Universally Composable Contributory Group Key Exchange

M. Choudary Gorantla, Colin Boyd and Juan Manuel Gonzàlez Nieto

Information Security Institute, Queensland University of Technology
GPO Box 2434, Brisbane, QLD 4001, Australia.
mc.gorantla@isi.qut.edu.au, {c.boyd,j.gonzaleznieto}@qut.edu.au

Abstract. We treat the security of group key exchange (GKE) in the universal composability (UC) framework. Analyzing GKE protocols in the UC framework naturally addresses attacks by malicious insiders. We define an ideal functionality for GKE that captures contributiveness in addition to other desired security goals. We show that an efficient two-round protocol securely realizes the proposed functionality in the random oracle model. As a result, we obtain the most efficient UC-secure contributory GKE protocol known.

Keywords. Group Key Exchange, Contributiveness, Universal Composition

1 Introduction

A group key exchange (GKE) protocol allows a group of parties to agree upon a common session key over a public network. GKE protocols are useful in a variety of group applications like audio/video conferences, multicast/broadcast communication and other collaborative systems [6]. Although used in different applications a common requirement for the underlying GKE protocol is the assurance that the protocol has desired security properties.

Bresson et al. [9, 7, 8] initiate the formal treatment of security for GKE protocols based on earlier models for two-party key exchange [4, 3]. They formalize authenticated key exchange (AKE) security and mutual authentication as the desired notions of security. Informally, AKE-security ensures that the established session key is computationally indistinguishable from a random string, whereas mutual authentication guarantees that each party is assured of the participation of every other party in the protocol.

The above models assume the adversary to be an outsider who is not part of the GKE protocol execution. Katz and Shin [24] define insider security for GKE protocols by separating the requirements of mutual authentication into *agreement* on the session key and *security against insider impersonation attacks*. Bohli et al. [5] revisit this notion in the weak corruption model, where session state is not revealed. They also present insider attacks on the protocols of Katz and Yung [25] and Kim et al. [26] that violate integrity of those protocols. Later, Bresson and Manulis [10] unify the insider security notions of Katz and Shin into their definition of mutual authentication.

Another important security notion, considered by Bohli et al. [5] and Bresson and Manulis [10], is *contributiveness* in the presence of malicious insiders. A protocol satisfying this notion ensures that a proper subset of insiders cannot predetermine the session key. Note that if the resulting session key is allowed to be controlled by insiders, the session may be fixed to any value including the keys established in the past sessions. Hence, the protocol in this case cannot guarantee even the basic key freshness property. Lack of contributiveness may also allow insiders to establish "covert channels" by fixing the key to a value agreed with an outsider beforehand [29, 19]. For example, if the session key is to be used for the purpose of achieving confidentiality of future communication, this will allow an insider to leak the sensitive information without being detected.

Universal Composability. In the universal composability (UC) framework [12] a cryptographic task is specified through an ideal functionality. The UC formulation allows cryptographic protocols to preserve their security under arbitrary protocol composition. This also facilitates modular design of complex protocols. However, defining an appropriate ideal functionality for some cryptographic tasks has proven not to be easy [12].

Canetti and Krawczyk [15] show that their earlier *game-based* notion of SK-security [13] for two-party key exchange (2PKE) is equivalent to a relaxed notion of UC-security. This implies that a 2PKE protocol that satisfies the SK-security notion preserves its security under arbitrary protocol composition. On the other hand, Katz and Shin [24] define an ideal functionality for GKE in the UC framework and show that this notion of UC-security is strictly stronger than the game-based notions for GKE. There is no known equivalence between the game-based notions and UC notions of security for GKE and it is known that game-based notions guarantee security only when the protocols are run "stand-alone". Hence, we treat the security of GKE in the UC framework.

Katz and Shin also present a compiler (we call this the KS-compiler) that turns an AKE-secure protocol into a protocol that can realize their proposed ideal functionality. The KS-compiler follows the informal suggestion of Canetti and Krawczyk [15] to ensure that the compiled protocol has the so-called "ACK-property" [15, 24]. However, it introduces an extra round of communication for broadcasting a signature authenticated acknowledgment message. As the existing AKE-secure GKE protocols require at least two rounds of communication [25, 20], a KS-compiled protocol will have at least three communication rounds. In spite of introducing an additional round, the KS-compiler does not provide contributiveness in the presence of malicious insiders [27, Section 9.5]. On the other hand, Bohli et al. [5] present a two-round GKE protocol that satisfies contributiveness in the presence of insiders. This protocol cannot be obtained through the KS-compiler.

Furukawa et al. [21] present an ideal functionality for GKE without considering contributiveness. They also propose a two-round GKE protocol based on bilinear pairings and use non-interactive proofs to guarantee the ACK-property. This protocol is also shown to securely realize their functionality in the standard model. However, to establish a session key among a group of n parties, the protocol requires each party to perform $2n + 1$ pairing computations apart from other operations. These computations make this protocol very inefficient when compared to existing insider secure protocols [5, 10], which are proven secure under game-based notions. In this paper, we focus on defining an ideal functionality which guarantees contributiveness and at the same time can be realized by efficient GKE protocols.

Our Approach. Canetti and Krawczyk [15] note that a two-party key exchange protocol not having the ACK-property does not necessarily have any security weakness. They also introduce a tool called "non-information oracle" to relax a natural two-party key exchange functionality. It is shown that the relaxed functionality can be securely realized by protocols which do not have the ACK-property. We apply this approach to the case of GKE.

Contributions. We first propose a GKE functionality using non-information oracles. Unlike the formulation of Canetti and Krawczyk [15], our functionality runs multiple copies of the non-information oracle. Canetti and Krawczyk informally remark that this approach is more natural with each copy of the non-information oracle representing single session execution within a single participant. Another important advantage of this approach is that it naturally allows us to model an unreliable broadcast channel, where the parties do not necessarily receive the same values. We show that the proposed UC notion implies existing game-based security notions of AKE-security, mutual authentication and contributiveness.

In the later part of the paper, we modify the protocol of Bohli et al. [5] and show that it securely realizes the proposed functionality in the random oracle model. The modification to the protocol introduces a slight computational overhead for each party but not any communication rounds. Besides assuring strong security properties, this protocol is the most efficient GKE protocol proven secure in the UC framework.

Organization. We give an overview of the UC framework and the ideal functionality of Katz and Shin in Section 2. Section 3 presents an ideal functionality for GKE with contributiveness. A protocol that securely realizes the proposed functionality is given in Section 4 with a proof of security. Appendix A reviews existing game-based notions of security and also presents a revised notion of contributiveness. In Appendix B, we show that the security guaranteed by our functionality implies existing game-based security notions for GKE.

2 Universally Composable Group Key Exchange

We first give a brief overview of the UC framework, assuming basic familiarity. Please refer to Canetti [12] for more details. We also discuss the assumptions we make. A brief overview of Katz and Shin's ideal functionality for GKE is then provided.

In the UC framework, the security requirements of a task at hand are captured by an ideal functionality $\mathcal{F}$, which runs instructions specified by a trusted party. In an ideal protocol ϕ for a given $\mathcal{F}$, the ideal (dummy) parties send their input and obtain output from $\mathcal{F}$, which computes the output as per the instructions. An ideal adversary $\mathcal{S}$ (also called "simulator") interacts with $\mathcal{F}$ and can participate in the ideal protocol ϕ through corrupted parties. The security of ϕ is inherently guaranteed as $\mathcal{S}$ at the maximum can learn or possibly modify only the internal state of a corrupted party. The real-world execution of a protocol π involves parties running π among themselves and a real-world adversary $\mathcal{A}$, who is allowed to control some of the parties and the communication among all the parties.

A protocol π is said to securely realize $\mathcal{F}$ if running π amounts to "emulating" ϕ. The notion of emulation is defined by introducing an additional entity called environment $\mathcal{Z}$. $\mathcal{Z}$ generates inputs to all parties, observes their outputs and is allowed to interact with the adversary in an arbitrary way throughout the course of the computation. The protocol π emulates ϕ if for any adversary $\mathcal{A}$ there exists an adversary $\mathcal{S}$ such that, the probability of a probabilistic polynomial time (PPT) environment $\mathcal{Z}$, running on the security parameter k and any input, distinguishing its interaction with π and $\mathcal{A}$ from an interaction with ϕ and $\mathcal{S}$ is negligible[1] in k. Note that $\mathcal{S}$ has to interact with $\mathcal{Z}$ just as $\mathcal{A}$ does, particularly, $\mathcal{S}$ cannot "rewind" $\mathcal{Z}$.

Let ρ be a protocol that securely realizes an ideal functionality $\mathcal{F}$ and let π be a protocol executing in the $\mathcal{F}$-hybrid model where the parties in π make ideal calls to multiple instances of $\mathcal{F}$ in addition to interacting in the usual way. Let π^ρ be a protocol which starts with the protocol π and replaces the interaction with each instance of $\mathcal{F}$ with an interaction with a separate instance of ρ. The *universal composition theorem* of Canetti [12] states that running the composed protocol π^ρ has essentially the same effect as running the protocol π in the $\mathcal{F}$-hybrid model. Particularly, if π securely realizes some ideal functionality $\mathcal{G}$ in the $\mathcal{F}$-hybrid model then π^ρ securely realizes $\mathcal{G}$.

The security of cryptographic protocols analyzed in the UC framework is preserved under arbitrary protocol composition. Furthermore, the UC formulation allows these protocols to be designed

[1] An event is negligible in k if it happens with a probability that is less than the inverse of any polynomial in k

Functionality $\mathcal{F}_{\mathcal{GKE}}$

$\mathcal{F}_{\mathcal{GKE}}$ proceeds as follows, running on security parameter k, with parties $U_1, \ldots, U_n$, and an ideal adversary $\mathcal{S}$.

Initialization: Upon receiving a value (new-session, sid, pid) from party U_i for the first time (where pid is a non-empty set of distinct user identities), record (sid, pid, U_i) and send this to $\mathcal{S}$. In addition, if there are already |pid|-1 recorded tuples (sid, pid, U_j) for $U_j \in$ pid$\setminus U_i$ then store (sid, pid, ready) and send this to $\mathcal{S}$.

Key Generation: Upon receiving a message (sid, pid, ok) from $\mathcal{S}$ for a recorded tuple (sid, pid, ready), do:

- If all $U \in$ pid are uncorrupted, choose $\kappa \overset{R}{\leftarrow} \{0,1\}^k$ and store (sid, pid, κ).
- If any of $U \in$ pid is corrupted, wait for $\mathcal{S}$ to send a message (key, κ) and then store (sid, pid, κ).

