IET Information Security (IET-IFS)

Vol 2, Issue 3, pp. 85–93. IET, September 2008.

Contributory Group Key Exchange in the Presence of Malicious Participants

Emmanuel Bresson and Mark Manulis

Abstract

In a group key exchange protocol, the resulting group key should be computed by all participants such that none of them can gain any advantage concerning the protocol's output: misbehaving participants might have personal advantage in influencing the final value of the key. In fact, the absence of trust relationship is the main feature of group key exchange (when compared to group key transport) protocols. This paper enlarges existing notions of security by identifying limitations in some previously proposed security models. To illustrate these notions, two efficient and provably secure generic solutions – compilers – are presented.

Index Terms

Group key exchange, Malicious participants, Key control, Contributiveness, Security model, Compiler

I. Introduction

Group Key Exchange protocols (GKE) is a method of key establishment characterized by the fact that no secure channels are needed and, more important, no party is allowed to choose the key on behalf of the group: in other words, group members do not trust each other. This strong but much realistic requirement provides background and motivation for considering malicious participants in such protocols and for defining in a formal way what security means in that case. Such formalization is one of the main goals of this paper. In the paradigm of *provable security*, security is analyzed in the framework of a security model. Such model has been defined for two-party protocols [2],

E. Bresson is at DCSSI Crypto Lab Paris, emmanuel@bresson.org

M. Manulis is at Horst Görtz Institute, Ruhr University Bochum, Germany, mark.manulis@nds.rub.de

[3] and multi-party protocols [8]: a security proof shows that the established key looks random to any outsider (this is called AKE (Authenticated Key Exchange) security), and that any pair of parties mutually agree on having computed the same key (this the MA (Mutual Authentication) property). We refer to [7], [18], [17] for refinements and to [22] for a survey.

A number of papers [24], [1], [11], [17] point out that the consideration of corrupted participants (either curious or malicious) is of prime importance in the group setting, because they can have catastrophic effects on the protocol security; for instance, Choo *et al.* [11] noticed that some protocols proven secure in the BCPQ-like models are vulnerable to *unknown key-share attacks*, in which the attacker is *believed* (from some participant's view) to be a group member.

II. Contributions and Organization

This paper provides an extended treatment of security of GKE protocols in the presence of malicious participants We formally define what a "secure group key" means in such scenario.

We start by discussing the related work and some limitations in currently known security models (Section III). Then we describe our extended model and formalize new security definitions (Section IV). Our model is both general and powerful: in particular, we formalize how to take into account "corruptions" of secrets held by the participants, both in case of long-term secrets (e.g. authentication keys) and ephemeral data (e.g. randomness or keying material). To prove the soundness and feasibility of our extensions, in Sections V and VI we propose two generic solutions (compilers) which turn any AKE-secure GKE protocol into an enhanced protocol, which provably satisfies our advanced security requirements.

III. Related Work

A. General Security Notions for GKE Protocols

AKE-security for group key exchange was formalized (and later refined) in [8], [7], [18]. We will refer to this model as the BCPQ security model. The security notion is powerful in the sense that it subsumes several informal security goals defined in the literature, among which: key secrecy [14], implicit key authentication [23], security against impersonation attacks [10], resistance against known-key attacks [26], [9], key independence [20]. Also it can be combined with *(perfect) forward secrecy* [16], [14], [23] which requires that the disclosure of long-lived keys must not compromise the secrecy of the previously established group keys. An even stronger requirement is that of *strong forward secrecy* [7], [25] in which the adversary can read the ephemeral secrets used during the protocol execution. Finally, the formal definition of MA-security in [8] has been designed to cover the informal definitions of key confirmation [23, § 12.2] and explicit key authentication [23, § 12.2].

B. Informal Definitions of Contributiveness in the Presence of Malicious Players

According to [11], the above definitions of AKE and MA-security are not sufficient to handle *unknown key-share attacks* [14], [5], in which a corrupted participant can make an honest participant believe that the key is shared with one party though in fact it is shared with another party.

There have been, however, only few attempts to consider malicious participants in GKE protocols. Misbehavior of protocol participants was first mentioned in [24], under the name of *key control*. Independently, Ateniese *et al.* [1] introduced the more general notion of *unpredictability* (which intuitively preventss key control), and further proposed a related notion called *(verifiable) contributory group key agreement*: the property by which each participant equally contributes to the resulting group key and guarantees its freshness in a verifiable manner. A weaker model (as in [6]) considers participants who are honest but have biased pseudo-random generators, such that the adversary can influence the key. In this paper we consider a stronger setting (in spirit of [4]), where malicious participants try to influence honest participants computing some special value as a group key (thus including the so-called *key replication* attacks [21]).

C. Formal Models dealing with Malicious Participants

1) The KS Model: Based on the aforementioned papers, Katz and Shin [17] proposed security definitions against malicious participants in a BCPQ-like model: they formalize the notion of adversary *impersonating player A to player B* and states what security means in such scenario.

The authors of [17] described a compiler to turn any AKE-secure protocol (in the sense of BCPQ) into a protocol secure in their extended model. It can be shown however, that their compiler is not complete in the sense that the resulting protocol may not be contributory if the basic protocol is not so.

2) The BVS Model: Another extension has been proposed by Bohli *et al.* [4] towards security goals in the presence of malicious participants. The process dealing with contributiveness, at an informal level, runs as follows. In a first stage, the adversary $\mathcal{A}$ interacts with the users and may corrupt some of them; $\mathcal{A}$ then specifies an unused instance oracle Π_i^s and a subset K in the session key space $\mathcal{K}$. In the second stage, the adversary tries to make Π_i^s accept a session key $k \in K$ but is not allowed to corrupt U_i. The BVS model defines a GKE protocol as being t*-contributory* if the adversary succeeds with only negligible probability, with the total number of corruptions remains (strictly) less than t. A n-contributory protocol between n participants is called a *key agreement*.

While quite appealing, this model suffers from two drawbacks: first, the adversary is not adaptive in her choice of Π_i^s and must commit to it in the first stage; second, strong corruptions are not allowed: contributiveness does not capture attacks in which $\mathcal{A}$ tries to influence the session key using the (passive) knowledge of some ephemeral secrets.

