Stronger Security of Authenticated Key Exchange

Brian LaMacchia
Microsoft Corp.
1 Microsoft Way, Redmond, WA
bal@exchange.microsoft.com

Kristin Lauter
Microsoft Research
1 Microsoft Way, Redmond, WA
klauter@microsoft.com

Anton Mityagin
University of California, San Diego
9500 Gilman Dr., La Jolla, CA
amityagin@cs.ucsd.edu

Abstract

In this paper we study security definitions for authenticated key exchange (AKE) protocols. We observe that there are several families of attacks on AKE protocols that lie outside the boundary of the current class of security definitions. In an attempt to bring these attacks within the scope of analysis we extend the AKE security definition to provide greater powers to the adversary. We provide a general framework for defining AKE security, which we call *strong AKE security*, such that existing security definitions occur as instances of the framework. We then introduce NAXOS, a new two-pass AKE protocol, and prove that it is secure in this stronger definition.

In addition, we formulate a notion of ephemeral secret key which captures all ephemeral information used in session establishment. We demonstrate the importance of this formulation by showing that a secure AKE protocol SIG-DH can become vulnerable when instantiated with signature schemes which are insecure against revelation of the secret random bits used in the signature generation.

1 Introduction

Security of authenticated key exchange (AKE) has a long history. To date, there exist essentially 4 different security definitions [4, 6, 3, 10]. Recently, Choo et al [8] compared these four definitions and showed that almost no one of the definitions is stronger than another one of them. On the other hand, there is clearly still a need for a good security definition, since there have been a great number of AKE protocols proposed, and many of them were subsequently broken. Recent work by Krawczyk [14] and Menezes [17] has highlighted the importance of understanding well the guarantees and limitations of formal security definitions when using them to prove the security of protocols.

In this paper, we unify the four existing security definitions into one framework. We then extend the security definition with new adversarial powers which allow our definition to capture additional attacks. Our general framework, called AKE security, considers the same security experiment as in [4] and [10], but allows different choices for four aspects of the security definition. This experiment involves parties which are connected by an unauthenticated network and an adversary who controls all communications. Parties execute key exchange sessions when activated by the adversary. Eventually, the adversary selects a test session and is given a challenge in which he must distinguish the session key from a random string. The protocol is said to be secure if the adversary has negligible advantage in correctly guessing the challenge. In the framework we propose, a particular security definition is specified by making choices for the following four aspects of the experiment: 1) definition of matching sessions and session identifiers; 2) key registration procedure; 3) adversarial powers; 4) definition of a fresh session. All four security definitions mentioned above occur (with minor technical changes) as instances in this framework.

In this framework, we extend and strengthen powers available to the adversary to obtain a new security definition which we call *strong AKE security*. Our security notion is strong in the following sense: the only corruption powers we do not give an adversary in the experiment are those that would allow him to trivially break any AKE protocol. More specifically, in an authenticated key exchange protocol, two parties exchange information and compute a secret key as a function of four pieces of secret information: their own long-term and ephemeral secret keys and the other party's long-term and ephemeral secret keys. Of the four pieces of information, we allow an adversary to reveal any subset of the four which does not contain both the long-term and ephemeral secrets of one of the parties. Extending AKE security in this way is natural, because previously queries of these types were allowed on sessions other than the test session anyway. It is reasonable to assume that if an adversary has such powers in other sessions, then it would have those same powers on the test session.

Additionally, we clarify what constitutes the ephemeral secret key of a party in a session and what information can be revealed by an adversary via a Session-State Reveal query. We require that the ephemeral secret key contain *all* session-specific secret information of a party. We illustrate the need for this requirement with the SIG-DH protocol [10], which provably satisfies Canetti-Krawczyk security if only part of the session-specific secret is revealed and which can be attacked if all session-specific secret information is leaked. The Canetti-Krawczyk definition allows a protocol to specify the data to be revealed by the Session-State Reveal query. This makes the security definition dependent on a protocol design and makes it hard to figure out what level of security the protocol provides. When some protocol is said to achieve AKE security, it is not clear what portion of the data is allowed to be revealed. By introducing a definition of ephemeral secret key and Ephemeral Key Reveal query, we avoid this ambiguity: 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 the literature.

Finally, we present a new two-pass AKE protocol, called NAXOS, which provably meets our definition of strong AKE security. We prove the security of NAXOS either under the Gap Diffie-Hellman assumption or under the Pairing Diffie-Hellman assumption (which provides for better concrete security). NAXOS combines the highest security level with comparable efficiency. Figure 1 compares security and efficiency of NAXOS with other recent AKE protocols such as HMQV [14], KEA+ [16], the protocol analyzed by Kudla and Paterson[15] and the protocol $\mathcal{TS}3$ of Jeong, Katz and Lee [9].

