Formally and Practically Relating the CK, CK-HMQV, and eCK Security Models for Authenticated Key Exchange

Cas J.F. Cremers*

Department of Computer Science, ETH Zurich
8092 Zurich, Switzerland
cas.cremers@inf.ethz.ch

Abstract. Many recent protocols for Authenticated Key Exchange have been proven correct in the CK, CK-HMQV, or eCK security models. The exact relation between the security models, and hence between the security guarantees provided by the protocols, is unclear. We show that the CK, CK-HMQV, and eCK security models are not only formally incomparable but also practically incomparable, by providing for each model attacks on existing protocols that are not considered by the other models. We analyse a set of recent key exchange protocols with respect to the models. Finally, we present the first two-message authenticated key exchange protocol that is correct in all three models.

Keywords. Security Models, Authenticated Key Exchange, Session-state, Ephemeral-key, Perfect Forward Secrecy, Weak Perfect Forward Secrecy, Key Compromise Impersonation

1 Introduction

Authenticated Key Exchange (AKE) protocols form a crucial component in many network protocols. As such, they have been subject to increasing requirements in terms of efficiency as well as security. In terms of security, one of the goals has been to design protocols that are secure in the presence of the strongest possible adversaries.

During the last ten years, many newly proposed AKE protocols have been proven secure with respect to strong security models. Some examples are the 2-pass ISO signed Diffie-Hellman protocol in the CK model from [1], the HMQV protocol in a closely related model $\mathrm{CK}_{\mathrm{HMQV}}$ in [2], and the Naxos [3] and Okamoto [4] protocols in the eCK model [3]. The underlying idea is that the newer security models are stronger, and hence protocols proven in the newer models are at least as secure as the protocols proven in earlier models. However, given the subtle differences among the models, this conclusion is not obvious. In fact, many technical differences suggest that the models are formally incomparable. However, even if two models are incomparable for minor technical reasons, it may still be that one model is stronger than the other for all realistic protocols.

The fact that the relation between strong AKE security models has not been made precise, combined with the unproven assumption that some models are stronger than

* This work was partially supported by the Hasler Foundation within the ComposeSec project.

others in practice, hinders the objective comparison of the security properties of the various protocol proposals. We address this situation by relating three recent (and closely related) security models for indistinguishability-based proofs of AKE security that have been used for the analysis of a larger number of protocols. The results we establish refute several claims made previously in the literature.

Contributions. First, we show that the CK, CK_{HMQV}, and eCK models have different formal proof obligations. To show the practical differences, we present protocol attacks (several of which are unreported) that are possible in one model but not in the others, and vice versa. Therefore, the three models are not only *formally* but also *practically* incomparable.

Second, we analyze a set of recent two-message AKE protocols in the three models, leading to new insights regarding the security provided by the protocols.

Third, we present (to the best of our knowledge) the first 2-message AKE protocol that is correct in all three security models.

We proceed as follows. In Section 2 we introduce the ideas underlying indistinguishability-based AKE security models, and describe the eCK, CK and CK_{HMQV} models. In Section 3 we analyse recent two-message AKE protocols with respect to the models. Using the attacks from the analysis we relate in Section 4 the three models formally as well as practically. Additionally, we discuss possible practical interpretations of each model. Next we present a protocol that is correct in all three models in Section 5. We discuss related work in Section 6 and provide conclusions and future work in Section 7.

2 Three security models for Authenticated Key Exchange

In this section we describe the CK, eCK, and CK_{HMQV} security models for authenticated key exchange. In order to provide some context for the technical definitions, we first describe the general setup of indistinguishability-based security models for AKE, after which we present the three models in detail.

2.1 Indistinguishability-based security models for Authenticated Key Exchange

Security models for authenticated key exchange (AKE) models define properties of protocols. At a conceptual level, AKE security models contain at least three elements: the execution model, the property that should be satisfied, and the adversary model.

The *execution model* defines how protocols are executed, i. e., how the protocol is executed by regular participants. The execution model defines elements of protocol execution that are not mentioned in the protocol specification. For example, the details of session creation or session termination may involve setting up session identifiers, accepting or rejecting particular incoming requests, or erasing session state. Between AKE security models

there are many technical differences in the execution models that have implications for the judgements made on protocols.

The *security* property defines what the combined system, consisting of the interaction between participants and the adversary, should satisfy. In AKE security models the main property of interest is that the adversary should not be able to distinguish the session key from a random bit string.

