A Generic Approach for Providing Revocation Support in Secret Handshake*

Yanjiang Yang[1], Haibing Lu[2], Jian Weng[3,4,**],
Xuhua Ding[5], and Jianying Zhou[1]

[1] Institute for Infocomm Research, Singapore
[2] Dept. of Operations Management and Information Systems,
The Leavey School of Business, Santa Clara University, USA
[3] Department of Computer Science, and Emergency Technology Research Center
of Risk Evaluation and Prewarning on Public Network Security,
Jinan University, China
[4] Shanghai Key Laboratory of Integrate Administration Technologies
for Information Security, Shanghai, China
[5] School of Information Systems, Singapore Management University
{yyang,jyzhou}@i2r.a-star.edu.sg, hlu@scu.edu,
cryptjweng@gmail.com, xhding@smu.edu.sg

Abstract. Privacy protection and user revocation are essentially conflicting requirements in many cryptographic protocols. It is a particularly challenging problem to harmonize them in a secret handshake protocol that is geared to offering strong privacy protection on the participants' group membership in the protocol execution. In this paper, we study this problem and propose a generic approach to provide revocation support in secret handshake protocols, without sacrificing the notion of privacy preserving. The main building block of our approach is CGC (Confidential Group Communication), a primitive formulated in this paper, and we present a concrete instantiation so as to realize our generic approach.

1 Introduction

Users nowadays are much more concerned with individual privacy than years ago. The growing privacy awareness calls for privacy-preserving techniques that can enable users to accomplish the desired functions over the Internet without compromising their privacy. Secret Handshake (SHS) firstly introduced in [3] is one such technique. In its simplest form, a secret handshake protocol allows two

* This work was supported by the National Science Foundation of China under Grant Nos. 60903178, 61070217, 61005049, 61133014 and 61272413, the Fok Ying Tung Education Foundation under Grant No. 131066, the Fundamental Research Funds for the Central Universities under Grant No. 21610204, the Guangdong Provincial Science and Technology Project under Grand No. 2010A032000002, and the Opening Project of Shanghai Key Laboratory of Integrate Administration Technologies for Information Security under Grand No. AGK2011003.

** Corresponding author.

T.W. Chim and T.H. Yuen (Eds.): ICICS 2012, LNCS 7618, pp. 276–284, 2012.

members from the same group, each holding a membership credential, to establish a shared session key and authenticate each other, with the following two requirements. Firstly, an eavesdropper observing a handshake session learns no meaningful information about the participants, including whether they belong to the same group and whether the handshake is successful or not. Secondly, a non-member (i.e., a user not in the group) cannot pretend to be a member. Compared to other privacy-preserving entity authentication primitives, a secret handshake protocol is "affiliation-hiding" in the sense that the protocol does not reveal which group the handshake participants belong to. Since its introduction by Balfanz et. al. in [3], the notion of secret handshake has attracted enormous attention due to many interesting applications, such as private mutual authentication between two secret government agents or two private club members.

User revocation is an indispensable component for any practical secret handshake scheme. A member may leave or be evicted from a group. It is also likely that a user's secret credential is compromised. These scenarios demand the system to have a timely and effective revocation mechanism such that those members' credential should be nullified and the credential holders cannot run secret handshake protocols successfully with other group members. Despite that numerous secret handshake schemes have been proposed in the literature, including [1, 3, 4, 10, 6–9], the revocation problem is neglected in almost all of those using *reusable credentials* except in [7, 10, 9]. However, the revocation in [10, 7] requires a synchronized rekey protocol upon all users, which is unscalable and inefficient, while the solution in [9] is problematic, because the approach of credential-validity-checking does not rule out the possibility that the revoked user authenticates her counterpart first.

Our Contributions. In this work, propose a generic approach to provide revocation support in secret handshake protocols. In specific, we first introduce a new cryptographic primitive called CGC (Confidential Group Communication), which allows two users with the same valid group membership to establish a secret channel. User revocation is handled in CGC in a way that revoked users are excluded from accessing such a secret channel. Building on top of CGC, we then propose a generic approach to provide revocation support in secret handshake schemes, with the basic idea being to execute secret handshake in the secret channel established by CGC. Our approach does not make any changes on the underlying secret handshake protocol, thus it is applicable to all existing and future reusable credential based secret handshake schemes.

