Generic One Round Group Key Exchange in the Standard Model

M. Choudary Gorantla[1], Colin Boyd[1], Juan Manuel González Nieto[1], and Mark Manulis[2]

[1] Information Security Institute, Faculty of IT, Queensland University of Technology
GPO Box 2434, Brisbane, QLD 4001, Australia.
Email: mc.gorantla@isi.qut.edu.au, {c.boyd,j.gonzaleznieto}@qut.edu.au
[2] Cryptographic Protocols Group, Department of Computer Science
TUDarmstadt & CASED, Germany.
Email: mark@manulis.eu

Abstract. Minimizing complexity of group key exchange (GKE) protocols is an important milestone towards their practical deployment. An interesting approach to achieve this goal is to simplify the design of GKE protocols by using generic building blocks. In this paper we investigate the possibility of founding GKE protocols based on a primitive called *multi key encapsulation mechanism (mKEM)* and describe advantages and limitations of this approach. In particular, we show how to design a one-round GKE protocol which satisfies the classical requirement of authenticated key exchange (AKE) security, yet without forward secrecy. As a result, we obtain the first one-round GKE protocol secure in the standard model. We also conduct our analysis using recent formal models that take into account both outsider and insider attacks as well as the notion of key compromise impersonation resilience (KCIR). In contrast to previous models we show how to model both outsider and insider KCIR within the definition of mutual authentication. Our analysis additionally implies that the insider security compiler by Katz and Shin from ACM CCS 2005 can be used to achieve more than what is shown in the original work, namely both outsider and insider KCIR.
Keywords. Group Key Exchange, Key Encapsulation Mechanism, Key Compromise Impersonation

1 Introduction

The computation of a common secret key among a group of members communicating over a public network is usually performed through a group key exchange (GKE) protocol. The secrecy (indistinguishability) of the established group key is modelled through the requirement called *authenticated key exchange (AKE) security* [12, 10, 11]. The classical AKE-security notion comes in different flavours depending on whether the protocol provides forward secrecy or not. Informally, a protocol with forward secrecy ensures that the secrecy of the group key is preserved despite possible user corruptions in the future. The user corruptions also have different flavours depending on whether only long-lived secrets are leaked (weak corruption [12, 23]) or the ephemeral, session-dependent information from the internal states can be revealed as well (strong corruption [14]). Bresson et al. [12] also define mutual authentication (MA) security as a desired notion of security for GKE protocols. The notion of MA-security requires that parties who complete the protocol should output identical session keys and that each party should be ensured of the identity of the other participating parties

However, as discussed by Katz and Shin [22] and Bohli et al. [2] the above security notions are not adequate if the GKE protocol should resist misbehaviour of its participants; in particular, preventing honest users from computing different keys and from having distinct views on the identities of other participants. Bresson and Manulis [14] merged the insider security requirements defined by Katz and Shin into their notion of MA-security in the presence of malicious insiders, improving upon the notion of MA-security from Bresson et al. [12]. This stronger MA-security can be obtained for any AKE-secure GKE protocol using Katz and Shin's compiler, which we refer to as the KS-compiler.

Recently, AKE- and MA-security notions have been further extended by Gorantla et al. [20] considering outsider and insider *key compromise impersonation resilience* (KCIR). Informally, a GKE protocol with KCIR ensures that an honest party cannot be impersonated by an adversary which has access to the private keys of other parties. The notion of outsider KCIR was modelled within AKE-security and insider KCIR was embedded into MA-security [20]. However, these notions defy the natural expectation that insider KCIR should imply outsider KCIR. It also remains unclear whether the KS-compiler can be used to achieve the additional insider KCIR or not.

KEY ENCAPSULATION MECHANISMS. Cramer and Shoup [16] formalised the concept of hybrid encryption which securely merges public and symmetric encryption techniques to encrypt messages. In short, the public key part called *key encapsulation mechanism (KEM)* is used to generate and encrypt a random session key, while *data encryption mechanism (DEM)* based on symmetric techniques is used to encrypt the actual message using that session key. The KEM primitive has been further extended to *multi KEM (mKEM)* by Smart [24]. mKEM is useful in scenarios where a single message should be encrypted for multiple recipients.

It is evident from their properties, especially by generating a random session key, that KEMs can also be utilized for the establishment of secure shared keys. In fact, the question of constructing key establishment protocols from KEMs has been investigated in the two-party setting: Gorantla et al. [21] provided generic constructions in both the directions based on signcryption KEMs, whereas Boyd et al. [6] presented a generic one-round protocol using plain encryption KEMs. The natural question is, thus, whether mKEMs in turn can be utilized for the design of GKE protocols? The non-triviality of the problem, in contrast to what one may think after having [21, 6], is the consideration of insider attacks which are not present in the two party case.

1.1 Our Contributions

In this paper, we extend the technique of Boyd et al. [6] to the group setting and present a generic one-round GKE protocol using mKEM as a building block which we prove AKE-secure in the standard model, yet without forward secrecy. The main reason for lack of forward secrecy is that mKEMs known today are not forward secure. Since forward secrecy is a desirable goal for some applications we also discuss the modified two-round version of the protocol based on one-time mKEM and digital signatures.

We enrich KCIR notions for GKE by including a definition of outsider KCIR into MA-security. In this way we achieve the natural implication between insider and outsider KCIR. We demonstrate the usefulness of the new notion by showing that the generic transformation of Bresson et al. [12] to achieve outsider MA-security is not sufficient for outsider KCIR.

Our new definition also highlights the separation between AKE-security and KCIR. As observed by Boyd and Mathuria [8, §5.5, p.166] two-party protocols can always achieve KCIR if each party encrypts its ephemeral pubic key with the partner's long-term public key. This holds also for the generic two-party protocol of Boyd et al.[6]. However, when we move to the group setting this observation does not hold any more with respect to AKE-security. As an example, we show that our one-round GKE protocol does not achieve AKE-security with outsider KCIR. Nevertheless, we show that our protocol still achieves MA-security with outsider KCIR. Thanks to the implication from insider KCIR to outsider KCIR we can show this by proving that our protocol when compiled with the KS-compiler achieves MA-security with insider KCIR.

Katz and Shin [22] informally mentioned that the KS-compiler could provide KCIR. Gorantla et al. [20] also speculated that when the KS-compiler is applied to the protocol of Boyd and González

Nieto [7] it would result in a GKE protocol secure under both AKE-security and MA-security with KCIR. However, we observe that the KS-compiler does not necessarily guarantee AKE-security with outsider KCIR.

