RUHR-UNIVERSITÄT BOCHUM

Horst Görtz Institute for IT Security

Technical Report TR-HGI-2006-002

Survey on Security Requirements and Models for Group Key Exchange

Mark Manulis

Chair for Network and Data Security

Ruhr-Universität Bochum
Horst Görtz Institute for IT Security
D-44780 Bochum, Germany

TR-HGI-2006-002
January 5, 2008

Survey on Security Requirements and Models for Group Key Exchange

Mark Manulis
mark.manulis@rub.de

Abstract

In this report we provide an analytical survey on security issues that are relevant for group key exchange (GKE) protocols. We start with the description of the security requirements that have been informally described in the literature and widely used to analyze security of earlier GKE protocols. Most of these definitions were originally stated for two-party protocols and then adapted to a group setting. These informal definitions are foundational for the later appeared formal security models for GKE protocols whose development, strengths, and weaknesses are also described and analyzed.

Keywords: group key exchange, security model, survey, analysis

Contents

1 Group Key Establishment

The establishment of group keys is fundamental for a variety of security mechanisms in group applications. For example, group keys can be utilized by symmetric encryption schemes for the purpose of confidentiality which is one of the most frequent security requirements in group applications; also message authentication codes require group keys for the purpose of group authentication and integrity. Thus, it is important to have mechanisms that provide group members with shared secret keys. We classify possible mechanisms based on the following definitions from [MvOV96, Chapter 12] which we adopted to a group setting.

Definition 1 (Group Key Establishment) *Group key establishment is a process or protocol whereby a shared secret becomes available to two or more parties, for subsequent cryptographic use.*

This general definition can further be shaped in two different classes: *group key transport/distribution* and *group key exchange/agreement*.

Definition 2 (Group Key Transport/Distribution) *A group key transport/distribution protocol or mechanism is a group key establishment technique where one party creates or otherwise obtains a secret value, and securely transfers it to the other(s).*

The main characteristic of group key transport protocols is that the group key k is chosen by a single party and then securely transferred to all group members. This definition leaves open whether a party which chooses the group key must be a group member. It is also imaginable to have some trusted third party (TTP) that chooses group keys on behalf of the group. Also the requirement on secure transfer of group keys forebodes the existence of secret communication channels between the party that chooses group keys and other group members.

Definition 3 (Group Key Exchange/Agreement) *A group key exchange/agreement protocol or mechanism is a group key establishment technique in which a shared secret is derived by two or more parties as a function of the information contributed by, or associated with, each of these, (ideally) such that no party can predetermine the resulting value.*

Obviously, in group key exchange protocols all group members have to interact in order to compute the group key. The main difference to group key transport techniques is that no party is allowed to choose the group key on behalf of the whole group. Also, group key exchange protocols do not require the existence of secure channels between participants since no secure transfer takes place.

Note that regardless of which group key establishment technique is used by an application the resulting group key must remain secret from unauthorized parties in order to guarantee the expected requirements from the utilized cryptographic mechanisms, like encryption schemes or message authentication codes.

Both group key establishment techniques can be analyzed in context of either static or dynamic groups. Of course it is always possible to establish the group key for the modified group by re-starting the protocol. However, this may be inefficient if groups are large or the protocol is computationally expensive. Therefore, many group key establishment protocols designed for dynamic groups provide more efficient operations for addition and exclusion of group members.

1.1 Group Key Transport/Distribution

In group key transport/distribution protocols the party which chooses group keys on behalf of the group is given enormous power and may, therefore, influence the security of the protocol. Whether a group application allows this kind of trust relationship depends surely on its goals and the environment in which it is executed. However, it seems evident that group key transport protocols are primarily used in group applications with centralized control over the group admission process. In these scenarios the party acting as group authority (GA) may also be in charge for the choice of the group key and its distribution to other members.

Obviously, the most challenging task in group key transport protocols is its protection during the protocol execution.

The following general mechanism is usually applied in group key transport protocols, e.g. [BD94, HY98]. After the party which is responsible for the choice of the group key chooses the key it encrypts it via an appropriate encryption scheme and distributes it to all other group members. Both, symmetric and public-key encryption schemes can be used for this purpose. In case that the applied scheme is symmetric the existence of shared secret keys between this party and each group member is indispensable. This means, that group members have to exchange secret keys with that party pairwise before it proceeds with the group key distribution. Another solution is to apply public-key encryption schemes which do not require any pre-shared secrets between group members and the central party, e.g. in [MY99]. However, public-key encryption is usually less efficient than symmetric encryption. Therefore, if group keys are distributed frequently, e.g. due to frequent group membership changes, then symmetric cryptography performs better.

The core of many group key distribution protocols builds a mechanism called *key hierarchy* [WGL98, WHA99]. It arranges group members at the leaves of a logical tree and assigns some secret value to each node of the tree. The secret value at the root of the tree represents

the group key or a secret material which can be used to derive the group key via some additional transformations. The goal of the distribution process in key hierarchies is to provide each group member with the information which it can use to compute all secret values in its path up to the root, including the group key. Key hierarchies are popular in dynamic group key distribution protocols for multicast and broadcast encryption, e.g. [WGL98, WHA99, Bri99, WCS+99, NNL01, SM03], since they provide various mechanisms based on modification of the logical tree structure to enhance protocol efficiency upon dynamic group changes.

1.2 Group Key Exchange/Agreement

The only trust assumption in group key exchange protocols is that members trust each other not to reveal any information which can be used to derive the group key to any third party which is not a valid member of the group. Especially, group members do not trust each other during the computation of the group key which should be composed of individual contributions of all group members. Thus, in contrast to group key transport protocols the design of group key exchange protocols is more challenging due to the distributed computation process of the group key.

Many group key exchange protocols can be seen as modifications of the two-party key exchange protocol proposed by Diffie and Hellman in their seminal paper [DH76]. This protocol allows two parties upon exchange of information over a public channel to compute the shared key using specific discrete mathematical constructions which prevent eavesdroppers from learning the established key.

1.3 Session Keys

Usually, group keys returned by group key establishment protocols are not used directly in the application. Instead, additional transformations are applied in order to derive further keys, so-called *session keys*, which are used by different security mechanisms within the application. For example, if an application requires confidentiality and group authentication then one session key is derived for the encryption scheme and another session key is derived for the message authentication code. Transformations which are used to derive session keys are usually one-way, i.e., given the output of the transformation it is computationally infeasible to obtain the input. Thus, even if a session key is leaked it is still hard to compute the original group key. The use of different session keys provides additional security in terms of independence of applications and deployed security mechanisms since leakage of one session key does not imply leakage of other session keys. Thus for example, if a third party learns the

session key used for group authentication then it can authenticate itself as a group member but is not able to decrypt encrypted group messages. Furthermore, session keys are *ephemeral*, i.e., they are valid for a short period of time. For example, digital conferences should use a different session key for each communication session. The use of ephemeral keys decreases the chance of cryptanalytic attacks. Also in dynamic groups any change of group membership should result in new session keys. In case that the group is static but application execution lasts long, e.g., if groupware is used in some long-term project, it is reasonable to refresh session keys periodically.

2 Survey on Informal Security Definitions

Security properties of cryptographic schemes are usually defined based on certain assumptions about the adversary whose goal is to break these properties. In case of cryptographic protocols it is common to distinguish between passive and active adversaries. A *passive adversary*, usually, only eavesdrops the communication channel without being able to modify or inject messages. An *active adversary* is more powerful since it is assumed to have a complete control over the communication channel resulting in its ability to alter sent messages or inject own messages during the execution of the protocol. In particular, an active adversary is able to mount so-called *man-in-the-middle attacks*. Additionally, security of a cryptographic protocol may depend on the behavior of its participants. Obviously, it is more challenging for a protocol to guarantee its security properties in case where legitimate participants are *malicious*, or dishonest, and do not act according to the protocol specification.

In the following we consider blocks of related security notions which we describe following the chronology of their appearance in the literature. We also specify which type of the adversary is reasonable to be assumed for each notion.

2.1 Semantic Security and Known-Key Attacks

The notion of *key privacy*, also called *key confidentiality* or *key secrecy* [DvOW92], was surfaced by Diffie and Hellman [DH76], and described later in the context of group key establishment [SSDW90, BD94]. According to the definition of Burmester and Desmedt [BD94] a group key establishment protocol guarantees privacy if it is computationally infeasible for a passive adversary to compute the group key k. Obviously, similar definition should hold against an active adversary who is not a legitimate protocol participant. A stronger definition of key privacy requires the indistinguishability of the computed group key from a random number. Thus, an adversary given either a real group key or a random string sampled from the same space should not be able to distinguish which value it has been given. This is in

spirit of the *semantic security* requirement proposed by Goldwasser and Micali [GM84] in the context of digital encryption schemes. Note, however, that this requirement can only hold under the assumption that the adversary does not obtain any additional information that would allow it to verify the given value, e.g., if the adversary obtains a cipher text computed using the real group key then it can easily distinguish between the real group key and some random value by comparing the corresponding cipher text.

The notion of *known-key security* [YS90, Bur94], strengthens the above requirements by assuming a stronger adversary who knows the group keys of past sessions. For example, members excluded from the group should not be able to compute or distinguish the updated group keys. The related notion of *key freshness* [MvOV96] requires that the protocol guarantees that the key is new, that is participants compute group keys which have not been used in the past. Steiner *et al.* [STW98] introduced the notion of *key independence* in the context of dynamic group key exchange protocols meaning that previously used group keys must not be discovered by joined group members and that former group members must not be able to compute the group keys used in the future. Obviously, this definition considers that the adversary was a legitimate protocol participant or may become one in the future.

Kim *et al.* [KPT00, KPT01] summarized the above requirements as follows: *weak backward secrecy* guarantees that previously used group keys must not be discovered by new group members; *weak forward secrecy* guarantees that new keys must remain out of reach of former group members; *computational group key secrecy* guarantees that it is computationally infeasible for a passive adversary to discover any group key; *forward secrecy* guarantees that a passive adversary who knows a contiguous subset of old group keys cannot discover subsequent group keys; *backward secrecy* guarantees that a passive adversary who knows a contiguous set of group keys cannot discover any preceding group keys; *key independence* guarantees that a passive adversary who knows any proper subset of group keys cannot discover any other group key. Note that the last four requirements do not make any assumptions about the group membership of the adversary. In their subsequent work, Kim *et al.* [KPT04] introduced the *decisional group key secrecy* whereby a passive adversary must not be able to distinguish the group key from a random number. Although [KPT00, KPT01] use passive adversaries in their definitions, it is mentioned that the same security requirements should also hold in the presence of active adversaries.

Remark 1 *Unfortunately, Kim* et al.*'s definitions concerning (weak) forward secrecy are in conflict with the commonly used meaning of the term forward secrecy (see Section 2.4 for further details).*

2.2 Impersonation Attacks

Security against *impersonation attacks* in the context of group key establishment was addressed by Burmester and Desmedt [BD94] and defined as a property of the protocol where an impersonator together with active or passive adversaries should be prevented from the computation of the group key. By an impersonator [BD94] denotes an adversary whose goal is to replace a legitimate participant in the execution of the protocol (thus impersonator is not considered to be a malicious participant but rather some external party). Further, [Bur94, BD93] extend the notion of known-key attacks by requiring that an active adversary who knows past session keys must not be able to impersonate one of the protocol participants.

The notion of *entity authentication* [BR93a], introduced by Bellare and Rogaway in the context of two-party authentication protocols, specifies a process whereby one party is assured of the identity of the second party involved in the protocol, and of the actual protocol participation of the latter. This requirement is equivalent to the requirement on resistance against impersonation attacks in the context of group key exchange protocols. The related notion called *(implicit) key authentication* [MvOV96] requires that each legitimate protocol participant is assured that no other party except for other legitimate participants learns the established group key. According to this definition a group key exchange protocol is *authenticated* if it provides (implicit) key authentication. Ateniese *et al.* [AST98] proposed a requirement on *group integrity* meaning that each protocol participant must be assured of every other party's participation in the protocol. Obviously, this notion is similar to the requirement of the entity authentication applied to a group setting.

All of these impersonation/authentication requirements consider an adversary that represents some external party and not a legitimate protocol participant. Therefore, these requirements are similar to those of the previous section assuming that the adversary is active, i.e., that the indistinguishability of the real group keys and the random numbers sampled from the same space remains preserved with respect to the attacks of an active adversary which is allowed to modify and inject messages during the protocol execution.