2 System Model

A secret handshake scheme considers a set of security-sensitive user groups operating under a global authority $\mathcal{GA}$ who is in charge of setting up global parameters. In an open network environment, e.g. the Internet, a member in a group expects to communicate with another group member in a private fashion such that they end up agreeing on a shared secret when they satisfy an agreed policy, e.g. the same group membership and/or certain attributes. Otherwise, none of them can determine her counterpart's membership information. We are interested in secret handshake supporting user revocation, where $\mathcal{GA}$ revokes a user whenever the user leaves her group voluntarily or is evicted. As a result of revocation, the revoked user is of no difference from non-members, e.g., she cannot successfully run a secret handshake with a legitimate member. The following definition is adapted from [1].

Definition 1. *A* secret handshake scheme with revocation *(SHS-R) consists of five algorithms {Setup, CreateGroup, AddMember, Handshake, Revoke} as described below.*

- *`Setup`: Given a security parameter 1^κ, the algorithm, executed by $\mathcal{GA}$, outputs a suite of global parameters denoted as `params` which are shared by all groups.*
- *`CreateGroup`: Taking `params` as input, a group manager GM runs this algorithm to initialize a group's public information G and the group's secret key sk_G.*
- *`AddMember`: GM runs this algorithm with a user U requesting to join its group. Taking as input public group information G and group secret key sk_G, the algorithm assigns a credential $cred_U$ to U as a result of user admission.*
- *`Handshake`: Two players A and B run this protocol interactively with their respective private input $cred_A$ and $cred_B$. When the protocol ends, if A and B satisfy each other's handshake rules, they successfully share a common secret key, and mutually authenticate each other (if needed). For all other scenarios, the handshake fails.*
- *`Revoke`: When a group member U with credential $cred_U$ is revoked, GM runs this algorithm to update the group's public revocation list $\mathcal{R}$ such that $\mathcal{R} = \mathcal{R} \cup \{U, cred_U\}$, and U is excluded from the group and is a non-member.*

A SHS-R scheme must satisfy the following core security properties:

- **Correctness.** Honest members satisfying the handshake rules will always successfully complete the handshake.
- **Impersonator resistance.** An adversary not satisfying the rules of the handshake is unable to impersonate a group member and to successfully establish a shared secret with an honest group member.
- **Detector resistance.** An adversary not satisfying the rules of the handshake cannot decide whether some honest party satisfies the rules or not. *Affiliation hiding* is implicit in this property.
- **Unlinkability.** It is not feasible to tell whether two executions of the handshake protocol were performed by the same party or not, even if both of them were successful.

3 A Generic Approach

As shown above, there exist several secret handshake schemes using reusable credentials, but the revocation issue is not well addressed, e.g., [1, 5, 8] do not consider revocation at all, while the revocation method in [9] is not secure. In this section, we give a generic approach to provide revocation support to these secret handshake schemes. In particular our approach transforms a secret handshake scheme without revocation support into a SHS-R scheme. As such, the approach also facilitates the development of new protocols as the protocol designers are relieved from considering user revocation.

The building block of our approach is *Confidential Group Communication* or CGC, a primitive defined as an encryption scheme for a group of users, allowing group members to communicate with one another in a secret manner such that the conversation remains confidential to any non-members. In other words, CGC enables a confidential channel exclusively for all group members. As a result, each group member can send and receive messages via the channel, and only group members can access the communicated messages. Moreover, CGC revokes members such that once a member is revoked, she immediately becomes a non-member, losing the access to the channel established by other group members.

3.1 Confidential Group Communication

We formally specify Confidential Group Communication CGC as follows:

Definition 2. *A CGC scheme is defined as the following six algorithms.*

– *`CGC.Setup`: Given a security parameter 1^κ, a global authority $\mathcal{GA}$ execute this algorithm to set up public global information `params`.*
– *`CGC.CreateGroup`: Taking `params` as input, a group manager GM runs this algorithm to initialize a group's public information G and the group's secret key K_G .*
– *`CGC.UserJoin`: Taking as input group secret key K_G and user identifier U, it outputs a secret member key sk_U. Depending on instantiations, sk_U could be unique to U or a shared secret among all members.*
– *`CGC.Enc`: Taking as input a message m and a member key sk_U, it outputs a ciphertext c.*
– *`CGC.Dec`: Taking as input sk_U and a ciphertext c, it outputs the corresponding plaintext m.*
– *`CGC.Revoke`: Taking as input user identifier U, GM executes this algorithm to revoke U by outputting the updated revocation list of the group. As a result of revocation, sk_U is no longer useful (in performing `CGC.Enc` and `CGC.Dec`).*

We impose the following security requirements upon a CGC scheme:

• **Plaintext Secrecy.** The `CGC.Enc` algorithm should keep the secrecy of the encrypted plaintexts. In particular, CPA (chosen plaintext attack) security suffices for our use in this work.

