

On Security Models and Compilers for Group Key Exchange Protocols

[Extended abstract]*

Emmanuel Bresson¹ and Mark Manulis^{2**} and Jörg Schwenk²

¹ DCSSI Crypto Lab Paris —

emmanuel.bresson@polytechnique.org emmanuel@bresson.org

² Horst Görtz Institute, Ruhr University Bochum, Germany

mark.manulis@nds.rub.de joerg.schwenk@nds.rub.de

Abstract. Group key exchange (GKE) protocols can be used to guarantee confidentiality and authentication in group applications. The paradigm of provable security subsumes an abstract formalization (security model) that considers the protocol environment and identifies its security goals. The first security model for GKE protocols was proposed by Bresson, Chevassut, Pointcheval, and Quisquater in 2001, and has been subsequently applied in many security proofs. Their definitions of AKE-security (authenticated key exchange; a.k.a. indistinguishability of the key) and MA-security (mutual authentication) became meanwhile standard.

In this paper we analyze the BCPQ model and some of its variants and identify several risks resulting from its technical core construction – the notion of *partnering*. Consequently, we propose a revised model extending AKE- and MA-security in order to capture attacks by malicious participants and strong corruptions.

Then, we turn to generic solutions (known as *compilers*) for AKE- and MA-security in BCPQ-like models. We describe a compiler **C-AMA** which provides AKE- and MA-security for any GKE protocol, under standard cryptographic assumptions, that eliminates some identified limitations in existing compilers.

1 Introduction

MOTIVATION. Security of many privacy-preserving multi-party applications like encrypted group communication for audio/video conferences, chat systems, computer-supported collaborative workflow systems, or secure server replication systems depends on group key exchange (GKE) protocols. Security of those latter is therefore critical. Security of earlier GKE protocols [29, 41, 17, 42, 2, 38, 33, 34] has been analyzed heuristically based on informal definitions so that some of

* A full version of this work is available at <http://eprint.iacr.org/2006/385/>

** The author was supported by the European Commission (under contract IST-2002-507932 ECRYPT).

them have been broken later, e.g., [37]. By contrast, the paradigm of *provable security* is used across the modern literature to prove in a mathematical way, and under reasonable assumptions, that a cryptographic scheme achieves the required security goals. Such a proof usually uses a formal setting that specifies: (1) the computing environment (involved users, their trust relationship, cryptographic parameters, communication. . .), (2) the adversarial environment and (3) the definitions of some concrete security goals. In 2001 Bresson, Chevassut, Pointcheval, and Quisquater [14] introduced the first security model (BCPQ model) designed for GKE protocols. They adopted some ideas previously proposed by Bellare and Rogaway [5, 7] in the context of two- and three-party key establishment protocols. The BCPQ model, as well as its refinements [11, 12] and variants [31, 13, 27, 30], have been meanwhile used in many GKE security proofs including [14, 11, 12, 13, 31, 10, 32, 27, 26, 30, 1] and became *de facto* standard.

MODULARITY OF PROTOCOL DESIGN. Seeing GKE protocols as building blocks for high-level applications, it is worth designing protocols that have specific security goals. Modular design allows to build such “*à la carte*” protocols. In order to provide these modular constructions in a generic way, so-called “compilers” have been developed, e.g., [31, 30]. They allow designers to enhance security of a protocol in a *black-box* manner, that is, independently of the implementation of the protocol being enhanced.

CONTRIBUTIONS AND ORGANIZATION. We start with a brief overview of the BCPQ; in Section 2.1 we point out a problem in the BCPQ model between its technical core – the notion of *partnering* – and its definition of MA-security. Next in Section 2.2 we analyze some variants of the BCPQ model, from the perspective of MA-security, in particular in presence of *malicious participants*. By malicious participants we mean legitimate protocol participants who are fully controlled by the adversary. We emphasize that consideration of malicious participants makes sense in the scope of MA-security but not of AKE-security.

After identifying some drawbacks in the mentioned variants we describe in Section 3 an extended security model for revised definitions of AKE- and MA-security with consideration of malicious participants. Our model is based on the more powerful BCPQ refinement from [12] that considers AKE-security in the presence of (partial) internal state corruptions (strong corruptions). We also introduce an additional notion of *backward secrecy* which leads to new corruption models in case of AKE-security.

In Section 4.2 we provide a brief analysis of some known security-enhancing compilers for GKE protocols. In particular, we show that the compiler proposed by Katz and Yung in [31] needs some additional assumptions in order to be considered as a really generic solution. To overcome this, and in order to show that our extended security definitions are feasible enough for the construction of practical reductionist security proofs, we describe in Section 4.3 a compiler C-AMA that satisfies our *stronger* definitions of AKE- and MA-security. We further prove its security under standard cryptographic assumptions.

2 The BCPQ Model and its Variants

The BCPQ model extends the methodology introduced by Bellare and Rogaway [6, 7] to a group setting. The model considers a set ID of n participants. Each $U_i \in ID$ has a long-lived key LL_i and an unlimited number of instances called *oracles* and denoted $\Pi_i^{s_i}$ (s_i -th instance of U_i) involved in different executions of P . The BCPQ model uses session ids to define a “*partnering*” relation which is used to define security goals. The session id of an oracle Π_i^s is defined as $SID(\Pi_i^s) := \{SID_{ij} \mid U_j \in ID\}$ where SID_{ij} is the concatenation of all flows that Π_i^s exchanges with any oracle Π_j^t . Then two oracles Π and Π' are called *directly partnered*, denoted $\Pi \leftrightarrow \Pi'$, if both oracles *accept* (compute the session key) and if $SID(\Pi) \cap SID(\Pi') \neq \emptyset$. Further, oracles Π and Π' are *partnered*, denoted $\Pi \rightsquigarrow \Pi'$, if, in the graph $G_{SIDS} := (V, E)$ with $V := \{\Pi_l^s \mid l \in [1, n], s \in \mathbb{N}\}$ and $E := \{(\Pi_i, \Pi_j) \mid \Pi_i \leftrightarrow \Pi_j\}$, there exists a sequence of oracles $(\Pi_1, \Pi_2, \dots, \Pi_k)$ with $\Pi = \Pi_1$, $\Pi' = \Pi_k$, and $\Pi_{l-1} \leftrightarrow \Pi_l$ for all $l \in [2, k]$. The *partner id* for an oracle Π_i^s is $PIDS(\Pi_i^s) = \{\Pi_l^t \mid \Pi_i^s \rightsquigarrow \Pi_l^t \forall l \in [1, n] \setminus \{i\}\}$.

The BCPQ model considers a Probabilistic Polynomial-Time (PPT) active adversary \mathcal{A} which can send messages to the oracles via a **Send** query, reveal the session key via a **Reveal** query, obtain a long-lived key via a **Corrupt** query, and ask a **Test** query to obtain either a session key or a random number. Using this adversarial setting two security goals are specified for a GKE protocol: AKE-security and MA-security.

