# 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

Version 3.0
July 27, 2010

**Abstract.** Many recent key exchange (KE) protocols have been proven secure in the CK, CK-HMQV, or eCK security models. The exact relation between these security models, and hence between the security guarantees provided by the protocols, is unclear. First, we show that the CK, CK-HMQV, and eCK security models are formally incomparable. Second, we show that these models are also practically incomparable, by providing for each model attacks that are not considered by the other models. Our analysis enables us to find several previously unreported flaws in existing protocol security proofs. We identify the causes of these flaws and show how they can be avoided.

**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

Key Exchange (KE) 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 main goals in recent years has been to design protocols that are secure in the presence of the strongest possible adversaries. For example, many recently proposed KE 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 protocol 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. 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.

There is a limited amount of related work investigating various notions of KE security. Some earlier models for key exchange have been compared in [4]. The use of session identifiers in KE models has been studied in [5]. The CK [1] model has been related to one of its variants with respect to specification of peers before or after the session in [6]. Several authors have suggested that the eCK model is the strongest security model, e. g., [3,7–13]. In [14] it was shown that this

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

is not the case. In contrast to [14], our work focusses not on a single query difference between two models but on the full security models, and relates multiple models in detail.

The fact that the relation between recent strong KE 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 KE security that have been used for the analysis of a large number of protocols. Our observations refute several claims made previously in the literature.

**Contributions.** First, we show that the CK, $\text{CK}_{\text{HMQV}}$, and eCK models are *formally* incomparable, based on their security prerequisites, adversary model, and application domain. Our analysis reveals many previously unreported subtleties in the interaction between the elements of the models.

Second, we show the practical differences, by showing attacks that are detected in one model but are not considered in the others, and vice versa. Therefore, we establish that the three models are not only formally but also *practically* incomparable.

Third, we identify two common sources of errors in protocol security proofs based on these models. We analyze recent protocol security proofs and find several previously unreported flaws. We show how such errors can be avoided when developing security proofs.

We proceed as follows. In Section 2 we recall the ideas underlying indistinguishability-based KE security models, and describe the eCK, CK and $\text{CK}_{\text{HMQV}}$ models. In Section 3 we show formal incomparability of the models. In Section 4 we show practical incomparability of the three models. In Section 5 we identify two common mistakes made in recent KE security proofs. We identify the sources of the problems and show how to avoid them. We draw conclusions and discuss future work in Section 6. Additionally, we discuss possible practical interpretations of each model in Appendix A.

## 2 Three security models for Key Exchange protocols

### 2.1 Elements of Indistinguishability-based security models for key exchange

KE security models define properties of protocols when executed in the presence of an active adversary. We distinguish between three main aspects: the execution model, the security property that should be satisfied, and the adversary model.

The *execution model* defines how protocols are executed by regular participants. The execution model defines aspects 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 KE 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 KE 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, KE security proofs often involve additional parameters. For example, proofs 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 that plays an important role here is the definition of matching sessions (sometimes referred to as partnering), which aims to capture when two sessions are "intended communication partners". Matching sessions are used in KE models in two distinct ways. First, they are used to define a minimal form of protocol correctness: matching sessions are required to compute the same key. Second, they are used to define the adversary capabilities (e. g., the adversary can reveal the session key of non-matching sessions).

Some elements of the KE models we present below seem to be strongly connected to (unspecified) domain-specific knowledge. For example, one unstated assumption of the $\mathrm{CK}_{\mathrm{HMQV}}$ and eCK models seems to be that each role of the protocol creates fresh values and includes these in outgoing messages as well as the key computation. Without such an assumption, matching sessions might not be unique, and one would need to consider replay attacks.

In our descriptions of the security models below, we try to stay close to their original formulations, and give detailed page references where possible. We reformulate the models slightly to provide a more uniform structure among the models to facilitate comparison later on.

### 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, as executed by a participant, 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 KE protocols, we require that both sessions compute the same session key. The KE models considered here each include a notion of *matching sessions* (sometimes called *partnering*) that aims to make precise when two sessions are partners, and thus should compute the same key.

A protocol is said to be *role-symmetric*, or have symmetric roles, when the messages of each role are identical up to their order. Many implicitly authenticated 2-message KE protocols such as MQV are role-symmetric, whereas most variants of signed Diffie-Hellman are *not* role-symmetric. For signed Diffie-Hellman protocols, the structure of the second message is usually similar to the first message, but additionally contains an element of the first message, such as the session identifier or the initiator's ephemeral public key.

The security notions are defined in terms of a *game* or *security experiment* in which a probabilistic polynomial-time (PPT) adversary must have a negligible advantage of winning. In this

game the adversary chooses a so-called *test session* and tries to distinguish the session key of the test session from a random bit string from the key space.

For a session $s$, we write $s_R$ to denote the role (initiator, responder) performed by the session. We write $s_A$ to denote the participant that executes $s$, and $s_B$ to denote 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.

### 2.3 The CK model

Canetti and Krawczyk proposed in [1] a security model for key-exchange protocols, accompanied by a proof methodology for a class of protocols.

In this paper we write "the CK model" or "CK" to refer to Definition 4 from [1]. In the original paper the CK model is called the "SK-security" model in the "unauthenticated links model (UM)" [1, p. 14]. A protocol that is secure in the CK model is said to be "SK-secure in the UM" (often abbreviated to "SK-secure").

