Stronger Security of Authenticated Key Exchange

Brian LaMacchia[1], Kristin Lauter[2], and Anton Mityagin[3]

[1] Microsoft Corporation, 1 Microsoft Way, Redmond, WA
bal@microsoft.com
[2] Microsoft Research, 1 Microsoft Way, Redmond, WA
klauter@microsoft.com
[3] Microsoft Live Labs, 1 Microsoft Way, Redmond, WA
mityagin@microsoft.com

Abstract. Recent work by Krawczyk [12] and Menezes [16] has highlighted the importance of understanding well the guarantees and limitations of formal security models when using them to prove the security of protocols. In this paper we focus on security models for authenticated key exchange (AKE) protocols. We observe that there are several classes of attacks on AKE protocols that lie outside the scope of the Canetti-Krawczyk model. Some of these additional attacks have already been considered by Krawczyk [12]. In an attempt to bring these attacks within the scope of the security model we extend the Canetti-Krawczyk model for AKE security by providing significantly greater powers to the adversary. Our contribution is a more compact, integrated, and comprehensive formulation of the security model. We then introduce a new AKE protocol called NAXOS and prove that it is secure against these stronger adversaries.

1 Introduction

In this paper we extend the Canetti-Krawczyk [11,12] security model for authenticated key exchange (AKE) to capture attacks resulting from leakage of ephemeral and long-term secret keys. Our security model for authenticated key exchange is defined in the spirit of Bellare and Rogaway [3] and Canetti and Krawczyk [11] by an experiment in which the adversary is given many corruption powers for various key exchange sessions and must solve a challenge on a test session. We extend adversarial capabilities to the following extent: the only corruption powers we do not give an adversary in the experiment are those that would trivially break an AKE protocol. We also define a new AKE protocol which is secure in our new model.

More specifically, in an authenticated key exchange protocol, two parties exchange information and compute a secret key as a function of at least four pieces of secret information: their own long-term (static) and ephemeral secret keys and the other party's long-term and ephemeral secret keys. Of the four pieces of

W. Susilo, J.K. Liu, and Y. Mu. (Eds.): ProvSec 2007, LNCS 4784, pp. 1–16, 2007.

information, we allow an adversary to *reveal*[1] any subset of the four which does not contain both the long-term and ephemeral secrets of one of the parties. To explain this more precisely, we divide AKE test sessions (sessions which are subject to attack by an adversary) into two types. In sessions of the first type ("passive" sessions), the adversary does not cancel or modify communications between the two parties. In sessions of the second type ("active" sessions), the adversary may forge the communication of the second party. Another way to phrase the distinction, as done by Krawczyk in the analysis of the HMQV protocol [12], is whether the adversary actively intervenes in the key exchange session or is a passive eavesdropper.

In addition to distinguishing between passive and active sessions, we identify which pieces of secret information the adversary can reveal without being able to trivially break the AKE protocol (compute the session key for any AKE protocol). In both types of sessions, if an adversary can reveal the long-term and the ephemeral secret keys of one of the parties in the session, then the adversary can trivially compute a session key as it has all the secret information of one of the legitimate parties in the session.

For passive sessions, an adversary may reveal both ephemeral secret keys, both long-term secret keys, or one of each from the two different parties without trivially breaking the protocol. Thus security in our model implies weak Perfect Forward Secrecy, defined by Krawczyk to be security against revelation of long-term secret keys after the session is completed (without active adversarial intervention in the session establishment).

For active sessions, the adversary may forge communications from one of the parties. Thus, if the adversary can also reveal the long-term secret key of that same party, then the adversary can trivially compute the session key. The same argument was used by Krawczyk to show that no 2-round AKE protocol can achieve full perfect forward secrecy (PFS). Still, an adversary can reveal a long-term secret key or ephemeral secret key of the other party without trivially breaking the session. So for another example, our extension to the Canetti-Krawczyk model also implies security against Key Compromise Impersonation (KCI) attacks, where the adversary first reveals a long-term secret of a party and then impersonates others to this party.

Considering attacks involving both types of sessions, it is natural to define a single security model which captures all of them. In our model, in passive test sessions we allow the adversary to reveal any subset of the four pieces of secret information which does not contain both the long-term and ephemeral secrets of one of the parties. In active test sessions, we allow the adversary to reveal only the long-term secret or the ephemeral secret key of the party which is executing the test session. In our security experiment, a test session is still considered *clean* even if the adversary has revealed any of the allowable combinations of secret keys of the two parties.

[1] We say that an adversary "reveals" a piece of secret information when that adversary chooses to learn the value of that information by performing the corresponding key reveal query as defined in Section 3.2.

Security in this extended Canetti-Krawczyk model also implies security against a number of other attacks not covered by the Canetti-Krawczyk model (see Section 2.2). In a sense, our model is just an extension of an instance of the Canetti-Krawczyk model, since we define the session state of a party to be the ephemeral secret key. On the other hand, *some* instance of the Canetti-Krawczyk model must be chosen when considering the security of any protocol, since the definition of the session-state reveal query must be specified, and our model is stronger than a model which does not include the ephemeral secret key as part of the session state for the session state reveal query. In addition, the Canetti-Krawczyk model does not allow the adversary to attack sessions against which a session state reveal query has been made. They consider such sessions broken, while our definition covers the security of these partially corrupted sessions. Krawczyk does extend the model in [12], but still some attacks are not covered because those sessions are not considered clean. Our model extends the notion of a clean session further, giving the adversary more power to reveal long-term and ephemeral secret keys. Our motivation to include revelations of ephemeral secret keys in the model comes from "practical" (i.e. engineering) considerations and scenarios such as active adversarial attacks or compromise of the random number generator (RNG) used by one of the parties.

We stress that our extension of the security model allows the adversary to register arbitrary public keys for adversary-controlled parties without any checks such as proof-of-possession done by the certificate authority. In contrast, some of the protocols in the literature [13,14] were proved secure assuming that the key registration is done honestly. Namely, that initially a trusted party generates keys for all, even adversary-controlled parties.