The *adversary model* describes the capabilities of the adversary, in whose presence the protocol should satisfy the security property. We assume that the adversary has complete control over the network and can eavesdrop, remove, or insert messages. The models differ in the additional powers attributed to the adversary, which include revealing some long-term or session keys, revealing the random numbers generated by participants, or revealing parts of the session-state of some sessions.

Besides the three aspects described above, AKE security proofs often involve additional parameters. Proofs may e. g. rely on various assumptions such as the computational or the decisional Diffie-Hellman assumption. Furthermore, some protocol proofs assume the Random Oracle Model, others the so-called standard model.

2.2 Preliminaries

A protocol consists of two or more roles, such as initiator, $\mathcal{A}$, or responder, $\mathcal{B}$. We assume any number of participants (A, B, ...) execute role instances. We call each such instance of a protocol role a *session*. Participants can execute multiple sessions concurrently.

During a normal protocol run (without adversary interference) between two participants A and B, there is a session at A and a session at B. For AKE models, we require that both sessions compute the same session key. The notion of *matching sessions* (sometimes called *partnering*), as described in detail for each model separately below, aims to make precise when two sessions are partners.

For technical reasons, the security definitions involve the notion of a particular protocol instance referred to as the *test session*, for which the indistinguishability of the established key is proven. This test session represents an arbitrary session, but is given a particular identifier to be able to formulate the capability of the adversary to reveal e. g. session keys of other sessions.

2.3 The eCK security model

We now present the eCK security model as defined in [3]. A unique feature of the eCK model is that it includes an adversary that, under some restrictions, can reveal part of the state of the test session. In particular, the *ephemeral keys* can be revealed. The "ephemeral key" terminology stems from the use of randomness for generating the short-term key pair in a Diffie-Hellman style protocol.

Definition 1 (Session identifier). *The session identifier of a session sid is defined as the tuple* $(role, ID, ID*, comm_1, \ldots, comm_n)$*, where role is the role performed by the session (here initiator or responder), ID is the name of the participant executing sid,* $ID*$ *the name of the intended communication partner, and* $comm_1, \ldots, comm_n$ *the list of messages that were sent and received.*

Definition 2 (Matching sessions for two-party protocols). *For a two-party protocol, sessions sid and* sid' *are said to match if and only if there exist roles* $role, role'$ $(role \neq role')$*, participants* ID, ID'*, and message list* $L = comm_1, \ldots, comm_n$*, such that the session identifier of sid is* $(role, ID, ID', L)$ *and the session identifier of* sid' *is* $(role', ID', ID, L)$*.*

Definition 3 (Ephemeral keys). *The ephemeral keys of a protocol are defined as the random values generated during the execution of the protocol.*

Definition 4 (*clean* for eCK). *In an AKE experiment (e. g. as defined in Definition 5 below), let sid be a completed AKE session performed by A, supposedly with some party B. Then sid is said to be* clean *if all of the following conditions hold:*

1. *A and B are not adversary-controlled, i. e., the adversary does not choose or reveal both the long-term and ephemeral keys of the participant and performs on its behalf.*
2. *The experiment does not include* Reveal(sid)*, i. e., the session key of session sid is not revealed.*
3. *The experiment does not include both* Long-term Key Reveal(A) *and* Ephemeral Key Reveal(sid)*.*
4. *If no session exists that matches sid, then the experiment does not include* Long-term Key Reveal(B)*.*
5. *If a session* $sid*$ *exists that matches sid, then the experiment does not include:*
   (a) Reveal$(sid*)$*, i. e. the session key of session* $sid*$ *is not revealed, and*
   (b) both Long-term Key Reveal(B) *and* Ephemeral Key Reveal$(sid*)$*.*

Definition 5 (AKE security experiment for eCK). *In the eCK AKE security experiment, the following steps are allowed:*

1. *The adversary may perform* Send$(A, B, comm)$*,* Long-term Key Reveal(A)*, and* Reveal(sid) *queries as in [3].*
2. *The adversary may perform an* Ephemeral Key Reveal(sid) *query, revealing the ephemeral keys of the session sid.*
3. *The adversary performs a* Test(sid) *query on a single* clean *session sid. A coin is flipped:* $b \xleftarrow{\$} \{0, 1\}$*. If* $b = 0$*, the test query returns a random bit string. If* $b = 1$*, the query returns the session key of sid. This query can be performed only once.*
4. *The adversary outputs a* Guess(b') *bit* b'*, after which the experiment ends.*

An adversary $\mathcal{M}$ wins *the experiment if the* Guess(b) *bit b is equal to the bit* b' *from the* Test(b') *query.*

Definition 6 (eCK security). *The advantage of the adversary $\mathcal{M}$ in the eCK AKE experiment with AKE protocol Π is defined as* $\mathbf{Adv}_{\Pi}^{AKE}(\mathcal{M}) = Pr[\mathcal{M}\ wins] - \frac{1}{2}$. *We say that an AKE protocol is secure in the eCK model if matching sessions compute the same session keys and no efficient adversary $\mathcal{M}$ has more than a negligible advantage in winning the above experiment.*

2.4 The CK security model

The CK model [1] is a predecessor of the other two models. For our results, we do not need to consider the full details of CK and only describe the main differences with respect to the eCK model.

