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

July 22, 2009

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 formally incomparable for a number of reasons. Second, we show these models are also practically incomparable, by providing for each model attacks on existing protocols that are not considered by the other models. Our analysis exposes many subtleties of these models, some of which can even be generalized to reveal shortcomings in security proofs in related AKE security models.

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

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

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

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 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 are *formally* incomparable, by showing differences in application domain, adversary model, and correctness prerequisites.

Second, we show the practical differences, by presenting protocols that are correct in one model but cannot be shown to be correct in the others, and vice versa. Therefore, we establish that the three models are not only *formally* but also *practically* incomparable.

Third, our analysis reveals subtleties about the interaction between matching sessions and key derivation that can be applied to these models but also to related models. Using our observations we reveal previously unreported shortcomings in protocol proofs.

We proceed as follows. In Section 2 we recall the ideas underlying indistinguishability-based AKE security models, and describe the eCK, CK and CK_{HMQV} models. In Section 3 we show formal incomparability of the models. In Section 4 we show practical incomparability of the three models using protocols from the literature. Additionally, we discuss possible practical interpretations of each model. In Section 5 we generalize our observations for symmetric-role protocols and apply them to a set of protocols and their proofs. We discuss related work in Section 6 and provide conclusions and future work in Section 7.

Acknowledgements

The author is grateful to Berkant Ustaoglu and Alfred Menezes, whose constructive comments and insightful discussions have lead to a complete rewrite of an earlier version of this paper.

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 security 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 properties of interest are that (1) intended communication partners compute the same key, and that (2) the adversary is not be able to distinguish the established 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.

One aspect which will play an important role in our discussion is the notion of matching sessions (sometimes referred to as partnering) which aims to capture when two sessions are "intended communication partners". This notion is both used in the specification of the security property (matching sessions should compute the same key) as well as in the definition of the adversary capabilities (e. g., the adversary can reveal the session key of non-matching sessions.)

2.2 Preliminaries and notational conventions

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, and thus should compute the same key.

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 enable formulation of adversarial capabilities, such as the capability to reveal e. g. session keys of *other* sessions.

Below we recall (parts of) the definitions of the three security models that can be skipped by readers familiar with the details of the models. In Section 3 we formally compare the models.

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:*

1. *compute the same session keys, and*
2. *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 notion of matching sessions is defined as sessions having the same session identifiers and reversed name assignments, i. e., a session with identifier sid executed by Alice talking to Bob, matches with a session with identifier sid executed by Bob communicating with Alice. The*

session identifiers are assumed to be externally provided to the security model, by e. g. the application layer. The session identifiers are subject to some constraints which we detail in the next section.

2. *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.*
3. *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.*
4. *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.*
5. *If A executes the test session, supposedly with B, 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 only be revealed if a matching session exists for the test session.*

In existing proofs in the CK model such as for the signed Diffie-Hellman protocol, the session state is defined as the ephemeral keys, or the random coins drawn by the participant. 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.

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. *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 $(ID*, ID, comm*, comm)$.*

2. *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.*
3. *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.*
4. *A separate analysis is performed in which the adversary is allowed to learn the ephemeral keys of the test session or the matching session.*

Changing the definition of matching sessions is relevant for protocols that have symmetrical roles such as MQV [5] and HMQV [2]. We will discuss this in detail in Section 3.3.

3 Formally relating the three models

We first discuss the formal differences between the three models with respect to three perspectives: application domain, adversary capabilities, and correctness prerequisites. Afterwards we summarize the formal differences in Table 1.

3.1 Differences in application domain

In general, the AKE security models are applicable to two-party protocols that aim to establish a fresh shared key. However, for the CK model, an additional domain restriction holds.

CK application domain Establishing a protocol correctness proof in the CK model requires the definition of a *session identifier*. Each session is assigned a session identifier. If two participants want to agree on a session key their respective sessions are assigned the same session identifier. How these session identifiers are established or identified in a given protocol is not specified in the CK model. However, the following constraints are given:

- A participant never uses the same session identifier twice [1, p. 4].
- Every message contains the session identifier [1, p. 11].