IV. Our Extended Security Model

In the following we propose a security model for GKE protocols that includes extended security definitions concerning MA-security and contributiveness, while taking into account strong corruptions.

A. Definition of a Group Key Exchange Protocol

1) Users and Oracles: Similar to [7] $\mathcal{U}$ is a set of N users; each user $U_i \in \mathcal{U}$ holds a long-lived key LL_i, and has an unlimited number of instances called *oracles*, involved in distinct concurrent protocol executions; Π_i^s, with $s \in \mathbb{N}$, denotes the s-th instance oracle of U_i. We write Π_U^s if no specific user is meant.

Every Π_U^s maintains an *internal state information* $\texttt{state}_U^s$ which is composed of all ephemeral information used during the protocol execution. The long-lived key LL_U is, in nature, excluded from it (moreover the long-lived key is specific to the user, not to the oracle).

An oracle Π_U^s is *unused* if it has never been initialized. Each unused oracle Π_U^s can be initialized with the long-lived key LL_U, when it becomes part of some group $\mathcal{G}$. When an oracle Π_U^s collects enough information to compute the session group key, it *accepts*. When it finishes to send or receive messages, the oracle *terminates* the protocol execution. If the execution fails (due to any adversarial actions) then Π_U^s terminates without having accepted, and the session key k_U^s is set to some undefined value.

2) Session ID, Partner ID, Session Group Key, Group Members: Every session is identified by a unique, publicly-known *session id* $\mathtt{sid}_U^s$. In each session each participating oracle Π_U^s gets a value $\mathtt{pid}_U^s$ that contains the identities of participating users (including U) and computes *session group key* $k_U^s \in \{0,1\}^\kappa$ where κ is the security parameter.

By $\mathcal{G}(\Pi_i^s) = \{\Pi_j^t,\text{ where } U_j \in \mathtt{pid}_{U_i}^s \text{ and } \mathtt{sid}_i^s = \mathtt{sid}_j^t\}$ we denote the *group of oracle* Π_i^s and say that Π_i^s and Π_j^t are *partnered* if $\Pi_j^t \in \mathcal{G}(\Pi_i^s)$ and $\Pi_i^s \in \mathcal{G}(\Pi_j^t)$.

Sometimes we simply write $\mathcal{G}$ to denote the *group of oracles* participating in the same protocol session. Then each oracle in $\mathcal{G}$ is called a *group member*. Note that oracles in $\mathcal{G}$ may be ordered, e.g., lexicographically based on the user identities.

Definition 1 (GKE Protocol): A group key exchange protocol $\mathtt{P}$ consists of the key generation algorithm $\mathtt{KeyGen}$, and a protocol $\mathtt{Setup}$ defined as follows:

- $\mathtt{P.KeyGen}(1^\kappa)$: On input a security parameter 1^κ each user in $\mathcal{U}$ is provided with a long-lived key LL_U.
- $\mathtt{P.Setup}(\mathcal{S})$: On input a set $\mathcal{S}$ of n unused oracles a new group $\mathcal{G}$ is created and set to be $\mathcal{S}$, then a probabilistic interactive protocol is executed between oracles in $\mathcal{G}$.

We call $\mathtt{P.Setup}$ an *operation*. We say that a protocol is *correct* if all oracles accept with the same group key. We assume it is the case for all protocols in this paper.

B. Adversarial Model

We now consider an adversary $\mathcal{A}$ which is a Probabilistic Polynomial-Time (PPT) algorithm having complete control over the network. $\mathcal{A}$ can invoke protocol execution and interact with protocol participants via queries to their oracles.

- *Execute*$(\mathcal{S})$: This query models $\mathcal{A}$ eavesdropping protocol executions. Formally, $\mathtt{P.Setup}(\mathcal{S})$ is run and $\mathcal{A}$ is given the transcript.
- *Send*(Π_U^s, m): This query models $\mathcal{A}$ sending messages to the oracles. $\mathcal{A}$ receives the response which Π_U^s would have generated after having processed the message m according to the description of $\mathtt{P}$. An new execution of

$\mathtt{P.Setup}(\mathcal{S})$ can be initiated via a *Send*$(\Pi_U^s, \mathcal{S})$ query, which returns the first message that Π_U^s would generate in this case.

- *RevealKey*(Π_U^s): $\mathcal{A}$ is given the session group key k_U^s. This query is answered only if Π_U^s has accepted.
- *RevealState*(Π_U^s): $\mathcal{A}$ is given the internal state information $\mathtt{state}_U^s$.
- *Corrupt*(U): $\mathcal{A}$ is given the long-lived key LL_U.

We say that Π_U^s is a *malicious participant* if the adversary has previously asked the *Corrupt*(U) query, thus gained the ability to act on behalf of U. In all other cases Π_U^s is *honest*. We say that the adversary *opens* an instance if it asks a *RevealState*(Π_U^s) query for some honest Π_U^s. This is possible since long-lived keys are separated from the ephemeral secrets stored in $\mathtt{state}_U^s$.

With the following definition we emphasize the substantial difference between weak and strong corruptions in our model, namely the access to the query *RevealState*.

Definition 2 (Weak/Strong Corruption Models): We say that a PPT adversary $\mathcal{A}$ operates in the *weak corruption model* if it is given access to the queries *Execute*, *Send*, *RevealKey*, *Corrupt*, and *Test*; and in the *strong corruption model* if it is additionally given access to the query *RevealState*.

In the following we provide definitions of AKE-/MA-security, and contributiveness whereby distinguishing between weak and strong corruption models.

C. AKE-Security with Weak/Strong Forward Secrecy

Perfect forward secrecy [8], which we also refer to as *weak forward secrecy* ($\mathtt{wfs}$), states that AKE-security of previously computed session keys is preserved if the adversary obtains long-lived keys of protocol participants in later protocol sessions. As extended in [7], *strong forward secrecy* ($\mathtt{sfs}$) states that AKE-security should still be preserved if the adversary obtains additionally ephemeral secrets of participating oracles in later protocol sessions.