Another advantage of our protocol over HMQV [14] is that it has a tight concrete security reduction (under the Pairing Diffie Hellman Assumption [16]). HMQV doesn't have a concrete security analysis and, as noted by Menezes [17], the reduction provided in HMQV appears to be inefficient.

Summary of Our Contributions:

1. We provide a general framework for defining AKE security such that existing security definitions occur as instances of the framework.

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

Figure 1: Comparison of recent AKE protocols. Efficiency is given by the number of exponentiations executed by one party. Communication in all these protocols (except for Katz-Jeong-Lee) is the same as in the original Diffie-Hellman. Key registration column specifies how parties register their public keys (see Section 2). Ephemeral column indicates whether an adversary is allowed to reveal ephemeral secret information of the parties. Security of all protocols is analyzed assuming that partnership is defined via matching conversations[1]. AKE denotes basic AKE security (basic definition of freshness; no perfect forward secrecy), KCI denotes security against key-compromise impersonation and wPFS denotes weak perfect forward secrecy. Strong AKE denotes our new security definition. Security assumptions include: RO – random oracle model [5], DDH – decisional Diffie-Hellman, GDH – Gap Diffie-Hellman [18], PDH – Pairing Diffie-Hellman [16] and KEA1 – knowledge of exponent assumption [2].

2. We extend AKE security with new adversarial powers and formalize a definition of strong AKE security which captures new attacks allowed by these powers.
3. We extend the notion of ephemeral secret key to include all ephemeral secret information of the party.
4. We point out a new attack on the SIG-DH protocol by Canetti and Krawczyk [10].
5. We discuss the pitfalls of using Bellare-Canetti-Krawczyk authenticators in security models where partnership is defined via matching conversations.
6. We propose NAXOS, a new AKE protocol which we prove satisfies strong AKE security; we also analyze the concrete security of NAXOS.

2 AKE Security

2.1 Framework for Defining the Security of Authenticated Key Exchange

The AKE security experiment involves multiple parties $\mathcal{A}, \mathcal{B}, \mathcal{C}, \ldots$ and an adversary $\mathcal{M}$ connected via an unauthenticated network. Each party can be activated to run an instance of the protocol, called an AKE *session*. The adversary selects parties to execute AKE sessions and selects an order in which the sessions will be executed.

Each party is supposed to register its public key with an authority; after registration any other party can securely obtain this registered public key. Existing definitions consider three variants of how key registration can be done:

- Honest key registration. All parties (even ones controlled by the adversary) follow the key generation procedures honestly and register the obtained keys at the beginning of experiment. The adversary can corrupt parties only after key registration has completed.

[1]Kudla and Paterson [15] define partnership via matching session identifiers (computed by the parties), although for their protocol this seems to be equivalent to matching conversations.

- Proof-of-possession. An authority performs some validity check (specified by the protocol) upon key registration. Usually, a party is required to prove knowledge of the corresponding secret key. The adversary can register keys for corrupted parties at any time in the experiment.

- Arbitrary key registration. Parties can register arbitrary keys (even the same key as some other party) without any validity checks. The adversary can register keys for corrupted parties at any time in the experiment.

When a party completes an AKE session, it computes a session key. 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*. A session which is supposed to be executed by a peer is called the *matching session* to an AKE session executed by the owner. The matching session might not exist if the communications were modified by the adversary. Existing definitions provide three ways to define matching sessions:

- Matching conversations [4]. Matching sessions are those which participate in matching conversations. Session identifiers are concatenations of messages which are sent and received by a party.

- Partner functions [6]. Associated to a protocol is a partner function, which takes as input a communication sent/received by a party in an AKE session and outputs the identity of its partner.

- Session IDs are assigned by the adversary [10]. The adversary, when activating a new AKE session executed by some honest party $\mathcal{A}$, provides to $\mathcal{A}$ a string of his choice which is used as a session identifier. Matching sessions are those having the same session identifiers. The adversary is not allowed to assign the same session identifier to different sessions executed by the same party.

The adversary controls the communications exchanged by the parties and can also corrupt parties in several ways. Existing security definitions specify subsets of queries the adversary can make out of the following queries.

- $\mathsf{Send}(M, \mathcal{A}, sid)$ – sends a message M to a session sid executed by a party $\mathcal{A}$ (on behalf of some other party) and returns a response according to the protocol specification. If the Send query is allowed, the adversary controls all the communication and can cancel and modify the existing messages as well as insert new ones. If the Send query is not allowed, the adversary can only passively eavesdrop on messages sent by the parties (in this case the communication is authenticated).