Finally, we present a new AKE protocol, called NAXOS, which provably meets our definition of AKE security. We prove the security of NAXOS under the standard Gap Diffie-Hellman assumption. We also improve the concrete security of NAXOS under the related Pairing Diffie-Hellman assumption. A version of the NAXOS protocol with key confirmation is also possible.

In Figure 1 we compare the efficiency and security of NAXOS with four other recent authenticated key exchange protocols: HMQV, KEA+ [15], protocol $\mathcal{TS}3$ by Jeong, Katz and Lee [13] and Kudla-Paterson [14][2]. The second column in the table, "Efficiency," lists the relative efficiency of the protocol as measured by the number of exponentiations executed by one party. (Communication costs in all of these protocols, except for Jeong-Katz-Lee, is the same as in the original Diffie-Hellman protocol.) Column 3, "Key Registration," specifies whether adversary-controlled parties can register arbitrary public keys or if honest key-registration is assumed. The fourth column, labeled "Ephemeral," indicates whether an adversary is allowed to reveal ephemeral secret information of the parties. Column 5 lists

[2] Kudla and Paterson [14] define partnership via matching session identifiers (computed by the parties), although for their protocol this appears to be equivalent to matching conversations.

Protocol	Effic.	Key Reg.	Ephemeral	Security	Assumptions
NAXOS	4	Arbitrary	yes	Extended CK	GDH (or PDH) + RO
HMQV	2.5	Arbitrary	yes	CK + wPFS + KCI	GDH + KEA1 + RO
KEA+	3	Arbitrary	yes	CK + wPFS + KCI	GDH (or PDH) + RO
Jeong-Katz-Lee	3	Honest	no	BR + wPFS	DDH + secure MACs
Kudla-Paterson	3	Honest	no	BR + KCI	GDH + RO

Fig. 1. Comparison of recent AKE protocols

the security model for each protocol[3]. Finally, the sixth column ("Assumptions") lists the security assumptions upon which each protocol depends[4]. We refer the reader to Chapter 7 of [6] for a good overview of Diffie-Hellman assumptions.

We begin with a brief review in Section 2 of the Canetti-Krawczyk security model and discuss some attacks not covered by their definition in Section 2.2. We introduce our extension of the Canetti-Krawczyk security model in Section 3. In Section 4 we describe the NAXOS protocol and prove its security in the extended model.

2 Previous Models

2.1 Overview of the Canetti-Krawczyk Model

The Canetti-Krawczyk security model is among a family of security models for authenticated key exchange that includes those of Bellare and Rogaway [3,5] and Bellare, Pointcheval and Rogaway [2]. We refer the reader to Choo et al. [9] for a concise summary of the differences among these various models. We give a high-level overview of the Canetti-Krawczyk model and introduce some notation which will be useful later in the paper. We remark that the model we describe differs from the original definition in that we use session identifiers defined via matching conversations. The same definition was used by Krawczyk when analyzing the security of the HMQV protocol [12] and it is now a commonly used variant of the Canetti-Krawczyk model.

The AKE security experiment involves multiple honest parties and an adversary $\mathcal{M}$ connected via an unauthenticated network. The adversary selects parties to execute key-exchange sessions and selects an order in which the sessions will be executed. Actions the adversary is allowed to perform include taking full

[3] CK denotes Canetti-Krawczyk security without perfect forward secrecy, assuming that partnership is defined via matching conversations. BR denotes the Bellare-Rogaway model [3], which appears to be equivalent to the Canetti-Krawczyk model with no ephemeral reveals allowed and key-registration done honestly [9]. KCI denotes security against key-compromise impersonation. wPFS denotes weak perfect forward secrecy. Extended CK denotes our extension of the Canetti-Krawczyk model.

[4] RO – random oracle model [4], DDH – Decisional Diffie-Hellman, GDH – Gap Diffie-Hellman [17], PDH – Pairing Diffie-Hellman [15] and KEA1 – knowledge of exponent assumption [1].

control of any party (a Corrupt query), revealing the session key of any session (a Reveal query), or revealing session-specific secret information of any session (a Session-State Reveal query).

We stress that an AKE session is executed by a single party: since all communication is controlled by an adversary, a party executing a session cannot know for sure with whom it is communicating. The party executing the session is called the *owner* of the session and the other party is called the *peer*. The *matching session* to an AKE session (by the owner with the peer) is the corresponding AKE session which is supposed to be executed by the peer with the owner. The matching session might not exist if the communications were modified by the adversary. The *session identifier* of an AKE session consists of the parties' identities concatenated with messages they exchanged in the session[5]. In [12], a completed session is definied to be "clean" if the session as well as its matching session (if it exists) is not corrupted (neither session key nor session state were revealed by $\mathcal{M}$) and if none of the participating parties were corrupted.

At some point in the experiment, the adversary is allowed to make one Test query: it can select any clean completed session (called the *test session*) and it is given a challenge which consists either of the session key for that session or a randomly selected string. The adversary's goal is to guess correctly which of the cases was selected.

Additionally, the Canetti-Krawczyk [11] definition has an optional perfect forward secrecy (PFS) requirement. In the variant of Canetti-Krawczyk security with PFS, the adversary is allowed to corrupt a participant of the test session (either owner or peer) after the test session is completed. As noted by Krawczyk [12], the PFS requirement is not relevant for 2-round AKE protocols since no 2-round protocol can achieve PFS. Krawczyk introduced the notion of *weak perfect forward secrecy* (wPFS) which can be achieved by 2-round protocols and which he demonstrated is achieved by HMQV [12]. Weak PFS guarantees perfect forward secrecy only for those AKE sessions where the adversary didn't modify communications between the parties. (Using the above terminology, the matching session exists for the test session and both test and matching sessions are clean.)