Definition 3 (Oracle α-Freshness): Let $\alpha \in \{\mathtt{wfs}, \mathtt{sfs}\}$. In the execution of $\mathtt{P}$ the oracle Π_U^s is $\mathtt{wfs}$-*fresh* if **all** of the following holds:

- no $U_i \in \mathtt{pid}_U^s$ is asked for a *Corrupt* query prior to a query of the form *Send*(Π_j^t, m) such that $U_j \in \mathtt{pid}_U^s$ before Π_U^s and all its partners accept;

- neither Π_U^s nor any of its partners is asked for a *RevealKey* query after having accepted.

We say oracle Π_U^s is $\mathtt{sfs}$-*fresh* if it is $\mathtt{wfs}$-fresh *and* neither Π_U^s nor its partners are asked for a *RevealState* query before Π_U^s and all its partners accept. We say that a *session is α-fresh* if all participating oracles are α-fresh.

We emphasize that the ephemeral data $\mathtt{state}_U^s$ is specific to a session (user instances are so). Thus, and oracle remains α-fresh if *RevealState* and *RevealKey* queries are asked to other oracles owned by the same user. Hence, in contrast to [7] (and [18]) our definition of $\mathtt{sfs}$-freshness allows the adversary to obtain knowledge of internal states from earlier sessions too.

Definition 4 (AKE-Security): Let b a uniformly chosen bit, and $\mathcal{A}$ an active adversary $\mathcal{A}$ operating in the weak ($\alpha = \mathtt{wfs}$) or strong ($\alpha = \mathtt{sfs}$) corruption model. We define the game $\mathsf{Game}_{\alpha,\mathrm{P}}^{\mathsf{ake}-b}(\mathcal{A},\kappa)$ defined as follows:

- in a first stage, $\mathcal{A}$ interacts with instance oracles using queries;
- at some point $\mathcal{A}$ asks a *Test* query to a α-**fresh** oracle Π_U^s which has accepted. This query is answered as follows: if $b=1$, $\mathcal{A}$ receives $k_1 := k_U^s$; if $b=0$, it receives $k_0 \in_R \{0,1\}^\kappa$;
- in the second stage, $\mathcal{A}$ continues interacting with instance oracles;
- when $\mathcal{A}$ terminates, it outputs a bit trying to guess b.

The output of $\mathcal{A}$ is the output of the game. The advantage function (over all adversaries running within time κ) in winning this game is defined as:

$$\mathsf{Adv}_{\alpha,\mathrm{P}}^{\mathsf{ake}}(\kappa) := \max_{\mathcal{A}} \left|2\Pr[\mathsf{Game}_{\alpha,\mathrm{P}}^{\mathsf{ake}-b}(\mathcal{A},\kappa) = b] - 1\right|$$

A GKE protocol P is *AKE-secure with weak forward secrecy* ($\mathtt{AGKE\text{-}wfs}$) if $\mathsf{Adv}_{\mathtt{wfs},\mathrm{P}}^{\mathsf{ake}}(\kappa)$ is negligible, and *AKE-secure with strong forward secrecy* ($\mathtt{AGKE\text{-}sfs}$) if $\mathsf{Adv}_{\mathtt{sfs},\mathrm{P}}^{\mathsf{ake}}(\kappa)$ is negligible.

D. MA-Security in the Presence of Malicious Participants

Our definition enlarges the one in [7], [8] by considering malicious participants. In the weak corruption model it can be seen as a replacement for definitions in [17]. In the strong corruption model it is even stronger due to *RevealState* queries to honest oracles.

Definition 5 (MA-Security): Let $\mathcal{A}$ be an active adversary and $\mathsf{Game}_{\mathrm{P}}^{\mathsf{ma}}(\mathcal{A},\kappa)$ the interaction between $\mathcal{A}$ and the instance oracles, in the weak/strong corruption model. We say that $\mathcal{A}$ *wins* if, at some point, there exist an uncorrupted user U_i whose instance oracle Π_i^s has accepted with k_i^s and another user U_j with $U_j \in \mathtt{pid}_i^s$ that is uncorrupted at the time Π_i^s accepts, such that

1) there is **no** instance oracle Π_j^t with $(\mathtt{pid}_j^t, \mathtt{sid}_j^t) = (\mathtt{pid}_i^s, \mathtt{sid}_i^s)$, **or**
2) there is **an** instance oracle Π_j^t with $(\mathtt{pid}_j^t, \mathtt{sid}_j^t) = (\mathtt{pid}_i^s, \mathtt{sid}_i^s)$ that accepted with $k_j^t \neq k_i^s$.

The maximum probability of this event (over all adversaries running within time κ) is denoted $\mathsf{Succ}_{\mathtt{P}}^{\mathsf{ma}}(\kappa)$. We say that a GKE protocol $\mathtt{P}$ is *MA-secure* ($\mathtt{MAGKE}$) if this probability is a negligible function of κ.

Note that U_i and U_j must be uncorrupted, however, $\mathcal{A}$ operating in the strong corruption model can ask *RevealState* queries to all honest oracles, including Π_i^s and Π_j^t.

E. Contributiveness in the Presence of Malicious Participants

We start with the definition of contributiveness in the strong corruption model. Informally, an active adversary allowed to corrupt $n-1$ group members and reveal internal states of all n oracles must not be able to predict the key computed by an (instance of) honest player.

Definition 6 (Contributiveness in the Strong Corruption Model): Let $\mathcal{A}$ be an adversary (in the strong corruption model) that interacts with oracles in two stages: $\mathsf{prepare}$ and attack, according to the following game $\mathsf{Game}_{\mathtt{P}}^{\mathsf{con}}(\mathcal{A}, \kappa)$:

- $\mathcal{A}(\mathsf{prepare})$ interacts with the oracles. At the end of the stage, it outputs $\tilde{k} \in \{0,1\}^\kappa$, and some state information ζ;
- the following sets are built: $\mathcal{G}_{\mathtt{us}}$ consisting of all honest used oracles, $\mathcal{G}_{\mathtt{run}}$ consisting of all honest oracles that are involved in one execution ($\mathcal{G}_{\mathtt{run}} \subseteq \mathcal{G}_{\mathtt{us}}$), and Ψ consisting of session ids $\mathtt{sid}_j^t$ for every $\Pi_j^t \in \mathcal{G}_{\mathtt{us}}$;
- $\mathcal{A}(\mathsf{attack}, \zeta)$ continues its interaction. At the end of the stage $\mathcal{A}$ outputs (s, U).