**Remark.** In this paper we focus only on the CK model (SK-security in the UM), but for clarity we briefly mention the role of the second model defined in [1]: the SK-security model in the *authenticated* links model (AM) [1, p. 14], which we denote here by $\mathrm{CK}_{\mathrm{AM}}$. The $\mathrm{CK}_{\mathrm{AM}}$ model is used in an intermediate step of the proof methodology that is proposed in [1]. The purpose of the methodology is to construct proofs of protocol security in the CK model. It applies to a class of protocols that use authentication mechanisms to prevent the adversary from modifying network traffic. In particular, the methodology applies to protocols whose authentication mechanisms correspond to the definition of MT-authenticators [15]. Let $P$ be such a protocol. The methodology is to prove that $P$ is secure in CK by the following steps. First, Consider a simpler protocol $P'$, which is obtained by "peeling off" the MT-authenticators from $P$. Second, prove that this protocol is $\mathrm{CK}_{\mathrm{AM}}$-secure, i. e., secure in model that is similar to CK but has a passive network adversary. Third, apply MT-authenticators to $P'$ to obtain protocol $P$. This way of constructing $P$ from the $\mathrm{CK}_{\mathrm{AM}}$-secure protocol $P'$ guarantees its security in CK by a generic theorem [1, p. 16].

Analysis of a protocol in the CK model requires that the protocol includes session identifiers. There are two requirements on session identifiers [1, p. 4]. First, two different sessions at the same party $A$ are required to have different session identifiers [1, p. 11]. Second, if two parties wish to exchange a key, the calling protocol must make sure they activate matching sessions.[1]

The session identifiers are used to define when two sessions are *matching* in CK. As we will see below, the notion of matching sessions plays a crucial role in the security definition. Two sessions $s$ and $s'$ are said to be *CK-matching* if and only if $s_A = s'_B \wedge s'_A = s_B \wedge s_{\mathrm{sid}} = s'_{\mathrm{sid}}$ [1, p. 11]. Note that CK-matching sessions may be performing the same role[2]

[1] It is unclear how the calling protocol can ensure the second requirement by communicating over an insecure network in the presence of an active adversary without an additional security mechanism. Additionally, the second requirement does not seem to play a role in the technical description of CK or the proofs, so it is possibly superfluous.

[2] The informal description of the models [1, p. 4] may seem to suggest that the roles of CK-matching sessions are different, but the technical description [1, p. 11] clearly mentions that this is not required.

In the CK security experiment, the participants start off by initializing their secret/private keys and disclosing any public information (such as the public keys) to the adversary. Next, the adversary is allowed to a sequence of queries from the following set [1, p. 9].

- activate session $s$, which can take two forms:
  - action request $q$. This models communication internal to $s_A$ between the KE protocol and other processes. For KE protocols, a crucial action request is establish-session$(A, B, sid, r)$. If the session identifier $sid$ has not been activated before by party $A$, start a new session $s$ and set $s_A := A, s_B := B, s_{\text{sid}} := sid, s_{\text{role}} := r$.
  - incoming message $m$ with sender $s_b$, modeling messages coming from the network. The protocol description determines how incoming messages are dealt with, and which (if any) response message is returned. For each session $s$, depending on the message $m$, the session may either be aborted execution or completed [1, p. 11]. If the session is aborted, the session-state of $s$ is erased. If the session is completed, the party computes the session key $k$ and erases the session-state except for $k$.
  - session expiration can be scheduled for completed sessions $s$.[3] Expiration erases the session state, i. e., erases the session key $k$.
- session-state reveal session $s$. This reveals the internal state of $s$.
  The CK model does not specify the contents of the internal state of a session, but requires KE protocols to specify the internal state explicitly. It is only required that the session-state does not contain the long-term secrets of the party.
- corrupt party $A$. This reveals all secrets of $A$ (e. g. private keys) as well as the internal states of all of $A$'s unexpired sessions.
- session-key reveal[4] a completed session $s$, revealing $s$'s session key.
- test-session query a completed but unexpired session $s$. A coin $b$ is tossed, i. e., $b$ is randomly drawn from $\{0, 1\}$. If $b = 0$ the query returns $s$'s session key, otherwise the return value is randomly chosen from the probability distribution of keys.[5]

In Figure 1 we illustrate the lifetime of a session and the timing of reveal queries that apply to sessions.

The security experiment considers a subset of all possible sequences of queries. To define the subset of considered sequences, two additional predicates on sessions in the context of an experiment are introduced. A session $s$ is said to be *locally exposed* [1, p. 11/12] if and only if the adversary performed one of the following queries:

- a session-state reveal query on $s$
- a session-key query on $s$
- corrupted $s_A$ before $s$ expired (including when $s_A$ was corrupted before $s$ is invoked or completed).

The session $s$ is said to be *exposed* if it is locally exposed or it has a CK-matching session that is locally exposed. [1, p. 12].

A sequence of queries in an experiment must meet the satisfy the following constraints [1, p. 14].

[3] This action is introduced at [1, p. 11].
[4] This query is called session-output query at [1, p. 9], but later renamed in the context of KE-protocols [1, p. 11].
[5] This action is introduced at [1, p. 13/14].

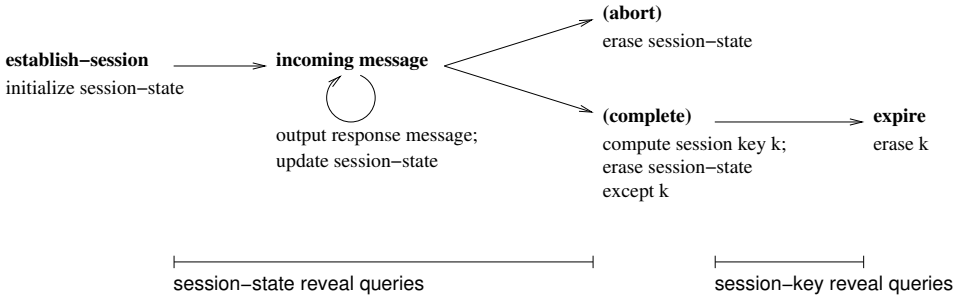


**Fig. 1.** Lifetime of a session in CK. Although session-state reveal queries are allowed after the session is completed, the returned state will be empty [1, p. 11], and we omit these from the graph.

