.

Step 11: HLR validates the certificate ρ_H of MU's anonymous public key y_k.

Step 11.1: Find $\sigma_H = (S_H, k, V_H)$ and $\sigma_M = (S_M, V_M)$ in HLR's database using y_K as the searching key.

Step 11.2: Recover the MU's signature $\rho_M = (s_M = s_H - k$ mod q, $V_M)$.

Step 11.3: Verify ρ_M by checking whether or not $V_M = h(g^{s_M} y_M^{-V_M}$ mod p, $m_M||ID_M||y_K)$. If ρ_M is valid, HLR is convinced that MU is legal.

Step 11.4: Validate the signature A by checking $Verify[A, N_1||n_2||Per||ID_V]_{y_K}$ to confirm the specified MU is the one knowing the private key x_K.

Step 12: HLR validates MU's promise of the Schnorr signature using $y_{k, new}$.

Step 12.1: Compute $S_H' = S_M'^{x_H}$ mod p and $SK = H(S_M||S_H'$ mod $p)$.

Step 12.2: Obtain $m_M||Y||ID_V||ID_M||V_M'||y_{K,new}||h(MK, m_M||Y||ID_V||ID_M||V_M'||y_{K,new})$ by decrypting C_M with the session key SK.

Step 12.3: Compute $MK = H$(SK) then check $h(MK, m_M||Y||ID_V||ID_M||V_M'||y_{K,new})$ to authenticate message.

Step 12.4: Check ID_V.

Step 12.5: Validate the promise (S_M', V_M') by checking whether or not $V_M' = h(S_M' y_K^{-V_M'}$ mod p, $m_M||ID_M||y_{K,new})$.

Step 13: HLR generates the response

Step 13.1: Select two random numbers r_H' and R'.

Step 13.2: Compute $V_H'= h(g^{r_H'}S_M'^{x_H} \bmod p, m_H||ID_H||y_{K,new})$ and $k'= (r_H'- V_H')x_H^{-1} \bmod q$. So the new promise of the certificate is $\sigma_H'= (S_H', k', V_H')$

Step 13.3: Compute $SK'=H(Y^{x_H} \bmod p)$.

Step 13.4: Choose a nonce n_3.

Step 13.5: Compute $RK_1=H(N_1||n_2||n_3||y_K)$ and store $L= N_1$.

Step 13.6: Generate and transmit $([[N_1||n_3||ID_V||R'||\sigma_H']_{SK'}||y_K||n_2||N_1||RK_1]_{K_{HV}}, ID_H, ID_V)$ to VLR.

Step 14: VLR validates HLR's response.

Step 14.1: Obtain $[N_1||n_3||ID_V||R'||\sigma_H']_{SK'}||y_K||n_2||N_1||RK_1$ by decrypting the ciphertext $[[N_1||n_3||ID_V||R'||\sigma_H']_{SK'}||y_K||n_2||N_1||RK_1]_{K_{HV}}$.

Step 14.2: Check the freshness of n_2. If n_2 is not fresh, then stop.

Step 14.3: Compute $ID_1= H([N_1||n_3||ID_V||R'||\sigma_H']_{SK'}||RK_1)$.

Step 14.4: Store $((\rho_H, m_H, ID_H, ID_1), RK_1, A, L= N_1, 1)$ into its database according to the order of ID_1.

Step 14.5: Transmit $([N_1||n_3||ID_V||R'||\sigma_H']_{SK'}, ID_V)$ to MU.

Step 15: MU decrypts HLR's ciphertext.

Step 15.1: Get $N_1||n_3||ID_V||R'||\sigma_H'$ by decrypting $[N_1||n_3||ID_V||R'||\sigma_H']_{SK'}$, where $SK'= H(y_H^x \bmod p)$.

Step 15.2: Check the freshness of N_1. If N_1 is not fresh, then stop.

Step 16: MU validates the promise σ_H' by checking $V_H'= h(g^{V_H'}y_H^{k'}S_H' \bmod p, m_H||ID_H||y_{K,new})$ with the new key $y_{K,new}$. If $V_H'= h(g^{V_H'}y_H^{k'}S_H' \bmod p, m_H||ID_H||y_{K,new})$ then accept; otherwise, reject.

Step 17: MU sets the new proxy private key as $x_{K,new}$ and the new proxy public key as $y_{K,new}$ for the next round.

Step 18: MU computes HLR's Schnorr-like signature $\rho_H'= (s_H'= s_M'+k' \bmod q, V_H')$.

Step 19: MU computes and stores $(RK_1= H(N_1||n_2||n_3||y_K), (h^1(n_1), h^2(n_1), h^3(n_1), \ldots, h^{(n+1)}(n_1)), y_K$, round number= 1, $ID_1=H([N_1||n_3||ID_V||R'||\sigma_H']_{SK'}||RK_1))$ into MU's database.

When $\rho_H'= (s_H', V_H')$ is used, HLR recovers MU's Schnorr signature $\rho_M'= (s_M'=s_H'-k' \bmod q, V_M')$ after obtaining ρ_H'.

Subsequent Login Authentication Phase

VLR authenticates MU repeatedly without contracting HLR in the subsequent login authentication. Suppose that this is ith round subsequent login authentication with the session key RK_i, where $i\leq n$, $RK_i= H(L, RK_{i-1})$, $RK_1= H(N_1||n_2||n_3||y_K)$, and $L= H^{(n-i+2)}(n_1)$. Therefore, the mobile user retrieves the record $(RK_i, (h^1(n_1), h^2(n_1), h^3(n_1), \ldots, h^{(n+1)}(n_1)), y_K$, round number= i) first.

Step 1: MU transmits $(ID_i, [h^{(n-i+1)}(n_1)]_{RK_i})$ to VLR.

Step 2: VLR finds $((\rho_H, m_H, ID_H, ID_i), RK_1, A, L, i)$ by searching its database according to ID_i and obtains $h^{(n-i+1)}(n_1)$ by decrypting $[h^{(n-i+1)}(n_1)]_{RK_i}$.