The adversary $\mathcal{A}$ wins in $\mathsf{Game}_{\mathcal{A},\mathtt{P}}^{\mathsf{con}}(\kappa)$ if **all** of the following holds:

1) Π_U^s has accepted and terminated with $\tilde{k}$, no *Corrupt*(U) has been asked, $\Pi_U^s \notin \mathcal{G}_{\mathtt{us}} \setminus \mathcal{G}_{\mathtt{run}}$ and $\mathtt{sid}_U^s \notin \Psi$.
2) There are at most $n-1$ corrupted users U_j having oracles Π_j^t partnered with Π_U^s.

The maximal probability (over all adversaries running within time κ) in winning the game is defined as

$$\mathsf{Succ}_{\mathtt{P}}^{\mathsf{con}}(\kappa) := \max_{\mathcal{A}} \left| \Pr[\mathcal{A} \text{ wins in } \mathsf{Game}_{\mathtt{P}}^{\mathsf{con}}(\mathcal{A}, \kappa)] \right|$$

We say that a GKE protocol $\mathtt{P}$ is *contributory in the strong corruption model* ($\mathtt{sCGKE}$) if this probability is a negligible function of κ.

a) Comments: The condition $\Pi_U^s \notin \mathcal{G}_{\mathtt{us}} \setminus \mathcal{G}_{\mathtt{run}}$ rules out the trivial case where $\mathcal{A}$ as malicious participant of a session outputs $\tilde{k}$ which is then accepted by Π_U^s participating in the same session (note that participants do not compute group key synchronously). The condition $\mathtt{sid}_U^s \notin \Psi$ rules out another trivial case where $\mathcal{A}$ during its attack stage outputs (s, U) such that Π_U^s has accepted with $\tilde{k}$ earlier in the $\mathsf{prepare}$ stage.

Definition 6 ensures unpredictability of group keys and is sufficient for preventing key-replication attacks. However, (similar to [4]) this definition does not deal with the unpredictability of *some bits* of the group key. The main reason is that all ephemeral secrets used by the honest participants during the protocol execution can be revealed by the adversary. Intuitively, unpredictability of some bits of the group key, or in other words the uniform distribution of session group keys computed in the presence of malicious participants, is related to the problem of *asynchronous distributed coin tossing* for probabilistic algorithms without trusted parties and trapdoor permutations for which a theoretical bound of at most $(n-1)/2$ corrupted parties exists [12]. On the other hand, in the weak corruption model (without *RevealState* queries) this can be easily achieved, e.g., via commitments as in [24], [19], [13], whereas strong corruptions can reveal the committed secrets as part of $\mathtt{state}_U^s$.

For completeness, we give in the following an alternative definition for contributiveness in the weak corruption model.

Definition 7 (Contributiveness in the Weak Corruption Model): Let b a uniformly chosen bit, $\mathcal{K} := \emptyset$ an initially empty *set of keys*, and $\mathcal{A}$ an adversary operating in the weak corruption model running in three stages, $\mathsf{prepare}$, attack and decide, according to $\mathsf{Game}_{\mathtt{P}}^{\mathsf{con}-b}(\mathcal{A}, \kappa)$:

- $\mathcal{A}(\mathsf{prepare})$ interacts with the oracles and finally outputs some state information ζ;
- the following sets are built: $\mathcal{G}_{\mathtt{us}}$ consisting of all honest used oracles, $\mathcal{G}_{\mathtt{run}}$ consisting of all honest oracles involved in an execution ($\mathcal{G}_{\mathtt{run}} \subseteq \mathcal{G}_{\mathtt{us}}$), and Ψ consisting of session ids $\mathtt{sid}_j^t$ for every $\Pi_j^t \in \mathcal{G}_{\mathtt{us}}$;
- $\mathcal{A}(\mathsf{attack}, \zeta)$ interacts with oracles and finally outputs some updated state information ζ';
- Let q_{s} denote the number of sessions invoked by $\mathcal{A}$ in the stages $\mathsf{prepare}$ and attack. Then the sets $\mathcal{G}_1, \ldots, \mathcal{G}_q$, $q \leq q_{\mathsf{s}}$ are built as follows: each set $\mathcal{G}_i$, $i \in [1, q]$ consists of all honest oracles Π_j^t holding the same pair

$(\mathtt{pid}_j^t, \mathtt{sid}_j^t)$ with $\Pi_j^t \notin \mathcal{G}_{\mathtt{us}}$ and $\mathtt{sid}_j^t \notin \Psi$. Then, for each $\mathcal{G}_i$, the session group key accepted by the *first honest* oracle in $\mathcal{G}_i$ (if $b = 1$) or a random element (if $b = 0$) is added to $\mathcal{K}$. If $\mathcal{K}$ is not empty then $\mathcal{A}$ is invoked for the decide stage.

- $\mathcal{A}(\mathsf{decide}, \zeta, \mathcal{K})$ without asking any further queries outputs a bit trying to guess b.

The output of $\mathcal{A}$ in the stage decide is the output of the game. The advantage function (over all adversaries running within time κ) in winning this game is defined as:

$$\mathsf{Adv}_{\mathtt{P}}^{\mathsf{con}}(\kappa) := \max_{\mathcal{A}} \left| 2 \Pr[\mathsf{Game}_{\mathtt{P}}^{\mathsf{con}-b}(\mathcal{A}, \kappa) = b] - 1 \right|$$

We say that a GKE protocol $\mathtt{P}$ is *contributory in the weak corruption model* ($\mathtt{wCGKE}$) if $\mathsf{Adv}_{\mathtt{P}}^{\mathsf{con}}(\kappa)$ is negligible.

b) Comments: The state information ζ returned by $\mathcal{A}$ in the $\mathsf{prepare}$ stage is given as input to the stages attack and decide. However, $\mathcal{A}$ in the decide stage does not obtain any state information from the attack stage. Thus, these stages run isolated. The core idea in the definition is to let $\mathcal{A}$ to distinguish whether elements of $\mathcal{K}$ are real session group keys, computed by honest participants in the presence of malicious ones ($b = 1$), or randomly chosen values ($b = 0$).