2.2 Attacks Not Covered by the Existing Definitions

We point out several attacks which are not captured by the previous definitions and explain which components of the Canetti-Krawczyk model prohibit these attacks from being considered. First, we observe that although the adversary is allowed to reveal the session state of the parties, he is not allowed to make Session-State Reveal queries against the session he wants to attack (the test session). That is, existing security models do not provide any security guarantees for a session if the ephemeral secret key of either party has been leaked. While Krawczyk ([12]) extends the Canetti-Krawczyk model by making a definition of clean session that allows him to consider resistance to Key Compromise

[5] We remark that for protocols, where participants do not have full view of the messages exchanged (for example, see [10]), it might not be possible to define such session identifiers.

Impersonation (KCI) attacks and achieve weak Perfect Forward Secrecy (wPFS), this extension still does not include attacks such as revelation of both ephemeral secret keys or both long-term secret keys. Krawczyk does consider resistance to revelation of both ephemeral secret keys separately, and proves HMQV secure against this attack under the stronger assumptions of GDH and KEA1.

Second, when the adversary corrupts an honest party, he takes full control over this party and reveals all its secret information. This definition of the Corrupt query does not allow attacks where the adversary reveals a long-term secret key of some party prior to the time when that party executes the test session. Here we summarize some attacks which are not allowed by the Canetti-Krawczyk model but are permitted under our new definition:

- Key-compromise impersonation (KCI) attack [7,12]: the adversary reveals a long-term secret key of a party and then impersonates others to this party.
- An adversary reveals the ephemeral secret key of a party and impersonates others to this party.
- Two honest parties execute matching sessions, and the adversary reveals the ephemeral secret keys of both of the parties and tries to learn the session key.
- Two honest parties execute matching sessions. The adversary reveals the ephemeral secret key of one party, the long-term secret key of the other party and tries to learn the session key
- Two honest parties execute matching sessions. The adversary reveals the long-term keys of both of the parties prior to the execution of the session and tries to learn the session key.

3 Definitions

3.1 Motivation for Our Security Definition

We modify the Canetti-Krawczyk model in the definition of adversarial power and in the notion of cleanness of the test session. Specifically, we replace the Session-State Reveal query with an "Ephemeral Key Reveal" query which reveals the ephemeral secret key of the party. Additionally, we give the adversary the power to reveal a long-term secret key, by making a Long-Term Key Reveal query, without corrupting the party. We remove the Corrupt query as it is no longer necessary: the adversary can achieve the same result as the Corrupt query by revealing all the secret information of the party through Long-Term Key Reveal, Ephemeral Key Reveal and Reveal queries and by computing everything on behalf of that party. We also modify the definition of a "clean session" by allowing the adversary to reveal the maximum possible amount of data. We disallow only those corruptions which allow the adversary to trivially break any AKE protocol.

We classify the test sessions as either "passive" or "active" depending on whether the adversary is able to cancel or modify the information sent between two honest participants. Formally, passive sessions are those where the matching

session was completed at some point in the experiment, and active sessions are those where no matching session was completed at any time in the experiment.

For passive sessions we allow the adversary to reveal any subset of the four secret keys (each party's ephemeral and long-term secret keys) which does not contain both the ephemeral and long-term secret keys of a single party. Note that the knowledge of both the ephemeral and long-term keys of one of the parties allows the adversary to compute the session key for any AKE protocol.

For active sessions the communication sent by the peer might be corrupted and thus we cannot define the ephemeral key of the peer. In this case we only allow the adversary to reveal either the ephemeral or the long-term secret key of the owner, as revealing both keys would trivially compromise the protocol. Note that we cannot allow the adversary to reveal the long-term secret of the peer (even after the test session is completed), since Krawczyk [12] shows that in this case one can break any AKE protocol. (This is the same attack which shows the impossibility of the full perfect forward secrecy requirement.)

3.2 Security Experiment for Extended Canetti-Krawczyk

Assume that the identities of the parties are binary strings (they can be derived from the actual names of the parties). We will use letters $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, ..., both for referring to the parties and for their identities. The adversary is given the power to select each party's identity (the binary string) if it so chooses.

There are a number of honest parties which are connected to the certificate authority, $\mathcal{CA}$, and to the adversary, $\mathcal{M}$. That is, the communication between the parties is fully controlled by $\mathcal{M}$ (and $\mathcal{M}$ cannot interfere with communication between a single party and the $\mathcal{CA}$). $\mathcal{M}$ is also connected to the certificate authority and can register fictitious parties. The adversary plays a central role in the experiment and is responsible for activating all other parties.

We call a particular instantiation of an AKE protocol executed by one of the parties an *AKE session*. Since all communication is controlled by the adversary, a party can never know if the second party actually exists and if the communication it receives was computed by an honest party or by the adversary. Legitimate execution of an AKE protocol by two parties $\mathcal{A}$ and $\mathcal{B}$ consists of two AKE sessions, matching sessions executed by $\mathcal{A}$ and by $\mathcal{B}$ respectively. Note that an instantiation of the AKE protocol is different depending on whether the executor is the owner or the peer.

We do not assume the existence of explicit session identifiers. Instead, we define a session identifier to consist of the identities of the 2 participants and the information they exchanged. Specifically, a session identifier

$$sid = (role, \mathrm{ID}, \mathrm{ID}^*, \mathrm{comm}_1, \ldots, \mathrm{comm}_\mathrm{n}),$$

where $\mathrm{ID} \in \{0,1\}^*$ is the identity of the party executing the session, $role \in \{O, P\}$ is its role (owner/peer) in the protocol, ID^* is the identity of the other party and $comm_i \in \{0,1\}^*$ is the i-th communication sent by the parties. As in the Canetti-Krawczyk model, we define the *matching session* to an AKE session to be the session executed by the other party with the same communications

being transmitted, albeit in different order. For example, in a 2-round protocol, if $\mathcal{A}$ executes the session $(O, \mathcal{A}, \mathcal{B}, comm_{\mathcal{A}}, comm_{\mathcal{B}})$, then the matching session is executed by $\mathcal{B}$ and has session identifier $(P, \mathcal{B}, \mathcal{A}, comm_{\mathcal{A}}, comm_{\mathcal{B}})$.

