Session-state Reveal is stronger than Ephemeral Key Reveal: Breaking the NAXOS key exchange protocol

MANUSCRIPT

Cas J.F. Cremers

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

September 4, 2008

Abstract

In the papers "Stronger Security of Authenticated Key Exchange" [LLM07, LLM06], a new security model for key exchange protocols is proposed. The new model is suggested to be at least as strong as previous models. In particular, the model includes a new notion of an Ephemeral Key Reveal adversary query, which is claimed in [LLM06, Oka07, Ust08] to be at least as strong as existing definitions of the Session-state Reveal query. We show that for some protocols, a straightforward interpretation of Session-state Reveal is strictly stronger than Ephemeral Key Reveal. In particular, we show that the NAXOS protocol from [LLM07, LLM06] does not meet its security requirements if the Session-state Reveal query is allowed in the security model.

1 Introduction

In the area of secure key agreement protocols many security models [BCPQ01, MU08, LLM07, CBH05, CK01, BPR00] and protocols have been proposed. Many of the proposed protocols have been shown to be correct in some particular security model, but have also shown to be incorrect in others. In order to get a grasp on the exact properties that are required from such protocols, a single unified security model would be desirable. However, given the very recent works such as [MU08] on security models for key agreement protocols, it seems that a single model is still not agreed upon.

In this paper we focus on a very specific detail of one such security model for key agreement protocols. In particular, we focus on the ability of the adversary to learn the local state of an agent. For example, when an agent chooses a random value, or computes the hash function of a certain input, many elements of the computation reside temporarily in the local memory of the agent. It may be possible for the adversary to learn such information. This ability is captured in security models for key agreement protocols by the Session-state Reveal query.

A drawback of the Session-state Reveal query in current security models such as the Cannetti-Krawczyk (CK) model [CK01], is that the query is underspecified. It is not clear what is exactly revealed in such a query for a given protocol. Effectively, this decision is postponed to the proof of a particular protocol.

In [LLM06,LLM07] a security model is proposed which is said to be stronger than existing AKE (Authenticated Key Exchange) security models. The model is based on the CK model, and is referred to in [LLM07] as the Extended Canetti-Krawczyk (eCK) model. The eCK model differs in a number of aspects from the CK model, where the main difference seems to be that the adversary is allowed to reveal part of the local state of participants even during a normal protocol session. A more subtle aspect in which the eCK model differs from the CK model is that it replaces the Session-state Reveal query by a new Ephemeral Key Reveal query. In this paper we focus on this aspect.

The Session-state Reveal query from the CK model leaves underspecified what exactly is revealed. To address this, the eCK model (re)defines the notion of ephemeral key and introduces a corresponding Ephemeral Key Reveal query that reveals this key. The ephemeral key is defined to contain all secret session-specific information. The authors argue for the new Ephemeral Key Reveal query that "by setting the ephemeral secret key equal to all session-specific secret information, we seem to cover all definitions of Session-state Reveal queries which exist in literature" [LLM06, p.2]. Similar arguments can be found in [Ust08, Oka07]. Within the resulting eCK model, the NAXOS protocol is proposed and proven correct in [LLM07].

In [BCNP08] it is argued that strictly speaking the eCK and CK models are incomparable. Regarding the difference between Session-state Reveal and Ephemeral Key Reveal, it is remarked that "The important point to note is that the ephemeral-key does not include session state that has been computed using the long-term secret of the party. This is not the case in the CK model where, in principle, the adversary is allowed access to all the inputs (including the randomness, but excluding the long-term secret itself) and the results of all the computations done by a party as part of a session" [BCNP08, Section 3.1]. However, the authors also remark that "it is arguable whether the differences between the two models are meaningful in reality" [BCNP08, Section 3.1].

In this paper we show that contrary to the claims in [LLM06, Ust08, Oka07], Ephemeral Key Reveal is not as strong as Session-state Reveal. We show this by providing two attacks on the NAXOS protocol, which can be performed using Session-state Reveal, but cannot be performed by using Ephemeral Key Reveal. The security model we use is nearly identical to the eCK model: we only replace Ephemeral Key Reveal by Session-state Reveal. Furthermore, our attacks also seem to be valid in the CK model, which implies that there is a meaningful difference between CK and eCK, as NAXOS was proven correct in the eCK model.