Step 3: VLR checks whether or not $h(h^{(n-i+1)}(n_1))= L$. If $h(h^{(n-i+1)}(n_1))= L$, VLR updates $L= h^{(n-i+1)}(n_1)$, $i= i+1$, $RK_{i+1}= h(L, RK_i)$, and $ID_{i+1}= H(ID_i||RK_{i+1})$ for the next round.

Step 4: VLR sends $[ACK]_{RK_i}$ back to MU, where $ACK= h(h^{(n-i+1)}(n_1)||ID_i)$.

Step 5: MU obtains ACK by decrypting $[ACK]_{RK_i}$ and validates ACK by checking whether or not $ACK= h(h^{(n-i+1)}(n_1)||ID_i)$.

If $i= n+1$, then MU repeats the login authentication by setting $y_K= y_{K,new}$.

4. Security Analysis and Proofs

Our protocol satisfies ten properties: Unlinkability, correctness, fairness, unforgeability, confidentiality, non-repudiation, mutual authentication, session key security, and user identity privacy, and user unlinkability. Finally, the assumption of semi-trust HLR is discussed.

Confidentiality

Our protocol satisfies IND-CCA2, if there is no PPT adversary α wins the IND-CCA2 game with the non-negligible probability.

IND-CCA2 Game

The game has two participators: Adversary α and Challenger C. Challenger C controls all oracles that α is allowed to query. For the un-occurred queries, the hash oracles h and H returns randomly chosen digests and stores the query and digest into their databases. For the occurred queries, the oracles h and H return the digests found in their databases. The decryption oracle returns decrypted messages for the queried ciphertexts, if the ciphertexts are legal.

The IND-CCA2 game contains the setup, collecting, and challenging and guessing phases. In the setup phase, all systems parameters, public functions, and the public key/private keys of all mobile users are constructed. The public parameters and public keys are sent to Adversary. In the collecting phase, adversary collects his/her chosen ciphertexts and the corresponding messages pairs with the help of the decryption oracle. At the end of the collecting phase, Adversary chooses and sends Challenger two messages with the same length. In the challenging and guessing phase, challenger gives adversary the challenging ciphertext of one of the two chosen messages from adversary. Adversary is still allowed to collecting the message and ciphertext pairs, where the ciphertexts may be chosen by Adversary.

Adversary finally guesses which message is chosen to generate the challenging ciphertext. If the guess is correct, adversary wins the game; otherwise, it loses.

Theorem 1: Assume the underlying symmetric encryption/decryption scheme is IND-CCA2. Our protocol is IND-CCA2 if no PPT algorithm wins the IND-CCA2 game with non-negligible probability based on the DDH assumption in the random oracle model.

Proof of Theorem 1: Consider the DDH instance (g^a mod p, g^b mod p, g^c mod p). Suppose that a PPT algorithm α wins the IND-CCA2 game, then a PPT DDH-α algorithm is proposed to solve DDH problem with non-negligible advantage Δ over 1/2.

Setup Phase

DDH-α first sets the system parameters p, q, and g. Then DDH-α generates the private/public key pair (x_M, $y_M = g^{x_M}$ mod p) for MU, where x_M is a random integer in Z_q^*. DDH-α also sets the HLR's public key $y_H = g^a$ mod p. DDH-α finally gives Adversary α these system parameters, the public keys y_M and y_H.

Collecting Phase

Adversary α chooses some signcrytexts (C_M, S_M') and obtains the corresponding message $m_M||Y||ID_V||ID_M||V_M'||y_{K,new}$ to C_M. The decryption oracle generates the response for the decryption query of (C_M, S_M') as follows.

Step 1: The decryption oracle first queries the hash oracle H to obtain the session key SK by giving (S_M', $\perp$). The H first searches its database for the query (S_M', $\perp$). If some record (($S_M'||DH$ mod p), SK) with partial matching value S_M' is found, then H returns SK; otherwise, H randomly chooses a SK and store the new record (($S_M'||\perp$ mod p), SK) into H's database, where DH is the Diffie-Hellman key of S_M' and y_H.

Step 2: The decryption oracle decrypts C_M to obtain $m_M||Y||ID_V||ID_M||V_M'||y_{K,new}||MAC$ using the session key SK. If one of the identities ID_V and ID_M are illegal; then the decryption oracle returns illegal response.

Step 3: The decryption oracle computes $MK = H(SK)$ with the help of H.

Step 4: The decryption oracle obtains the message digest MAC of (MK, $m_M||Y||ID_V||ID_M||V_M'||y_{K,new}$) by querying H.

Step 5: The decryption oracle returns that the ciphertext (C_M, S_M') is illegal if $MAC \neq H(MK, m_M||Y||ID_V||ID_M||V_M'||y_{K,new}||MAC)$; otherwise the decryption oracle returns the message $m_M||Y||ID_V||ID_M||V_M'||y_{K,new}$.

Step 6: The decryption oracle queries the hash oracle h with the input (S_M', y_M, $m_M||ID_M, V_M'$). The oracle h searches its database to find the record (S_M', y_M, $m_M||ID_M, V_M''$) with the partial matched (S_M', y_M, $m_M||ID_M$). If the record is found, the oracle h returns V_M''; otherwise, the oracle h returns V_M' and stores the record (S_M', y_M, $m_M||ID_M, V_M'$) into its database.

Step 7: The decryption oracle returns that the ciphertext (C_M, S_M') is illegal if $V_M' \neq h(S_M y_M^{-V_M'}$ mod p, $m_M||ID_M)$; otherwise, returns message $m_M||Y||ID_V||ID_M||V_M'||y_{K,new}$.

Challenging and Guessing Phase

The adversary chooses and sends two different messages $m_M||ID_V||ID_M||y_{K,1}$ and $m_M||ID_V||ID_M||y_{K,2}$ to the challenger DDH-α, where $y_{K,1} \neq y_{K,2}$. DDH-α generates and sends the challenge ($C_{M\beta}$, $S_{M\beta}'$) to α, where ($C_{M\beta}$, $S_{M\beta}'$) is the signcrytext of $m_M||Y||ID_V||ID_M||y_{K,1}$ or $m_M||Y||ID_V||ID_M||y_{K,2}$.

Step 1: Choose a random value $\beta \in \{1, 2\}$.