The AKE-security requires that adversary \mathcal{A} asks a single **Test** query to a fresh oracle. An oracle $\Pi_i^{s_i}$ is *fresh* if (1) it has accepted, (2) no oracle has been asked for a **Corrupt** query before $\Pi_i^{s_i}$ accepts, and (3) neither $\Pi_i^{s_i}$ nor any of its partners have been asked for a **Reveal** query. A GKE protocol is said to be AKE-secure if \mathcal{A} cannot guess which value it has received in response to its **Test** query, i.e., the session key or a random number, significantly better than at random. This definition of AKE-security encompass earlier informal definitions:

- *key secrecy* [24] (a.k.a. *implicit key authentication* [36]) which ensures that no other party except for legitimate participants learns the established group key;
- *resistance against known-key attacks* [43, 16] (a.k.a. *key independence* [33]) meaning that an adversary knowing group keys of other sessions cannot compute subsequent session keys;
- *perfect forward secrecy* [28, 24, 36] requiring that the disclosure of long-term keying material must not compromise the secrecy of the established keys from earlier protocol runs.

The MA-security means that \mathcal{A} cannot impersonate a participant U_i through its oracle $\Pi_i^{s_i}$. Impersonation would imply that there exists one oracle $\Pi_i^{s_i}$ which accepts with $|PIDS(\Pi_i^{s_i})| \neq n - 1$ must be negligible. In other words, if each oracle $\Pi_i^{s_i}$ accepts with $|PIDS(\Pi_i^{s_i})| = n - 1$ then no impersonation attacks could have occurred — thus the informal notion of *mutual authentication* [6] is satisfied. Further, we recall the following claims given by the authors of [14]:

“In the definition of partnering, we do not require that the session key computed by partnered oracles be the same since it can easily be proven that the probability that **partnered** oracles come up with different session keys is negligible.” [14, Footnote 3]

“We are not concerned with partnered oracles coming up with different session keys, since our definition of partnering implies the oracles have exchanged *exactly* the same flows.” [14, Section 7.4]

If these claims hold then MA-security captures the following informal earlier security goals:

- *key confirmation*: all participants that have accepted¹ hold identical group keys;
- *explicit key authentication* [36]: both key confirmation and mutual authentication.

2.1 Problems with the Definition of MA-Security in the BCPQ Model

In this section, we explain why the definition of MA-security might not be general enough for GKE protocols. We do not pretend having broken some provably MA-secure scheme. In contrast, we explain why, if every participating oracle $\Pi_i^{s_i}$ accepts with $|\text{PIDS}(\Pi_i^{s_i})| = n - 1$, it does not necessarily mean that the considered protocol provides mutual authentication and key confirmation:

1. There exist GKE protocols where an active adversary \mathcal{A} can impersonate one of the participants through its oracle but nevertheless every participating oracle $\Pi_i^{s_i}$ accepts with $|\text{PIDS}(\Pi_i^{s_i})| = n - 1$.
2. There exist GKE protocols where each oracle $\Pi_i^{s_i}$ accepts with $|\text{PIDS}(\Pi_i^{s_i})| = n - 1$ but there are at least two partnered oracles that have computed different keys.

We start from the protocols presented in [14, 11, 12, 13]. We first study the case at an abstract level. On Figure 1, we show an execution with 3 participants in which \mathcal{A} impersonates U_1 and modifies message m_1 to \tilde{m}_1 (Figure 1) such that $\text{SID}_{21} = \tilde{m}_1$ (Figure 2 for the construction of sessions ids).

We may assume that there exists protocols for which all oracles accept after this modification. Here every oracle $\Pi_i^{s_i}$ accepts with $|\text{PIDS}(\Pi_i^{s_i})| = 2$. To check this we first note that $\text{SID}_{12} = m_1$. Though $\text{SID}(\Pi_1^{s_1}) \cap \text{SID}(\Pi_2^{s_2}) = \{m_1, m_3\} \cap \{\tilde{m}_1, m_2 | m_3\} = \emptyset$ and thus $\Pi_1^{s_1} \not\rightsquigarrow \Pi_2^{s_2}$, we still have $\text{SID}(\Pi_1^{s_1}) \cap \text{SID}(\Pi_3^{s_3}) = \{m_1, m_3\} \cap \{m_3, m_2 | m_3\} = m_3$ and $\text{SID}(\Pi_3^{s_3}) \cap \text{SID}(\Pi_2^{s_2}) = \{m_3, m_2 | m_3\} \cap \{\tilde{m}_1, m_2 | m_3\} = m_2 | m_3$ so that $\Pi_1^{s_1} \rightsquigarrow \Pi_2^{s_2}$: all oracles are still partnered despite the impersonation attack. However, oracle $\Pi_2^{s_2}$ has received a modified message and this may result in different keys computed by $\Pi_1^{s_1}$ and $\Pi_2^{s_2}$.

¹ This is a slightly modified definition from [36] wrt. to the arguments from [39] on impossibility of the assurance of some participant that other participants have actually accepted the key.

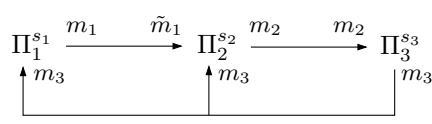


Fig. 1. Protocol execution where \mathcal{A} impersonates U_1

$\text{SID}(\Pi_i^{s_i})$	SID_{i1}	SID_{i2}	SID_{i3}
$\text{SID}(\Pi_1^{s_1})$	\emptyset	m_1	m_3
$\text{SID}(\Pi_2^{s_2})$	\tilde{m}_1	\emptyset	$m_2 m_3$
$\text{SID}(\Pi_3^{s_3})$	m_3	$m_2 m_3$	\emptyset

Fig. 2. $\text{SID}(\Pi_i^{s_i})$ in the protocol execution with impersonation of U_1

CONCRETE EXAMPLE. As a concrete example, we illustrate how a replay attack would work on the protocol from [14] if the additional confirmation round is missing. On Figure 3, $[m]_{U_i}$ denotes a message m signed by $\Pi_i^{s_i}$, and $V(m) \stackrel{?}{=} 1$ its verification; g is a generator of a finite cyclic group.