- The test-session query can only be performed once.
- The test-session is not exposed during the experiment.

**Definition 1 (Secure in the CK model).** *[1, p. 14] A protocol $P$ is said to be secure in the CK model, if and only if for all PPT adversaries $\mathcal{M}$ as defined above, we have that*

1. *when two uncorrupted parties complete CK-matching sessions, they output the same key, and*
2. *the probability that $\mathcal{M}$ guesses the bit $b$ (i. e., outputs $b' = b$) from the test-session query correctly is no more than $1/2$ plus a negligible fraction in the security parameter.*

### 2.4 The $\mathrm{CK}_{\mathrm{HMQV}}$ model

Krawczyk uses in [2] variants of the CK model to prove the security of the HMQV protocol. The security of HMQV is proven in two phases. First, HMQV is proven secure in a *basic security model* [2, p. 28], which we call $\mathrm{CK}^{\mathrm{basic}}_{\mathrm{HMQV}}$ here, that removes a number of adversarial capabilities from the CK model. Second, HMQV is shown to be secure in three stronger versions of the $\mathrm{CK}^{\mathrm{basic}}_{\mathrm{HMQV}}$ model [2, p. 40].

The $\mathrm{CK}^{\mathrm{basic}}_{\mathrm{HMQV}}$ model is identical to the CK model except for the following modifications.

- There is no session identifier available after session activation. Rather, matching is defined in terms of sent and received messages: Two *completed* sessions $s$ and $s'$ are said to be *$CK_{HMQV}$-matching* if and only if $s_A = s'_B \wedge s'_A = s_B \wedge s_{send} = s'_{recv} \wedge s'_{send} = s_{recv}$ [2, p. 28]. For incomplete sessions, no definition for matching is provided in [2].
- All occurrences of CK-matching are replaced by $\mathrm{CK}_{\mathrm{HMQV}}$-matching.
- The adversary is not allowed to expire sessions. Because the test session must not become exposed, this effectively prevents the adversary from corrupting the parties that execute the test session and its matching session, and corresponds to not checking for Perfect Forward Secrecy [1, p. 17], [2, p. 29].

– The session-state contents are defined to consist of information known to the adversary (names, sent and received messages) and the session key. The session-state does not include the ephemeral keys used in a Diffie-Hellman style exchange [2, p. 29][6].

In the extended analysis of HMQV, three additional security models are considered, described below.

A session $s$ is said to be *$CK_{HMQV}$-clean* [2, p. 41] in an experiment if no session-state reveal query was performed on $s$ and no session-key reveal query was performed on $s$.

– $\mathrm{CK}^{\mathrm{KCI}}_{\mathrm{HMQV}}$ [2, p. 41]: In addition to the $\mathrm{CK}^{\mathrm{basic}}_{\mathrm{HMQV}}$ queries, the adversary is allowed to reveal the long-term key of the actor $s_A$ of the test session $s$, if (i) $s$ is $\mathrm{CK}_{\mathrm{HMQV}}$-clean, and (ii) $s_B$ is not corrupted.
– $\mathrm{CK}^{\mathrm{wPFS}}_{\mathrm{HMQV}}$ [2, p. 42]: In addition to the $\mathrm{CK}^{\mathrm{basic}}_{\mathrm{HMQV}}$ queries, the adversary is allowed to reveal the long-term keys of the actor $s_A$ and peer $s_B$ of the test session $s$, if (i) $s$ is $\mathrm{CK}_{\mathrm{HMQV}}$-clean, and (ii) a $\mathrm{CK}_{\mathrm{HMQV}}$-matching session of $s$ exists[7], and (iii) all $\mathrm{CK}_{\mathrm{HMQV}}$-matching sessions of $s$ are $\mathrm{CK}_{\mathrm{HMQV}}$-clean.
– $\mathrm{CK}^{\mathrm{eph}}_{\mathrm{HMQV}}$ [2, p. 44/45]: In addition to the $\mathrm{CK}^{\mathrm{basic}}_{\mathrm{HMQV}}$ queries, the adversary is allowed to reveal the ephemeral secret keys of all sessions.

The basic security proof of HMQV in the $\mathrm{CK}^{\mathrm{basic}}_{\mathrm{HMQV}}$ model depends on the computational Diffie-Hellman (CDH) assumption. The main reason to consider multiple models is that the security of HMQV in the $\mathrm{CK}^{\mathrm{eph}}_{\mathrm{HMQV}}$ model depends on stronger assumptions (Gap Diffie-Hellman and KEA1) [2, p. 45].

We say that a protocol is secure in the $\mathrm{CK}_{\mathrm{HMQV}}$ model if and only if it is secure in the four variants, i. e., secure in $\mathrm{CK}^{\mathrm{basic}}_{\mathrm{HMQV}}$, $\mathrm{CK}^{\mathrm{KCI}}_{\mathrm{HMQV}}$, $\mathrm{CK}^{\mathrm{wPFS}}_{\mathrm{HMQV}}$, and $\mathrm{CK}^{\mathrm{eph}}_{\mathrm{HMQV}}$.