- $\mathsf{Reveal}(sid, \mathcal{A})$ – reveals a session key computed by $\mathcal{A}$ in a completed session sid.

- $\mathsf{Corrupt}(\mathcal{A})$ – by making this query the adversary takes full control of a party $\mathcal{A}$ and obtains the long-term secret key of $\mathcal{A}$ as well as the ephemeral secret information of all current AKE sessions run by $\mathcal{A}$. Sometimes $\mathcal{M}$ is also allowed to replace $\mathcal{A}$'s long-term public key with a value of his choice. In order to capture perfect forward secrecy, session keys of past AKE sessions by $\mathcal{A}$ are not revealed.

- Session-State Reveal$(\mathcal{A}, sid)$ – reveals session-specific secret information (specified by a protocol) of a session sid being executed by $\mathcal{A}$.

- Test$(\mathcal{A}, sid)$ – the only oracle query that does not correspond to any of A's abilities. Depending on a randomly chosen bit b, $\mathcal{A}$ is given either the actual session key or a session key drawn randomly from the session key distribution. The adversary is limited to only one such query, which can be made at any time during the experiment. The target AKE session is called the test session.

Finally, a security experiment must specify a notion of "freshness" for AKE sessions. Since the adversary may mount different corruptions against the test session, he may be able to trivially solve the challenge. To make the security definition meaningful, the adversary should only run a Test query on uncorrupted sessions. A notion of freshness specifies what "uncorrupted" means.

- The basic notion of freshness (which is the same in Bellare-Rogaway [4, 6] and Canetti-Krawczyk [10]) states that a session is fresh if the adversary does not corrupt the owner or the peer of the session and, additionally, the adversary must not mount any Revealor Session-State Revealqueries against the session or its matching session (if the latter exists).

- In addition to the basic notion for freshness, Bellare-Pointcheval-Rogaway [3] also prohibits the adversary from making Corrupt queries against any party.

- In order to capture perfect forward secrecy, some definitions of freshness allow the adversary to corrupt the owner or the peer of the session after the session or the matching session respectively are completed.

The adversary's goal is to guess whether the challenge is a true session key or a randomly selected key. When the adversary terminates, it outputs a bit b'. The adversary wins the experiment if the test session is fresh and if $b' = b$, where b is the random bit chosen in the Test query. We remark that freshness of the session is determined at the end of the experiment and not at the time the Test query was made. The *advantage* $\mathbf{Adv}_{\Phi}^{\mathrm{AKE}}$ of the adversary $\mathcal{M}$ in breaking AKE security of an AKE protocol Φ is defined as:

$$\mathbf{Adv}_{\Phi}^{\mathrm{AKE}}(\mathcal{A}) = \Pr\left[\mathcal{A} \text{ wins the above AKE experiment with } \Phi\right] - \frac{1}{2}.$$

Definition 1 (General AKE Security) *Consider some choice of requirements for key registration, partnership, adversarial abilities and a notion of freshness. The protocol Φ is said to achieve AKE security under the specified definition if*

1. *Parties who complete matching sessions compute the same session key.*

2. *For any efficient adversary $\mathcal{A}$, $\mathbf{Adv}_{\Phi}^{AKE}(\mathcal{A})$ is sufficiently small.*

Remark 1 *We remark that when partnership is defined via matching conversations, the first requirement of Definition 1 is not necessary. If parties send and receive communications without adversary's modifications (that is, execute matching sessions), the correctness of a protocol guarantees that they compute the same session keys. On the other hand, if session identifiers are assigned by the adversary, this requirement becomes crucial.*

2.2 Review of Existing Security Notions

The first definition of AKE security by Bellare and Rogaway [4] gives rather limited powers to the adversary. All parties are assumed to perform key registration honestly and the adversary only has access to Send and Reveal queries. Partnership is defined via matching conversations and a session is considered to be fresh if the adversary did not make any Reveal queries against either the session itself or its matching session. Subsequent definitions by Bellare and Rogaway [6] and Bellare, Pointcheval and Rogaway [3] introduce Corrupt queries and modify the definition of freshness to capture perfect forward secrecy. We refer the reader to Choo, Boyd and Hitchcock [8] for a comprehensive comparison of these security definitions.

Canetti and Krawczyk [10] extended the security definition by adding the Session-State Reveal query, which allows the adversary to reveal ephemeral secret information of the parties. Also, they introduced a model in which the adversary provides session identifiers (SIDs) to the parties and partnership is defined via matching SIDs.