Another related requirement called *unknown key-share resilience* surfaced in [DvOW92] means that an active adversary must not be able to make one protocol participant believe that the key is shared with one party when it is in fact shared with another party. Note that in this attack the adversary may be a malicious participant and does not need necessarily to learn the established group key [BM03].

Finally, we mention *key-compromise impersonation resilience* [BWJM97]. This security property prevents the adversary who obtains a long-term key of a user from being able to impersonate other users to that one. Note that long-term (a.k.a. long-lived) keys are usually

either private keys used for signature generation or digital decryption, or shared secret (small entropy) passwords that remain unchanged for a long period of time. In both cases long-lived keys are used primarily for the purpose of authentication rather than for the actual computation of the group key. Obviously, this attack concerns only protocols whose goal is to establish a session key which is then used for the purpose of authentication. Therefore, it is arguable (e.g. [KS05]) whether this requirement is general for all group key exchange protocols. Note that if an adversary obtains long-lived keys of participants then it can usually act on behalf of these participants in subsequent protocol executions.

2.3 Key Confirmation and Mutual Authentication

The requirement called *key confirmation* [MvOV96] means that each protocol participant must be assured that every other protocol participant actually has possession of the computed group key. According to [MvOV96] key confirmation in conjunction with (implicit) key authentication results in *explicit key authentication*, i.e., each identified protocol participant is known to actually possess the established group key. The same goal states the requirement of *mutual authentication* introduced in [BR93b] when considered for group key exchange protocols. As noted in [AST98] key confirmation makes a group key exchange protocol a more robust and a more autonomous operation. According to [BM03] key confirmation mechanisms can be used to provide resistance against unknown key-share attacks mentioned in the previous section. We stress that the requirements on key confirmation and mutual authentication should also be considered from the perspective of the attacks by malicious protocol participants who try to prevent honest participants from computing identical group keys.

2.4 (Perfect) Forward Secrecy

The notion of *(perfect) forward secrecy* (sometimes called *break-backward protection* [MvOV96]) was surfaced by Günter [Gün90] and rephrased by Diffie *et al.* [DvOW92] as a property of an authenticated key agreement protocol requiring that the disclosure of long-term keying material does not compromise the secrecy of the established keys from earlier protocol sessions. The idea behind this notion is to maintain the protection of the secure traffic in the future. Note that the compromised long-term keys make future protocol sessions nonetheless susceptible to impersonation attacks. A weaker form of (perfect) forward secrecy is *partial forward secrecy* [BM03] which considers the case where one (or more but not all) principals' long-term keys become compromised.

2.5 Key Control and Contributiveness

The issue of *key control* described by Mitchel *et al.* [MWW98] in the context of two-party group key exchange protocols considers malicious protocol participants who wish to influence the computation of the group key.

Ateniese *et al.* [AST98] introduced a notion of *contributory group key agreement* meaning such protocols where each party equally contributes to the established group key and guarantees its freshness (see also [Ste02]). This notion also subsumes the requirement of *unpredictability* of the computed group keys. This is also in spirit of the later introduced resilience against *key replication attacks* [Kra05] by which the adversary should not be able to enforce the same value of the group key in two different sessions. Note that all these requirements clearly exclude group key distribution protocols (see Section 1.1) where some trusted party is responsible for the generation of the group keys. Ateniese *et al.* defined additionally *complete group key authentication* as a property of a group key exchange protocol whereby all parties compute the same group key only if each of the parties have contributed to its computation. This notion can be seen as a combination of contributiveness and mutual (or explicit key) authentication. Ateniese *et al.* [AST98] proposed further a more stronger notion of *verifiable contributory group key agreement* meaning protocols where each participant is assured of every other participant's contribution to the group key. We stress that these requirements should also hold in the presence of malicious protocol participants.

Another related requirement that is stated from the perspective of an adversary that is not a malicious participant is *key integrity* [JT93] which requires that the established group key has not been modified by the adversary, or equivalently only has inputs from legitimate protocol participants. Ateniese *et al.* [AST98] extended the definition of key integrity by requiring that the established group key is a function of only the individual contributions of legitimate protocol participants such that extraneous contribution(s) to the group key must not be tolerated even if it does not afford the adversary with any additional knowledge. Obviously, this can be achieved if the protocol is contributory and provides mutual authentication. Still it is arguable whether key integrity or its stronger definition is useful since it would suffice to require that the key is fresh as long as at least one contribution is fresh, independent of whether the adversary is able to inject own contributions or not. This is because we are not dealing with the secrecy of the key (this is already done in Sections 2.1 and 2.2) but with its freshness.

3 Analytical Survey on Formal Security Models

As already noted in the introduction provable security of cryptographic protocols can be achieved using an appropriate security model that considers protocol participants, their trust relationship, communication environment, and further relevant aspects, and contains definitions of required security goals.

Unfortunately, there exist no common *goodness* criteria for the evaluation of such security models. In our opinion a security model should be *abstract* meaning that it should not depend on any implementation-specific definitions or assumptions. Further, a model should be *self-contained*, i.e., there should be no parameters whose specification is not defined within the model or depends on certain assumptions beyond it. This property allows design of autonomous protocols. A model should be *precise*, i.e., it should disallow any ambiguous interpretations for its definitions and requirements. A security model should be *modular*, i.e., allow security proofs for protocols that provide only a subset of specified security goals. This property allows design of protocols with respect to their practical deployment in applications that do not require the full range of security; on the other hand it allows construction of generic solutions. Another advantage of modular security models is a possible integration of additional security definitions which may become essential in the future.

In the following we provide an analytical survey of security models proposed for group key exchange protocols. In addition to the description we specify which of the most important informal security requirements have been considered by the definitions of a model, thereby focusing on semantic security or indistinguishability of the computed group keys from random numbers considering known-key attacks and active adversaries, key confirmation and mutual authentication with respect to malicious protocol participants, (perfect) forward secrecy, and the issues related to key control and contributiveness. Additionally, we judge each model with respect to the specified goodness criteria. Some of the models described in this section were proposed in the context of two- or three-party key exchange protocols. However, they provide some interesting definitions and constructions that became foundational for a variety of the later appeared security models designed for the group setting.

3.1 Models by Bellare and Rogaway (BR, BR$^+$)

3.1.1 BR

Bellare and Rogaway [BR93a] proposed the first computational security model for authentication and security goals of two-party key exchange protocols which we refer to as the BR model. This model allows reductionist proofs of security. Each protocol participant is assumed to have an identity and a long-lived key. The adversary can initiate different sessions

between the same participants. It has an infinite collection of *oracles* $\Pi^s_{i,j}$ such that each oracle represents participant i trying to authenticate participant j in session s. The adversary communicates with the oracles via queries which contain sender and receiver identities, the session id, and the actual message. Hence, the adversary is considered to be active. However, the model assumes a *benign* adversary which faithfully forwards all message flows between the oracles, and is, therefore, not allowed to modify the messages. It can invoke any oracle to start the protocol execution. A variable $\kappa^s_{i,j}$ keeps track of the conversation between i and j in session s. The security goal of mutual authentication is defined based on the notion of *matching conversations* between the participants. Roughly, this means that all messages sent by one participant have been subsequently delivered to another participant without modification, and vice versa. According to the BR model a protocol provides mutual authentication if for any polynomial time adversary the oracles $\Pi^s_{i,j}$ and $\Pi^s_{j,i}$ have matching conversations and if one of the oracles, say $\Pi^s_{i,j}$, accepts (does not fail) then there is always another oracle $\Pi^s_{j,i}$ with the engaged matching conversation. Note that from this definition mutual authentication also results in key confirmation since the exchanged message flows, and thus computed keys, must be equal. The authors also show the uniqueness of matching partners. Additionally, the adversary is allowed to ask *Reveal* queries to obtain session keys computed by $\Pi^s_{i,j}$. In order to model known-key attacks the BR model specifies the notion of *fresh* oracles, i.e., an oracle $\Pi^s_{i,j}$ is fresh if it has accepted (computed the session key) and no *Reveal* query was asked to $\Pi^s_{i,j}$ or to its matching partner $\Pi^s_{j,i}$. Further, the adversary is allowed to ask exactly one *Test* query to any oracle which is fresh. In response to this query it receives either the real session key computed by the oracle or a random number of the same range and has to decide which value it has received. The BR model calls an authenticated key exchange protocol secure if in the presence of the benign adversary both oracles, $\Pi^s_{i,j}$ and $\Pi^s_{j,i}$, accept with the equal session keys which are randomly distributed over the key space, and the success probability of the adversary to decide correctly in response to the value obtained from its *Test* query is non-negligibly greater that $1/2$.

The main weakness of the BR model is that it disallows the adversary to modify messages, or to corrupt participants obtaining their long-lived keys. Hence, the model does not consider the notion of (perfect) forward secrecy. Further, the model does not deal with attacks of malicious participants. Thus, definition of mutual authentication is defined only for honest protocol participants, and no definitions concerning issues related to key control and contributiveness are available in the BR model.

3.1.2 BR$^+$

In their subsequent work, Bellare and Rogaway [BR95] extended the BR model to deal with key distribution scenarios in a three-party setting which involves two participants wishing to establish a shared key and a key distribution center (key server). Nonetheless, this model, denoted BR$^+$, is of particular interest for us since it provides some interesting definitions which are also relevant for the models dealing with key exchange protocols. The actions of the adversary are specified by a number of queries which it may ask to the *instances* of parties participating in the protocol. Each party may have a multiple number of instances and so participate in different sessions of the protocol. Using a *SendPlayer* or a *SendS* query the adversary can send messages to one of the participants or to the key distribution center, respectively, that reply according to the protocol specification or do not reply if the received message is unexpected. So the adversary is active. With a *Reveal* query to a specified instance the adversary may obtain the final key computed by that instance. Additionally, the adversary is allowed to ask *Corrupt* queries which return the complete internal state of the instance to the adversary together with the long-lived key of the party and allows the adversary to replace this long-lived key with any value of its choice. Note that this kind of corruptions became later known as *strong corruptions*. After an adversary asks a *Corrupt* query for some party all instances of this party use the changed value for the long-lived key in all subsequent protocol executions. Although the adversary is allowed to corrupt parties and reveal their session keys the security of the protocol may also depend on instances of other parties who participate in the same session. To consider all protocol participants the BR$^+$ model specifies an abstractly defined *partner function* which roughly speaking means that two instances are partnered if they participate in the same session and compute the shared key. Further, *at the end* of its execution the adversary asks a *Test* query to an instance which holds a *fresh session key*, i.e., a key computed during the session such that no *Reveal* or *Corrupt* queries have been previously asked to the instance or any of its partners. The adversary receives either the session key computed by that instance or a random number of the same range, and similar to the BR model must decide which value it has received. This security definition subsumes the informal requirement on indistinguishability of computed group keys from random numbers while also considering active adversaries.

The BR$^+$ model has some weaknesses described in the following. Although the adversary is allowed to reveal long-lived keys of participants through *Corrupt* queries it is allowed to ask its *Test* query only at the end of its execution. Obviously, the adversary may not use the knowledge of corrupted long-term keys to make its guess because of the freshness requirement. Therefore, the model does not capture the requirement of (perfect) forward

secrecy. Second, the BR$^+$ model does not deal with the attacks concerning key confirmation or mutual authentication, neither with respect to honest participants nor to malicious. This observation has also been mentioned in [CBH05a, CBH05b]. Third, the BR$^+$ model does not consider attacks related to the issue of key control. This, however, is reasonable since the model has been proposed for key distribution protocols for which such requirements are not relevant because of the trust assumption concerning the key server. The partnering function in the BR$^+$ model is not concretely specified. This contradicts to our goodness criteria for the security models.

