Examining Indistinguishability-Based Security Models for Key Exchange Protocols: The case of CK, CK-HMQV, and eCK

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

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 the relation between the security guarantees provided by the protocols, is unclear. We show first 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 on protocols from the literature that are not considered by the other models. Third, our analysis enables us to find previously unreported flaws in protocol security proofs from the literature. We identify the causes of these flaws and show how they can be avoided.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols—*protocol verification*; D.4.6 [**Management of Computing and Information Systems**]: Security and Protection—*cryptographic protocols, authentication*

General Terms

Theory, Security

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. During recent years, numerous new protocols have been developed that are more efficient than their predecessors and provide stronger security guarantees. This has led to the development of increasingly stronger security models in which these protocols have been proven secure. Some examples are the 2-pass ISO signed Diffie-Hellman protocol in the CK model [5, 6], the HMQV protocol in a closely related model $\mathrm{CK}_{\mathrm{HMQV}}$ [16], and the Naxos protocol in the eCK model [17, 18]. 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. Such claims occur often in the literature, e. g., [8, 17, 18, 20, 21, 24, 26, 27]. However, given the subtle differences among the models, this conclusion is not obvious. In fact, the many technical differences suggest that the models are formally incomparable. Such claims have also been made, e. g., [4, 11], contradicting the works cited above. To further complicate matters, we observe that even if two models are incomparable for minor technical reasons, it may still be the case that one model is stronger than another model for all realistic protocols.

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 analysing and 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.

There is a limited amount of related work investigating various notions of KE security. Some earlier models for key exchange have been compared [9]. The use of session identifiers in KE models has been studied [10]. The CK [5] model has been related to one of its variants with respect to specification of peers before or after the session [22]. Several authors have suggested that the eCK model is the strongest security model, e. g., [8, 17, 18, 20, 21, 24, 26, 27]. In contrast to recent work that shows that eCK is not the strongest model [11], our work focusses not on a single query difference between two models but on the full security models, and relates multiple models in detail.

Contributions. First, we show that the CK, $\mathrm{CK}_{\mathrm{HMQV}}$, and eCK models are *formally* incomparable, based on their security prerequisites, adversary model, and application do-

*This work was supported by the Hasler Foundation within the ComposeSec project and by the ETH Research Grant ETH-30 09-3.

ASIACCS '11, March 22–24, 2011, Hong Kong, China.

main. 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 on protocols from the literature that are detected in one model but are not considered in the others, and vice versa. Our findings imply that the three models are not only formally but also *practically* incomparable.

Third, we identify common sources of errors in protocol security proofs based on these models. We analyze recent protocol security proofs and find 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 CK_{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 several subtleties in recent proofs and related KE security models, and show how to avoid common problems. We consider possible practical interpretations of each model in Section 6. We draw conclusions and discuss future work in Section 7.

Acknowledgements

The author is grateful to Berkant Ustaoğlu and Alfred Menezes, whose constructive comments and insightful discussions have lead to a complete rewrite of an earlier version of this paper, to Michèle Feltz for many constructive suggestions, and to the anonymous reviewers.

2. THREE SECURITY MODELS FOR KEY EXCHANGE PROTOCOLS

We first provide a high-level overview of KE models before describing three models in detail.

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 general 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 able to distinguish the exchanged session key from a random bit string with more than non-negligible probability.

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. Additionally, the adversary may have additional powers, such as revealing some long-term or session keys, revealing the random numbers generated by participants, or revealing parts of the session-state of some sessions.

One important element of KE models 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). Note that earlier KE models, e. g., [3], simply assumed the existence of a matching session definition: however, as we will show in Section 5.3, only assuming existence is not sufficient when the adversary can perform queries on incomplete sessions, as is the case for the models considered here.

