Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels*

Ran Canetti[1] and Hugo Krawczyk[2,**]

[1] IBM T.J. Watson Research Center, Yorktown Heights, New York 10598.
canetti@watson.ibm.com
[2] EE Department, Technion, Haifa, Israel.
hugo@ee.technion.ac.il

Abstract. We present a formalism for the analysis of key-exchange protocols that combines previous definitional approaches and results in a definition of security that enjoys some important analytical benefits: (i) any key-exchange protocol that satisfies the security definition can be composed with symmetric encryption and authentication functions to provide provably secure communication channels (as defined here); and (ii) the definition allows for simple modular proofs of security: one can design and prove security of key-exchange protocols in an idealized model where the communication links are perfectly authenticated, and then translate them using general tools to obtain security in the realistic setting of adversary-controlled links.
We exemplify the usability of our results by applying them to obtain the proof of two classes of key-exchange protocols, Diffie-Hellman and key-transport, authenticated via symmetric or asymmetric techniques.

1 Introduction

Key-exchange protocols (KE, for short) are mechanisms by which two parties that communicate over an adversarially-controlled network can generate a common secret key. KE protocols are essential for enabling the use of shared-key cryptography to protect transmitted data over insecure networks. As such they are a central piece for building secure communications (a.k.a "secure channels"), and are among the most commonly used cryptographic protocols (contemporary examples include SSL, IPSec, SSH, among others).

The design and analysis of secure KE protocols has proved to be a non-trivial task, with a large body of work written on the topic, including [15,30,10,7,16,5,6], [26,2,34] and many more. In fact, even today, after two decades of research, some important issues remain without satisfactory treatment. One such issue is how to guarantee the adequacy of KE protocols for their most basic application: the generation of shared keys for implementing secure channels. Providing this

* This proceedings version is a condensed high-level outline of the results in this work; for a complete self-contained treatment the reader is referred to [13].

** Supported by Irwin and Bethea Green & Detroit Chapter Career Development Chair.

B. Pfitzmann (Ed.): EUROCRYPT 2001, LNCS 2045, pp. 453–474, 2001.

guarantee (with minimal requirements from KE protocols) is the main focus and objective of this work. The other central goal of the paper is in simplifying the usability of the resultant security definitions via a modular approach to the design and analysis of KE protocols. We exemplify this approach with a proof of security for two important classes of KE protocols.

This paper adopts a methodology for the analysis of KE protocols that results from the combination of two previous works in this area: Bellare and Rogaway [5] and Bellare, Canetti and Krawczyk [2]. A main ingredient in the formalization of [5] is the use of the indistinguishability approach of [20] to defining security: roughly speaking, a key-exchange protocol is called secure if under the allowed adversarial actions it is infeasible for the attacker to distinguish the value of a key generated by the protocol from an independent random value. Here we follow this exact same approach but replace the adversarial model of [5] with an adversarial model derived from [2]. This combination allows to achieve the above two main objectives. We elaborate on these main aspects of our work.

First, the formalization of [2] captures not only the specific needs of KE protocols but rather develops a more general model for the analysis of security protocols. This allows formulating and proving the statement that KE protocols proven secure according to our definition (we call these protocols SK-secure) can be used in standard ways to provide "secure channels". More specifically, consider the common security practice by which pairs of parties establish a "secure channel" by first exchanging a *session key* using a KE protocol and then using this key to encrypt and authenticate the transmitted data under symmetric cryptographic functions. We prove that if in this setting one uses an SK-secure KE protocol together with secure MAC and encryption functions combined appropriately then the resultant channel provides both authentication and secrecy (in a sense that we define precisely) to the transmitted data. While this property of ensuring secure channels is indeed an obvious requirement from a secure KE protocol it turns out that formalizing and proving this property is non-trivial. In fact, there are "seemingly secure" key exchange protocols that do not necessarily guarantee this (e.g. those that use the session key during the exchange itself), as well as proposed definitions of secure key-exchange that do not suffice to guarantee this either (e.g., the definitions in [5,8,9,2]). Moreover, although several works have addressed this issue (see Section 1.1), to the best of our knowledge the notion of secure channels was never formalized in the context of KE protocols, let alone demonstrating that some definition of KE protocols suffices for this basic task. Indeed, one of the contributions of this work is a formalization of the secure channels task. While this formalization is not intended to provide general composability properties for arbitrary cryptographic settings, it arguably provides sufficient security guarantee for the central task of protecting the integrity and authenticity of communications over adversarially-controlled links.

Second, the approach of [2] allows for a substantial simplification in designing KE protocols and proving their security. This approach postulates a two-step methodology by which protocols can first be designed and analyzed in a much

simplified adversarial setting where the communication links are assumed to be ideally authenticated (i.e., the attacker is not allowed to insert or change information transmitted over the communication links between parties). Then, in a second step, these protocols are "automatically" transformed into secure protocols in the realistic scenario of fully adversary-controlled communications by applying a protocol translation tool (or "compiler") called an *authenticator.* Fortunately, simple and efficient realizations of authenticators based on different cryptographic functions exist [2] thus making it a useful and practical design and analysis tool. (We stress that our framework does not *mandate* this methodology; i.e., it is possible of course to prove security of a KE protocol directly in the fully adversarial model.)

We use this approach to prove the security of two important classes of key-exchange protocols: Diffie-Hellman and key-transport protocols. All one needs to do is to simply prove the security of these protocols in the ideal authenticated-links model and then, thanks to the above modular approach, one obtains versions of these protocols that are secure in a realistic adversary-controlled network. The "authenticated" versions of the protocols depend on the authenticators in use. These can be based either on symmetric or asymmetric cryptographic techniques (depending on the trust model) and result in natural and practical KE protocols. The security guarantees that result from these proofs are substantial as they capture many of the security concerns in real communications settings including the asynchronous nature of contemporary networks, the run of multiple simultaneous sessions, resistance to man-in-the-middle and known-key attacks, maintaining security of sessions even when other sessions are compromised, and providing "perfect forward secrecy", i.e., protection of past sessions in case of the compromise of long-term keying material.

1.1 Related Work

Since its introduction in the seminal work of Diffie and Hellman [15] the notion of a key-exchange protocol has been the subject of many works (see [28] for an extensive bibliography). Here we mention some of the works that are more directly related to the present work.