A party computes a communication $comm_i$ as a function of its own ephemeral and long-term secret keys, its partner's public key and previous messages exchanged. Once a party receives all the communications, it computes a session key as a function of its own ephemeral and long-term secret keys, its partner's public key, and all communications, and completes the session.

The experiment proceeds as follows. Initially $\mathcal{M}$ selects the identities of all honest parties (which can be arbitrary distinct binary strings) and honest parties generate and register their public keys with the $\mathcal{CA}$. The adversary can register arbitrary public keys (even the same as those of some honest parties) on behalf of adversary-controlled parties. Then the adversary makes any sequence of the following queries:

- Send$(\mathcal{A}, \mathcal{B}, comm)$. Sends a message $comm$ to $\mathcal{A}$ on behalf of $\mathcal{B}$. Returns $\mathcal{A}$'s response to this message. This query allows $\mathcal{M}$ to order $\mathcal{A}$ to start an AKE session with $\mathcal{B}$ and to provide communications from $\mathcal{B}$ to $\mathcal{A}$.
- Long-Term Key Reveal$(\mathcal{A})$. Reveals a long-term key of a party $\mathcal{A}$.
- Ephemeral Key Reveal(sid). Reveals an ephemeral key of a session sid (possibly incomplete).
- Reveal(sid). Reveals a session key of a completed session sid.

Eventually (at any time in the experiment), $\mathcal{M}$ selects a completed session sid, makes a query Test(sid) and is given a challenge value C. $\mathcal{M}$ continues the experiment after the Test query. The experiment terminates as soon as $\mathcal{M}$ makes the Guess(b') query. The experiment answers the adversary's queries as follows:

- Test(sid) // can be made only once.
  Pick $b \stackrel{\$}{\leftarrow} \{0,1\}$. If $b = 1$, let $C \leftarrow$ Reveal(sid); otherwise pick $C \stackrel{\$}{\leftarrow} \{0,1\}^{\lambda}$.
  Return C.
- Guess(b') // $\mathcal{M}$ terminates after making this query.
  If $b' = b$, return 1, otherwise return 0.

An adversary $\mathcal{M}$ *wins* the experiment if the selected test session is *clean* and if he guesses the challenge correctly (that is, if the Guess query returns 1).

We now define what it means for a test session to be *clean*. Let sid be an AKE session completed by a party $\mathcal{A}$ with some other party $\mathcal{B}$, and denote by sid^* the matching session to sid, supposedly executed by $\mathcal{B}$ (sid^* may not exist in the experiment). Denote by $sk_{\mathcal{A}}$ and $sk_{\mathcal{B}}$ long-term secret keys of $\mathcal{A}$ and $\mathcal{B}$. Denote by $esk_{\mathcal{A}}$ and $esk_{\mathcal{B}}$ ephemeral secret keys generated by $\mathcal{A}$ and $\mathcal{B}$ in sid and sid^* (the latter is defined only if sid^* exists). We say that an AKE session sid is *not clean* if an adversary can trivially compute the session key. That is, a session sid is not clean if any of the following conditions hold:

- $\mathcal{A}$ or $\mathcal{B}$ is an adversary-controlled party. This means in particular that $\mathcal{M}$ chooses or reveals both the long-term and ephemeral secret keys for the party and performs all communications and computations on behalf of the party.

- $\mathcal{M}$ reveals the session key of sid or sid^* (if the latter exists).
- Session sid^* exists and $\mathcal{M}$ reveals either both $sk_{\mathcal{A}}$ and $esk_{\mathcal{A}}$, or both $sk_{\mathcal{B}}$ and $esk_{\mathcal{B}}$.
- Session sid^* doesn't exist and $\mathcal{M}$ reveals either $sk_{\mathcal{B}}$ or both $sk_{\mathcal{A}}$ and $esk_{\mathcal{A}}$.

A session sid is clean if *none* of these conditions hold. We remark that the cleanness of the test session can be identified only after the experiment is completed: the third and fourth conditions above can only be determined in the end of the experiment. That is, the adversary wins the experiment if he correctly guesses the challenge for the test session and this session remains clean until the end of the experiment.

Definition 1 (Extended Canetti-Krawczyk security). *The* advantage *of the adversary $\mathcal{M}$ in the AKE experiment with AKE protocol Π is defined as*

$$\mathbf{Adv}_{\Pi}^{AKE}(\mathcal{M}) \;=\; \Pr[\mathcal{M} \textit{ wins}] - \frac{1}{2}.$$

We say that an AKE protocol is secure (in the extended Canetti-Krawczyk model) if matching sessions compute the same session keys and no e cient adversary $\mathcal{M}$ has more than a negligible advantage in winning the above experiment.

4 NAXOS AKE Protocol

4.1 Assumptions

All the arithmetic in this section is assumed to be in a mathematical group G of known prime order q. We denote by g a generator of G and write the group operation multiplicatively.

The discrete logarithm function $DLOG(\cdot)$ in G takes input an element $a \in G$ and returns $x \in \mathbb{Z}_q$ such that $a = g^x$. The computational Diffie-Hellman (CDH) function $CDH(\cdot,\cdot)$ takes as input a tuple of elements $(a, b) \in G^2$ and returns $g^{DLOG(a)\cdot DLOG(b)}$. The Decisional Diffie-Hellman (DDH) function $\mathrm{DDH}(\cdot,\cdot,\cdot)$ takes as input a triple of elements $(a, b, c) \in G^3$ and returns 1 if $c = CDH(a, b)$ and 0 otherwise.