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 CK_{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 that we consider in this paper all 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 computed or handled by the same algorithm when abstracting away from the order in which they are sent or received. Many implicitly authenticated 2-message KE protocols such as MQV are role-symmetric: In both roles sending a message consists of generating a random ephemeral key z and sending g^z. Receiving a message is also dealt with in the same way in both roles. In contrast, most variants of signed Diffie-Hellman are not role-symmetric. For signed Diffie-Hellman protocols, the first message obviously does not depend on any previously communicated messages, but the second message usually contains an element of the first message that was

received, such as the initiators ephemeral public key.

Note that role-symmetry does not imply that the key is computed in the same way for both roles. For example, the Naxos protocol [18] is role-symmetric because its messages are computed and handled using the same algorithm for both roles, except for their order. However, in the key computation, an agent in the initiator role puts his name as the one-before-last parameter of the input to the hash function H_2, while an agent in the responder role puts his own name as the last parameter.

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 a security model for key-exchange protocols [5, 6], 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 [5]. In the original paper the CK model is called the "SK-security" model in the "unauthenticated links model (UM)" [5, 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 other models defined in [5]. This includes the SK-security model in the *authenticated* links model (AM) [5, 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 [5]. 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 [2]. 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 a 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 [5, p. 16]. Additionally, in [5] variants of both models are defined that do not consider perfect forward secrecy. In these alternative models sessions cannot be expired. This restriction implies that a reveal of the long-term keys of the actor or the peer exposes the test session, which effectively disallows the reveal of these keys.

Analysis of a protocol in the CK model requires that the protocol includes session identifiers. There are two requirements on session identifiers [5, p. 4]. First, two different sessions at the same party A are required to have different session identifiers [5, 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 $s_A = s'_B \wedge s'_A = s_B \wedge s_{\mathrm{sid}} = s'_{\mathrm{sid}}$ [5, p. 11]. Note that CK-matching sessions may be performing the same role.[2]

In the CK security experiment, the participants start 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 perform a sequence of queries from the following set [5, p. 9].

- activate session s, which can take two forms. The first form, action request q, 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 $s_A := A, s_B := B, s_{\mathrm{sid}} := sid, s_{\mathrm{role}} := r$.

  The second form, incoming message m with sender s_b, models 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 [5, 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 [5, p. 11]. 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.

[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. Furthermore, 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 [5, p. 4] may seem to suggest that the roles of CK-matching sessions are different, but the technical description [5, p. 11] clearly mentions that this is not required.

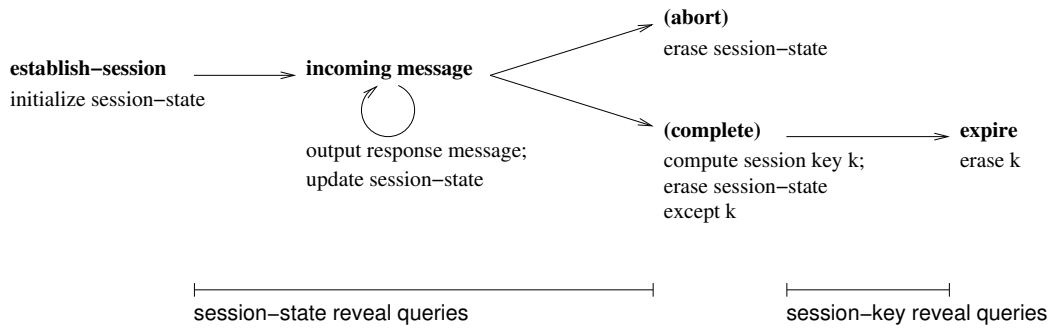

Figure 1: Lifetime of a session in CK. Although session-state reveal queries are allowed after the session is completed, the returned state will be empty [5, p. 11], and we omit these from the graph.

- session-key reveal[3] a completed session s, revealing s's session key.
- test-session query a completed but unexpired session s. Additionally s must not be *exposed*: this notion will be defined below. 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, p. 13/14].

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* [5, 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, or
- 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. [5, p. 12].

A sequence of queries in an experiment must satisfy the following constraints [5, p. 14].