1.2 Related Work

Boyd [4] presented three classes of one round key exchange protocols, which may be seen as a different paradigm to the classical Diffie-Hellman key exchange. In Class 1, the parties exchange random nonces in clear as their contributions towards the session key. A long-term shared symmetric key is then used to derive the session key. Constructing concrete GKE protocols in this class is not very interesting as the assumption that all the parties in the group initially share a common symmetric key seems unrealistic [5]. In Class 2, only one party uses confidential and authentication channels to send its nonce while all other parties send their nonces in clear. Concrete protocols in the Class 2 can be constructed using public key encryption and signature schemes. The protocol of Boyd and González Nieto [7] falls into this class. In Class 3, all the parties use confidential channels to send their nonces. Our proposed protocol falls into this class. A major drawback of protocols in all the three classes is that they cannot provide forward secrecy. However, a distinctive feature of Class 3 protocols is that they can be proven secure under the AKE-security notion without employing public key signatures. Hence, Class 3 protocols seem suitable to construct efficient deniable GKE protocols [3]. We do not formally explore this possibility in the current work.

Bohli et al. [2] defined *contributiveness* as another desired security notion for GKE protocols. This notion demands that a proper subset of insiders should not predetermine the resulting session key. Bresson and Manulis [14] strengthened this notion by considering strong corruptions where the ephemeral session state of an instance might also be revealed in addition to the long-term private key of the party. They also proposed compilers to achieve contributiveness in both weak and strong corruption models [13].

1.3 Organization

Section 2 reviews existing notions of AKE-security and MA-security with insider KCIR and also presents a new notion of MA-security with outsider KCIR. In Section 3, we describe our one-round GKE protocol based on mKEM and prove it AKE-secure without forward secrecy. Additionally, we mention how to extend this protocol with an additional round and obtain forward secrecy. In Section 4, we prove that the compiler by Katz and Shin [22] if executed with our protocol provides MA-security with outsider and insider KCIR. Section 5 gives security and efficiency comparison of existing GKE protocols. Appendix A describes the background concepts that serve as building blocks in our paper.

2 Security Model for GKE Protocols

In this section, we review existing notions of AKE-security and MA-security considered for GKE protocols. We also present our new notion of MA-security with outsider KCIR.

Let $\mathcal{U} = \{U_1, \ldots, U_N\}$ be a set of N parties. The protocol may be run among any subset of these parties. Each party U_j for $j \in [1, N]$ is assumed to have a pair of long-term public and private keys, (pk_j, sk_j) generated during an initialization phase prior to the protocol run. A GKE protocol π executed among $n \leq N$ users is modelled as a collection of n programs running at the n different parties in $\mathcal{U}$. Each instance of π within a party is defined as a session and each party may have multiple such sessions running concurrently.

Let π_U^i be the i-th run of the protocol π at party $U \in \mathcal{U}$. Each protocol instance at a party is identified by a unique session ID. We assume that the session ID is derived during the run of the protocol. The session ID of an instance π_U^i is denoted by sid_U^i. We assume that each party knows who the other participants are for each protocol instance. The partner ID pid_U^i of an instance π_U^i, is a set of identities of the parties with whom π_U^i wishes to establish a common group key. Note that pid_U^i includes the identity of U itself.

An instance π_U^i enters an *accepted* state when it computes a session key sk_U^i. Note that an instance may terminate without ever entering into an accepted state. The information of whether an instance has terminated with acceptance or without acceptance is assumed to be public. Two instances π_U^i and $\pi_{U'}^j$ at two different parties U and U' respectively are considered *partnered* iff (1) both the instances have accepted, (2) $\mathsf{sid}_U^i = \mathsf{sid}_{U'}^j$ and (3) $\mathsf{pid}_U^i = \mathsf{pid}_{U'}^j$.

The communication network is assumed to be fully controlled by an adversary $\mathcal{A}$, which schedules and mediates the sessions among all the parties. $\mathcal{A}$ is allowed to insert, delete or modify the protocol messages. If the adversary honestly forwards the protocol messages among all the participants, then all the instances are partnered and output identical session keys. Such a protocol is called a correct GKE protocol. In addition to controlling the message transmission, $\mathcal{A}$ is allowed to ask the following queries.

- $\mathsf{Execute}(\mathsf{pid})$ prompts a complete execution of the protocol among the parties in pid with a unique session ID sid. $\mathcal{A}$ is given all the protocol messages, modelling passive attacks.
- $\mathsf{Send}(\pi_U^i, m)$ sends a message m to the instance π_U^i. If the message is pid, the instance π_U^i is initiated with partner ID pid. The response of π_U^i to any Send query is returned to $\mathcal{A}$.
- $\mathsf{RevealKey}(\pi_U^i)$ If π_U^i has accepted, $\mathcal{A}$ is given the session key sk_U^i established at π_U^i.
- $\mathsf{Corrupt}(U_j)$ The long-term secret key sk_j of U_j is returned to $\mathcal{A}$. Note that this query returns neither the session key (if computed) nor any session specific internal state.
- $\mathsf{RevealState}(\pi_U^i)$ The ephemeral internal state of π_U^i is returned to $\mathcal{A}$. We assume that the internal state is erased once π_U^i has accepted.
- $\mathsf{Test}(\pi_U^i)$ A random bit b is secretly chosen. If $b = 1$, $\mathcal{A}$ is given $\kappa_1 = sk_U^i$ established at π_U^i. Otherwise, a random value κ_0 chosen from the session key probability distribution is given. Note that a Test query is allowed only once that too on an accepted instance.

Corrupted Parties, Corrupted Instances and Insiders. We call a *party* U corrupted if it has been issued a $\mathsf{Corrupt}$ query, while a *protocol instance* π_U^i is called corrupted if a $\mathsf{RevealState}(\pi_U^i)$ query has been asked. Note that there exist uncorrupted protocol instances at corrupted parties when the session specific ephemeral secrets are not revealed. A party is called an *insider* in a particular protocol run if both the party and the protocol instance are corrupted or if the adversary issues a $\mathsf{Corrupt}$ query to the party and then impersonates it i.e. when the adversary issues a Send query on behalf of π_U^i with a message m not previously output by π_U^i.

2.1 AKE-Security

The notion of freshness is central to the definition of AKE-security. Informally, a session is considered *fresh* if the session key established in that session is not trivially compromised. In Figure 1, we review different notions of freshness defined for GKE protocols in the literature. The first notion is a slightly revised notion from Katz and Yung [23], considering RevealState queries. This notion does not capture either forward secrecy or KCIR. Hence, the adversary is not allowed to corrupt any party associated with the test session. The second notion considers forward secrecy and may be seen as a stronger notion than that of Bresson and Manulis [14], where the corruption of a party U' is allowed after the session π_U^i has accepted. This differs from the notion of Bresson and Manulis, where the adversary is not allowed to issue a corrupt query until π_U^i and all its partners have accepted. The second notion may also be seen as a revised notion from Katz and Shin [22] considering RevealState queries. The third notion, which was recently defined by Gorantla et al. [20], considers both forward secrecy and outsider KCIR.