Step 2: Encrypt $m_M||Y_\beta||ID_V||ID_M||y_{K\beta}$ and ID_M.

Step 2.1: Choose a random number $x \in Z_q^*$ and compute $Y = g^x$ mod p.

Step 2.2: Set $S_{M\beta} = g^b$ mod p.

Step 2.3: Choose $V_{M\beta}$ randomly and set $V_{M\beta} = h(S_{M\beta} y_M^{-V_{M\beta}}$ mod p, $m_M||ID_M||y_{k\beta})$ with the help of the oracle h.

Step 2.4: Compute $SK = H(S_{M\beta}||g^c$ mod $p)$ with the help of the hash oracle H.

Step 2.5: Compute $MK = H(SK)$ with the help of H.

Step 2.6: Compute $MAC = H(MK, m_M||Y||ID_V||ID_M||V_{M\beta}||y_{K\beta}))$.

Step 2.7: Obtain the ciphertext $C_{M\beta}$ by encrypting $m_M||Y||ID_V||ID_M||V_{M\beta}||y_{K\beta}$ and MAC with the key SK.

Step 3: Send the challenge ($C_{M\beta}$, $S_{M\beta}'$) to α.

After DDH-α gives the challenge, α is still allowed to collect the other message and signcryptexts. Finally the adversary α outputs the guess $\beta'=1$ or 2. If $\beta'=\beta$, DDH-α returns $g^{ab}\equiv g^{c}\ (\mathrm{mod}\ p)$, otherwise, DDH-$\alpha$ returns g^{ab} mod $p\neq g^{c}$ mod p. If α cannot answer the question over the polynomial-time upper bound of α, DDH-α returns g^{ab} mod $p\neq g^{c}$ mod p.

Probability analysis

There are three cases in the probability analysis of DDH-α algorithm.

Case 1: g^{ab} mod $p= g^{c}$ mod p.

The correct guess $\beta'=\beta$ of α means DDH-α returns the correct answer for the DDH instance (g^{a} mod p, g^{b} mod p, g^{c} mod p). Since the winning probability of α is $1/2+\Delta$, the successful probability of DDH-α is $\frac{1/2+\Delta}{q}$ in this case. On the other hand, the incorrect guess $\beta\neq\beta'$ of α means that DDH-α also returns the incorrect answer. Since the losing probability of α is $1/2-\Delta$, the failure probability of DDH-α is $\frac{1/2-\Delta}{q}$.

Case 2: g^{ab} mod $p\neq g^{c}$ mod p with the collision of H.

Since g^{ab} mod $p\neq g^{c}$ mod p, the encryption key SK' is correct only when the collision of hash function H occurs. The collision probability of hash function H is $(1/q)$. The output $\beta'=\beta$ of α means DDH-α returns the incorrect answer of this game. The failure probability of DDH-α is $\frac{q-1}{q}\times\frac{1/2+\Delta}{q}$ in this case. The output $\beta'\neq\beta$ of α means DDH-α returns the correct answer of this game, so the success probability of DDH-α is $\frac{q-1}{q}\times\frac{1/2-\Delta}{q}$.

Case 3: g^{ab} mod $p\neq g^{c}$ mod p without the collision of H.

Without the collision of H and g^{ab} mod $p\neq g^{c}$ mod p, the encryption key SK' is incorrect, so the challenge is also illegal. Then, α cannot give its guess during its polynomial-time bound. After exceeding α's polynomial-time bound, DDH-α returns g^{ab} mod $p\neq g^{c}$ mod p. This is successful case of DDH-α with probability $\frac{q-1}{q}\times\frac{1/2-\Delta}{q}$.

Thus the total failure probability of DDH-α is $(\frac{(1/2-\Delta)}{q}+\frac{q-1}{q}\times\frac{1/2+\Delta}{q})<\frac{1}{2q}+\frac{1}{q}$. Since q is a large prime number, the upper bound $\frac{1}{2q}+\frac{1}{q}$ is negligible. So the failure probability of DDH-α is negligible. Based on the DDH assumption, the proposed protocol is IND-CCA2.

Unlinkability between the Promise of Signcrytext and HLR's Ciphertext

Unlinkability among distinct roaming services means no one can deduce any two distinct roaming services belonging to the same MU. Two cases cause linkability problem. One case is that adversaries find out the linkage between ρ_H' and (C_M, S_M') from the same MU on two successive roaming. Since (C_M, S_M') is the signcryptext of the promise $\sigma_M'=(S_M', V_M')$, the adversary may find out the linkage between ρ_H' and σ_M' with the help of the linkage between ρ_H' and (C_M, S_M'), if the decryption of C_M is feasible. To determine the linkage between ρ_H' and (C_M, S_M') is at least harder than to explore the linkage between ρ_H' and σ_M'. The linkage problem between ρ_H' and σ_M' is also the linkage problem in Nguyen's scheme that is our underlying scheme. Fortunately, to find the linkage is infeasible for the unlinkability of Nguyen's scheme [12].

The other case is to find out the linkage between the promise of signcrytext (C_M, S_M') and HLR's ciphertext $[N_1||n_3||ID_V||R'||\sigma_H']_{SK'}$. To show that the linkage find is infeasible, the unlinkability game is defined. Our protocol is unlinkabile between the promise of signcrytext (C_M, S_M') and HLR's ciphertext $[N_1||n_3||ID_V||R'||\sigma_H']_{SK'}$ if no polynomial-time adversary α wins the unlinkability game with a non-negligible probability.

Unlinkability Game between the Promise of Signcrytext and HLR's Ciphertext

This game has two participators, the adversary α and the challenger C. Challenger C controls all hash oracles whom α is allowed to query. These hash oracles are the same as the hash oracles in the security proof of confidentiality.

This game consists of two phases: Setup, and challenging and guessing phases. In the setup phase, all systems parameters, public functions, and the public keys/private keys of all Users are constructed. The public parameters and all public keys are sent to the adversary.