3.2 Model by Bellare, Canetti, and Krawczyk (BCK)

Bellare, Canetti, and Krawczyk [BCK98] proposed a computational security model for authentication and key exchange, which we denote as the BCK model. This model allows security proofs based on the simulatability approach. The BCK model supports modular constructions and deals with *message-driven* protocols, i.e., after being invoked by a party the protocol waits for an activation which may either be caused by the arrival of a certain message or by an external request (which may come from other processes executed by the party). The authors essentially define two different adversarial models: *authenticated-links model* (AM) and *unauthenticated-links model* (UM). The AM model considers a passive adversary who has a full control over the communication channel but is assumed to deliver messages faithfully without modifying any of them, however, is allowed to change their delivery order. Further the adversary is allowed to activate any party using external requests, but not own protocol messages. The UM model assumes that the adversary is active, i.e., can activate parties with arbitrary incoming messages. Further, the BCK model specifies the notion of *emulation of protocols* in UM using a so-called authenticator which is considered to be a compiler translating the execution of a protocol in AM into UM while preserving its security requirements. Similar to the BR$^+$ model the BCK model considers several executions of the protocols, and calls each execution a *session*. In order to distinguish different sessions the model uses *session ids* which should be unique for the sender and the receiver (recall that the model was defined for two parties). The adversary in the AM and UM models is allowed to *corrupt sessions* such that it learns the internal state associated with a particular session identified via unique session IDs for which the model does not provide any concrete construction. Although the BCK model was proposed in the context of two-party key exchange protocols, the authors tried to provide definitions which also hold in a multi-party setting. They defined the notion of the *ideal key exchange* and the *ideal adversary*. The ideal adversary is allowed to invoke any party to establish the session key with any other party such that the adversary learns the transcript of the exchanged protocol messages and the session

id value, but not the established key. Further, if the ideal adversary corrupts a session using the corresponding session id then it obtains the established key for this session, and if the adversary corrupts a party then it obtains all keys (including long-lived key) known to this party. Corrupted parties may continue participating in the protocol. However, in this case the model allows the adversary to choose the established keys. Therefore, no security definitions related to the requirement on (perfect) forward secrecy are considered. For the same reason, the BCK model does not provide any security definitions for the case in which honest participants interacting with the adversary represented as a (subset of) malicious participant(s) try to contribute to the resulting group key. Hence, the BCK model does not consider attacks concerning key control and unpredictability. Also, the BCK model does not capture possible attacks of malicious participants against key confirmation and mutual authentication. Note that the corrupt query reveals internal state information together with the long-lived key of a party. This, however, disallows consideration of scenarios where long-lived keys are revealed without revealing the internal state information (note that long-lived keys may have a different protection mechanism). In order to define security goals the BCK model specifies the notion of a *global output* of the protocol execution in the presence of the adversary. It consists of cumulative concatenations of the outputs (sent messages) of all parties and their random inputs, together with the output of the adversary which is a function of its random input and all information seen by the adversary throughout its execution. A key exchange protocol is called secure in the BCK model if the global output of the ideal protocol execution is indistinguishable from the global output of the protocol execution in either the AM or UM model. Note that this is the typical approach for security models that allow security proofs based on simulatability/indistinguishability.

Interesting about the BCK model is that its modular construction allows to prove protocol security in the AM model and then apply the described authenticator to obtain a protocol which is secure in the UM model.

3.3 Model by Bellare, Pointcheval and Rogaway (BPR)

The following model proposed by Bellare, Pointcheval, and Rogaway in [BPR00], which we denote BPR, is based on the previously described BR$^+$ model, and considers two-party key exchange protocols. The BPR model is described w.r.t. the communication between a client and a server. Similar to the BR$^+$ model each participant may have different instances, called *oracles*. In addition to the queries *Reveal* and *Test* which return the computed key of the instance respectively the computed session key or a random number of the same range (in contrast to the BR$^+$ model the adversary in the BPR model may ask the *Test* query at any time during its execution and not only at the end), the BPR model specifies *Execute* and

Send queries. *Execute* queries can be used by the adversary to invoke an honest execution of the protocol and obtain a transcript of exchanged messages. The *Send* query allows the adversary to send messages to the instances, i.e., behave actively. Send queries can also be used to achieve honest execution (as *Execute* queries) simply by invoking the protocol execution at instances of adversary's choice and then forwarding messages between these instances without any modification. However, *Execute* query allows on the one hand a better handling of dictionary attacks in case where long-lived keys are shared passwords, because the adversary can be granted access to plenty of honest executions, and on the other hand it allows to treat passive adversaries separately though the BPR model does not take use of this second advantage. Additionally, the BPR model allows the adversary to ask *Oracle* queries in order to deal with non-standard models, like the Random Oracle Model (ROM) [BR93b] or the Ideal Cipher Model (ICM) [Sha49, Bla05]. In case when the protocol is designed to achieve security in the standard model the *Oracle* query can be omitted. The BPR model specifies two forms of a *Corrupt* query (unlike the previously described models): a strong form (called *strong corruption model*) means that the adversary obtains the long-lived key of the party and its internal state (excluding the session key), and a weak form (called *weak corruption model*) means that the adversary obtains only the long-lived key of the party.

The model assumes the existence of unique *session ids* and specifies *partner ids*. The partner id of an instance is a public value and consists of the identities of all parties with which the oracle believes it has just exchanged the session key with. According to the BPR model *two oracles are partnered* if they compute (accept with) equal session keys, equal session ids, have each other's identity as part of the computed partner ids, and there is no other oracle who accepts with the same partner id. The BPR model defines two flavors of session key freshness: a session key is *fresh* if there have been no *Reveal* queries to the oracle or any of its partners, and no *Corrupt* queries at all; a session key is *fs-fresh* (*fs* for forward secrecy) if there have been no *Reveal* queries to the oracle or any of its partners, and if there have been no *Corrupt* queries prior to the *Test* query such that further *Send* queries have been asked to the tested oracle. The latter means that the adversary can corrupt participants before the test session but is then not allowed to send any messages to the oracle whose key (or a random value instead) it later receives in response to its *Test* query. Based on these two flavors the model provides two definitions of security against known-key attacks: (1) security without forward-secrecy meaning that the adversary asks a *Test* query to an oracle which holds a fresh session key, and (2) security with forward-secrecy meaning that the adversary asks a *Test* query to an oracle which holds a fs-fresh session key. Similar to the BR and BR^+ models the goal of the adversary is to distinguish whether it obtains a session key or a random number. Obviously, this security definition subsumes the informal requirements

related to the indistinguishability of group keys from random numbers with respect to active adversaries. Furthermore, security proofs in which the *Test* query is asked to an oracle holds a fs-fresh session key consider attacks related to (perfect) forward secrecy.

Additionally, the BPR model gives a definition of *server-to-client* and *client-to-server authentication* which are violated in case where a server respectively a client accepts with a session key but does not have any partner. *Mutual authentication* is, therefore, violated if either a server-to-client or client-to-server authentication is violated (recall that the BPR model was proposed for the two-party key exchange protocols).

In the BPR model partnering is defined using session ids. However, the authors do not provide further details within the model concerning the construction of the unique session ids. In the proposed protocol, however, they are constructed as a concatenation of all flows exchanged between both participants. Note that this is an appropriate method in case of two parties, however, cannot be generally applied to the multi-party case where participants do not generally need to send each message to every other participant. Similar to the BR^+ model the BPR model does not consider attacks related to the issues of key control and contributiveness.

3.4 Model by Canetti and Krawczyk (CK)

Canetti and Krawczyk [CK01] proposed a formal model, which we refer to as CK, based on the methodology of the BR and BCK models. Similar to the BCK model the CK model deals with message-driven protocols that involve only two parties. Different executions of a protocol are called sessions which are identified by unique session ids. The CK model describes the notion of *matching sessions* (related to the matching conversations in the BR model), and treats participants of matching sessions as *partners*.

As the BCK model the CK model specifies an unauthenticated-links (UM) and an authenticated-links (AM) adversarial models. In the UM model the adversary passes messages from one participant to another, but has control over their scheduling (including initiation of the protocol), and is allowed to ask *Reveal* queries to obtain the computed session keys and *Corrupt* queries to obtain all the internal memory of the party including its long-lived key and specific session-internal information (such as internal state of incomplete sessions and session-keys of already completed sessions). From the moment a party is corrupted it is fully controlled by the adversary. This models attacks against (perfect) forward secrecy. Additionally, in the CK model the adversary is allowed to reveal internal state of a party for an incomplete session without necessarily corrupting that party (we call this *RevealState* queries). In the CK model a session becomes *locally exposed* if any of these three queries (*Reveal*, *RevealState*, or *Corrupt*) have been asked to a party during that session, and the session becomes *exposed*

if it or any of its matching sessions are locally exposed. Further, the CK model specifies *session expiration* which can be performed by a party causing the erasure of that session key and any session-specific information from the party's memory, and used in the model for the definition of security without (perfect) forward secrecy. In the AM model the adversary has the same capabilities as in the UM model, but is required to pass messages between the parties truly, i.e., without modifying them (this is comparable to a passive adversary who only eavesdrops the communication). The UM and AM models are linked together over the emulation paradigm based on authenticators as in the BCK model.

The CK model introduces the notion of *session-key security* as a security goal for the key exchange protocols. The definition in the UM model allows the adversary to ask a *Test* query (with similar response as in the BR, BR$^+$, and BPR models) to a party during the session which is completed, unexpired and unexposed at that time. Having asked this *Test* query the adversary is allowed to continue with regular actions according to the UM model but is not allowed to expose the test-session. At the end of its execution the adversary has to output a guess concerning the response of the *Test* query. A protocol is called *session-key secure* if for any UM-adversary holds that if any two parties complete matching sessions then they compute the same session key, and that the probability of the adversary's correct guess is no more than 1/2 plus a negligible fraction. This definition also captures informal requirements on indistinguishability notions considering known-key attacks and active adversaries. Additionally, the CK model provides a weaker definition of security for protocols in which no (perfect) forward secrecy is available or desirable. For this purpose the model disallows session expirations. A protocol is called *session-key secure without (perfect) forward secrecy* if it is session-key secure and the UM-adversary is not allowed to corrupt any partner from the test-session, i.e., the security of the session key can be no more guaranteed if any partner who computes this key gets corrupted. The authors mention that similar definitions are applicable for the AM model.

Compared to the BCK and BR models, the CK model provides a stronger adversarial setting since it allows *RevealState* queries. Interesting is that session-key security in the CK model implies the known-key security of the protocol in the BR$^+$ and BPR models. For the proof of this fact and further analysis of the relations between the security definitions in the BR, BR$^+$, BPR and CK models we refer to the work of Choo, Boyd and Hitchcock [CBH05b]. Note that one drawback of the CK model in the context of key exchange is that it leaves the construction of the session ids open, and this might have an impact on the security of the protocols designed based on this model. Also the CK model does not deal with the issues of key confirmation and mutual authentication as well as key control and contributiveness.

3.5 Model by Shoup

Shoup [Sho99] proposed a modular formal model for secure key exchange between two parties that can be seen as an extension of the BCK model. Shoup's model considers protocol composition, i.e., it contains security definitions for the case where the key exchange protocol should be used within another high-level application protocol. The model allows security proofs based on simulatability approach. It consists of two settings: the *ideal-world* model and the *real-world* model (this is somewhat similar to the ideal key exchange process in the BCK model). For each setting there exists a different adversary. For both the real-world and the ideal-world adversaries a transcript is generated which consists of all occurred events and messages. The security of the protocol for every real-world adversary means that there exists a corresponding ideal-world adversary, such that the generated transcripts are computationally indistinguishable. Shoup's model specifies three corruption settings: *static*, *adaptive*, and *strong adaptive* corruptions.

In case of *static corruptions* the ideal model allows the ideal adversary to initialize the party and its instance (oracle) using the identity of its partner, abort a session between two instances, invoke a session between two instances using a so-called *connection assignment* which specifies how the session key is computed, i.e., either honestly, or using the key of another connection, or chosen by the adversary (compromised). The latter is available only if this instance is not partnered with any other party's instance. As noted by Shoup, connection assignments are used instead of session ids to identify the connection between two parties. Further, the ideal-world adversary is allowed to call the *application* query which returns a partial information about the session key computed as a function within the application protocol. Additionally, the ideal-world adversary is allowed to ask an *implementation* query which simply adds the attached comment to the transcript. At the end of the adversary's execution a transcript which contains all actions (and outputs) taken by the adversary is generated. In the real-world model there is an additional trusted party which can be queried by the adversary in order to register the identities of the parties. However, the model requires that if a party is registered by the adversary then its identity should not be in the set of identities of the parties for which the adversary used its initialization query. So the adversary is able to obtain a long-lived key for a party which will not be initialized, and, therefore, will not participate actively in the honest protocol execution (this is the reason for the notion of static corruptions). In this case the model does not consider attacks concerning the interaction between honest and malicious participants, that are issues of key confirmation and mutual authentication, (perfect) forward secrecy, and key control. Instead of session invocations the real-world adversary is allowed to *deliver* messages (via a corresponding query) to the

instances who process them and return a protocol-specific output together with a status information which specifies whether the instance is waiting for further messages (*continue*), has already computed the session key (*accept*), or is finished without being able to compute the key (*reject*). At the end of the execution of the real-world adversary a transcript with all its actions is generated. For the setting of static corruptions Shoup specifies three security goals: *termination* (each instance either accepts or rejects after a polynomially bounded number of sessions), *liveness* (whenever the real-world adversary faithfully delivers all protocol messages both instances accept with the same session key), and *simulatability* (for every real-world adversary there exists a ideal-world adversary such that the transcripts of their actions are computationally indistinguishable).