Among the early works on this subject we note [30,10,7,16] as being instrumental in pointing out to the many subtleties involved in the analysis of KE protocols. The first complexity-theoretic treatment of the notion of security for KE protocols is due to Bellare and Rogaway [5] who formalize the security of KE protocols in the realistic setting of concurrent sessions running in an adversary-controlled network. As said above, [5] apply the indistinguishability definitional approach that we follow here as well. While [5] focused on the shared-key model of authentication, other works [8,9,4] extended the techniques to the public-key setting. One important contribution of [4] is in noting and fixing a shortcoming in the original definition of [5]; this fix, that we adopt here, is fundamental for proving our results about secure channels.

Bellare, Canetti, and Krawczyk [2] present a model for studying general session-oriented security protocols that we adopt and extend here. They also introduce the "authenticator" techniques that allow for greatly simplifying the analysis of protocols and that we use as a basic tool in our work. In addition, [2] proposes a definition of security of KE protocols rooted in the simulatability (or "ideal third party") approach used to define security of multiparty computation [19,29,1,11]. While this definitional approach is intuitively appealing the actual KE security definition of [2] comes short of the expectations. On one hand, it seems over-restrictive, in the sense that it rules out protocols that seem to provide sufficient security (and as demonstrated here can be safely used to obtain secure channels); on the other, it is not clear whether their definition suffices to prove composition theorems even in the restricted sense of secure channels as dealt with in this paper.

More recently, Shoup [34] presents a framework for the definition of security of KE protocols that follows the basic simulatability approach as in [2] but introduces significant modifications in order to overcome some of the shortcomings of the KE definition in [2] as well as to seek composability with other cryptographic applications. In particular, [34] states as a motivational goal the construction of "secure sessions" (similar to our secure channels), and it informally claims the sufficiency of its definitions to achieve this goal. A more rigorous and complete elaboration of that work will be needed to assess the correctness of these claims. In addition, [34] differs from our work in several other interesting aspects. In order to keep this introduction short, we provide a more extensive comparison with [34] in Appendix A.

A promising general approach for the analysis of reactive protocols and their concurrent composition has been developed by Pfitzmann, Schunter and Waidner [32,31,33] and Canetti [12]. This approach, that follows the simulatability tradition, can be applied to the task of key exchange to obtain a definition of KE protocols that guarantees secure concurrent composition with any set of protocols that make use of the generated keys. See more details in [14].

A subjective discussion. The above works follow two main distinct approaches to defining security of KE protocols: *simulation-based* and *indistinguishability-based.* The former is more intuitively appealing (due to its modeling of security via an ideally-trusted third party), and also appears to be more amenable to demonstrating general composability properties of protocols. On the other hand, the complexity of the resulting definitions, once all details are filled in, is considerable and makes for definitions that are relatively complex to work with. In contrast, the indistinguishability-based approach yields definitions that are simpler to state and easier to work with, however their adequacy for modeling the task at hand seems less clear at first glance. The results in this paper indicate the suitability of the indistinguishability-based approach in the context of KE protocols — if the goal is the application of KE protocols to the specific task of secure channels as defined here. By following this approach we gain the benefit of simpler analysis and easier-to-write proofs of security. At the same time, our work borrows from the simulation-based approach the modu-

larity of building proofs via the intermediate ideally-authenticated links model, thus enjoying the "best of both worlds".

Organization. Due to lack of space, the presentation here is kept at an informal level, and omits some important pieces. A complete and rigorous treatment, including model details and proofs, appears in [13]. Section 2 presents an overview of the protocol and adversary models used throughout this work. The definition of SK-security for KE protocols is presented in Section 3. Section 4 states the security of sample protocols. Section 5 demonstrates the suitability of our notion of security to realizing secure channels.

2 Protocol and Adversary Models: An Overview

In order to to define what is meant by the security of a key-exchange (KE) protocol we first need to establish a formalism for the most basic notions: what is meant by a protocol in general and a key-exchange protocol in particular, what are sessions, and what is an 'attacker' against such protocols. Here we use a formalism based on the approach of [2], where a general framework for studying the security of session-based multi-party protocols over insecure channels is introduced. We extend and refine this formalism to better fit the needs of practical KE protocols.

In order to motivate and make the formalism easier to understand, *we start by providing a high-level overview of our model.* The precise technical description appears in [13]. After introducing the protocol and adversary models we proceed to define the security of KE protocols in Section 3.

2.1 Protocols, Sessions and Key-Exchange

Message-driven protocols. We consider a set of parties (probabilistic polynomial-time machines), which we usually denote by $P_1, \ldots, P_n$, interconnected by point-to-point links over which messages can be exchanged. Protocols are collections of interactive procedures, run concurrently by these parties, that specify a particular processing of incoming messages and the generation of outgoing messages. Protocols are initially triggered at a party by an external "call" and later by the arrival of messages. Upon each of these events, and according to the protocol specification, the protocol processes information and may generate and transmit a message and/or wait for the next message to arrive. We call these message-driven protocols. (We note the asynchronous nature of protocols defined in this way which reflects the prevalent form of communication in today's networks.)

Sessions and protocol output. Protocols can trigger the initiation of subprotocols (i.e. interactive subroutines) or other protocols, and several copies of such protocols may be simultaneously run by each party. We call each copy of a protocol run at a party a session. Technically, a session is an interactive subroutine executed inside a party. Each session is identified by the party that runs

it, the parties with whom the session communicates and by a session-identifier. These identifiers are used in practice to bind transmitted messages to their corresponding sessions. Each invocation of a protocol (or session) at a given party creates a local state for that session during execution, and produces local outputs by that party. This output can be a quantity (e.g a session key) returned to the calling program, or it can be the recording of a protocol event (such as a successful or failed termination). These local outputs serve to represent the "history" of a protocol and are important to formalize security. When a session ends its run we call it complete and assume that its local state is erased.

Key-exchange protocols. Key-exchange (KE) protocols are message-driven protocols (as defined above) where the communication takes place between pairs of parties and which return, upon completion, a secret key called a session key. More specifically, the input to a KE protocol within each party P_i is of the form $(P_i, P_j, s, role)$, where P_j is the identity of another party, s is a session id, and *role* can be either initiator or responder. A session within P_i and a session within P_j are called matching if their inputs are of the form $(P_i, P_j, s, \text{initiator})$ and $(P_j, P_i, s, \text{responder})$. The inputs are chosen by a "higher layer" protocol that "calls" the KE protocol. We require the calling protocol to make sure that the session id's of no two KE sessions in which the party participates are identical. Furthermore, we leave it to the calling protocol to make sure that two parties that wish to exchange a key will activate matching sessions. Note that this may require some communication before the actual KE sessions are activated.[1]

Upon activation, the partners P_i and P_j of two matching sessions exchange messages (the initiator goes first), and eventually generate local outputs that include the name of the partners of the session, the session identifier, and the value of the computed session key. A key establishment event is recorded only when the exchange is completed (this signals, in particular, that the exchanged key can be used by the protocol that called the KE session). We note that a session can be completed at one partner but not necessarily at the other.