The basic notion of freshness (not considering forward secrecy or KCIR) [23]

An instance π_U^i is fresh if the following conditions hold:

1. the instance π_U^i or any of its partners has not been asked a RevealKey after their acceptance
2. the instance π_U^i or any of its partners has not been asked a RevealState before their acceptance
3. there has not been a $\mathsf{Corrupt}(U')$ query for any $U' \in \mathsf{pid}_U^i$ (including $U' = U$)

The notion of freshness with forward secrecy [22, 14]

An instance π_U^i is fs-fresh if the following conditions hold:

1. the instance π_U^i or any of its partners has not been asked a RevealKey after their acceptance
2. the instance π_U^i or any of its partners has not been asked a RevealState before their acceptance
3. there has not been a $\mathsf{Corrupt}(U')$ query for any $U' \in \mathsf{pid}_U^i$ (including $U' = U$) before π_U^i has accepted

The notion of freshness with outsider KCIR and forward secrecy [20]

An instance π_U^i is kcir-fs-fresh if the following conditions hold:

1. the instance π_U^i or any of its partners has not been asked a RevealKey after their acceptance
2. the instance π_U^i or any of its partners has not been asked a RevealState before their acceptance
3. If $\pi_{U'}^j \in \mathsf{pid}_U^i$ and $\mathcal{A}$ asked $\mathsf{Corrupt}(U')$, then any message that $\mathcal{A}$ sends to π_U^i on behalf of $\pi_{U'}^j$ must come from $\pi_{U'}^j$ intended to π_U^i .

Fig. 1. Notions of Freshness

Definition 1 (AKE-Security)**.** An adversary $\mathcal{A}^{AKE}$ against the AKE-security notion is allowed to make Execute, Send, RevealState, RevealKey and Corrupt queries in Stage 1. $\mathcal{A}^{AKE}$ makes a Test query to an instance π_U^i at the end of Stage 1 and is given a challenge key κ_b as described earlier. It can continue asking queries in Stage 2. Finally, $\mathcal{A}^{AKE}$ outputs a bit b' and wins the AKE security game if (1) $b' = b$ **and** (2) the instance π_U^i that was asked Test query remained fresh(or fs-fresh/kcir-fs-fresh correspondingly) till the end of $\mathcal{A}^{AKE}$'s execution. Let $\mathsf{Succ}_{\mathcal{A}^{AKE}}$ be the success probability of $\mathcal{A}^{AKE}$ in winning the AKE security game. The advantage of $\mathcal{A}^{AKE}$ in winning this game is $\mathsf{Adv}_{\mathcal{A}^{AKE}} = |2 \cdot \Pr[\mathsf{Succ}_{\mathcal{A}^{AKE}}] - 1|$. A protocol is called AKE-secure if $\mathsf{Adv}_{\mathcal{A}^{AKE}}$ is negligible in the security parameter k for any polynomial time $\mathcal{A}^{AKE}$.

Remark 1. It is clear that if a GKE protocol does not have forward secrecy, the AKE-security of the session key can be compromised by revealing the long-term key of a protocol participant. An adversary can perform a KCI attack on GKE protocols without forward secrecy by replaying messages of past successful executions or even by relaying messages from an honest party. The KCI attacks of Gorantla et al. [20] on Boyd and González Nieto [7] and Bresson et al. [9] protocols work in the same way. As the AKE-security notion with outsider KCIR implies that at most $n-1$

corruptions are allowed, it is necessary for a protocol realizing this notion to have at least (partial) forward secrecy when $n-1$ parties are corrupted or (full) forward secrecy. However, as evident by Gorantla et al.'s KCI attack on Al-Riyami and Paterson's protocol [1], having forward secrecy alone is not sufficient for a GKE protocol to have AKE-security with outsider KCIR. We leave it an open problem to define AKE-security notion with outsider KCIR and partial forward secrecy and to construct a GKE protocol realizing it.

2.2 MA-Security

We present two notions of MA-security with KCIR, one in the presence of only outsiders and another in the presence of insiders. The notion of MA-security with KCIR in the presence of insiders was defined by Gorantla et al. [20], while the notion of MA-security with outsider KCIR is new.

Definition 2 (MA-Security with Outsider KCIR)**.** An adversary $\mathcal{A}^{MA}$ against the MA-security of a correct GKE protocol π is allowed to ask Execute, Send, RevealState, RevealKey and Corrupt queries. $\mathcal{A}^{MA}$ violates the MA-security of the GKE protocol if at some point during the protocol run, there exists an uncorrupted instance π_U^i that has accepted with a key sk_U^i and another party $U' \in \mathsf{pid}_U^i$ that is uncorrupted at the time π_U^i accepts such that there are no insiders in pid_U^i and

1. there exists no instance $\pi_{U'}^j$ with $(\mathsf{pid}_{U'}^j, \mathsf{sid}_{U'}^j) = (\mathsf{pid}_U^i, \mathsf{sid}_U^i)$ **or**
2. there exists an instance $\pi_{U'}^j$ with $(\mathsf{pid}_{U'}^j, \mathsf{sid}_{U'}^j) = (\mathsf{pid}_U^i, \mathsf{sid}_U^i)$ that has accepted with $sk_{U'}^j \neq sk_U^i$.

The above definition implies that $\mathcal{A}^{MA}$ must be passive for any corrupted party in pid_U^i. Note that in a protocol execution with n parties, the above definition also implies that $\mathcal{A}^{MA}$ is allowed to corrupt up to $n-1$ parties.

Let $\mathsf{Succ}_{\mathcal{A}^{MA}}$ be the success probability of $\mathcal{A}^{MA}$ in winning the above security game. A protocol is said to provide MA-security with outsider KCIR if $\mathsf{Succ}_{\mathcal{A}^{MA}}$ is negligible in the security parameter k for any polynomial time $\mathcal{A}^{MA}$.

Definition 3 (MA-Security with Insider KCIR)**.** An adversary $\mathcal{A}^{MA}$ against the MA-security of a correct GKE protocol π is allowed to ask Execute, Send, RevealState, RevealKey and Corrupt queries. $\mathcal{A}^{MA}$ violates the MA-security of the GKE protocol if at some point during the protocol run, there exists an uncorrupted instance π_U^i (although the party U may be corrupted) that has accepted with a key sk_U^i and another party $U' \in \mathsf{pid}_U^i$ that is uncorrupted at the time π_U^i accepts such that

1. there is no instance $\pi_{U'}^j$ with $(\mathsf{pid}_{U'}^j, \mathsf{sid}_{U'}^j) = (\mathsf{pid}_U^i, \mathsf{sid}_U^i)$ **or**
2. there is an instance $\pi_{U'}^j$ with $(\mathsf{pid}_{U'}^j, \mathsf{sid}_{U'}^j) = (\mathsf{pid}_U^i, \mathsf{sid}_U^i)$ that has accepted with $sk_{U'}^j \neq sk_U^i$.