In the challenging and guessing phase, the adversary first chooses two anonymous public keys that may belong to two different MUs to the challenger. The challenges have two types. Type 0 is that the challenge consists of two promises of signcryptext using two anonymous public keys and one HLR's ciphertext for one anonymous public key. The adversary guesses which the promise matches the HLR's ciphertext. Type 1 is that the challenge consists of two HLR's ciphertexts using two anonymous public keys and one promise of signcryptext for one anonymous public key. The adversary guesses that the HLR's ciphertext matches the promise of signcryptext.

Theorem 2: Our protocol provides unlinkablity between the promise of signcrytext and HLR's ciphertext if no PPT algorithm wins the above unlinkability game with non-negligible probability based on the DDH assumption in random oracle model.

Proof of Theorem 2

Let that the instance of DDH problem be ($g^a \bmod p$, $g^b \bmod p$, $g^c \bmod p$). Suppose that a PPT algorithm α wins the above unlinkability game between the promise of signcrytext and HLR's ciphertext with advantage Δ over 1/2, then an algorithm DDH-α is proposed to solve DDH problem with non-negligible probability.

Setup Phase

The challenger DDH-α first adopts p, q, and g as the system parameters and constructs the private/public key pair (x_M, $y_M = g^{x_M} \bmod p$) of User, where x_M is randomly selected in Z_q^*. DDH-α sets the public key of HLR $y_H = g^a \bmod p$. Finally, DDH-α gives the adversary α these system parameters, the public keys y_M, and y_H.

Challenging and Guessing Phase

Adversary α produces and gives two anonymous public key pairs of MU y_{K0} and y_{K1} to DDH-α. DDH-α produces the promises of signcrytexts and HLR's ciphertexts by the following steps. Finally, DDH-α gives the challenge to α. The challenge consists of either two promises of User's Schnorr signcryptexts and one promise of HLR's ciphertext, or one promise of User's Schnorr signcryptext and two promises of HLR's ciphertext.

Step 1: Choose δ, r_{M_0} and $r_{M_1} \in Z_q^*$.

Step 2: Obtain $V_{M_0} = h(g^{r_{M_0}} \bmod p, m_M || ID_M || y_{K_0})$ and $V_{M_1} = h(g^{r_{M_1}} \bmod p, m_M || ID_M || y_{K_1})$ with the help of the hash oracle h.

Step 3: Let $Y_0 = g^b \bmod p$ and $Y_1 = (g^b)^\delta \bmod p$.

Step 4: Compute $s_{M_0} = r_{M_0} + V_{M_0} x_M \bmod q$ and $s_{M_1} = r_{M_1} + V_{M_1} x_M \bmod q$.

Step 5: Compute $S_{M_0} = g^{s_{M_0}} \bmod p$ and $S_{M_1} = g^{s_{M_1}} \bmod p$.

Step 6: Obtain $SK_0 = H(S_{M_0} || y_H^{s_{M_0}} \bmod p)$ and $SK_1 = H(S_{M_1} || y_H^{s_{M_1}} \bmod p)$.

Step 7: Obtain $MK_0 = H(SK_0)$ and $MK_1 = H(SK_1)$.

Step 8: Compute the message digests $Dig_0 = H(MK_0, m_M || Y_0 || ID_V || ID_M || V_{M_0} || y_{K,new})$ and $Dig_1 = H(MK_1, m_M || Y_1 || ID_V || ID_M || V_{M_1} || y_{K,new})$.

Step 9: Obtain the ciphertexts $C_{M_0} = [m_M || Y_0 || ID_V || ID_M || V_{M_0} || y_{K_0} || Dig_0]_{SK_0}$ and $C_{M_1} = [m_M || Y_1 || ID_V || ID_M || V_{M_1} || y_{K_1} || Dig_1]_{SK_1}$ using the symmetric encryption function.

Step 10: Choose k_0, k_1, V_{H_0}, and V_{H_1} randomly and compute $g^{r_{H_0}} \bmod p = y_H^{k_0} g^{V_{H_0}} \bmod p$ and $g^{r_{H_1}} \bmod p = y_H^{k_1} g^{V_{H_1}} \bmod p$.

Step 11: Set $V_{H_0} = h(g^{r_{H_0}} y_H^{s_{M_0}} \bmod p, m_H || ID_H || y_{k_0})$ and $V_{H_1} = h(g^{r_{H_1}} y_H^{s_{M_1}} \bmod p, m_H || ID_H || y_{k_1})$ with the help of the hash oracle h on the given V_{H_0} and V_{H_1}.

Step 12: Compute $S_{H_0} = y_H^{S_{M_0}} \bmod p$ and $S_{H_1} = y_H^{S_{M_1}} \bmod p$, then generate $SK'_0 = H(g^c \bmod p)$ and $SK_1' = H((g^c)^\delta \bmod p)$.

Step 13: Choose (N_{1_0}, n_{2_0}, n_{3_0}, R_0) and (N_{1_1}, n_{2_1}, n_{3_1}, R_1) randomly, and generate $[N_{1_0} || n_{2_0} || n_{3_0} || R_0 || \sigma_{H_0} = (S_{H_0}, k_0, V_{H_0})]_{SK'_0}$ and $[N_{1_1} || n_{2_1} || n_{3_1} || \mathrm{R}_1 || \sigma_{H_1} = (S_{H_1}, k_2, V_{H_1})]_{SK'_1}$ by the symmetric encryption function.

Step 14: Choose a random bit β and the type-bit. If type-bit is 0, give α the challenge C_{M_0}, C_{M_1}, and $[N_{1_\beta} || n_{2_\beta} || n_{3_\beta} || R_\beta || \sigma_{H_\beta} = (S_{H_\beta}, k_\beta, V_{H_\beta})]_{SK'_\beta}$; otherwise, give α the challenge C_{M_β}, $[N_{1_0} || n_{2_0} || n_{3_0} || R_{1_0} || \sigma_{H_0}]_{SK'_0}$, and $[N_{1_1} || n_{2_1} || n_{3_1} || R_{1_1} || \sigma_{H_1}]_{SK'_1}$ to α.

On the challenge, the adversary α outputs a bit β'= 0 or 1. If β'=β, DDH-α returns $g^{ab} \equiv g^c \pmod p$; otherwise, DDH-α returns $g^{ab} \bmod p \neq g^c \bmod p$.