Key Delivery: If $\mathcal{S}$ sends a message (deliver, U_i, sid, pid) for a recorded tuple (sid, pid, κ) and $U_i \in$pid, then send (sid, pid, κ) to party U_i.

Party Corruption: If $\mathcal{S}$ corrupts $U_i \in$ pid for a recorded tuple (sid, pid, κ) and message (sid, pid, κ) has not yet been sent to U_i, then $\mathcal{S}$ is given κ. Otherwise, $\mathcal{S}$ is given nothing.

Fig. 1. Functionality $\mathcal{F}_{\mathcal{GKE}}$ [24]

and analyzed in a modular way. It should be noted that the preserved security is as guaranteed by the corresponding ideal functionality.

Multiple Sessions. In the UC framework, it is sufficient to analyze the security of a single instance of a protocol i.e. it suffices to show that a single instance of a protocol ρ securely realizes some ideal functionality $\mathcal{F}$. The UC theorem can be used to show that multiple concurrent instances of ρ securely realize multiple concurrent instances of $\mathcal{F}$. However, this analysis is valid only if the instances of ρ have mutually disjoint state. The security of a multi-session extension of a protocol ρ whose instances share some joint state can be deduced by applying the *universal composition with joint state* theorem [17]. Hence, we analyze the security of only a single instance of a GKE protocol.

Party IDs and Session IDs. Similar to all existing work (both in non-UC and UC models) on GKE, we assume the pre-specified peer model [14], where each party is assumed to know the identities of the intended peers to the session when it commences the protocol. The partner ID (pid) of an instance at a party U is a set of identities of intended peers, including U itself. We assume that unique session IDs are provided by a higher level protocol when the GKE protocol is first initiated, which is in line with the UC formulation. However, as shown by Furukawa et al [21] one can combine the protocol initialization functionality of Barak et al. [2] with a GKE functionality. The combined functionality can be realized by a protocol in which the session ID is derived during the protocol execution.

Corruption Model. We assume that once a party is corrupted the adversary is given the entire current internal state and the adversary takes control of the party from then onwards. This corruption behavior of parties naturally models the so-called strong corruption model [24]. This corruption behavior does not model opening attacks [10], where only the ephemeral state of a session is revealed. However, we provide an alternate way of modeling opening attacks.

Katz-Shin's functionality for GKE [24]. Figure 1 outlines the ideal functionality $\mathcal{F}_{\mathcal{GKE}}$ of Katz and Shin and now we briefly explain $\mathcal{F}_{\mathcal{GKE}}$. In the **Initialization** phase the functionality waits to be notified by each party in pid. Once it receives such notifications from each of the parties in pid with matching sid and pid, $\mathcal{F}_{\mathcal{GKE}}$ enters a "ready" state and sends the adversary $\mathcal{S}$ a ready message. The **Key Generation** phase starts only after the functionality receives an ok message from $\mathcal{S}$. Intuitively, this ensures mutual authentication in the sense that each party in pid gets the common key only after it has notified that it wishes to exchange a key with the other parties in pid. $\mathcal{F}_{\mathcal{GKE}}$ chooses a random key if all the parties in pid are uncorrupted. However, $\mathcal{S}$ is allowed to choose the common group key if at least one of the parties in pid is corrupted. The schedule of key delivery to individual parties is determined by $\mathcal{S}$, particularly the key is delivered to a party immediately after $\mathcal{S}$ requests $\mathcal{F}_{\mathcal{GKE}}$ to do so. Finally, to model *forward secrecy*, the adversary is not given the session key on corruption of a party if the key has already been delivered to that party.

3 Universally Composable GKE with Contributiveness

The functionality $\mathcal{F}_{\mathcal{GKE}}$ allows the adversary to freely choose the common group key if at least one party in pid is corrupted. Clearly, this modeling lets an insider have complete control over the resulting key. Hence, $\mathcal{F}_{\mathcal{GKE}}$ guarantees insider security only with respect to impersonation and agreement but not with respect to contributiveness. Informally, a GKE protocol guarantees contributiveness in the presence of malicious participants if no proper subset of participating parties can influence the resulting common key to their advantage. Note that it is possible for an insider to mount denial of service attack by not following the protocol, but we do not deal with such attacks.

3.1 A GKE functionality that guarantees contributiveness

$\mathcal{F}_{\mathcal{GKE}}$ can be easily modified to arrive at a GKE functionality which assures of a random session key as long as there exist a single honest party. However such a functionality cannot be realized by any protocol in the strong corruption model where the entire internal state of a party is revealed upon corruption. To see why, a GKE protocol guaranteeing such strong contributiveness can be seen as a special type of asynchronous distributed coin-tossing protocol. Cleve [18] derive an upper bound of $(n-1)/2$ corrupted parties for a coin-tossing protocol among n parties to obtain an unbiased output. Hence, as argued by Desmedt et al. [19], if there is no honest majority of the parties in a GKE protocol the resulting session key can be biased by non-negligible amount. Thus the straightforward modification of $\mathcal{F}_{\mathcal{GKE}}$ assuming up to $(n-1)$ corrupted parties cannot be realized.

We now present an ideal functionality $\mathcal{F}^{+}_{\mathcal{GKE}}$ for GKE protocols using "non-information oracle" [15]. Informally, a non-information oracle has the property that its local output remains indistinguishable from a random string for any PPT adversary that it interacts with. A formal definition is given below:

Definition 1 (Non-information Oracle [15])**.** Let $\mathcal{N}$ be a PPT interactive Turing machine (ITM). Then $\mathcal{N}$ is a non-information oracle if no PPT ITM $\mathcal{M}$, having interacted with $\mathcal{N}$ on security parameter k, can distinguish with non-negligible probability between the local output of $\mathcal{N}$ and a value drawn uniformly from $\{0,1\}^k$.

The functionality $\mathcal{F}^{+}_{\mathcal{GKE}}$ is presented in Figure 2. $\mathcal{F}^{+}_{\mathcal{GKE}}$ invokes a new copy $\mathcal{N}_i$ of a non-information oracle $\mathcal{N}$ for each unique notification from the parties in pid and allows each copy to interact with the ideal adversary $\mathcal{S}$. Each $\mathcal{N}_i$ represents a single session execution of the group key exchange in an individual participating party. This is in contrast to the formulation used by Canetti and Krawczyk [15] for two-party key exchange, where a single non-information oracle captures both the sessions run by the partners of a session. Although complex, the current formulation naturally allows us to model an unreliable broadcast channel in the case of GKE as $\mathcal{S}$ is allowed to interact with each copy of $\mathcal{N}$ separately.

We now informally argue that the functionality $\mathcal{F}^{+}_{\mathcal{GKE}}$ intuitively captures contributiveness. The major difference between the functionalities $\mathcal{F}_{\mathcal{GKE}}$ and $\mathcal{F}^{+}_{\mathcal{GKE}}$ is in the key generation phase. Note

Functionality $\mathcal{F}_{\mathcal{GKE}}^{+}$

$\mathcal{F}_{\mathcal{GKE}}^{+}$ proceeds as follows, running on security parameter k, with parties $U_1, \ldots, U_n$, and an ideal adversary $\mathcal{S}$. $\mathcal{F}_{\mathcal{GKE}}^{+}$ is parameterized by non-information oracle $\mathcal{N}$.

Initialization: Upon receiving a value (sid, pid, new-session) from party $U_i \in$ pid for the first time (where pid is a set of at least two distinct party identities), record (sid, pid, U_i) and send this to $\mathcal{S}$. In addition do the following:

1. Invoke a copy $\mathcal{N}_i$ of $\mathcal{N}$ with fresh random input.
2. If there are already |pid|-1 recorded tuples (sid, pid, U_j) for $U_j \in$ pid then store (sid, pid, ready) and send this to $\mathcal{S}$.

Whenever $\mathcal{N}_i$ generates a message send this to $\mathcal{S}$ and whenever $\mathcal{S}$ sends a message to $\mathcal{N}_i$ forward this to $\mathcal{N}_i$.

Key Generation: Upon receiving a message (sid, pid, ok) from $\mathcal{S}$ for a recorded tuple (sid, pid, ready) do:

– If all the parties $U_i \in$ pid are uncorrupted: After all the corresponding copies $\mathcal{N}_i$ have generated local output, verify if these outputs are the same. Then choose $\kappa \xleftarrow{R} \{0,1\}^k$ and store (sid, pid, κ).

– If there exist at least one uncorrupted party $U_i \in$ pid: After all the corresponding copies $\mathcal{N}_i$ have generated their local output, verify if these outputs are the same. Then set κ to be one of these local outputs and store (sid, pid, κ).

Key Delivery: If $\mathcal{S}$ sends a message (deliver, U_i) when there is a recorded tuple (sid, pid, κ) and for $U_i \in$ pid then send (sid, pid, κ) to U_i immediately.

Party Corruption: If $\mathcal{S}$ corrupts $U_i \in$ pid for a recorded tuple (sid,pid,k) and message (sid,pid,k) has not yet been sent to U_i, then $\mathcal{S}$ is given the internal states of all the copies of $\mathcal{N}$ (including their local outputs if generated). Otherwise, $\mathcal{S}$ is given nothing.

Fig. 2. $\mathcal{F}_{\mathcal{GKE}}^{+}$: A GKE functionality that guarantees contributiveness

that $\mathcal{F}_{\mathcal{GKE}}$ allows $\mathcal{S}$ to choose the common group key κ when at least one party is corrupted. This allows even a single malicious insider to fix the resulting group key to a value of its choice. On the other hand $\mathcal{F}_{\mathcal{GKE}}^{+}$ sets the common key to be the output of the copies $\mathcal{N}_i$ that correspond to the uncorrupted parties, after verifying that these outputs are the same, when there is at least one uncorrupted party. As discussed earlier, in the presence of at least $(|\mathsf{pid}| - 1)/2$ corrupted parties the output distribution of $\mathcal{N}_i$ may be biased. But the output cannot be predetermined. When $\mathcal{S}$ corrupts at least one of the parties, $\mathcal{F}_{\mathcal{GKE}}^{+}$ gives the internal states of all the copies of $\mathcal{N}$. Hence none of the copies of $\mathcal{N}$ is a non-information oracle any longer. This reflects the fact that the common group key output by honest parties cannot be kept confidential from insiders. $\mathcal{F}_{\mathcal{GKE}}^{+}$ chooses the common key uniformly at random from $\{0,1\}^k$, when there is no corrupted party.