- The test-session query can only be performed once.
- The test-session must not be exposed by the time the test query is issued and the adversary is not allowed to expose it until the end of the experiment.

Definition 1. (Secure in the CK model) [5, 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.

[3]This query is called session-output query at [5, p. 9], but later renamed in the context of KE-protocols [5, p. 11].

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

Krawczyk [16] uses 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* [16, 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 [16, 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 $s_A = s'_B \wedge s'_A = s_B \wedge s_{send} = s'_{recv} \wedge s'_{send} = s_{recv}$ [16, p. 28]. For incomplete sessions, no definition for matching is provided in $\mathrm{CK}_{\mathrm{HMQV}}$ [16].
- 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 [5, p. 17], [16, p. 29].
- Whereas in CK the adversary can only learn the long-term key of the peer after the end of the partner session, the $\mathrm{CK}_{\mathrm{HMQV}}$ model allows the adversary to learn the long-term keys of the peer even before the start of the partner session, as long as the adversary does not actively interfere with the communication between the test session and its partner session (in particular when checking for PFS, [16, p. 42]).
- 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 [16, p. 29][4]

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

A session s is said to be *CK_{HMQV}-clean* [16, 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.

- $\text{CK}^{\text{KCI}}_{\text{HMQV}}$ [16, p. 41]: In addition to the $\text{CK}^{\text{basic}}_{\text{HMQV}}$ 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 CK_{HMQV}-clean, and (ii) s_B is not corrupted.
- $\text{CK}^{\text{wPFS}}_{\text{HMQV}}$ [16, p. 42]: In addition to the $\text{CK}^{\text{basic}}_{\text{HMQV}}$ 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 CK_{HMQV}-clean, and (ii) a CK_{HMQV}-matching session of s exists[5], and (iii) all CK_{HMQV}-matching sessions of s are CK_{HMQV}-clean.
- $\text{CK}^{\text{eph}}_{\text{HMQV}}$ [16, p. 44/45]: In addition to the $\text{CK}^{\text{basic}}_{\text{HMQV}}$ queries, the adversary is allowed to reveal the ephemeral secret keys of all sessions.

The basic security proof of HMQV in the $\text{CK}^{\text{basic}}_{\text{HMQV}}$ model depends on the computational Diffie-Hellman (CDH) assumption. The main reason to consider multiple models is that the security of HMQV in the $\text{CK}^{\text{eph}}_{\text{HMQV}}$ model depends on stronger assumptions (Gap Diffie-Hellman and KEA1) [16, p. 45].

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

Remark. The notion of weak perfect forward secrecy (wPFS) is introduced because the basic version HMQV belongs to a class of protocols that cannot satisfy PFS [16]. A generic attack is sketched by Krawczyk [16, p. 15] but no proof is given. The class of protocols seems to cover all implicitly-authenticated two-message KE protocols. Note that contrary to several statements made in the literature, this result does not hold for all two-message KE protocols.

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 [16, 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 CK_{HMQV}-matches the test session. This requirement is slightly too strong: CK_{HMQV}-matching also requires that the matching session of the initiator role has received the final message. This is not needed to prevent the attack.

Remark. The CK_{HMQV} model [16] is presented as if it were a specialization of the CK model. However, it is not a specialization in any technical sense but rather a closely related but *incomparable* model. Correctness of a protocol in either model does not imply correctness in the other model.

2.5 The eCK security model

The eCK security model ("extended-CK") was defined by LaMacchia, Lauter, and Mityagin [17, 18]. There are minor differences between these two works. Because [17] was published significantly before [18], we choose to consider the latter as the proper definition of the eCK model.

We say that two sessions s and s' are *eCK-matching* if and only if $s_A = s'_B \wedge s'_A = s_B \wedge s_{send} = s'_{recv} \wedge s'_{send} = s_{recv} \wedge s_{role} \neq s'_{role}$ [18, p. 7/8].[6] Observe that eCK-matching is equal to (CK_{HMQV}-matching $\wedge\, s_{role} \neq s'_{role}$).