Probability analysis

There are three cases in the failure probability analysis of DDH-α algorithm.

Case 1: g^{ab} mod p= g^c mod p.

The incorrect guess β'≠ β of α means DDH-α returns the wrong answer for the DDH instance. Since the losing probability of α is 1/2-Δ, the failure probability of DDH-α is $\frac{1/2-\Delta}{q}$ for the incorrect guess.

Case 2: g^{ab} mod p≠ g^c mod p with the collision of H.

Since $g^{ab} \bmod p \neq g^c \bmod p$, the encryption key SK' is correct only when the collision of hash function H occurs. The collision probability of an ideal hash function H is ($1/q$). The correct guess β'= β of α means DDH-α returns the incorrect answer of this game, so the failure probability of DDH-α is $\frac{q-1}{q} \times \frac{1/2+\Delta}{q}$.

Case 3: g^{ab} mod p≠ g^c mod p without the collision of H.

Since no collision of H occurs and $g^{ab} \bmod p \neq g^{c} \bmod p$, the encryption key SK' is incorrect and the challenge is illegal. The adversary α cannot give its guess during its polynomial-time bound, so DDH-α returns the response $g^{ab} \bmod p \neq g^{c} \bmod p$. This is the only successful case of DDH-α with probability $\frac{q-1}{q} \times \frac{q-1}{q}$.

Thus, the total failure probability of DDH-α is $(\frac{(1/2-\Delta)}{q} + \frac{q-1}{q} \times \frac{1/2+\Delta}{q})$. Since q is a large prime number and $(\frac{(1/2-\Delta)}{q} + \frac{q-1}{q} \times \frac{1/2+\Delta}{q}) < \frac{1}{2q} + \frac{1}{q}$, DDH-α's failure probability is negligible. Based on the DDH assumption, our protocol satisfies unlinkability between promise of signcrytext and HLR's ciphertext.

Fairness

The fairness in our protocol means that User (HLR) can obtain the HLR's (User's) signature but HLR (User) cannot. There are two cases violating the fairness.

Case 1: Given the promises $\sigma_M = (S_M, V_M)$ satisfying $V_M = H(S_M y_M^{-V_M} \bmod p, m_M || ID_M || y_K)$, HLR generates alone the corresponding $\rho_M = (s_M, V_M)$ satisfying $V_M = H(g^{s_M} y_M^{-V_M} \bmod p, m_M || ID_M || y_K)$.

Case 2: Given two promises $\sigma_M = (S_M, V_M)$ and $\sigma_H = (S_H, k, V_H)$ satisfying $V_M = H(S_M y_M^{-V_M} \bmod p, m_M || ID_M || y_K)$ and $V_H = H(g^{V_H} y_H^{k} S_H \bmod p, m_H || ID_H || y_K)$, respectively, User obtains $\rho_H = (s_H, V_H)$ satisfying $V_H = H(g^{V_H} y_H^{s_H} \bmod p, m_H || ID_H || y_K)$ but HLR cannot obtain $\rho_M = (s_M, V_M)$ satisfying $V_M = H(g^{s_M} y_M^{-V_M} \bmod p, m_M || ID_M || y_K)$.

Theorem 3 shows that our protocol is fair.

Theorem 3: Our protocol is fair since neither Case 1 nor Case 2 occurs based on the discrete logarithm and DDH assumption.

Proof of Theorem 3

In Case 1, after obtaining $\sigma_M = (S_M, V_M)$, HLR generates alone the valid Schnorr signature $\rho_M = (s_M, V_M)$ on the message $m_M || ID_M || y_K$ without giving σ_H. Fortunately, Case 1 is harder than the forgeability of Schnorr signatures. Suppose that a polynomial-time adversary can generates the valid Schnorr signature $\rho_M = (s_M, V_M)$ on the message $m_M || ID_M || y_K$ by giving the promise $\sigma_M = (S_M, V_M)$. With the help of this adversary, a Schnorr signature forging algorithm is stated below. On the given message m and the public key y_M, the forging algorithm easily constructs the promise $\sigma_m = (S_m, V_m)$ without the signer's private key x_M by the following steps.

Step 1: Choose $r_m \in Z_q^*$.

Step 2: Obtain $V_m = h(g^{r_m} \bmod p, m_m || ID_m || y_{Km})$, where $m = m_m || ID_m || y_{Km}$.

Step 3: Compute $S_m = g^{r_m} y_M^{V_m} \bmod p$.

Step 4: Give the adversary the promise $\sigma_m = (S_m, V_m)$ and the message m and obtain the Schnorr signature (s_m, V_m) from the adversary with non-negligible probability.

Finally, the forged Schnorr signature (s_m, V_m) is obtained. However, [13-14] show that Schnorr signatures are strong against existentially forgery based on the discrete logarithm problem, Case 1 cannot occurs with non-negligible probability.

In Case 2, since the legal signature $\rho_M = (s_M, V_M)$ satisfies $V_M = H(g^{s_M} y_M^{-V_M} \bmod p, m_M || ID_M || y_K)$, then $s_M \neq s_H - k \bmod q$; otherwise, HLR obtains the legal signature $\rho_M = (s_M, V_M)$. For $s_M \neq s_H - k \bmod q$ and $S_H \equiv S_M^{x_H} \equiv (g^{s_M})^{x_H} \pmod p$, $V_H = H(g^{V_H} y_H^{k} S_H \bmod p, m_H || ID_H || y_K) \neq H(g^{V_H} y_H^{k} g^{(s_H - k) x_H} \bmod p, m_H || ID_H || y_K) = H(g^{V_H} y_H^{s_H} \bmod p, m_H || ID_H || y_K)$. Since $H(g^{V_H} y_H^{k} S_H \bmod p, m_H || ID_H || y_K) \neq H(g^{V_H} y_H^{s_H} \bmod p, m_H || ID_H || y_K)$, $g^{V_H} y_H^{k} S_H \bmod p \neq g^{V_H} y_H^{s_H} \bmod p$. However, $H(g^{V_H} y_H^{s_H} \bmod p, m_H || ID_H || y_K) = V_H = H(g^{V_H} y_H^{k} S_H \bmod p, m_H || ID_H || y_K)$. The contradiction happens and Case 2 never occurs.