We assume arbitrarily malicious behavior for all the corrupted parties to model insider security. When considering such a behavior, the environment $\mathcal{Z}$ is not given read access to the corrupted parties' output tape [16]. Hence, the outputs of the parties are not relevant when all the parties in pid are corrupted.

3.2 Relation between relaxed UC-security and previous notions

We show that the security guaranteed by $\mathcal{F}_{\mathcal{GKE}}^{+}$ implies the existing notions of security reviewed in Appendix A. The claims and the corresponding proofs are in Appendix B.

Bresson and Manulis [10] define a session as *opened* if the adversary reveals ephemeral secrets of a session without revealing the long-term secret key. They observe that the simulation-based security models like universal composability do not handle opening attacks. The standard corruption model considered in the UC framework is the Byzantine corruption, where the adversary upon corrupting a party learns the entire internal state of that party and controls it from then onwards. To model

```
Round 1:
    **Computation** Each $U_i$ does the following:
        1. Chooses $k_i \xleftarrow{R} \{0,1\}^k$, $x_i \xleftarrow{R} \mathbb{Z}_q$ and computes $y_i = g^{x_i}$ and $H(k_i)$
        2. Sets $M_i^I = H(k_i) \| y_i$ and computes a signature $\sigma_i^I$ on $M_i^I \| \mathsf{pid}$.
    **Broadcast** Each $U_i$ broadcasts $M_i^I \| \sigma_i^I$.
    **Check** Each $U_i$ checks all signatures $\sigma_j^I$ of incoming messages $M_j^I \| \sigma_j^I$.
Round 2:
    **Computation** Each $U_i$ does the following:
        1. Computes $t_i^L = H(y_{i-1}^{x_i})$, $t_i^R = H(y_{i+1}^{x_i})$, $T_i = t_i^L \oplus t_i^R$, $\mathsf{auth}_i = H(\mathsf{pid} \| H(k_1) \ldots \| H(k_n))$, $\mathsf{mask}_i = k_i \oplus t_i^R$.
        2. Sets $M_i^{II} = \mathsf{mask}_i \| T_i \| \mathsf{auth_i}$ and computes a signature $\sigma_i^{II}$ on $M_i^{II}$.
    **Broadcast** Each $U_i$ broadcasts $M_i^{II} \| \sigma_i^{II}$.
    **Check** Each $U_i$ does the following for each $j = 1, \ldots, n$, $j \neq i$. $U_i$ aborts in case any of the checks fail.
        1. Verify the signature $\sigma_j^{II}$ on the message $M_j^{II}$
        2. Check $T_1 \oplus \cdots \oplus T_n \overset{?}{=} 0$ and $\mathsf{auth}_i \overset{?}{=} \mathsf{auth}_j$
        3. Parse each incoming message $M_j^{II}$ as $\mathsf{mask}_j \| T_j \| \mathsf{auth}_j$ and extract $k_j = \mathsf{mask}_j \oplus T_{j+1} \oplus \cdots \oplus T_{i-1} \oplus t_i^L$
        4. Checks the commitments $H(k_j)$ sent in Round 1 for each $k_j$ extracted in Round 2.
    **Key Computation** Each $U_i$ computes the session key $sk_i = H(\mathsf{pid} \| k_1 \| \ldots \| k_n)$
```

Fig. 3. A protocol that realizes $\mathcal{F}_{\mathcal{GKE}}^{+}$

opening attacks, one may consider honest-but-curious behavior for parties upon corruption, in addition to the Byzantine corruption behavior. Such a formulation requires modifications to $\mathcal{F}_{\mathcal{GKE}}^{+}$ as the environment should now be allowed to access the output tapes of parties who have been issued only an honest-but-curious corrupt query.

Although we do not explicitly consider opening attacks, note that the functionality $\mathcal{F}_{\mathcal{GKE}}^{+}$ does allows $\mathcal{S}$ to obtain internal states of all the copies of $\mathcal{N}$ when at least one party in pid is corrupted. Hence, a protocol that securely realizes $\mathcal{F}_{\mathcal{GKE}}^{+}$ guarantees mutual authentication and contributiveness in the presence of at most $(|\mathsf{pid}| - 2)$ and $(|\mathsf{pid}| - 1)$ insiders respectively, while the internal states of all the parties are revealed!

4 A protocol that realizes $\mathcal{F}_{\mathcal{GKE}}^{+}$

In Figure 3, we present a protocol that securely realizes $\mathcal{F}_{\mathcal{GKE}}^{+}$. The protocol is a slightly modified version of Bohli et al.'s protocol [5], which itself is inspired from earlier protocols [26, 11].

Let $\{U_1, \ldots, U_n\}$ be the set of parties who wish to establish a common group key. We assume that the parties are ordered in a logical ring with U_{i-1} and U_{i+1} being the left and right neighbors of U_i for $1 \leq i \leq n$, $U_0 = U_n$ and $U_{n+1} = U_1$. We also assume the pre-specified peer model.

During the initialization phase, a cyclic group $\mathbb{G}$ of prime order q, an arbitrary generator g of $\mathbb{G}$ and the description of a hash function H that maps to $\{0,1\}^k$ are chosen. We assume that each party has a pair of long-term private and public key pair for a public key signature scheme. Figure 3 outlines the execution of the protocol after the initialization phase. Unlike the protocol of Bohli et al. [5], the protocol in Figure 3 avoids rushing attack by broadcasting the commitment to their contribution $H(k_i)$ for all the parties, in Round 1. This modification is suggested by Bohli et.al themselves based on the technique of Mitchell et al. [28]. Note that this does not introduce any extra rounds. But the computational cost and message size are slightly increased as each party has to compute and broadcast $H(k_i)$ in the first round.

```
1. Choose $x_i \xleftarrow{R} \mathbb{Z}_q$, $k_i \xleftarrow{R} \{0,1\}^k$ and compute $y_i = g^{x_i}$
2. Compute $M_{i_1} = H(k_i) \| y_i$ and send it to $\mathcal{M}$
3. On receiving a set of $(|\mathsf{pid}| - 1)$ messages $\{M_{j_1} | j \neq i, j \in [1, |\mathsf{pid}|]\}$, parse each $M_{j_1}$ as $H(k_j) \| y_j$
4. Compute $t_i^L = H(y_{i-1}^{x_i}), t_i^R = H(y_{i+1}^{x_i}), T_i = t_i^L \oplus t_i^R$ (with $y_0 = y_{|\mathsf{pid}|}$ and $y_{|\mathsf{pid}|+1} = y_1$)
5. Compute $M_{i_2} = k_i \oplus t_i^R \| T_i$ and send it to $\mathcal{M}$
6. On receiving a set of $(|\mathsf{pid}| - 1)$ messages $\{M_{j_2} | j \neq i, j \in [1, |\mathsf{pid}|]\}$, parse each $M_{j_2}$ as $\mathsf{mask_j} \| T_j$
7. If $T_1 \oplus \cdots \oplus T_{|\mathsf{pid}|} \neq 0$, choose $sk_i \xleftarrow{R} \{0,1\}^k$, locally output $sk_i$ and halt. Otherwise continue to next step.
8. For each $j = 1, \ldots, |\mathsf{pid}|, j \neq i$, extract $k_j = \mathsf{mask}_j \oplus T_{j+1} \oplus \cdots \oplus T_{i-1} \oplus t_i^L$ from the messages parsed in Step 6
9. Compute hash values for each extracted $k_j$ and check to see if the output matches with the corresponding
   hash value parsed in Step 3. If there is at least one mismatch choose $sk_i \xleftarrow{R} \{0,1\}^k$, locally output $sk_i$ and
   halt. Otherwise continue to next step.
10. Compute $sk_i = H(\mathsf{pid} \| k_1 \| \ldots \| k_{|\mathsf{pid}|})$ and locally output.
```

Fig. 4. A non-information oracle $\mathcal{N}_{gke}$

4.1 Security Analysis

In order to prove that the protocol in Figure 3 realizes $\mathcal{F}^{+}_{\mathcal{GKE}}$ we first have to show that there exists a non-information oracle for the exchanged keys. We now present a construction of a non-information oracle $\mathcal{N}_{gke}$, which will be used in the security proof of the protocol.

Let $\mathcal{M}$ be the ITM that is interacting with $\mathcal{N}_{gke}$. When activated, $\mathcal{N}_{gke}$ expects a message that contains the description of a group $\mathbb{G}$ of prime order q and generator g, the description of a hash function H and pid. An i-th copy of $\mathcal{N}_{gke}$ proceeds as outlined in Figure 4. We now show that our construction of $\mathcal{N}_{gke}$ is indeed a non-information oracle.

Claim 1. *Let* $\mathsf{SuccNIO}_{\mathcal{M}}$ *be the success probability of* $\mathcal{M}$ *in distinguishing the output of* $\mathcal{N}_{gke}$ *from random. Then* $\mathsf{SuccNIO}_{\mathcal{M}} \leq \frac{q_{ro}+|\mathsf{pid}|+3}{2^k} + \frac{q_{ro} \cdot \mathsf{Succ}_{CDH}}{|\mathsf{pid}|} + \frac{q_{ro}}{2^k}$*, where* q_{ro} *is the polynomial bound for the number of queries to the random oracle* H*,* $|\mathsf{pid}|$ *is the number of copies of* $\mathcal{N}_{gke}$ *and* Succ_{CDH} *is the success probability of solving the CDH problem in the group* G*.*

Proof. (Sketch) We prove the claim in a sequence of games. Let S_i be the event that $\mathcal{M}$ distinguishes the output of $\mathcal{N}_{gke}$ from random in Game$_i$.