Note that this notion implies that there can be up to $n-2$ insiders (i.e. except U and U'). Let $\mathsf{Succ}_{\mathcal{A}^{MA}}$ be the success probability of $\mathcal{A}^{MA}$ in winning the above security game. A protocol is said to provide MA-security with insider KCIR if $\mathsf{Succ}_{\mathcal{A}^{MA}}$ is negligible in the security parameter k for any polynomial time $\mathcal{A}^{MA}$.

3 One Round GKE protocol from mKEM

Smart [24] formalised the notion of mKEM by extending the concept of KEM. Using an mKEM scheme, a user who wants to encrypt a large message to n parties can encapsulate a single session key

Computation

Each U_i executes an mKEM with public keys $\{pk_j | 1 \leq j \leq n; j \neq i\}$ as input and obtains the symmetric key and encapsulation pair (K_i, C_i)

$$(K_i, C_i) \leftarrow \mathsf{Encap}(\{pk_j | 1 \leq j \leq n; j \neq i\})$$

Broadcast

Each U_i broadcasts the computed encapsulation C_i along with its identity.

$$U_i \rightarrow \mathcal{U} \setminus \{U_i\} : \quad C_i, U_i$$

Key Computation

1. Each U_i executes the decapsulation algorithm using its private key sk_i and on each of the incoming encapsulations C_j and obtains the symmetric keys K_j, where $1 \leq j \leq n$, $j \neq i$.

$$K_j \leftarrow \mathsf{Decap}(sk_i, C_j) \text{ for each } 1 \leq j \leq n, \ j \neq i$$

2. Each U_i then computes the session ID as the concatenation of all the outgoing and incoming messages exchanged i.e. $\mathsf{sid} = (C_1 \| \cdots \| C_n \| \mathcal{U})$, where $\mathcal{U}$ is the set of identities of all the n users.
3. The session key κ is then computed as

$$\kappa \ = f_{K_1}(\mathsf{sid}) \oplus f_{K_2}(\mathsf{sid}) \oplus \cdots \oplus f_{K_n}(\mathsf{sid})$$

where f is a pseudo random function.

Fig. 2. A generic GKE protocol from mKEM

to all the parties at once and then apply a DEM with the session key to encrypt the actual message. Smart also defined the notion of indistinguishability under chosen ciphertext attacks (IND-CCA) for mKEM. The definition and security model for mKEM have been reviewed in Appendix A.1.

In Figure 2, we present a generic construction of GKE protocol based on mKEM. The parties can establish the group session key by executing an mKEM in parallel. Let $\mathcal{U} = \{U_1, U_2, \cdots, U_n\}$ be the set of protocol participants. The protocol uses an mKEM scheme (KeyGen, Encap,Decap). Let (pk_i, sk_i) be the public-private key pair of the party U_i, generated using the KeyGen algorithm. Each party starts the protocol by running the Encap algorithm and then broadcasts the encapsulation C_i to the other parties in the group. Upon receiving the encapsulations each party runs the Decap algorithm for each encapsulation intended for it and retrieves the symmetric keys. The session ID is defined as the concatenation of all the encapsulations along with the group identity $\mathcal{U}$.

The session key is finally computed by each party from the symmetric key it has generated during the Encap algorithm and all the symmetric keys decapsulated. A pseudo random function (PRF) f is used to derive the session key. Note that the session key derivation in our protocol is slightly different from the approach used in Boyd et al. [6]. In Boyd et al.'s protocol a randomness extraction function is first applied to the symmetric keys K_i's before using them as seeds to a PRF to derive the session key. In our protocol, we directly use the symmetric keys generated by the IND-CCA secure mKEM as seeds to f to simplify the protocol design. As shown in the proof below this does not effect the security of the protocol.

3.1 Proof of Security

Theorem 1. *The protocol in Figure 2 is AKE-secure without forward secrecy as per Definition 1 assuming the underlying mKEM is IND-CCA secure. The advantage of $\mathcal{A}^{AKE}$ is given as*

$$Adv_{\mathcal{A}^{AKE}} \leq n \cdot \frac{q_s^2}{|\mathcal{C}|} + n \cdot q_s \cdot (Adv_{\mathcal{A}^{CCA}} + n \cdot Adv_{\mathcal{A}^{PRF}})$$

where $n \leq N$ is the number of parties in the protocol, N is the number of public keys in the system, q_s is the number of sessions $\mathcal{A}^{AKE}$ is allowed to activate, $|\mathcal{C}|$ is the size of the ciphertext space, $\mathcal{A}^{CCA}$ is a polynomial adversary against the IND-CCA security of the underlying mKEM and $\mathcal{A}^{PRF}$ is a polynomial adversary adversary against the pseudo random function.

Sketch of Proof. We prove the theorem in a sequence of games. Let S_i be the event that $\mathcal{A}^{AKE}$ wins the AKE-security game in Game i.

Game 0. This is the original AKE-security game as per Definition 1. We have

$$Adv_{\mathcal{A}^{AKE}} = |2 \cdot \Pr[S_0] - 1| \tag{1}$$

Game 1. This game is the same as the previous one except that if two different sessions at user U_i output identical message C_i, then the game aborts. Let Repeat be such an event. As there are n users in the protocol, we have

$$|\Pr[S_1] - \Pr[S_0]| \leq n \cdot \Pr[\mathsf{Repeat}] \tag{2}$$

As the adversary is allowed to activate at most q_s number of sessions, we have

$$\Pr[\mathsf{Repeat}] \leq \frac{q_s^2}{|\mathcal{C}|} \tag{3}$$

Game 2. This is the same as the previous game except that a value $t \xleftarrow{R} [1, q_s]$ is chosen. If the Test query does not occur in the t-th session the game aborts and outputs a random value. Let E_2 be the event that the guess is correct.

$$\Pr[S_2] = \Pr[S_2|E_2]\Pr[E_2] + \Pr[S_2|\neg E_2]\Pr[\neg E_2] = \Pr[S_1]\frac{1}{q_s} + \frac{1}{2}\left(1 - \frac{1}{q_s}\right) \tag{4}$$

Game 3. This is the same as the previous game except that a value $i^* \xleftarrow{R} [1, n]$ is chosen. If a session at the user U_{i^*} is not asked the Test query the game aborts and outputs a random value. Let E_3 be the event that the guess is correct.

$$\Pr[S_3] = \Pr[S_3|E_3]\Pr[E_3] + \Pr[S_3|\neg E_3]\Pr[\neg E_3] = \Pr[S_2]\frac{1}{n} + \frac{1}{2}\left(1 - \frac{1}{n}\right) \tag{5}$$