We stress that the approach of [10] to defining partnership via SIDs assigned by the adversary is very different from defining partnership via matching conversations, even if the other parameters of the AKE security definition are the same. Not only are the two security notions independent (neither of them is implied by the other), but also the protocols under either of the definitions can not be generally transformed into protocols under the other definition in a way that preserves session identifiers. For example, a protocol under the Canetti-Krawczyk definition that uses SIDs in the middle of a protocol execution cannot generally be transformed into a protocol under the Bellare-Rogaway definition. (In the latter definition the SID, which is a concatenation of exchanged communications, becomes available only after the protocol is completed.) On the other hand, in the Bellare-Rogaway definition SIDs are concatenations of all communications exchanged by the parties; clearly, they cannot be assigned by the adversary.

Choo, Boyd and Hitchcock [8] showed that if partnership in Canetti-Krawczyk is defined via matching conversations (which would drastically change the definition), then this new definition of AKE security is the strongest of a family of definitions that also includes those of Bellare and Rogaway [4] and Bellare, Pointcheval and Rogaway [3].

Krawczyk [14] presented a modification to the Canetti-Krawczyk definition of AKE security in which he defined partnership via matching conversations. Additionally, Krawczyk introduced two extra security notions: security against key-compromise impersonation [7] and weak perfect forward secrecy (wPFS). Krawczyk showed that when an adversary is allowed to make Session-State Reveal queries, no two-move protocol can achieve full PFS. 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, these are sessions for which the matching session completes) and can be achieved by two-move protocols.

2.3 Attacks not Covered by the Existing Definitions

We observe the surprising fact that although the Canetti-Krawczyk definition allows the adversary to reveal ephemeral secret information of the parties via Session-State Reveal queries, it prohibits making these queries against the test session. This essentially limits the adversary's abilities as he can only reveal ephemeral information of AKE sessions he doesn't want to attack. Although Session-State Reveal queries are allowed, the Canetti-Krawczyk security definition does not provide any security guarantees for a session if the ephemeral secret key of either party has been leaked.

Another potential weakness of previous definitions is the definition of the Corrupt query. In existing security definitions, when the adversary makes a Corrupt query he takes a full control over the target party from that moment on. This definition does not allow attacks involving revelation of a long-term secret key of some party prior to the time when that party executes the test session. Key-compromise impersonation [7] is probably the most important such attack of this type.

We point out a few examples of attacks which are technically allowed by existing definitions but which are not considered violations of protocol security:

- Key-compromise impersonation (KCI) attack [7]: 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 who initiates an AKE session and impersonates the other participant of this session.
- Two honest parties execute matching sessions, while the adversary reveals ephemeral secret keys of both parties and tries to learn the session key.
- Two honest parties execute matching sessions, and 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, while the adversary reveals long-term keys of both of the parties prior to the session execution and tries to learn the session key.

2.4 Our Variant of AKE Security: Strong AKE Security

As we showed earlier, the existing security definitions give the adversary very strong abilities in corrupting the parties, but they essentially limit his ability to fully utilize these powers. We are motivated to address this surprising gap in existing AKE security definitions and also by a desire to create a single over-arching AKE security definition which guarantees resistance to all the above attacks. We present a new variant of AKE security, called "strong AKE security," which improves upon the existing definitions by allowing the adversary to mount the maximum number of corruptions possible against the test session. We believe that our definition is the strongest of practical significance in that giving any more powers to the adversary will allow him to break security of any AKE protocol. More specifically, in an AKE protocol, two parties exchange information and compute a secret key as a function of 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 information, we allow an adversary to reveal any subset of the four which does not contain both the long-term and ephemeral secrets of one of the parties.

To this end, we introduce a new adversarial query, called Long-Term Key Reveal($\mathcal{A}$), which reveals a long-term secret key of a party $\mathcal{A}$ without giving the adversary full control of $\mathcal{A}$ or revealing any ephemeral secret information. We do not use the original Corrupt query as it is no longer necessary: the adversary can achieve the result of the Corrupt query by revealing all the secret information of the party via Long-Term Key Reveal, Session-State Reveal and Reveal queries; then he can perform any operation on behalf of that party.