The assumption needed for our attacks is that when a participant computes $H(x)$, where H is a hash function, then $x_1, \dots, x_n$ are be in the local state before the computation, excluding any long-term keys. More precisely, a Session-state Reveal query just before the computation of $H(x_1, \dots, x_n)$ reveals all x_i $(0 < i \leq n)$ except for any long-term keys.[1]

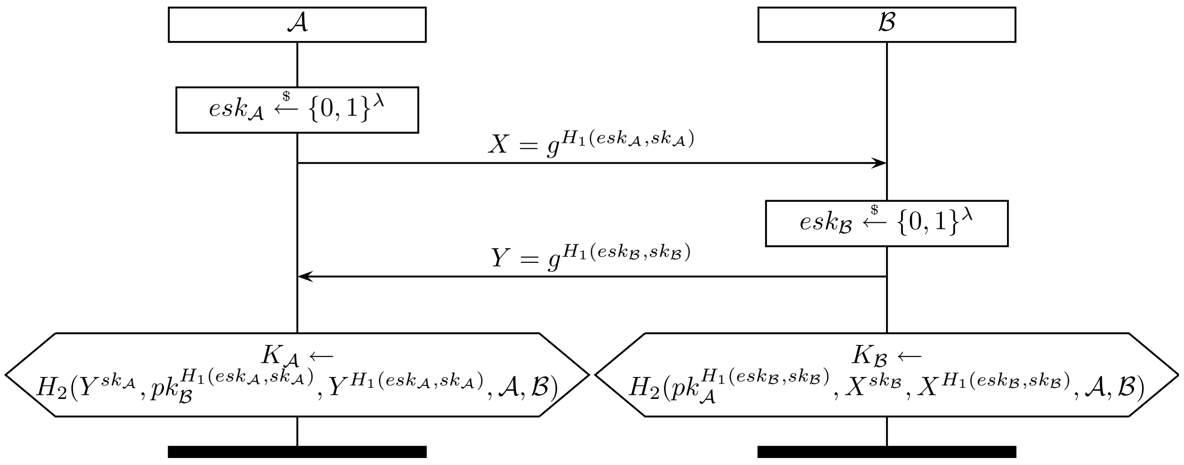
We base our assumption on two generic observations and two observations specific to NAXOS. First, our interpretation of Session-state Reveal does not seem to violate the constraints on the security model specified in [CK01], and therefore seems a valid possible interpretation of the CK model. Second, given that the difference between long-term private keys and local state seems to be inspired by TPM based scenarios in which only the long-term keys are protected, it seems reasonable to assume that intermediate computations are performed in local memory. Third, for protocols such as NAXOS, it seems that such a value x_i needs

[1]An example of an execution model for NAXOS where this assumption holds is a TPM which securely stores the long-term keys, and provides an interface to compute $H_1(x, sk_{\mathsf{a}})$ for a given x, where sk_{a} is the long-term private key of the agent. Alternatively, H_1 may be completely computed inside the TPM whereas H_2 may be computed in unprotected memory.

to be stored in memory for a similar time duration as the ephemeral key. Fourth, in the case of NAXOS, if we assume H_2 is computed entirely in a TPM, there is no reason to store the ephemeral key in unprotected memory, in which case the local state effectively becomes empty.

We proceed as follows. In Section 2 we explain some notation, and present the NAXOS protocol. Then, in Section 3 we show two attacks on this protocol that use Session-state Reveal. We conclude in Section 4.

2 The NAXOS key exchange protocol



$\mathcal{A}$
$\mathcal{B}$
$esk_{\mathcal{A}} \overset{\$}{\leftarrow} \{0,1\}^{\lambda}$
$X = g^{H_1(esk_{\mathcal{A}}, sk_{\mathcal{A}})}$
$esk_{\mathcal{B}} \overset{\$}{\leftarrow} \{0,1\}^{\lambda}$
$Y = g^{H_1(esk_{\mathcal{B}}, sk_{\mathcal{B}})}$
$K_{\mathcal{A}} \leftarrow H_2(Y^{sk_{\mathcal{A}}}, pk_{\mathcal{B}}^{H_1(esk_{\mathcal{A}}, sk_{\mathcal{A}})}, Y^{H_1(esk_{\mathcal{A}}, sk_{\mathcal{A}})}, \mathcal{A}, \mathcal{B})$
$K_{\mathcal{B}} \leftarrow H_2(pk_{\mathcal{A}}^{H_1(esk_{\mathcal{B}}, sk_{\mathcal{B}})}, X^{sk_{\mathcal{B}}}, X^{H_1(esk_{\mathcal{B}}, sk_{\mathcal{B}})}, \mathcal{A}, \mathcal{B})$

Figure 1: The NAXOS protocol. At the end of a normal execution we have that $K_{\mathcal{A}} = K_{\mathcal{B}}$ ($pk_x = g^{sk_x}$).