Game 4. This is identical to the previous game except that, the test query asked is answered as follows: The symmetric generated by the user U_{i^*} in the test instance $\pi^t_{U_{i^*}}$ is replaced by a random key chosen uniformly from $\{0,1\}^k$. We claim that

$$|\Pr[S_4] - \Pr[S_3]| \leq Adv_{\mathcal{A}^{CCA}} \tag{6}$$

The queries asked by $\mathcal{A}^{AKE}$ can be simulated by $\mathcal{A}^{CCA}$ as follows: $\mathcal{A}^{CCA}$ generates the key pairs for the user U_{i^*} and obtains the public key corresponding to the users $\mathcal{U}' = \{U_1, \cdots, \mathcal{U}_n\} \setminus$

$\{U_{i^*}\}$ from its challenger. $\mathcal{A}^{CCA}$ returns the set $\mathcal{U}'$ to its challenger and obtains the challenge $(C^*, \mathcal{K}_0, \mathcal{K}_1)$ as described in Definition 4.

The Execute or Send query issued to $\pi^t_{U_{i^*}}$ is answered using the challenge encapsulation C^* as the outgoing message. The Execute or Send queries issued to the partner instances of $\pi^t_{U_{i^*}}$ are answered trivially as the per the protocol specification. Note that a Corrupt query on any user in $\mathcal{U}$ and RevealState or RevealKey on the test instance (or its partner instances) are not allowed. RevealKey queries on all other sessions are answered using the decapsulation oracle access as part of the IND-CCA game. When $\mathcal{A}^{AKE}$ asks the Test query, $\mathcal{A}^{CCA}$ selects a random bit θ and picks $\mathcal{K}_\theta$ from the keys $\{\mathcal{K}_0, \mathcal{K}_1\}$ given by the mKEM challenger. $\mathcal{A}^{CCA}$ now uses the key $\mathcal{K}_\theta$ as a seed to the pseudo-random function f and computes the challenge key $\kappa^* = f_{K_1}(\mathsf{sid}) \oplus \cdots \oplus f_{\mathcal{K}_\theta}(\mathsf{sid}) \oplus \cdots \oplus f_{K_n}(\mathsf{sid})$. Note that the symmetric keys K_i for $1 \leq i \leq n$, $i \neq i^*$ are generated by $\mathcal{A}^{CCA}$. The key κ^* is returned to $\mathcal{A}^{AKE}$.

Let θ' be the output of $\mathcal{A}^{AKE}$. If $\theta' = 1$ (guess for real key), θ is returned to the mKEM challenger. Otherwise $1 - \theta$ is returned. This game is essentially $\mathcal{A}^{AKE}$ playing the IND-CCA game against the mKEM with respect to the public keys $\{pk_1, \cdots, pk_n\} \setminus \{pk_{i^*}\}$. Hence, the IND-CCA game can be won at least whenever $\mathcal{A}^{AKE}$ succeeds in this game.

Game 5. This is identical to the previous game except that the output of each f_{K_i} for $1 \leq i \leq n$ is replaced by a random value chosen uniformly from $\{0,1\}^k$. We have,

$$|\Pr[S_5] - \Pr[S_4]| \leq n \cdot Adv_{\mathcal{A}^{PRF}} \tag{7}$$

By combining Equations 1 to 7, we have the desired advantage for $\mathcal{A}^{AKE}$.

□

3.2 Instantiating the protocol

Smart [24] presented an efficient IND-CCA secure mKEM based on ElGamal encryption scheme. However, it has been proven secure in the random oracle model. Although our generic construction does not assume random oracles, a concrete realization with this mKEM will only be secure in the random oracle model.

Smart also proposed a generic mKEM from any public key encryption scheme. This construction was proven IND-CCA secure assuming that the underlying encryption scheme was IND-CCA secure [24, Theorem 2]. Hence, generic mKEMs in the standard model can be constructed from public key encryption schemes which are also secure in the standard model [17, 16, 15]. This means that our protocol can be realized in the standard model by using the generic mKEM construction. However, note that the security in the standard model comes at the price of additional computational efficiency and longer message size. Nevertheless, this instantiation will result in the first concrete GKE protocol which has only one round of communication.

3.3 Achieving Forward Secrecy

Our one-round protocol in Figure 2 does not provide forward secrecy. However, it can be used as a building block for a two-round GKE protocol that achieves this additional goal. This protocol runs as follows: In the first round, each user U_i chooses an *ephemeral* asymmetric key pair (pk_i, sk_i) for mKEM and broadcasts pk_i to the group. In the second round users perform the one-round protocol in Figure 2 using asymmetric mKEM keys from the first round. It is easy to see that such construction involving one-time mKEMs results in an unauthenticated GKE protocol with forward secrecy. The AKE-security of this protocol can be achieved using digital signatures similar to [23]; in particular, one can treat one-time pk_i as a nonce of U_i and require the additional signature of U_i on $C_i|pk_1|\dots|pk_n$ in the second round.

4 Achieving MA-Security with KCIR

Bresson et al. [12] proposed a generic transformation that turns an AKE-secure GKE protocol π into a protocol π' that provides MA-security in the presence of an outsider adversary. Yet, their notion of MA-security did not consider KCIR. The transformation uses the well known technique of constructing an "authenticator" using the shared session key established in π. It works as follows: Let κ_i be the session key computed by U_i in protocol π. The protocol π' requires an additional round in which each party U_i computes a message $auth_i = \mathcal{H}(\kappa_i, i)$, where $\mathcal{H}$ is a hash function (modelled as random oracle in the proof) and broadcasts it to all other parties. Each party verifies the incoming messages using the session key established at their end. If the verification is successful, π' terminates with each party U_i accepting the session key $\kappa'_i = \mathcal{H}(\kappa_i, 0)$.

We show that the above transformation does not necessarily guarantee MA-security with outsider KCIR. For example, consider a protocol π which does not have forward secrecy like the protocol of Boyd and González Nieto [7] or our one-round protocol in Figure 2. Definition 2 implies that an adversary against MA-security with outsider KCIR can issue up to $n-1$ $\mathsf{Corrupt}$ queries but must then remain passive on behalf of corrupted users. As the protocol π does not have forward secrecy, corrupting a single party U_i is enough to obtain the session key κ_i. The adversary can now easily impersonate an uncorrupted party U_j in protocol π' by computing $auth_j = \mathcal{H}(\kappa_i, j)$. Hence, transformations based on shared keys cannot be used to obtain MA-security with outsider KCIR.