**Remark.** The notion of weak perfect forward secrecy (wPFS) is introduced in [2] because the basic version HMQV belongs to a class of protocols that cannot satisfy PFS. A generic attack is sketched by Krawczyk [2, p. 15] but no proof is given. The class of protocols seems to cover all implicitly-authenticated two-message KE protocols. Note that this class excludes protocols such as a signed Diffie-Hellman protocol, which can provide PFS.

**Remark.** wPFS is defined by requiring that the adversary is passive during the interaction between the test session and its matching session. This prevents Krawczyk's attack [2, p. 15]. In the context of KE protocols, in which each role generates fresh values and includes them in their outgoing messages, this requirement can be enforced by requiring that a session exists that $\mathrm{CK}_{\mathrm{HMQV}}$-matches[8] the test session. The requirement is slightly too strong: $\mathrm{CK}_{\mathrm{HMQV}}$-matching requires that the matching session of the initiator role has received the final message. This is not needed to prevent the attack.

[6] The $\mathrm{CK}_{\mathrm{HMQV}}$ definition of session-state seems to render the session-state reveal query useless, because they can only be performed on uncompleted sessions, i. e., before computation of the session keys, and thus before the session keys are part of the session state. The remaining session-state contents are publicly known, and therefore do not provide the adversary with additional powers.

[7] The existence of the $\mathrm{CK}_{\mathrm{HMQV}}$-matching session of $s$ models the adversary being passive during the test session, and prevents the adversary from learning or modifying the ephemeral secrets used in the test session.

[8] Alternatively, it can be required that they eCK-match as in Section 2.5.

### 2.5 The eCK security model

The eCK security model ("extended-CK") was defined by LaMacchia, Lauter, and Mityagin in [12] and [3].

We say that two sessions $s$ and $s'$ are *eCK-matching* if and only if $s_A = s'_B \wedge s'_A = s_B \wedge s_{send} = s'_{recv} \wedge s'_{send} = s_{recv} \wedge s_{role} \neq s'_{role}$ [3, p. 7/8].[9] Observe that eCK-matching is equal to ($\text{CK}_{\text{HMQV}}$-matching $\wedge\ s_{role} \neq s'_{role}$).

In the eCK model, the adversary can perform the following queries [3, p. 8].

- $\mathsf{Send}(A, B, m)$. Sends a message $m$ to $A$ on behalf of $B$ and returns the response. Additionally, this query allows the adversary to establish a new session. After receiving the sequence of messages as specified by the protocol, sessions compute a session key and are considered to be *completed*.
- $\mathsf{Long\text{-}Term\ Key\ Reveal}(A)$. Reveals a long-term key of $A$.
- $\mathsf{Ephemeral\ Key\ Reveal}(s)$. Reveals an ephemeral key of a session $s$.[10]
- $\mathsf{Reveal}(s)$. Reveals a session key of a completed session $s$.
- $\mathsf{Test}(s)$ can be performed on a completed session $s$. A coin $b$ is tossed, i. e., $b \xleftarrow{\$} \{0, 1\}$. If $b = 0$, returns a session key of $s$. If $b = 1$, a random bit string from the key space is returned.
- $\mathsf{Guess}(b')$. If $b'$ is equal to $b$ from the test query, return 1, otherwise return 0.

As in the previous models, only a subset of all possible query sequences is considered in eCK.

A session $s$ is said to be *not eCK-clean* (analogous to the concept of "exposed" in CK) if any of the following conditions hold: [3, p. 8/9]

- $s_A$ or $s_B$ is adversary-controlled, i. e., the adversary chooses or reveals both the long-term and ephemeral keys of the participant and performs on its behalf.[11]
- The experiment includes $\mathsf{Reveal}(s)$.
- A session $s'$ exists that is eCK-matching with $s$, and the experiment includes $\mathsf{Reveal}(s')$.
- The experiment includes both $\mathsf{Long\text{-}term\ Key\ Reveal}(s_A)$ and $\mathsf{Ephemeral\ Key\ Reveal}(s)$.
- A session $s'$ exists that is eCK-matching with $s$, and the experiment includes both $\mathsf{Long\text{-}term\ Key\ Reveal}(s_B)$ and $\mathsf{Ephemeral\ Key\ Reveal}(s')$.
- No session exists that is eCK-matching with $s$, and the experiment includes $\mathsf{Long\text{-}term\ Key\ Reveal}(s_B)$.

An eCK experiment must satisfy the following constraints [3, p. 8/9]:

- The $\mathsf{Test}$ query can only be performed once.
- The test session is eCK-clean.
- The $\mathsf{Guess}$ query is performed exactly once, as the last query of the experiment.

[9] In the original formulation the sequence of exchanged messages is required to be equal. Because the roles are distinct, and we only consider executable two-party protocols, this is equivalent to our reformulation.

[10] In the original description [3] sessions are identified by a unique session identifier $(role, actor, peer, m_1, \ldots, m_n)$. Without loss of generality we can replace the session identifiers in our reformulation of eCK by the abstract session $s$, which helps us to avoid confusion with the session identifiers from the CK model (which are shared among matching sessions).

[11] This condition is needed to ensure that the adversary does not fake the first message as in the generic PFS attack on implicitly authenticated two-message protocols from [2], modeling the aspect of weak PFS.

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

**Definition 2 (eCK security).** *[3, p. 9] A protocol $P$ is said to be secure in the eCK model, if and only if for all PPT adversaries $\mathcal{M}$ as defined above, we have that*

1. *when two uncorrupted parties complete eCK-matching sessions, they compute the same key, and*
2. *no efficient adversary $\mathcal{M}$ has more than a negligible advantage in winning the experiment, where the advantage of the adversary is defined as $\mathbf{Adv}_P^{KE}(\mathcal{M}) = Pr[\mathcal{M}\ wins] - \frac{1}{2}$.*

## 3 Formally relating the three security models

We describe the main differences between the three models with respect to (i) the security prerequisites and (ii) the adversary capabilities. We summarize the differences in Table 1.