• **Key Privacy.** The CGC scheme must also be *key private*, which intuitively means that it is infeasible to learn under which key a ciphertext is generated. In our setting, key privacy implies affiliation hiding, i.e., the ciphertexts do not disclose information on the group to which the ciphertexts are intended.

An Instantiation. We note that public-key broadcast encryption cannot directly instantiate CGC, because it cannot attain key privacy[1]. Our instantiation still needs to make use of public-key broadcast encryption, but using it for key update in case of user revocation. Specifically, we require group members to share a secret key, and instantiate the encryption algorithm by a key private symmetric key encryption scheme. The main issue at this point is how to enable the group members to update the shared secret key in case of user revocation. Our solution is to use public-key broadcast encryption, such that a new key is encrypted under the broadcast encryption. As public-key broadcast encryption supports user revocation, all group members except the revoked members can decrypt and get the new key. We stress that in this instantiation, since broadcast encryption is not directly involved in the encryption algorithm, it is not required to be key-private, thus any existing public-key broadcast encryption scheme can be used as long as it supports user revocation. The details of the instantiation are below:

- `CGC.Setup`: Determines a key-private symmetric key encryption scheme $\mathtt{SE}$, e.g., AES-CBC , and sets $params = \langle \mathtt{SE} \rangle$.
- `CGC.CreateGroup`: GM selects a secret key $\mathtt{SE}.k_G$ for $\mathtt{SE}$; also, determines a public-key broadcast encryption scheme $\mathtt{PBE}_G$, and establishes the public key $\mathtt{PBE}_G.pk$ and a set of user private keys $\{\mathtt{PBE}_G.sk_i\}_i$ for $\mathtt{PBE}_G$. Sets the group public information as $G = \langle \mathtt{SE}, \mathtt{PBE}_G, pub_G \rangle$, and sets the group secret key as $K_G = \langle \mathtt{SE}.k_G, \{\mathtt{PBE}_G.sk_i\}_i \rangle$.
- `CGC.UserJoin`: To enrol a user, U, set its member secret key $sk_U = \langle \mathtt{SE}.k_G,$ $\mathtt{PBE}_G.sk_\ell \rangle$, where $\mathtt{PBE}_G.sk_\ell$ is a un-assigned user private key of $\mathtt{PBE}_G$.
- `CGC.Enc`: Given a message m, set the ciphertext as $c = \mathtt{SE}.Enc(m, \mathtt{SE}.k_G)$, where $\mathtt{SE}.Enc(\cdot)$ is the encryption algorithm of $\mathtt{SE}$.
- `CGC.Dec`: Given a ciphertext c, decrypt c as $m = \mathtt{SE}.Dec(c, \mathtt{SE}.k_G)$, where $\mathtt{SE}.Dec(\cdot)$ is the decryption algorithm of $\mathtt{SE}$.
- `CGC.Revoke`: Upon revocation of user U, update the revocation list of $\mathtt{PBE}_G$ to include U (the revocation list is a part of $\mathtt{PBE}_G.pk$); update $\mathtt{SE}.k_G$ by assigning a new value, and encrypt it using the public-key broadcast encryption as $rekey\_msg = \mathtt{PBE}_G.Enc(\mathtt{SE}.k_G)$; publish $rekey\_msg$ in a public directory, such that group members can retrieve and decrypt it using their respective $\mathtt{PBE}_G.sk$.

Security. Since the above CGC scheme relies on a key private symmetric key encryption scheme for data encryption, it straightforwardly inherits plaintext secrecy and key privacy. Thus, we have the following theorem.

Theorem 1. *The above instantiation is a secure CGC scheme, given that the underlying symmetric key encryption* SE *is plaintext secret and key private.*

[1] As far as we know, all existing public-key broadcast encryption schemes in the literature are not key private. The key private broadcast encryption scheme in [2] is more precisely multicast encryption.

3.2 Our Approach

Basic Idea. As indicated earlier, we cannot expect to invoke explicit credential validity checking to address the revocation issue in secret handshake. Thus, our rationale is to eliminate the possibility that a revoked member can participate in a secret handshake protocol. In particular, equip a group with a CGC scheme, such that group members run the secret handshake protocol in a secret channel established by the CGC scheme. User revocation is implicitly handled by the CGC scheme, which guarantees that only group members can access the secret channel, while non-members including revoked members are excluded from the channel.