Instead, we show that the KS-compiler [22] when applied to our protocol achieves MA-security with both outsider and insider KCIR. The KS-compiler has been reviewed in Appendix A.2. Katz and Shin [22] showed that this compiler provides AKE-security and MA-security in the presence of insiders, yet without considering KCI attacks. Here, we show that this compilation technique is also sufficient to obtain MA-security with outsider and insider KCIR. It is easy to see that MA-security with insider KCIR implies MA-security with outsider KCIR, i.e. given an adversary against MA-security with outsider KCIR, one can construct an adversary against MA-security with insider KCIR. For this reason we only need to prove that the compiled protocol guarantees MA-security with insider KCIR.

Theorem 2. *If we apply the KS-compiler (in Figure 3) to our protocol in Figure 2, the resulting protocol provides MA-security with insider KCIR. The success probability of the adversary $\mathcal{A}^{MA}$ is given as*

$$n^2 \cdot Adv_{\mathcal{A}^{CMA}} + n \cdot \frac{q_s^2}{|\mathcal{C}|} + Adv_{\mathcal{A}^{\mathsf{coll}}}$$

where n is the number of parties in the protocol, q_s is the number of sessions $\mathcal{A}^{MA}$ is allowed to activate, $|\mathcal{C}|$ is the size of the ciphertext space, $\mathcal{A}^{CMA}$ is a polynomial adversary against the unforgeability of the signature scheme under chosen message attack and $\mathcal{A}^{\mathsf{coll}}$ is a polynomial adversary adversary against the collision resistance of the pseudo-random function F in the KS-compiler.

Sketch of Proof. The proof is given in a sequence of games. Let S_i be the event that $\mathcal{A}^{MA}$ wins the MA-security game in Game i.

Game 0. This is the original MA-security game as per Definition 1. We have

$$Succ_{\mathcal{A}^{MA}} = \Pr[S_0] \tag{8}$$

Game 1. This game is identical to the previous game except that the simulation fails when $\mathcal{A}^{MA}$ issues a Send query that contains a valid signature σ_i on the message $(U_i, \mathsf{sid}_i, \mathsf{pid}_i, \mathsf{ack}_i)$ such that the message has not been previously output by an oracle π_U^i and U_i has not been corrupted. Let Forge be such an event. We have

$$|\Pr[S_1] - \Pr[S_0]| \leq \Pr[\mathsf{Forge}] \tag{9}$$

If Forge occurs, we can use $\mathcal{A}^{MA}$ to forge a signature generated by the underlying signature scheme in the KS-compiler for a given public key in a chosen message attack as follows: The given public key is assigned to a party U_i, one of the n parties in the group. All other parties are initialized as normal according to the protocol. All queries to the parties can be easily answered by following the protocol specification since all secret keys are known, except for the signing key corresponding to the given public key of the forgery attack game. In the latter, the signing oracle that is available as part of the chosen message attack can be used to simulate the answers. Note that as part of the MA with insider KCIR game, $\mathcal{A}^{MA}$ is allowed to corrupt up to $n-1$ parties. Hence, the probability of $\mathcal{A}^{MA}$ not corrupting U_i is $\frac{1}{n}$. The probability of $\mathcal{A}^{MA}$ outputting a valid forgery on behalf of this user is also $\frac{1}{n}$. Hence $Adv_{\mathcal{A}^{CMA}} \geq \frac{1}{n^2} \cdot \Pr[\mathsf{Forge}]$ i.e., we have

$$\Pr[\mathsf{Forge}] \leq n^2 \cdot Adv_{\mathcal{A}^{CMA}} \tag{10}$$

Game 2. This game is the same as the previous one except that if two different sessions at user U_i output identical message C_i, then the game aborts. Let Repeat be such an event. As there are n users in the protocol, we have

$$|\Pr[S_2] - \Pr[S_1]| \leq n \cdot \Pr[\mathsf{Repeat}] \tag{11}$$

As the adversary is allowed to activate at most q_s number of sessions, we have

$$\Pr[\mathsf{Repeat}] \leq \frac{q_s^2}{|\mathcal{C}|} \tag{12}$$

Game 3. This is the same as the previous game except that the simulation fails if a collision occurs in F. Let $\mathsf{Collision}$ be the event.

$$|\Pr[S_3] - \Pr[S_2]| \leq \Pr[\mathsf{Collision}] \tag{13}$$

$\mathsf{Collision}$ occurs when two honest parties U_i and U_j compute keys sk_i and sk_j such that

$$\mathsf{ack}_i = F_{sk_i}(v_0) = F_{sk_j}(v_0) = \mathsf{ack}_i \text{ but, } sk_i' = F_{sk_i}(v_1) \neq F_{sk_j}(v_1) = sk_j'$$

Hence, we have

$$\Pr[\mathsf{Collision}] \le Adv_{\mathcal{A}^{\mathsf{coll}}} \tag{14}$$

If Game 3 does not fail, all the honest partnered parties compute the same session key. Hence, $\Pr[S_3] = 0$.

By combining Equations 8 to 14, we have the claimed advantage for $\mathcal{A}^{MA}$.

□

Remark 2. Note that the protocol obtained after applying Katz and Shin's compiler to our one-round GKE protocol still does not achieve forward secrecy. Hence, as discussed in Remark 1, it cannot achieve AKE-security with KCIR. However, from Theorem 2 it is evident that forward secrecy is not necessary for a GKE protocol to achieve MA-security with insider KCIR.

5 Conclusion

	Rounds	AKE	AKE-FS	AKE-KCIR	MA	MA-Out-KCIR	MA-In-KCIR	Model
Boyd and González Nieto [7]	1	Yes	No	No	No	No	No	ROM
Katz and Yung [23]	3	Yes	Yes	Yes*	honest	Yes*	No	Std.
Bohli et al. [2]	2	Yes	Yes	Yes	Yes	Yes	Yes	ROM
Bresson and Manulis [14]	3	Yes	Yes	Yes*	Yes	Yes	Yes*	Std.
Furukawa et al. [18]	2	Yes	Yes	Yes*	Yes	Yes	Yes*	Std.
Our Protocol	1	Yes	No	No	No	No	No	Std.
Our Protocol + KS-compiler	2	Yes	No	No	Yes	Yes	Yes	Std.

Table 1. Security and efficiency comparison among existing GKE protocols

Table 1 gives a comparison of the security of some of the existing GKE protocols. The column "Rounds" shows the number of communication rounds required to complete the protocol. The terms "AKE" refers to AKE-security, "AKE-FS" refers to AKE-security with forward secrecy and "AKE-KCIR" refers to AKE-security with KCI resilience. Similarly "MA" refers to mutual authentication and "MA-Out-KCIR" and "MA-In-KCIR" refers to mutual authentication with outsider and insider KCIR respectively. The entry "Yes*" says that the corresponding protocol appears to be secure under the notion but there is no formal proof. The last column in the table says whether the protocol is proven in the random oracle model or in the standard model.