The NAXOS protocol is shown in Figure 1. The purpose of the protocol is to establish a shared symmetric key between two parties. Both parties have a long-term private key, e. g. sk_{a}, and initially know the public key of all other participants, e. g. pk_{b}. In Table 1 we give an overview of the notation used in the protocol as well as the remainder of this paper. We follow the notation from [LLM07] where possible.

The protocol is designed to be secure in a very strong sense: the adversary is assumed to have the capability of learning long-term private keys, and also has the capability of learning short term data generated during a protocol session that does not include the private key.

The intuition behind the design of the protocol is that by strongly connecting the long-term private key with the short term ephemeral key, the adversary would need to have both of these elements to construct an attack. For example, the protocol should be secure if the adversary either (a) learns the long-term key of a participant during a session, or (b) learns the short-term data (except for the long-term key) of a participant during a session. A typical scenario for (b) is that the participant stores the long-term key on a TPM, and computes other operations in unprotected memory. For full details we refer the reader to [LLM07, LLM06].

At the end of a normal protocol execution, the session key is computed as

$$H_2(g^{H_1(esk_{\mathcal{B}}, sk_{\mathcal{B}})sk_{\mathcal{A}}}, g^{H_1(esk_{\mathcal{A}}, sk_{\mathcal{A}})sk_{\mathcal{B}}}, g^{H_1(esk_{\mathcal{A}}, sk_{\mathcal{A}})H_1(esk_{\mathcal{B}}, sk_{\mathcal{B}})}, \mathcal{A}, \mathcal{B}). \tag{1}$$

In a normal execution, we have the following equivalences based on the properties of the

Table 1: Notation

$\mathcal{A}, \mathcal{B}$	The *initiator* and *responder* roles of the protocol.
a, b	Agents (participants) executing roles of the protocol.
G	A mathematical group of known prime order q.
g	A generator of the group G.
sk_{a}	The long-term private key of the agent a, where $sk_{\mathsf{a}} \in \mathbb{Z}_q$.
pk_{a}	The long-term public key of the agent a, where $pk_{\mathsf{a}} = g^{sk_{\mathsf{a}}}$.
H_1, H_2	Hash functions, where $H_1 : \{0,1\}^* \to \mathbb{Z}_q$ and $H_2 : \{0,1\}^* \to \{0,1\}^\lambda$ (for some constant λ).
$esk_{\mathsf{a}}, esk'_{\mathsf{a}}$	Two different ephemeral keys of the agent a, generated in different sessions.
$\circ$	Written in place of a (bigger) term that is not relevant for the explanation at that point.
λ	A constant.
$x \xleftarrow{\$} S$	The variable x is drawn uniformly from the set S.
$x \leftarrow e$	The variable x is assigned the result of the expression e.

modular exponentiation:

$$\begin{aligned}
X^{sk_{\mathcal{B}}} &= g^{H_1(esk_{\mathcal{A}}, sk_{\mathcal{A}})sk_{\mathcal{B}}} &&= pk_{\mathcal{B}}^{H_1(esk_{\mathcal{A}}, sk_{\mathcal{A}})} && (2)\\
Y^{sk_{\mathcal{A}}} &= g^{H_1(esk_{\mathcal{B}}, sk_{\mathcal{B}})sk_{\mathcal{A}}} &&= pk_{\mathcal{A}}^{H_1(esk_{\mathcal{B}}, sk_{\mathcal{B}})} && (3)\\
Y^{H_1(esk_{\mathcal{A}}, sk_{\mathcal{A}})} &= g^{H_1(esk_{\mathcal{B}}, sk_{\mathcal{B}})H_1(esk_{\mathcal{A}}, sk_{\mathcal{A}})} &&= X^{H_1(esk_{\mathcal{B}}, sk_{\mathcal{B}})} && (4)
\end{aligned}$$

3 Attacking NAXOS using Session-state Reveal

3.1 Security model

We use a slightly modified security model from the one defined in [LLM07]. The only change is that we replace the Ephemeral Key Reveal query by the Session-state Reveal query throughout the security definition.

Regarding the information revealed by a Session-state Reveal query, we assume the following. If an agent computes a hash function $h(x_1, \ldots, x_n)$ at some point in the protocol, the local state directly before the computation of the hash includes all x_i $(0 < i \leq n)$ except for any long-term keys. As a result, performing a Session-state Reveal query at the start of the computation of $h(x_1, \ldots, x_n)$ reveals all x_i, excluding any long-term keys.

We show two attacks: One using test queries on oracles of the initiator type $\mathcal{A}$ and one using the responder type $\mathcal{B}$.