Unforgeability

The unforgeability means that both MU's Schnorr signatures and HLR's Schnorr-like signatures are unforgeability. Fortunately, User's Schnorr signatures are existentially unforgeable against chosen-message attacks [13, 14]. HLR's Schnorr-like signatures are unforgeability [12].

Correctness

The correctness means that MU obtains the legal HLR's signature and HLR obtains the legal MU's signature by performing our protocols step by step.

Theorem 5: In our protocol, MU and HLR can correctly exchange their signature, respectively.

Proof

The correctness of the signature exchanges consists of the correctness of the promise of the Schnorr (Schnorr-like) signature and the correctness of the exchanged Schnorr (Schnorr-like) signature. On Step 4 in the initialization phase, HLR validates σ_M by checking whether or not $V_M = h(S_M y_M^{-V_M} \bmod p, m_M || ID_M || y_K)$. The verification is correct sine $s_M= r_M+ V_Mx_M \bmod q$, $S_M= g^{s_M} \bmod p$, and $V_M= h(g^{r_M} \bmod p, m_M||ID_M||y_K)= h(g^{s_M-V_Mx_M} \bmod p, m_M||ID_M||y_K)= h(g^{s_M}(g^{x_M})^{-V_M} \bmod p, m_M||ID_M||y_K)= h(S_My_M^{-V_M} \bmod p, m_M||ID_M||y_K)$. On Step 7 in the initialization phase, User validates σ_H by $V_H= h(g^{V_H}y_H^kS_H \bmod p, m_H||ID_H||y_K)$. This verification is correct since $k= (r_H-V_H)x_H^{-1} \bmod q$, and $V_H= h(g^{r_H}S_H \bmod p, m_H||ID_H||y_K)= h(g^{V_H+x_Hk}S_H \bmod p, m_H||ID_H||y_K)= h(g^{V_H}(g^{x_H})^kS_H \bmod p, m_H||ID_H||y_K)= h(g^{V_H}y_H^kS_H \bmod p, m_H||ID_H||y_K)$. Therefore, the verification of the promises of Schnorr (Schnorr-like) signatures is correct.

Consider the correctness of the Schnorr (Schnorr-like) signatures in our protocols. On Step 6 in the login authentication phase, the Schnorr-like signature, ρ_H, is validated by $V_H= h(g^{V_H}y_H^{s_H} \bmod p, m_H||ID_H||y_K)$. This verification is correct because $S_H\equiv S_M^{x_H}\equiv g^{s_Mx_H}\equiv y_H^{s_M} \pmod p$, $s_H= s_M+k \bmod q$, and $V_H= h(g^{V_H}y_H^kS_H \bmod p, m_H||ID_H||y_K)= h(g^{V_H}y_H^{k+s_M} \bmod p, m_H||ID_H||y_K)= h(g^{V_H}y_H^{s_H} \bmod p, m_H||ID_H||y_K)$. On Step 7 in login authentication phase, HLR validates the Schnorr signature ρ_M by $V_M= h(g^{s_M}y_M^{-V_M} \bmod p, m_M||ID_M||y_K)$. The validation is correct for $V_M= h(S_My_M^{-V_M} \bmod p, m_M||ID_M||y_K)= h(g^{s_M}y_M^{-V_M} \bmod p, m_M||ID_M||y_K)$ and $S_M= g^{s_M} \bmod p$.

Non-Repudiation

The non-repudiation in our protocol consists of the HLR's non-repudiation on the authorization of legal MUs and users' non-repudiation on the roaming service usage. On Step 5 in the login authentication phase, MU uses ρ_H to convince VLRs that it is the authorized MU. So the HLR's non-repudiation on the authorization of MUs is based on the non-repudiation on the Schnorr-like signature $\rho_H= (s_H, V_H)$ and fairness of our protocol. Theorem 3 and the non-repudiation of ρ_H [12] guarantee the HLR's non-repudiation.

In the login authentication phase, MU sends the signature A on the anchor N_1 of the one-way hash chain $h^1(n_1)$, $h^2(n_1)$, $h^3(n_1)$, …, $h^{(n+1)}(n_1)(= N_1)$ by using the authorized private key x_K. The public key y_K is authorized HLR and only the MU knows the authorized private key x_K, so the MU cannot deny the signature A on the anchor N_1. Because the one-way property of hash functions, the one can give the hash value $h^i(n_1)$ is the signer of the signature A. Therefore, the non-repudiation of MU's roaming services usage is based on the unforgability of the signature A and the one-way property of h.

Session Key Security

The first session key RK_1 is secure. The ciphertext $[[N_1||n_3||ID_V||R'||\sigma_H']_{SK'}||y_K||n_2||N_1||RK_1]_{K_{HV}}$ contains RK_1 and the ciphertext $[N_1||n_3||ID_V||R'||\sigma_H']_{SK'}$. Since the underlying symmetric cipher is the IND-CCA2, no one obtains RK_1 from the ciphertext $[[N_1||n_3||ID_V||R'||\sigma_H']_{SK'}||y_K||n_2||N_1||RK_1]_{K_{HV}}$. Since the first session key $RK_1= H(N_1||n_2||n_3||y_K)$, the security of RK_1 is based on the confidentiality of the secret random number n_3 generated by HLR. Fortunately, the underlying symmetric cipher is the IND-CCA2 secure, only the designated MU can obtain the secret random number n_3 to compute RK_1.

Additionally, the subsequent session keys are also secure. For the IND-CCA2 symmetric cipher assumption, the ciphertext $[h^{(n-i+1)}(n_1)]_{RK_i}$ leaks nothing about the session key RK_i and the message $h^{(n-i+1)}(n_1)$. No one derives $RK_{i+1}= h(L, RK_i)$ for both L and RK_i are secret. Thus, the protocol provides the session key security property.

Mutual Authentication

Our protocol provides the mutual authentication property between MU and VLR, VLR and HLR, and MU and HLR, respectively.