After describing these 'mechanics" of a KE protocol we need to define what is meant by a "secure" KE protocol. This is the subject of Section 3 and it is based on the adversarial model that we introduce next.

2.2 The Unauthenticated-Links Adversarial Model (UM)

In order to talk about the security of a protocol we need to define the adversarial setting that determines the capabilities and possible actions of the attacker. We want these capabilities to be as generic as possible (as opposed to, say, merely representing a list of possible attacks) while not posing unrealistic requirements. We follow the general adversarial formalism of [2] but specialize and extend it

[1] Indeed, in practice protocols for setting up a secure session typically exchange some messages before the actual cryptographic key-exchange starts. The IKE protocol of the IPSEC standard is a good example [23].

here for the case of KE protocols. Using the terminology of [2] we call this model the Unauthenticated Links Model (UM).

Basic attacker capabilities. We consider a probabilistic polynomial-time (PPT) attacker that has full control of the communications links: it can listen to all the transmitted information, decide what messages will reach their destination and when, change these messages at will or inject its own generated messages. The formalism represents this ability of the attacker by letting the attacker be the one in charge of passing messages from one party to another. The attacker also controls the scheduling of all protocol events including the initiation of protocols and message delivery.

Obtaining secret information. In addition to these basic adversarial capabilities (given "for free" to the attacker), we let the attacker obtain secret information stored in the parties memories via explicit attacks. We consider all the secret information stored at a party as potentially vulnerable to break-ins or other forms of leakage. However, when defining security of a protocol it is important to guarantee that the leakage of some form of secret information has the least possible effect on the security of other secrets. For example, we will want to guarantee that the leakage of information specific to one session (such as the leakage of a session key or ephemeral state information) will have no effects on the security of other sessions, or that even the leakage of crucial long-term secrets (such as private keys) that are used across multiple sessions will not necessarily compromise secret information from all past sessions. In order to be able to differentiate between various vulnerabilities and to be able to guarantee as much security as possible in the event of information exposures, we classify attacks into three categories depending on the type of information accessed by the adversary:

Session-state reveal. The attacker provides the name of a party and a session identifier of a *yet incomplete* session at that party and receives the internal state of that session (since we see sessions as procedures running inside a party then the internal state of a session is well defined). An important point here is what information is included in the local state of a session. We leave this to be specified by each KE protocol. Therefore, our definition of security is parameterized by the type and amount of information revealed in this attack. For instance, the information revealed in this way may be the exponent x used by a party to compute a value g^x in a Diffie-Hellman key-exchange protocol, or the random bits used to encrypt a quantity under a probabilistic encryption scheme during a session. Typically, the revealed information will include all the local state of the session and its subroutines, except for the local state of the subroutines that directly access the long-term secret information, e.g. the local signature/decryption key of a public-key cryptosystem, or the long-term shared key.

Session-key query. The attacker provides a party's name and a session identifier of a *completed* session at that party and receives the value of the key generated by the named session This attack provides the formal modeling for leakage of information on specific session keys that may result from events such as break-

ins, cryptanalysis, careless disposal of keys, etc. It will also serve, indirectly, to ensure that the unavoidable leakage of information produced by the use of session keys in a security application (e.g., information leaked on a key by its use as an encryption key) will not help in deriving further information on this and other keys.

Party corruption. The attacker can decide at any point to corrupt a party, in which case the attacker learns *all* the internal memory of that party including long-term secrets (such as private keys or master shared keys used across different sessions) and session-specific information contained in the party's memory (such as internal state of incomplete sessions and session-keys corresponding to completed sessions). Since by learning its long term secrets the attacker can impersonate a party in all all its actions then a party is considered completely controlled by the attacker from the time of corruption and can, in particular, depart arbitrarily from the protocol specifications.

Terminology: if a session is subject to any of the above three attacks (i.e. a session-state reveal, a session-key query or the corruption of the party holding the session) then the session is called locally exposed. If a session or its matching session is locally exposed then we call the session exposed.

Session expiration. One important additional element in our security model is the notion of session expiration. This takes the form of a protocol action that when activated causes the erasure of the named session key (and any related session state) from that party's memory. We allow a session to be expired at one party without necessarily expiring the matching session. The effect of this action in our security model is that the value of an expired session key cannot be found via any of the above attacks if these attacks are performed after the session expired. This has two important consequences: it allows us to model the common (and good) security practice of limiting the life-time of individual session keys and it allows for a simple modeling of the notion of perfect forward secrecy (see Section 3.2). We note that in order for a session to be locally exposed (as defined above) the attack against the session must happen *before* the session expires.

Bootstrapping the security of key-exchange protocols. Key-exchange protocols, as other cryptographic applications, require the bootstrapping of security (especially for authentication) via some assumed-secure means. Examples include the secure generation of parties' private keys, the installation of public keys of other parties, or the installation of shared "master" keys. Here too we follow the approach of [2] where the bootstrapping of the authentication functions is abstracted into an initialization function that is run prior to the initiation of any key-exchange protocol and that produces in a secure way (i.e. without adversarial participation) the required (long-term) information. By abstracting out this initial phase we allow for the combination of different protocols with different initialization functions: in particular, it allows our analysis of protocols (such as Diffie-Hellman) to be applicable under the two prevalent settings of authentication: symmetric and a-symmetric authentication. Two points to note are (1) the specification of the initialization function is part of the definition

of each KE protocol; and (2) secret information generated by this function at a given party can be discovered by the attacker only upon corruption of that party. We stress that while this abstraction adds to the simplicity and applicability of our analysis techniques, the bootstrapping of security in actual protocols is an element that must be carefully analyzed (e.g., the interaction with a CA in the case of public-key based protocols). Integrating these explicit elements into the model can be done either directly as done in [34], or in a more modular way via appropriate protocol composition.