3.2 Attacking the initiator

In Figure 2, we show an attack for a test query on an initiator oracle of NAXOS. The attack requires an active adversary, that can reveal the local state of an agent.

The adversary can compute K_{a} on the basis of the revealed information (based on the

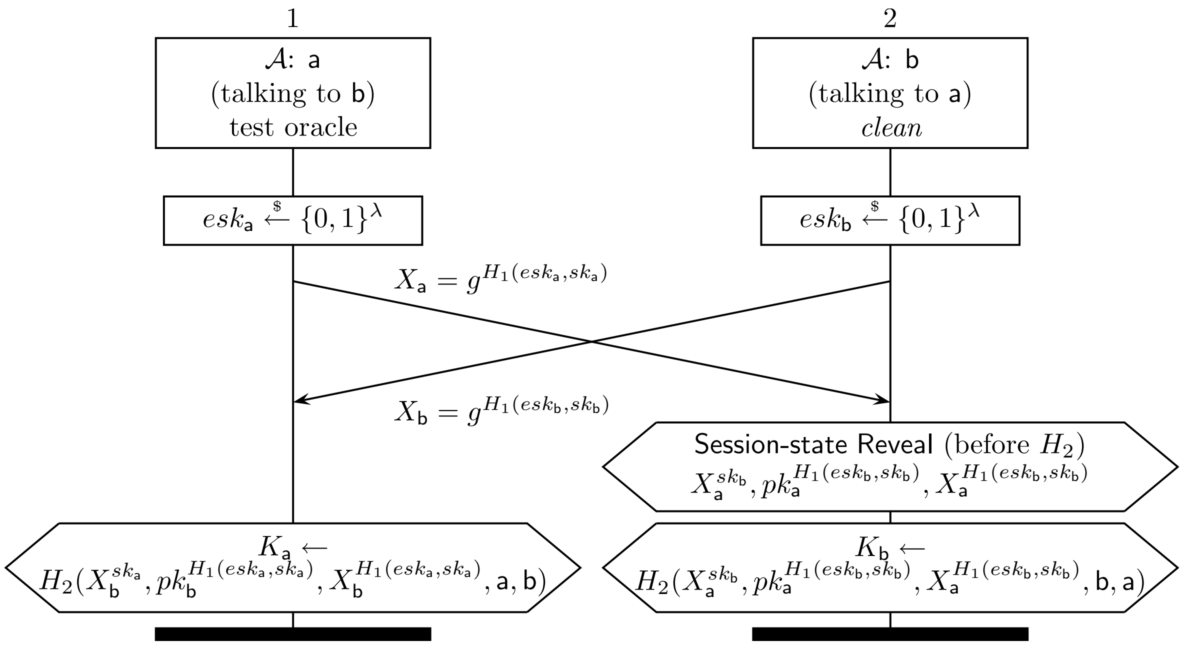

1
2
$\mathcal{A}$: a
(talking to b)
test oracle
$\mathcal{A}$: b
(talking to a)
clean
$esk_{\mathsf{a}} \xleftarrow{\$} \{0,1\}^{\lambda}$
$esk_{\mathsf{b}} \xleftarrow{\$} \{0,1\}^{\lambda}$
$X_{\mathsf{a}} = g^{H_1(esk_{\mathsf{a}},sk_{\mathsf{a}})}$
$X_{\mathsf{b}} = g^{H_1(esk_{\mathsf{b}},sk_{\mathsf{b}})}$
Session-state Reveal (before H_2)
$X_{\mathsf{a}}^{sk_{\mathsf{b}}}, pk_{\mathsf{a}}^{H_1(esk_{\mathsf{b}},sk_{\mathsf{b}})}, X_{\mathsf{a}}^{H_1(esk_{\mathsf{b}},sk_{\mathsf{b}})}$
$K_{\mathsf{a}} \leftarrow H_2(X_{\mathsf{b}}^{sk_{\mathsf{a}}}, pk_{\mathsf{b}}^{H_1(esk_{\mathsf{a}},sk_{\mathsf{a}})}, X_{\mathsf{b}}^{H_1(esk_{\mathsf{a}},sk_{\mathsf{a}})}, \mathsf{a}, \mathsf{b})$
$K_{\mathsf{b}} \leftarrow H_2(X_{\mathsf{a}}^{sk_{\mathsf{b}}}, pk_{\mathsf{a}}^{H_1(esk_{\mathsf{b}},sk_{\mathsf{b}})}, X_{\mathsf{a}}^{H_1(esk_{\mathsf{b}},sk_{\mathsf{b}})}, \mathsf{b}, \mathsf{a})$

Figure 2: Attacking an initiator oracle. Note that $K_{\mathsf{a}} \neq K_{\mathsf{b}}$. The adversary can compute K_{a} after compromising the local state of b.

algebraic properties of the group exponentiation, which are required for the core of the protocol).