Definition 7 (CK security). *The CK security model is similar to the eCK model, except for the following main differences.*

1. *The CK model has no* Ephemeral Key Reveal *query. Instead, the CK has a similar query called* Session-state Reveal. *The* Session-state Reveal *query reveals the contents of the session state. The contents of the session state are not specified within the CK model, but are assumed to be defined separately for each protocol. Therefore the contents of the session state can either be equal to, or differ from, the ephemeral keys that are revealed by the similar* Ephemeral Key Reveal *query in the eCK model.*
2. *The* Session-state Reveal *query may only be performed on a session that is not the test session or its matching session. In contrast, the* Ephemeral Key Reveal *query in the eCK model may also be performed on the test session or its matching session.*
3. *The long-term private key of the participant that executes the test session may not be revealed before the end of the test session. In the eCK model, the long-term private key of the participant that executes the test session may be revealed.*
4. *If A executes the test session, supposedly with B, and no matching session for the test session exists, then the long-term private key of B may be revealed after the test session. In the eCK model the long-term private key of B may not be revealed in this case.*
5. *The notion of matching sessions, as used for specifying the adversary capabilities, is defined as sessions having the same session identifiers. These session identifiers are assumed to be externally provided to the security model, by e. g. the application layer.*

In existing proofs in the CK model such as for the signed Diffie-Hellman protocol, the session state is defined as the ephemeral keys. In general, the long-term private keys are not considered to be part of the session state in the CK model, as this would trivially allow for violation of the security property.

Point 3 relates to the ability of the security model to detect so-called Key Compromise Impersonation (KCI) attacks [5]. Protocols that are proven to be CK-secure may still be vulnerable to such attacks.

Point 4 relates to Perfect Forward Secrecy (PFS). Protocols proven secure in the CK model satisfy PFS. In contrast, the eCK model only guarantees the strictly weaker notion of Weak Perfect Forward Secrecy (WPFS) [2].

2.5 The $\mathbf{CK_{HMQV}}$ security model

Definition 8 ($\mathbf{CK_{HMQV}}$ security). *The CK_{HMQV} security model is similar to the CK model with the modifications described in [2]:*

1. *As in the eCK model, the long-term private key of the participant that executes the test session, may be revealed before the end of the test session.*
2. *As in the eCK model, If A executes a test session, supposedly with B, for which no matching session exists, then the long-term private key of B may not be revealed after the test session.*
3. *The definition of matching sessions is related to that of Def. 2 except that the role is omitted from the session identifier and the messages are ordered differently, i. e., in Def. 1 the session identifier consists of the tuple $(ID, ID*, comm, comm*)$ where ID is the participant name and comm the message it sent. The starred variants are respectively the partner name and the received message. A session $(ID, ID*, comm, comm*)$ matches with the session $(ID*, ID, comm*, comm)$.*

Changing the definition of matching sessions is relevant for protocols that have symmetrical roles such as MQV [6], HMQV [2], and CMQV [7]. As we will see in the next section, such protocols allow for the following behaviour: A and B both execute an initiator session with the other, and their messages cross, after which they compute the same session key. According to the matching sessions definition of $\mathrm{CK_{HMQV}}$ they will be partners. In contrast, these sessions are not considered partners in the CK and eCK models. Thus, if the session of A is the test session, the adversary is capable of revealing the session key of B's session.

Table 1 provides an overview of adversary capabilities which differ among the models. We will use this table in Section 4 when explaining the formal difference between models.

3 Analysis of two-message AKE protocols

In this section we analyse a number of recent two-message AKE protocols in the three security models. We combine known results with a number of previously unreported attacks.

In the Venn diagram in Figure 1 each circle represents a security model. Within the circles, protocols are listed that are correct in the corresponding security model. For example, the 2DH-ISO [1] protocol is correct in the CK and $\mathrm{CK_{HMQV}}$ models, but admits an attack in the eCK model. We motivate the placement of each protocol below, except for "protocol 1" that we present in Section 5.