Game 0: This is the initial game in which $\mathcal{M}$ interacts with $\mathcal{N}_{gke}$ as per the definition 1. We have

$$\Pr[S_0] = \mathsf{SuccNIO}_{\mathcal{M}} \tag{1}$$

Game 1: This is the same as the previous game except that the game aborts if an event $\mathsf{Collision}$ occurs, where $\mathsf{Collision}$ is the event that a collision occurs in the random oracle. We have

$$|\Pr[S_1] - \Pr[S_0]| \leq \Pr[\mathsf{Collision}] \tag{2}$$

Note that $\mathcal{N}_{gke}$ makes $|\mathsf{pid}| + 3$ queries to H in its execution. Hence, the number of calls to the random oracle is bounded by $q_{ro} + |\mathsf{pid}| + 3$ and the probability that $\mathsf{Collision}$ occurs is

$$\Pr[\mathsf{Collision}] \leq \frac{q_{ro} + |\mathsf{pid}| + 3}{2^k} \tag{3}$$

Game 2: This is the same as the previous game except that the game aborts if an event Ask occurs, where Ask is an event that $\mathcal{M}$ queries a pair-wise CDH component (y_{l-1}^{xl} or y_{l+1}^{xl}) to H. We have

$$|\Pr[S_2] - \Pr[S_1]| \leq \Pr[\mathsf{Ask}] \tag{4}$$

Note that there exist exactly $|\mathsf{pid}|$ unique pair-wise CDH instances that can be formed from the messages sent and received by $\mathcal{N}_{gke}$. If the event Ask happens, we can use $\mathcal{M}$ to solve the CDH problem by randomly picking one of the entries of the H asked by $\mathcal{M}$. The success probability of solving CDH Succ_{CDH} is given as $\mathsf{Succ}_{CDH} = \frac{\Pr[\mathsf{Ask}]\cdot|\mathsf{pid}|}{q_{ro}}$. Rewriting the equation we have

$$\Pr[\mathsf{Ask}] = \frac{q_{ro} \cdot \mathsf{Succ}_{CDH}}{|\mathsf{pid}|} \tag{5}$$

Game 3: This game is the same as the previous game except it aborts if $\mathcal{M}$ queries H with the key material $\mathsf{pid}\|k_1\|\ldots\|k_{|\mathsf{pid}|}$ as input. This at the maximum happens with a negligible probability $\frac{q_{ro}}{2^k}$. Hence we have,

$$|\Pr[S_3] - \Pr[S_2]| \leq \frac{q_{ro}}{2^k} \tag{6}$$

By substituting $\Pr[S_3] = 0$ and from Equations 1-6 we have

$$\mathsf{SuccNIO}_{\mathcal{M}} \leq \frac{q_{ro} + |\mathsf{pid}| + 3}{2^k} + \frac{q_{ro}\cdot\mathsf{Succ}_{CDH}}{|\mathsf{pid}|} + \frac{q_{ro}}{2^k}$$

which is negligible in k. Hence, $\mathcal{N}_{gke}$ is a non-information oracle when H is modeled as a random oracle and if CDH is computationally hard in $\mathbb{G}$. □

Having shown that there exists a non-information oracle $\mathcal{N}_{gke}$ for the protocol in Figure 3, we now claim that the protocol securely realizes the functionality $\mathcal{F}^{+}_{\mathcal{GKE}}$.

Claim 2. *The protocol π_{gke} in Figure 3 securely realizes $\mathcal{F}^{+}_{\mathcal{GKE}}$ for the non-information oracle $\mathcal{N}_{gke}$ assuming that the signature used in the protocol is existentially unforgeable against chosen message attacks (EU-CMA).*

Proof. (Sketch) Let $\mathcal{A}$ be a real adversary interacting with π_{gke} and real-world parties. We construct an ideal adversary $\mathcal{S}$ such that no environment $\mathcal{Z}$ can tell whether it is interacting with $\mathcal{A}$ and parties running π_{gke} in the real world or with $\mathcal{S}$ and dummy parties communicating with $\mathcal{F}^{+}_{\mathcal{GKE}}$.

The general proof idea is similar to the approach of the two-party cases [15, 23], but as stated earlier we make use of multiple copies of $\mathcal{N}_{gke}$. These copies of $\mathcal{N}_{gke}$ provide $\mathcal{S}$, transcripts of the protocol for its simulation. If all the parties in pid remain uncorrupted, $\mathcal{S}$ uses these transcripts to simulate messages among the parties. If only a proper subset of parties in pid is corrupted $\mathcal{S}$ obtains the internal state of the copies of $\mathcal{N}_{gke}$ through $\mathcal{F}^{+}_{\mathcal{GKE}}$ that is consistent with the transcripts. Moreover, $\mathcal{S}$ can now have the chance to perform the group key exchange on behalf of the corrupted parties with the copies of $\mathcal{N}_{gke}$ that correspond to the uncorrupted parties. In both the cases, during its interaction with $\mathcal{Z}$, $\mathcal{S}$ supplies transcripts on demand to $\mathcal{Z}$, which are consistent with the final output (that contains the common key) of the uncorrupted parties in the ideal model as observed by $\mathcal{Z}$. $\mathcal{S}$ proceeds as follows:

1. $\mathcal{S}$ internally keeps a simulation of the parties $U_1^{(s)}, \ldots, U_n^{(s)}$ running the protocol π_{gke}. Any message from $\mathcal{Z}$ is forwarded to $\mathcal{A}$ (as if coming from $\mathcal{A}$'s environment) and any message from $\mathcal{A}$ is forwarded to $\mathcal{Z}$($\mathcal{S}$'s environment).

2. $\mathcal{S}$ generates public-private key pairs for all the simulated parties $U_1^{(s)}, \ldots, U_n^{(s)}$ and gives the resulting public keys to $\mathcal{A}$. It also chooses the common parameters: the description of a group $\mathbb{G}$ of prime order q and a generator g of $\mathbb{G}$.
3. Upon receiving a message $(\mathsf{sid}, \mathsf{pid}, U_i)$ from $\mathcal{F}_{\mathcal{GKE}}^{+}$ for an uncorrupted party U_i, $\mathcal{S}$ initiates the simulation of π_{gke} for $\mathcal{A}$, being run by $U_i^{(s)}$ with $\mathsf{sid}^{(s)} = \mathsf{sid}$ and $\mathsf{pid}^{(s)} = \mathsf{pid}$. It sends the common parameters and $\mathsf{pid}^{(s)}$ to the i-th copy of $\mathcal{N}_{gke}$ and obtains a message M_{i_1}.
4. For each $i \in \{1, \ldots, |\mathsf{pid}^{(s)}|\}$ do:
   When a simulated party $U_i^{(s)}$ wants to broadcast a message $(M_{i_1}^{(s)}, \sigma_{i_1}^{(s)})$ and if the corresponding ideal party U_i is uncorrupted, $\mathcal{S}$ first checks if $\sigma_{i_1}^{(s)}$ is a valid signature on $M_{i_1}^{(s)}$ and aborts the simulation if it is not a valid signature. Otherwise, $\mathcal{S}$ generates a signature σ_{i_1} on $M_{i_1} \| \mathsf{pid}^{(s)}$ using the private key of $U_i^{(s)}$, where M_{i_1} is the message obtained from the i-th copy of $\mathcal{N}_{gke}$ and simulates the message $(M_{i_1} \| \mathsf{pid}^{(s)}, \sigma_{i_1})$ for $\mathcal{A}$.
5. For each $i \in \{1, \ldots, |\mathsf{pid}^{(s)}|\}$ do:
   When a simulated party $U_i^{(s)}$ is delivered a set of $|\mathsf{pid}^{(s)}| - 1$ messages $\{(\mathsf{sid}^{(s)}, M_{j_1}^{(s)}, \sigma_{j_1}^{(s)}) | j \neq i, 1 \leq j \leq |\mathsf{pid}^{(s)}|\}$ and if the corresponding ideal party U_i is uncorrupted, $\mathcal{S}$ checks if each $\sigma_{j_1}^{(s)}$ is a valid signature on the corresponding $M_{j_1}^{(s)}$. $\mathcal{S}$ aborts if there is at least one invalid signature and proceeds as follows otherwise: $\mathcal{S}$ parses each $M_{j_1}^{(s)}$ that $U_i^{(s)}$ received as $M_{j_1} \| \mathsf{pid}^{(\mathsf{s})}$ and sends each M_{j_1} to the i-th copy of $\mathcal{N}_{gke}$ and waits till it obtains a message M_{i_2}. It now generates a signature σ_{i_2} on the message $M_{i_2} \| \mathsf{auth}_i$ using the private key of $U_i^{(s)}$ and simulates a message $(M_{i_2} \| \mathsf{auth}_i, \sigma_{i_2})$ for $\mathcal{A}$, where $\mathsf{auth}_i = H(\mathsf{pid}^{(s)} \| H(k_1) \| \ldots \| H(k_{|\mathsf{pid}^{(s)}|}))$. The $H(k_j)$'s used in computing auth_i are parsed from the messages received by $U_i^{(s)}$.
6. For each $i \in \{1, \ldots, |\mathsf{pid}^{(s)}|\}$ do:
   When a simulated party $U_i^{(s)}$ is delivered a set of $|\mathsf{pid}^{(s)}| - 1$ messages $\{(\mathsf{sid}^{(s)}, M_{j_2}^{(s)}, \sigma_{j_2}^{(s)}) | j \neq i, 1 \leq j \leq |\mathsf{pid}^{(s)}|\}$ and if the corresponding ideal party U_i is uncorrupted, $\mathcal{S}$ checks if each $\sigma_{j_2}^{(s)}$ is a valid signature on the corresponding $M_{j_2}^{(s)}$. It aborts if there is at least one invalid signature and proceeds as follows otherwise: $\mathcal{S}$ parses each $M_{j_2}^{(s)}$ that $U_i^{(s)}$ received as $M_{j_2} \| \mathsf{auth}_i$ and sends each M_{j_2} to the i-th copy of $\mathcal{N}_{gke}$.
7. When $\mathcal{A}$ corrupts a party $U_i^{(s)}$, $\mathcal{S}$ proceeds as follows:
   (a) If $\mathcal{S}$ has not yet received a message $(\mathsf{sid}, \mathsf{pid}, U_i)$ from $\mathcal{F}_{\mathcal{GKE}}^{+}$, $\mathcal{A}$ is given the long term private key of $U_i^{(s)}$.
   (b) If $U_i^{(s)}$ already chose the internal state but did not yet erase it, $\mathcal{A}$ is given the internal state of $U_i^{(s)}$ along with its private key as follows:
      i. $U_i^{(s)}$ has not yet sent a message $(M_{i_1}^{(s)}, \sigma_{i_1}^{(s)})$ (first round message), $\mathcal{S}$ gives the current internal state of $U_i^{(s)}$
      ii. $U_i^{(s)}$ has already sent the first round message $(M_{i_1}^{(s)}, \sigma_{i_1}^{(s)})$, $\mathcal{S}$ obtains the internal state of the i-th copy of $\mathcal{N}_{gke}$ by corrupting the party U_i in the ideal world and replaces the current internal state of $U_i^{(s)}$ with this state. $\mathcal{A}$ is then given the internal state of $U_i^{(s)}$.
   (c) If $U_i^{(s)}$ already erased its internal state, $\mathcal{S}$ has to give the common key to $\mathcal{A}$.
      i. If no uncorrupted party in the simulation generated an output, $\mathcal{S}$ obtains the local output of $\mathcal{N}_i$ by corrupting U_i in the ideal model. The key is handed over to $\mathcal{A}$.

ii. If some uncorrupted has generated an output, $\mathcal{S}$ corrupts U_i, asks $\mathcal{F}^{+}_{\mathcal{GKE}}$ to deliver the session key to U_i and gives the key to $\mathcal{A}$.