V. Compiler C-MACONS

In this section we propose a compiler which can be used to turn any AKE-secure GKE protocol into a GKE protocol which is additionally MA-secure and provides contributiveness in the strong corruption model. If $\mathtt{P}$ is a GKE protocol, by $\mathtt{C\text{-}MACONS}_{\mathtt{P}}$ we denote the compiled protocol.

A. Main Ideas

In the following, we assume that each message sent by Π_U^s can be parsed as $U|m$ consisting of the sender's identity U and a message m; for simplicity we will use the same s for all oracles of the session. Additionally, an authentication token σ, e.g., a digital signature on m, can be attached.

After computing the session group key k in the underlying protocol $\mathtt{P}$ participants execute $\mathtt{C\text{-}MACONS}$. In a first communication round they exchange random nonces r_i that are concatenated into a session id $\mathtt{sid}$ (a classical way to define unique session ids). Then, each participant iteratively computes values $\rho_1, \ldots, \rho_n$ by adequately using a pseudo-random function f, in such a way that every random nonce (contribution of each participant) is embedded into the computation of $K := \rho_n$. The intuition is that malicious participants cannot influence this computation. The second communication round of C-MACONS is used to ensure key confirmation. For this purpose, as in [17], every participant computes a key confirmatory token $\mu_i = f_K(v_2)$ using a public input value v_2, signs it and sends it to other participants. After verifying signatures each party accepts with the session group key $\mathbf{K} = f_K(v_3)$ with public input value $v_3 \neq v_2$. All intermediate values are then erased.

Definition 8 (Compiler C-MACONS*):* Let P be a GKE protocol from Definition 1, $\pi : \{0,1\}^\kappa \rightarrow \{0,1\}^\kappa$ a permutation, $F := \{f_k\}_{k \in \{0,1\}^\kappa}$, $\kappa \in \mathbb{N}$ a function ensemble with domain and range $\{0,1\}^\kappa$, and $\Sigma :=$ (Gen, Sign, Verify) a digital signature scheme. A *compiler for MA-security and contributiveness in the strong corruption model*, denoted $\texttt{C-MACONS}_{\texttt{P}}$, consists of the algorithm INIT and a two-round protocol MACONS defined as follows:

- INIT: In the initialization phase each $U_i \in \mathcal{U}$ generates own private/public key pair (sk'_i, pk'_i) using $\Sigma.\texttt{Gen}(1^\kappa)$. This is in addition to any key pair (sk_i, pk_i) used in P.
- MACONS: After an oracle Π_i^s computes k_i^s in the execution of P it proceeds as follows.

Round 1: It chooses a random *MACON nonce* $r_i \in_R \{0,1\}^\kappa$ and sends $U_i|r_i$ to every oracle Π_j^s with $U_j \in \texttt{pid}_i^s$. After Π_i^s receives $U_j|r_j$ from Π_j^s with $U_j \in \texttt{pid}_i^s$ it checks whether $|r_j| \stackrel{?}{=} \kappa$. If this verification fails then Π_i^s terminates without accepting;

Round 2: Otherwise, after having received and verified these messages from all other partnered oracles it computes $\rho_1 := f_{k_i^s \oplus \pi(r_1)}(v_1)$ and each $\rho_l := f_{\rho_{l-1} \oplus \pi(r_l)}(v_1)$ for all $l \in \{2, \ldots, n\}$ where v_1 is a public value. Then, it defines the intermediate key $K_i^s := \rho_n$ and $\texttt{sid}_i^s := r_1|\ldots|r_n$ and computes a *MACON token* $\mu_i := f_{K_i^s}(v_2)$ where v_2 is a public value, together with a signature $\sigma_i := \Sigma.\texttt{Sign}(sk'_i, \mu_i|\texttt{sid}_i^s|\texttt{pid}_i^s)$. Then, it sends $U_i|\sigma_i$ to every oracle Π_j^s with $U_j \in \texttt{pid}_i^s$ and every other private information from $\texttt{state}_i^s$ (including k_i^s and each ρ_l, $l \in [1, n]$).

After Π_i^s receives $U_j|\sigma_j$ from Π_j^s with $U_j \in \texttt{pid}_i^s$ it checks whether $\Sigma.\texttt{Verify}(pk'_j, \mu_i|\texttt{sid}_i^s|\texttt{pid}_i^s, \sigma_j) \stackrel{?}{=} 1$. If this verification fails then Π_i^s terminates without accepting; otherwise it accepts with the session group key $\mathbf{K}_i^s := f_{K_i^s}(v_3)$ where $v_3 \neq v_2$ is another public value, and erases every other private information from $\texttt{state}_i^s$ (including K_i^s).

B. *Complexity of* C-MACONS

Obviously, C-MACONS requires two communication rounds. This is similar to the KS compiler [17] in case that no session ids are predefined and have to be negotiated first. Each participant must generate one digital signature and verify n signatures where n is the total number of session participants. This is also similar to the KS compiler. C-MACONS achieves contributiveness at an additional cost of n executions of the one-way permutation π and n executions of the pseudo-random function f per participant. Note that costs of XOR operations are usually omitted in the complexity analysis if public-key cryptography operations are present. Note also that pseudo-random functions can be realized using techniques of the symmetric cryptography massively reducing the required computational effort.

C. *Security Analysis of* C-MACONS

Let $\mathtt{P}$ be a GKE protocol from Definition 1. For this analysis we require Σ to be *existentially unforgeable under chosen message attacks* (EUF-CMA) [15], π to be *one-way*, and F to be *collision-resistant pseudo-random* [17].

Recall that we assume ephemeral secret information being independent of the long-lived key; that is, $\mathtt{state}_U^s$ may contain ephemeral secrets used in $\mathtt{P}$, the session key k_U^s computed in $\mathtt{P}$, and $\rho_1, \ldots, \rho_n$ together with some (implementation specific) temporary variables used to compute these values. Note that $\mathtt{state}_U^s$ is erased at the end of the protocol. By contrast, temporary data used by $\Sigma.\mathtt{Sign}(sk'_U, m)$ usually depends on the long-lived key and thus should be executed under the same protection mechanism as sk'_U, e.g., in a smart card [7][1]. Let q_{s} be the total number of executed protocol sessions during the attack.