In the eCK model, the adversary can perform the following queries [18, 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.[7]
- $\mathsf{Reveal}(s)$. 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: [18, 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.

[4] In the context of the HMQV proof in the $w\text{CK}^{\text{basic}}_{\text{HMQV}}$ model, the CK_{HMQV} definition of session-state seems to render the session-state reveal query useless, because it can only be performed on sessions that are not completed, i. e., before computation of the session keys, and thus *before* the session keys (or any intermediate computations for the key) are part of the revealable session state. The only non-public information in incomplete sessions of HMQV is the ephemeral key, but this is explicitly excluded from the state [16, p. 29], and therefore performing the session-state reveal query does not seem to provide the adversary with any information he did not know before. Note that the leakage of the ephemeral keys is analyzed separately in the $\text{CK}^{\text{eph}}_{\text{HMQV}}$ model.

[5] The existence of the CK_{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.

[6] 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.

[7] In the original description [18] 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).

- The experiment includes Reveal(s).
- A session s' exists that is eCK-matching with s, and the experiment includes Reveal(s').
- The experiment includes both Long-term Key Reveal(s_A) and Ephemeral Key Reveal(s).
- A session s' exists that is eCK-matching with s, and the experiment includes both Long-term Key Reveal(s_B) and Ephemeral Key Reveal(s').
- No session exists that is eCK-matching with s, and the experiment includes Long-term Key Reveal(s_B).

An eCK experiment must satisfy the following constraints [18, p. 8/9]:

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

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

Definition 2. (eCK security) [18, 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 $\mathbf{Adv}_P^{KE}(\mathcal{M}) = \Pr[\mathcal{M} \text{ wins}] - \frac{1}{2}$.

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 Differences in 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 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).

We focus on two requirements:

- Matching sessions must compute the same key.
- The session key of the test session must be indistinguishable from the keys computed by non-matching sessions (as the latter may be revealed).

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

$$s \approx_A s' \stackrel{\text{def}}{=} \big(s_A = s'_B \wedge s'_A = s_B \wedge s_{send} = s'_{recv} \wedge s'_{send} = s_{recv}\big) \quad (1)$$

$$s \approx_B s' \stackrel{\text{def}}{=} \big(s \approx_A s' \wedge (s_R \neq s'_R \vee s_A = s_B)\big) \quad (2)$$

$$s \approx_C s' \stackrel{\text{def}}{=} \big(s \approx_A s' \wedge s_R \neq s'_R\big) \quad (3)$$

$$s \approx_D s' \stackrel{\text{def}}{=} \big(s_A = s'_B \wedge s'_A = s_B \wedge s_{sid} = s'_{sid}\big) \quad (4)$$

In the CK_{HMQV} model two completed sessions s and s' are matching iff $s \approx_A s'$ [16, p. 10]. 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 [18, p. 7]. 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 [5, p. 11]. 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 a 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', we have that

$$s \approx s' \Rightarrow KDF_P(s) = KDF_P(s')$$

and also that, with overwhelming probability,

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

As examples, we observe that MQV [19] has key type $\approx_A$, the NIST-variant of MQV [1] has key type $\approx_B$, Naxos [18] has key type $\approx_C$, and the signed Diffie-Hellman variant from [5] has key type $\approx_D$.

Note that in this section we focus on the differences in matching with respect to *completed* sessions. We will return to the differences in matching with respect to *incomplete* sessions in Section 5.2.

3.1.1 CK *security prerequisites.*

In CK, completed sessions s and s' are matching iff $s \approx_D s'$ [5, p. 11].[8] Protocol messages are required to include the

[8]Note that although the informal introduction of the CK model [5, p. 4] suggests that roles must be distinct, the technical description of the model clearly states that roles are not required to be distinct [5, p. 11].