(d) If $\mathcal{S}$ has already sent a $(\mathsf{deliver},\ U_i)$ message to $\mathcal{F}^{+}_{\mathcal{GKE}}$, $\mathcal{A}$ gets no internal state.

8. If a simulated party $U_i^{(s)}$, whose ideal counterpart U_i is uncorrupted, produces an output $sk_i^{(s)}$, $\mathcal{S}$ proceeds as follows:
   (a) If no parties in pid are corrupted:
      i. If $\mathcal{S}$ has not yet sent (ok) message to $\mathcal{F}^{+}_{\mathcal{GKE}}$, then $\mathcal{S}$ checks if it has received $(\mathsf{sid},\mathsf{pid},\mathsf{ready})$ from $\mathcal{F}^{+}_{\mathcal{GKE}}$. If not $\mathcal{S}$ aborts. Otherwise, it sends (ok) and then $(\mathsf{deliver},\ U_i)$ to $\mathcal{F}^{+}_{\mathcal{GKE}}$.
      ii. If $\mathcal{S}$ has already sent an (ok) message to $\mathcal{F}^{+}_{\mathcal{GKE}}$, $\mathcal{S}$ sends $(\mathsf{deliver},\ U_i)$ to $\mathcal{F}^{+}_{\mathcal{GKE}}$.
   (b) Let $\mathcal{C} \subseteq \mathsf{pid} \setminus \{U_i\}$ be the set of corrupted parties.
      i. If $\mathcal{S}$ has not yet sent (ok) to $\mathcal{F}^{+}_{\mathcal{GKE}}$, then $\mathcal{S}$ first sends $(\mathsf{sid},\mathsf{pid},\mathsf{new\text{-}session})$ to $\mathcal{F}^{+}_{\mathcal{GKE}}$ on behalf of the parties in $\mathcal{C}$ who have not done so. If $\mathcal{S}$ does not receive $(\mathsf{sid},\mathsf{pid},\mathsf{ready})$ after doing so, it aborts. Otherwise, it sends (ok) back to $\mathcal{F}^{+}_{\mathcal{GKE}}$, followed by $(\mathsf{deliver},\ U_i)$.
      ii. $\mathcal{S}$ aborts if two simulated parties corresponding to uncorrupted ideal parties output different keys.
      iii. If $\mathcal{S}$ already sent (ok) message to $\mathcal{F}^{+}_{\mathcal{GKE}}$, $\mathcal{S}$ now sends $(\mathsf{deliver},\ U_i)$.

With the above description of $\mathcal{S}$, we now argue that no PPT environment $\mathcal{Z}$ can distinguish its interaction with $\mathcal{S}$ in the ideal world from that with $\mathcal{A}$ in the real world. We explain only the case where $\mathcal{S}$ aborts in its simulation.

– In Step 4, $\mathcal{S}$ simulates Round 1 messages on behalf of the uncorrupted parties using the messages obtained from the respective copies of $\mathcal{N}_{gke}$. Note that $\mathcal{S}$ aborts the simulation in the case of a simulated party corresponding to an uncorrupted ideal party tries to send a message containing invalid signature. However, this probability is negligible as an uncorrupted simulated party follows the protocol and generates a valid signature using the specified signature scheme.
– In Step 5, $\mathcal{S}$ validates the received Round 1 messages and simulates Round 2 messages on behalf of the uncorrupted parties. It aborts simulation if any message received by a simulated party corresponding to an uncorrupted party has invalid signature. However, in this case the real world execution of the protocol would also abort. The adversary $\mathcal{A}$ has to forge the signature in case it wants $\mathcal{S}$ to accept a signature on a modified message. However, as the signature used is EU-CMA secure this probability negligible. The Round 2 messages simulated by $\mathcal{S}$ on behalf of the uncorrupted parties in this step do not introduce any difference. The simulation in Step 6 is also valid based on the above arguments.
– Corruptions of parties at different stages of the protocol execution are handled in Step 7, without introducing any difference from the point of view of $\mathcal{Z}$. Note, in particular, that the internal state provided in Step 7(b)ii is consistent with the earlier simulated messages.
– The probability of $\mathcal{S}$ aborting in Step 8(a)i is negligible. $\mathcal{S}$ starts simulating the protocol π_{gke} for a simulated party $U_i^{(s)}$ (corresponding to an uncorrupted U_i) only after receiving the message $(\mathsf{sid},\ \mathsf{pid},\ U_i)$ from $\mathcal{F}^{+}_{\mathcal{GKE}}$ as described in Step 3. $\mathcal{F}^{+}_{\mathcal{GKE}}$ sends the $(\mathsf{sid},\ \mathsf{pid},\ \mathsf{ready})$ message once it receives the $(\mathsf{sid},\ \mathsf{pid},\ \mathsf{new\text{-}session})$ message from all $U_i \in \mathsf{pid}$. As all the parties in pid are uncorrupted, a party $U_i^{(s)}$ outputting a key in the simulation before $\mathcal{S}$ receiving the $(\mathsf{sid},\ \mathsf{pid},\ \mathsf{ready})$ message is negligible.
– In Steps 8(a)i,8(a)ii when all the parties in pid are uncorrupted, the common key output (chosen uniformly at random from $\{0,1\}^k$) by these parties remain indistinguishable from a random

string. When a proper subset of parties in pid is corrupted, as explained earlier, the output key of the uncorrupted parties is consistent with the earlier simulated messages and this key is an output of one of the copies of $\mathcal{N}_{gke}$.

- $\mathcal{S}$ aborts in Step 8(b)ii if two simulated parties corresponding to two uncorrupted parties in the ideal world output two different keys, which happens with negligible probability as the protocol is correct.

If a GKE protocol employs signatures, it is generally assumed that a strong corrupt query or a session state reveal query do not reveal the randomness used in generating the signatures, as this may potentially leak the long-term secret key itself [8, 10]. This is easily simulated by $\mathcal{S}$ by revealing internal state of the copies of $\mathcal{N}_{gke}$ on an appropriate query from the environment. Note that the copies of $\mathcal{N}_{gke}$ do not compute any signatures. Hence, the environment cannot distinguish its interaction with the protocol in the real world and a real adversary from an interaction with $\mathcal{S}$ and $\mathcal{F}^{+}_{\mathcal{GKE}}$ running the copies of $\mathcal{N}_{gke}$ in the ideal world.

This completes our sketch of proof that the simulation by $\mathcal{S}$ is valid. □

4.2 Discussion

Desmedt et al. [19] propose a notion of shielded insider privacy, which guarantees that the session key distribution is not biased. As explained in Section 3.1, this level of contributiveness can be achieved only assuming weak corruptions and honest majority of participants. On the other hand, if there exists no honest majority of participants, our formulation allows the session key distribution to be biased. As explained by Bresson and Manulis [10], this scenario cannot be addressed when assuming strong corruptions and corrupted parties up to $(|\mathsf{pid}|-1)$. Hence there is trade-off between the capability of the adversary and the attack it can mount.

In independent work, Furukawa et al. [21] present an ideal functionality for GKE protocols without assuming the availability of unique session IDs. However, they do not consider contributiveness. They also propose a two-round protocol that can realize their functionality. The **Initialization** phase of our functionality can also be modified in the same way and the protocol in Figure 3 can be shown to realize the resulting functionality with auth_i as the session ID. Although the protocol of Furukawa et al. [21] is proven secure in the standard model, its security relies on a new non-standard assumption called linear oracle bilinear Diffie-Hellman assumption. On the other hand, our protocol is proven secure in the random oracle model assuming the hardness of the standard computational Diffie-Hellman problem. It is arguable whether proofs in the standard model assuming a strong computational assumption are of more practical importance than proofs in the random oracle model assuming a weak computational assumption [30].

We have assumed the hash function H used in the protocol to be a random oracle, while proving that $\mathcal{N}_{gke}$ is a non-information oracle but not as a helper functionality in the simulation [22, 1]. This allowed the proof to be simple and yet demonstrating the usefulness of the functionality $\mathcal{F}^{+}_{\mathcal{GKE}}$. We leave open the task of constructing efficient protocols which can realize $\mathcal{F}^{+}_{\mathcal{GKE}}$ without the random oracle assumption.

Acknowledgments

The authors thank Mark Manulis for his comments on an earlier version of this paper. This work has been supported by the Australian Research Council Discovery Project grant DP0773348.