The following theorem shows that $\mathtt{C\text{-}MACONS}_{\mathtt{P}}$ preserves the AKE-security with strong forward secrecy of the underlying protocol $\mathtt{P}$. Due to space limitations all the theorem proofs have been left out and will be available from the authors' websites.

Theorem 1 (AKE-Security of $\mathtt{C\text{-}MACONS}_{\mathtt{P}}$*):* For any AGKE-sfs protocol $\mathtt{P}$ if Σ is EUF-CMA and F is pseudo-random then $\mathtt{C\text{-}MACONS}_{\mathtt{P}}$ is also a AGKE-sfs protocol, and

$$\mathsf{Adv}^{\mathsf{ake}}_{\mathtt{sfs},\mathtt{C\text{-}MACONS}_{\mathtt{P}}}(\kappa) \quad \leq \quad 2N\mathsf{Succ}^{\mathtt{euf-cma}}_{\Sigma}(\kappa) + \frac{Nq_{\mathsf{s}}^2}{2^{\kappa-1}} + 2q_{\mathsf{s}}\mathsf{Adv}^{\mathsf{ake}}_{\mathtt{sfs},\mathtt{P}}(\kappa) + 2(N+2)q_{\mathsf{s}}\mathsf{Adv}^{\mathsf{prf}}_{F}(\kappa).$$

[1]Smart cards have limited resources. However, in C-MACONS each Π_U^s has to generate only one signature.

The following theorems concern the MA-security and the contributiveness of $\mathtt{C\text{-}MACONS}_{\mathtt{P}}$ in the presence of malicious participants and strong corruptions.

Theorem 2 (MA-Security of $\mathtt{C\text{-}MACONS}_{\mathtt{P}}$*):* For any GKE protocol $\mathtt{P}$ if Σ is EUF-CMA and F is collision-resistant then $\mathtt{C\text{-}MACONS}_{\mathtt{P}}$ is $\mathtt{MAGKE}$, and

$$\mathsf{Succ}^{\mathsf{ma}}_{\mathtt{C\text{-}MACONS}_{\mathtt{P}}}(\kappa) \quad\leq\quad N\mathsf{Succ}^{\mathtt{euf-cma}}_{\Sigma}(\kappa) + \frac{Nq_{\mathsf{s}}^2}{2^\kappa} + q_{\mathsf{s}}\mathsf{Succ}^{\mathsf{coll}}_{F}(\kappa).$$

Theorem 3 (Contributiveness of $\mathtt{C\text{-}MACONS}_{\mathtt{P}}$*):* For any GKE protocol $\mathtt{P}$ if π is one-way and F is collision-resistant pseudo-random then $\mathtt{C\text{-}MACONS}_{\mathtt{P}}$ is $\mathtt{sCGKE}$, and

$$\mathsf{Succ}^{\mathsf{con}}_{\mathtt{C\text{-}MACONS}_{\mathtt{P}}}(\kappa) \quad\leq\quad \frac{Nq_{\mathsf{s}}^2 + Nq_{\mathsf{s}} + 2q_{\mathsf{s}}}{2^\kappa} + (N+2)q_{\mathsf{s}}\mathsf{Succ}^{\mathsf{coll}}_{F}(\kappa) + q_{\mathsf{s}}\mathsf{Adv}^{\mathsf{prf}}_{F}(\kappa) + Nq_{\mathsf{s}}\mathsf{Succ}^{\mathsf{ow}}_{\pi}(\kappa).$$

Remark 1: Note that the contributiveness of $\mathtt{C\text{-}MACONS}_{\mathtt{P}}$ depends neither on AKE-security of $\mathtt{P}$ nor on the security of the digital signature scheme Σ. Hence our compiler can also be used for unauthenticated GKE protocols by omitting digital signatures of exchanged messages. However, in this case it would guarantee only contributiveness but not MA-security in the presence of malicious participants. The latter can be only guaranteed using digital signatures (as also noticed in [17] for their definition of security against insider attacks). Note also that $\mathtt{C\text{-}MACONS}_{\mathtt{P}}$ provides contributiveness in some even stronger sense than required in Definition 6, i.e., $\mathcal{A}$ may even be allowed to output $\tilde{\mathbf{K}}$ before the uncorrupted user's oracle Π_U^s (that is supposed to accept with $\tilde{\mathbf{K}}$ in $\mathsf{Game}^{\mathsf{con}}_{\mathtt{C\text{-}MACONS}_{\mathtt{P}}}(\mathcal{A},\kappa)$) starts with the $\mathtt{MACONS}$ protocol of the compiler, and not necessarily before the execution of the new $\mathtt{C\text{-}MACONS}_{\mathtt{P}}$ session.

VI. Compiler C-MACONW

In this section we slightly modify C-MACONS to obtain a compiler C-MACONW which provides MA-security and contributiveness for any AKE-secure GKE protocol in the weak corruption model.

A. *Main Differences to* C-MACONS

The main difference to C-MACONS is that every participant uses own random nonce r_i as a seed for the pseudo-random function f to compute the PRF commitment c_i which is sent to all other participants in the first communication round. Upon receiving PRF commitments from other participants the random nonce will be revealed such that after the verification of PRF commitments C-MACONW proceeds similar to C-MACONS. Intuitively, pseudo-randomness and collision-resistance of f ensure that malicious participants chose own nonces before they learn nonces chosen by the honest participants, which will then be used to derive the session group key. Further differences are: (i) the computation of the one-way permutation π on chosen MACON nonces are omitted, and (ii) due to the absence of *RevealState* queries no erasure of internal state information is necessary.