In case of *adaptive corruptions* the model extends the requirements from the setting of static corruptions. The real-world adversary is additionally allowed to corrupt parties and reveal their long-lived keys before their initialization, and then register them by the trusted party. This enables the adversary to participate in the protocol on behalf of the corrupted user. The same operation is available to the ideal-world adversary, however, the adversary is allowed to choose a compromised connection assignment in its session invocation query to a party's instance either if this party is not partnered with any other party, or the party is partnered with another corrupted party, or the party itself is already corrupted. These changes capture the notions of key confirmation and mutual authentication, and (perfect) forward secrecy.

The setting of *strong adaptive corruptions* extends the setting of adaptive corruptions and considers a more powerful real-world adversary who obtains not only the long-lived keys, but also the internal state information if this has not been previously erased by the high-level application protocol. The same operation is given to the ideal-world adversary. This kind of corruptions is comparable to the strong corruption model of the BPR model.

Additionally, Shoup's model specifies similar security definitions for key exchange protocols between anonymous parties, i.e., parties whose identities are not registered with a trusted party. In these protocols the established key is used to derive a password which can be used for the purpose of authentication. Further, Shoup compares his model to the BR^+ and BCK models. This results in the equivalence of some security notions, like security against static and adaptive corruptions in the Shoup's and the BR^+ models.

As already noted Shoup's model is strongly focused on the composition of key exchange protocols with high-level application protocols. Therefore, it is unavoidable that some definitions of the model rely on the assumptions about the application protocol, e.g., computation and erasure of session keys. Further, security definitions in Shoup's model do not consider attacks related to key control and contributiveness.

3.6 Model by Bresson, Chevassut, Pointcheval, and Quisquater (BCPQ)

The formal model proposed by Bresson, Chevassut, Pointcheval and Quisquater [BCPQ01], which we refer to as the BCPQ model, is truly the first computational security model which has been designed for group key exchange protocols. The model allows reductionist security proofs and extends the methodology used in the BR, BR$^+$, and BPR models to a group setting. Similar to the mentioned models each protocol participant $U_i \in ID$[1], $i = 1, \ldots, n$ is modeled by an unlimited number of instances called *oracles* and denoted $\Pi_i^{s_i}$ (s_i-th instance of U_i) that can be involved in different concurrent protocol executions. Each user U_i is assumed to have a long-lived key LL_i (either symmetric or asymmetric). As in the BPR model the BCPQ model uses session ids to define the notion of partnering used in the definition of security goals. Unlike the BPR model which assumes the existence of unique session ids the BCPQ model describes their concrete construction. A session id of an oracle $\Pi_i^{s_i}$ is defined as $\mathrm{SID}(\Pi_i^{s_i}) := \{\mathrm{SID}_{ij} \mid U_j \in ID\}$ where SID_{ij} is the concatenation of all flows that $\Pi_i^{s_i}$ exchanges with another oracle $\Pi_j^{s_j}$. According to the BCPQ model two oracles $\Pi_i^{s_i}$ and $\Pi_j^{s_j}$ are called *directly partnered*, denoted $\Pi_i^{s_i} \leftrightarrow \Pi_j^{s_j}$, if both oracles *accept* (compute the session key) and if $\mathrm{SID}(\Pi_i^{s_i}) \cap \mathrm{SID}(\Pi_j^{s_j}) \neq \emptyset$. Further, oracles $\Pi_i^{s_i}$ and $\Pi_j^{s_j}$ are *partnered* if there exists a graph $G_{\mathrm{SIDS}} := (V, E)$ with $V := \{\Pi_l^{s_l} |\ U_l \in ID,\ l = 1, \ldots, n\}$ and $E := \{(\Pi_l^{s_l}, \Pi_{l'}^{s_{l'}}) |\ \Pi_l^{s_l} \leftrightarrow \Pi_{l'}^{s_{l'}}\}$ such that there exists a sequence of oracles $(\Pi_{l_1}^{s_{l_1}}, \Pi_{l_2}^{s_{l_2}}, \ldots, \Pi_{l_k}^{s_{l_k}})$ with $l_k > 1$, $\Pi_i^{s_i} = \Pi_{l_1}^{s_{l_1}}$, $\Pi_j^{s_j} = \Pi_{l_k}^{s_{l_k}}$, and $\Pi_{l-1}^{s_{l-1}} \leftrightarrow \Pi_l^{s_l}$ for all $l = l_2, \ldots, l_k$. This kind of partnering is denoted $\Pi_i^{s_i} \leftrightsquigarrow \Pi_j^{s_j}$. The BCPQ model uses graph G_{SIDS} to construct (in polynomial time $|V|$) the graph of partnering $G_{\mathrm{PIDS}} := (V', E')$ with $V' = V$ and $E' = \{(\Pi_l^{s_l}, \Pi_{l'}^{s_{l'}}) \mid \Pi_l^{s_l} \leftrightsquigarrow \Pi_{l'}^{s_{l'}}\}$, and defines the *partner id* for an oracle $\Pi_i^{s_i}$ as $\mathrm{PIDS}(\Pi_i^{s_i}) = \{\Pi_l^{s_l} \mid \Pi_i^{s_i} \leftrightsquigarrow \Pi_l^{s_l}\ \forall\ l \in \{1, n\} \setminus \{i\}\}$.

The adversary $\mathcal{A}$ in the BCPQ model is allowed to send messages to the oracles (and invoke the protocol execution) via *Send* queries, reveal the session key computed by the oracles via *Reveal* queries, obtain long-lived keys of the users via *Corrupt* queries (note that the oracle's internal state information is not revealed, that is similar to the weak-corruption notion in the BPR model), and ask a *Test* query to obtain either a session key or a random number. Using this adversarial setting the BCPQ model specifies two security goals for a group key exchange protocol: *authenticated key exchange (AKE) security* and *mutual authentication (MA) security*, both based on the notion of partnering.

For the AKE-security the model requires that during its execution $\mathcal{A}$ which is given access to the above mentioned queries asks a single *Test* query to an oracle which is fresh. An oracle Π_i^s is *fresh* if: (1) it has accepted, (2) neither Π_i^s nor any of its partners have been asked for a *Corrupt* query before Π_i^s accepts, and (3) neither Π_i^s nor any of its partners have

[1] ID is a set of n participants involved in the *current* protocol execution and is part of a larger set that contains all possible participants.

been asked for a *Reveal* query. A group key exchange protocol is said to be AKE-secure if the probability that $\mathcal{A}$ correctly guesses which value it has received in response to its $Test$ query, i.e., the session key or a random number, is negligibly greater than that of a random guess. This definition of AKE-security subsumes the informal security goals related to the indistinguishability of group keys from random numbers with respect to known group keys of other sessions in the presence of active adversaries, as well as the requirement of (perfect) forward secrecy.

The definition of MA-security in the BCPQ model is intended to capture the intuitive notion that it should be hard for a computationally bounded adversary $\mathcal{A}$ to impersonate any participant U_i through its oracle $\Pi_i^{s_i}$. For this purpose the authors require that the probability that during the execution of $\mathcal{A}$ which is given access to the above queries (thereby $Test$ query is irrelevant) there exists at least one oracle $\Pi_i^{s_i}$ which accepts with $|\text{PIDS}(\Pi_i^{s_i})| \neq n-1$ is negligible. In other words, if each participating oracle $\Pi_i^{s_i}$ has accepted with $|\text{PIDS}(\Pi_i^{s_i})| = n-1$ then no impersonation attacks could have occurred, thus the informal notion of mutual authentication meaning that each participating oracle is assured of every other oracle's participation in the protocol is satisfied. In the next paragraph we show, following Bresson, Manulis, and Schwenk [BMS07], that this is not generally the case, i.e., that there exists protocols where $\mathcal{A}$ impersonates U_i through some $\Pi_i^{s_i}$ but nevertheless all participating oracles accept and remain partnered, i.e., $|\text{PIDS}(\Pi_j^{s_j})| = n-1$ for every participating $\Pi_j^{s_j}$. Further, the authors claim

> 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. [BCPQ01, 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. [BCPQ01, Section 7.4]

If these claims hold then the above definition of MA-security captures further informal security goals related to key confirmation and mutual authentication (but only for honest protocol participants). In the next paragraph, following [BMS07], we explain that these claims do not hold for just any GKE protocol either. We show that an impersonation attack may likely result in different group keys accepted by different partnered oracles. In fact we are able to show that the definition of MA-security in the BCPQ model is not general enough to be used just for any GKE protocol, i.e., if in a GKE protocol every participating oracle $\Pi_i^{s_i}$ accepts

with $|\text{PIDS}(\Pi_i^{s_i})| = n-1$ then it does not necessarily mean that this protocol provides mutual authentication and key confirmation.

Additionally, we stress that the BCPQ model does not consider attacks aiming to reveal the internal state information (strong corruptions) and attacks of malicious participants aiming to control the resulting key value.

Problems with the definition of MA-Security in the BCPQ model Bresson, Manulis, and Schwenk provide in [BMS07] examples for the following two problems: (1) there exists 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 $|\text{PIDS}(\Pi_i^{s_i})| = n-1$ (the definition of MA-security in the BCPQ remains satisfied even though impersonation attacks have occurred); (2) there exists GKE protocols where each participating 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 (the definition of MA-security in the BCPQ remains satisfied even though some of the oracles complete with different group keys). Note that these problems become visible only in the group setting with at least three protocol participants. Therefore, it does not concern the original definition of mutual authentication given by Bellare and Rogaway [BR93a] based on matching conversations.

Prior to the examples based on a concrete GKE protocol we provide an abstract description of the flaw. Figure 1 shows the abstract messages denoted m_i (index i specifies the order in which messages have been sent) that have been exchanged between the oracles (at least three participants are required) during the honest execution of any GKE protocol from [BCPQ01, BCP01, BCP02a, BCP02b]. A concrete equivalent message of each abstract message m_i can be found in the corresponding *up-* or *downflow* stage of any of these GKE protocols. By m_i at the beginning of the arrow we mean the original message sent by the oracle, and by m_i at the end of the arrow we mean the corresponding message received by another oracle. If both messages are equal then the original message was not modified during the transmission.