	CK	CK_{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 ($\approx_D$)	CK_{HMQV}-matching ($\approx_A$)	eCK-matching ($\approx_C$)
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 test_A	if test has expired, test_A can be corrupted	[wPFS]: if session matching test exists, and both CK_{HMQV}-clean; [KCI]: if test session CK_{HMQV}-clean, and test_B not corrupted; [basic,eph]: never	ephemeral keys of test not revealed
reveal long-term key of test_B	if the session that CK-matches test has expired, test_B can be corrupted	[wPFS]: if session that CK-matches test exists, and both test and the matching session are CK_{HMQV}-clean; [basic,KCI,eph]: never	if eCK-matching sessions exist, and the ephemeral keys of these sessions are not revealed
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 CK_{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 CK_{HMQV}-matching test	never

Table 1: Summary of formal differences between the models. For the CK_{HMQV} model we identify the relevant submodels in square brackets.

session identifier [5, 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 with overwhelming probability, which violates condition 1 of SK-security in the CK model [5, p. 14]. Therefore P is not secure in the CK model. □

3.1.2 CK_{HMQV} *security prerequisites.*

In CK_{HMQV}, completed sessions s and s' are matching iff $s \approx_A s'$ [16, p. 10].

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 CK_{HMQV} is based on the relation $\approx_A$, and the violated condition is (1) in Def. 11 of [16, p. 11]. □

3.1.3 eCK *security prerequisites.*

In eCK, completed sessions s and s' are matching iff $s \approx_C s'$ [18, 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 [18, 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 Differences in adversarial capabilities

3.2.1 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 [5, 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 [5, p. 14]. As a result, the CK model is not able to detect *key compromise impersonation* (KCI) attacks [13]. After the test session expires, the adversary is allowed to corrupt the participant that executes the test session [5, 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 (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 cannot accept this incoming message, as a session identifier can only occur once at each participant [5, p. 11].

3.2.2 CK_{HMQV} *adversary capabilities.*

The CK_{HMQV} model allows for state-reveal queries [16, p. 6] as in the CK model. In order to detect KCI attacks [13], 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) [16, p. 41]. The $\text{CK}^{\text{eph}}_{\text{HMQV}}$ model [16, 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) in the $\text{CK}^{\text{wPFS}}_{\text{HMQV}}$ model is allowed if a matching session exists [16, p. 42]. Secrecy with respect to this definition is known as *weak Perfect Forward Secrecy* (wPFS).

3.2.3 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 [18, 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 [18, p. 9] and thus can be used to detect KCI attacks. The corruption of the peer is allowed if a matching session exists [18, p. 9] but the ephemeral keys of this session are not revealed. This includes checking for wPFS.

4. PRACTICALLY RELATING THE THREE SECURITY MODELS

Even if two key exchange models are theoretically incomparable, it may still be the case that, for all practical protocols, correctness in one model implies correctness in the other model and vice versa. In this section we show *practical* incomparability of the models, i. e., we show that in each model, attacks on protocols from the literature exist that are not considered in the other models.

(1) eCK security does not imply CK_{HMQV} security. We provide two attacks in the CK_{HMQV} model on the two-message Naxos protocol [18], which was proven secure in the eCK model. Recall that the eCK model does not consider attacks that involve intermediate computations of the protocol, whereas the CK_{HMQV} model allows the session-state reveal query for this purpose. An attack on Naxos has been described that exploits revealing intermediate computations [11]. The first 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. This attack is also possible in the CK_{HMQV} model. Second, the Naxos protocol is role-symmetric and has key type $\approx_C$ and is therefore insecure in the CK_{HMQV} model: matching initiators do not compute the same key.

(2) eCK security does not imply CK security. The first attack given above in (1), which exploits session-state reveal to attack the Naxos protocol, also applies in the CK model.

(3) CK security does not imply eCK security. A counterexample using the reveal of ephemeral keys of the test session is described along with the eCK model [18]. The basic signed Diffie-Hellman protocol variant [5] 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) CK security does not imply CK_{HMQV} security. The counterexample from (3) also applies in the $\text{CK}^{\text{eph}}_{\text{HMQV}}$ model.