In contrast, the other two models do not require the existence of a session identifier.

3.2 Differences in adversarial capabilities

The models differ most in their definitions of adversarial capabilities.

CK adversary capabilities The CK model allows for *state-reveal* queries. These allow the adversary to learn the contents of the local state of all sessions except for the test session and its matching session. The state contents act as a parameter of the security model. The only requirement is that the local state does not contain the long-term private keys [1, p. 6].

The compromise of the long-term private key of the *actor* (the participant that executes the test session) before the test session expires, is not allowed in the CK model [1, p. 14]. As a result, the CK model is not able to detect *key compromise impersonation* (KCI) attacks [6].

After the test session expires, the adversary is allowed to corrupt the participant that executes the test session [1, p. 12]. Similarly, after the session that matches the test session expires, the adversary is allowed to corrupt the participant that executes the matching session. Unlike in the CK_{HMQV} and eCK models, this is allowed regardless of whether the adversary actively interferes with the communication between the test session and its partner. This corresponds to checking for Perfect Forward Secrecy.

Attacks on regular protocol sessions (during which the adversary is passive with respect to the test session and its partner) in which Alice talks to Alice are not considered in the CK model. This is a side effect of the definition of the session identifiers: Once Alice starts a session with identifier s and sends a message m (that contains s), other sessions of Alice can not accept this incoming message, as a session can only occur once at a single participant [1, p. 11].

CK_{HMQV} adversary capabilities The CK_{HMQV} model allows for state-reveal queries [2, p. 6] as in the CK model.

In order to detect KCI attacks [6], the CK_{HMQV} model allows the compromise of the long-term private key of the actor (also before the test session ends) [2, p. 41].

The extended analysis model of CK_{HMQV} [2, p. 54] allows for revealing the ephemeral key of test session of its matching session, provided that the long-term private key of the agent that generated the revealed ephemeral key, remains secret.

The corruption of the actor or the peer (the intended partner participant) after the end of the test session is only allowed if a matching session exists [2, p. 42]. Secrecy with respect to this definition is defined as *weak Perfect Forward Secrecy*.

eCK adversary capabilities The eCK model does not include the state-reveal query but instead defines the *ephemeral-key reveal* query. This reveals the ephemeral secrets, i. e., the randomness, of a session [3, p. 6].

The ephemeral-key reveal query also allows for revealing the ephemeral secrets of a session s that computes the same key as the test session (i. e., the test session or its matching session), provided that the long-term private key of the participant executing s is not revealed.

The eCK model allows for the reveal of the long-term private key of the actor before the end of the test session [3, p. 8] and thus can be used to detect KCI attacks.

The corruption of the actor or the peer after the end of the test session is only allowed if a matching session exists [3, p. 9], corresponding to weak Perfect Forward Secrecy.

3.3 Correctness prerequisites in each model

The three models differ in a further aspect, which is neither strictly part of the application domain nor of the adversary model. In each model, certain *generic attacks* exist: some classes of AKE protocols are by definition of incorrect in the model. For each of the models here, different generic attacks exist. We refer to this aspect as the *correctness prerequisites* of a model: a protocol can only possibly be correct in a model, if it does not belong to the class of protocols for which generic attacks exist in the model. If these prerequisites are met, one can attempt a proof in the model (which may then still succeed or fail.)

Before we describe the detailed prerequisites of each model, we introduce notation to express similar prerequisites in a uniform way. The reason behind this is that many of our generic attacks depend on a mismatch between the key derivation function and session matching, which we explain below. We will re-use this notation later in Section 5.