Another important change we introduce concerns revelation of ephemeral secret information via Session-State Reveal queries. These queries are motivated by practical scenarios, such as if the state-specific secret information is stored in insecure memory on a device or if the random-number generator of a party is corrupted. The Canetti-Krawczyk definition of AKE security [10] leaves it up to a protocol to specify which data is allowed to be revealed by the Session-State Reveal query. We find this ambiguity in the definition of Session-State Reveal to be rather unsettling. It makes the security definition dependent on a protocol design and thus, makes it hard to figure out what level of security the protocol provides. To illustrate this problem, in Section 3 we present an attack on the SIG-DH protocol which was proven to achieve Canetti-Krawczyk security in [10] (where the adversary is only allowed to reveal part of the ephemeral secret information). We describe the adversary who reveals all the ephemeral information of a party and is able to compute a long-term secret key of that party. We address this ambiguity by specifying what constitutes the "ephemeral secret information" of an AKE session and introducing an Ephemeral Key Reveal query. We require that an ephemeral secret key of an AKE session should contain *all* session-specific information used by a party in the AKE session. That is, all computations done by a party must deterministically depend on that party's ephemeral key, long-term secret key, and communication received from the other party. We remark that any AKE protocol can conform to this specification by setting all random coins used by a party in an AKE session as the ephemeral secret key of that session. In turn, the Ephemeral Key Reveal$(\mathcal{A}, sid)$ query reveals an ephemeral secret key of AKE session sid run by a party $\mathcal{A}$. In our security definition, we replace Session-State Reveal query by Ephemeral Key Reveal.

Definition 2 (Freshness for Strong AKE Security) *Let sid be a completed session, which was executed by a party $\mathcal{A}$ with another party $\mathcal{B}$. We define sid to be fresh if none of the following conditions hold:*

- *$\mathcal{M}$ reveals the session key of sid or of its matching session (if the latter exists).*
- *$\mathcal{B}$ is engaged in a session sid^*, matching to sid, and $\mathcal{M}$ either makes queries:*
  - *both* Long-Term Key Reveal$(\mathcal{A})$ *and* Ephemeral Key Reveal$(\mathcal{A}, sid)$ *or*
  - *both* Long-Term Key Reveal$(\mathcal{B})$ *and* Ephemeral Key Reveal$(\mathcal{B}, sid^*)$.
- *No sessions matching to sid exist and $\mathcal{M}$ either makes queries:*
  - *both* Long-Term Key Reveal$(\mathcal{A})$ *and* Ephemeral Key Reveal$(\mathcal{A}, sid)$ *or*
  - Long-Term Key Reveal$(\mathcal{B})$ *before the completion of session sid.*

Definition 3 (Freshness for Strong AKE Security for Two-Pass Protocols) *A session sid is fresh if it satisfies a variant of Definition 2 where the last requirement is replaced by the following:*

- *No sessions matching to sid exist and $\mathcal{M}$ either makes queries:*
  - *both* Long-Term Key Reveal$(\mathcal{A})$ *and* Ephemeral Key Reveal$(\mathcal{A}, sid)$ *or*
  - Long-Term Key Reveal$(\mathcal{B})$ *at any time.*

Definition 4 *We say that an AKE protocol satisfies strong AKE security (resp., strong AKE security for two-pass protocols) if it satisfies Definition 1 where the AKE experiment involves:*

- *Arbitrary key registration*
- *Partnership is defined via matching conversations*
- *Adversary is allowed to make* Send*,* Reveal*,* Ephemeral Key Reveal *and* Long-Term Key Reveal *queries.*

- *Freshness is defined via Definition 2 (resp. Definition 3)*

Our notion of strong AKE security considers the strongest model for key registration and gives the adversary the strongest powers compared to the previous definitions. Strong AKE security implies all existing security definitions, where partnership is defined via matching conversations. Strong AKE security is strictly stronger than any of the previously defined security definitions because it covers additionally all of the attacks described in Section 2.3.

As noted by Krawczyk [14], no two-pass AKE protocol can achieve full perfect forward secrecy if partnership is defined via matching conversations and revelation of ephemeral secret information is allowed. Consequently, no two-pass protocol can achieve strong AKE security, which is why we define strong AKE security separately for the case of two-pass protocols. Strong AKE security for two-pass protocols implies Canetti-Krawczyk security with weak perfect forward secrecy as well as security against KCI attacks as defined in [14], and thus is strictly stronger than the latter.