(5) CK_{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.

(1) CK_{HMQV} security does not imply CK security. As a counterexample, we observe that the HMQV protocol [16] was proven secure in CK_{HMQV}, does not provide perfect forward secrecy. If we add CK-model style session identifiers to each message and the key derivation function, the resulting protocol is still secure in the CK_{HMQV} model. However, it is not secure in the CK model: the generic attack on PFS [16] applies to this protocol in the CK model.

5. ANALYSIS OF PROOFS AND RELATED KE MODELS

Although security models based on the CK model have been used for almost a decade, many recent proofs still con-

protocol	key equivalence type of protocol	matching sessions equivalence type in proof	comments
HMQV [16, p. 3]	$\approx_A$	$\approx_A$	Matching initiators compute the same key
MQV [19, p. 131]	$\approx_A$	n.a.	Matching initiators compute the same key
MQV (NIST) [1, p. 46]	$\approx_B$	n.a.	Matching initiators do not compute the same key (unlike MQV)
HMQV variant [16, 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 [24]	$\approx_C$	$\approx_A$	Flaw in proof: Matching sessions do not always compute the same key
CMQV [26]	$\approx_C$	$\approx_A$	Flaw in proof: Matching sessions do not always compute the same key
HuangCao [12]	$\approx_C$	$\approx_A$	Matching sessions do not always compute the same key (but requirement omitted from the model)

Table 2: Key equivalence versus matching sessions in protocols

tain 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 discuss two of these elements: first, the interaction between key equivalence types and matching sessions in role-symmetric protocols, and second, the interaction between queries that can be performed on incomplete sessions and the definition of matching sessions.

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

We applied the observations described in Section 3.1 and Table 1 to several role-symmetric protocols from the literature, and summarized the results in Table 2. 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.

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 [24] and CMQV [26] 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 exchange, 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 relevance of matching 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, can be performed on incomplete sessions and depend on the definition of *matching*. For example, session-state reveal in CK and $\mathrm{CK}_{\mathrm{HMQV}}$ can only be performed on sessions that are not the test session and do not match the test session. Thus, the definition of matching for incomplete sessions influences whether session-state reveal queries are possible. Similarly, for statements like "a matching session exists" in eCK when revealing the long-term key of the peer, 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 $\mathrm{CK}_{\mathrm{HMQV}}$ and eCK models.

In $\mathrm{CK}_{\mathrm{HMQV}}$, the notion of matching is only defined for completed sessions [16, p. 29]. As a result, it is undefined whether session-state reveal queries, which are only allowed on incomplete sessions, are allowed in $\mathrm{CK}^{\mathrm{basic}}_{\mathrm{HMQV}}$. However, as observed earlier, the session-state query is redundant in the security proof of HMQV in $\mathrm{CK}^{\mathrm{basic}}_{\mathrm{HMQV}}$, and the ephemeral-key leakage is handled separately in the $\mathrm{CK}^{\mathrm{eph}}_{\mathrm{HMQV}}$ model [16, Section 7], 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 nev-

er match the (completed) test session. In follow-up works to the security models presented here, some authors have closely followed the original eCK formulation, e. g. [23, 24]. Other authors have adapted the definition of matching sessions, e. g., [12,14,20,21,26], with the result that incomplete sessions that could be completed to a matching session (by adding additional queries), are considered to be matching sessions as well. This modification strengthens the adversary model slightly: Consider the case in which there exists an incomplete session that might be completed to a session that matches the test session, but no (completed) sessions exist that eCK-match the test session. In the original eCK model, the long-term keys of the peer may not be revealed in such a situation because no eCK-matching session exists. If incomplete sessions may be matching, as in some of the follow-up works, then the long-term keys of the peer can be revealed in this situation.