Definition 9 (Compiler C-MACONW*):* Let P be a GKE protocol from Definition 1, $\pi : \{0,1\}^\kappa \rightarrow \{0,1\}^\kappa$ a permutation, $F := \{f_k\}_{k \in \{0,1\}^\kappa}$, $\kappa \in \mathbb{N}$ a function ensemble with domain and range $\{0,1\}^\kappa$, and $\Sigma := (\mathtt{Gen}, \mathtt{Sign}, \mathtt{Verify})$ a digital signature scheme. A *compiler for MA-security and contributiveness in the weak corruption model*, denoted $\mathtt{C\text{-}MACONW}_{\mathtt{P}}$, consists of the algorithm INIT and a three-round protocol MACONW defined as follows:

- INIT: In the initialization phase each $U_i \in \mathcal{U}$ generates own private/public key pair (sk'_i, pk'_i) using $\Sigma.\mathtt{Gen}(1^\kappa)$. This is in addition to any key pair (sk_i, pk_i) used in P.
- MACONW: After an oracle Π_i^s computes k_i^s in the execution of P it proceeds as follows.

Round 1: It chooses a random *MACON nonce* $r_i \in_R \{0,1\}^\kappa$, computes the *PRF commitment* $c_i := f_{r_i}(v_0)$ where v_0 is a public value, and sends $U_i|c_i$ to every oracle Π_j^s with $U_j \in \mathtt{pid}_i^s$.

Round 2: After having received these messages from all other participating oracles it sends $U_i|r_i$ to every oracle Π_j^s with $U_j \in \mathtt{pid}_i^s$. After Π_i^s receives $U_j|r_j$ from Π_j^s with $U_j \in \mathtt{pid}_i^s$ it checks whether $c_j \mathrel{?=} f_{r_j}(v_0)$ and $|r_j| \mathrel{?=} \kappa$. If these verifications fail then Π_i^s terminates without accepting.

Round 3: Otherwise, after having received and verified these messages from all other oracles it computes $\rho_1 := f_{k_i^s \oplus r_1}(v_1)$ and each $\rho_l := f_{\rho_{l-1} \oplus r_l}(v_1)$ for all $l \in \{2, \ldots, n\}$ where v_1 is a public value. Then, it defines the intermediate key $K_i^s := \rho_n$ and $\mathtt{sid}_i^s := r_1|\ldots|r_n$ and computes a *MACON token* $\mu_i := f_{K_i^s}(v_2)$ where v_2 is a public value, together with a signature $\sigma_i := \Sigma.\mathtt{Sign}(sk'_i, \mu_i|\mathtt{sid}_i^s|\mathtt{pid}_i^s)$. Then, it sends $U_i|\sigma_i$ to every oracle Π_j^s with $U_j \in \mathtt{pid}_i^s$.

After Π_i^s receives $U_j|\sigma_j$ from Π_j^s with $U_j \in \mathtt{pid}_i^s$ it checks whether $\Sigma.\mathtt{Verify}(pk'_j, \mu_i|\mathtt{sid}_i^s|\mathtt{pid}_i^s, \sigma_j) \mathrel{?=} 1$. If this verification fails then Π_i^s terminates without accepting; otherwise it accepts with the session group key $\mathbf{K}_i^s := f_{K_i^s}(v_3)$ where $v_3 \neq v_2$ is another public value.

B. Complexity of C-MACONW

C-MACONW requires three communication rounds. Each participant must generate one digital signature and verify n signatures where n is the total number of session participants. Furthermore, C-MACONW requires $2n+1$ executions of the pseudo-random function f per participant.

C. Security Analysis of C-MACONW

Security analysis of C-MACONW is widely similar to that of C-MACONS. Especially the requirements of AKE- and MA-security require only minor additional modifications.

Theorem 4 (AKE-Security of $\mathtt{C\text{-}MACONW}_{\mathtt{P}}$*):* For any AGKE-wfs protocol P if Σ is EUF-CMA and F is collision-resistant pseudo-random then $\mathtt{C\text{-}MACONW}_{\mathtt{P}}$ is also a AGKE-wfs protocol, and

$$\begin{aligned}\mathsf{Adv}^{\mathsf{ake}}_{\mathtt{wfs},\mathtt{C\text{-}MACONW}_{\mathtt{P}}}(\kappa) \quad &\leq \quad 2N\mathsf{Succ}^{\mathtt{euf-cma}}_{\Sigma}(\kappa) + \frac{Nq_{\mathsf{s}}^2}{2^{\kappa-1}} + 2Nq_{\mathsf{s}}\mathsf{Succ}^{\mathsf{coll}}_{F}(\kappa) + \\ &\quad +2q_{\mathsf{s}}\mathsf{Adv}^{\mathsf{ake}}_{\mathtt{wfs},\mathtt{P}}(\kappa) + 2(N+2)q_{\mathsf{s}}\mathsf{Adv}^{\mathsf{prf}}_{F}(\kappa).\end{aligned}$$

Theorem 5 (MA-Security of $\mathtt{C\text{-}MACONW}_{\mathtt{P}}$*):* For any GKE protocol P if Σ is EUF-CMA and F is collision-resistant then $\mathtt{C\text{-}MACONW}_{\mathtt{P}}$ is MAGKE, and

$$\mathsf{Succ}^{\mathsf{ma}}_{\mathtt{C\text{-}MACONW}_{\mathtt{P}}}(\kappa) \quad \leq \quad N\mathsf{Succ}^{\mathtt{euf-cma}}_{\Sigma}(\kappa) + \frac{Nq_{\mathsf{s}}^2}{2^{\kappa}} + q_{\mathsf{s}}\mathsf{Succ}^{\mathsf{coll}}_{F}(\kappa).$$

Theorem 6 (Contributiveness of $\mathtt{C\text{-}MACONW}_{\mathtt{P}}$*):* For any GKE protocol P if F is collision-resistant pseudo-random then $\mathtt{C\text{-}MACONW}_{\mathtt{P}}$ is wCGKE, and

$$\mathsf{Adv}^{\mathsf{con}}_{\mathtt{C\text{-}MACONW}_{\mathtt{P}}}(\kappa) \quad \leq \quad \frac{Nq_{\mathsf{s}}^2}{2^{\kappa-1}} + 2Nq_{\mathsf{s}}\mathsf{Succ}^{\mathsf{coll}}_{F}(\kappa) + 2(N+1)q_{\mathsf{s}}\mathsf{Adv}^{\mathsf{prf}}_{F}(\kappa).$$

VII. Conclusion

In this paper we have addressed the main difference in the trust relationship between participants of group key exchange (GKE) and whose of group key transport (GKT) protocols, namely, the question of key control and contributiveness. This has been done from the perspective of malicious participants and powerful adversaries who are able to reveal the internal memory of honest participants. The proposed security model based on the extension of the well-known notion of AKE-security with strong forward secrecy from [7] towards additional requirements of MA-security and contributiveness seems to be stronger than the previous models for group key exchange protocols that address similar issues. The described compilers C-MACONS and C-MACONW satisfy these additional security requirements and extend the list of currently known compilers for GKE protocols, i.e., the compiler for AKE-security by Katz and Yung [18] and the compiler for security against "insider attacks" by Katz and Shin [17] (that according to our model provides MA-security but not contributiveness).