Preliminaries: matching keys versus matching sessions In AKE security models the definitions of matching sessions are strongly connected to the key derivation functions used in protocols. Intuitively, this stems from the following two requirements in AKE models. First, AKE models require matching sessions to compute the same key. Second, the adversary is able to reveal the session key of a non-matching session. Together these conditions imply that for protocols that are correct in a security model, two completed sessions match in the model if and only if they compute the same key.

For example, in the $\mathrm{CK}_{\mathrm{HMQV}}$ model, two completed sessions s and s' are said to match if and only if: the participant executing one session is the peer of the other and its sent messages are the received messages of the other, and vice versa [2, p. 29]. Thus, the key derivation function must meet a similar condition in all valid experiments: exactly when two sessions match then their key derivations must yield identical results.

Below we formalize this insight and use it later to infer conditions that protocols must meet to be correct in each of the models.

Notational conventions. Let s be a completed session in a security experiment. Then s_R denotes the role (initiator, responder) performed by the session, s_A the participant that executes s, and s_B the intended peer of the session. Furthermore, s_{send} denotes the concatenation of the messages sent by s and s_{recv} the concatenation of the messages received. In the context of the CK model we write s_{sid} to denote the session identifier of the session.

We now define four relations between sessions which we will use to categorize the AKE models in the following sections.

Definition 9 (Relation $\approx_A$). *Given two completed sessions s and s', we define*

$$s \approx_A s' \Leftrightarrow \left(s_A = s'_B \wedge s'_A = s_B \wedge s_{send} = s'_{recv} \wedge s'_{send} = s_{recv}\right)$$

In the $\mathrm{CK}_{\mathrm{HMQV}}$ model we have that two completed sessions s and $'s$ are matching if and only if $s \approx_A s'$. For symmetric-role protocols, this definition allows two initiator sessions to be partners.

Definition 10 (Relation $\approx_B$). *Given two completed sessions s and s', we define*

$$s \approx_B s' \Leftrightarrow \left((s_R \neq s'_R \vee s_A = s_B) \wedge s \approx_A s'\right)$$

Relation $\approx_B$ is a variant of $\approx_A$ and does not occur in matching session definitions of the models described here. Instead, the relation occurs in some key derivation functions used in the examples below.

Definition 11 (Relation $\approx_C$). *Given two completed sessions s and s', we define*

$$s \approx_C s' \Leftrightarrow \left(s_R \neq s'_R \wedge s \approx_A s'\right)$$

Relation $\approx_C$ corresponds to the matching sessions definition in the *eCK* model. It explicitly requires the roles to be distinct, thereby excluding two initiators from having matching sessions.

Definition 12 (Relation $\approx_D$). *In the context of the CK model, for two completed sessions s and s', we define*

$$s \approx_D s' \Leftrightarrow \left(s_A = s'_B \wedge s'_A = s_B \wedge s_{sid} = s'_{sid}\right)$$

Relation $\approx_D$ corresponds to the matching sessions definition in the CK model. It depends on the session identifiers, but allows for two initiators of symmetric-role protocols to have matching sessions.

The above definitions now allow us to categorize the AKE models and describe generic attacks. We will use the definitions to make statements about matching sessions in the models, e. g., two completed sessions s and s' match iff $s \approx_A s'$. We will relate this to classes of protocols with particular key derivation functions.

Definition 13 (Key type). *Let P be an AKE protocol and let $\approx$ be a relation on completed sessions. Let $KDF_P(s)$ denote the key computed by the key derivation function of P for any completed session s.*

We say P has key type $\approx$ if and only if for all completed sessions s and s' we have that

$$KDF_P(s) = KDF_P(s') \Leftrightarrow s \approx s'.$$

Here we will not explore the matching behaviour of non-completed sessions, even though this is another important aspect in which the models differ.

CK correctness prerequisites In the CK model, completed sessions s and s' are matching iff $s \approx_D s'$ [1, p. 11].