We start with a basic protocol. At the core of the signed **2DH-ISO** protocol [1] is a Diffie-Hellman exchange where the messages are signed using the long-term key of the

Table 1. Adversarial capabilities in the CK [1], CK_{HMQV} [2], and eCK [3] security models. Assume A executes the test session, supposedly with B.

Capability	Description	CK	CK_{HMQV}	eCK
C1	Session-state/Ephemeral-key reveal contents can differ from ephemeral keys.	Yes	Yes	No
C2	Session-state/Ephemeral-key reveal may be performed on test or matching session.	No	No	Yes
C3	Long-term key reveal of A possible before end of test session (part of checking for KCI.)	No	Yes	Yes
C4	Long-term key reveal of B possible even if no matching session exists (part of checking for PFS.)	Yes	No	No
C5	Reveals on sessions that compute the same session key, but do not have matching histories, are possible (relates to the matching sessions definition in the CK_{HMQV} model.)	Yes	No	Yes

sender. As role names are included in the messages, and the first message cannot be mistaken for the second, the resulting protocol can be proven correct in the CK model [1]. The use of signatures in all messages prevents key compromise impersonation. As a result, the protocol seems also correct in the CK_{HMQV} model. The signed 2DH-ISO protocol is not correct in the eCK model as revealing the ephemeral keys of the test session trivially allows reconstruction of the session key [3]. A variant of the protocol in which the signatures are replaced by MACs based on the long-term symmetric keys shared between the participants, denoted by **2DH-ISO-MAC**, is also correct in CK but is vulnerable to key compromise impersonation attacks, and is therefore not correct in the CK_{HMQV} model.

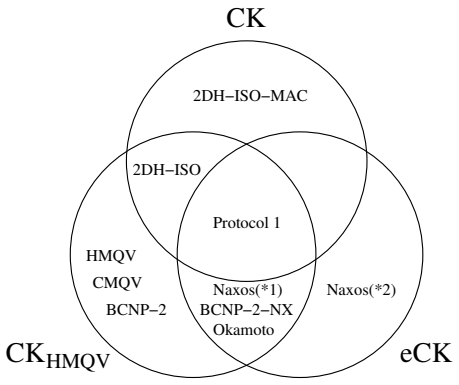

Fig. 1. Security of two-message protocols

The **Naxos** protocol [3] was proven correct in the eCK model. In [8] it was shown that if the session-state includes the inputs to the final step of the key derivation, Naxos is vulnerable to a state-reveal attack, and hence not correct in the CK and CK_{HMQV} models. To distinguish between two possible definitions of the session-state in Naxos, we write **Naxos(*1)** to denote the protocol where the session state contains only the ephemeral keys. We write **Naxos(*2)** to denote the version of the Naxos protocol where the session-state includes the inputs to the final key derivation step (applying a hash function to a concatenation of messages.) For full details we refer

the reader to [3, 8]. Both versions of the Naxos protocol do not satisfy Perfect Forward Secrecy, as the adversary can generate an arbitrary value x and send g^x to B, as if it were sent by A. B cannot distinguish g^x from what would have been sent by the real A, namely, $g^{H_1(esk_A, sk_A)}$. After B sends a response, the session of B ends, and the adversary can compromise the long-term of A, and reconstruct the key. As a result, Naxos is not correct in the CK model. This attack is a variant of the attack scheme on Perfect Forward Secrecy described in [2], and is described in detail in Appendix A.

The **BCNP-2** protocol [9] was proven secure in a variant of the CK_{HMQV} model, under the assumption that the ephemeral secrets are *not* in the session state. Furthermore, it was conjectured in [9] that a transformation[1] of the protocol along the lines of the mechanism used in Naxos [3] would cause the resulting protocol to be correct in the eCK model. We denote this protocol by **BCNP-2-NX**. Both protocols are incorrect in the CK model because they do not provide PFS.

The protocols **MQV** [6] and **HMQV** [2] are closely related. HMQV can be viewed as a "cleaned up" version of MQV and was proven correct in CK_{HMQV} under the assumption that the session-state contains only the ephemeral keys. As pointed out in [10], both protocols allow symmetrical behaviour: two initiators may start the protocol with each other and establish the same session key, even though they are executing the same role. These sessions are not partners according to the CK and eCK models. If one of the sessions is the test session, then the adversary can trivially reveal the session key by revealing the other initiator's session key. Strictly speaking this leads to attacks in the CK and eCK model. Furthermore, both protocols only provide weak perfect forward secrecy, and are also for that reason incorrect in CK.