The *advantage* of an algorithm $\mathcal{M}$ in solving the Discrete Logarithm problem, $\mathbf{Adv}^{\mathrm{DLOG}}(\mathcal{M})$, is the probability that, given $a \xleftarrow{\$} G$, $\mathcal{M}$ correctly returns $DLOG(a)$. Similarly, the advantage of an algorithm $\mathcal{M}$ in solving the Gap Diffie-Hellman (GDH) problem, $\mathbf{Adv}^{\mathrm{GDH}}(\mathcal{M})$, is the probability that, given as input $(a, b) \xleftarrow{\$} G^2$ and oracle access to $\mathrm{DDH}(\cdot,\cdot,\cdot)$, $\mathcal{M}$ correctly outputs $CDH(a, b)$. We say that G satisfies the GDH assumption if no feasible adversary can solve the GDH problem with non-negligible probability. The GDH assumption was introduced by Okamoto and Pointcheval [17] and is now a standard cryptographic assumption used to establish the security of many protocols.

Let G' be another group of order q. A function $e : G \times G \to G'$ is a bilinear pairing if it is non-degenerate and if for any pair $g^a, g^b \in G$, $e(g^a, g^b) = e(g, g)^{ab}$. The Pairing Diffie-Hellman (PDH) problem recently introduced by Mityagin and

Lauter [15] is to solve the CDH problem when given access to the pairing oracle e. The advantage $\mathbf{Adv}^{\mathrm{PDH}}(\mathcal{M})$ of an algorithm $\mathcal{M}$ in solving the PDH problem is the probability that $\mathcal{M}$, given $(a,b) \stackrel{\$}{\leftarrow} G^2$ and a pairing oracle e, computes $CDH(a,b)$. We say that G satisfies the PDH assumption if no feasible adversary solves the PDH problem with non-negligible probability. In groups which have a bilinear pairing, the PDH problem is equivalent to the original CDH problem, although one can also consider the PDH problem in groups where no efficient pairing operation is known. We find the Pairing Diffie-Hellman assumption to be as justified as the GDH assumption since the only known way to compute DDH in groups where CDH is hard is via a pairing function.

4.2 Protocol Description

The NAXOS AKE protocol uses a mathematical group G and two hash functions, $H_1 : \{0,1\}^* \to \mathbb{Z}_q$ and $H_2 : \{0,1\}^* \to \{0,1\}^\lambda$ (for some constant λ). A long-term secret key of a party $\mathcal{A}$ is an exponent $sk_{\mathcal{A}} \in \mathbb{Z}_q$, and the corresponding long-term public key of $\mathcal{A}$ is the power $pk_{\mathcal{A}} = g^{sk_{\mathcal{A}}} \in G$. In the following description of an AKE session of NAXOS executed between the parties $\mathcal{A}$ and $\mathcal{B}$ we assume that each party knows the other's public key and that public keys are in the group G. Additionally, we use the syntax $H(x_1, x_2, ...)$ to represent the application of the hash function H to the concatenation of its arguments $x_1||x_2||....$

The session execution proceeds as follows. The parties pick ephemeral secret keys $esk_{\mathcal{A}}$ and $esk_{\mathcal{B}}$ at random from $\{0,1\}^\lambda$. Then the parties exchange values $g^{H_1(esk_{\mathcal{A}}, sk_{\mathcal{A}})}$ and $g^{H_1(esk_{\mathcal{B}}, sk_{\mathcal{B}})}$, check if received values are in the group G and only compute the session keys if the check succeeds. The session key $K \in \{0,1\}^\lambda$ is computed as

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

The last two components in the hash are the identities of $\mathcal{A}$ and $\mathcal{B}$, which we assume to be binary strings. Figure 2 depicts the protocol.

Theorem 1. *NAXOS satisfies Extended Canetti-Krawczyk security if H_1 and H_2 are modeled by independent random oracles.*

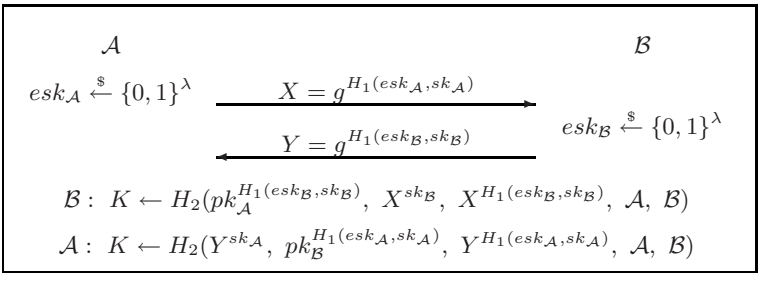

Fig. 2. NAXOS AKE Protocol

For any AKE adversary $\mathcal{M}$ against NAXOS that runs in time at most t, involves at most n honest parties and activates at most k sessions, we show that there exists a GDH solver $\mathcal{S}$, a PDH solver $\mathcal{R}$ and a DLOG solver $\mathcal{T}$ such that

$$\mathbf{Adv}^{GDH}(\mathcal{S}) = \mathbf{Adv}^{PDH}(\mathcal{R})$$

$$\geq \frac{1}{2}\left(\min\left\{\frac{2}{k^2}, \frac{1}{nk}\right\} \cdot \mathbf{Adv}^{AKE}_{NAXOS}(\mathcal{M}) - 2n \cdot \mathbf{Adv}^{DLOG}(\mathcal{T}) - O\left(\frac{k^2}{2^\lambda}\right)\right),$$

where $\mathcal{S}$ runs in time $O(tk)$, $\mathcal{R}$ runs in time $O(t \log t)$ and $\mathcal{T}$ runs in time $O(t)$.

The proof of Theorem 1 is given in Appendix A.