The attack proceeds as follows.

1. a starts an initiator instance, wanting to communicate with b.
2. a chooses her ephemeral key esk_{a}, and sends out $X_{\mathsf{a}} = g^{H_1(esk_{\mathsf{a}},sk_{\mathsf{a}})}$. The adversary learns this message.
3. b also starts an initiator instance, wanting to communicate with a.
4. b chooses her ephemeral key esk_{b}, and sends out $X_{\mathsf{b}} = g^{H_1(esk_{\mathsf{b}},sk_{\mathsf{b}})}$. The adversary learns this message.
5. The adversary sends the message X_{b} to a.
6. a computes the session key
$$K_{\mathsf{a}} = H_2(X_{\mathsf{b}}^{sk_{\mathsf{a}}}, pk_{\mathsf{b}}^{H_1(esk_{\mathsf{a}},sk_{\mathsf{a}})}, X_{\mathsf{b}}^{H_1(esk_{\mathsf{a}},sk_{\mathsf{a}})}, \mathsf{a}, \mathsf{b}). \tag{5}$$
7. The adversary sends the message X_{a} to b.
8. b computes the session key
$$K_{\mathsf{b}} = H_2(X_{\mathsf{a}}^{sk_{\mathsf{b}}}, pk_{\mathsf{a}}^{H_1(esk_{\mathsf{b}},sk_{\mathsf{b}})}, X_{\mathsf{a}}^{H_1(esk_{\mathsf{b}},sk_{\mathsf{b}})}, \mathsf{b}, \mathsf{a}). \tag{6}$$

   During the computation of K_{b}, the adversary uses Session-state Reveal to learn the input to H_2. In particular, the adversary learns $X_{\mathsf{a}}^{sk_{\mathsf{b}}}$, $pk_{\mathsf{a}}^{H_1(esk_{\mathsf{b}},sk_{\mathsf{b}})}$, and $X_{\mathsf{a}}^{H_1(esk_{\mathsf{b}},sk_{\mathsf{b}})}$.

9. The adversary now knows

$$pk_{\mathsf{a}}^{H_1(esk_{\mathsf{b}},sk_{\mathsf{b}})} = g^{sk_{\mathsf{a}}H_1(esk_{\mathsf{b}},sk_{\mathsf{b}})} = X_{\mathsf{b}}^{sk_{\mathsf{a}}}, \tag{7}$$

$$X_{\mathsf{a}}^{sk_{\mathsf{b}}} = g^{H_1(esk_{\mathsf{a}},sk_{\mathsf{a}})sk_{\mathsf{b}}} = pk_{\mathsf{b}}^{H_1(esk_{\mathsf{a}},sk_{\mathsf{a}})}, \tag{8}$$

$$X_{\mathsf{a}}^{H_1(esk_{\mathsf{b}},sk_{\mathsf{b}})} = g^{H_1(esk_{\mathsf{a}},sk_{\mathsf{a}})H_1(esk_{\mathsf{b}},sk_{\mathsf{b}})} = X_{\mathsf{b}}^{H_1(esk_{\mathsf{a}},sk_{\mathsf{a}})}. \tag{9}$$

The three terms on the right-hand side are the first three components of the session key K_{a} from Formula 5.

10. The adversary combines the elements with the names a and b, and applies H_2, resulting in K_{a}.

The above sequence of actions forms an attack on the protocol, because the adversary can learn the session key of the initiator a by revealing the local state of the second oracle. This second oracle is *clean* according to the security definition in [LLM07]. In particular, in terms of [LLM07, p.8-9], the second oracle is clean because (a) neither a nor b are adversary-controlled, (b) no Session-key Reveal queries are performed by the adversary, (c) the second oracle is not a partner to the test oracle (in [LLM07] partnering is defined in terms of matching sessions), and (d) no long-term keys are revealed. Therefore, the attack violates security in the adapted eCK model in which Session-state Reveal is allowed.

Some further observations regarding this attack:

- The oracles compute different session keys: $K_{\mathsf{a}} \neq K_{\mathsf{b}}$, because the order of the participant names a, b is reversed.
- The adversary does not need to learn any ephemeral key for this attack.
- Even in other existing interpretations of the partner function (or freshness) from literature (external session identifiers, explicit session identifiers, etc.) the two oracles are not partners. Consequently, it seems that the NAXOS protocol is therefore also not secure in other models that allow Session-state Reveal, such as the CK model [CK01] (under the assumption that Session-state Reveal is interpreted to include non-long term x_i before computation of $h(x_1, \ldots, x_n)$).