2.3 The AM, Protocol Emulation and Authenticators

A central ingredient in our analyses is the methodology introduced in [2] by which one can design and analyze a protocol under the highly-simplifying assumption that the attacker cannot change information transmitted between parties, and then transform these protocols and their security assurance to the realistic UM where the adversary has full control of the communication links. The main components in the formalization of this methodology are shortly described here (see [2,13] for complete details).

First, an adversarial model called authenticated-links model (denoted AM) is defined in a way that is identical to the UM with one fundamental difference: *the attacker is restricted to only deliver messages truly generated by the parties without any change or addition to them.* Then, the notion of "emulation" is introduced in order to capture the equivalence of functionality between protocols in different adversarial models, in particular between the UM and AM. Roughly speaking, a protocol π' emulates protocol π in the UM if for any adversary that interacts with π' in the UM there exists an adversary that interacts with π in the AM such that the two interactions "look the same" to an outside observer. Finally, special algorithms called authenticators are developed with the property that on input the description of a protocol π the authenticator outputs the description of a protocol π' such that π' emulates protocol π in the UM. That is, authenticators act as an automatic "compiler" that translate protocols in the AM into equivalent (or "as secure as") protocols in the UM.

In order to simplify the construction of authenticators, [2] offers the following methodology. First consider a very simple one-flow protocol in the AM, called MT, whose sole functionality is to transmit a single message from sender to recipient. Now build a restricted-type authenticator, called MT-authenticator, required to provide emulation for this particular MT protocol only. Finally, to any such MT-authenticator λ one associates an algorithm (or compiler) C_λ that translates any input protocol π into another protocol π' as follows: to each of the messages defined in protocol π apply the MT-authenticator λ. It is proven in [2] that C_λ is an authenticator (i.e., the resultant protocol π' emulates π in the UM). Particular realizations of MT-authenticators are presented in [2] based on different type of cryptographic functions (e.g., digital signatures, public-key encryption, MAC, etc.)

3 Session-Key Security

After having defined the basic formal model for key-exchange protocols and adversarial capabilities, we proceed to define *what is meant for a key-exchange protocol to be secure*. While the previous section was largely based on the work of [2], our definition of security closely follows the definitional approach of [5]. The resultant notion of security, that we call *session-key security* (or *SK-security*), focuses on ensuring the security of individual session-keys as long as the session-key value is not obtained by the attacker via an explicit key exposure (i.e. as long as the session is *unexposed* – see the terminology in the previous section). We want to capture the idea that the attacker "does not learn anything about the value of the key" from interacting with the key-exchange protocol and attacking other sessions and parties. As it is standard in the semantic-security approach this is formalized via the infeasibility to distinguish between the real value of the key and an independent random value.

We stress that this formulation of SK-security is very careful about tuning the definition to offer enough strength as required for the use of key-exchange protocols to realize secure channels (Section 5), as well as being realistic enough to avoid over-kill requirements which would prevent us from proving the security of very useful protocols (Section 4). We further discuss these aspects after the presentation of the definition.

3.1 Definition of SK-Security

We first present the definition for the UM. The formalization in the AM is analogous. We start by defining an "experiment" where the attacker $\mathcal{U}$ chooses a session in which to be "tested" about information it learned on the session-key; specifically, we will ask the attacker to differentiate the real value of the chosen session key from a random value. (Note that this experiment is an artifact of the definition of security, and not an integral part of the actual key-exchange protocols and adversarial intervention.)

For the sake of this experiment we extend the usual capabilities of the adversary, $\mathcal{U}$, in the UM by allowing it to perform a *test-session query*. That is, in addition to the regular actions of $\mathcal{U}$ against a key-exchange protocol π, we let $\mathcal{U}$ to choose, at any time during its run, a *test-session* among the sessions that are completed, unexpired and unexposed at the time. Let κ be the value of the corresponding session-key. We toss a coin b, $b \stackrel{R}{\leftarrow} \{0,1\}$. If $b = 0$ we provide $\mathcal{U}$ with the value κ. Otherwise we provide $\mathcal{U}$ with a value r randomly chosen from the probability distribution of keys generated by protocol π. The attacker $\mathcal{U}$ is now allowed to continue with the regular actions of a UM-adversary but *is not allowed to expose the test-session* (namely, it is not allowed session-state reveals, session-key queries, or partner's corruption on the test-session or its matching

session.[2]) At the end of its run, $\mathcal{U}$ outputs a bit b' (as its guess for b).
We will refer to an attacker that is allowed test-session queries as a KE-adversary.

Definition 1. *A* KE *protocol* π *is called* SK-secure *if the following properties hold for any* KE*-adversary* $\mathcal{U}$ *in the* UM.

1. *Protocol* π *satisfies the property that if two* uncorrupted *parties complete matching sessions then they both output the same key; and*
2. *the probability that* $\mathcal{U}$ *guesses correctly the bit* b *(i.e., outputs* $b' = b$*) is no more than 1/2 plus a negligible fraction in the security parameter.*

If the above properties are satisfied for all KE*-adversaries in the* AM *then we say that* π *is* SK-secure in the AM.

The first condition is a "consistency" requirement for sessions completed by two *uncorrupted* parties. We have no requirement on the session-key value of a session where one of the partners was corrupted before the session completed – in fact, most KE protocols allow a corrupted party to strongly influence the exchanged key. The second condition is the "core property" for SK-security. We note that the term 'negligible' refers, as customary, to any function (in the security parameter) that diminishes asymptotically faster than any polynomial fraction. (This formulation allows, if so desired, to quantify security via a concrete security treatment. In this case one quantifies the attacker's power via specific bounds on computation time, number of corruptions, etc., while its advantage is bounded through a specific parameter ε.)

Discussion. We highlight three aspects of Definition 1.

- The attacker can keep running and attacking the protocol even after receiving the response (either real or random) to its test-session query. This ability (which represents a substantial strengthening of security relative to [5], see also [4]) is *essential* for proving the main property of SK-security shown in this paper, namely its guarantee of security when used to generate secure channels as described in Section 5.
- The attacker is not allowed to corrupt partners to the test-session or issue any other exposure command against that session while unexpired. This reflects the fact that there is no way to guarantee the secure use of a session-key that was exposed via an attacker's break-in (or cryptanalysis). In particular, this restriction is instrumental for proving the security of specific important protocols (e.g., Diffie-Hellman key exchange) as done in Section 4.
- The above restriction on the attacker by which it cannot corrupt a partner to the test-session is lifted as soon as the session expires at that partner. In this case the attacker should remain unable to distinguish between the real

[2] We stress, however, that the attacker *is allowed* to corrupt a partner to the test-session as soon as the test-session (or its matching session) expires at that party. See the discussion below. This may be the case even if the other partner has not yet expired the matching session or not even completed it.

value of the key from a random value. This is the basis to the guarantee of "perfect forward secrecy" provided by our definition and further discussed in Section 3.2.