References

1. Bellare, M., Palacio, A.: The Knowledge-of-Exponent Assumptions and 3-Round Zero-Knowledge Protocols. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 273–289. Springer, Heidelberg (2004)
2. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated Key Exchange Secure Against Dictionary Attacks. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 139–155. Springer, Heidelberg (2000)
3. Bellare, M., Rogaway, P.: Entity Authentication and Key Distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 110–125. Springer, Heidelberg (1994)
4. Bellare, M., Rogaway, P.: Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. ACM Conference on Computer and Communications Security, 62–73 (1993)
5. Bellare, M., Rogaway, P.: Provably Secure Session Key Distribution: the Three Party Case. In: STOC 1995. Proc. 27th Annual Symposium on the Theory of Computing, ACM Press, New York (1995)
6. Bellare, M., Rogaway, P.: Introduction to Modern Cryptography. Course notes for UCSD cryptography course, available at `http://www-cse.ucsd.edu/users/mihir/cse207/classnotes.html`
7. Blake-Wilson, S., Johnson, D., Menezes, A.: Key Agreement Protocols and their Security Analysis. In: Darnell, M. (ed.) Cryptography and Coding. LNCS, vol. 1355, pp. 30–45. Springer, Heidelberg (1997)
8. Choo, K.-K.R., Boyd, C., Hitchcock, Y.: Errors in Computational Complexity Proofs for Protocols. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 624–643. Springer, Heidelberg (2005)
9. Choo, K.-K.R., Boyd, C., Hitchcock, Y.: Examining Indistinguishability-Based Proof Models for Key Establishment Protocols. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 585–604. Springer, Heidelberg (2005)
10. Choo, K.-K.R.: A Proof of Revised Yahalom Protocol in the Bellare and Rogaway (1993) Model. The Computer Journal, Oxford University; also available at Cryptology ePrint Archive: Report 2007/188 (to appear, 2007)
11. Canetti, R., Krawczyk, H.: Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 453–474. Springer, Heidelberg (2001)

12. Krawczyk, H.: HMQV: A High-Performance Secure Diffie-Hellman Protocol. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 546–566. Springer, Heidelberg (2005)
13. Jeong, I.R., Katz, J., Lee, D.H.: One-Round Protocols for Two-Party Authenticated Key Exchange. In: Jakobsson, M., Yung, M., Zhou, J. (eds.) ACNS 2004. LNCS, vol. 3089, Springer, Heidelberg (2004)
14. Kudla, C., Paterson, K.G.: Modular Security Proofs for Key Agreement Protocols. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 549–565. Springer, Heidelberg (2005)
15. Lauter, K., Mityagin, A.: Security Analysis of KEA Authenticated Key Exchange. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 378–394. Springer, Heidelberg (2006)
16. Menezes, A.: Another look at HMQV. Journal of Mathematical Cryptology (to appear)
17. Okamoto, T., Pointcheval, D.: The Gap Problems: A New Class of Problems for the Security of Cryptographic Schemes. In: Kim, K.-c. (ed.) PKC 2001. LNCS, vol. 1992, pp. 104–118. Springer, Heidelberg (2001)

A Security Proof for NAXOS

Let $\mathcal{A}$ be any AKE adversary against NAXOS. We start by observing that since the session key of the test session is computed as $K = H_2(\sigma)$ for some 5-tuple σ, the adversary $\mathcal{M}$ has only two ways to distinguish K from a random string:

1. Forging attack. At some point $\mathcal{M}$ queries H_2 on the same 5-tuple σ.
2. Key-replication attack. $\mathcal{M}$ succeeds in forcing the establishment of another session that has the same session key as the test session.

A similar argument was used in the security proofs of the HMQV [12] and KEA+ [15] AKE protocols. If random oracles produce no collisions, the key-replication attack is impossible as equality of session keys implies equality of the corresponding 5-tuples (which are hashed to produce session keys). In turn, distinct AKE sessions must have distinct 5-tuples. Therefore, if random oracles produce no collisions (collisions happen with probability $O(k^2/2^\lambda)$), $\mathcal{M}$ must perform a forging attack. Next we show that if $\mathcal{M}$ can mount a successful forging attack, then we can construct a Gap Diffie-Hellman solver $\mathcal{S}$ which uses $\mathcal{M}$ as a subroutine. Most of the remaining proof is devoted to the construction of $\mathcal{S}$.

$\mathcal{S}$ takes as input a GDH challenge (X_0, Y_0). Then $\mathcal{S}$ executes the Extended Canetti-Krawczyk (ECK) experiment with $\mathcal{M}$ the adversary against the NAXOS protocol, and modifies the data returned by the honest parties in such a way that if $\mathcal{M}$ breaks security of NAXOS, then $\mathcal{S}$ can reveal the solution to the GDH problem from $\mathcal{M}$.

We distinguish between two cases of $\mathcal{M}$'s behavior: whether $\mathcal{M}$ selects a test session for which the matching session exists or if the test session has no matching session. We handle analysis of these cases differently and note that at least one of them happens with probability $\geq 1/2$.

A.1 Matching Session Exists

Assume that $\mathcal{M}$ selects a test session for which the matching session exists. Then $\mathcal{S}$ modifies the experiment as follows. $\mathcal{S}$ selects at random matching sessions executed by some honest parties $\mathcal{A}$ and $\mathcal{B}$ (in fact, $\mathcal{S}$ selects two sessions at random and continues only if they are matching – $\mathcal{S}$ successfully guesses them with probability $2/k^2$). Denote by $comm_A$ and $comm_B$ the communications sent by the respective parties in these matching sessions. When either of these sessions is activated, $\mathcal{S}$ does not follow the protocol. Instead, $\mathcal{S}$ generates $esk_{\mathcal{A}}$ and $esk_{\mathcal{B}}$ normally but sets $comm_A \leftarrow X_0$ (in place of $g^{H_1(sk_{\mathcal{A}}, esk_{\mathcal{A}})}$) and $comm_B \leftarrow Y_0$ (in place of $g^{H_1(sk_{\mathcal{B}}, esk_{\mathcal{B}})}$).