### 3.1 Security prerequisites

The three models differ in one main aspect, which depends both on the application domain and the adversary model. In each model, certain *generic attacks* exist: some classes of KE protocols are by definition of insecure in the model. We refer to this aspect as the *security prerequisites* of a model. If these prerequisites are met, one can attempt a proof in the model (which may then still succeed or fail).

The models considered here include two main requirements.

- Matching sessions must compute the same key.
- The session key of the test session must remain secret, even though the adversary is able to reveal the session key of a non-matching session.

The second condition implies that for protocols that are secure in a security model, two completed sessions that compute the same key must (with overwhelming probability) be matching sessions. Hence, the definition of matching sessions is strongly connected to the key derivation functions used in protocols.

We define four relations between sessions, which we will use to classify protocols and models.

**Definition 3 (Relations $\approx_A$,$\approx_B$,$\approx_C$,$\approx_D$).** *Given two completed sessions $s$ and $s'$, we define*

$$s \approx_A s' \stackrel{def}{=} \left(s_A = s'_B \wedge s'_A = s_B \wedge s_{send} = s'_{recv} \wedge s'_{send} = s_{recv}\right) \tag{1}$$

$$s \approx_B s' \stackrel{def}{=} \left(s \approx_A s' \wedge (s_R \neq s'_R \vee s_A = s_B)\right) \tag{2}$$

$$s \approx_C s' \stackrel{def}{=} \left(s \approx_A s' \wedge s_R \neq s'_R\right) \tag{3}$$

$$s \approx_D s' \stackrel{def}{=} \left(s_A = s'_B \wedge s'_A = s_B \wedge s_{sid} = s'_{sid}\right) \tag{4}$$

In the $\mathrm{CK}_{\mathrm{HMQV}}$ model two completed sessions $s$ and $s'$ are matching iff $s \approx_A s'$. For role-symmetric protocols, this definition allows two initiator sessions to be partners, because $\approx_A$ ignores the order in which messages were sent. Hence, the messages of two initiators can cross and they may have

matching sessions even though they are both in the same role. Relation $\approx_B$ is a variant of $\approx_A$ and does not occur in matching session definitions of the models described here. Instead, as we will see later, this relation occurs for some key derivation functions. 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. Relation $\approx_D$ corresponds to the matching sessions definition in the CK model. It allows two initiators of role-symmetric protocols to have matching sessions.

The above definitions allow us to categorize the KE models and describe generic attacks, by characterizing the notion of matching sessions in each of the models, e. g., two completed sessions $s$ and $s'$ match iff $s \approx_A s'$. We relate the characteristics of matching sessions to classes of protocols with particular key derivation functions.

**Definition 4 (Key type).** *Let $P$ be an KE 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 that $P$ has key type $\approx$ iff for all completed sessions $s$ and $s'$, it holds that $KDF_P(s) = KDF_P(s') \Leftrightarrow s \approx s'$ .*

In this section, we do not compare of definitions of matching for non-completed sessions, even though this is another important aspect in which they differ, which influences the adversary's ability to perform state-reveal and ephemeral-key reveal queries. We will return to incomplete sessions in Section 5.2.

**CK security prerequisites.** In CK, completed sessions $s$ and $s'$ are matching iff $s \approx_D s'$ [1, p. 11].[12] Protocol messages are required to include the session identifier [1, p. 11], which is provided by the calling application.

**Theorem 1.** *Role-symmetric protocols with key type $\approx_B$ or $\approx_C$ are insecure in CK.*

*Proof.* Let $P$ be a role-symmetric protocol with key type $\approx_B$ or $\approx_C$, i. e., $P$ has a role-symmetric execution. The adversary establishes sessions $s$ and $s'$, performed respectively by Alice and Bob, that together form a role symmetric execution: both Alice and Bob perform the initiator role and their messages cross. Because the messages contain the session identifier in the CK model, and each session accepts the messages of the other session, the session identifiers of both sessions must be equal. Furthermore, the names of the participants of each session correspond to the names of the other session in reverse order, i. e., (Alice,Bob) and (Bob,Alice). Hence $s$ and $s'$ are matching in the CK model (as can be seen from the definition $\approx_D$). Observe that $s_A \neq s_B$ and $s_R = s'_R$ and the key type is $\approx_B$ or $\approx_C$. This implies that the matching sessions $s$ and $s'$ compute different keys, which violates condition 1 of SK-security in the CK model [1, p. 14]. Therefore $P$ is not secure in the CK model.

**$\mathrm{CK}_{\mathrm{HMQV}}$ security prerequisites.** In $\mathrm{CK}_{\mathrm{HMQV}}$, completed sessions $s$ and $s'$ are matching iff $s \approx_A s'$ [2, p. 10].

[12] Note that although the informal introduction of the CK model [1, p. 4] suggests that roles must be distinct, the technical description of the model clearly states that roles are not required to be distinct [1, p. 11].

**Theorem 2.** *Role-symmetric protocols with key type $\approx_B$ or $\approx_C$ do not satisfy $CK_{HMQV}$ security.*

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

**eCK security prerequisites.** In eCK, completed sessions $s$ and $s'$ are matching iff $s \approx_C s'$ [3, p. 7].

**Theorem 3.** *Role-symmetric protocols with key type $\approx_A$ or $\approx_B$ are insecure in eCK.*

*Proof.* Let $P$ be a role-symmetric 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.