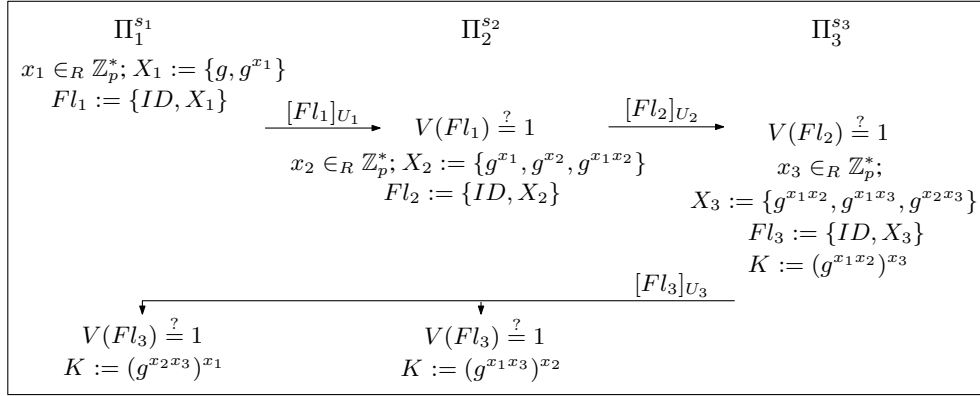


Fig. 3. Execution of the protocol in [14] with three participants

Here \mathcal{A} drops the flow $[Fl_1]_{U_1}$ and replays a previously sent message. The replayed message is likely to be $[Fl_1]_{U_1}$ with $\widetilde{Fl}_1 := (ID, \{g, g^{\widetilde{x}_1}\})$ for some $\widetilde{x}_1 \neq x_1$. Obviously, $V(\widetilde{Fl}_1) = 1$ holds. It is easy to see that $X_2 = \{g^{\widetilde{x}_1}, g^{x_2}, g^{\widetilde{x}_1 x_2}\}$ and $X_3 := \{g^{\widetilde{x}_1 x_2}, g^{\widetilde{x}_1 x_3}, g^{x_2 x_3}\}$ so that $\Pi_1^{s_1}$ computes $K = g^{x_1 x_2 x_3}$ whereas $\Pi_2^{s_2}$ and $\Pi_3^{s_3}$ compute another value, i.e., $K = g^{\widetilde{x}_1 x_2 x_3}$. Moreover, it is easy to check that $|\text{PIDS}(\Pi_i^{s_i})| = 2$ for every $\Pi_i^{s_i}$, $i \in \{1, 2, 3\}$.

Thus $|\text{PIDS}(\Pi_i^{s_i})| = n - 1$ does not ensure mutual authentication and key confirmation. We stress that this does not contradict the MA-security of the proposed protocol when the additional round is executed. However, it is worth studying the notion of MA-security independently from some concrete protocol design.

Furthermore, we stress that a more general definition of MA-security should also consider attacks by malicious protocol participants (AKE-security in such setting is hopeless). As noted in [21], this is the reason the BCPQ model fails

to provide security against *unknown key-share attacks* [9]. The same remark is done in [20] for the protocol from [10].

2.2 MA-Security in some BCPQ-Variants

VARIANTS BY KATZ-YUNG AND DUTTA *et al.* These are two modifications [31,27] to the BCPQ model, that propose a different construction of partnering. However neither of them propose a specific definition of MA-security, and they just refer to BCPQ.

A VARIANT BY KATZ AND SHIN. Katz and Shin [30] proposed a different security model (KS) for GKE protocols, and provide a security analysis in the framework of Universal Composability (UC) [18]. This provides the first formal treatment of GKE protocols security in the presence of malicious participants. Here unique session ids are assumed to be provided by some high-level application mechanism [3], and partner ids is just the set of users whose oracle intend to establish a key. In addition to their model, Katz and Shin proposed a compiler to turn any GKE protocol which is secure in the BCPQ model into a protocol which is secure in the KS model.

Among other things, the KS model defines *security against insider attacks* as a combination of two requirements: *agreement* and *security against insider impersonation attacks*:

- the adversary \mathcal{A} *violates agreement* if two partnered, uncorrupted oracles accept with different session keys.
- the adversary \mathcal{A} *impersonates uncorrupted U_j to accepting Π_i^s* if U_j belongs to the (expected) partner id of Π_i^s but in fact no oracle Π_j^t is partnered with Π_i^s . In other words, the instance Π_i^s computes the session key and U_i believes that U_j does so, but in fact an adversary has participated in the protocol on behalf of U_j .

Intuitively, agreement considers key confirmation in case that all other participants are malicious (corrupted); and security against insider impersonation attacks considers mutual authentication and unknown key-share resilience in the presence of malicious participants.

EXTENSIONS BY BOHLI *et al.* AND DESMEDT *et al.* These extensions of the BCPQ model takes consideration of attacks whose goal is to “bias” the value of the group key: Bohli *et al.* [8] consider malicious participants who *deliberately* wish to influence the key. Desmedt *et al.* [22] formalized a related notion called *non-malleability*. In our parallel work [15] we propose a generic compiler C-MACON which addresses these security requirements from the perspective of malicious participants.

3 The Revised “Game-based” Security Model

In the following we extend the BCPQ model under consideration of malicious participants. We revisit the definition of MA-security, so that the unified definition can replace both definitions of agreement and security against insider impersonation attacks of the KS model.

Our refinements are in two directions: *strong corruptions* (in the sense of [12]), and a new requirement which we call *backward secrecy*.

3.1 Protocol Participants, Variables

USERS, INSTANCE ORACLES. We consider \mathcal{U} as a set of N users in the universe. Each user $U_i \in \mathcal{U}$ holds a long-lived key LL_i . In order to handle participation of U_i in distinct concurrent protocol executions we consider that U_i has an unlimited number of instances called *oracles*; Π_i^s , with $s \in \mathbb{N}$, denotes the s -th instance oracle of U_i .

INTERNAL STATES. Every Π_U^s maintains an *internal state information* \mathbf{state}_U^s which is composed of all private, 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).

SESSION GROUP KEY, SESSION ID, PARTNER ID. In each session we consider a new group \mathcal{G} of $n \in [1, N]$ participating oracles. Each oracle in \mathcal{G} is called a *group member*. Every participating oracle $\Pi_U^s \in \mathcal{G}$ computes the *session group key* $k_U^s \in \{0, 1\}^\kappa$. Every session is identified by a unique *session id* \mathbf{sid}_U^s known to all oracles in the session. Similarly, each oracle $\Pi_U^s \in \mathcal{G}$ gets a value \mathbf{pid}_U^s that contains the identities of participating users (including U). We say that two oracles, $\Pi_i^{s_i}$ and $\Pi_j^{s_j}$, are *partnered* if $U_i \in \mathbf{pid}_j^{s_j}$, $U_j \in \mathbf{pid}_i^{s_i}$, and $\mathbf{sid}_i^{s_i} = \mathbf{sid}_j^{s_j}$.

INSTANCE ORACLE STATES. An oracle Π_U^s is originally *unused* and becomes *used*, initialized with LL_U , when it becomes part of a group \mathcal{G} . Then it turns into the *stand-by* state where it waits for an invocation to execute the protocol. Upon invocation, the oracle Π_U^s learns its partner id \mathbf{pid}_U^s (and possibly \mathbf{sid}_U^s) and turns into a *processing* state where it sends, receives and processes messages. During this phase, the internal state \mathbf{state}_U^s is maintained by the oracle. When oracle Π_U^s has enough information to compute the session key, it *accepts*. After some possible auxiliary steps, it *terminates* meaning that it would not send or receive further messages. If the execution fails (due to any adversarial actions) then Π_U^s terminates without having accepted, and the key k_U^s is set to some *undefined* value.

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

$P.\mathbf{KeyGen}(1^\kappa)$: On input a security parameter 1^κ each user in \mathcal{U} is provided with a long-lived key LL_U .

$P.\text{Setup}(\mathcal{S})$: On input a set \mathcal{S} of n unused oracles a new group $\mathcal{G} := \mathcal{S}$ is created and a probabilistic interactive protocol is executed between oracles in \mathcal{G} .

We call $P.\text{Setup}$ an *operation*. We say that a protocol is *correct* if all oracles in \mathcal{G} accept with the same group key k and assume it is always the case in this paper.