3 Attack on a SIG-DH Protocol

3.1 Attack against SIG-DH

The Canetti-Krawczyk [10] definition of AKE security grants the adversary powers to reveal state-specific information of a party without revealing its long-term secret key via Session-State Reveal query. As an example of a protocol which is secure against such adversaries, Canetti and Krawczyk present the SIG-DH protocol, which can be instantiated with any digital signature scheme. All parties in the SIG-DH protocol are assumed to possess secret keys for this signature scheme. The SIG-DH protocol is depicted in Figure 2, where $SIG_{\mathcal{A}}(M)$ denotes a signature of M under a secret key of the party $\mathcal{A}$. Canetti and Krawczyk [10] prove that SIG-DH is secure against an adversary

$\mathcal{A}$		$\mathcal{B}$
$x \xleftarrow{\$} \mathbb{Z}_q$	$\xrightarrow{\mathcal{A}, sid, g^x}$	
	$\xleftarrow{\mathcal{B}, sid, g^y, SIG_{\mathcal{B}}(\mathcal{B}, sid, g^y, g^x, \mathcal{A})}$	$y \xleftarrow{\$} \mathbb{Z}_q$
	$\xrightarrow{\mathcal{A}, sid, SIG_{\mathcal{A}}(\mathcal{A}, g^x, g^y, \mathcal{B})}$	
$K \leftarrow g^{xy}$		$K \leftarrow g^{xy}$

Figure 2: SIG-DH authenticated key-exchange protocol [10]. Session identifier *sid* is provided to both parties by the adversary.

who can reveal ephemeral secret keys (x or y from Figure 2) of the parties.

We observe that the values x and y are not only session-specific secret information used by the parties—they also includes secret random coins used by the parties in the signature generation. Moreover, if the adversary is able to reveal those random coins, it can violate the security of the protocol. Specifically, for certain digital signature schemes such as ElGamal[?], DSA[12], GQ[13] or Schnorr[19], by revealing random coins used in signature generation the adversary can obtain a long-term secret key of the party. We formulate this deficiency in Proposition 2, which is well-known.

Proposition 2 *For randomized signature schemes such as ElGamal, Schnorr, DSA and GQ there exists an efficient algorithm which given a public key, a legitimately computed signature of any message (under the corresponding secret key) and random coins used by a signer, computes the secret key.*

Using the above proposition, the adversary can attack the SIG-DH protocol instantiated with one of the specified signature schemes as follows. First, the adversary initiates an AKE session between two parties $\mathcal{A}$ and $\mathcal{B}$ with an arbitrary session identifier sid and delivers all messages $\mathcal{A}$ and $\mathcal{B}$ send to each other. The adversary records $\mathcal{A}$'s public key and a signature included in the last message from $\mathcal{A}$. Next, the adversary obtains the random coins used by $\mathcal{A}$ during signature generation by making the query $\mathsf{Session\text{-}State\ Reveal}(\mathcal{A}, sid)$. Finally, the adversary uses Proposition 2 to obtain the long-term secret key of $\mathcal{A}$.

3.2 Remarks on Bellare-Canetti-Krawczyk Authenticators

Our attack on SIG-DH raises an obvious question requiring further discussion, namely how does this attack co-exist with the security proof presented in [10]. The SIG-DH protocol was constructed from the two-pass protocol 2DH [10] via the signature-based authenticator of Bellare, Canetti and Krawczyk [1]. The 2DH protocol was proven to be secure in the ideally-authenticated model AM (where the Send query is not allowed). Authenticators, as defined in [1], convert any protocol π designed for AM into some "equivalent" protocol π' in the setting of AKE experiment. (This equivalency is non-trivially defined in terms of UC functionalities provided by protocols.) Theorem 1 from [10] established that applying any authenticator of [1] to protocols secure in AM produces secure AKE protocols under Canetti-Krawczyk definition.

Our first observation, which is not obvious from [10] and [1], is that Session-State Reveal queries in the Canetti-Krawczyk model do not apply equally to both π and π'. Specifically, a Session-State Reveal query may only reveal ephemeral information from the original protocol π and cannot reveal new ephemeral information that was introduced in π' by the authenticator. This explains how Theorem 1 of Canetti and Krawczyk [10] and results of Bellare, Canetti and Krawczyk [1] may co-exist with our attack on SIG-DH.

Our second observation is that Theorem 1 from [10] only holds for the Canetti-Krawczyk variant of AKE security in which session identifiers are assigned by the adversary. Theorem 1 does not hold if partnership is defined via matching conversations, even if Session-State Reveal queries are allowed. A counter-example to Theorem 1 under the matching conversations definition of partnership is the signature-based MT-authenticator λ_{SIG} of [1] instantiated with the following digital signature scheme SIG. Let SIG be allowed to efficiently modify existing signatures to obtain different valid signatures on the same message but be secure against forging signatures on new messages. One can repeat the proof of Proposition 4 from [1] to make sure that λ_{SIG} is still a secure authenticator. On the other hand, by applying λ_{SIG} to any protocol in AM, we obtain an insecure AKE protocol (if partnership is defined via matching conversations). The AKE adversary may use a vulnerability of SIG to modify signatures sent between the parties, causing communicating parties that participate in different AKE sessions to compute the same session key, thus violates AKE security.