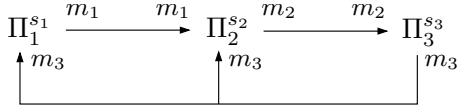

Figure 1: Example: Honest Execution of Protocols in [BCPQ01, BCP01, BCP02a, BCP02b]

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

Figure 2: Example: $\text{SID}(\Pi_i^{s_i})$ in the Honest Protocol Execution

Obviously, Figure 1 shows a correct execution of the protocol since there are no modified messages. Figure 2 specifies the session ids of the oracles $\Pi_1^{s_1}, \ldots, \Pi_3^{s_3}$ during this honest protocol execution using the construction from the BCPQ model. Under the assumption that

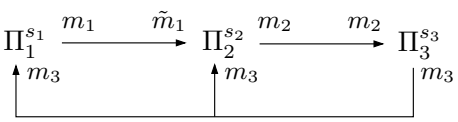

Figure 3: Example: Protocol Execution with Impersonation Attack

$\text{SID}(\Pi_i^{s_i})$	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)	$\emptyset$

Figure 4: Example: $\text{SID}(\Pi_i^{s_i})$ in the Attacked Protocol Execution

the protocol is correct each participating oracle $\Pi_i^{s_i}$ accepts with $|\text{PIDS}(\Pi_i^{s_i})| = 2$. To show the first problem consider the case where $\mathcal{A}$ impersonates U_1 and modifies message m_1 to $\tilde{m}_1$ (Figure 3) such that $\text{SID}_{21} = \tilde{m}_1$ (Figure 4). The goal is to show that nevertheless every participating oracle $\Pi_i^{s_i}$ accepts with $|\text{PIDS}(\Pi_i^{s_i})| = 2$. It cannot be generally assumed that all oracles accept after this modification but it can be assumed that there exists protocols where this is the case (the example later is such a protocol where the oracles nevertheless accept). With this assumption the first part of the goal, i.e., the acceptance of every participating $\Pi_i^{s_i}$, is satisfied. In order to show that $|\text{PIDS}(\Pi_i^{s_i})| = 2$ holds for every $\Pi_i^{s_i}$ one needs to show that $\Pi_i^{s_i} \leftrightsquigarrow \Pi_j^{s_j}$ (or $\Pi_i^{s_i} \leftrightarrow \Pi_j^{s_j}$) still holds for any two participating $\Pi_i^{s_i}$ and $\Pi_j^{s_j}$. For this purpose we need to look more precisely on the session ids of the oracles. 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 $\Pi_1^{s_1} \not\leftrightarrow \Pi_2^{s_2}$, there still exists a sequence of oracles $\Pi_1^{s_1}$, $\Pi_3^{s_3}$, $\Pi_2^{s_2}$ such that

$$\begin{aligned}\text{SID}(\Pi_1^{s_1}) \cap \text{SID}(\Pi_3^{s_3}) &= \\ &= \{m_1, m_3\} \cap \{m_3, m_2|m_3\} \\ &= m_3\end{aligned}$$

$$\begin{aligned}\text{SID}(\Pi_3^{s_3}) \cap \text{SID}(\Pi_2^{s_2}) &= \\ &= \{m_3, m_2|m_3\} \cap \{\tilde{m}_1, m_2|m_3\} \\ &= m_2|m_3\end{aligned}$$

so that $\Pi_1^{s_1} \leftrightsquigarrow \Pi_2^{s_2}$. Note also that these equations imply the direct partnering $\Pi_1^{s_1} \leftrightarrow \Pi_3^{s_3}$ and $\Pi_2^{s_2} \leftrightarrow \Pi_3^{s_3}$. Hence, every $\Pi_i^{s_i}$, $i \in \{1, 2, 3\}$ has $|\text{PIDS}(\Pi_i^{s_i})| = 2$. Thus, all oracles are still partnered though the impersonation attack occurred whereby $\Pi_2^{s_2}$ received a different message than the one originally sent by $\Pi_1^{s_1}$. This may result in different group keys computed by $\Pi_1^{s_1}$ and $\Pi_2^{s_2}$.

In order to illustrate the described attack on a concrete example consider the GKE protocol described in the same paper as the BCPQ model [BCPQ01] but without the additional confirmation round which the authors described independently of the protocol. Note that the additional confirmation round belongs to a concrete protocol design but not to a general

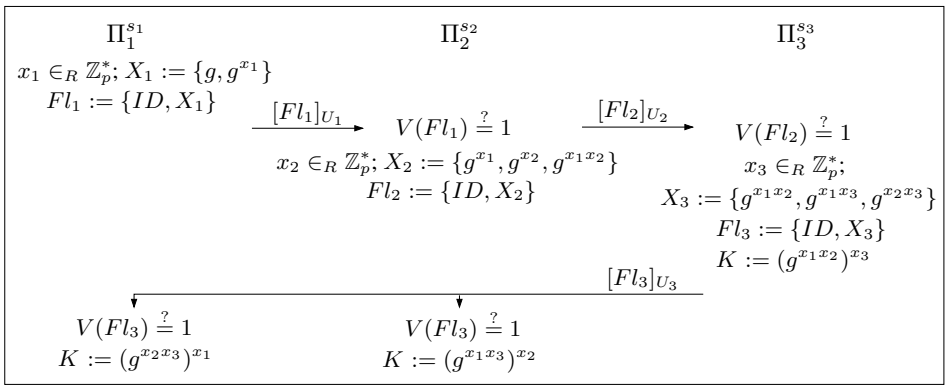

Figure 5: Example: Execution of the Protocol in [BCPQ01] with Three Participants

security model; otherwise the model cannot be applied to the protocols that do not have this round. Recall, the goal is to show that despite of the acceptance of each participating oracle $\Pi_i^{s_i}$ with $|\text{PIDS}(\Pi_i^{s_i})| = 2$ mutual authentication and key confirmation are not necessarily provided. The protocol proceeds as described in Figure 5 (for simplicity consider three participants). $[m]_{U_i}$ denotes a digital signature on m computed by the corresponding $\Pi_i^{s_i}$ (the signature is attached to m), and $V(m) \stackrel{?}{=} 1$ its verification; g is a generator of a cyclic group of prime order p. Upon computing $K = g^{x_1 x_2 x_3}$ each oracle derives the resulting group key $k := \mathcal{H}(ID, FL_3, K)$ with a cryptographic hash function $\mathcal{H} : \{0,1\}^* \to \{0,1\}^l$ where l is the security parameter. In order to apply the above attack consider that $\Pi_1^{s_1}$ chooses $x_1 \in \mathbb{Z}_p^*$ but $\mathcal{A}$ drops the original message $[Fl_1]_{U_1}$ and replays a corresponding message from some previous protocol execution (note $\mathcal{A}$ can invoke several subsequent protocol executions with the same ID via its *Send* query). The replayed message is likely to be $[\widetilde{Fl_1}]_{U_1}$ with $\widetilde{Fl_1} := (ID, \widetilde{X_1})$ where $\widetilde{X_1} := \{g, g^{\widetilde{x_1}}\}$ for some $\widetilde{x_1} \neq x_1$ (since each x_i is chosen at random for every new session). Obviously, $\Pi_2^{s_2}$ can still verify the replayed message, i.e., $V(\widetilde{Fl_1}) = 1$ 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., $K = g^{\widetilde{x_1} x_2 x_3}$. This also implies that the derived group keys are different. Note also that all oracles accept since all signature verifications remain correct. Beside that we have (similar to the abstract problem description

above)

$$\begin{aligned}
&\mathrm{SID}(\Pi_1^{s1}) \cap \mathrm{SID}(\Pi_3^{s3}) = \\
&\qquad = \{[Fl_1]_{U_1}, [Fl_3]_{U_3}\} \cap \{[Fl_3]_{U_3}, [Fl_2]_{U_2}|[Fl_3]_{U_3}\} \\
&\qquad = [Fl_3]_{U_3}
\end{aligned}$$

$$\begin{aligned}
&\mathrm{SID}(\Pi_3^{s3}) \cap \mathrm{SID}(\Pi_2^{s2}) = \\
&\qquad = \{[Fl_3]_{U_3}, [Fl_2]_{U_2}|[Fl_3]_{U_3}\} \cap \{[\widetilde{Fl_1}]_{U_1}, [Fl_2]_{U_2}|[Fl_3]_{U_3}\} \\
&\qquad = [Fl_2]_{U_2}|[Fl_3]_{U_3}
\end{aligned}$$

so that $|\mathrm{PIDS}(\Pi_i^{s_i})| = 2$ for every $\Pi_i^{s_i}$, $i \in \{1, 2, 3\}$. Thus, although all oracles accept with $|\mathrm{PIDS}(\Pi_i^{s_i})| = 2$ the protocol does not provide mutual authentication and key confirmation. This contradicts to the idea behind the definition of MA-security in the BCPQ model. Thus, it is not always true that if every $\Pi_i^{s_i}$ accepts with $|\mathrm{PIDS}(\Pi_i^{s_i})| = n - 1$ then mutual authentication and key confirmation are provided. Note that this is true if the protocol from [BCPQ01] is executed with the additional confirmation round, but there may exist other protocols (including the above example) for which this statement is not true (i.e., if MA-security is achieved by some other techniques). This shows that the definition of MA-security given in the BCPQ model is not generally applicable just for any GKE protocol.

Furthermore, a more general definition of MA-security should also consider possible attacks of malicious protocol participants those goal is to influence honest participants to come up with different group keys. As described in [CBH05a, CBH05b], not considering malicious participants is the reason why the BCPQ model and some of its later appeared variants do not capture unknown key-share attacks in their definitions. Note also that the construction of session ids based on concatenation of exchanged messages has one significant drawback - it becomes available only after the protocol is executed. However, some protocols use uniqueness of session ids as protection against replay and protocol interference attacks. In this case it is desirable to have a unique session id prior to the protocol execution.

3.7 Models by Bresson, Chevassut, and Pointcheval (BCP, BCP$^+$)

3.7.1 BCP

In their subsequent work, Bresson, Chevassut, and Pointcheval [BCP01] extend the BCPQ model to deal with dynamic group key exchange protocols where group membership may change during the protocol execution. We denote this extended model BCP. According to it a dynamic GKE protocol consists of an initialization algorithm executed for each participant,

a setup protocol between all founding group members for the initialization of the group and computation of initial session key, a join protocol executed between current group members and a set of joining members, and a remove protocol executed between the remaining group members after the exclusion of a subset of members from the group. The protocols for setup, join, and remove are called *operations*. The BCP model also specifies three additional queries, *Setup*, *Join*, and *Remove*, enabling an adversary to invoke the corresponding operation between the protocol participants. The BCP model defines AKE-security and MA-security as in the BCPQ model, i.e., based on the adversary's guess on the response of its *Test* query to a fresh oracle and based on the partnering condition, respectively.

The BCP model has the same drawbacks as the BCPQ model concerning the security of the protocol in case of the attacks that aim to reveal the internal state information of the oracles, attacks by malicious participants, and the identified problems with the definition of MA-security.

3.7.2 BCP$^+$

Bresson, Chevassut, and Pointcheval [BCP02a] revised their BCP model to cover attacks against the protocol based on the revealed internal state information of the oracles (strong corruptions). We denote this revised model as BCP$^+$. The model assumes that the security-relevant internal state information is maintained within a secure coprocessor, and that the long-lived keys of participants are stored within a smart card. Therefore, the model specifies additional queries which an adversary is allowed to ask, i.e., a $Send_c$ query which allows the adversary to communicate directly with the coprocessor, a $Corrupt_c$ query which reveals the private memory of the device together with all messages which have been exchanged between the coprocessor and the smart card, a $Send_s$ query which allows the adversary to communicate directly with the smart card, and a $Corrupt_s$ query which reveals the oracle's long-lived key.

Further, the BCP$^+$ model defines two flavors of forward secrecy: *weak forward secrecy* (wfs) and *strong forward secrecy* (fs). For the case of weak forward secrecy the BCP$^+$ model defines a *weak corruption model* in which the adversary is allowed to ask *Send*, *Setup*, *Join*, *Remove*, *Reveal*, and *Test* queries (all of which are answered as in the BCP model) as well as $Send_c$, $Send_s$, and $Corrupt_s$ queries. According to the weak corruption model an oracle is called *wfs-fresh* if no $Corrupt_s$ query has been asked by the adversary since the beginning of its execution, and in the execution of the current operation the oracle has accepted (holds the session key) and neither this oracle nor any of its partners (although the model does not specify the definition of partnering it seems to be the same as in the BCP model) have been asked for a *Reveal* query. The adversary must ask its *Test* query to an oracle which is wfs-fresh.

Consequently, for the case of strong forward secrecy the BCP$^+$ model defines a *strong corruption model* in which the adversary may additionally ask $Corrupt_c$ queries to obtain the internal state of the coprocessor and all messages which have been exchanged between the coprocessor and the smart card, and *Reveal* queries reveal not only the session key but also all messages which have been exchanged between the oracle and its secure coprocessor. An oracle is called *fs-fresh* if neither $Corrupt_c$ nor $Corrupt_s$ queries have been asked by the adversary since the beginning of its execution, and in the execution of the current operation the oracle has accepted and neither this oracle nor any of its partners have been asked for a *Reveal* query. In the strong corruption model the adversary must ask its *Test* query to an oracle which is fs-fresh.

The BCP$^+$ model defines AKE-security of a GKE protocol using the adversary's guess for the response to its *Test* query as in the BCP model but can be of two types with respect to the chosen corruption model. Hence, the BCP$^+$ model considers definitions of semantic security with respect to known-key attacks and active adversaries as well as (perfect) forward secrecy. The BCP$^+$ model does not explicitly specify MA-security, however, the authors mention that its definition can be taken from their BCP model. Although the model deals with the attacks concerning internal states of participants it still does not consider attacks of malicious participants, neither for MA-security nor for the issues of key control and contributiveness. It is also arguable whether the assumptions about the existence of smart cards and secure coprocessors for the protocol execution are of major importance and should be considered within an abstract model. The ability of the adversary to reveal the internal state information and the long-lived keys of participants can also be modeled by corresponding queries without these assumptions. This would also simplify the model.