5.3 On assuming only existence of the matching relation

In the three-party Bellare Rogaway security model [3], no explicit definition is given for when two sessions match (in [3], matching is called *partnering*). Instead, the existence of a matching relation is simply assumed in the KE model. Because the adversary is allowed to reveal the session-keys of non-matching sessions, the matching relation (for any protocol that is correct in the model) must ensure that non-matching sessions compute different keys. This setup may sometimes lead to unintuitive matching definitions, but seems reasonable in the case that reveal queries are only considered on completed sessions.

However, when considering reveal queries that depend on partnering and that may be performed on incomplete sessions, such as the session-state reveal query in the CK and $\mathrm{CK}_{\mathrm{HMQV}}$ models, only assuming existence is not adequate. The reason is that the requirements on the existence only provide constraints for *completed* sessions. Thus, the requirements on the matching relation can be met by a matching relation that considers all incomplete sessions to be matching to all other sessions. This effectively disallows the adversary from performing any session-state reveal queries on incomplete sessions. In other words, a security proof in a model that assumes only the existence of a suitable partner relation in this way may not provide any guarantees against session-state reveal queries on incomplete sessions.

5.4 Avoiding matching problems

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

First, 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.

Second, ensure that the key derivation function computes the same key for two completed sessions if and only if the sessions are matching.

Finally, 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 [15], it is proposed to define matching (called partnering in [15]) 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. POSSIBLE PRACTICAL INTERPRETATIONS OF THE SECURITY MODELS

In this section 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 the CK model [5] 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 from 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, Hardware Security 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 or the presence of malware, as this 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 *atomically*, i. e., without being interrupted: receive a message, perform internal computations, and send a response. Consequently, the adversary is not allowed to compromise session-state contents or ephemeral keys during these three actions. Thus, if one faithfully models the evolving session-state contents, any intermediate computations that are only needed during these three steps and erased afterwards can never be revealed by the adversary. This restriction on the adversary is at least debatable from a practical point of view.

The CK_{HMQV} security notion was developed from the CK model, 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, which in turn helps to achieve deniability [25]. The change of partnering function seems to be driven by the symmetry of the roles and the fact that pre-established session identifiers may not be available.

7. 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, mainly 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. However, 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 CK_{HMQV} models for KE security are not only formally but also practically incomparable, thereby refuting several claims made in the literature, e. g., [17,24,26,27]. For each model, there are attacks on protocols from the literature that it detects but 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 Okamoto [24] and CMQV [26] 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 [1], or even invalidate proofs, as for example in the case of the HMQV variant in [16]. We have shown subtleties of matching for incomplete sessions, and have given a procedure to construct specifications of matching sessions that avoid these 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 [7] and the security models considered here.

8. REFERENCES

[1] E. Barker, D. Johnson, and M. Smid. NIST special publication 800-56A: Recommendation for pair-wise key establishment schemes using discrete logarithm cryptography (revised). Technical report, NIST, March 2007.

[2] M. Bellare, R. Canetti, and H. Krawczyk. 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*, pages 419–428, New York, NY, USA, 1998. ACM.

[3] M. Bellare and P. Rogaway. Provably secure session key distribution: the three party case. In *STOC '95: Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*, pages 57–66, New York, NY, USA, 1995. ACM.

[4] C. Boyd, Y. Cliff, J. M. G. Nieto, and K. G. Paterson. One-round key exchange in the standard model. *IJACT*, 1(3):181–199, 2009.

[5] R. Canetti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. Cryptology ePrint Archive, Report 2001/040, 2001. `http://eprint.iacr.org/`.

[6] R. Canetti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In *EUROCRYPT'01*, volume 2045 of *LNCS*, pages 453–474. Springer, 2001.

[7] R. Canetti and H. Krawczyk. Universally composable notions of key exchange and secure channels. In *EUROCRYPT'02*, Lecture Notes in Computer Science, pages 337–351. Springer-Verlag, 2002.

[8] Q. Cheng, G. Han, and C. Ma. A new efficient and strongly secure authenticated key exchange protocol. *Information Assurance and Security, International Symposium on*, 1:499–502, 2009.