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 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 $\mathcal{M}$ given $a \xleftarrow{\$} G$ returns $DLOG(g)$. 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 $\mathcal{M}$ given as input $(a, b) \xleftarrow{\$} G^2$ and oracle access to $\mathrm{DDH}(\cdot, \cdot, \cdot)$ 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 [18] and is now a standard cryptographic assumption used to establish the security of multiple 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 [16] 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 $(a, b) \xleftarrow{\$} G^2$ 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 as 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.

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 3 depicts the protocol.

$$
\begin{array}{ccc}
\mathcal{A} & & \mathcal{B} \\
esk_{\mathcal{A}} \stackrel{\$}{\leftarrow} \{0,1\}^{\lambda} & \xrightarrow{X = g^{H_1(esk_{\mathcal{A}}, sk_{\mathcal{A}})}} & \\
 & \xleftarrow{Y = g^{H_1(esk_{\mathcal{B}}, sk_{\mathcal{B}})}} & esk_{\mathcal{B}} \stackrel{\$}{\leftarrow} \{0,1\}^{\lambda}
\end{array}
$$

$$\mathcal{B}: \; K \leftarrow H_2(pk_{\mathcal{A}}^{H_1(esk_{\mathcal{B}}, sk_{\mathcal{B}})}, \; X^{sk_{\mathcal{B}}}, \; X^{H_1(esk_{\mathcal{B}}, sk_{\mathcal{B}})}, \; \mathcal{A}, \; \mathcal{B})$$

$$\mathcal{A}: \; K \leftarrow H_2(Y^{sk_{\mathcal{A}}}, \; pk_{\mathcal{B}}^{H_1(esk_{\mathcal{A}}, sk_{\mathcal{A}})}, \; Y^{H_1(esk_{\mathcal{A}}, sk_{\mathcal{A}})}, \; \mathcal{A}, \; \mathcal{B})$$

Figure 3: NAXOS AKE Protocol.

Theorem 3 *NAXOS satisfies the strong AKE security for two-pass protocols if H_1 and H_2 are modeled by independent random oracles.*

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 construct GDH solver $\mathcal{S}$, PDH solver $\mathcal{R}$ and DLOG solver $\mathcal{T}$ such that

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

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 outline of the security proof of Theorem 3 is given in Appendix A.

References

[1] M. Bellare, R. Canetti and H. Krawczyk, *A Modular Approach to the Design and Analysis of Authentication and Key Exchange Protocols*, STOC '98: Proceedings of the thirtieth annual ACM symposium on Theory of computing, ACM Press, 1998

[2] M. Bellare, A. Palacio, *The Knowledge-of-Exponent Assumptions and 3-Round Zero-Knowledge Protocols*, Advances in Cryptology — CRYPTO '04, pp. 273–289, Springer-Verlag, 2004

[3] M. Bellare, D. Pointcheval, P. Rogaway, *Authenticated Key Exchange Secure Against Dictionary Attacks*, Advances in Cryptology — Eurocrypt '00, pp. 139–155, Springer-Verlag, 2000

[4] M. Bellare and P. Rogaway, *Entity Authentication and Key Distribution*, Advances in Cryptology — CRYPTO '93, pp. 110–125, Springer-Verlag, 1993

[5] M. Bellare and P. Rogaway, *Random Oracles are Practical: A Paradigm for Designing Efficient Protocols*, ACM Conference on Computer and Communications Security 1993, pp. 62–73

[6] M. Bellare and P. Rogaway, *Provably Secure Session Key Distribution: the Three Party Case*, STOC '95: Proc. 27th Annual Symposium on the Theory of Computing, ACM, 1995

[7] S. Blake-Wilson, D. Johnson, and A. Menezes, *Key Agreement Protocols and their Security Analysis*, 6th IMA International Conference on Cryptography and Coding, LNCS 1355, pp. 30-45, Springer-Verlag, 1997

[8] K.-K. R. Choo, C. Boyd and Y. Hitchcock, *Examining Indistinguishability-Based Proof Models for Key Establishment Protocols*, Advances in Cryptology — Asiacrypt '05, Springer-Verlag, 2005

[9] I. R. Jeong, J. Katz, D. H. Lee, *One-Round Protocols for Two-Party Authenticated Key Exchange*, ACNS '04, 2004

[10] R. Canetti and H. Krawczyk, *Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels*, Advances in Cryptology — EUROCRYPT '01, pp. 453–474, Springer-Verlag, 2001

[11] T. Elgamal, *A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms*, IEEE Transactions on Information Theory, 1985

[12] Federal Information Processing Standards Publication (FIPS PUB) 186, *Digital Signature Standard*, May 19, 1994.