3.2 Adversarial Model

QUERIES TO THE INSTANCE ORACLES. The adversary \mathcal{A} (passive or active) is represented by a PPT machine and may interact with group members by making the following oracle queries:

- $\text{Setup}(\mathcal{S})$: This query models \mathcal{A} eavesdropping an honest operation execution. $P.\text{Setup}(\mathcal{S})$ is executed and \mathcal{A} is given the transcript of the execution.
- $\text{Send}(\Pi_U^s, m)$: 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 . When asking $\text{Send}(\Pi_U^s, \mathcal{S})$, \mathcal{A} gets the first message of the protocol.
- $\text{RevealKey}(\Pi_U^s)$: \mathcal{A} is given the session group key k_U^s , provided Π_U^s has accepted.
- $\text{RevealState}(\Pi_U^s)$: \mathcal{A} is given the internal state information state_U^s .²
- $\text{Corrupt}(U)$: \mathcal{A} is given the long-lived key LL_U .
- $\text{Test}(\Pi_U^s)$: This query will be used to model the AKE-security of a GKE protocol. It can be asked at any time, but only once. It is answered as follows: the oracle generates a random bit b . If $b = 1$ then \mathcal{A} is given k_U^s , and if $b = 0$ then \mathcal{A} is given a random string.

In spirit of [19], we restrict a *passive* adversary not to replay, modify or inject messages; but, it can still change their delivery order, drop or delay them.

FORWARD SECRECY. A completed session is “*forward-secure*” if its security remains whatever the adversary does in the future. Similar to [12] we distinguish between *weak-forward secrecy* (**wfs**) where \mathcal{A} is allowed to ask Corrupt queries, and *strong-forward secrecy* (**sfs**) where it can also ask RevealState queries.

BACKWARD SECRECY. The notion of backward secrecy is symmetric to forward secrecy in the sense that it considers damages to the AKE-security of future sessions after actions of the adversary in the past/current sessions. The notion might seem useless at first glance (such actions can make secrecy just impossible), however, there might exist intermediate actions, such as corruptions of internal states, that do not compromise future session keys (or at least not all of them). We distinguish between *weak-backward secrecy* (**wbs**) where \mathcal{A} is allowed to ask

² This kind of the adversarial query has previously been mentioned by Canetti and Krawczyk in their model for two-party protocols [19].

RevealState queries, and *strong-backward secrecy* (**sbs**) where it can also ask Corrupt queries³.

ORACLE FRESHNESS, CORRUPTION MODELS, ADVERSARIAL SETTINGS. To properly deal with forward and backward secrecy, we need to distinguish cases where the adversary may have participate in the protocol. In order to consider only non-participating adversaries we introduce the following notion of α -fresh sessions.

The notion of freshness for an oracle Π_U^s is needed to distinguish between various definitions of security wrt. different flavors of backward or forward secrecy. Each flavor $\alpha \in \{\emptyset, \text{wbs}, \text{wfs}, \text{sbs}, \text{sfs}\}$ leads to a different definition of freshness.

Definition 2 (α -Freshness). *Let $\alpha \in \{\emptyset, \text{wfs}, \text{wbs}, \text{sfs}, \text{sbs}\}$. The oracle $\Pi_U^s \in \mathcal{G}$ is*

\emptyset -fresh: *neither Π_U^s nor any of its partners have been asked for a RevealKey query after having accepted;*

wbs-fresh: *(1) neither Π_U^s nor any of its partners have been asked for a query RevealState after \mathcal{G} is created, and (2) neither Π_U^s nor any of its partners have been asked for a RevealKey query after having accepted;*

wfs-fresh: *(1) no $U_i \in \text{pid}_U^s$ have been asked for a Corrupt query prior to a query of the form $\text{Send}(\Pi_j^{s_j}, m)$ such that $U_j \in \text{pid}_U^s$ before Π_U^s and all its partners accept, and (2) neither Π_U^s nor any of its partners have been asked for a RevealKey query after having accepted;*

sbs-fresh: *(1) no $U_i \in \text{pid}_U^s$ have been asked for a Corrupt query prior to a query of the form $\text{Send}(\Pi_j^{s_j}, m)$ such that $U_j \in \text{pid}_U^s$ after \mathcal{G} is created, (2) neither Π_U^s nor any of its partners have been asked for a RevealState query after \mathcal{G} is created, and (3) neither Π_U^s nor any of its partners have been asked for a RevealKey query after having accepted;*

sfs-fresh: *(1) no $U_i \in \text{pid}_U^s$ have been asked for a Corrupt query prior to a query of the form $\text{Send}(\Pi_j^{s_j}, m)$ such that $U_j \in \text{pid}_U^s$ before Π_U^s and all its partners accept, (2) neither Π_U^s nor any of its partners have been asked for a RevealState query before they accept, and (3) neither Π_U^s nor any of its partners have been asked for a RevealKey query after having accepted.*

We say that a session is α -fresh if all participating oracles are α -fresh.

Obviously, the **wfs**-freshness allows Corrupt queries to any user *after* the oracles in \mathcal{G} have accepted (and the **sfs**-freshness allows, additionally, RevealState queries). Beside this, Corrupt and RevealState queries are allowed for oracles outside of \mathcal{G} .

³ In case of backward secrecy Corrupt queries are more damageable than RevealState queries because the long-lived keys are usually used for authentication and their knowledge allows the adversary to impersonate users in subsequent sessions and learn the session group key.

The notion of α -fresh sessions becomes important in security proofs in order to distinguish between “honest” and “corrupted” sessions. Intuitively, the adversary will not be allowed to ask some “bad” queries. To properly manage the adversarial capabilities for each scenario of freshness, we distinguish between the following corruption models.

Definition 3 (Corruption Model β). *An adversary \mathcal{A} , in addition to Setup, RevealKey, Send, and Test, may ask RevealState and Corrupt queries as specified by the used corruption model β , which can be one of the following:*

- weak corruption model wcm:** \mathcal{A} may ask neither RevealState nor Corrupt.
- weak corr. model for forward secrecy wcm-fs:** \mathcal{A} may ask just Corrupt.
- weak corr. m. for backward secrecy wcm-bs:** \mathcal{A} may ask just RevealState.
- strong corr. model scm:** \mathcal{A} may ask both queries, RevealState and Corrupt.

A concrete proof of AKE-security needs to specify capabilities of the adversary depending on the intended freshness type. Combining definitions for freshness and corruptions, we obtain a set of *reasonable adversarial settings* $(\alpha, \beta) \in \{(\emptyset, \text{wcm}), (\text{wfs}, \text{wcm-fs}), (\text{wbs}, \text{wcm-bs}), (\text{sbs}, \text{scm}), (\text{sfs}, \text{scm})\}$. Note that other imaginable settings are not reasonable from the perspective of the attacks.

Remark 4. In practice long-lived keys are used to achieve authentication, and thus if an adversary is able to corrupt a group member (obtaining its long-lived key) then it can impersonate that member in subsequent sessions. Therefore, achieving AKE-security in the (sbs, scm) appears tricky. To the contrary, the adversarial setting $(\text{wbs}, \text{wcm-bs})$ appears of great interest since it is independent of any long-term secrets. Moreover we argue that $(\text{wbs}, \text{wcm-bs})$ is important since in previous models [12, 30] a persistent internal state is used in both past and future sessions, and thus, while forward secrecy looks at (state) corruptions in later sessions, backward secrecy must legitimately look at state corruptions in previous sessions.