[9] K.-K. Choo, C. Boyd, and Y. Hitchcock. Examining indistinguishability-based proof models for key establishment proofs. In *ASIACRYPT*, volume 3788 of *Lecture Notes in Computer Science*, pages 624–643. Springer, 2005.

[10] K.-K. Choo, C. Boyd, Y. Hitchcock, and G. Maitland. On session identifiers in provably secure protocols. In *SCN*, volume 3352 of *Lecture Notes in Computer Science*, pages 351–366. Springer-Verlag, 2004.

[11] C. Cremers. Session-state Reveal is stronger than Ephemeral Key Reveal: Attacking the NAXOS key exchange protocol. In *ACNS'09*, Lecture Notes in Computer Science, 2009.

[12] H. Huang and Z. Cao. Strongly secure authenticated key exchange protocol based on computational Diffie-Hellman problem. Cryptology ePrint Archive, Report 2008/500, 2008. `http://eprint.iacr.org/`.

[13] M. Just and S. Vaudenay. Authenticated multi-party key agreement. In *Advances in Cryptology-ASIACRYPT 1996*, volume 1163 of *Lecture Notes in Computer Science*, pages 36–49, 1996.

[14] M. Kim, A. Fujioka, and B. Ustaoğlu. Strongly secure authenticated key exchange without NAXOS' approach. In *IWSec*, volume 5824/2009 of *Lecture*

Notes in Computer Science, pages 174–191. Springer-Verlag, 2009.

[15] K. Kobara, S. Shin, and M. Strefler. Partnership in key exchange protocols. In *ASIACCS '09: Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, pages 161–170, New York, NY, USA, 2009. ACM.

[16] H. Krawczyk. HMQV: A high-performance secure Diffie-Hellman protocol. In *CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 546–566. Springer-Verlag, 2005.

[17] B. LaMacchia, K. Lauter, and A. Mityagin. Stronger security of authenticated key exchange. Cryptology ePrint Archive, Report 2006/073, 2006. `http://eprint.iacr.org/`.

[18] B. LaMacchia, K. Lauter, and A. Mityagin. Stronger security of authenticated key exchange. In *ProvSec*, volume 4784 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2007.

[19] L. Law, A. Menezes, M. Qu, J. Solinas, and S. Vanstone. An efficient protocol for authenticated key agreement. *Designs, Codes and Cryptography*, 28:119–134, 2003.

[20] J. Lee and C. S. Park. An efficient authenticated key exchange protocol with a tight security reduction. Cryptology ePrint Archive, Report 2008/345, 2008. `http://eprint.iacr.org/`.

[21] J. Lee and J. H. Park. Authenticated key exchange secure under the computational Diffie-Hellman assumption. Cryptology ePrint Archive, Report 2008/344, 2008. `http://eprint.iacr.org/`.

[22] A. Menezes and B. Ustaoğlu. Comparing the pre- and post-specified peer models for key agreement. In *Proceedings of ACISP 2008*, volume 5107 of *Lecture Notes in Computer Science*, pages 53–68, 2008.

[23] D. Moriyama and T. Okamoto. An eCK-secure authenticated key exchange protocol without random oracles. In *ProvSec*, volume 5848 of *Lecture Notes in Computer Science*, pages 154–167. Springer-Verlag, 2009.

[24] T. Okamoto. Authenticated key exchange and key encapsulation in the standard model. In *ASIACRYPT*, volume 4833 of *Lecture Notes in Computer Science*, pages 474–484, 2007.

[25] M. D. Raimondo, R. Gennaro, and H. Krawczyk. Deniable authentication and key exchange. Cryptology ePrint Archive, Report 2006/280, 2006. `http://eprint.iacr.org/`.

[26] B. Ustaoğlu. Obtaining a secure and efficient key agreement protocol from (H)MQV and NAXOS. *Des. Codes Cryptography*, 46(3):329–342, 2008.

[27] J. Xia, J. Wang, L. Fang, Y. Ren, and S. Bian. 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.