Theorem 1. *Protocols with session identifiers that do not include the session identifier in a cryptographic operation, and for which the outer (final) operation of the key computation is a cryptographic operation, are incorrect in CK.*[1]

Proof. Let P be a protocol that has the properties described in the theorem. Let s be the test session of a role of P (Alice talking to Bob), and s' a session of the other role (Bob talking to Alice), and assume $s_{sid} \neq s'_{sid}$. The adversary intercepts all messages sent, and replaces s_{sid} by $s_{sid'}$, and vice versa, before forwarding the messages. According to the matching sessions definition of CK (type $\approx_D$) s and s' are not matching. However,

[1] A preliminary version of this theorem and its proof are due to A. Menezes (private communication.)

s and s' compute the same key (as the session identifier is not used in the key computation.) Thus the adversary can reveal the session key of s' thereby learning the (identical) session key of the test session s, and P is thus not correct in the CK model.

Theorem 2. *Protocols with symmetric roles and key type $\approx_B$ or $\approx_C$ are not correct in CK.*

Proof. Let P be a symmetric-role protocol with key type $\approx_B$ or $\approx_C$. Let s be the test session, in particular, an initiator session of Alice communicating with Bob. Let s' be an initiator session of Bob communicating with Alice. Because P has symmetric roles, it is possible that the messages sent by s are received by s' and vice versa. Because the messages contain the session identifier in the CK model, and the names of the participants match, s and s' are matching in the CK model (as can be seen from the definition $\approx_D$.) However, we have that $s_A \neq s_B$ and $s_R = s'_R$ and the key type is $\approx_B$ or $\approx_C$. This implies that s and s' compute different keys. This violates condition 1 of SK-security in the CK model [1, p. 14] and therefore P is not secure in the CK model.

$\mathbf{CK_{HMQV}}$ correctness prerequisites In $\mathrm{CK_{HMQV}}$, completed sessions s and s' are matching iff $s \approx_A s'$ [2, p. 10].

Theorem 3. *Protocols with symmetric roles and key type $\approx_B$ or $\approx_C$ do not satisfy CK_{HMQV} security.*

Proof. The proof is similar to the proof of Theorem 2, except that the reason that s and s' are matching in $\mathrm{CK_{HMQV}}$ is based on the relation $\approx_A$, and the violated condition is (1) in Def. 11 of [2, p. 11].

eCK correctness prerequisites (generic attacks) In eCK, completed sessions s and s' are matching iff $s \approx_C s'$ [3, p. 7].

Theorem 4. *Protocols with symmetric roles and key type $\approx_A$ or $\approx_B$ are not correct in eCK.*

Proof. Let P be a symmetric-role protocol with key type $\approx_A$ or $\approx_B$. First consider the case in which P has key type $\approx_A$. Let s be the test session in the initiator role, executed by Alice communicating with Bob. Let s' be an initiator session of Bob communicating with Alice. Because P has symmetric roles, it is possible that the messages sent by s are received by s' and vice versa. Because the definition of matching sessions in eCK

follows $\approx_C$, which requires roles to be distinct, s and s' are *not* matching in the eCK model. Thus the adversary can do a $\mathsf{Reveal}(s')$ query to reveal the session key of s'. However, because the key type is $\approx_A$, we have that s and s' compute the same keys. Thus, the adversary trivially breaks the security definition [3, p. 9] and therefore P is not secure in the eCK model. For the second case, in which P has key type $\approx_B$, we define s and s' are executed by Alice while communicating with Alice, and proceed analogously.

3.4 Summary of formal differences

In Table 1 we summarize the formal differences between the models. The first column lists abbreviates according to the aspects described before, e. g., "D1" is the first **D**omain restriction, and "A2" is the second **A**dversary capability.

Table 1. Summary of formal differences between the models.

Difference	Description	CK	CK_{HMQV}	eCK
D1	Session identifier	Yes	No	No
A1	Can reveal more of state than eph. keys	Yes	Yes	No
A2	Can reveal eph. keys of session that computes session key	No	Yes[2]	Yes
A3	Can corrupt actor and peer after test session end	Yes	No	No
A4	Can reveal key of actor	No	Yes	Yes
C1	Session identifier in cryptographic operations	Yes	No	No
C2	Incompatible key equivalence types for symmetric-role protocols	$\approx_B$, $\approx_C$	$\approx_B$, $\approx_C$	$\approx_A$, $\approx_B$

4 Practically 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 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.