3.3 Security Goals

In this section we describe security goals for a GKE protocol. We give a formal definition of (Authenticated)KeyExchange-security (indistinguishability of session group keys), and a new definition of MA-security that considers malicious participants and internal state corruptions of honest participants.

Definition 5 ((A)KE-Security). *Let P be a correct GKE protocol and b a uniformly chosen bit. Consider a reasonable adversarial setting (α, β) and an (active) adversary \mathcal{A} . We define game $\text{Game}_{\alpha, \beta, P}^{(a)\text{ke}-b}(\mathcal{A}, \kappa)$ as follows:*

- after initialization \mathcal{A} interacts with instance oracles using queries;
- if \mathcal{A} asks a Test query to an α -fresh oracle Π_U^s which has accepted, it receives either $\text{key}_1 := k_U^s$ (if $b = 1$) or $\text{key}_0 \in_R \{0, 1\}^\kappa$ (if $b = 0$);
- \mathcal{A} continues interacting with instance oracles;

- when \mathcal{A} terminates, it outputs a bit trying to guess which case it was dealing with.

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

$$\text{Adv}_{\alpha,\beta,P}^{(a)\text{ke}}(\kappa) := \max_{\mathcal{A}} |2 \Pr [\text{Game}_{\alpha,\beta,P}^{(a)\text{ke}-b}(\mathcal{A}, \kappa) = b] - 1|$$

We say that P is an (A)KE-secure protocol with α -secrecy, denoted (A)GKE- α , if the advantage $\text{Adv}_{\alpha,\beta,P}^{(a)\text{ke}}(\kappa)$ is negligible. If $\alpha = \emptyset$, we just say that P is (A)KE-secure.

Definition 6 (MA-Security). Let P be a correct GKE protocol and $\text{Game}_P^{\text{ma}}(\mathcal{A}, \kappa)$ the interaction between the instance oracles and an active adversary \mathcal{A} which can query `Send`, `Setup`, `RevealKey`, `RevealState`, and `Corrupt`. We say that \mathcal{A} wins if at some point during the interaction there exist an uncorrupted user U_i whose instance oracle $\Pi_i^{s_i}$ has accepted with $k_i^{s_i}$ and another user U_j with $U_j \in \text{pid}_i^{s_i}$ that is uncorrupted at the time $\Pi_i^{s_i}$ accepts, such that

1. there exists **no** instance oracle $\Pi_j^{s_j}$ with $(\text{pid}_j^{s_j}, \text{sid}_j^{s_j}) = (\text{pid}_i^{s_i}, \text{sid}_i^{s_i})$, **or**
2. there exists **an** instance oracle $\Pi_j^{s_j}$ with $(\text{pid}_j^{s_j}, \text{sid}_j^{s_j}) = (\text{pid}_i^{s_i}, \text{sid}_i^{s_i})$ that accepted with $k_j^{s_j} \neq k_i^{s_i}$.

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

Note that U_i and U_j must be uncorrupted, however, \mathcal{A} is allowed to reveal internal states of their oracles. Hence, our MA-security definition seems to be stronger than definitions of security against insider attacks in the KS model.

4 Compiler for AKE- and MA-Security in the Standard Model

4.1 Security-Enhancing Compilers and their Goals

Imagine, there exists a *black-box* implementation of a GKE protocol which should be used by some group application and assume this implementation does not satisfy all security requirements desired for the group application. Instead of designing and implementing a new GKE protocol in an *ad-hoc* fashion, it is desirable to have a generic technique which can be applied to the given *black-box* implementation in order to enhance its security.

Obviously, a good strategy (though not always optimal) is to implement a GKE protocol in a modular way: one starts with the basic implementation that satisfies the most common set of security requirements, then continues with the implementation of optional modules that can be added to provide extended security requirements. The main goal of security-enhancing GKE protocol compilers is to enable secure construction of GKE protocols in such a modular way.

Definition 7 (Security-Enhancing GKE Protocol Compiler \mathcal{C}). A security-enhancing GKE compiler \mathcal{C} is a procedure which takes as input a GKE protocol P and outputs a compiled GKE protocol \mathcal{C}_P with additional security properties possibly missing in P .

4.2 Discussion on Existing Compilers for GKE Protocols

In the following we provide some analysis on currently known security-enhancing compilers for GKE protocols.

Compiler for AKE-Security KE-security, i.e., key indistinguishability with respect to passive adversaries is the basic security requirement for a GKE protocol. Authentication may be optional if a network or a high-level application already provides it. Seeing AKE as an additional property, Katz and Yung designed the corresponding compiler in [31].

Definition 8 (Compiler for AKE-Security [31]). Let P be a GKE protocol and $\Sigma := (\text{Gen}, \text{Sign}, \text{Verify})$ a digital signature scheme. A compiler for AKE-security consists of an initialization algorithm and a modified protocol execution:

Initialization: each $U_i \in \mathcal{U}$ generates his own additional private/public key pair (sk'_i, pk'_i) using $\Sigma.\text{Gen}(1^{\kappa'})$.

The protocol: prior to any operation execution of P :

- Each Π_i^s chooses a nonce $r_i \in_R \{0, 1\}^\kappa$ and sends $U_i|0|r_i$ to its partners.
- After Π_i^s receives the $U_j|0|r_j$'s it computes $\text{sid}_i^s := U_1|r_1| \dots |U_n|r_n$.

Then, members of \mathcal{G} execute P with the following changes:

- When Π_i^s is supposed to send a message $U_i|t|m$, it computes $\sigma_i := \Sigma.\text{Sign}(sk'_i, t|m|\text{sid}_i^s)$ and sends the modified message $U_i|t|m|\sigma_i$.
- When Π_i^s receives $U_j|t|m|\sigma_j$ it checks whether (1) $U_j \in \text{pid}_i^s$, (2) t is the next expected sequence number, and (3) $\Sigma.\text{Verify}(pk'_j, t|m|\text{sid}_i^s, \sigma_j) = ?1$. If all checks pass, it proceeds as in P upon receiving $U_j|t|m$.
- After Π_i^s computes the session key k_i^s in P , it accepts with this key.

Missing Generality of Katz-Yung Compiler Katz and Yung proved security of this compiler assuming an unreliable asynchronous broadcast channel and a passive adversary, which is only an *eavesdropper*. We show that in this case their compiler is not really generic: there exist GKE protocols that are secure against eavesdroppers but become insecure against active adversaries (even after the execution of the above compiler).