References

1. Michel Abdalla, Dario Catalano, Céline Chevalier, and David Pointcheval. Efficient Two-Party Password-Based Key Exchange Protocols in the UC Framework. In *Topics in Cryptology–CT-RSA'08*, volume 4964 of *LNCS*, pages 335–351. Springer, 2008.
2. Boaz Barak, Yehuda Lindell, and Tal Rabin. Protocol Initialization for the Framework of Universal Composability. Cryptology ePrint Archive, Report 2004/006, 2004.
3. M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated Key Exchange Secure against Dictionary Attacks. In *Advances in Cryptology–EUROCRYPT'00*, volume 1807 of *LNCS*, pages 139–155. Springer, 2000.
4. M. Bellare and P. Rogaway. Entity Authentication and Key Distribution. In *Advances in Cryptology–CRYPTO'93*, volume 773 of *LNCS*, pages 232–249. Springer, 1993.
5. Jens-Matthias Bohli, Maria Isabel Gonzalez Vasco, and Rainer Steinwandt. Secure group key establishment revisited. *Int. J. Inf. Sec.*, 6(4):243–254, 2007.
6. Colin Boyd and Anish Mathuria. *Protocols for Authentication and Key Establishment.* Information Security and Cryptography. Springer, August 2003.
7. Emmanuel Bresson, Olivier Chevassut, and David Pointcheval. Provably Authenticated Group Diffie-Hellman Key Exchange - The Dynamic Case. In *Advances in Cryptology–ASIACRYPT'01*, volume 2248 of *LNCS*, pages 290–309. Springer, 2001.
8. Emmanuel Bresson, Olivier Chevassut, and David Pointcheval. Dynamic Group Diffie-Hellman Key Exchange under Standard Assumptions. In *Advances in Cryptology–EUROCRYPT'02*, volume 2332 of *LNCS*, pages 321–336. Springer, 2002.
9. Emmanuel Bresson, Olivier Chevassut, David Pointcheval, and Jean-Jacques Quisquater. Provably authenticated group Diffie-Hellman key exchange. In *CCS'01: Proceedings of the 8th ACM conference on Computer and Communications Security*, pages 255–264. ACM, 2001.
10. Emmanuel Bresson and Mark Manulis. Securing Group Key Exchange against Strong Corruptions. In *Proceedings of ACM Symposium on Information, Computer and Communications Security (ASIACCS'08)*, pages 249–260. ACM Press, 2008.
11. Mike Burmester and Yvo Desmedt. A Secure and Efficient Conference Key Distribution System (Extended Abstract). In *EUROCRYPT*, pages 275–286, 1994.
12. Ran Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. Cryptology ePrint Archive, Report 2000/067, 2000. `http://eprint.iacr.org/`, Version updated on 13 Dec 2005.
13. Ran Canetti and Hugo Krawczyk. Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In *Advances in Cryptology - EUROCRYPT'01*, volume 2045 of *LNCS*, pages 453–474. Springer, 2001.
14. Ran Canetti and Hugo Krawczyk. Security Analysis of IKE's Signature-Based Key-Exchange Protocol. In *Advances in Cryptology–CRYPTO'02*, volume 2442 of *LNCS*, pages 143–161. Springer, 2002.
15. Ran Canetti and Hugo Krawczyk. Universally Composable Notions of Key Exchange and Secure Channels. In *Advances in Cryptology - EUROCRYPT'02*, volume 2332 of *LNCS*, pages 337–351. Springer, 2002. Full Version at `http://eprint.iacr.org/2002/059`.
16. Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *STOC '02: Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 494–503. ACM, 2002.
17. Ran Canetti and Tal Rabin. Universal Composition with Joint State. In *Advances in Cryptology–CRYPTO'03*, volume 2729 of *LNCS*, pages 265–281. Springer, 2003.
18. R Cleve. Limits on the security of coin flips when half the processors are faulty. In *STOC '86: Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 364–369, New York, NY, USA, 1986. ACM.
19. Yvo Desmedt, Josef Pieprzyk, Ron Steinfeld, and Huaxiong Wang. A Non-malleable Group Key Exchange Protocol Robust Against Active Insiders. In *Information Security–ISC'06*, volume 4176 of *LNCS*, pages 459–475. Springer, 2006.
20. R. Dutta and R. Barua. Provably Secure Constant Round Contributory Group Key Agreement in Dynamic Setting. *IEEE Transactions on Information Theory*, 54(5):2007–2025, May 2008.
21. Jun Furukawa, Frederik Armknecht, and Kaoru Kurosawa. A Universally Composable Group Key Exchange Protocol with Minimum Communication Effort. In *Security and Cryptography for Networks–SCN 2008*, volume 5229 of *LNCS*, pages 392–408. Springer, 2008.
22. Dennis Hofheinz and Jörn Müller-Quade. Universally Composable Commitments Using Random Oracles. In *Theory of Cryptography–TCC'04*, volume 2951 of *LNCS*, pages 58–76. Springer, 2004.
23. Dennis Hofheinz, Jörn Müller-Quade, and Rainer Steinwandt. Initiator-Resilient Universally Composable Key Exchange. In *Computer Security - ESORICS'03*, volume 2808 of *LNCS*, pages 61–84. Springer, 2003.

24. Jonathan Katz and Ji Sun Shin. Modeling insider attacks on group key-exchange protocols. In *Proceedings of the 12th ACM Conference on Computer and Communications Security–CCS'05*, pages 180–189. ACM, 2005.
25. Jonathan Katz and Moti Yung. Scalable Protocols for Authenticated Group Key Exchange. In *Advances in Cryptology–CRYPTO'03*, volume 2729 of *LNCS*, pages 110–125. Springer, 2003.
26. Hyun-Jeong Kim, Su-Mi Lee, and Dong Hoon Lee. Constant-Round Authenticated Group Key Exchange for Dynamic Groups. In *Advances in Cryptology–ASIACRYPT'04*, volume 3329 of *LNCS*, pages 245–259. Springer, 2004.
27. Mark Manulis. *Provably Secure Group Key Exchange*, volume 5 of *IT Security*. Europäischer Universitätsverlag, Berlin, Bochum, Dülmen, London, Paris, August 2007.
28. C.J. Mitchell, M. Ward, and P. Wilson. Key control in key agreement protocols. *IEE Electronic Letters*, 34(10):980–981, 1998.
29. J. Pieprzyk and H. Wang. Key Control in Multi-party Key Agreement Protocols. In *Workshop on Coding, Cryptography and Combinatorics (CCC 2003)*, volume 23 of *Progress in Computer Science and Applied Logic (PCS)*, pages 277–288, 2003.
30. David Pointcheval. Provable Security for Public Key Schemes. In *Contemporary Cryptology*, pages 133–189. Birkhuser, 2005.

A Review of Game-based Notions of Security for GKE

We first review the communication and adversarial model based on Katz and Shin [24]. We try to be as brief as possible due to lack of space.

Let $\mathcal{U} = \{U_1, \ldots, U_n\}$ be a fixed set of n parties. The protocol may be run among any subset of these parties. Each party has a pair of long-term public and private keys, (PK_U, SK_U) generated during an initialization phase prior to the protocol run. A group key exchange protocol π is modeled 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 invocation of the protocol π at party U.

Following [24], we assume that a unique session ID for each instance of the protocol is provided by a higher-level protocol. The session ID of an instance π_U^i is denoted by sid_U^i. We assume the pre-specified peer model and hence each party knows who the other participating parties are. 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 a public information. 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 communications network is controlled by an adversary $\mathcal{A}$, which schedules and mediates all sessions among the parties. If the adversary honestly forwards all messages between instances of parties in a given set pid, and each such instance holds the same value sid, then these instances all accept 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.

– $\mathsf{Execute}(\mathsf{sid},\mathsf{pid})$ prompts a complete execution of the protocol among the parties in pid using the unique session ID sid. $\mathcal{A}$ is given all the protocol messages, modeling passive attacks.
– $\mathsf{Send}(\pi_U^i,\mathsf{m})$ sends a message m to the instance π_U^i. If the message is $(\mathsf{sid},\mathsf{pid})$, the instance π_U^i is initiated with $(\mathsf{sid},\mathsf{pid})$. 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.

– $\mathsf{Corrupt(U)}$ The complete internal state of U including the long-term secret key SK_U of U is returned to $\mathcal{A}$. Note that this query does not return the session key, if computed.
– $\mathsf{Test}(\pi_U^i)$ A random bit b is secretly chosen. If $b = 1$, $\mathcal{A}$ is given sk_U^i established at π_U^i. Otherwise, a random string chosen from the session key probability distribution is given. Note that a Test query is allowed only on an accepted instance.

A.1 AKE Security

We present here the AKE-security notion in the strong corruption model, defined by Katz and Shin [24]. Unlike Bresson and Manulis [10], we do not separate strong corruption into long-term secret Key reveal and session state reveal queries.

The notion of *freshness* is central to defining AKE-security for GKE protocols. An instance π_U^i is unfresh if the instance π_U^i or any of its partners is asked a $\mathsf{RevealKey}$ after having accepted **or** a $\mathsf{Corrupt}(U')$ query is asked for some $U' \in \mathsf{pid}_U^i$ before π_U^i and its partners have terminated. In all other cases π_U^i is assumed to be fresh.

Definition 2. An adversary $\mathcal{A}_{ake}$ against the AKE-security notion is allowed to make $\mathsf{Execute}$, Send, $\mathsf{RevealKey}$ and $\mathsf{Corrupt}$ queries in Stage 1. $\mathcal{A}_{ake}$ makes a Test query to an instance π_U^i at the end of Stage 1 and it is given a challenge key K_b as described above. 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 remains fresh 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 $\mathsf{Adv}_{\mathcal{A}_{ake}} = 2 \cdot |\Pr[\mathsf{Succ}_{\mathcal{A}_{ake}}] - \frac{1}{2}|$. A protocol is called AKE-secure if $\mathsf{Adv}_{\mathcal{A}_{ake}}$ is negligible in the security parameter k for any polynomial time $\mathcal{A}_{ake}$.

A.2 Mutual Authentication

The notion of mutual authentication presented here is a modified version of the one by Bresson and Manulis [10], as we do not consider session state reveal separately.

Definition 3. An adversary $\mathcal{A}_{ma}$ against the mutual authentication of a correct GKE protocol π is allowed to ask $\mathsf{Execute}$, Send, $\mathsf{RevealKey}$ and $\mathsf{Corrupt}$ queries. $\mathcal{A}_{ma}$ wins the mutual authentication security game if at some point during the protocol run, there exist an uncorrupted party U whose instance π_U^i 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 $sk_{U'}^j \neq sk_U^i$.

Let $\mathsf{Succ}_{\mathcal{A}_{ma}}$ be the success probability of $\mathcal{A}_{ma}$ in winning the mutual authentication game. A protocol is said to provide mutual authentication in the presence of insiders if $\mathsf{Succ}_{\mathcal{A}_{ma}}$ is negligible in the security parameter k for any polynomial time $\mathcal{A}_{ma}$.

A.3 Contributiveness

The notion of contributiveness presented here can be seen as a strengthened notion of contributiveness defined by Bohli et al. [5], by considering strong and adaptive corruptions. As stated earlier, this notion does not consider opening attacks.

Definition 4. An adversary $\mathcal{A}_{con}$ against the contributiveness of correct GKE protocol π is allowed ask Execute, Send, RevealKey and Corrupt queries and It operates in two stages prepare and attack:

prepare. $\mathcal{A}_{con}$ queries the instances of π and outputs some state information ζ along with a description of a boolean valued algorithm χ. We denote by $\mathcal{K}_\chi$ a set of keys $\mathcal{K}_\chi = \{\tilde{k} | \tilde{k} \in \{0,1\}^k \text{ and } \chi(\tilde{k}) = \mathsf{true}\}$ such that $\frac{|\mathcal{K}_\chi|}{2^k}$ is negligible in the security parameter k.

At the end of prepare stage, a set Π is built such that Π consists of honest instances which have been asked either Execute or Send queries

attack. On input (χ, ζ, Π), $\mathcal{A}_{con}$ interacts with the instances of π as in the prepare stage.