### 3.2 Adversary 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 [16]. 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 $\text{CK}_{\text{HMQV}}$ 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 (PFS). 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 identifier can only occur once at a single participant [1, p. 11].

| | CK | $\text{CK}_{\text{HMQV}}$ | eCK |
|---|---|---|---|
| domain restrictions | protocol messages contain session identifier | | |
| behaviour restrictions | | | adversary passive during communication between test session and its eCK matching session |
| sessions that should compute the same key | CK-matching | $\text{CK}_{\text{HMQV}}$-matching | eCK-matching |
| incompatible key equivalence types for role-symmetric protocols | $\approx_B$, $\approx_C$ | $\approx_B$, $\approx_C$ | $\approx_A$, $\approx_B$ |
| reveal long-term key of $\text{test}_A$ | if test has expired, $\text{test}_A$ can be corrupted | [wPFS]: if session matching test exists, and both $\text{CK}_{\text{HMQV}}$-clean; [KCI]: if test session $\text{CK}_{\text{HMQV}}$-clean, and $\text{test}_B$ not corrupted; [basic,eph]: never | ephemeral keys of test not revealed |
| reveal long-term key of $\text{test}_B$ | if the session that CK-matches test has expired, $\text{test}_B$ can be corrupted | [wPFS]: if session matching test exists, and both $\text{CK}_{\text{HMQV}}$-clean; [basic,KCI,eph]: never | if no reveal of ephemeral keys of sessions that eCK-match test |
| reveal ephemeral keys of $s$ | if ephemeral keys are in session state and $s \neq$ test and $s$ is not CK-matching test | [basic,KCI,wPFS]: never [eph]: anytime | if $s =$ test, the long-term key of $s_A$ must not be revealed; if $s$ eCK-matches test, the long-term key of $s_B$ must not be revealed; otherwise allowed |
| reveal session keys of $s \neq$ test | if $s$ is not CK-matching test | if $s$ is not $\text{CK}_{\text{HMQV}}$-matching test | if $s$ is not eCK-matching test |
| reveal other session state of $s$ (e. g. intermediate computations) | if $s \neq$ test and $s$ is not CK-matching test | if $s \neq$ test and $s$ is not $\text{CK}_{\text{HMQV}}$-matching test | never |

**Table 1.** Summary of formal differences between the models. For the $\text{CK}_{\text{HMQV}}$ model we identify the relevant submodels in square brackets.

**$\text{CK}_{\text{HMQV}}$ adversary capabilities.** The $\text{CK}_{\text{HMQV}}$ model allows for state-reveal queries [2, p. 6] as in the CK model. In order to detect KCI attacks [16], the $\text{CK}^{\text{KCI}}_{\text{HMQV}}$ model allows the compromise of the long-term private key of the actor (also before the test session ends) [2, p. 41]. The $\text{CK}^{\text{eph}}_{\text{HMQV}}$ model [2, p. 54] allows for revealing the ephemeral key of the test session and 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 in the $\text{CK}^{\text{wPFS}}_{\text{HMQV}}$ model is only allowed if a matching session exists [2, p. 42]. Secrecy with respect to this definition is known as *weak Perfect Forward Secrecy* (wPFS).

**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 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 peer after the end of the test session is only allowed if a matching session exists [3, p. 9], corresponding to weak-PFS.

## 4 Practically relating the CK, $\mathrm{CK}_{\mathrm{HMQV}}$, and eCK security models

In this section we show practical incomparability of the models, i. e., we show that in each model, attacks on protocols from literature exist that are not considered by the other models.

**(1) $\mathrm{CK}_{\mathrm{HMQV}}$ security does not imply CK security.** As a counterexample, we point out that the HMQV protocol [2] was proven secure in $\mathrm{CK}_{\mathrm{HMQV}}$, does not provide perfect forward secrecy and is therefore not secure in the CK model. For example, the generic attack sketched in [2] applies to HMQV in the CK model.

**(2) CK security does not imply eCK security.** A counterexample using the reveal of ephemeral keys of the test session 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.

**(3) eCK security does not imply $\mathrm{CK}_{\mathrm{HMQV}}$ security.** The eCK model does not consider attacks that involve intermediate computations of the protocol, whereas the $\mathrm{CK}_{\mathrm{HMQV}}$ model allows the session-state reveal query for this purpose. The basic two-message Naxos protocol [3] has key type $\approx_C$ and is therefore insecure in $\mathrm{CK}_{\mathrm{HMQV}}$, even though it was proven secure in eCK[13]. Another type of attack in $\mathrm{CK}_{\mathrm{HMQV}}$ on Naxos, which exploits revealing intermediate computations of Naxos, is described in [14]. 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) CK security does not imply $\mathrm{CK}_{\mathrm{HMQV}}$ security.** The counterexample from (2) also applies for the $\mathrm{CK}^{\mathrm{eph}}_{\mathrm{HMQV}}$ model.

**(5) $\mathrm{CK}_{\mathrm{HMQV}}$ security does not imply eCK security.** The HMQV protocol is insecure in eCK because the adversary can trivially reveal the session key in the two-initiators scenario, as in the proof of Theorem 3. The insecurity is based on a mismatch between the equivalence type of matching sessions and the equivalence type of derived keys.

**(6) eCK security does not imply CK security.** The counterexample from (3) applies here as well.

[13] This statement assumes a fixed version of the eCK model: as we will show in Section 5.2, Naxos is strictly speaking not secure in eCK because of a bug in the definition of matching for incomplete sessions.