We consider the following (pathologic, but illustrative) protocol between Π_1^s and Π_2^s . First Π_1^s chooses his exponent $x_1 \in_R \mathbb{Z}_q^*$ and sends $X_1 := g^{x_1}$ to Π_2^s . If Π_2^s receives X_1 within some specified time period δ then Π_2^s replies with $X_2 := g^{x_2}$ for some randomly chosen $x_2 \in_R \mathbb{Z}_q^*$ and accepts with the Diffie-Hellman session key $g^{x_1 x_2}$. Similar if Π_1^s receives X_2 within time δ then it accepts with $g^{x_1 x_2}$ too. However, if an oracle does not receive data in time, it accepts with g . If the passive adversary is just an eavesdropper, messages are delivered

on time, and the protocol is “passively” secure. But an active adversary can drop messages so that both participants accept with g . Obviously, it is insufficient to restrict passive adversaries to be just eavesdroppers. Passive attacks should also model the unreliability of the communication, like we do.

Compilers for MA-Security The first compiler for key confirmation and mutual authentication was proposed by Bresson *et al.* [14]. However, their definition of MA is (*de facto*) the old one, and the proof is conducted in the Random Oracle Model [6].

Katz and Shin [30] showed how to turn an AKE-secure GKE protocol into a UC-secure GKE protocol that provides security against insider attacks (see Section 2.2).

Definition 9 (Compiler for Security against Insider Attacks by Katz and Shin). *Let P be a GKE protocol, $\Sigma := (\text{Gen}, \text{Sign}, \text{Verify})$ a digital signature scheme, $F := \{f_k\}_{k \in \{0,1\}^\kappa}$ a function ensemble with range $\{0,1\}^\lambda$, $\lambda \in \mathbb{N}$ and domain $\{0,1\}^\kappa$, and sid_i^s is a unique session id. A compiler for security against insider attacks consists of an initialization algorithm and a protocol defined as follows:*

Initialization: *each $U_i \in \mathcal{U}$ generates his own additional private/public key pair (sk'_i, pk'_i) using $\Sigma.\text{Gen}(1^{\kappa'})$.*

The protocol: *After an oracle Π_i^s accepts with $(k_i^s, \text{pid}_i^s, \text{sid}_i^s)$ in P :*

- *it computes $\mu_i := f_{k_i^s}(v_0)$ where v_0 is a constant public value;*
- *it computes $K_i^s := f_{k_i^s}(v_1)$ where $v_1 \neq v_0$ is another constant public value;*
- *it erases its local state information except for $\mu_i, K_i^s, \text{pid}_i^s$, and sid_i^s ;*
- *it computes a signature $\sigma_i := \Sigma.\text{Sign}(sk'_i, \mu_i | \text{sid}_i^s | \text{pid}_i^s)$ and sends $U_i | \sigma_i$ to its partnered oracle Π_j^s .*

After Π_i^s receives $U_j | \sigma_j$ from all its partnered oracle Π_j^s :

- *it checks whether $\Sigma.\text{Verify}(pk'_j, \mu_i | \text{sid}_i^s | \text{pid}_i^s, \sigma_j) \stackrel{?}{=} 1$;*
- *if all checks pass, it accepts with the session key K_i^s .*

4.3 Compiler C-AMA

In the following we describe a compiler (denoted C-AMA) which provides both AKE- and MA-security. C-AMA uses nonces to achieve uniqueness of protocol sessions and security of concurrent executions without relying on session ids given by high-level applications.

Definition 10 (Compiler for AKE- and MA-Security C-AMA). *Let P be a GKE protocol, $\Sigma := (\text{Gen}, \text{Sign}, \text{Verify})$ a digital signature scheme, $F := \{f_k\}_{k \in \{0,1\}^\kappa}$ a function ensemble with range $\{0,1\}^\lambda$, $\lambda \in \mathbb{N}$ and domain $\{0,1\}^\kappa$. A compiler for AKE-security and MA-security, denoted C-AMA, consists of an algorithm INIT and a protocol AMA defined as follows:*

INIT: *each $U_i \in \mathcal{U}$ generates private/public key pair (sk'_i, pk'_i) using $\Sigma.\text{Gen}(1^{\kappa'})$.*

AMA: prior to the execution of P :

- Each Π_i^s chooses a AMA nonce $r_i \in_R \{0, 1\}^\kappa$ and sends $U_i|r_i$ to its partners.
- After Π_i^s receives all $U_j|r_j$, it computes $\text{sid}_i^s := r_1 | \dots | r_n$.

Then it invokes the execution of P and proceeds as follows:

- If Π_i^s in P outputs a message $U_i|m$ then in C-AMA_P it computes additionally $\sigma_i := \Sigma.\text{Sign}(sk'_i, m|\text{sid}_i^s|\text{pid}_i^s)$ and outputs a modified message $U_i|m|\sigma_i$.
- On receiving $U_j|m|\sigma_j$ from a partner Π_j^s it checks whether $\Sigma.\text{Verify}(pk'_j, m|\text{sid}_i^s|\text{pid}_i^s, \sigma_j) =? 1$. If this fails then Π_i^s terminates; otherwise it proceeds as in P upon receiving $U_j|m$.
- After an oracle Π_i^s computes k_i^s in P it computes an AMA token $\mu_i := f_{k_i^s}(v_0)$ where v_0 is a constant public value, a signature $\sigma_i := \Sigma.\text{Sign}(sk'_i, \mu_i|\text{sid}_i^s|\text{pid}_i^s)$ and sends $U_i|\sigma_i$ to every Π_j^s with $U_j \in \text{pid}_i^s$.
- On receiving $U_j|\sigma_j$ from its partner, Π_i^s checks if $\Sigma.\text{Verify}(pk'_j, \mu_i|\text{sid}_i^s|\text{pid}_i^s, \sigma_j) =? 1$.
- If all checks pass, Π_i^s computes $K_i^s := f_{k_i^s}(v_1)$ where $v_1 \neq v_0$ is another constant public value, erases all private state information from state_i^s (including k_i^s) and accepts with K_i^s .

PERFORMANCE ANALYSIS. C-AMA requires two further rounds: one to exchange random nonces and another one to exchange signatures on AMA tokens. As for the computation costs, every participant generates one signature for every message sent during P and one additional signature on the computed AMA token. Furthermore, every participant must verify one signature for every incoming message in P and $n - 1$ signatures during the additional confirmation round. The computation of the AMA token μ and of the session key K can be seen as negligible.

SECURITY ANALYSIS. Our first theorem (full proof will be given in the full version) shows that C-AMA adds AKE-security to any KE-secure GKE protocol. Following Remark 4, we do not consider the adversarial setting (sbs, scm) . For the definition of collision-resistance for F we refer for example to [30].

Theorem 11 (AKE-Security of C-AMA_P). *Let $(\alpha, \beta) \in \{(\emptyset, \text{wcm}), (\text{wbs}, \text{wcm-bs}), (\text{wfs}, \text{wcm-fs}), (\text{sfs}, \text{scm})\}$ be an adversarial setting, let P be a GKE- α protocol, and \mathcal{A} an active adversary in the corruption model β launching at most q_s sessions of C-AMA_P . If Σ is EUF-CMA and F is pseudo-random then C-AMA_P is AGKE- α , and*

$$\text{Adv}_{\alpha, \beta, \text{C-AMA}_P}^{\text{ake}}(\kappa) \leq 2N \text{Succ}_{\Sigma}^{\text{euf-cma}}(\kappa) + \frac{Nq_s^2}{2^{\kappa-1}} + 2q_s \text{Adv}_{\alpha, \beta, P}^{\text{ke}}(\kappa) + 4q_s \text{Adv}_F^{\text{prf}}(\kappa).$$

Proof (Sketch). We define a sequence of games [40] \mathbf{G}_i , $i = 0, \dots, 6$ and denote by $\text{Win}_i^{\text{ake}}$ the event that the adversary correctly guesses the bit b in \mathbf{G}_i .