With probability $1/k^2$ $\mathcal{M}$ picks one of the selected sessions as the test session and another as its matching session. We claim that if $\mathcal{M}$ wins in the forging attack, $\mathcal{S}$ can solve the CDH challenge. Indeed, the supposed session key for the selected session is $H_2(\sigma)$, where the 5-tuple σ includes the value $CDH(X_0, Y_0)$. To win, $\mathcal{M}$ must have queried σ to the random oracle H_2.

If the selected session is indeed the test session, $\mathcal{M}$ is allowed to reveal a subset of $\{\ sk_{\mathcal{A}},\ sk_{\mathcal{B}},\ esk_{\mathcal{A}}$ and $esk_{\mathcal{B}}\ \}$, but it is not allowed to reveal both $(sk_{\mathcal{A}}, esk_{\mathcal{A}})$ or both $(sk_{\mathcal{B}}, esk_{\mathcal{B}})$. We observe that in this case, the only way that $\mathcal{M}$ can distinguish this simulated ECK experiment from a true ECK experiment is if $\mathcal{M}$ queries $(sk_{\mathcal{A}}, esk_{\mathcal{A}})$ or $(sk_{\mathcal{B}}, esk_{\mathcal{B}})$ to H_1 (this way, $\mathcal{M}$ will find out that $comm_{\mathcal{A}}$ and $comm_{\mathcal{B}}$ were not computed correctly). Proposition 1 shows that the probability that $\mathcal{M}$ makes such queries is at most

$$2n \cdot \mathbf{Adv}^{\mathrm{DLOG}}(\mathcal{T})$$

for some discrete logarithm solver $\mathcal{T}$.

Therefore (assuming that $\mathcal{M}$ always selects a test session which has a matching session)

$$\mathbf{Adv}^{\mathrm{GDH}}(\mathcal{S}) \ \geq\ \frac{2}{k^2} \cdot \mathbf{Adv}^{\mathrm{AKE}}_{\mathrm{NAXOS}}(\mathcal{M}) - 2n \cdot \mathbf{Adv}^{\mathrm{DLOG}}(\mathcal{T}) - O\left(\frac{k^2}{2^\lambda}\right).$$

Note that in this case $\mathcal{S}$ doesn't make any queries to the DDH oracle and runs in time $O(t)$.

A.2 No Matching Session

Now assume that $\mathcal{M}$ selects a test session for which no matching session exists. In this case $\mathcal{S}$ modifies the experiment as follows. $\mathcal{S}$ selects a random party $\mathcal{B}$ and sets $pk_{\mathcal{B}} \leftarrow X_0$. Note that $\mathcal{S}$ doesn't know the secret key corresponding to this public key and thus it cannot properly simulate ECK sessions executed by $\mathcal{B}$. $\mathcal{S}$ handles ECK sessions executed by $\mathcal{B}$ as follows (assume that $\mathcal{B}$ is the owner). $\mathcal{S}$ randomly selects $esk_{\mathcal{B}}$, picks h at random from $\mathbb{Z}_q$ and sets $comm_{\mathcal{B}} = g^h$ instead of $g^{H_1(esk_{\mathcal{B}}, DLOG(X_0))}$. $\mathcal{S}$ sets a session key K (which is supposed to be $H_2(CDH(X_0, comm_{\mathcal{C}}), pk^h_{\mathcal{C}}, comm^h_{\mathcal{C}}, \mathcal{B}, \mathcal{C})$) to be a random value. Note that $\mathcal{S}$ can handle session key and ephemeral secret key reveals by revealing K and $esk_{\mathcal{B}}$, but cannot handle long-term secret key reveals.

If $\mathcal{C}$ is an adversary-controlled party, $\mathcal{M}$ can compute the session key on its own, reveal K and detect that it is fake. To address this issue, $\mathcal{S}$ watches $\mathcal{M}$'s random oracle queries and if $\mathcal{M}$ ever queries $(Z, pk_{\mathcal{C}}^h, comm_{\mathcal{C}}^h, \mathcal{B}, \mathcal{C})$ to H_2 (for some $Z \in G$), $\mathcal{S}$ checks if $DDH(X_0, comm_{\mathcal{C}}, Z) = 1$ and if yes, replies with the key K. Similarly, on the computation of K, $\mathcal{S}$ checks if K should be equal to any previous response from the random oracle. Because of these checks $\mathcal{S}$ runs in quadratic time of the number of random oracle's queries.

$\mathcal{M}$ cannot detect that it is in the simulated ECK experiment unless it either queries $(esk_{\mathcal{B}}, DLOG(X_0))$ to H_1 or reveals a long-term secret key of $\mathcal{B}$. The first event reveals $DLOG(X_0)$ and allows $\mathcal{S}$ to solve the CDH problem – by Proposition 1 it happens with probability at most

$$n \cdot \mathbf{Adv}^{\mathrm{DLOG}}(\mathcal{T})$$

for some discrete logarithm solver $\mathcal{T}$. The second event is impossible as otherwise the test session will no longer be clean.

$\mathcal{S}$ also randomly selects an ECK session in which $\mathcal{B}$ is the peer. Denote the owner of this session by $\mathcal{A}$. When the selected session is activated, $\mathcal{S}$ follows the protocol only partially: $\mathcal{S}$ generates $esk_{\mathcal{A}}$ normally but sets $comm_A \leftarrow Y_0$ (in place of $g^{H_1(sk_{\mathcal{A}}, esk_{\mathcal{A}})}$).

With probability at least $1/nk$ ($1/n$ to pick the correct party $\mathcal{B}$ and $1/k$ to pick the correct session), $\mathcal{M}$ picks the selected session as the test session, and if it wins, it solves the CDH problem. The supposed session key for the selected session is $H_2(\sigma)$, where the 5-tuple σ includes the value $CDH(X_0, Y_0)$. To win, $\mathcal{M}$ must have queried σ to the random oracle H_2.