We stress that in spite of its "compact" formulation Definition 1 is very powerful and can be shown to ensure many specific properties that are required from a good key-exchange protocol (see, for example, chapter 12 of [28]). Some of these properties include the guarantee that session-keys belong to the right probability distribution of keys (except if one of the partners is corrupted at time of exchange), the "authenticity" of the exchange (namely, a correct and consistent binding between keys and parties' identities), resistance to man-in-the-middle attacks (for protocols proven SK-secure in the UM), resistance to known-key attacks, forward secrecy, and more. However, we note that all these properties (which are sometimes listed as a replacement to a formal definition of security) in combination do not suffice to guarantee the most important aspect of key-exchange security that SK-security enjoys: namely, the composition of the key-exchange protocols with cryptographic functions to enable secure channels (e.g., the original definition of security in [5] does satisfy the above list of properties but is insufficient to guarantee secure channels).

We finally remark that Definition 1 makes security requirements from a KE protocol only in case that the protocol completes KE-sessions. No guarantee is made that KE-sessions will ever return, or that they will not be aborted, i.e., that the corresponding session key will not be null. (In fact, a KE protocol where all KE-sessions "hang" and never return satisfies the definition.) One can add an explicit termination requirement for sessions in which the parties are uncorrupted and all messages are correctly delivered by the attacker. For simplicity, we choose to leave the analysis of the termination properties of protocols out of the scope of the definition of security.

3.2 Forward Secrecy

Informally, the notion of "perfect forward secrecy" (PFS) [22,16] is stated as the property that "compromise of long-term keys does not compromise past session keys". In terms of our formalism this means that even if a party is corrupted (in which case all its stored secrets – short-term and long-term – become known to the attacker) then nothing is learned about sessions within that party that were previously unexposed and *expired* before the party corruption happened.

The provision that *expired* session-keys remain indistinguishable from random values even if a partner to that session is corrupted guarantees the perfect forward secrecy of SK-secure protocols. Put in other words, when proving a protocol to be SK-secure using Definition 1 one automatically gets a proof that that protocol guarantees PFS.

On the other hand, while PFS is a very important security property it is not required for all application scenarios, e.g., when only authentication is required, or when short-term secrecy suffices. Indeed, it is common to find in practice

protocols that do not provide PFS and still are not considered insecure. One such typical case are "key-transport protocols" in which public key encryption is used to communicate a session-key from one party to another. (In this case, even if session-keys are erased from memory when no longer required, the corruption of a party may allow an attacker to compute, via the discovered long-term private keys, all the past session-keys.) Due to the importance of such protocols (they are commonly used in, e.g., SSL), and given that achieving PFS usually has a non-negligible computational cost, we define a notion of "SK-security without PFS" by simply disallowing the protocol's action of key expiration. That is, under this modified model, *session-keys never expire*. This results in a weaker notion of security since now by virtue of Definition 1 the attacker is *never allowed* to corrupt a partner to the test-session (or in other words, this weaker definition of security does not guarantee the security of a session-key for which one of the partners is ever corrupted).

Definition 2. *We say that a* KE *protocol satisfies* SK-security without PFS *if it enjoys SK-security relative to any* KE*-adversary in the* UM *that is* not *allowed to expire keys. (Similarly, if the above holds for any such adversaries in the* AM *then we say that* π *is* SK-secure without PFS in the AM.*)*

4 SK-Secure Protocols

This section demonstrates the usability of our definition of SK-security for proving the security of some simple and important key-exchange protocols. One is the original Diffie-Hellman protocol, the other is a simple "key transport" protocol based on public-key encryption. We first show that these protocols are secure in the simpler authenticated-links model (AM). Then, using the methodology from [2] we can apply to these protocols a variety of (symmetric or asymmetric) authentication techniques to obtain key-exchange protocols that are secure in the realistic UM model. Namely, applying any MT-authenticator (see Section 2.3) to the messages of the AM-protocol results in a secure KE protocol in the UM The next Theorem (proven in [13]) states that this methodology does work for our purposes.

Theorem 1. *Let* π *be a SK-secure key-exchange protocol in the* AM *with* PFS *(resp., without* PFS*) and let* λ *be an authenticator. Then* $\pi' = C_\lambda(\pi)$ *is a SK-secure key-exchange protocol in the* UM *with* PFS *(resp., without* PFS*).*

For lack of space we only describe here the protocol based on Diffie-Hellman exchange. The key-transport protocol based on public-key encryption is presented and analyzed in [13].

4.1 Two-Move Diffie-Hellman

We demonstrate that under the Decisional Diffie-Hellman (DDH) assumption the 'classic' two-move Diffie-Hellman key-exchange protocol designed to work

against eavesdroppers-only is SK-secure in the AM. We denote this protocol by 2DH and describe it in Figure 1. Here and in the sequel all exponentiations are modulo the defined prime p.

Protocol 2DH

Common information: Primes p, q, q/p–1, and g of order q in Z_p^*.

Step 1: The initiator, P_i, on input (P_i, P_j, s), chooses $x \stackrel{R}{\leftarrow} Z_q$ and sends $(P_i, s, \alpha = g^x)$ to P_j.

Step 2: Upon receipt of (P_i, s, α) the responder, P_j, chooses $y \stackrel{R}{\leftarrow} Z_q$, sends $(P_j, s, \beta = g^y)$ to P_i, *erases* y, and outputs the session key $\gamma = \alpha^y$ under session-id s.

Step 3: Upon receipt of (P_j, s, β), party P_i computes $\gamma' = \beta^x$, *erases* x, and outputs the session key γ' under session-id s.

Fig. 1. The two-move Diffie-Hellman protocol in the AM

Theorem 2. *Assuming the Decisional Diffie-Hellman (DDH) assumption, protocol* 2DH *is SK-secure in the* AM.