Game \mathbf{G}_0 : This is the real $\text{Game}_{\alpha, \beta, \text{C-AMA}_P}^{\text{ake-b}}(\mathcal{A}, \kappa)$ played between the simulator \mathcal{S} and an active adversary \mathcal{A} .

Game \mathbf{G}_1 : Here the simulation fails and bit b' is set at random if \mathcal{A} asks a **Send** query on some $U_i|m|\sigma$ (or $U_i|\sigma$) such that σ is a valid signature that has not been previously output by an oracle Π_i^s before querying **Corrupt**(U_i), i.e., if a forgery occurs. One can show that $|\Pr[\text{Win}_1^{\text{ake}}] - \Pr[\text{Win}_0^{\text{ake}}]| \leq N\text{Succ}_{\Sigma}^{\text{euf-cma}}(\kappa)$.

Game \mathbf{G}_2 : In this game the simulation fails and bit b' is set at random if an AMA nonce r_i is used by any uncorrupted user's oracle Π_i^s in two different sessions. Considering collisions for the choice of AMA nonces one can show that $|\Pr[\text{Win}_2^{\text{ake}}] - \Pr[\text{Win}_1^{\text{ake}}]| \leq \frac{Nq_s^2}{2^\kappa}$. This game excludes replay attacks.

Game \mathbf{G}_3 : In this game we add the following rule: \mathcal{S} chooses $q_s^* \in [1, q_s]$ as a guess for the number of sessions invoked before \mathcal{A} asks the query **Test**. If this query does not occur in the q_s^* -th session then the simulation fails and bit b' is set at random. One can show that $\Pr[\text{Win}_2^{\text{ake}}] = q_s \left(\Pr[\text{Win}_3^{\text{ake}}] - \frac{1}{2} \right) + \frac{1}{2}$.

Game \mathbf{G}_4 : In this game \mathcal{S} acts as a passive adversary against the KE-security of \mathbf{P} that participates in $\text{Game}_{\alpha, \beta, \mathbf{P}}^{\text{ke-1}}(\kappa)$, i.e., the **Test** query of \mathcal{S} to an accepted α -fresh oracle Π_i^s in \mathbf{P} is answered with the real session group key k_i^s . In the full version we show how \mathcal{S} answers the queries of \mathcal{A} such that $\Pr[\text{Win}_4^{\text{ake}}] = \Pr[\text{Win}_3^{\text{ake}}]$.

Game \mathbf{G}_5 : The only difference to \mathbf{G}_4 is that \mathcal{S} participates as a passive adversary in $\text{Game}_{\alpha, \beta, \mathbf{P}}^{\text{ke-0}}(\kappa)$, i.e., the **Test** query is answered with a random bit string. Hence, $|\Pr[\text{Win}_5^{\text{ake}}] - \Pr[\text{Win}_4^{\text{ake}}]| \leq \text{Adv}_{\alpha, \beta, \mathbf{P}}^{\text{ke}}(\kappa)$.

Game \mathbf{G}_6 : This game is identical to \mathbf{G}_5 except that in the q_s^* -th session K and the AMA token μ are replaced by random values sampled from $\{0, 1\}^\kappa$. Hence, $|\Pr[\text{Win}_6^{\text{ake}}] - \Pr[\text{Win}_5^{\text{ake}}]| \leq 2\text{Adv}_F^{\text{prf}}(\kappa)$ and $\Pr[\text{Win}_6^{\text{ake}}] = \frac{1}{2}$. This results in the desired inequality. \square

Our next theorem shows that **C-AMA** provides **MA-security** (with malicious participants and strong corruptions) for any **GKE** protocol \mathbf{P} .

Theorem 12 (MA-Security of C-AMA $_P$). *Let \mathbf{P} be a GKE protocol and \mathcal{A} an active adversary launching at most q_s sessions of **C-AMA $_P$** . If Σ is **EUFCMA** and F is collision-resistant then **C-AMA $_P$** is **MAGKE**, and*

$$\text{Succ}_{\text{C-AMA}_P}^{\text{ma}}(\kappa) \leq N\text{Succ}_{\Sigma}^{\text{euf-cma}}(\kappa) + \frac{Nq_s^2}{2^\kappa} + q_s\text{Succ}_F^{\text{coll}}(\kappa).$$

Proof (Sketch). We define a sequence of games \mathbf{G}_i , $i = 0, \dots, 2$ and events Win_i^{ma} meaning that \mathcal{A} wins in \mathbf{G}_i . \mathcal{A} 's queries are answered by a simulator \mathcal{S} .

Game \mathbf{G}_0 : This is the real $\text{Game}_{\text{C-AMA}_P}^{\text{ma}}(\mathcal{A}, \kappa)$ played between \mathcal{S} and \mathcal{A} . The goal of \mathcal{A} is to achieve that there exists an uncorrupted user U_i whose corresponding oracle Π_i^s accepts with K_i^s and another user $U_j \in \text{pid}_i^s$ that is uncorrupted at the time Π_i^s accepts and either does not have a corresponding oracle Π_j^s with $(\text{pid}_j^s, \text{sid}_j^s) = (\text{pid}_i^s, \text{sid}_i^s)$ or has such an oracle but this oracle accepts with $K_j^s \neq K_i^s$.

Game \mathbf{G}_1 : Here the simulation fails if a forgery occurs. Similar to the previous proof we have: $|\Pr[\text{Win}_1^{\text{ma}}] - \Pr[\text{Win}_0^{\text{ma}}]| \leq N\text{Succ}_{\Sigma}^{\text{euf-cma}}(\kappa)$.

Game \mathbf{G}_2 : Here we abort in case of nonces collision. $|\Pr[\text{Win}_2^{\text{ma}}] - \Pr[\text{Win}_1^{\text{ma}}]| \leq \frac{Nq_s^2}{2^\kappa}$.

Having excluded forgeries and replay attacks, we follow that for every $U_j \in \text{pid}_i^s$ that is uncorrupted when Π_i^s accepts, there exists a corresponding Π_j^s with $(\text{pid}_j^s, \text{sid}_j^s) = (\text{pid}_i^s, \text{sid}_i^s)$. Thus, according to Definition 6 \mathcal{A} wins in this game only if any of these oracles has accepted with $K_j^s \neq K_i^s$. Arguing by contradiction one can show that $\Pr[\text{Win}_2^{\text{ma}}] = \Pr[K_i^s \neq K_j^s \wedge f_{k_i^s}(v_0) = f_{k_j^s}(v_0)] \leq q_s \text{Succ}_F^{\text{coll}}(\kappa)$ and obtain the desired inequality. \square

5 Conclusion

In this paper we found some problems with the definition of MA-security in the foundational BCPQ model. We proposed a revised definition which considers malicious participants and unifies many of the well-known informal notions. Additionally, we extended the (strong) forward secrecy in AKE-security by the symmetrically opposed notion of (strong) backward secrecy. Further we described the provably secure generic compiler $\mathcal{C}\text{-AMA}$ that adds AKE- and MA-security to any GKE protocol which is passively secure (wrt. to an adversary which can change the delivery order of messages, and delay or drop them).