References

[1] Giuseppe Ateniese, Michael Steiner, and Gene Tsudik. Authenticated Group Key Agreement and Friends. In *Proc. of the 5th ACM Conf. on Computer and Communications Security (CCS'98)*, pages 17–26. ACM Press, 1998.

[2] Mihir Bellare and Phillip Rogaway. Entity Authentication and Key Distribution. In *Advances in Cryptology–CRYPTO'93*, volume 773 of *LNCS*, pages 232–249. Springer, 1993.

[3] Mihir Bellare and Phillip Rogaway. Provably Secure Session Key Distribution: The Three Party Case. In *Proc. of the Twenty-Seventh Annual ACM Symposium on Theory of Computing (STOC'95)*, pages 57–66. ACM Press, 1995.

[4] Jens-Matthias Bohli, Maria Isabel Gonzalez Vasco, and Rainer Steinwandt. Secure Group Key Establishment Revisited. Cryptology ePrint Archive, Report 2005/395, 2005. http://eprint.iacr.org/.

[5] Colin Boyd and Anish Mathuria. *Protocols for Authentication and Key Establishment*. Springer, 2003. ISBN:3-540-43107-1.

[6] Emmanuel Bresson and Dario Catalano. Constant Round Authenticated Group Key Agreement via Distributed Computation. In *Proc. of the 7th Intl. Workshop on Theory and Practice in Public Key Cryptography (PKC'04)*, volume 2947 of *LNCS*, pages 115–129. Springer, 2004.

[7] 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.

[8] Emmanuel Bresson, Olivier Chevassut, David Pointcheval, and Jean-Jacques Quisquater. Provably Authenticated Group Diffie-Hellman Key Exchange. In *Proc. of the 8th ACM Conf. on Computer and Communications Security (CCS'01)*, pages 255–264. ACM Press, 2001.

[9] M. Burmester. On the Risk of Opening Distributed Keys. In *Advances in Cryptology – CRYPTO'94*, volume 839 of *LNCS*, pages 308–317. Springer, August 1994.

[10] M. Burmester and Y. Desmedt. A Secure and Efficient Conference Key Distribution System. In *Advances in Cryptology – EUROCRYPT'94*, volume 950 of *LNCS*, pages 275–286. Springer, May 1994.

[11] Kim-Kwang Raymond Choo, Colin Boyd, and Yvonne Hitchcock. Examining Indistinguishability-Based Proof Models for Key Establishment Protocols. In *Advances in Cryptology – ASIACRYPT'05*, volume 3788 of *LNCS*, pages 585–604. Springer, 2005.

[12] Richard Cleve. Limits on the Security of Coin Flips When Half the Processors are Faulty. In *Proceedings of the 18th ACM Symposium on Theory of Computing (STOC'86)*, pages 364–369. ACM Press, 1986.

[13] Yvo Desmedt, Josef Pieprzyk, Ron Steinfeld, and Huaxiong Wang. A Non-malleable Group Key Exchange Protocol Robust Against Active Insiders. In *Proc. of the 9th Intl. Conf. on Information Security (ISC 2006)*, volume 4176 of *LNCS*, pages 459–475. Springer, 2006.

[14] Whitfield Diffie, Paul C. van Oorschot, and Michael J. Wiener. Authentication and Authenticated Key Exchanges. *Designs, Codes and Cryptography*, 2(2):107–125, 1992.

[15] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. *SIAM Journal of Computing*, 17(2):281–308, 1988.

[16] Christoph G. Günther. An Identity-Based Key-Exchange Protocol. In *Advances in Cryptology – EUROCRYPT'89*, volume 434 of *LNCS*, pages 29–37. Springer, 1990.

[17] Jonathan Katz and Ji Sun Shin. Modeling Insider Attacks on Group Key-Exchange Protocols. In *Proc. of the 12th ACM Conf. on Computer and Communications Security (CCS'05)*, pages 180–189. ACM Press, 2005.

[18] 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.

[19] 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, 2004.

[20] Yongdae Kim, Adrian Perrig, and Gene Tsudik. Simple and Fault-Tolerant Key Agreement for Dynamic Collaborative Groups. In *Proc. of the 7th ACM Conf. on Computer and Communications Security (CCS'00)*, pages 235–244. ACM Press, 2000.

[21] Hugo Krawczyk. HMQV: A High-Performance Secure Diffie-Hellman Protocol. In *Advances in Cryptology – CRYPTO'05*, volume 3621 of *LNCS*, pages 546–566. Springer, 2005.

[22] Mark Manulis. Survey on Security Requirements and Models for Group Key Exchange. Technical Report 2006/02, Horst-Görtz Institute, Network and Data Security Group, November 2006. also available at `http://eprint.iacr.org/2006/388`.

[23] Alfred Menezes, P. van Oorschot, and Scott Vanstone. *Handbook of Applied Cryptography*. CRC Press, October 1996.

[24] C. J. Mitchell, Mike Ward, and Piers Wilson. Key Control in Key Agreement Protocols. *Electronic Letters*, 34(10):980–981, 1998.

[25] Victor Shoup. On Formal Models for Secure Key Exchange (Version 4). Technical Report RZ 3120, IBM Research, November 1999. Also available at `http://shoup.net/`.

[26] Yacov Yacobi and Zahava Shmuely. On Key Distribution Systems. In *Advances in Cryptology – CRYPTO'89*, volume 435 of *LNCS*, pages 344–355. Springer, August 1990.