3.8 Modifications of the BCPQ, BCP, and BCP$^+$ Models

In this section we describe some existing modifications of the models in Sections 3.6 and 3.7.

3.8.1 Modification by Bresson, Chevassut, and Pointcheval

In [BCP02b] the authors slightly modified the BCPQ model to be used with group key exchange protocols where authentication is achieved by the means of shared passwords. In addition to *Send*, *Reveal*, *Corrupt*, and *Test* queries the adversary is allowed to ask an *Execute* which models an honest protocol execution between the participants specified in the query, or in other words the protocol is executed in the presence of a passive adversary. This allows to provide tighter security proofs with respect to dictionary attacks because the number of *Send* queries which an adversary is allowed to ask and that it actually uses to

try own passwords is independent of the number of *Execute* queries for which the adversary obtains the transcript of an honest execution where participants use the shared password.

3.8.2 Modification by Katz and Yung (KY)

Katz and Yung [KY03] revised the BCPQ model from the perspective of static group key exchange protocols in which all messages are sent over a broadcast channel, i.e., received by all participants. Similar as the modification in [BCP02b] the authors consider an additional *Execute* query. However, the main difference to the BCPQ model is a different construction of session ids and partner ids. The session id of an oracle Π_i^s is simply the concatenation of all message flows that were sent or received by Π_i^s. Since each protocol message is received by every protocol participant it is clear that at the end of an honest execution all participants compute the same session id. The partner id of an oracle Π_i^s consists of the identities of participants with whom the oracle intends to establish the session key including U_i (note that in the BCPQ model U_i is not part of $\text{PIDS}(\Pi_i^s)$). Note that according to this definition partner ids are known in advance whereas in the BCPQ model they become known at the end of the protocol execution. Therefore, the equality of partner ids alone is not sufficient to decide whether oracles have actually participated in the same protocol session or not. For this reason Katz and Yung define two oracles as being *partnered* if they have equal partner ids and equal session ids. Also, Katz and Yung slightly modified the definition of the freshness of an oracle, i.e., an oracle Π_i^s is *fresh* if the adversary did not ask a *Send* query to Π_i^s or any of its partners after having corrupted U_i or any of its partners, and neither Π_i^s nor any of its partners have been asked for a *Reveal* query. Hence, this modification allows the adversary to corrupt a party but then the adversary is not allowed to participate in the protocol on behalf of the corrupted party. This does not give the adversary any advantage compared to the definition of freshness in the BCPQ model. For the definition of security of a protocol Katz and Yung consider a modular approach. They call a protocol: (a) a *secure group key exchange protocol* if a passive adversary which is not allowed to ask any *Send* queries (note that this is the reason for the additional *Execute* query) is successful in its guess concerning the *Test* query, and (b) a *secure authenticated group key exchange protocol* if the same holds for an active adversary. However, their modifications do not explicitly consider mutual authentication and key confirmation and also do not deal with the attacks of malicious participants concerning the issues of key control and contributiveness. Note also that the proposed construction of session ids can be used only in the group key exchange protocols where each message is sent over a broadcast channel, i.e., received by each other party. Thus, the model is not abstract enough.

3.8.3 Modification by Kim, Lee, and Lee

Kim, Lee, and Lee [KLL04] proposed a modification of the BCP model considering modifications done by Katz and Yung for the BCPQ model. In their model partner id of an oracle Π_i^s corresponds to the set of group members excluding the identity U_i, so that the partner ids are already known prior to the execution of the protocol. The proposed model specifies unique session ids for each oracle, however, they do not describe how these session ids are constructed. Obviously, they assume the same construction as in the modification by Katz and Yung. Two oracles are defined to be partnered if: (1) their session ids are equal, (2) each oracle's partner id consists of identities of all other group members' oracles, and (3) if the oracles compute equal session keys. Obviously, the latter requirement on the equality of computed session keys is somehow redundant, because if oracles compute equal session ids which are built by the concatenation of exchanged messages then they also compute equal session keys. Note also that this modification has similar limitations as the modification by Katz and Yung concerning protocols in which messages are not exchanged over a broadcast channel, and it also does not deal with attacks related to mutual authentication and key control.

3.8.4 Modification by Dutta, Barua, and Sarkar

Dutta, Barua, and Sarkar [DBS04] proposed another variant of the BCPQ model. Similar modifications have been later applied by Dutta and Barua [DB05b, DB05a, DB06] to the BCP model for dynamic protocols and to the model in [BCP02b] for the password-based authenticated protocols. The authors use the same construction of partner ids as Katz and Yung, however, they proposed a different construction of session ids. Instead of using the concatenation of exchanged messages the authors set the session id of an oracle Π_i^s to be a set of pairs $\{(U_1, s_1), \ldots, (U_n, s_n)\}$ where each pair (U_j, s_j), $j \in \{1, \ldots, n\}$ corresponds to the instance oracle $\Pi_j^{s_j}$ of the protocol participant U_j, and say that two oracles are partnered if they have equal partner ids and equal session ids.

In order to keep session ids unique the authors require the uniqueness of oracles for each new session. In [DB05a] the authors suggest to use a counter value as an additional parameter which should be increased for every new oracle of the user. Though this construction makes unique session ids available prior to the protocol execution it has the following weakness if used in the actual protocol implementation: the counter value must be saved after each execution of the protocol and, furthermore, it must be protected from manipulation; otherwise, an adversary may reset it to some previous value and cause impacts on the security of the protocol. A more practical approach seems to be using random values (nonces) for each

new initialization of the oracle. However, in this case one has to consider possible collisions between nonces used in different sessions. Note that the if session ids are not unique then an adversary may mount attacks based on interference of different sessions, e.g., replay attacks.

3.9 Models by Katz and Shin (KS, UC-KS)

Katz and Shin [KS05] proposed two different security models for GKE protocols: a computational model (referred to as the KS model), and a model in the framework of Universal Composability (UC) [Can01] (referred to as the UC-KS model). These models provide the first formal treatment of security of GKE protocols in the presence of malicious participants.

3.9.1 KS

The KS model is an extension of the BCPQ model and is also based on the modification of the latter by Katz and Yung [KY03]. However, Katz and Shin assume that unique session ids are already provided to the protocol by some high-level application protocol whereas the BCPQ model and the mentioned extension provide their own construction of the session ids. Partner ids of the oracles and the partnering relation are specified in the same way as proposed by Katz and Yung. The KS model allows the adversary to ask *Execute*, *Send*, *Reveal*, *Corrupt* and *Test* queries capturing the informal security definitions of semantic security with respect to known-key attacks and active adversaries as well as (perfect) forward secrecy. Similar to the BCP$^+$ model the KS model specifies two types of oracle freshness with respect to two different corruption models. In the *weak corruption model* an oracle Π_i^s is fresh if no *Reveal* query has been asked to Π_i^s or to any of its partners, and no *Corrupt* query has been asked to U_i or to any other participants in the session before the oracles have computed the session key and terminated. In the *strong corruption model* an oracle Π_i^s is fresh if no *Reveal* query has been asked to Π_i^s or to any of its partners, and no *Corrupt* query has been asked to a party whose identity belongs to the partner id of Π_i^s before Π_i^s has computed the session key and terminated. Thus, in the strong corruption model the adversary is allowed to corrupt partners of Π_i^s. Further, in the strong corruption model the *Corrupt* query returns not only the long-lived key of a party but also the internal state of any active oracle which belongs to this party. Based on these two corruption models the KS model defines *security of authenticated group key exchange* protocols based on the adversary's guess with respect to its *Test* query (similar to the AKE-security from the BCPQ model and its previously described variants).

Additionally, the KS model considers insider attacks executed by misbehaving, malicious participants. It defines a security goal called *agreement* such that an adversary *violates agreement* if there exist two oracles, Π_i^s and $\Pi_j^{s'}$, which are partnered and neither U_i nor

U_j are corrupted but Π_i^s and $\Pi_j^{s'}$ have accepted with different session keys. Intuitively, this considers key confirmation in case that all other participants are malicious (corrupted).

Further, the KS model says that $\mathcal{A}$ *impersonates* U_j *to* (accepted) Π_i^s if U_j is uncorrupted and belongs to the partner id of Π_i^s but in fact there exists no oracle $\Pi_j^{s'}$ which is partnered with Π_i^s. In other words, the oracle 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. Note that there are no assumptions about the corruption of other protocol participants. This is a subject of the following two different definitions of the protocol security against impersonation attacks. A protocol is secure against *outsider impersonation attacks* if there exists a party U_j and an oracle Π_i^s such that for any adversary $\mathcal{A}$ the probability that $\mathcal{A}$ impersonates U_j to Π_i^s and no parties which belong to the partner id of Π_i^s are corrupted before Π_i^s accepts is negligible. Thus, the adversary is not allowed to corrupt protocol participants during the execution of the protocol. Hence, an active adversary may try to inject messages on behalf of U_j or manipulate messages sent by valid protocol participants. Further, a protocol is secure against *insider impersonation attacks* if there exists a party U_j and an oracle Π_i^s such that for any adversary $\mathcal{A}$ the probability that $\mathcal{A}$ impersonates U_j to Π_i^s and neither U_j nor U_i are corrupted before Π_i^s accepts is negligible. Obviously, this (stronger) definition requires the existence of at least two uncorrupted protocol participants and allows the adversary to corrupt other participants. Intuitively, this requirement considers mutual authentication and unknown key-share resilience in the presence of malicious participants. Katz and Shin say that an authenticated group key exchange protocol is *secure against insider attacks* if it guarantees agreement and is secure against insider impersonation attacks.

3.9.2 UC-KS

The UC-KS model has a different concept (which is common for all UC-based models) that describes what an ideal GKE protocol execution is. Security proofs carried out in this model are based on the simulatability/indistingushability approach. This is different to the security proofs carried out in the game-based security models (like other mentioned variants of the BCPQ model) that define security from the perspective of adversarial games modeling certain security threats and attacks. Note that in addition to both models Katz and Shin proposed a provably secure compiler to turn any GKE protocol which is secure in the BCPQ (or more precisely in the KS) model into a protocol which is secure in their UC-based model. A drawback of the KS model is that its definitions of security against insider attacks do not capture the issues related to key control and contributiveness, which partially model missing trust relationship between participants of a GKE protocol, and should, therefore, be considered as one of the requirements related to insider attacks.

3.10 Model by Bohli, Vasco, and Steinwandt (BVS)

Bohli, Vasco, and Steinwandt [BVS05] proposed in their unpublished work an extension (which we refer to as the BVS model) of the BCPQ model and its modification by Katz and Yung towards security goals in the presence of malicious participants. Their definitions of session ids, partner ids, the notion of partnering, the adversarial queries, oracle freshness, and security of authenticated group key exchange are identical to those in [KY03]. Therefore, the BVS model captures indistinguishability of group keys from random numbers with respect to known-key attacks and active adversaries as well as (perfect) forward secrecy. Additionally, the BVS model defines a security goal called *session integrity* which is provided if all oracles of uncorrupted participants that have accepted with equal session ids hold identical session keys and partner ids which encompass the identities of all honest parties having accepted with the same session id. The second security goal defined by the BVS model is *strong entity authentication* to an oracle Π_i^s meaning that Π_i^s accepts and for all uncorrupted U_j which belong to the partner id of Π_i^s there exists an oracle $\Pi_j^{s'}$ which holds the same session id as Π_i^s and U_i belongs to the partner id of $\Pi_j^{s'}$. Intuitively, compared to the KS model the definition of session integrity is related to the definition of agreement, and the notion of *strong entity authentication* is similar to the security against impersonation attacks. Therefore, a combination of session integrity and strong entity authentication subsumes informal definitions of mutual authentication and key confirmation in the presence of malicious participants.

The BVS model also deals with the issues of key control and contributiveness. The adversary is given access to *Execute*, *Send*, *Reveal*, and up to $t-1$ *Corrupt* queries and has to output a tuple (i, s, χ_κ, a) where i and s correspond to an unused oracle Π_i^s (oracle is unused if it has not been initialized earlier) of an uncorrupted participant U_i, χ_κ is a boolean-valued algorithm with $\kappa := \{k \in \mathcal{K} |\ \chi_\kappa(k) = true\}$ such that $\mathcal{K}$ is a key space and $|\kappa|$ is polynomial in the security parameter, and a is some state information. Then on input a the adversary tries to make Π_i^s accept a session key $k \in \kappa$. In this second stage the adversary is allowed to ask *Execute*, *Send*, *Reveal*, and *Corrupt* queries, but is not allowed to corrupt U_i, and the total number of *Corrupt* queries in both stages should remain $\leq t-1$. The BVS model defines a group key establishment protocol as being *t-contributory* if the adversary succeeds with only negligible probability. In case that a protocol is n-contributory where n is a number of protocol participants then the BSV model calls it a *key agreement*. It is clear that the above definition enforces each participant to provide own contribution to the computation of the session key. Unfortunately the authors do not show feasibility of their contributiveness definition since their proofs are heuristic.