If the selected session is indeed the test session, $\mathcal{M}$ is not allowed to reveal both $sk_{\mathcal{A}}$ and $esk_{\mathcal{A}}$ and is not allowed to corrupt $\mathcal{B}$. In this case, the only way that $\mathcal{M}$ can distinguish this simulated ECK experiment from a true ECK experiment is if M queries $(sk_{\mathcal{A}}, esk_{\mathcal{A}})$ to H_1. However, by Proposition 1 it happens with probability at most

$$n \cdot \mathbf{Adv}^{\mathrm{DLOG}}(\mathcal{T})$$

for some discrete logarithm solver $\mathcal{T}$.

Overall, if $\mathcal{M}$ always selects a test session which doesn't have a matching session then the success probability of $\mathcal{S}$ is at most

$$\mathbf{Adv}^{\mathrm{GDH}}(\mathcal{S}) \;\geq\; \frac{1}{nk} \cdot \mathbf{Adv}^{\mathrm{AKE}}_{\mathrm{NAXOS}}(\mathcal{M}) - 2n \cdot \mathbf{Adv}^{\mathrm{DLOG}}(\mathcal{T}) - O\left(\frac{k^2}{2^\lambda}\right),$$

where $\mathcal{T}$ is some discrete logarithm solver. $\mathcal{S}$ runs in time $O(kt)$.

A.3 Efficiency Analysis

We observe that the running time of $\mathcal{S}$ is $O(kt)$. For each session key computation done by $\mathcal{B}$ (where Y is the incoming communication in that session) the solver $\mathcal{S}$ has to go over all previous H_2 queries and for each H_2 query of the form $(\dots, Z, \dots)$ check if $\mathrm{DDH}(X_0, Y, Z) = 1$. Similarly, on each DDH query of the form $(\dots, Z, \dots)$, $\mathcal{S}$ has to go over all previous session key computations

done by $\mathcal{B}$ and for each such computation $\mathcal{S}$ checks if $\mathrm{DDH}(X_0, Y, Z)$ (where Y the incoming communication in that session). Since $\mathcal{M}$ can activate at most k sessions and make at most t H_2 queries, the total running time is $O(tk)$.

The running time of the solver can be improved if the solver has access to the pairing oracle instead of to the DDH oracle. We construct the PDH solver $\mathcal{R}$ in the same way as $\mathcal{S}$ with the only difference being that $\mathcal{R}$ must also handle the checks discussed above. Note that $\mathrm{DDH}(X_0, Y, Z) = 1$ if and only if $e(Z, g) = e(X_0, Y)$. Therefore $\mathcal{R}$ can store corresponding values $e(Z, g)$ in a balanced binary tree and on each session executed by $\mathcal{B}$ check for X_0, Y by computing $e(X_0, Y)$ and searching for this value in the binary tree (which can be done in $\log t$ steps). Therefore, $\mathcal{R}$ has the same advantage as $\mathcal{S}$ and runs in time $O(t \log t)$.

A.4 Reduction to the Discrete Logarithm Problem

Finally, we are left to prove the proposition which reduces breaking secret keys of honest parties to solving the Discrete Logarithm problem.

Consider any adversary $\mathcal{M}$ against the NAXOS protocol. $\mathcal{M}$ can obtain long-term secret keys of some honest parties via Long-Term Key Reveal queries and can attempt to break long-term keys of uncorrupted parties. We claim that he cannot do so unless he solves the discrete logarithm problem. Let "$\mathcal{M}$ breaks a secret key" denote an event that $\mathcal{M}$ makes a random oracle query $H_1(*, sk_{\mathcal{A}})$ for some honest party $\mathcal{A}$ against which $\mathcal{M}$ didn't make the Long-Term Key Reveal query.

Proposition 1. *For any adversary $\mathcal{M}$ against the NAXOS protocol who runs in time t and involves at most n honest parties, there exists a discrete logarithm solver $\mathcal{T}$ such that*

$$\mathit{Prob}[\text{“}\mathcal{M} \textit{ breaks a secret key}\text{”}] \;<\; n\mathit{Adv}^{DLOG}(\mathcal{T}),$$

where $\mathcal{T}$ runs in time $O(t)$.

Proof. Since the security experiment involves at most n honest parties, we can assume that $\mathcal{M}$ queries $(*, sk_{\mathcal{A}})$ to H_1 for a certain party $\mathcal{A}$ with probability at least

$$\frac{1}{n}\mathrm{Prob}[\text{“}\mathcal{M} \text{ breaks a secret key”}].$$

The discrete logarithm solver $\mathcal{T}$ is given a challenge X; T runs the AKE experiment with $\mathcal{M}$ and sets the public key of a party $\mathcal{A}$ to be X.

$\mathcal{T}$ can perfectly simulate all actions of the parties except for computing $H_1(esk_{\mathcal{A}}, sk_{\mathcal{A}})$ during key-exchange sessions involving $\mathcal{A}$ (here $sk_{\mathcal{A}}$ is supposed to be $DLOG(X)$). In these cases $\mathcal{T}$ randomly selects distinct random oracle values for distinct values of $esk_{\mathcal{A}}$.

The only way that $\mathcal{M}$ can distinguish this simulation from the true experiment is by querying $(esk_{\mathcal{A}}, DLOG(X))$ to the random oracle. However in this case (as we see below) $\mathcal{T}$ automatically wins the DLOG experiment.

Whenever $\mathcal{M}$ makes a query of the form (y, z) to the random oracle H_1, $\mathcal{T}$ verifies whether $X = g^z$ and if true, submits z as an answer to the DLOG experiment. Note that in the simulated experiment $\mathcal{M}$ makes a random oracle query $(*, sk_{\mathcal{A}} = DLOG(X))$ with probability at least

$$\frac{1}{n}\text{Prob}[\text{“}\mathcal{M} \text{ breaks a secret key”}].$$

Therefore, $\mathcal{T}$ succeeds in solving discrete logarithm of X at least with this probability.