The **CMQV** protocol [7] is a variant of HMQV, adapted with a transformation similar to the one used in Naxos. CMQV was proven secure in an eCK-like model that uses the session identifier definition from the CK_{HMQV} model. However, because of symmetrical behaviour similar to (H)MQV, the protocol is strictly speaking not correct in the eCK model. When two initiators establish the same key, they are not considered partners in the original eCK model, allowing a session-key reveal attack.

The **Okamoto** protocol [4] is proven correct in the eCK model. Assuming that its session state contains exactly the ephemeral keys, it is also correct in the CK_{HMQV} model. However, the protocol does not satisfy Perfect Forward Secrecy and is therefore incorrect in the CK model. An attack on PFS resembling the attack on Naxos is described in detail in Appendix B.

4 Relating the CK, CK_{HMQV}, and eCK security models

The outcome of our analysis is that the three models defined in Section 2 are formally incomparable, i. e., security in one security model does not imply security any of the other

[1] The transformation used in Naxos consists of replacing each usage of a random value x of a principal A by $H(sk_A, x)$, where H is a hash function and sk_A is the long-term private key of A.

two models. Additionally, we show practical incomparability, i. e., we show that in each model, attacks on protocols from literature exist that are not considered by the other models. Our formal comparisons are based on differences in proof obligations, summarized in Table 1, and our practical comparisons build on the attacks described in the previous section.

4.1 $\mathrm{CK}_{\mathrm{HMQV}}$ security does not imply CK security

Additional proof obligations: C4 and C5.
Attack example. As pointed out by its author, the (two-message) HMQV protocol from [2] is not correct in the CK model because it does not satisfy Perfect Forward Secrecy because of a generic attack described in [2]. The $\mathrm{CK}_{\mathrm{HMQV}}$ model was designed not to consider this type of attack, such that a particular class of protocols can be proven secure in the $\mathrm{CK}_{\mathrm{HMQV}}$ model. With respect to C5, a key-reveal attack of this type was pointed out for HMQV in [10].

4.2 CK security does not imply eCK security

Additional proof obligations: C2 and C3.
Attack example. An example using C2 is described in [3]. The basic signed two-pass Diffie-Hellman protocol provides CK- and $\mathrm{CK}_{\mathrm{HMQV}}$-security, but is subject to a straightforward attack if the adversary learns the ephemeral key of one of the participants by means of an Ephemeral Key Reveal query. This allows the adversary to compute the session key.

4.3 eCK security does not imply $\mathrm{CK}_{\mathrm{HMQV}}$ security

Additional proof obligations: C1. Note that if the session state contains only the ephemeral keys, then security in eCK does imply security in $\mathrm{CK}_{\mathrm{HMQV}}$.
Attack example. An example attack in $\mathrm{CK}_{\mathrm{HMQV}}$ on the Naxos protocol [3], which was proven correct in the eCK model, is described in [8]. The attack occurs when the session state of the protocol contains the inputs to the hash function H_2 that is used in the final step of the session key computation.

4.4 CK security does not imply $\mathrm{CK}_{\mathrm{HMQV}}$ security

Additional proof obligations: C3.
Attack example. The 2DH-ISO-MAC protocol, as described in the previous section, is correct in CK but vulnerable to key compromise impersonation: if the adversary learns the long-term keys used by A, he can impersonate any agent to A.

4.5 CK_{HMQV} security does not imply eCK security

Additional proof obligations: C2 and C5.
Attack examples. Attacks exploiting C2 exists for any basic two-message signed Diffie-Hellman exchange, as pointed out in [3]. If the adversary reveals the ephemeral private key of the test session or its matching session, the session key can be computed.

An attack on the HMQV protocol that exploits C5 was presented in [10] and was described in the previous section. Two sessions, both executing the initiator role, accept each other's initial message as the response message. As a result, they compute the same session key. However, they are not partners in the eCK model and hence the adversary can reveal the session key of one session even if the other session is the test session.

4.6 eCK security does not imply CK security

Additional proof obligations: C1 and C4.
Attack example. An example involving capability C1 can be found in Section 4.2. With respect to C4, we observe that the Okamoto protocol [4] does not satisfy Perfect Forward Secrecy, as described in the previous section.

4.7 Possible practical interpretations of the security models

The Ephemeral-key Reveal query, which is the only query unique to the eCK model, corresponds to an adversary capable of learning the ephemeral key after it was generated (but not any other elements of the state) of any session. A corresponding practical scenario is a random number generator (RNG) that leaks values upon generation. This may be due to the fact that the values can be retrieved, e. g., by eavesdropping communications or side-channel attacks. The RNG is not malicious in the sense that values can be manipulated, i. e., the adversary cannot choose the values. Furthermore, the RNG is also not predictable, because the adversary can only learn the ephemeral keys after they have been generated.