Mutual Authentication between MU and VLR

The mutual authentication between MU and VLR consists of login mutual authentication and subsequent login mutual authentication. On Step 6 in login authentication processes, VLR authenticates MU by validating $\rho_H= (s_H, V_H)$ with $V_H= h(g^{V_H}y_H^{s_H} \bmod p, m_H||ID_H||y_K)$ and HLR's public key y_H. If ρ_H is valid, VLR is convinced that MU is some HLR's anonymous legal mobile user. MU adopts the random number N_1 to authenticate VLR with the help of HLR. On Step 8 in the one-line authentication phase, MU sends VLR the challenge N_1. On Step 9, the ciphertext $[N_1||n_2||y_K||(C_M,S_M')||\rho_H||Dig]_{K_{HV}}$ is used to convince HLR that the VLR specified by MU has the fresh challenge N_1 because K_{HV} is the shared secret key between HLR and VLR and (C_M,S_M') is the signcryptext from MU. On Step 15.3, the ciphertext $[N_1||n_3||ID_V||R'||\sigma_H']_{SK'}$ is used as the HLR guarantee that the intended VLR actually receives N_1 sent on Step 8. Therefore, MU authenticates VLR with the help of HLR.

The subsequent login mutual authentication between MU and VLR is based on the share secret key RK_i. On Step 3 in subsequent login processes, after decrypting the authentication messages from MU by using RK_i, VLR authenticates MU by checking whether or not $h(h^{(n-i+1)}(n_1))= L$.

Mutual Authentication between VLR and HLR

The mutual authentication between VLR and HLR is based on the secret long-term key K_{HV}. On the Step 4.3 in login authentication processes, MU tells HLR who the intended VLR is by using the signcryptext C_M containing the intended VLR's identity ID_V. On the Step 9, VLR transmits $[N_1||n_2||y_K||(C_M,S_M')||\rho_H||Dig]_{K_{HV}}$ to HLR. After decrypting the ciphertext $[N_1||n_2||y_K||(C_M,S_M')||\rho_H||Dig]_{K_{HV}}$, HLR authenticates who the VLR is because only VLR and HLR know the long-term key K_{HV}. On Step 12.4, after decrypting and validating the signcryptext (C_M,S_M'), HLR actually check whether the ID_V decrypted from C_M is consistent with the current VLR's identity. If it is consistent, HLR authenticates VLR.

After decrypting the ciphertext $[[N_1||n_3||ID_V||R'||\sigma_H']_{SK'}||y_K||n_2||N_1||RK_1]_{K_{HV}}$ from HLR, VLR checks the freshness of n_2. If n_2 is the same as he/she sent on Step 9, VLR authenticates HLR since K_{HV} is the shared secret key between HLR and VLR.

Mutual Authentication between MU and HLR

In login authentication processes, HLR authenticates MU based on the fact that only the MU knows the private key x_K. After validating ρ_H, HLR recovers ρ_M to guarantee that only the MU actually knows x_K. After checking the signature A, HLR confirms that the specified MU is the one knowing x_K.

The MU authenticates HLR by the Diffie-Hellman key Y^{x_H} mod P. After decrypting the correct message from the ciphertext $[N_1||n_3||ID_V||R'||\sigma_H']_{SK'}$ and checking the freshness of N_1 and n_3, MU authenticates the identity of HLR. The reason is that only HLR and the one choosing Y can compute $SK' = H(Y^{x_H} \bmod p)$.

User Identity Privacy

User identity privacy means during the roaming services, no one, except HLR, can obtain MU's real identity ID_M. In our protocol, only the signcryptext C_M contains ID_M, the confidentiality protects MU's identity. Fortunately, Theorem 1 shows that our protocol also provides IND-CCA2, C_M releases nothings about ID_M. In the login authentication phase, MU adopts anonymous public key y_K to hide the actual identity. In the subsequent login authentication phase, MU adopts aliases, ID_i's, on the communications with VLR. Hence, our protocol provides the user identity privacy.

User Unlinkability

Moreover, the proxy public key y_K is updated with $y_{K,new}$ after finishing each login authentication process. For an adversary, it is hard to determine whether two instances belong to the same MU by linking proxy public keys. Moreover, during the subsequent login communication between MU and VLR, different aliases, ID_i's, are used for different communications. So, our protocol provides the unlinkable identities protection.

5. Comparison and Discussions

Table 1 shows the security property comparison among Youn and Lim, Chen et al., and our protocols. The major contribution of our protocol is to reduce the trust HLR assumption to enhance the non-repudiation property for MUs. In both Youn and Lim's and Chen et al.'s protocols, the HLR must be trust completely, because HLR knows MU's proxy private key. Both MU and HLR shares the user's proxy private key, so users' non-repudiation is completely based on HLR's trust. However, this assumption is impractical and allows HLR to damage MUs' benefit by forging any roaming records.

To enhance the users' non-repudiation by reducing trust HLR assumption, in our protocol, only MU knows his/her proxy private key x_K. To show the proxy authorization on the proxy key y_K and provide MUs' anonymity protection, the anonymous concurrent signcryption scheme to exchange the promise σ_H of $\rho_H = (s_H, V_H)$ of HLR and the promise σ_M of $\rho_M = (s_M, V_M)$ of MU. After exchanging the promises of signatures, MU first obtains the Schnorr-like signature $\rho_H = (s_H, V_H)$ of HLR as the authentication message. HLR cannot obtain he Schnorr signature $\rho_M = (s_M, V_M)$ until MU uses the Schnorr-like signature $\rho_H = (s_H, V_H)$. After obtaining $\rho_H = (s_H, V_H)$ from MU, HLR also obtains the evidence $\rho_M = (s_M, V_M)$ to show that MU actually applies the proxy public key y_K. The evidence ρ_M is used to not only protect the HLR's right but also enhance the MUs' non-repudiation by reducing trust HLR assumption. Therefore, the non-repudiation of our protocol is stronger than the non-repudiation both in Youn and Lim, and Chen et al. protocols.