It can be observed from the table that our protocol is the only one-round GKE protocol secure in the standard model. Although the protocol of Bohli et al. satisfies all the desired notions of security, it requires two-rounds of communication and moreover proven secure only in the random oracle model. Of the other protocols which are proven secure in the standard model, the protocols of Bresson and Manulis and Furukawa et al. [18] appear to satisfy all the desired notions, but they require three and two communication rounds respectively. Applying the KS-compiler to our protocol results in a two-round GKE protocol that satisfies the MA-security notion with insider KCIR. However, the resulting protocol still cannot provide forward secrecy or AKE-security with KCIR. The approach outlined in Section 3.3 with the combination of the KS-compiler results in a

GKE protocol that appears to satisfy all the desired notions of security. However, this protocol will have three rounds of communication.

Although our one-round GKE protocol cannot achieve all the security notions, it will be very useful in scenarios where communication efficiency highly desired. Unlike the previously known one-round GKE protocol, our protocol has been proven secure in the standard model. We have also discussed generic techniques with which the security of the protocol can be enhanced. However, as expected this additional security guarantee comes at the price of extra number of rounds. We leave it an open problem to construct an efficient mKEM in the standard model, which can in turn be used to construct efficient one-round GKE protocol using our approach.

References

1. Al-Riyami, S.S., Paterson, K.G.: Tripartite Authenticated Key Agreement Protocols from Pairings. In: Cryptography and Coding, 9th IMA International Conference. Volume 2898 of LNCS., Springer (2003) 332–359
2. Bohli, J.M., Gonzalez Vasco, M.I., Steinwandt, R.: Secure group key establishment revisited. Int. J. Inf. Sec. **6**(4) (2007) 243–254
3. Bohli, J.M., Steinwandt, R.: Deniable Group Key Agreement. In: Progress in Cryptology–VIETCRYPT'06, Revised Selected Papers. Volume 4341 of LNCS., Springer (2006) 298–311
4. Boyd, C.: Towards a classification of key agreement protocols. In: The Eighth IEEE Computer Security Foundations Workshop–CSFW'95, IEEE Computer Society (1995) 38–43
5. Boyd, C.: On Key Agreement and Conference Key Agreement. In: Information Security and Privacy–ACISP'97. Volume 1270 of LNCS., Springer (1997) 294–302
6. Boyd, C., Cliff, Y., González Nieto, J.M., Paterson, K.G.: One-Round Key Exchange in the Standard Model. International Journal of Applied Cryptography **1**(3) (2009) 181–199
7. Boyd, C., González Nieto, J.M.: Round-Optimal Contributory Conference Key Agreement. In: Public Key Cryptography–PKC'03. Volume 2567 of LNCS., Springer (2003) 161–174
8. Boyd, C., Mathuria, A.: Protocols for Authentication and Key Establishment. Information Security and Cryptography. Springer (August 2003)
9. Bresson, E., Chevassut, O., Essiari, A., Pointcheval, D.: Mutual Authentication and Group Key Agreement for Low-Power Mobile Devices. In: Proc. of MWCN 03, World Scientific Publishing (October 2003) 5962
10. Bresson, E., Chevassut, O., Pointcheval, D.: Provably Authenticated Group Diffie-Hellman Key Exchange - The Dynamic Case. In: Advances in Cryptology–ASIACRYPT'01. Volume 2248 of LNCS., Springer (2001) 290–309
11. Bresson, E., Chevassut, O., Pointcheval, D.: Dynamic Group Diffie-Hellman Key Exchange under Standard Assumptions. In: Advances in Cryptology–EUROCRYPT'02. Volume 2332 of LNCS., Springer (2002) 321–336
12. Bresson, E., Chevassut, O., Pointcheval, D., Quisquater, J.J.: Provably authenticated group Diffie-Hellman key exchange. In: CCS'01: Proceedings of the 8th ACM conference on Computer and Communications Security, ACM (2001) 255–264
13. Bresson, E., Manulis, M.: Contributory Group Key Exchange in the Presence of Malicious Participants. IET Information Security **2**(3) (2008) 85–93
14. Bresson, E., Manulis, M.: Securing Group Key Exchange against Strong Corruptions. In: Proceedings of ACM Symposium on Information, Computer and Communications Security (ASIACCS'08), ACM Press (2008) 249–260
15. Canetti, R., Halevi, S., Katz, J.: Chosen-Ciphertext Security from Identity-Based Encryption. In: Advances in Cryptology–EUROCRYPT'04. Volume 3027 of LNCS., Springer (2004)
16. Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. Technical report, http://shoup.net/ (2002)
17. Cramer, R., Shoup, V.: A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack. In: Advances in Cryptology–CRYPTO'98. Volume 1462 of LNCS., Springer (1998)
18. Furukawa, J., Armknecht, F., Kurosawa, K.: A Universally Composable Group Key Exchange Protocol with Minimum Communication Effort. In: Security and Cryptography for Networks–SCN 2008. Volume 5229 of LNCS., Springer (2008) 392–408
19. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. J. ACM **33**(4) (1986) 792–807
20. Gorantla, M.C., Boyd, C., González Nieto, J.M.: Modeling Key Compromise Impersonation Attacks on Group Key Exchange Protocols. In: Public Key Cryptography–PKC'09. Volume 5443 of LNCS., Springer (2009) 105–123

21. Gorantla, M.C., Boyd, C., Nieto, J.M.G.: On the connection between signcryption and one-pass key establishment. In Galbraith, S.D., ed.: IMA Int. Conf. Volume 4887 of LNCS., Springer (2007) 277–301
22. Katz, J., Shin, J.S.: Modeling insider attacks on group key-exchange protocols. In: Proceedings of the 12th ACM Conference on Computer and Communications Security–CCS'05, ACM (2005) 180–189
23. Katz, J., Yung, M.: Scalable Protocols for Authenticated Group Key Exchange. In: Advances in Cryptology–CRYPTO'03. Volume 2729 of LNCS., Springer (2003) 110–125
24. Smart, N.P.: Efficient Key Encapsulation to Multiple Parties. In: Security in Communication Networks–SCN'04. Volume 3352 of LNCS., Springer (2004) 208–219

A Preliminaries

We first review the definition and notion of security considered for mKEM and then briefly describe Katz and Shin's compiler.

A.1 Multi KEM

An mKEM takes the public keys of n parties as input and outputs a session key K and an encapsulation of K under all the n public keys. It is formally specified by three algorithms as described below:

- KeyGen: This is a probabilistic algorithm that takes the domain parameters as input and outputs a public-private key pair (pk, sk).
- Encap: This is a probabilistic algorithm that takes the domain parameters, the public keys of n receivers $(pk_1, \ldots, pk_n)$ and outputs a session key $K \in \{0,1\}^k$ and an encapsulation C of K under the public keys $(pk_1, \ldots, pk_n)$.
- Decap: This is a deterministic algorithm that takes the domain parameters, an encapsulation C and a private key sk_i as input and outputs either a key K or $\perp$.