We refer to our parallel work in [15,35] for further development, in particular, in the light of the *contributory* nature of GKE protocols and the appropriate generic solution for this security goal.

References

1. M. Abdalla, E. Bresson, O. Chevassut, and D. Pointcheval. Password-Based Group Key Exchange in a Constant Number of Rounds. In *PKC'06*, pp. 427–442.
2. G. Ateniese, M. Steiner, and G. Tsudik. Authenticated Group Key Agreement and Friends. In *CCS'98*, pp. 17–26.
3. B. Barak, Y. Lindell, and T. Rabin. Protocol Initialization for the Framework of Universal Composability. <http://eprint.iacr.org/2004/006>.
4. M. Bellare. Practice-Oriented Provable-Security. In *ISW'97*, pp. 221–231.
5. M. Bellare and P. Rogaway. Entity Authentication and Key Distribution. In *CRYPTO'93*, pp. 232–249.
6. M. Bellare and P. Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *CCS'93*, pp. 62–73.
7. M. Bellare and P. Rogaway. Provably Secure Session Key Distribution: The Three Party Case. In *STOC'95*, pp. 57–66.
8. J.-M. Bohli, M. I. G. Vasco, and R. Steinwandt. Secure Group Key Establishment Revisited. <http://eprint.iacr.org/2005/395>.
9. C. Boyd and A. Mathuria. *Protocols for Authentication and Key Establishment*. Springer, 2003. ISBN:3-540-43107-1.
10. C. Boyd and J. M. Nieto. Round-Optimal Contributory Conference Key Agreement. In *PKC'03*, pp. 161–174.
11. E. Bresson, O. Chevassut, and D. Pointcheval. Provably Authenticated Group Diffie-Hellman Key Exchange — The Dynamic Case. In *ASIACRYPT'01*, pp. 290–390.
12. E. Bresson, O. Chevassut, and D. Pointcheval. Dynamic Group Diffie-Hellman Key Exchange under Standard Assumptions. In *EUROCRYPT'02*, pp. 321–336.

13. E. Bresson, O. Chevassut, and D. Pointcheval. Group Diffie-Hellman Key Exchange Secure against Dictionary Attacks. In *ASIACRYPT'02*, pp. 497–514.
14. E. Bresson, O. Chevassut, D. Pointcheval, and J.-J. Quisquater. Provably Authenticated Group Diffie-Hellman Key Exchange. In *CCS'01*, pp. 255–264.
15. E. Bresson and M. Manulis. Malicious Participants in Group Key Exchange: Key Control and Contributiveness in the Shadow of Trust. In *ATC'07*, pp. 395–409.
16. M. Burmester. On the Risk of Opening Distributed Keys. In *CRYPTO'94*, pp. 308–317.
17. M. Burmester and Y. Desmedt. A Secure and Efficient Conference Key Distribution System. In *EUROCRYPT'94*, pp. 275–286.
18. R. Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *FOCS'01*, pp. 136–145.
19. R. Canetti and H. Krawczyk. Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In *EUROCRYPT'01*, pp. 453–474.
20. K.-K. R. Choo, C. Boyd, and Y. Hitchcock. Errors in Computational Complexity Proofs for Protocols. In *ASIACRYPT'05*, pp. 624–643.
21. K.-K. R. Choo, C. Boyd, and Y. Hitchcock. Examining Indistinguishability-Based Proof Models for Key Establishment Protocols. In *ASIACRYPT'05*, pp. 585–604.
22. Y. Desmedt, J. Pieprzyk, R. Steinfeld, and H. Wang. A Non-malleable Group Key Exchange Protocol Robust Against Active Insiders. In *ISC'06*, pp. 459–475.
23. W. Diffie and M. E. Hellman. New Directions in Cryptography. *IEEE Trans. on Information Theory*, IT-22(6):644–654, 1976.
24. W. Diffie, P. C. van Oorschot, and M. J. Wiener. Authentication and Authenticated Key Exchanges. *Designs, Codes and Cryptography*, 2(2):107–125, 1992.
25. R. Dutta and R. Barua. Constant Round Dynamic Group Key Agreement. In *ISC'05*, pp. 74–88.
26. R. Dutta and R. Barua. Dynamic Group Key Agreement in Tree-Based Setting. In *ACISP'05*, pp. 101–112.
27. R. Dutta, R. Barua, and P. Sarkar. Provably Secure Authenticated Tree Based Group Key Agreement. In *ICICS'04*, pp. 92–104.
28. C. G. Günther. An Identity-Based Key-Exchange Protocol. In *EUROCRYPT'89*, pp. 29–37.
29. I. Ingemarsson, D. T. Tang, and C. K. Wong. A Conference Key Distribution System. *IEEE Trans. on Information Theory*, 28(5):714–719, 1982.
30. J. Katz and J. S. Shin. Modeling Insider Attacks on Group Key-Exchange Protocols. In *CCS'05*, pp. 180–189.
31. J. Katz and M. Yung. Scalable Protocols for Authenticated Group Key Exchange. In *CRYPTO'03*, pp. 110–125.
32. H.-J. Kim, S.-M. Lee, and D. H. Lee. Constant-Round Authenticated Group Key Exchange for Dynamic Groups. In *ASIACRYPT'04*, pp. 245–259.
33. Y. Kim, A. Perrig, and G. Tsudik. Simple and Fault-Tolerant Key Agreement for Dynamic Collaborative Groups. In *CCS'00*, pp. 235–244.
34. Y. Kim, A. Perrig, and G. Tsudik. Communication-Efficient Group Key Agreement. In *IFIP/Sec'01*, pp. 229–244.
35. M. Manulis. *Provably Secure Group Key Exchange*. PhD thesis, Ruhr University Bochum, June 2007.
36. A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, October 1996.
37. O. Pereira and J.-J. Quisquater. Some Attacks upon Authenticated Group Key Agreement Protocols. *J. of Computer Security*, 11(4):555–580, 2003.

38. A. Perrig. Efficient Collaborative Key Management Protocols for Secure Autonomous Group Communication. In *CryptEC'99*, pp. 192–202.
39. V. Shoup. On Formal Models for Secure Key Exchange (Version 4). IBM RZ 3120, November 1999. <http://shoup.net/>.
40. V. Shoup. Sequences of Games: A Tool for Taming Complexity in Security Proofs. <http://eprint.iacr.org/2004/332>.
41. D. G. Steer, L. Strawczynski, W. Diffie, and M. J. Wiener. A Secure Audio Teleconf. System. In *CRYPTO'88*, pp. 520–528.
42. M. Steiner, G. Tsudik, and M. Waidner. CLIQUES: A New Approach to Group Key Agreement. In *ICDCS'98*, pp. 380–387.
43. Y. Yacobi and Z. Shmueli. On Key Distribution Systems. In *CRYPTO'89*, pp. 344–355.