At the end of this stage $\mathcal{A}_{con}$ outputs (U, i) and wins the game if an honest instance π_U^i has terminated accepting $\tilde{k} \in \mathcal{K}_\chi$ with $\pi_U^i \notin \Pi$

Let $\mathsf{Succ}_{\mathcal{A}_{con}}$ be the success probability of $\mathcal{A}_{con}$ in winning the above game. A protocol is said to provide contributiveness in the presence of insiders, if $\mathsf{Succ}_{\mathcal{A}_{con}}$ is negligible in the security parameter k for any polynomial time $\mathcal{A}_{con}$.

B Relaxed UC-security implies the existing notions

Claim 3. *Let π be a GKE protocol that securely realizes $\mathcal{F}_{\mathcal{GKE}}^{+}$. Then π is AKE secure as defined in Appendix A.1.*

Proof. (Sketch) We follow the proof idea of Canetti and Krawczyk [15], but give a direct proof without casting the AKE-security notion in the UC framework.

Assume that π securely realizes $\mathcal{F}_{\mathcal{GKE}}^{+}$ for some ITM $\mathcal{N}$. Hence $\hat{\pi}$ securely realizes $\hat{\mathcal{F}}_{\mathcal{GKE}}^{+}$, where $\hat{\pi}$ and $\hat{\mathcal{F}}_{\mathcal{GKE}}^{+}$ are multi-session extensions of π and $\mathcal{F}_{\mathcal{GKE}}^{+}$ respectively. Now, assume to the contrary that $\hat{\pi}$ is not AKE-secure i.e. there exists an adversary $\mathcal{A}_{ake}$ against the AKE-security of $\hat{\pi}$ such that $\mathsf{Adv}_{\mathcal{A}_{ake}}$ is non-negligible in the security parameter k. Then, we show that $\mathcal{N}$ is not a non-information oracle.

We use $\mathcal{A}_{ake}$ to construct an environment $\mathcal{Z}_{ake}$ and a real-world adversary $\mathcal{A}$. $\mathcal{A}$ runs $\mathcal{A}_{ake}$ as subroutine and gives it any public keys that were given to $\mathcal{A}$. Whenever $\mathcal{A}_{ake}$ asks a query or sends a message, $\mathcal{A}$ forwards it to $\mathcal{Z}_{ake}$ and any message sent by $\mathcal{Z}_{ake}$ is forwarded to $\mathcal{A}_{ake}$. $\mathcal{A}$ follows the instructions of $\mathcal{Z}_{ake}$ and they both proceed as follows:

1. When $\mathcal{A}_{ake}$ asks Execute(ssid,pid) query
   - If none of the parties in pid is corrupted, $\mathcal{Z}_{ake}$ invokes all parties in pid with input (new-session,sid,ssid,pid). The real-world adversary $\mathcal{A}$ sends these messages to $\mathcal{Z}_{ake}$, which will then be forwarded to $\mathcal{A}_{ake}$. Let (ssid,pid,κ) be the output of a party $U \in$ pid as observed by $\mathcal{Z}_{ake}$, $\mathcal{Z}_{ake}$ records session (U,ssid,pid,κ) and marks it completed and fresh.
   - If there exists at least one corrupted party, $\mathcal{Z}_{ake}$ invokes the uncorrupted parties (if exist) in pid as above and instructs $\mathcal{A}$ to run the protocol honestly on behalf of all corrupted parties. $\mathcal{A}$ forwards the messages from the corrupted parties to $\mathcal{Z}_{ake}$. The protocol transcript is also forwarded to $\mathcal{A}_{ake}$. If an uncorrupted party $U \in$ pid outputs (ssid,pid,κ), $\mathcal{Z}_{ake}$ records (U,ssid,pid,κ) and marks this session completed and unfresh.
2. When $\mathcal{A}_{ake}$ asks Send($\hat{\pi}_U^i$,(ssid,pid)) query
   - If no parties in pid is corrupted, $\mathcal{Z}_{ake}$ invokes U with input (new-session,sid,ssid,pid) and records (U,ssid,pid,*) as an uncompleted and fresh session.

- If some parties in pid are corrupted, $\mathcal{Z}_{ake}$ invokes uncorrupted parties (if any) in pid as above and instructs $\mathcal{A}$ to run run the protocol honestly on behalf of the corrupted parties. $\mathcal{Z}_{ake}$ records the session (U,ssid,pid,*) as uncompleted and unfresh.
  In both the cases above, $\mathcal{A}$ forwards the messages among the real parties to $\mathcal{Z}_{ake}$ and the transcripts are subsequently forwarded to $\mathcal{A}_{ake}$ as instructed by $\mathcal{Z}_{ake}$.

3. When $\mathcal{A}_{ake}$ asks a Send($\hat{\pi}^i_U$,m) query (let $\hat{\pi}^i_U$ be associated with the session (ssid, pid))
   - U is uncorrupted: If U is not yet invoked, $\mathcal{Z}_{ake}$ first invokes U with the input (new-session, sid, ssid, pid). It instructs $\mathcal{A}$ to deliver m to the appropriate instance at U and subsequently to deliver the outgoing message of this instance to $\mathcal{A}_{ake}$. If there is exist no previous record for this session, $\mathcal{Z}_{ake}$ records (U,ssid,pid,*) as uncompleted and fresh.
   - If U is corrupted, then $\mathcal{Z}_{ake}$ instructs $\mathcal{A}$ to execute the next step of the protocol honestly on behalf of U.
4. When a party U outputs a value (ssid,pid,κ), $\mathcal{Z}_{ake}$ records the output value and marks the instance as completed. Note that the instance would be fresh if U is uncorrupted and unfresh if U is corrupted (the output key in this case would have been computed locally by $\mathcal{Z}_{ake}$).
5. If $\mathcal{A}_{ake}$ asks a RevealKey($\hat{\pi}^i_U$) query for a recorded completed session with session key κ, $\mathcal{Z}_{ake}$ instructs $\mathcal{A}$ to hand over the key to $\mathcal{A}_{ake}$. Let (ssid, pid) be the session associated with the instance; $\mathcal{Z}_{ake}$ marks the session (U,ssid,pid,κ) as unfresh.
6. When $\mathcal{A}_{ake}$ issues a Corrupt(U) query, $\mathcal{A}$ provides the complete internal state of U to $\mathcal{A}_{ake}$. Each uncompleted session (ssid,pid) at U and any uncompleted sessions of the form (U', ssid, pid, *) (the partnered sessions) are marked unfresh. Note also that $\mathcal{Z}_{ake}$ marks all the future sessions established at U as unfresh.
7. When $\mathcal{A}_{ake}$ asks a Test($\hat{\pi}^i_U$) query on a completed session, $\mathcal{Z}_{ake}$ first chooses a bit $b \overset{R}{\leftarrow} \{0,1\}$. If $b = 0$, $\mathcal{A}_{ake}$ is given (via $\mathcal{A}$) the session key κ recorded as established at the instance. If $b = 1$, $\mathcal{A}_{ake}$ is given (via $\mathcal{A}$) a random key drawn from probability distribution of the session keys.
8. When $\mathcal{A}_{ake}$ outputs its guess bit b' for the test session, $\mathcal{Z}_{ake}$ proceeds as follows: Let (ssid, pid) be the session associated with the test instance $\hat{\pi}^i_U$. If there exist at least one unfresh record that contains (ssid, pid) (the test session and its partners), $\mathcal{Z}_{ake}$ outputs a random bit. Else, if $b' = b$, $\mathcal{Z}_{ake}$ outputs 1; otherwise, outputs 0.

It is easy to see that the simulation done by $\mathcal{Z}_{ake}$ and $\mathcal{A}$ for $\mathcal{A}_{ake}$ is valid. Since, $\hat{\pi}$ is not AKE-secure, $\mathcal{A}_{ake}$ wins the AKE-security game with non-negligible advantage. Hence, when $\mathcal{A}_{ake}$ interacts with $\mathcal{Z}_{ake}$ (via $\mathcal{A}$), the output of $\mathcal{Z}_{ake}$ is skewed non-negligibly away from fifty-fifty. Since $\hat{\pi}$ securely realizes $\hat{\mathcal{F}}^{+}_{\mathcal{GKE}}$ there exists an ideal adversary $\mathcal{S}$ that causes the same skew in the output of $\mathcal{Z}_{ake}$ after its interaction with $\mathcal{S}$ and an ideal process for $\hat{\mathcal{F}}^{+}_{\mathcal{GKE}}$. We use S to construct a distinguisher $\mathcal{M}$ that interacts with the copies of $\mathcal{N}$ and distinguishes between the output of any of these copies and a random value, thus showing that a copy of $\mathcal{N}$ is not a non-information oracle.

Let m be the maximum number of sessions invoked by $\mathcal{S}$ in the ideal world. $\mathcal{M}$ starts by choosing $l \overset{R}{\leftarrow} \{1, \ldots, m\}$ and simulates the interactions of $\mathcal{S}$ with $\mathcal{Z}_{ake}$ and an instance of $\hat{\mathcal{F}}^{+}_{\mathcal{GKE}}$ as follows:

1. Whenever $\mathcal{S}$ asks $\mathcal{Z}_{ake}$ to activate a subset of parties in $\mathcal{U}$ to invoke a session (with session ID sid) among them, $\mathcal{M}$ simulates $\hat{\mathcal{F}}^{+}_{\mathcal{GKE}}$ for $\mathcal{S}$. For each query of this type, $\mathcal{M}$ first invokes a new instance of $\mathcal{F}^{+}_{\mathcal{GKE}}$ within $\hat{\mathcal{F}}^{+}_{\mathcal{GKE}}$, which also requires running |pid| copies of $\mathcal{N}$ corresponding to that session. $\mathcal{M}$ runs these copies of $\mathcal{N}$ by itself except those that correspond to the l-th instance of $\mathcal{F}^{+}_{\mathcal{GKE}}$. The messages sent by $\mathcal{S}$ to the copies $\mathcal{N}$ that correspond to the l-th instance

of $\mathcal{F}^{+}_{\mathcal{GKE}}$ are forwarded to those copies and similarly all the messages from these copies of $\mathcal{N}$ are forwarded to $\mathcal{S}$ as coming from the l-th copy of $\mathcal{F}^{+}_{\mathcal{GKE}}$.