Using Theorem 1 we can apply any authenticator to this protocol to obtain a secure Diffie-Hellman exchange against realistic UM attackers. For illustration, a particular instance of such a SK-secure protocol in the UM, using digital signatures for authentication, is shown in Section 4.2. Other flavors of authenticated DH protocols can be derived in a similar way by using other authenticators (e.g. based on public key encryption or on pre-shared keys [2]).

4.2 SK-Secure Diffie-Hellman Protocol in the UM

Here we apply the signature-based authenticator of [2] to the protocol 2DH from Figure 1 to obtain a Diffie-Hellman key-exchange that is SK-secure in the UM. We present the resultant protocol in Figure 2 (it is very similar to a protocol specified in [24]). Its SK-security follows from Theorems 1 and 2.

Remarks on protocol SIG-DH**.** The protocol is the result of applying the signature-based authenticator of [2] to the 2-pass Diffie-Hellman protocol of Figure 1 where the values α and β (the DH exponentials) serve as the challenges required by the signature-based authenticator. This assumes (as specified in protocol 2DH) that these exponentials are chosen afresh for each new exchange (otherwise each party can add an explicit nonce to the messages which is also included under the signature). We note that the identity of the destination party included under the signatures is part of the specification of the signature-based authenticator of [2] and is fundamental for the security of the protocol.

The description of SIG-DH in Figure 2 assumes, as formalized in our model, that the value s of the session-id is provided to the parties. In practice, one

Protocol SIG-DH

Initial information: Primes p, q, q/p–1, and g of order q in Z_p^*. Each player has a private key for a signature algorithm SIG, and all have the public verification keys of the other players.

Step 1: The initiator, P_i, on input (P_i, P_j, s), chooses $x \stackrel{R}{\leftarrow} Z_q$ and sends $(P_i, s, \alpha = g^x)$ to P_j.

Step 2: Upon receipt of (P_i, s, α) the responder, P_j, chooses $y \stackrel{R}{\leftarrow} Z_q$, and sends to P_i the message $(P_j, s, \beta = g^y)$ together with its signature $\text{SIG}_j(P_j, s, \beta, \alpha, P_i)$; it also computes the session key $\gamma = \alpha^y$ and *erases* y.

Step 3: Upon receipt of (P_j, s, β) and P_j's signature, party P_i verifies the signature and the correctness of the values included in the signature (such as players identities, session id, the value of exponentials, etc.). If the verification succeeds then P_i sends to P_j the message $(P_i, s, \text{SIG}_j(P_i, s, \alpha, \beta, P_j))$, computes $\gamma' = \beta^x$, *erases* x, and outputs the session key γ' under session-id s.

Step 4: Upon receipt of the triple (P_i, s, sig), P_j verifies P_i's signature sig and the values it includes. If the check succeeds it outputs the session key γ under session-id s.

Fig. 2. Diffie-Hellman protocol in the UM: authentication via signatures.

usually generates the session identifier s as a pair (s_1, s_2) where s_1 is a value chosen by P_i and different (with very high probability) from all other such values chosen by P_i in his other sessions with P_j. Similarly, s_2 is chosen by P_j with an analogous uniqueness property. These values s_1, s_2 can be exchanged by the parties as a prologue to the above protocol (this may be the case of protocols that implement such a prologue to exchange some other system information and to negotiate exchange parameters; see for example [23]). Alternatively, s_1 can be included by P_i in the first message of SIG-DH, and s_2 be included by P_j in the second message. In any case, it is important that these values be included under the parties' signatures.

5 Applications to Secure Channels

It is common practice to protect end-to-end communications by letting the end parties exchange a secret session key and then use this key to authenticate and encrypt the transmitted data under symmetric cryptographic functions. In order for a key-exchange protocol to be considered secure it needs to guarantee that the above strategy for securing data works correctly, namely, that by using a shared key provided by the KE protocol one achieves sound authentication and secrecy. As it is customary, we will refer to a link between a pair of parties that achieves these properties as a secure channel. While secure channels may have different formalizations, here we restrict our treatment to the above setting of securing communications using symmetric cryptography with a key derived from a key-

exchange protocol. *We prove that an SK-secure key-exchange protocol, together with a secure MAC and symmetric encryption appropriately combined, suffices for realizing such secure channels.*

For lack of space, this extended abstract contains only our treatment of the task of authenticating the communication. Full treatment, including the task of providing secrecy (both in the AM and in the UM) appears in [13].

A Template Protocol: Network Channel. We start by formalizing a "template protocol" that captures a generic session-oriented KE-based protocol for secure channels between pairs of parties in a multi-party setting with parties $P_1, \ldots, P_n$. This template protocol, called NetChan, applies to the unauthenticated-links model UM as well as to the authenticated-links model AM. Later we will see specific implementations of this template protocol where the generic 'send' and 'receive' primitives defined there are instantiated with actual functions (e.g., for providing authentication and/or encryption). We will also define what it means for such an implementation to be "secure".

A (session-based) network channel protocol, $\mathsf{NetChan}(\pi, \mathsf{snd}, \mathsf{rcv})$, is defined on the basis of a KE protocol π, and two generic functions snd and rcv. (A more general treatment can be obtained by considering these functions as interactive protocols but we leave this more general approach beyond the scope of the present paper.) Both snd and rcv are probabilistic functions that take as arguments a session-key (we denote this key as a subscript to the function) and a message m. The functions may also depend on other session information such as a session-id and partner identifiers. The output of snd is a single value m', while the output of rcv is a pair (v, ok) where ok is a bit and v an arbitrary value. (The bit ok will be used to return a verification value, e.g. the result of verifying an authentication tag.) On the basis of such functions we define $\mathsf{NetChan}(\pi, \mathsf{snd}, \mathsf{rcv})$ in Figure 3.

Network Authentication. On the basis of the above formalism, we treat the case of network channels that provide authentication of information over adversary-controlled channels. Namely, we are interested in a NetChan protocol that runs in the unauthenticated-links model UM and yet provides authenticity of transmitted messages. This implementation of NetChan (which we call NetAut) will be aimed at capturing the practice by which communicating parties use a key-exchange protocol to establish a shared session key, and use that key to authenticate (via a message authentication function, MAC) the information exchanged during that session. Namely, if P_i and P_j share a matching session s and P_i wants to send a message m to P_j during that session then P_i transmits m together with $\mathrm{MAC}_\kappa(m)$ where κ is the corresponding session key. Thus, in this case we will instantiate the snd and rcv functions of NetChan with a MAC function as follows.