[2] This is not covered in the basic analysis but only in the extended analysis [2, p. 54].

4.1 $\mathbf{CK_{HMQV}}$ security does not imply CK security

As a counterexample, we point out that the HMQV protocol [2] was proven correct in $\mathrm{CK_{HMQV}}$, but is not secure in CK because it has no session identifiers that occur in all messages and is therefore outside of the application domain of CK.

4.2 CK security does not imply eCK security

A counterexample using difference A2 from Table 1 is described in [3]. The basic signed Diffie-Hellman protocol from [1] provides CK-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 $\mathbf{CK_{HMQV}}$ security

The basic two-message Naxos protocol [3] has key type $\approx_C$ and is therefore not correct in $\mathrm{CK_{HMQV}}$ (conform difference C2), even though it was proven correct in eCK. Another type of attack in $\mathrm{CK_{HMQV}}$ on Naxos, which exploits difference A1, is described in [7]. 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 $\mathbf{CK_{HMQV}}$ security

The counterexample of Section 4.2 also applies for the extended analysis model of $\mathrm{CK_{HMQV}}$.

4.5 $\mathbf{CK_{HMQV}}$ security does not imply eCK security

The HMQV protocol is not correct in eCK because of difference C2: the adversary can trivially reveal the session key in the two-initiators scenario, as in the proof of Theorem 4.

4.6 eCK security does not imply CK security

The counterexample from Section 4.2. applies here as well.

4.7 Possible practical interpretations of the security models

Interpreting the magic session identifiers One of the interpretations suggested in the CK model description is that the application that invokes the protocol instance supplies the session identifier s. In practical applications, CK seems to imply that an information exchange mechanism precedes the actual protocol steps, e. g., by exchanging nonces between the participants and defining s as the concatenation of these nonces. An alternative interpretation (suggested by the examples in CK) is that the initiating participant chooses a fresh s and includes it explicitly in the cryptographic operations of the transmitted messages. On receipt of the first message, the responder checks whether s was used by him as a session identifier before. If so, he aborts. If not, s is stored as the session identifier of the current session. One way to implement this behaviour requires storage of previously observed session identifiers (at least in the order of magnitude of the security parameter.)

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 $\mathrm{CK}_{\mathrm{HMQV}}$ 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 Matching sessions and key equivalence types for symmetric role AKE protocols

In Section 3.3, and in particular in Table 1, we clarified the difference between the models in terms of their matching sessions definitions, and how these relate to the key type, leading to generic attacks.

However, our observations are not limited to the CK, CK_{HMQV} and eCK models only, as the same analysis can also be applied to symmetric-role protocols that were proven correct in other (but related) AKE security models.

In Table 2 we summarize the results of analyzing a number of protocols and their proofs in the models discussed here, as well as in related security models. If the key equivalence type defined by the protocol differs from the matching sessions equivalence type used in the proof, the proof is incorrect. Note that this does not imply that there are practical attacks, rather, side cases may have been missed in the proofs.

Table 2. Key equivalence versus matching sessions in protocols

protocol	key equivalence type of protocol	matching sessions equivalence type in proof
HMQV [2, p. 3]	$\approx_A$	$\approx_A$
MQV [5, p. 131]	$\approx_A$	n.a.
MQV (NIST) [8, p. 46]	$\approx_B$	n.a.
HMQV variant [2, p. 54]	$\approx_B$	n.a.
Okamoto [4]	$\approx_C$	$\approx_A$
CMQV [9]	$\approx_C$	$\approx_A$
Naxos [3]	$\approx_C$	$\approx_C$
BCNP-1	$\approx_C$	$\approx_C$
BCNP-2	$\approx_C$	$\approx_C$