Concretely, given a (general) secret handshake scheme without revocation support Γ, we compile it into a SHS-R scheme Γ' as follows. Exploiting a CGC scheme, e.g., the above instantiation, all group members share a secret key for a (global) symmetric key encryption scheme, and a private key for the group's public-key broadcast encryption scheme. When running Γ with a peer, a group member encrypts the messages that she needs to send out with the symmetric key encryption; while upon receipt of a message from her peer, she decrypts the message first, and then behaves upon the decrypted message following the specification of Γ. User revocation is handled in a straightforward way by the CGC scheme.

We note that in case of user revocation, the remaining group members do not need to update their shared secret key as long as they have no plan to participate in handshakes. In other words, it suffices for a group member to update her key right before her participation in a handshake protocol. Due to the use of broadcast encryption, even if she misses some previous rekey messages, she can still get the latest key. Therefore, using the above CGC scheme in our approach allows group members to synchronize key update on the necessity basis, contrasting to [7, 10] which require strict synchronization.

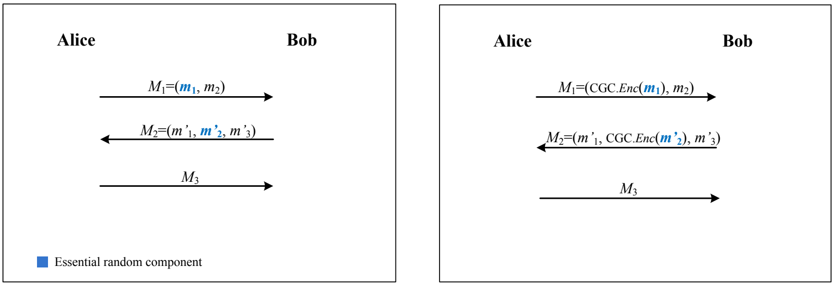
Caveat. We have two comments. (1) If the original secret handshake scheme Γ itself involves a shared secret (e.g., [5]), we make that secret a persistent quantity in Γ', and it no longer needs to be updated in case of user revocation. (2) The approach seems not applicable to the secret handshake with dynamic matching scheme in [1], which allows groups members from certain different groups to perform handshake. In fact, the approach still works if we let the groups whose members are allowed to make handshake share a secret key.

A Revisit. Let us briefly examine the security of the approach. Adversaries to a secret handshake protocol include not only non-members, but also group members who do not participate the handshake or who do not satisfy the handshake rules in question. For a non-member adversary, since the CGC encryption is key private, intuitively the adversary cannot get extra information. But for a member adversary, the situation is complicated. In particular, having access to the secret channel, the adversary can see the execution of the handshake protocol (i.e., Γ). Even though the adversary cannot learn useful information from the execution of Γ itself (as Γ is a secret handshake protocol), a subtle vulnerability is the following: as the handshake protocol (i.e., Γ) is supposed to be executed in a secret channel, once the adversary can decide that what it sees is a handshake, then the secret channel must be established by group members of the group it belongs to. This compromises affiliation hiding.

To be concrete, suppose that the original secret handshake protocol Γ involves a party sending out a message, which consists of a 80-bit component and a 120-bit component. If a member adversary observing an execution of Γ' intercepts a corresponding message, and decrypts using its own secret key and gets a message of the above format, then the adversary can know that Γ' is running between two members of its group. In this example, the attack still works even if the message of Γ is of a single component, but with a recognizable structure, e.g., it is a timestamp.

We have to rectify this issue. Our intuition is that it is in fact not necessary to encrypt all messages of Γ, and it suffices to only encrypt the message that is essential for key establishment and mutual authentication. We observe that within all the messages sent out by a party, there must be at least one random component. Depending on specific schemes, it could be an element intended for key establishment or a randomized credential element, or even a nonce . Such random components must exist in a secret handshake protocol, due to the need to be secure against replay attacks and the need to randomize a user's credential. The characteristic of such a component is that it is a uniformly distributed element within its domain (e.g., a finite field).