[13] L. Guillou and J. J. Quisquater, *A Paradoxical Identity-Based Signature Scheme Resulting from Zero-knowledge*, Advances in Cryptology — CRYPTO 88, LNCS 403, Springer-Verlag, 1988

[14] H. Krawczyk, *HMQV: A High-Performance Secure Diffie-Hellman Protocol*, Advances in Cryptology — CRYPTO '05, LNCS 3621, pp. 546–566, Springer-Verlag, 2005

[15] C. Kudla and K. G. Paterson, *Modular Security Proofs for Key Agreement Protocols*, Advances in Cryptology — ASIACRYPT '05, pp. 549–565, Springer-Verlag, 2005

[16] K. Lauter and A. Mityagin, *Security Analysis of KEA Authenticated Key Exchange*, to appear in PKC '06, IACR Eprint archive, `http://eprint.iacr.org/2005/265`, 2005

[17] A. Menezes, *Another Look at HMQV*, IACR Eprint archive, `http://eprint.iacr.org/2005/205`, 2005

[18] T. Okamoto and D. Pointcheval, *The Gap Problems: A New Class of Problems for the Security of Cryptographic Schemes*, Public Key Cryptology — PKC '01, LNCS 1992, pp. 104–118, Springer-Verlag, 2001

[19] C. P. Schnorr, *Efficient Signature Generation by Smart Cards*, Journal of Cryptology 4, 3, pp. 161-174, 1991.

[20] V. Shoup, *On Formal Models for Secure Key Exchange*, Theory of Cryptography Library, `http://www.shoup.net/papers/skey.ps`, 1999

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.

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 AKE experiment with $\mathcal{M}$ against the NAXOS protocol and modifies the data returned by the honest parties in such a way that if $\mathcal{M}$ breaks the AKE security of NAXOS, then $\mathcal{S}$ can reveal the solution to the GDH problem from $\mathcal{M}$.

Matching Session Exists

Assume that $\mathcal{M}$ always 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). 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 $2/k^2$ $\mathcal{M}$ 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 $sk_{\mathcal{A}}$, $sk_{\mathcal{B}}$, $esk_{\mathcal{A}}$ 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 AKE experiment from a true AKE 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). However, $\mathcal{M}$ cannot do this unless he computes the discrete logarithm of either $g^{sk_{\mathcal{A}}}$ or $g^{sk_{\mathcal{B}}}$. This corresponds to a hypothetic discrete logarithm adversary $\mathcal{T}$ in the security statement.

No Matching Session

Assume now that $\mathcal{M}$ always selects a test session such that the matching session doesn't exist. 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 AKE sessions executed by $\mathcal{B}$. $\mathcal{S}$ handles AKE sessions executed by $\mathcal{B}$ (assume that $\mathcal{B}$ is the initiator). $\mathcal{S}$ 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 $H_2(CDH(X_0, comm_{\mathcal{C}}), pk_{\mathcal{C}}^h, comm_{\mathcal{C}}^h, \mathcal{B}, \mathcal{C}))$ 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.

Note that if $\mathcal{C}$ is a fictitious 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 $\mathcal{M}$ 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 equal any previous response from the random oracle.

$\mathcal{M}$ cannot detect that it is in the simulated AKE experiment unless it either queries $(esk_{\mathcal{B}}, DLOG(X_0))$ to H_1 (by doing this, $\mathcal{M}$ can detect that $comm_{\mathcal{B}}$ is computed incorrectly) or tries to reveal a long-term secret key of $\mathcal{B}$. The first event will reveal $DLOG(X_0)$ and will clearly allow $\mathcal{S}$ to solve the CDH problem. As we note below, the second event is impossible if $\mathcal{S}$ correctly guesses the test session.

Now $\mathcal{S}$ also randomly selects an AKE 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 AKE experiment from a true AKE experiment is if M queries $(sk_{\mathcal{A}}, esk_{\mathcal{A}})$ to H_1. However, $\mathcal{M}$ cannot do this unless he computes the discrete logarithm of $g^{sk_{\mathcal{A}}}$.

Analysis

We observe that the running time of $\mathcal{S}$ is $O(kt)$. For each session key computation done by $\mathcal{B}$ (let Y be 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 (let Y the incoming communication in that session) $\mathcal{S}$ checks if $\mathrm{DDH}(X_0, Y, Z)$. Since $\mathcal{M}$ can activate at most k sessions and make at most t H_2 queries, 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 $O(t \log t)$.