In particular, Table 2 shows that the core MQV and HMQV protocols as described in the papers do allow for two matching initiators to compute the same key. However, in further variants (such as the NIST version of MQV) the agents names are included in a particular order in the key derivation. This changes the key derivation function to type $\approx_B$ which implies that they are strictly speaking incorrect in $\mathrm{CK}_{\mathrm{HMQV}}$ (but also in eCK.) Furthermore, the Okamoto [4] and CMQV [9] protocols use key derivation functions of type $\approx_C$ but their respective proofs use matching session definitions of type $\approx_A$. Consequently partners will not always compute the same keys.

Discussion. In general, it seems that the protocol designer can choose either behaviour $\approx_A$ or $\approx_C$ for a symmetric-role protocol: either the symmetric behaviour is intended and used in practice, leading to $\approx_A$, or the symmetric behaviour is only a theoretical option and should not allow for shared key establishment, leading to $\approx_C$. We do not see immediate reasons to choose $\approx_B$ or $\approx_D$. This choice should not lead to a different security model: rather, one would expect both options to be alternatives of a single security model. The matching sessions definition could then be set to $\approx_A$ or $\approx_C$ in line with the key symmetry type.

In case of MQV and HMQV, it seems the core designs allow for two matching initiators to compute the same key, which would allow them to communicate. However, if the NIST variant of MQV is implemented then this functionality is not provided (for $A \neq B$.) A similar observation holds for the HMQV variant. For Okamoto's protocol and CMQV, whose proofs use type $\approx_A$ matching sessions, a user may falsely assume that a similar matching-initiators functionality is provided. Clearly, protocol designers could be explicit about this choice which would avoid confusion for users as well as avoid mistakes in proofs.

6 Related work

Many indistinguishability-based proof models for AKE security have been proposed, e. g., [1–3, 10, 11]. Some earlier models have been compared in [12]. The use of session identifiers has been studied in [13]. 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 [14].

Several authors have suggested that the eCK model is the strongest security model, e. g., [4, 9, 15, 16]. In [7] it was shown that this is not the case. In contrast to [7], 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 [16] 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 [17, 18]. The security models in [18] are weaker than the models described here. The definitions in [17] 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 CK_{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, 9, 15, 16] about the relation between the models. For each model, there exist protocols that can be shown correct in it, but that cannot be shown correct in the other models. This both shows the implications of the differences as well as provides a new view on some of these protocols.

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. Furthermore we have argued in Section 5 that designers of symmetric-role protocols should explicitly specify whether or not matching initiators compute the same key, both to provide clarity for users as well as avoid mistakes in proofs.

As future work it would be of interest to determine the exact relation between the guarantees provided by simulation based AKE security notions [17] 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. 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
6. 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
7. 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)
8. Barker, E., Johnson, D., Smid, M.: Nist special publication 800-56a: Recommendation for pair-wise key establishment schemes using discrete logarithm cryptography (revised). Technical report (March 2007)
9. Ustaoglu, B.: Obtaining a secure and efficient key agreement protocol from (H)MQV and NAXOS. Des. Codes Cryptography **46**(3) (2008) 329–342
10. 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
11. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated key exchange secure against dictionary attacks. In: EUROCRYPT. Lecture Notes in Computer Science, Springer (2000) 139–155
12. 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
13. Choo, K.K., Boyd, C., Hitchcock, Y., Maitland, G.: On session identifiers in provably secure protocols. **3352** (2005) 351–366
14. 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
15. LaMacchia, B., Lauter, K., Mityagin, A.: Stronger security of authenticated key exchange. Cryptology ePrint Archive, Report 2006/073 (2006) http://eprint.iacr.org/.
16. 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.
17. 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
18. Shoup, V.: On formal models for secure key exchange (version 4) (November 1999) revision of IBM Research Report RZ 3120 (April 1999).