The Session-State reveal query of the CK and CK_{HMQV} models allows the adversary to learn part of the session state. Two elements of the definition are that (1) the session state contents should not reveal the long-term keys of the participant, and (2) the adversary only passively learns the contents and cannot manipulate the state. Thus a practical scenario would be an implementation of the protocol using a Tamper-Proof Module or cryptographic coprocessor, which protects at least the long term keys, while other parts of the protocol are executed in unprotected memory. The adversary then is able to gain read-only access to this memory, e. g., by side channel attacks, or by attacks such as freezing the memory. The model does not realistically model an adversary gaining administrator/root access to the machine, as that would require modeling active manipulation of the session-state.

The CK_{HMQV} security notion was developed in tandem with the HMQV protocol, and it seems that the requirements on HMQV have influenced the security model. Relaxing

the condition of Perfect Forward Secrecy to Weak Perfect Forward Secrecy seems driven by the requirement of implicit authentication. Similarly, the change of partnering function seems driven by the requirement of symmetry of the roles. Otherwise, the model is similar to the CK model but additionally considers Key Compromise Impersonation attacks.

5 Designing a protocol correct in all three models

In Figure 2 we present a two-message protocol that is correct in all three models. The protocol design is essentially a straightforward signed Diffie-Hellman exchange, combined with the Ephemeral Key Reveal countering technique used for the Naxos protocol in [3]. To the best of our knowledge, protocol 1 is the only two-message protocol that is correct in the CK, eCK, and CK_{HMQV} models.

We assume each participant A has a long-term private/public signature key pair (sk_A, pk_A) and that the agents can verify the signatures of others (e. g. by pre-distribution or the inclusion of certificates in the message exchange.) We assume H_1 and H_2 are hash functions. The agents A and B perform a basic signed Diffie-Hellman key exchange. However, as in Naxos, instead of using the random values r_A and r_B as their ephemeral private keys the agents use $x_A = H_1(sk_A, r_A)$ and $x_B = H_1(sk_B, r_B)$. Other design decisions are (1) the first and second message are sufficiently distinct, and (2) A's ephemeral public key is contained in the signature of the second message to bind it to B's ephemeral key in the response. At the end of a session, all state except for the session key is erased. (If needed, an explicit session identifier can be defined using a hash of the agent names, long-term public keys and ephemeral public keys.)

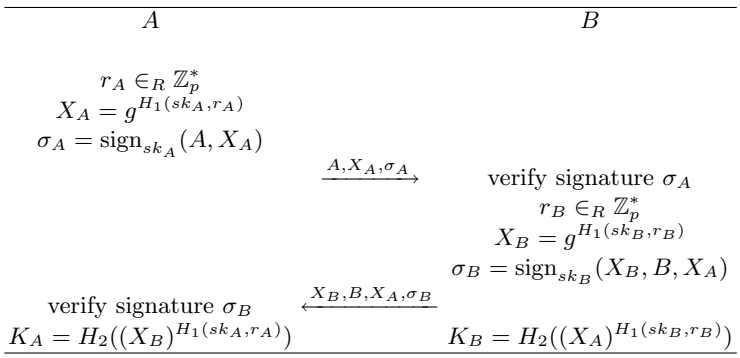

Fig. 2. Protocol 1: A variant of signed Diffie-Hellman using the Naxos transformation

Security argument. A security proof of Protocol 1 will be given in the full version of this paper. Here we observe that the security proof in the CK and CK_{HMQV} models can reuse existing proofs of a basic signed Diffie-Hellman key exchange based on the assumption that

H_1 is a hash function. With respect to the eCK model, as pointed out in [9], leaking r_A using Ephemeral Key Reveal does not allow the adversary to compute the key unless sk_A is also revealed. Revealing both r_A of the test session as well as sk_A (or r_B and sk_B) is disallowed in the eCK model.

6 Related work

Many indistinguishability-based proof models for AKE security have been proposed, e. g., [1–3, 11, 12]. Some earlier models have been compared in [13]. The use of session identifiers has been studied in [14]. Recently, the CK [1] model has been related to one of its variants with respect to specification of peers before or after the session in [15].

Several authors have suggested that the eCK model is the strongest security model, e. g., [4, 7, 16, 17]. In [8] it was shown that this is not the case. In contrast to [8], our work focusses not on a single query difference between two models but on the full security models, and relates them in detail. We observe that the proof of the theorem in [17] that eCK is the strongest security model, only considers the existence of certain query types, and not the conditions under which these queries may occur in the security experiment.