3.3 Attacking the responder

Second, we show an attack for a test query on a responder oracle in Figure 3. It seems this attack is more easily exploited than the previous one.

The attack proceeds as follows.

1. The adversary chooses an arbitrary bit string κ.
2. The adversary computes g^{κ} and sends the result to a responder instance of a, with sender address b.
3. a receives the message and assigns $X_{\mathsf{b}} = g^{\kappa}$.
4. a chooses her ephemeral key esk_{a}, and sends out $X_{\mathsf{a}} = g^{H_1(esk_{\mathsf{a}},sk_{\mathsf{a}})}$. The adversary learns this message.

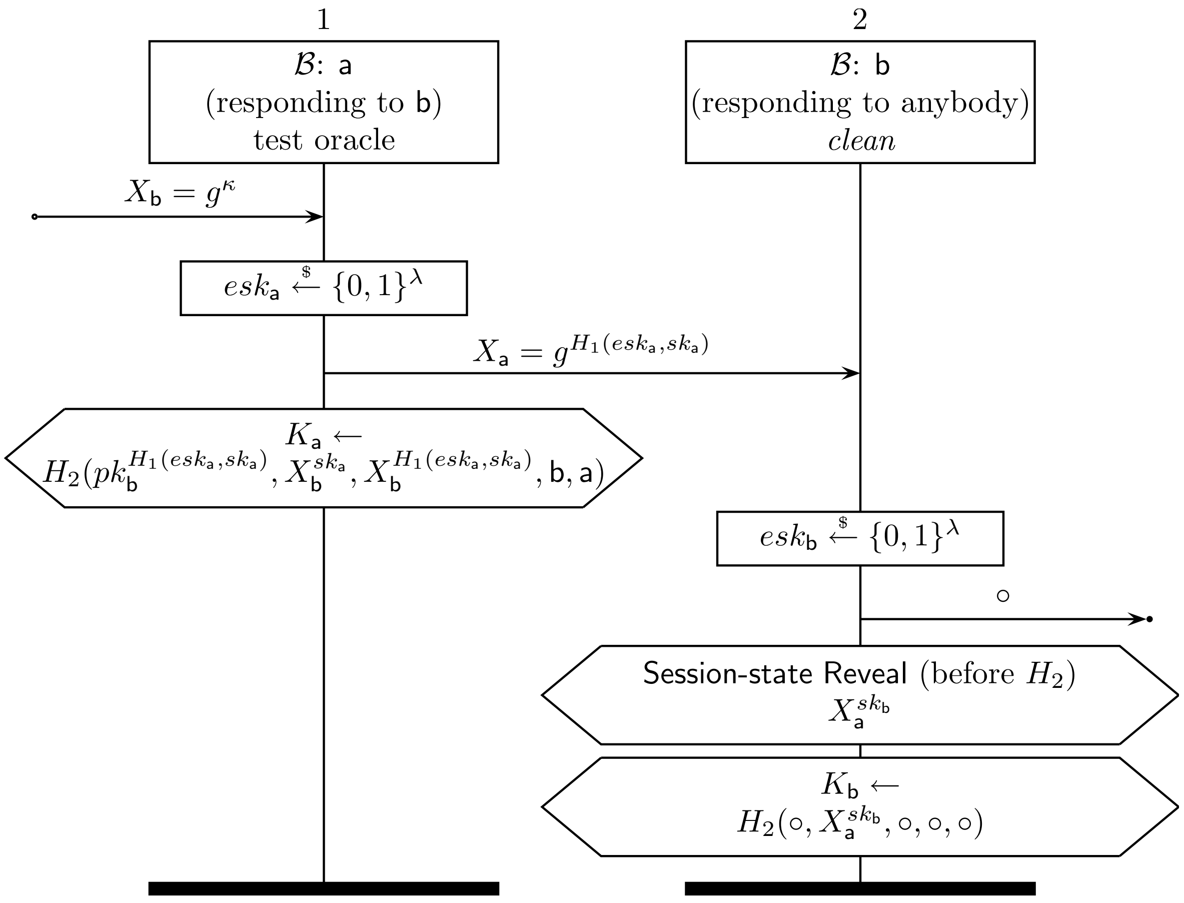

1
2
$\mathcal{B}$: a
(responding to b)
test oracle
$\mathcal{B}$: b
(responding to anybody)
clean
$X_\mathsf{b} = g^\kappa$
$esk_\mathsf{a} \overset{\$}{\leftarrow} \{0,1\}^\lambda$
$X_\mathsf{a} = g^{H_1(esk_\mathsf{a}, sk_\mathsf{a})}$
$K_\mathsf{a} \leftarrow H_2(pk_\mathsf{b}^{H_1(esk_\mathsf{a}, sk_\mathsf{a})}, X_\mathsf{b}^{sk_\mathsf{a}}, X_\mathsf{b}^{H_1(esk_\mathsf{a}, sk_\mathsf{a})}, \mathsf{b}, \mathsf{a})$
$esk_\mathsf{b} \overset{\$}{\leftarrow} \{0,1\}^\lambda$
$\circ$
Session-state Reveal (before H_2) $X_\mathsf{a}^{sk_\mathsf{b}}$
$K_\mathsf{b} \leftarrow H_2(\circ, X_\mathsf{a}^{sk_\mathsf{b}}, \circ, \circ, \circ)$

Figure 3: Attack on a responder oracle. We have $K_\mathsf{a} \neq K_\mathsf{b}$. The adversary can compute (and even contribute to) K_a after revealing the local state of b.

5. a computes the session key

$$K_\mathsf{a} = H_2(pk_\mathsf{b}^{H_1(esk_\mathsf{a}, sk_\mathsf{a})}, X_\mathsf{b}^{sk_\mathsf{a}}, X_\mathsf{b}^{H_1(esk_\mathsf{a}, sk_\mathsf{a})}, \mathsf{b}, \mathsf{a}) \tag{10}$$

   which is equal to

$$H_2(pk_\mathsf{b}^{H_1(esk_\mathsf{a}, sk_\mathsf{a})}, g^{\kappa\ sk_\mathsf{a}}, g^{\kappa\ H_1(esk_\mathsf{a}, sk_\mathsf{a})}, \mathsf{b}, \mathsf{a}). \tag{11}$$

6. The adversary redirects X_a to a responder instance of b. The adversary can insert an arbitrary participant name in the sender field of the message, which b takes to be the origin of the message.

7. b computes his ephemeral secret, combines it with his long term key, and sends out the corresponding message.

8. b computes his session key K_b (which differs from K_a). Before applying H_2, b computes the second component $X_\mathsf{a}^{sk_\mathsf{b}}$.

9. The adversary uses Session-state Reveal on the session of b directly before the application of H_2 to learn $X_\mathsf{a}^{sk_\mathsf{b}}$.