## 5 Two common flaws in KE security proofs

Although security models based on the CK model have been used for almost a decade, many recent proofs still contain very basic flaws. These flaws do not directly imply practical attacks, but show that some elements of the KE models are still not well understood. We identify two common sources of errors: the first depends on the interaction between key equivalence types and matching sessions in role-symmetric protocols, and the second relates to queries that can be performed on incomplete sessions and their interaction with the definition of matching sessions.

### 5.1 Matching sessions and key equivalence types for role-symmetric protocols

The observations from Section 3.1 and Table 1 can be applied to role-symmetric protocols that were proven secure in KE security models. In Table 2 we summarize the results of our analysis. If the key equivalence type defined by the protocol differs from the matching sessions equivalence type used in the proof, the protocol is technically insecure in the model. The cause of the problem is either that side cases were missed in the proof, or an inappropriate definition of matching was used.

| protocol | key equivalence type of protocol | matching sessions equivalence type in proof | comments |
|---|---|---|---|
| HMQV [2, p. 3] | $\approx_A$ | $\approx_A$ | |
| MQV [17, p. 131] | $\approx_A$ | n.a. | |
| MQV (NIST) [18, p. 46] | $\approx_B$ | n.a. | Matching initiators do not compute the same key (unlike MQV) |
| HMQV variant [2, p. 54] | $\approx_B$ | n.a. | Matching initiators do not compute the same key (unlike basic HMQV), therefore insecure in $\mathrm{CK}_{\mathrm{HMQV}}$ |
| Okamoto [10] | $\approx_C$ | $\approx_A$ | Flaw in proof: Matching sessions do not always compute the same key |
| CMQV [11] | $\approx_C$ | $\approx_A$ | Flaw in proof: Matching sessions do not always compute the same key |
| HuangCao [19] | $\approx_C$ | $\approx_A$ | Matching sessions do not always compute the same key (but requirement omitted from the model) |

**Table 2.** Key equivalence versus matching sessions in protocols

The basic MQV and HMQV protocols allow for two matching initiators to compute the same key, which allows them to communicate. In variants of these protocols, such as the NIST version of MQV, the agents' names are included in a particular order in the key derivation function. This changes the key derivation function to type $\approx_B$, which implies that these variants are insecure in $\mathrm{CK}_{\mathrm{HMQV}}$ and eCK. Furthermore, in practice, these variants offer less functionality than the originals: two matching initiators cannot communicate.

The Okamoto [10] and CMQV [11] protocols use key derivation functions of type $\approx_C$ but their respective proofs use matching session definitions of type $\approx_A$. As a result, partners may compute

different keys, violating the security definition. A user of these protocols might falsely assume from the security model that a matching-initiators functionality is provided.

We assume that a protocol designer can choose either behaviour $\approx_A$ or $\approx_C$ for a role-symmetric 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 for choosing $\approx_B$ 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. Protocol designers could state these choices explicitly to avoid confusion for users and avoid mistakes in proofs.

### 5.2 The interaction between matching and queries for incomplete sessions

In the KE security models, there is no requirement that all sessions in an experiment are completed. In fact, this might not be possible, e. g., because the adversary injects fake messages such that the final message required to complete a session cannot be constructed.

Some adversary queries, such as session-state reveal or Ephemeral-Key reveal can be performed on incomplete sessions. These queries interact with the definition of *matching* for incomplete sessions. For example, session-state reveal in CK can only be performed on sessions that are not the test session and do not CK-match the test session; Ephemeral-Key reveal in eCK can only be performed on a session that eCK-matches the test session if the peer's long-term key is not revealed. Thus, the definition of matching for incomplete sessions influences whether session-state reveal or Ephemeral-Key reveal queries are possible. Similarly, for statements like "a matching session exists" the status of incomplete sessions is also relevant.

In the CK model, matching is defined in terms of the session identifier and the identities, which are all fixed once a session is established, and the session identifier does not change during session execution. This is different in the $\text{CK}_{\text{HMQV}}$ and eCK models.

In $\text{CK}_{\text{HMQV}}$, the notion of matching is only defined for completed sessions [2, p. 29]. As a result, it is undefined whether session-state reveal queries, which are only allowed on incomplete sessions, are allowed in $\text{CK}_{\text{HMQV}}$. However, as we observed earlier, the session-state query is redundant in the basic security proof of HMQV, and therefore this underspecification has no consequences for HMQV.

In eCK, matching is defined in terms of communicated messages; this implies that an incomplete session can never match the (completed) test In follow-up works to the security models presented here, some authors have closely followed the original eCK formulation, e. g. [10, 20]. Other authors have silently adapted the definition of matching sessions, e. g., [7, 9, 11, 19, 21], to ensure that incomplete sessions that could be completed to a matching session (by adding additional queries), are considered to be matching sessions as well.

### 5.3 Avoiding matching problems

The above observations suggest a straightforward procedure to construct appropriate matching definitions and help to define session key derivation functions.

1. Determine the intended behaviours for the protocol in terms of completed sessions: given a completed session $s$, which other completed sessions are intended to compute the same

key? In the case of two-party protocols this boils down to deciding whether role-symmetric functionality is desired. The matching definition for completed sessions should be specified accordingly.
2. Ensure that the key derivation function computes the same key for two completed sessions if and only if the sessions are matching.
3. Define matching for incomplete sessions in the following way. For all complete sessions $s_1$, $s_2$ that are matching, we define that all prefixes $s_2'$ of $s_2$ also match $s_1$, i. e., all incomplete sessions $s_2'$ that can be extended to $s_2$ are also considered matching.

The first two steps ensure that the intended functionality is reflected in the security definition and key derivation function. The third step effectively gives the adversary access to all sessions not involved in the intended behaviour.