For an mKEM to be considered valid it is required that for all key pairs (pk_i, sk_i), $i \in [1, n]$ if $(K, C) = \mathsf{Encap}(\{pk_1, pk_2, \ldots, pk_n\})$ then $\mathsf{Decap}(C, sk_i) = K$ for each $i \in [1, n]$.

The IND-CCA notion of security for mKEM is defined in a similar way to the traditional KEMs as below.

Definition 4. An mKEM is IND-CCA secure if the advantage of any probabilistic polynomial time adversary in the following game is negligible in the security parameter k.

Setup: The challenger runs the KeyGen algorithm and obtains n key pairs (pk_i, sk_i) for $1 \leq i \leq n$. All the public keys $\mathcal{P} = \{pk_1, \cdots, pk_n\}$ are given to the adversary.

Phase 1: The adversary is allowed to issue decapsulation queries as below:

Decap: The adversary issues this query with input $\mathcal{P}' \subseteq \mathcal{P}$ and an encapsulation C. The challenger returns either a key K or $\perp$ after executing the Decap algorithm on C using the private keys corresponding to $\mathcal{P}'$. Note that if Decap on input C produces different symmetric keys for two different private keys of users in $\mathcal{P}'$, then the encapsulation C is deemed invalid and the adversary is returned $\perp$.

Challenge: The adversary gives a set of keys $\mathcal{P}^* \subseteq \mathcal{P}$ to the challenger. The challenger first chooses $b \in \{0, 1\}$. It then runs the Encap algorithm using $\mathcal{P}^*$ and generates $(\mathcal{K}_b, C^*)$. It then sets $\mathcal{K}_{1-b}$ to be a random key drawn uniformly from the key space i.e., $\mathcal{K}_{1-b} \stackrel{R}{\leftarrow} \{0,1\}^k$. Both the keys $\{\mathcal{K}_0, \mathcal{K}_1\}$ are given to the adversary along with the challenge encapsulation C^*.

Phase 2: The adversary is allowed to issues queries to challenger as in Phase 1 with the following restriction: A decapsulation query on an encapsulation C' (includes $C' = C^*$) that trivially reveals the session key $\mathcal{K}_b$ is not allowed.[3]

Guess The goal of the adversary is to guess which one of the two keys $\{\mathcal{K}_0, \mathcal{K}_1\}$ is encapsulated in C^*. It finally outputs a guess bit b' and it succeeds if $b' = b$. The advantage of the adversary is given as $Adv_{\mathcal{A}^{CCA}} = |2 \cdot \Pr[b' = b] - 1|$.

A.2 Katz and Shin Compiler

Katz and Shin [22] proposed a compiler that turns any AKE secure GKE protocol into a universally composable GKE protocol that achieves mutual authentication in the presence of insiders. The compiler uses messages authenticated with signatures generated by long-term private keys of the parties. Let $\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ be a public key signature scheme which is existentially unforgeable against chosen message attack, where Gen is an algorithm that generates a signing key pair, Sign is a signing algorithm and Verify is a verification algorithm. The compiler also uses a pseudo random function family [19] F with *collision-resistance.* A formal definition of collision-resistant pseudo random function family is given below.

Definition 5 ([22])**.** Let $\mathcal{F} = \{F^k\}$ with $F^k = \{F_s\}_{s \in \{0,1\}^k}$ be a pseudo random function family (PRF). We say that $\mathcal{F}$ is collision-resistant PRF if there is an efficient procedure Sample such that the following is negligible in k for all polynomial time adversaries $\mathcal{A}$:

$$\Pr\left[v_0 \leftarrow \mathsf{Sample}(1^k);\ s, s' \leftarrow \mathcal{A}(1^k, v_0) \quad : \quad s, s' \in \{0,1\}^k \bigwedge s \neq s' \bigwedge F_s(v_0) = F_{s'}(v_0)\right]$$

Informally, the definition requires that for all k there exists an (efficiently computable) v_0 such that the function defined by $g(x) \stackrel{\text{def}}{=} F_x(v_0)$ is collision-resistant. Katz and Shin also sketch a way of constructing collision-resistant PRF in the standard model from any one-way permutation. Note that the above definition of collision-resistance for PRFs is different from the (standard) collision resistance considered for keyed hash functions.

In Figure 3, we present the KS-compiler. The users first compute a session key sk by executing an initial GKE protocol π. The compiler starts by using the session key as a seed to the PRF with two publicly known strings to compute an acknowledgment message ack and a session key sk'. The message ack is then signed with the user's signing key and the signature along with the user's identity is broadcast to all other users, which serves as key confirmation message. If all the incoming signatures verify correctly , the compiled protocol π' accepts sk' as the session key; otherwise, π' terminates without accepting.

[3] This restriction is necessary to address benign malleability [24].

Let F be a collision-resistant PRF, and assume that v_0 is output by $\mathsf{Sample}(1^k)$ and publicly-known. Let $v_1 \neq v_0$ also be publicly-known.

Initialization Phase: During the initialization phase of π', each player U_i runs $\mathsf{Gen}(1^k)$ to generate long-term verification/signing keys (PK_i, SK_i) (in addition to any keys needed for π).

The Protocol: Players run protocol π. If U_i would terminate without accepting in π, then it terminates without accepting in π'. Otherwise, if U_i would accept (in protocol π) with $(\mathsf{sid}_i, \mathsf{pid}_i, sk_i)$, this player performs the following additional steps:

1. U_i computes $\mathsf{ack}_i = F_{sk_i}(v_0)$ and $sk'_i = F_{sk_i}(v_1)$. Next, U_i erases all its local state except for ack_i, sk'_i, sid_i and pid_i. Then, U_i computes a signature $\sigma_i \leftarrow \mathsf{Sign}_{SK_i}(U_i, \mathsf{sid}_i, \mathsf{pid}_i, \mathsf{ack}_i)$ and sends the message (U_i, σ_i) to all players in pid_i.
2. Upon receipt of $|\mathsf{pid}_i - 1|$ messages (U_j, σ_j) from all other players $U_j \in \mathsf{pid}_i \setminus \{U_i\}$, player U_i checks that $\mathsf{Vrfy}_{PK_j}((U_j, \mathsf{sid}_i, \mathsf{pid}_i, \mathsf{ack}_i), \sigma_j) = 1$ for all $U_j \in \mathsf{pid}_i$. Assuming all verifications succeed, U_i accepts, erases its internal state, and outputs $(\mathsf{sid}_i, \mathsf{pid}_i, sk'_i)$. If any of the verifications do not succeed, U_i terminates without accepting (and with no output).

Fig. 3. Katz and Shin Compiler [22]