10. The adversary knows κ, X_{a}, and $X_{\mathsf{a}}^{sk_{\mathsf{b}}}$. Furthermore, as the public keys are by definition public, the adversary also knows pk_{a}. Hence the adversary also knows, or can compute:

$$\begin{aligned} X_{\mathsf{a}}^{sk_{\mathsf{b}}} &= g^{H_1(esk_{\mathsf{a}},sk_{\mathsf{a}})sk_{\mathsf{b}}} &&= pk_{\mathsf{b}}^{H_1(esk_{\mathsf{a}},sk_{\mathsf{a}})}, \qquad (12) \\ (pk_{\mathsf{a}})^{\kappa} &= g^{sk_{\mathsf{a}}\kappa} &&= X_{\mathsf{b}}^{sk_{\mathsf{a}}}, \qquad (13) \\ (X_{\mathsf{a}})^{\kappa} &= g^{H_1(esk_{\mathsf{a}},sk_{\mathsf{a}})\kappa} &&= X_{\mathsf{b}}^{H_1(esk_{\mathsf{a}},sk_{\mathsf{a}})}. \qquad (14) \end{aligned}$$

The three terms on the right-hand side are the first three components of the session key K_{a} from Formula 10.

11. The adversary combines the elements and applies H_2, resulting in K_{a}.

This sequence forms an attack on the protocol, because the adversary can use data revealed from a clean oracle (as defined in [LLM07]) in order to compute the session key of the test oracle. In practical terms, this attack even allows the adversary to determine a part of the session key of a.

For this attack there are also some observations to be made:

- The responder oracle of b is not a partner to the oracle of a in terms of matching sessions. Also, in other partner existing interpretations from literature (external session identifiers, explicit session identifiers, etc.) they would also not match.
- The adversary chooses κ, and can therefore influence the session key.
- In this attack, the adversary does not need to learn any long term private keys or ephemeral keys.
- The attack seems to be also valid in the CK model: the oracles are not partners for a number of reasons, for example because their choice of agents differs. Oracle 1 has $\{\mathsf{a}, \mathsf{b}\}$ and oracle 2 has $\{\mathsf{b}, z\}$ where z is an arbitrary participant. Hence the adversary can choose $z \neq \mathsf{a}$.
- Based on this attack, one can construct an alternative attack on two agents performing matching sessions. Given a regular protocol run, and hence matching sessions, by initiator b and responder a, an adversary can perform a Session-state Reveal on b to reveal $H_1(esk_{\mathsf{b}}, sk_{\mathsf{b}})$. If the adversary then chooses $\kappa = H_1(esk_{\mathsf{b}}, sk_{\mathsf{b}})$, a second instance of b (as responder) can be exploited along the lines of Figure 3 to reconstruct the session key or the regular protocol run.