Based on this observation, we revise our above approach to be such that each party, when running Γ, selects a single random component within her messages that is critical for the handshake, and encrypts with the CGC scheme. Figure 1 depicts the idea, where m_1 (resp. m_2') is one of the essential random components that Alice (resp. Bob) needs to send out; and only m_1 (resp. m_2') needs to be encrypted among Alice's (resp. Bob's) messages, leaving other components intact. For this approach to work, of course we assume that the secret handshake protocol Γ is *natural* , in the sense that during the course of Γ, there will be no


Alice
Bob
M_1=(m_1, m_2)
M_2=(m'_1, m'_2, m'_3)
M_3
■ Essential random component
Alice
Bob
M_1=(CGC.Enc(m_1), m_2)
M_2=(m'_1, CGC.Enc(m'_2), m'_3)
M_3

(a) Original secret handshake scheme Γ

(b) Secret handshake scheme with revocation Γ'

Fig. 1. Generic Approach of Transforming Γ to Γ'

other messages that de-randomize the component (e.g., the protocol does not involve the appearance of the component or its hash value in other messages). Note that if the bit length of the component to be encrypted does not match that of the encryption scheme, padding of random bits applies and the random padding will be ignored at the decryption side. Finally, it is clear that the effect of the CGC scheme for revocation remains.

Security. For security of our approach, we have the following theorem, and the proof can be found in the full version.

Theorem 2. *If Γ is a secret handshake protocol and the CGC scheme satisfies plaintext secrecy and key privacy, then the resulting Γ' by applying our approach on Γ is a secret handshake protocol.*

Comparison with [10]. The secret handshake protocol in [10] also relies on group members sharing a secret to handle user revocation. Our proposal in this work distinguishes from [10] mainly as follows. (1) First of all, [10] presents a concrete (membership-credential based) scheme, while our proposal is a generic approach intending to provide revocation support to all reusable credential based secret handshake schemes. (2) Secondly, even though both require group members to share a secret key for symmetric key encryption, there are distinctions on the way the shared secret key is used. Specifically, in [10] the participants in a handshake run Diffie-Hellman key exchange first to generate an ephemeral key, which is then XORed with the shared secret key to generate a session key, and the session key is used for symmetric key encryption. As a result, the symmetric key encryption is not necessarily key private. In contrast, the shared secret key is directly used for symmetric key encryption in our approach, thus the encryption must be explicitly key private. (3) In [10], the entire handshake session is carried out in the confidential channel established by the symmetric key encryption. But as analyzed earlier, this is not secure in our approach as we are considering *general* secret handshake where the adversary can compromise group members and can be active (in [10] such an adversary can only be passive). (4) Finally, as mentioned earlier the method in [10] requires strict synchronization for key update for group members (in case of user revocation), while our approach solves this problem and group members update the shared key on the necessity basis, due to the use of public-key broadcast encryption.

References

1. Ateniese, G., Blanton, M., Kirsch, J.: Secret Handshakes with Dynamic and Fuzzy Matching. In: Proc. Network and Distributed System Security Symposium, NDSS 2007 (2007)
2. Barth, A., Boneh, D., Waters, B.: Privacy in Encrypted Content Distribution Using Private Broadcast Encryption. In: Di Crescenzo, G., Rubin, A. (eds.) FC 2006. LNCS, vol. 4107, pp. 52–64. Springer, Heidelberg (2006)
3. Balfanz, D., Durfee, G., Shankar, N., Smetters, D., Staddon, J., Wong, H.: Secret Handshakes from Pairing-Based Key Agreements. In: Proc. IEEE Security & Privacy, pp. 180–1966 (2003)

4. Castelluccia, C., Jarecki, S., Tsudik, G.: Secret Handshakes from CA-Oblivious Encryption. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 293–307. Springer, Heidelberg (2004)
5. Hoepman, J.H.: Private Handshakes. In: Stajano, F., Meadows, C., Capkun, S., Moore, T. (eds.) ESAS 2007. LNCS, vol. 4572, pp. 31–42. Springer, Heidelberg (2007)
6. Jarechi, S., Kim, J., Tsudik, G.: Authenticated Group Key Agreement Protocols with the Privacy Property of Affilation-hidding. In: Proc. CT-RSA Conference (2007)
7. Jarecki, S., Liu, X.: Unlinkable Secret Handshakes and Key-Private Group Key Management Schemes. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 270–287. Springer, Heidelberg (2007)
8. Sorniotti, A., Molva, R.: A Provably Secure Secret Handshake with Dynamic Controlled Matching. In: Gritzalis, D., Lopez, J. (eds.) SEC 2009. IFIP AICT, vol. 297, pp. 330–341. Springer, Heidelberg (2009)
9. Sorniotti, A., Molva, R.: Secret Handshakes With Revocation Support. In: Lee, D., Hong, S. (eds.) ICISC 2009. LNCS, vol. 5984, pp. 274–299. Springer, Heidelberg (2010)
10. Tsudik, G., Xu, S.: Flexible Framework for Secret Handshakes, Cryptology ePrint Archive, Report 2005/034