The BVS model has some drawbacks discussed in the following. First, the adversary is

required to commit to a certain oracle Π_i^s which remains uncorrupted and whose computation of the session key it tries to influence. Hence, the adversary is not adaptive in the sense that it can freely choose Π_i^s during the second stage of the attack. Second, the BCPQ model and consequently the BVS model disallows the adversary to reveal internal states of the oracles (strong corruptions). Therefore, the model considers neither (perfect) forward secrecy with respect to strong corruptions nor its definition of contributiveness does capture attacks aiming to influence the computation of the session key by Π_i^s using the knowledge of its internal state information but without corrupting U_i. Third, it is not clear how to specify the algorithm χ_κ, i.e., according to which criteria one can distinguish whether the key computed by Π_i^s is influenced by the adversary or is real in the sense of the protocol.

3.11 Models by Bresson, Manulis, and Schwenk (BMS, BM)

Bresson, Manulis, and Schwenk [BMS07] proposed refined definitions of AKE-security and a more general definition of MA-security that captures attacks of malicious participants. We call this model BMS. The definition of partnering is kept more general, i.e., the authors do not require any specific construction for the session ids except for their uniqueness. Two oracles are said to be partnered if they have identical session ids and their identities are part of each other's partner ids, which are defined similar to Katz and Yung [KY03]. Note that in [BMS07] oracles do not need to accept in order to be partnered.

As for the adversarial queries, the BMS model utilizes the *RevealState* query proposed by Canetti and Krawczyk for two-party key exchange protocols in [CK01]. This query reveals ephemeral secret information stored by oracles in their internal states without revealing the long-lived key. The motivation is that long-lived keys are more valuable and may experience better protection than session-related ephemeral secrets. Moreover, the same long-live key can be used in different applications where it can become leaked. The adversary is also given further classical queries, i.e., *Execute*, *Send*, *RevealKey* (which is identical to *Reveal* from [BCPQ01]), *Corrupt*, and *Test*.

The definition of AKE-security is refined in the BMS model in the sense that it captures five different types of secrecy, i.e., no secrecy, or weak/strong forward/backward secrecy depending on the restrictions for *RevealState* and *Corrupt* queries. For example, forward secrecy defined by the KY model [KY03] corresponds to weak forward secrecy in [BMS07], whereas strong forward secrecy is similar to that in the BCP$^+$ model [BCP02a]. Definitions of weak/strong backward secrecy are new and have been designed based on the ability of the adversary to ask *RevealState* queries that do not fully corrupt the participants. In this way the adversary may reveal internal states of participants prior to the protocol execution in order to obtain information which can be useful to attack secrecy in a later session (thus,

symmetrically opposed to forward secrecy). For example, protocols in which ephemeral secret information is pre-computed off-line for better performance are no longer AKE-secure in the sense of backward secrecy. Decision whether backward secrecy is weak or strong depends on the access to the *Corrupt* query which allows impersonation attacks. Strong backward secrecy cannot be achieved unless the long-lived key of a party is chosen anew [BCM06]. To the contrary weak backward secrecy can be achieved, e.g., through the *erasure technique* [CFIJ99] by which all ephemeral secrets used to compute the session group key are erased.

The BMS model extends the MA-security of the BCPQ model [BCPQ01] towards consideration of malicious participants aiming to include key confirmation, mutual authentication, and unknown key-share resilience required by [CBH05b, CBH05a]. The definition requires that at least two participants remain honest, but allows to reveal internal states of these participants through the *RevealState* queries. Note that the MA-security as defined in the BMS model can be seen alternative to the definitions of agreement and security against insider impersonation attacks from the KS model [KS05] since it captures the same informal security goals. Finally, the authors describe a compiler which adds AKE-, and MA-security to any GKE protocol which is secure against passive attacks whereby their definition of passive attacks is slightly different than the one used for the AKE-security compiler in [KY03]. In the BMS model the passive adversary is given additional power to drop, delay messages and deliver them out of order, but not to modify or inject new messages. The authors also show that without considering such stronger passive adversaries the compiler from [KY03] is not generic by providing a corresponding counter-example. The compiler described in [BMS07] can be seen as a combination of the modified AKE-security compiler from [KY03] and the compiler for security against insider attacks from [KS05].

In their parallel work, Bresson and Manulis [BM07] extended the BMS model towards consideration of key control and contributiveness issues (cf. Section 2.5). We call this extended model BM. The BM model contains definitions of three main security goals. The goals of AKE- and MA-security are similar to [BMS07] except that the authors do not use separation into weak forward and weak backward secrecy, but combine them by allowing the adversary to reveal ephemeral secrets of participants in earlier *and* later sessions, but not during the attacked session. The definition of MA-security is kept similar to the BMS model [BMS07] which already includes the case of malicious participants. The additional goal of the BM model is *contributiveness* and is defined by strengthening the definition proposed in the BVS model [BVS05] towards consideration of malicious participants and attacks that reveal ephemeral secrets via the *RevealState* query. The BM model allows the adversary to corrupt up to $n-1$ protocol participants and reveal the ephemeral secrets of all participants (including the uncorrupted ones) at any stage of the attack against the contributiveness of a GKE protocol.

The adversarial game consists of the two stages: `prepare` and `attack`. In the `prepare` stage the adversary outputs a value for the group key, and then in the `attack` stage tries to enforce at least one uncorrupted participant to accept that value as a group key. In particular, this game captures unpredictability of the accepted group keys and security against key replication attacks.

In addition to the definitions of AKE-/MA-security and contributiveness the authors provide informal arguments that their security definitions capture several informal security goals specified in [AST98], namely unpredictability of the group key, (verifiable) contributory group key agreement, and complete group key authentication. Finally, Bresson and Manulis propose a compiler which can be used to turn any AKE-secure GKE protocol into a protocol which provides MA-security and contributiveness. For the MA-security the authors use the ideas from [KS05, BMS07] whereas contributiveness is achieved through a novel iterative key derivation process which embeds public random nonces chosen by the protocol participants into the computation of the final group key in such a way that this group key remains fresh as long as at least one nonce is chosen truly at random. The initial iteration starts with the temporary group key computed by the underlying protocol, which is not supposed to be unpredictable. The public random nonces can, therefore, be seen as individual contributions to the final group key.

4 Summary and Discussion

In the following we summarize results of our analysis of currently known security models for group key exchange protocols. We focus on the BCPQ, BCP, BCP$^+$, KY, KS/UC-KS, BVS, and BM models while leaving out security models specified only for two or three protocol participants. We also do not consider variations of the above models in [BCP02b, KLL04, DBS04, DB05b, DB05a, DB06, BMS07] since these are minor modifications, mostly of technical nature, and without significant consequences for the actual security definitions. Note that almost all considered models (BCPQ, BCP, BCP$^+$, KY, KS, BVS, and BM) provide game-based definitions of security whereas UC-KS has been designed using the simulatability/indistinguishability-based approach. Still, the security requirements stated in the UC-KS model correspond to those stated in the KS model. Table 4 provides a comparison of these models. Columns two to five specify blocks of the most important informally defined security requirements from Section 2 whereby

- IND is the requirement on the indistinguishability of the group key computed in one session from a random value with respect to active adversaries and known group keys of other sessions,

Table 1: Analysis of Security Models for Group Key Exchange Protocols

Model		IND	MA	FS	CON	Strong Corr.	S/D
BCPQ	[BCPQ01]	+	H	+	-	-	S
BCP	[BCP01]	+	H	+	-	-	D
BCP$^+$	[BCP02b]	+	-	+	-	+	D
KY	[KY03]	+	-	+	-	-	S
KS/UC-KS	[KS05]	+	M	+	-	+	S
BVS	[BVS05]	+	M	+	+	-	S
BM	[BM07]	+	M	+	+	+	S

- MA is the requirement on mutual authentication between all participants of the group key exchange protocol that also subsumes the requirement on key confirmation and unknown key-share resilience (M points out that definitions of the model consider malicious participants; H points out that definitions of the model consider only honest participants;),
- FS is the requirement on (perfect) forward secrecy, in particular that IND still holds if the adversary is able to reveal the long-lived keys of all participants in later sessions,
- CON is the requirement on contributiveness of the group key exchange protocol, in particular that each participant equally contributes to the resulting group key and guarantees its freshness (this also subsumes the notion of key control and unpredictability).

Additionally, Table 4 specifies whether the considered model provides definitions with respect to strong corruptions, i.e., the adversary should be allowed to reveal internal (private) information of protocol participants used in the protocol execution. This may have consequences on the definitions of considering FS and CON. The last column provides the type of the protocol dynamics considered by the model (S for static, D for dynamic).

4.1 Informal Requirements

Obviously, all considered security models provide definitions that consider the requirements on indistinguishability of computed group keys (IND) and forward secrecy (FS). The requirement on mutual authentication and key confirmation has been found to be not general enough

in the definitions of the BCPQ and BCP models. Furthermore, these definitions do not consider attacks of malicious participants. The BCP$^+$ and KY models do not contain any definitions concerning MA-security. Only the KS/UC-KS, BVS, and BM models provide sufficient definitions concerning mutual authentication, key confirmation and unknown key-share resilience (MA) concerning attacks of malicious participants. Note that BVS and BM are the only models that consider issues concerning key control and contributiveness (CON).

4.2 Strong Corruptions

Only the BCP$^+$, KS/UC-KS, and BM models consider strong corruptions in their definitions of security goals. However, the BCP$^+$ and KS/UC-KS models address strong corruptions with respect to forward secrecy (FS). To the contrary the BM model extends this consideration towards mutual authentication and contributiveness, and also towards the backward secrecy (recall, there the ephemeral secrets may be leaked in earlier sessions too). The BVS model as an extension of the KY model does not deal with strong corruptions so that its definitions concerning key control and contributiveness are weaker than in the BM model.

4.3 Group Dynamics

Only the BCP model and its stronger variant BCP$^+$ provide definitions concerning dynamic group key exchange protocols. All other models focus on static GKE protocols. Note that dynamic protocols provide additional operations for the efficient update of the group key upon occurring changes of the group formation. Due to the risk that efficiency is achieved at the expense of weaker security it is important to consider these operations in the stated security requirements.

Acknowledgements

The author wishes to thank Emmanuel Bresson for his valuable input and useful discussions concerning the identified problems in the BCPQ model in Section 3.6. The author is also thankful to the anonymous referees at Indocrypt 2006 for their comments on the early draft of Section 3.6. Finally, the author is thankful to Jonathan Katz for the comments regarding the UC-KS model and to Jörg Schwenk for the proof-reading of the earlier draft of this work.