Protocol NetAut. Let π be a KE protocol and let f be a MAC function. Protocol $\mathsf{NetAut}(\pi, f)$ is protocol $\mathsf{NetChan}(\pi, \mathsf{snd}, \mathsf{rcv})$ as defined in Figure 3, where functions snd and rcv are defined as:

- On input m, $\mathsf{snd}_\kappa(m)$ produces output $m' = (m, t) = (m, f_k(m))$.

Protocol $\mathsf{NetChan}(\pi, \mathsf{snd}, \mathsf{rcv})$

$\mathsf{NetChan}(\pi, \mathsf{snd}, \mathsf{rcv})$ is initialized with the same initialization function I of the KE protocol π. It can then be invoked within a party P_i under the following activations:

1. $\mathsf{establish\text{-}session}(P_i, P_j, s, role)$: this triggers a KE-session under π within P_i with partner P_j, session-id s and $role \in \{\mathsf{initiator}, \mathsf{responder}\}$. If the KE-session completes P_i records in its local output "`established session` s `with` P_j" and stores the generated session key.
2. $\mathsf{expire\text{-}session}(P_i, P_j, s)$: P_i marks session (P_i, P_j, s) (if it exists at P_i) as expired and the session key is erased. P_i records in its local output "`session` s `with` P_j `is expired`".
3. $\mathsf{send}(P_i, P_j, s, m)$: P_i checks that session (P_i, P_j, s) has been completed and not expired, if so it computes $m' = \mathsf{snd}_\kappa(m)$, using the corresponding session key κ, sends (P_i, s, m') to P_j, and records "`sent message` m `to` P_j `within session` s" in the local output.
4. On incoming message (P_j, s, m'), P_i checks that the session (P_i, P_j, s) has been completed and not expired, if so it computes $(m, \mathsf{ok}) = \mathsf{rcv}_\kappa(m')$ under the corresponding session key κ. If $\mathsf{ok} = 1$ then P_i records "`received message` m `from` P_j `within session` s." If $\mathsf{ok} = 0$ then no further action is taken.

Fig. 3. A generic network channels protocol

– On input m', $\mathsf{rcv}_\kappa(m')$ outputs (v, ok) as follows. If m' is of the form (m, t), and the pair (m, t) passes the verification function of f under key κ, then $\mathsf{ok} = 1$ and $v = m$. Otherwise, $\mathsf{ok} = 0$ and $v = null$.

In order to simplify and shorten presentation we assume that no two send activations within a session contain the same message. One can easily implement this assumption by specifying that the sender concatenates to the message a unique message id. In the cases where we care about preventing replay of messages by the attacker (as it is usually the case when providing message authentication) then message id's need to be specified in a way that the receiver can check their uniqueness (in this case sender and receiver maintain a shared state).

Our goal is to show that if the key-exchange protocol π is SK-secure and the MAC function f is secure (against chosen-message attacks) then the resultant network channels protocol $\mathsf{NetAut}(\pi, f)$ provides authenticated transmission of information. This requirement can be formulated under the property that "any message recorded by P_i as received from P_j has been necessarily recorded as sent by P_j, except if the pertinent session is exposed". We will actually strengthen this requirement and ask that a network channels protocol provides authentication if it *emulates* (i.e. imitates) the transmission of messages in the *ideally* authenticated-links model AM. Formally, we do so using the notion of protocol emulation and the formalization (see Section 2.3) of the message transmission protocol (MT) in the AM as done in [2]. Recall that MT is a simple protocol that defines the transmission of individual messages in the AM. Here we extend the

basic definition of MT to a *session-based* message transmission protocol called SMT. By proving that the network channels protocol NetAut emulates SMT in the UM we get the assurance that transmitting messages over unauthenticated-links using NetAut is as secure as transmitting them in the presence of an attacker that is not allowed to change or inject messages over the communication links.

The SMT protocol. We extend protocol MT from [2] to fit our session-based setting in which transmitted messages are grouped into different sessions. We call the extended protocol a session-based message transmission protocol (SMT), and define it in Figure 4. Note the structural similarity between SMT and NetChan – the differences are that no actual key-exchange is run in SMT, and the functions snd and rcv are instantiated to simple "identity functions".

Protocol SMT

Protocol SMT can be invoked within a party P_i under the following activations:

1. establish-session(P_i, P_j, s): in this case P_i records in its local output "`established session` s `with` P_j".
2. expire-session(P_i, P_j, s): in this case P_i records in its local output "`session` s `with` P_j `is expired`".
3. send(P_i, P_j, s, m): in this case P_i checks that session (P_i, P_j, s) has been established and not expired, if so it sends message m to P_j together with the session-id s (i.e., the values m and s are sent over the ideally-authenticated link between P_i and P_j); P_i records in its local output "`sent message` m `to` P_j `within session` s".
4. On incoming message (m, s) received over its link from P_j, P_i checks that session (P_i, P_j, s) is established and not expired, if so it records in the local output "`received message` m `from` P_j `within session` s".

Fig. 4. SMT: The session-based MT protocol in the AM.

Protocol SMT represents a perfectly authenticated exchange of messages. An implementation of protocol NetChan is said to be a secure network authentication protocol if it *emulates* (see Section 2.3) protocol SMT in the UM.

Definition 3. *Protocol* NetChan$(\pi, \mathsf{snd}, \mathsf{rcv})$ *is called a* secure network authentication protocol *if it emulates protocol* SMT *in the* UM.

The following theorem is proven in [13]:

Theorem 3. *If π is a SK-secure key-exchange protocol in the* UM *and* snd, rcv *are based as described above on a MAC function f that is secure against chosen message attacks, then protocol* NetAut$(\pi, \mathsf{snd}, \mathsf{rcv})$ *is a secure network authentication protocol.*

Network Encryption and Secure Channels Protocols. For lack of space, we omit from this extended abstract two basic components in our work (the

complete treatment appears in [13]). One is the formalization of a network encryption protocol and its security, the other is the definition of a secure channels protocol. These formalizations are based, as in the case of the network authentication protocol, on the above generic network channels template. In the case of the network encryption protocol, security (in the sense of secrecy) is formulated following the indistinguishability approach. Secure channels are then defined as network channel protocols that are simultaneously secure network authentication and secure network encryption protocols. Implementations of such secure protocols are presented using SK-secure key-exchange protocols and secure encryption and authentication functions. One particularly interesting aspect of our work is highlighted by recent results in [25] where it is demonstrated that the specific ordering of encryption and authentication as applied here is instrumental for achieving secure channels (if one assumes the standard strength, i.e. against chosen-plaintext attacks, of the encryption function). As it turns out other common orderings of these functions do not guarantee secure channels in this case (even if the KE protocol in use is secure).