4 Conclusion

In common definitions of AKE security the Session-state Reveal query is underspecified. In many cases the definition is only made explicit in particular protocol proofs. This approach turns the exact definition of Session-state Reveal into a parameter of the exact security provided by the protocol. As a result, stating that two protocols are "AKE secure" does not mean they meet exactly the same property.

In [LLM07,LLM06] the Session-state Reveal query is replaced by the Ephemeral Key Reveal query, which is claimed to be at least as strong as Session-state Reveal. Thus, the notion of Session-state Reveal is reduced to Ephemeral Key Reveal. Reducing Session-state Reveal to

Ephemeral Key Reveal simplifies proofs significantly: one does not need to define what exactly is part of the ephemeral key, but one only needs to prove that no information about the ephemeral key is revealed [LLM07, Ust08, Oka07]. However, the validity of this reduction has not been proven.

The validity of the reduction is informally argued in [LLM06], and similar arguments can be found in other works that use the eCK model [Ust08, Oka07], e. g. in [Ust08, p.333]: "In general, by specifying that the session specific private information (the session state) is part of the ephemeral private key, the Session-state Reveal and Ephemeral Key Reveal queries can be made functionally equivalent".

In this paper we have shown that the reduction is invalid, that is, a security model with Ephemeral Key Reveal is not as strong as a model with Session-state Reveal. The attacks presented here on the NAXOS protocol, which was proven correct for Ephemeral Key Reveal in [LLM07], strictly depend on the use of the Session-state Reveal query.

The attacks presented here fall just outside the eCK security model, and they therefore do not indicate a problem with the proofs in [LLM07]. Instead, what the attacks indicate is that the eCK security model, and similarly the property that is proved correct, is not as strong as suggested in e. g. [LLM07]. Furthermore, the attacks seem also to be valid in our interpretation of the CK model, which shows that contrary to the statement in [BCNP08], the difference between CK and eCK is in fact meaningful in reality. In particular, we have shown that one can prove real protocols secure in eCK which are not secure in CK, and are vulnerable to attacks where the local state is revealed.

The idea behind the NAXOS protocol is appealing: by strongly connecting the long- and short-term information, the adversary would be required to know both elements to perform an attack. However, in order to use the combination of these elements securely in the protocol, in particular for transmission, there are further computations needed. The way in which the ephemeral key (and hence Ephemeral Key Reveal) is defined and used in proofs, excludes the intermediate products of such subsequent computations. This is the ultimate problem with the reduction from Session-state Reveal to Ephemeral Key Reveal, as was already noted in [BCNP08]. Precisely this difference is exploited by the attacks shown here.

The question remains whether it is possible to adapt NAXOS to satisfy a security model similar to eCK that allows for Session-state Reveal queries.

References

[BCNP08] C. Boyd, Y. Cliff, J.G. Nieto, and K.G. Paterson. Efficient one-round key exchange in the standard model. In *ACISP*, volume 5107 of *Lecture Notes in Computer Science*, pages 69–83. Springer, 2008.

[BCPQ01] E. Bresson, O. Chevassut, D. Pointcheval, and J. Quisquater. Provably authenticated group diffie-hellman key exchange. In *CCS '01: Proceedings of the 8th ACM conference on Computer and Communications Security*, pages 255–264, New York, USA, 2001. ACM.

[BPR00] Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks. In *EUROCRYPT*, Lecture Notes in Computer Science, pages 139–155. Springer, 2000.

[CBH05] K-K.R. 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.

[CK01] R. Canetti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In *EUROCRYPT*, volume 2045 of *LNCS*, pages 453–474. Springer, 2001.

[LLM06] Brian LaMacchia, Kristin Lauter, and Anton Mityagin. Stronger security of authenticated key exchange. Cryptology ePrint Archive, Report 2006/073, 2006. `http://eprint.iacr.org/`.

[LLM07] B.A. 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.

[MU08] A. Menezes and B. Ustaoglu. 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.

[Oka07] 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.

[Ust08] B. Ustaoglu. Obtaining a secure and efficient key agreement protocol from (H)MQV and NAXOS. *Des. Codes Cryptography*, 46(3):329–342, 2008.