References

[AST98] Giuseppe Ateniese, Michael Steiner, and Gene Tsudik. Authenticated Group Key Agreement and Friends. In *Proceedings of the 5th ACM conference on Computer*

and Communications Security (CCS'98), pages 17–26. ACM Press, 1998. 10, 11, 12, 38

[BCK98] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. A Modular Approach to the Design and Analysis of Authentication and Key Exchange Protocols (Extended Abstract). In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing (STOC'98)*, pages 419–428. ACM Press, 1998. 16

[BCM06] Colin Boyd, Kim-Kwang Raymond Choo, and Anish Mathuria. An Extension to Bellare and Rogaway (1993) Model: Resetting Compromised Long-Term Keys. In *Proceedings of the 11th Australasian Conference on Information Security and Privacy (ACISP'06)*, volume 4058 of *Lecture Notes in Computer Science*, pages 371–382. Springer, 2006. 37

[BCP01] Emmanuel Bresson, Olivier Chevassut, and David Pointcheval. Provably Authenticated Group Diffie-Hellman Key Exchange - The Dynamic Case. In *Advances in Cryptology – ASIACRYPT'01*, volume 2248 of *Lecture Notes in Computer Science*, pages 290–390. Springer, December 2001. 25, 28, 39

[BCP02a] 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 *Lecture Notes in Computer Science*, pages 321–336. Springer, Mai 2002. 25, 29, 36

[BCP02b] Emmanuel Bresson, Olivier Chevassut, and David Pointcheval. Group Diffie-Hellman Key Exchange Secure against Dictionary Attacks. In *Advances in Cryptology – ASIACRYPT'02*, volume 2501 of *Lecture Notes in Computer Science*, pages 497–514. Springer, December 2002. 25, 30, 31, 32, 38, 39

[BCPQ01] Emmanuel Bresson, Olivier Chevassut, David Pointcheval, and Jean-Jacques Quisquater. Provably Authenticated Group Diffie-Hellman Key Exchange. In *Proceedings of the 8th ACM conference on Computer and Communications Security (CCS'01)*, pages 255–264. ACM Press, 2001. 23, 24, 25, 26, 27, 28, 36, 37, 39

[BD93] M. Burmester and Y. Desmedt. Towards Practical Proven Secure Authenticated Key Distribution. In *Proceedings of the 1st ACM Conference on Computer and Communications Security (CCS'93)*, pages 228–231. ACM Press, 1993. 10

[BD94] M. Burmester and Y. Desmedt. A Secure and Efficient Conference Key Distribution System. In *Advances in Cryptology – EUROCRYPT'94*, volume 950 of *Lecture Notes in Computer Science*, pages 275–286. Springer, May 1994. 6, 8, 10

[Bla05] John Black. The Ideal-Cipher Model, Revisited: An Uninstantiable Blockcipher-Based Hash Function. Cryptology ePrint Archive, Report 2005/210, 2005. http://eprint.iacr.org/2005/210.pdf. 18

[BM03] Colin Boyd and Anish Mathuria. *Protocols for Authentication and Key Establishment*. Springer, 2003. ISBN:3-540-43107-1. 10, 11

[BM07] Emmanuel Bresson and Mark Manulis. Malicious Participants in Group Key Exchange: Key Control and Contributiveness in the Shadow of Trust. In *Proceedings of the 4th Autonomic and Trusted Computing Conference (ATC 2007)*, volume 4610 of *Lecture Notes in Computer Science*, pages 395–409. Springer-Verlag, July 2007. 37, 39

[BMS07] Emmanuel Bresson, Mark Manulis, and Jörg Schwenk. On Security Models and Compilers for Group Key Exchange Protocols. In *Proceedings of the 2nd International Workshop on Security (IWSEC 2007)*, volume 4752 of *Lecture Notes in Computer Science*, pages 292–307. Springer-Verlag, October 2007. 24, 25, 36, 37, 38

[BPR00] Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated Key Exchange Secure Against Dictionary Attacks. In *Advances in Cryptology–EUROCRYPT'00*, volume 1807 of *Lecture Notes in Computer Science*, pages 139–155. Springer, May 2000. 17

[BR93a] Mihir Bellare and Phillip Rogaway. Entity Authentication and Key Distribution. In *Advances in Cryptology–CRYPTO'93*, volume 773 of *Lecture Notes in Computer Science*, pages 232–249. Springer, 1993. 10, 13, 25

[BR93b] Mihir Bellare and Phillip Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security (CCS'93)*, pages 62–73. ACM Press, 1993. 11, 18

[BR95] Mihir Bellare and Phillip Rogaway. Provably Secure Session Key Distribution: The Three Party Case. In *Proceedings of the 27th Annual ACM Symposium on Theory of Computing (STOC'95)*, pages 57–66. ACM Press, 1995. 15

[Bri99] Bob Briscoe. MARKS: Zero Side Effect Multicast Key Management Using Arbitrarily Revealed Key Sequences. In *Proceedings of the First International Workshop on Networked Group Communication (NGC'99)*, volume 1736 of *Lecture Notes in Computer Science*, pages 301–320. Springer, 1999. 7

[Bur94] M. Burmester. On the Risk of Opening Distributed Keys. In *Advances in Cryptology – CRYPTO'94*, volume 839 of *Lecture Notes in Computer Science*, pages 308–317. Springer, August 1994. 9, 10

[BVS05] 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/. 35, 37, 39

[BWJM97] Simon Blake-Wilson, Don Johnson, and Alfred Menezes. Key Agreement Protocols and Their Security Analysis. In *Proceedings of the 6th IMA International Conference on Cryptography and Coding*, volume 1355 of *Lecture Notes in Computer Science*, pages 30–45. Springer, December 1997. 10

[Can01] Ran Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *Proceedings of 42nd Annual Symposium on Foundations of Computer Science (FOCS 2001)*, pages 136–145. IEEE CS, 2001. 33

[CBH05a] Kim-Kwang Raymond Choo, Colin Boyd, and Yvonne Hitchcock. Errors in Computational Complexity Proofs for Protocols. In *Advances in Cryptology – ASIACRYPT'05*, volume 3788 of *Lecture Notes in Computer Science*, pages 624–643. Springer, 2005. 16, 28, 37

[CBH05b] 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 *Lecture Notes in Computer Science*, pages 585–604. Springer, 2005. 16, 20, 28, 37

[CFIJ99] Giovanni Di Crescenzo, Niels Ferguson, Russell Impagliazzo, and Markus Jakobsson. How to Forget a Secret. In *16th Annual Symposium on Theoretical Aspects of Computer Science (STACS'99)*, volume 1563 of *Lecture Notes in Computer Science*, pages 500–509. Springer, 1999. 37

[CK01] Ran Canetti and Hugo Krawczyk. Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In *Advances in Cryptology - EUROCRYPT'01*, volume 2045 of *Lecture Notes in Computer Science*, pages 453–474. Springer, 2001. 19, 36

[DB05a] Ratna Dutta and Rana Barua. Constant Round Dynamic Group Key Agreement. In *Information Security: 8th International Conference (ISC'05)*, volume 3650 of *Lecture Notes in Computer Science*, pages 74–88. Springer, August 2005. 32, 38

[DB05b] Ratna Dutta and Rana Barua. Dynamic Group Key Agreement in Tree-Based Setting. In *Proceedings of the 10th Australasian Conference on Information Security and Privacy (ACISP'05)*, volume 3574 of *Lecture Notes in Computer Science*, pages 101–112. Springer, 2005. 32, 38

[DB06] Ratna Dutta and Rana Barua. Password-Based Encrypted Group Key Agreement. *International Journal of Network Security*, 3(1):23–34, July 2006. Available at http://isrc.nchu.edu.tw/ijns/. 32, 38

[DBS04] Ratna Dutta, Rana Barua, and Palash Sarkar. Provably Secure Authenticated Tree Based Group Key Agreement. In *Proceedings of the 6th International Conference on Information and Communications Security (ICICS'04)*, volume 3269 of *Lecture Notes in Computer Science*, pages 92–104. Springer, 2004. 32, 38

[DH76] W. Diffie and M. E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, November 1976. 7, 8

[DvOW92] Whitfield Diffie, Paul C. van Oorschot, and Michael J. Wiener. Authentication and Authenticated Key Exchanges. *Designs, Codes and Cryptography*, 2(2):107–125, 1992. 8, 10, 11

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic Encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984. 9

[Gün90] Christoph G. Günther. An Identity-Based Key-Exchange Protocol. In *Advances in Cryptology – EUROCRYPT'89*, volume 434 of *Lecture Notes in Computer Science*, pages 29–37. Springer, 1990. 11

[HY98] Shouichi Hirose and Susumu Yoshida. An Authenticated Diffie-Hellman Key Agreement Protocol Secure Against Active Attacks. In *Proceedings of the First International Workshop on Practice and Theory in Public Key Cryptography (PKC'98)*, volume 1431 of *Lecture Notes in Computer Science*, pages 135–148. Springer, 1998. 6

[JT93] Philippe Janson and Gene Tsudik. Secure and Minimal Protocols for Authenticated Key Distribution. *Computer Communications*, 18(9):645–653, September 1993. 12

[KLL04] 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 *Lecture Notes in Computer Science*, pages 245–259, 2004. 32, 38

[KPT00] Yongdae Kim, Adrian Perrig, and Gene Tsudik. Simple and Fault-Tolerant Key Agreement for Dynamic Collaborative Groups. In *Proceedings of the 7th ACM Conference on Computer and Communications Security (CCS'00)*, pages 235–244. ACM Press, 2000. 9

[KPT01] Yongdae Kim, Adrian Perrig, and Gene Tsudik. Communication-Efficient Group Key Agreement. In *Proceedings of IFIP TC11 Sixteenth Annual Working Conference on Information Security (IFIP/Sec'01)*, volume 193 of *IFIP Conference Proceedings*, pages 229–244. Kluwer, 2001. 9

[KPT04] Yongdae Kim, Adrian Perrig, and Gene Tsudik. Tree-Based Group Key Agreement. *ACM Transactions on Information and System Security*, 7(1):60–96, February 2004. 9

[Kra05] Hugo Krawczyk. HMQV: A High-Performance Secure Diffie-Hellman Protocol. In *Advances in Cryptology – CRYPTO'05*, volume 3621 of *Lecture Notes in Computer Science*, pages 546–566. Springer, 2005. 12

[KS05] Jonathan Katz and Ji Sun Shin. Modeling Insider Attacks on Group Key-Exchange Protocols. In *Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS'05)*, pages 180–189. ACM Press, 2005. 11, 33, 37, 38, 39

[KY03] Jonathan Katz and Moti Yung. Scalable Protocols for Authenticated Group Key Exchange. In *Advances in Cryptology - CRYPTO'03*, volume 2729 of *Lecture Notes in Computer Science*, pages 110–125. Springer, 2003. 31, 33, 35, 36, 37, 39

[MvOV96] Alfred Menezes, P. van Oorschot, and Scott Vanstone. *Handbook of Applied Cryptography*. CRC Press, October 1996. ISBN:0-8493-8523-7. 5, 9, 10, 11

[MWW98] C. J. Mitchell, Mike Ward, and Piers Wilson. Key Control in Key Agreement Protocols. *Electronic Letters*, 34(10):980–981, 1998. 12

[MY99] Alain J. Mayer and Moti Yung. Secure Protocol Transformation via "Expansion": From Two-Party to Groups. In *Proceedings of the 6th ACM Conference on Com-*

puter and Communications Security (CCS'99), pages 83–92. ACM Press, 1999. 6

[NNL01] Dalit Naor, Moni Naor, and Jeffery Lotspiech. Revocation and Tracing Schemes for Stateless Receivers. In *Advances in Cryptology – CRYPTO '01*, volume 2139 of *Lecture Notes in Computer Science*, pages 41–62. Springer, 2001. 7

[Sha49] C. E. Shannon. Communication Theory of Secrecy Systems. *The Bell Systems Technical Journal*, 28(4):656–715, 1949. 18

[Sho99] 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/. 21

[SM03] Alan T. Sherman and David A. McGrew. Key Establishment in Large Dynamic Groups Using One-Way Function Trees. *IEEE Transactions on Software Engineering*, 29(5):444–458, 2003. 7

[SSDW90] D. G. Steer, L. Strawczynski, Whitfield Diffie, and Michael J. Wiener. A Secure Audio Teleconference System. In *Advances in Cryptology – CRYPTO'88*, volume 403 of *Lecture Notes in Computer Science*, pages 520–528. Springer, 1990. 8

[Ste02] Michael Steiner. *Secure Group Key Agreement*. PhD thesis, Saarland University, March 2002. 12

[STW98] Michael Steiner, Gene Tsudik, and Michael Waidner. CLIQUES: A New Approach to Group Key Agreement. In *Proceedings of the 18th International Conference on Distributed Computing Systems (ICDCS'98)*, pages 380–387. IEEE Computer Society Press, 1998. 9

[WCS+99] M. Waldvogel, G. Caronni, D. Sun, N. Weiler, and B. Plattner. The VersaKey Framework: Versatile Group Key Management. *IEEE Journal on Selected Areas in Communications*, 17(9):1614–1631, September 1999. 7

[WGL98] Chung Kei Wong, Mohamed Gouda, and Simon S. Lam. Secure group communications using key graphs. In *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM'98)*, pages 68–79. ACM Press, 1998. 6, 7

[WHA99] D.M. Wallner, E.J. Harder, and R.C. Agee. Key Management for Multicast: Issues and Architectures. Internet RFC/STD/FYI/BCP Archives, June 1999. RFC 2627. Available at http://www.faqs.org/rfcs/rfc2627.html. 6, 7

[YS90] Yacov Yacobi and Zahava Shmuely. On Key Distribution Systems. In *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 344–355. Springer, August 1990. 9