Both our and Chen et al.'s protocols avoid the exhausted search problem during the subsequent login authentication. In our protocol, after receiving $(ID_i, [h^{(n-i+1)}(n_1)]_{RK_i})$ from MU in subsequent login authentication processes, the alias ID_i helps VLR easily find the session key RK_i to decrypt the ciphtertext $[h^{(n-i+1)}(n_1)]_{RK_i}$. Unfortunately, Youn and Lim's protocol suffers the exhausted search problem during the subsequent login authentication.

Both our and Chen et al.'s protocols provide not only the first login but also subsequent login MU unlinkability to authenticate anonymous MUs. In the subsequent login authentication in our or Chen et al.'s protocols, MU submits different aliases ID_i's to login VLR, so the subsequent login users' unlinkability is provided. However, in the subsequent login authentication in Youn and Lim's protocol, the anonymous MU has no chance to change the alias, so their protocol does not provide the subsequent login user unlinkability.

Moreover, in our protocol, the anonymous concurrent signcryption scheme is also proposed. The anonymous concurrent signcryption scheme satisfies the correctness, unlinkability, fairness, unforgability, and IND-CCA2 confidentiality properties. Among those properties, the fairness property is useful to protect MU's and HLR's benefits in the delegation-based authentication protocol for PCSs. The users' benefit is protected since only the delegated MU knows the proxy private key. The HLR's benefit is protected since HLR obtains the evidence $\rho_M = (s_M, V_M)$ only after the MU actually uses the proxy public key and the Schnorr-like signature $\rho_H = (s_H, V_H)$. By using $\rho_M = (s_M, V_M)$, the HLR really proves that the proxy public key is actually form the MU.

Table 1: Security Property Comparison among Youn and Lim, Chen et al., and Our Protocols

Protocols / Property	Youn and Lim	Chen et al.	Our protocol
Non-repudiation	Yes with Trust HLR	Yes with Trust HLR	Yes with Semi-Trust HLR
Mutual authentication	Yes	Yes	Yes
Session key security	Yes	Yes	Yes
User identity privacy	Yes	Yes	Yes
User unlinkability	On-line	On-line/subsequent login	On-line/subsequent login
No exhausted search	No	Yes	Yes
Semi-trust HLR	No	No	Yes
Unlinkability	N/A	N/A	Yes
Correctness	N/A	N/A	Yes
Fairness	N/A	N/A	Yes
Unforgeability	N/A	N/A	Yes
Confidentiality	N/A	N/A	Yes

6. **Conclusions**

By utilizing our concurrent signcryption scheme, the trust HLR assumption is reduced to the semi-trust HLR assumption in our new protocol. Consequently, the users' non-repudiation is enhanced. Moreover, the performance of the subsequent login authentication is improved by removing the exhaustive search problem when VLR has to find the alias of an anonymous MU. Since the mobile user adopts different aliases to login VLR in subsequent login authentication every time, our protocol supposes the subsequent login user's unlinkability to protect the user identity privacy. The concurrent signcryption scheme also supposes the fairness property to fairly protect mobile users' and HLR's benefits.

Acknowledgement

This work was supported in part by the National Science Council of the Republic of China under contract NSC 101-2221-E-032 -043 -.

References

[1] W.-B. Lee, C.-K. Yeh, "A new delegation-based authentication protocol for use in portable communication systems, " *IEEE Transactions on Wireless Communications*, Vol. 4, No.1, pp. 57-64, 2005.

[2] K. Al-Tawill, A. Akrami, H. Youssef, "A new authentication protocol for GSM networks," in *Proceedings of 23rd Annual IEEE Conference on Local Computer Networks*, pp. 21-30, 1999.

[3] C.-H. Lee, M.-S. Hwang, W.-P. Yang, "Enhanced privacy and authentication for the global system for mobile communications," *Wireless Networks*, Vol. 5, No. 4, pp. 231-243, 1999.

[4] M. J. Beller, L.-F. Chang, Y. Yacobi, "Privacy and authentication on a portable communication system," *IEEE Journal on Selected Areas in Communications*, Vol. 11, No. 6, pp. 821-829, 1993.

[5] C.-C. Lo, Y.-J. Chen, "Secure communication mechanisms for GSM networks," *IEEE Transactions on Consumer Electronics*, Vol. 45, No. 4, pp. 1074-1080, 1999.

[6] T.-F. Lee, S.-H. Chang, T. Hwang, S.K. Chong, "Enhanced delegation-based authentication protocol for PCSs," *IEEE Transactions on Wireless Communications*, Vol. 8, No. 5, pp. 2166-2171, 2009.

[7] T.-Y. Youn, J. Lim, "Improved delegation-based authentication protocol for secure roaming service with unlinkability," *IEEE Communications Letters*, Vol. 14, No. 9, pp. 791-793, 2011.

[8] R.-C. Wang, W.-S. Juang, C.L. Lei, "A privacy and delegation-enhanced user authentication protocol for portable communication systems," *International Journal of Ad Hoc and Ubiquitous Computing*, Vol. 6, No. 3, pp. 183-190, 2011.

[9] S.-J Hwang, C.-H You, "Weakness of Wang et al.'s privacy and delegation enhanced user authentication protocol for PCSs," *CSCIST 2011 and iCube 2011*, Taipei, 2011.

[10] H.-B Chen, Y.-H. Lai, K.-W Chen, W.-B Lee, "Enhanced delegation based authentication protocol for secure roaming service with synchronization," *Journal of Electronic Science and Technology,* Vol. 9, No. 4, pp. 345-351, 2011.

[11] S.-J. Hwang, Y.-H. Sung, "Confidential deniable authentication using promised signcryption," *Journal of Systems and Software*, Vol. 84, pp.1652-1659, 2011.

[12] K. Nguyen, "Asymmetric Concurrent Signatures," in *Proceedings of Information and Communications Security Conference* (ICICS 2005), 3783 of Lecture Notes in Computer Science, New York: Springer-Verlag, pp. 181-193, 2005.

[13] C. Schnorr, "Efficient signature generation by smart cards," *Journal of Cryptology*, No. 3, Vol. 4, pp.161-174, 1991.

[14] D. Pointcheval, J. Stern, "Security arguments for digital signatures and blind signatures," *Journal of Cryptography*, No. 3, Vol. 13, pp. 361-396, 2000.