Alternative definitions of AKE security in a simulation-based context have been suggested in [18, 19]. The security models in [19] are weaker than the models described here. The definitions in [18] are intended for protocol construction and not immediately applicable to the analysis of existing protocols.

7 Conclusions and Future Work

We have shown that the CK, eCK, and $\mathrm{CK}_{\mathrm{HMQV}}$ models for AKE security are not only formally but also practically incomparable, thereby refuting several (informal) claims made in the literature in e. g. [4,7,16,17] about the relation between the models. For each model, there exist attacks on protocols that are not considered by the other models. We have analysed several protocols and determined in which AKE models they are not correct. This both shows the implications of the differences as well as provides a new view on some of these protocols. Finally, we presented a new protocol that is (to the best of our knowledge) the first two-message AKE protocol to be secure in all three models.

The complexity of strong AKE security models makes them hard to compare or to relate to practice. This complexity seems to be caused by the aim of making the security notion, and thus the adversary, as strong as possible, such that any stronger adversary would be able to break all protocols. However, because there is no total order on adversaries, there is no single strongest model for which there are still correct protocols. As a result, multiple "strong" models can coexist. However, if the practical implications of a security model are made clear, as in Section 4.7, it becomes possible to choose among the security models based on the target application domain.

As future work, we consider the possibility of a two-message AKE protocol that is more efficient than Protocol 1 and additionally provides identity protection. Finally, it would be of interest to determine the exact relation between the guarantees provided by simulation based AKE security notions [18] and the security models considered here.

References

1. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: EUROCRYPT. Volume 2045 of LNCS., Springer (2001) 453–474
2. Krawczyk, H.: HMQV: A high-performance secure Diffie-Hellman protocol. In: CRYPTO 2005. Volume 3621 of Lecture Notes in Computer Science., Springer-Verlag (2005) 546–566
3. LaMacchia, B., Lauter, K., Mityagin, A.: Stronger security of authenticated key exchange. In: ProvSec. Volume 4784 of Lecture Notes in Computer Science., Springer (2007) 1–16
4. Okamoto, T.: Authenticated key exchange and key encapsulation in the standard model. In: ASIACRYPT. Volume 4833 of Lecture Notes in Computer Science. (2007) 474–484
5. Just, M., Vaudenay, S.: Authenticated multi-party key agreement. In: Advances in Cryptology-ASIACRYPT 1996. Volume 1163 of Lecture Notes in Computer Science. (1996) 36–49
6. Law, L., Menezes, A., Qu, M., Solinas, J., Vanstone, S.: An efficient protocol for authenticated key agreement. Designs, Codes and Cryptography **28** (2003) 119–134
7. Ustaoglu, B.: Obtaining a secure and efficient key agreement protocol from (H)MQV and NAXOS. Des. Codes Cryptography **46**(3) (2008) 329–342
8. Cremers, C.: Session-state Reveal is stronger than Ephemeral Key Reveal: Attacking the NAXOS key exchange protocol. In: ACNS'09. Lecture Notes in Computer Science (2009)
9. Boyd, C., Cliff, Y., Nieto, J., Paterson, K.: Efficient one-round key exchange in the standard model. In: ACISP. Volume 5107 of Lecture Notes in Computer Science., Springer (2008) 69–83
10. Basin, D., Cremers, C.: From Dolev-Yao to strong adaptive corruption: Analyzing security in the presence of compromising adversaries. Cryptology ePrint Archive, Report 2009/079 (2009) http://eprint.iacr.org/.
11. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: CRYPTO '93: Proceedings of the 13th annual international cryptology conference on Advances in cryptology, New York, NY, USA, Springer-Verlag (1994) 232–249
12. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated key exchange secure against dictionary attacks. In: EUROCRYPT. Lecture Notes in Computer Science, Springer (2000) 139–155
13. Choo, K.K., Boyd, C., Hitchcock, Y.: Examining indistinguishability-based proof models for key establishment proofs. In: ASIACRYPT. Volume 3788 of Lecture Notes in Computer Science., Springer (2005) 624–643
14. Choo, K.K., Boyd, C., Hitchcock, Y., Maitland, G.: On session identifiers in provably secure protocols. **3352** (2005) 351–366
15. Menezes, A., Ustaoglu, B.: Comparing the pre- and post-specified peer models for key agreement. In: Proceedings of ACISP 2008. Volume 5107 of Lecture Notes in Computer Science. (2008) 53–68
16. LaMacchia, B., Lauter, K., Mityagin, A.: Stronger security of authenticated key exchange. Cryptology ePrint Archive, Report 2006/073 (2006) http://eprint.iacr.org/.
17. Xia, J., Wang, J., Fang, L., Ren, Y., Bian, S.: Formal proof of relative strengths of security between ECK2007 model and other proof models for key agreement protocols. Cryptology ePrint Archive, Report 2008/479 (2008) http://eprint.iacr.org/, retrieved on April 1st, 2009.
18. Canetti, R., Krawczyk, H.: Universally composable notions of key exchange and secure channels. In: EUROCRYPT. Lecture Notes in Computer Science, Springer-Verlag (2002) 337–351
19. Shoup, V.: On formal models for secure key exchange (version 4) (November 1999) revision of IBM Research Report RZ 3120 (April 1999).