2. When $\mathcal{S}$ corrupts an ideal party U, $\mathcal{M}$ checks to see if this results in obtaining the internal state of copies of $\mathcal{N}$ which correspond to the l-th copy of $\mathcal{F}^{+}_{\mathcal{GKE}}$. If so it outputs a random bit and halts as the simulation failed. Otherwise, $\mathcal{M}$ gives $\mathcal{S}$ the internal states of the corresponding copies of $\mathcal{N}$(As $\mathcal{M}$ runs these copies of $\mathcal{N}$ by itself it knows the internal states).
3. When $\mathcal{S}$ chooses a test session that is not the l-th session, then $\mathcal{M}$ outputs a random bit as its simulation failed. If the test session corresponds to the l-th copy of $\mathcal{F}^{+}_{\mathcal{GKE}}$, then $\mathcal{M}$ is given either the local output of one of the copies of $\mathcal{N}$ (As the test session is fresh the local outputs of all these of $\mathcal{N}$ would be identical) or a random value. $\mathcal{M}$ then forwards this test value to $\mathcal{S}$ as the one given by $\mathcal{Z}_{ake}$.
4. When $\mathcal{S}$ outputs its guess for the bit b, $\mathcal{M}$ simply outputs the same guess.

Since $\hat{\pi}$ securely realizes $\hat{\mathcal{F}}^{+}_{\mathcal{GKE}}$ $\mathcal{S}$ outputs its guess with probability non-negligibly more than $\frac{1}{2}$. If $\mathcal{S}$ chooses the l-th session as the test session, then $\mathcal{M}$ also succeeds in distinguishing the output of any of the $|\mathsf{pid}|$ copies of $\mathcal{N}$ from a random value with the same probability. Note that there are a maximum of m sessions among a maximum of n parties. Hence, a total of $n \cdot m$ copies of $\mathcal{N}$ could be invoked. It follows that $\mathcal{M}$ succeeds in its guess with an advantage that is $\frac{|\mathsf{pid}|}{n \cdot m}$ times that of $\mathcal{S}$, which is non-negligible. □

Claim 4. *Let π be a GKE protocol that securely realizes $\mathcal{F}^{+}_{\mathcal{GKE}}$. Then π satisfies the mutual authentication notion as defined in Appendix A.2.*

Proof. Note that the definition in Appendix A.2 unifies the game-based notions of insider security defined by Katz and Shin [24]. The proof of this claim is similar to that of Claim 3 in Katz and Shin. We give a sketch of the proof here for completeness.

As π securely realizes $\mathcal{F}^{+}_{\mathcal{GKE}}$, we have $\hat{\pi}$ securely realizing $\hat{\mathcal{F}}^{+}_{\mathcal{GKE}}$, where $\hat{\pi}$ and $\hat{\mathcal{F}}^{+}_{\mathcal{GKE}}$ are multi-session extensions of π and $\mathcal{F}^{+}_{\mathcal{GKE}}$ respectively. Now, assume to the contrary that $\hat{\pi}$ does not satisfy the definition of mutual authentication i.e. there exists an adversary $\mathcal{A}_{ma}$ against $\hat{\pi}$ that has non-negligible advantage in winning the mutual authentication game defined in Appendix A.2.

We use $\mathcal{A}_{ma}$ to construct an environment $\mathcal{Z}_{ma}$ and a real-world adversary $\mathcal{A}$ such that for any ideal adversary $\mathcal{S}$, $\mathcal{Z}_{ma}$ can distinguish whether it interacts with $\mathcal{A}$ and the players running the real protocol $\hat{\pi}$ or with $\mathcal{S}$ and the ideal process for $\hat{\mathcal{F}}^{+}_{\mathcal{GKE}}$. $\mathcal{A}$ runs $\mathcal{A}_{ma}$ as subroutine and gives it any public keys that were given to $\mathcal{A}$. Whenever $\mathcal{A}_{ma}$ asks a query or sends a message, $\mathcal{A}$ forwards it to $\mathcal{Z}_{ma}$ and any message sent by $\mathcal{Z}_{ma}$ is forwarded to $\mathcal{A}_{ma}$. $\mathcal{A}$ follows the instructions of $\mathcal{Z}_{ma}$ and they both proceed as follows:

1. All the queries of asked by $\mathcal{A}_{ma}$ are handled as described in the proof of Claim 3 above.
2. $\mathcal{Z}_{ma}$ outputs 1 if any of the following events happens:
   (a) There exist two players U and U' and a completed and fresh session (U, ssid, pid, κ) such that U' was not corrupted before this session was completed, $U' \in$ pid, but there is no session (U',ssid, pid, $*$).
   (b) There exist two players U and U' and sessions (U, ssid, pid, κ) and (U', ssid, pid, κ'), which are completed and fresh but $\kappa \neq \kappa'$.
3. In all other events, $\mathcal{Z}_{ma}$ outputs 0.

It is easy to see that the simulation done by $\mathcal{Z}_{ma}$ and $\mathcal{A}$ for $\mathcal{A}_{ma}$ is valid. Note that we have assumed that $\mathcal{A}_{ma}$ has non-negligible advantage in winning the mutual authentication game. Hence, $\mathcal{Z}_{ma}$ outputs 1 whenever $\mathcal{A}_{ma}$ violates the mutual authentication in the protocol $\hat{\pi}$.

Now consider $\mathcal{Z}_{ma}$'s interaction with $\mathcal{S}$ and an ideal process for $\hat{\mathcal{F}}^{+}_{\mathcal{GKE}}$. In this case we argue that $\mathcal{Z}_{ma}$ never outputs 1. Firstly, the Event 2a above does not happen in the ideal world because, as per the definition of the ideal functionality $\mathcal{F}^{+}_{\mathcal{GKE}}$, the copy of $\mathcal{F}^{+}_{\mathcal{GKE}}$ running within $\hat{\mathcal{F}}^{+}_{\mathcal{GKE}}$ does not proceed to **Key Generation** phase unless it receives $(\mathsf{sid}, \mathsf{ssid}, \mathsf{pid}, \mathsf{new\text{-}session})$ from all users $U \in \mathsf{pid}$. Next, the Event 2b also does not happen in the ideal world. To see why, note that when all the parties $U \in \mathsf{pid}$ are uncorrupted $\mathcal{F}^{+}_{\mathcal{GKE}}$ generates a uniformly random key. When there exist at least one uncorrupted party, $\mathcal{F}^{+}_{\mathcal{GKE}}$ first checks if the copies of non-information oracles $\mathcal{N}_i$ corresponding to the uncorrupted parties $U_i \in \mathsf{pid}$ output identical session key values. In both the cases, $\mathcal{F}^{+}_{\mathcal{GKE}}$ distributes the same session key to all uncorrupted parties.

Since, $\mathcal{Z}_{ma}$ can distinguish its interaction with $\hat{\pi}$ running in the presence of $\mathcal{A}$ from its interaction with $\mathcal{S}$ and an ideal process for $\hat{\mathcal{F}}^{+}_{\mathcal{GKE}}$, it violates the UC-security of $\hat{\pi}$. This is a contradiction to our initial assumption. Hence, π satisfies the mutual authentication notion. □

Claim 5. *Let π be a GKE protocol that securely realizes $\mathcal{F}^{+}_{\mathcal{GKE}}$. Then π satisfies the contributiveness notion as defined in Appendix A.3.*

Proof. Assume that π securely realizes $\mathcal{F}^{+}_{\mathcal{GKE}}$. Thus $\hat{\pi}$ securely realizes $\hat{\mathcal{F}}^{+}_{\mathcal{GKE}}$. Now, assume that $\hat{\pi}$ does not guarantee contributiveness as per Definition 4 in Appendix A.3, then we show that $\hat{\pi}$ does not securely realize $\hat{\mathcal{F}}^{+}_{\mathcal{GKE}}$. Let $\mathcal{A}_{con}$ be an adversary that violates the contributiveness of $\hat{\pi}$.

We construct an environment $\mathcal{Z}_{con}$ and a real-world adversary $\mathcal{A}$ using $\mathcal{A}_{con}$ such that for an ideal adversary $\mathcal{S}$, $\mathcal{Z}_{con}$ distinguishes an execution of $\mathcal{A}$ with the parties running the protocol $\hat{\pi}$ from an execution of $\mathcal{S}$ with the ideal process for $\hat{\mathcal{F}}^{+}_{\mathcal{GKE}}$. $\mathcal{Z}_{con}$ and $\mathcal{A}$ proceed as follows:

1. The queries Execute, Send, Corrupt and RevealKey asked by $\mathcal{A}_{con}$ are handled in the same way as $\mathcal{Z}_{ake}$ does in the proof of Claim 3.
2. $\mathcal{Z}_{con}$ records the output of $\mathcal{A}_{con}$ at the end of prepare and continues to answer $\mathcal{A}_{con}$'queries in attack stage. Specifically, it constructs the set $\mathcal{K}_\chi$ from $\mathcal{A}_{con}$'s output.
3. During its execution $\mathcal{Z}_{con}$ outputs 1 if the following event happens: there exists a completed and fresh session $(U, \mathsf{ssid}, \mathsf{pid}, \tilde{\kappa})$ that is invoked after $\mathcal{A}_{con}$'s prepare stage such that $\tilde{\kappa} \in \mathcal{K}_\chi$.
4. $\mathcal{Z}_{con}$ outputs 0, otherwise.

Note that, whenever $\mathcal{A}_{con}$ violates the contributiveness of $\hat{\pi}$, $\mathcal{Z}_{con}$ outputs 1. As we assumed that $\hat{\pi}$ does not guarantee contributiveness this happens with a non-negligible probability. On the other hand, in an interaction with $\mathcal{S}$ and the ideal process for $\hat{\mathcal{F}}^{+}_{\mathcal{GKE}}$, the event in Step 3 happens with negligible probability; thus $\mathcal{Z}_{con}$ outputting 1 in this interaction is also negligible. To see why, note that as long as there exists at least one uncorrupted party U, the copy of $\mathcal{F}^{+}_{\mathcal{GKE}}$ corresponding to the session (sid,ssid,pid) running within $\hat{\mathcal{F}}^{+}_{\mathcal{GKE}}$, sets the session key to be the local output of a copy of $\mathcal{N}$. The probability of the copy of $\mathcal{N}$ corresponding to the uncorrupted party outputting a key from the set $\mathcal{K}_\chi$ is $\frac{|\mathcal{K}_\chi|}{2^k}$, which is negligible in the security parameter as per Definition 4. This implies that $\hat{\pi}$ and consequently π do not securely realize $\hat{\mathcal{F}}^{+}_{\mathcal{GKE}}$ and $\mathcal{F}^{+}_{\mathcal{GKE}}$ respectively. Hence, we conclude that a protocol π that securely realizes $\mathcal{F}^{+}_{\mathcal{GKE}}$ guarantees contributiveness. □