In [22], it is proposed to define matching (called partnering in [22]) based on the key derivation, thereby ensuring that mismatches between the partnering of complete sessions and key derivation cannot occur. However, they do not provide a definition of partnering for incomplete sessions. In the models considered here, such a definition is needed to deal with queries such as session-state reveal or ephemeral-key reveal, which can also occur in incomplete sessions.

## 6 Conclusions and Future Work

The complexity of strong KE 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 secure protocols. As a result, multiple "strong" models can coexist. Furthermore, if the practical implications of a security model are made clear, it becomes possible to choose among the security models based on the target application domain.

In this paper we have shown that the CK, eCK, and $\text{CK}_{\text{HMQV}}$ models for KE security are not only formally but also practically incomparable, thereby refuting several claims made in the literature, e. g. in [10–13]. For each model, there are attacks that it detects which are not detected by the other models.

Our analysis of the relations between the key derivation function and the definition of matching sessions reveals subtle mistakes in existing security proofs, e. g., for the Okamoto08 [10] and CMQV [11] protocols. Additionally, we show that a simple operation such as adding ordered names to the key derivation function can cause loss of functionality, as for example the NIST version of MQV [18], or even invalidate proofs, as for example in the case of the HMQV variant in [2]. We have shown subtleties of matching for incomplete sessions, and identify several errors in protocol proofs We have given a procedure to construct specifications of matching sessions that avoid the detected problems.

The flaws we detect in recent proofs show that the subtleties of strong KE models are not yet widely understood.

As future work it would be of interest to determine the exact relation between the guarantees provided by simulation based KE security notions [23] 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'01. 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. 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
5. Choo, K.K., Boyd, C., Hitchcock, Y., Maitland, G.: On session identifiers in provably secure protocols. **3352** (2005) 351–366
6. Menezes, A., Ustaoğlu, 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
7. Lee, J., Park, J.H.: Authenticated key exchange secure under the computational diffie-hellman assumption. Cryptology ePrint Archive, Report 2008/344 (2008) http://eprint.iacr.org/.
8. Cheng, Q., Han, G., Ma, C.: A new efficient and strongly secure authenticated key exchange protocol. Information Assurance and Security, International Symposium on **1** (2009) 499–502
9. Lee, J., Park, C.S.: An efficient authenticated key exchange protocol with a tight security reduction. Cryptology ePrint Archive, Report 2008/345 (2008) http://eprint.iacr.org/.
10. 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
11. Ustaoğlu, B.: Obtaining a secure and efficient key agreement protocol from (H)MQV and NAXOS. Des. Codes Cryptography **46**(3) (2008) 329–342
12. LaMacchia, B., Lauter, K., Mityagin, A.: Stronger security of authenticated key exchange. Cryptology ePrint Archive, Report 2006/073 (2006) http://eprint.iacr.org/.
13. 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.
14. 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)
15. Bellare, M., Canetti, R., Krawczyk, H.: A modular approach to the design and analysis of authentication and key exchange protocols (extended abstract). In: STOC '98: Proceedings of the thirtieth annual ACM symposium on Theory of computing, New York, NY, USA, ACM (1998) 419–428
16. 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
17. 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
18. 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)
19. Huang, H., Cao, Z.: Strongly secure authenticated key exchange protocol based on computational diffie-hellman problem. Cryptology ePrint Archive, Report 2008/500 (2008) http://eprint.iacr.org/.
20. Moriyama, D., Okamoto, T.: An eck-secure authenticated key exchange protocol without random oracles. In: ProvSec. Volume 5848 of Lecture Notes in Computer Science., Springer-Verlag (2009) 154–167
21. Kim, M., Fujioka, A., Ustaoğlu, B.: Strongly secure authenticated key exchange without naxos approach. In: IWSec. Volume 5824/2009 of Lecture Notes in Computer Science., Springer-Verlag (2009) 174–191
22. Kobara, K., Shin, S., Strefler, M.: Partnership in key exchange protocols. In: ASIACCS '09: Proceedings of the 4th International Symposium on Information, Computer, and Communications Security, New York, NY, USA, ACM (2009) 161–170
23. Canetti, R., Krawczyk, H.: Universally composable notions of key exchange and secure channels. In: EUROCRYPT'02. Lecture Notes in Computer Science, Springer-Verlag (2002) 337–351

## A Possible practical interpretations of the security models

In this appendix we give some possible practical interpretations of concepts that occur in KE security models.

It is not immediately clear how to interpret the definition of session identifiers in the CK model. In [1] it is suggested 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 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.

It is common to define the KE security model in terms of a game that involves a reactive system, in which the adversary is given access to a Send query. A Send query triggers a participant to perform three actions without being interrupted: receive a message, perform internal computations, and send a response. During these three actions, the adversary is not allowed to compromise state and/or ephemeral keys, in any of the models presented here. This restriction on the adversary behaviour is at least debatable from a practical point of view.

The $\mathrm{CK}_{\mathrm{HMQV}}$ 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.

## B Version history

The contents of this paper have evolved significantly over time and reflect the complexity and ambiguities in the definition of security models for key agreement.

### B.1 Version 1.0: June 2009

– Initial version.

### B.2 Version 1.1: July 2009

– Removed protocol specification as it was not sufficiently related to the subject and was lacking a full proof.
– Minor fixes to interpretations.

### B.3 Version 2.0: August 2009

– Significant update to key / partnering relations.
– Minor fixes to interpretations following discussions with A. Menezes and B. Ustaoğlu.

### B.4 Version 3.0: July 2010

– Attempt at improving formalization of existing security notions which should clarify the relations.
– Added section with suggested fixes to partnering definitions.
– Added CK timeline to clarify relative timing of queries.