A Attack on the Naxos protocol in the CK model

The Naxos protocol described in [3] does not satisfy Perfect Forward Secrecy and is therefore not correct in the CK model. The idea of the attack is a generalization of the observation in [2]: if the message exchange is not authenticated initially, an adversary can impose as the responder B and send messages of the form g^z for some z that he knows. Then, after the test session (of the initiator A) ends, the adversary reveals the long-term private key of B, and computes the session key. In the case of Naxos, an honest participant in the responder role would send $g^{H_1(esk_B, sk_B)}$ where esk_B is the ephemeral private key of B. However, since A can only check that the result is a group element, no authentication takes place.

The attack proceeds as follows. The initiator A starts the test session, trying to communicate with B. A generates esk_A and computes $X = g^{H_1(esk_A, sk_A)}$. This message is sent to B. The adversary intercepts the message and chooses any z. He computes $Y = g^z$ and sends this to A. A checks that Y is in the group G. A computes the session key as

$$K_A = H_2(Y^{sk_A}, pk_B^{H_1(esk_A, sk_A)}, Y^{H_1(esk_A, sk_A)}, A, B).$$

Now the test session ends. The adversary reveals the long-term key sk_B of B as is allowed in the CK model. Unfolding $Y = g^z$ we get

$$K_A = H_2((g^z)^{sk_A}, pk_B^{H_1(esk_A, sk_A)}, (g^z)^{H_1(esk_A, sk_A)}, A, B).$$

Based on the equivalences $(g^x)^y = (g^y)^x$) and $pk_X = g^{sk_X}$ we find

$$K_A = H_2(pk_A^z, (g^{H_1(esk_A, sk_A)})^{sk_B}, (g^{H_1(esk_A, sk_A)})^z, A, B),$$

which, combined with $X = g^{H_1(esk_A, sk_A)}$ yields

$$K_A = H_2(pk_A^z, X^{sk_B}, X^z, A, B).$$

The adversary knows z, X, pk_A, and (by long-term key reveal) sk_B and is therefore able to compute the session key.

B Attack on the Okamoto protocol in the CK model

The Okamoto protocol from [4] is vulnerable to an attack on Perfect Forward Secrecy that resembles the attack on Naxos described in Appendix A. The initiator A starts the test session with B and computes and sends X_1, X_2, X_3. The adversary intercepts the first message. Instead of computing y, y_3 using the long-term private key of B, the adversary chooses arbitrary values for y, y_3. He computes the ephemeral public keys $Y_1 = g_1^y$, $Y_2 = g_2^y$,

and $Y_3 = g_1^{y_3}$, which are sent back to A. The remainder of the attack continues as in the Naxos example. In particular, A computes

$$\sigma_A = Y_1^{a_1+ca_3} Y_2^{a_2+ca_4} Y_3^{x_3} pk1_B{}^x pk2_B{}^{dx}$$

where $a_1, \ldots, a_4$ are components of A's long-term private key, and $pk1_B = g_1^{b_1} g_2^{b_2}$ and $pk2_B = g_1^{b_3} g_2^{b_4}$. We have that $c = H_A(A, B, Y_1, Y_2, Y_3)$ and $d = H_B(B, A, X_1, X_2, X_3)$, where c and d can be computed by a passive adversary. Together with the session identifier, which is known to the adversary, σ_A is used to compute the session key. Hence the security of the key depends on the adversary not being able to compute σ_A.

After the end of A's session the adversary reveals B's long-term private key $(b_0, b_1, b_2, b_3, b_4)$. Now, by the design of the protocol, we have that

$$\sigma_A = \sigma_B = X_1^{b_1+db_3} X_2^{b_2+db_4} X_3^{y_3} pk1_A{}^y pk2_A{}^{cy}.$$

The adversary has chosen y, y_3. He knows A's public keys $pk1_A, pk2_A$, and knows X_1, X_2, X_3 and $b_1, \ldots, b_4$. He can compute c and d. Hence he is also able to compute σ_B and therefore the session key.