References

1. D. Beaver, "Secure Multi-party Protocols and Zero-Knowledge Proof Systems Tolerating a Faulty Minority", J. Cryptology (1991) 4: 75-122.
2. M. Bellare, R. Canetti and H. Krawczyk, "A modular approach to the design and analysis of authentication and key-exchange protocols", *30th STOC*, 1998.
3. M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway, "Relations Among Notions of Security for Public-Key Encryption Schemes", *Advances in Cryptology - CRYPTO'98 Proceedings*, Lecture Notes in Computer Science Vol. 1462, H. Krawczyk, ed., Springer-Verlag, 1998, pp. 26–45.
4. M. Bellare, E. Petrank, C. Rackoff and P. Rogaway, "Authenticated key exchange in the public key model," manuscript 1995–96.
5. M. Bellare and P. Rogaway, "Entity authentication and key distribution", *Advances in Cryptology, - CRYPTO'93*, Lecture Notes in Computer Science Vol. 773, D. Stinson ed, Springer-Verlag, 1994, pp. 232-249.
6. M. Bellare and P. Rogaway, "Provably secure session key distribution– the three party case," *Annual Symposium on the Theory of Computing (STOC)*, 1995.
7. R. Bird, I. Gopal, A. Herzberg, P. Janson, S. Kutten, R. Molva and M. Yung, "Systematic design of two-party authentication protocols," *IEEE Journal on Selected Areas in Communications* (special issue on Secure Communications), 11(5):679–693, June 1993. (Preliminary version: Crypto'91.)
8. S. Blake-Wilson, D. Johnson and A. Menezes, "Key exchange protocols and their security analysis," *Proceedings of the sixth IMA International Conference on Cryptography and Coding*, 1997.
9. S. Blake-Wilson and A. Menezes, "Entity authentication and key transport protocols employing asymmetric techniques", *Security Protocols Workshop*, 1997.
10. M. Burrows, M. Abadi and R. Needham, "A logic for authentication," DEC Systems Research Center Technical Report 39, February 1990. Earlier versions in *Proceedings of the Second Conference on Theoretical Aspects of Reasoning about Knowledge*, 1988, and *Proceedings of the Twelfth ACM Symposium on Operating Systems Principles*, 1989.

11. R. Canetti, "Security and Composition of Multiparty Cryptographic Protocols", *Journal of Cryptology,* Vol. 13, No. 1, 2000.
12. R. Canetti, "A unified framework for analyzing security of Protocols", manuscript, 2000. Available at http://eprint.iacr.org/2000/067.
13. R. Canetti and H. Krawczyk, "Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels (Full Version)", http://eprint.iacr.org/2001.
14. R. Canetti and H. Krawczyk, "Proving secure composition of key-exchange protocols with any application", in preparation.
15. W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Trans. Info. Theory* IT-22, November 1976, pp. 644–654.
16. W. Diffie, P. van Oorschot and M. Wiener, "Authentication and authenticated key exchanges", *Designs, Codes and Cryptography*, 2, 1992, pp. 107–125.
17. O. Goldreich, *"Foundations of Cryptography (Fragments of a book)"*, Weizmann Inst. of Science, 1995. (Available at `http://philby.ucsd.edu/cryptolib.html`)
18. O. Goldreich, S. Goldwasser and S. Micali, "How to construct random functions," *Journal of the ACM,* Vol. 33, No. 4, 210–217, (1986).
19. S. Goldwasser, and L. Levin, "Fair Computation of General Functions in Presence of Immoral Majority", *CRYPTO '90, LNCS 537,* Springer-Verlag, 1990.
20. S. Goldwasser and S. Micali, Probabilistic encryption, *JCSS,* Vol. 28, No 2, April 1984, pp. 270-299.
21. S. Goldwasser, S. Micali and C. Rackoff, "The Knowledge Complexity of Interactive Proof Systems", *SIAM Journal on Comput.,* Vol. 18, No. 1, 1989, pp. 186-208.
22. C.G. Günther, "An identity-based key-exchange protocol", *Advances in Cryptology - EUROCRYPT'89*, Lecture Notes in Computer Science Vol. 434, Springer-Verlag, 1990, pp. 29-37.
23. D. Harkins and D. Carrel, ed., "The Internet Key Exchange (IKE)", *RFC 2409,* November 1998.
24. ISO/IEC IS 9798-3, "Entity authentication mechanisms — Part 3: Entity authentication using asymmetric techniques", 1993.
25. H. Krawczyk, "The order of encryption and authentication for protecting communications (Or: how secure is SSL?)", manuscript.
26. H. Krawczyk, "SKEME: A Versatile Secure Key Exchange Mechanism for Internet,", *Proceedings of the 1996 Internet Society Symposium on Network and Distributed System Security*, Feb. 1996, pp. 114-127.
27. P. Lincoln, J. Mitchell, M. Mitchell, A. Schedrov, "A Probabilistic Poly-time Framework for Protocol Analysis", *5th ACM Conf. on Computer and System Security*, 1998.
28. A. Menezes, P. Van Oorschot and S. Vanstone, "Handbook of Applied Cryptography," CRC Press, 1996.
29. S. Micali and P. Rogaway, "Secure Computation", unpublished manuscript, 1992. Preliminary version in *CRYPTO 91.*
30. R. Needham and M. Schroeder, "Using encryption for authentication in large networks of computers," *Communications of the ACM*, Vol. 21, No. 12, December 1978, pp. 993–999.
31. B. Pfitzmann, M. Schunter and M. Waidner, "Secure Reactive Systems", IBM Research Report RZ 3206 (#93252), IBM Research, Zurich, May 2000.
32. B. Pfitzmann and M. Waidner, "A General Framework for Formal Notions of 'Secure' System", Hildesheimer Informatik-Berichte 11/94 Institut für Informatik, Universität Hildesheim, April 1994.

33. B. Pfitzmann and M. Waidner, "A model for asynchronous reactive systems and its application to secure message transmission", IBM Research Report RZ 3304 (#93350), IBM Research, Zurich, December 2000.
34. V. Shoup, "On Formal Models for Secure Key Exchange", Theory of Cryptography Library, 1999. Available at: http://philby.ucsd.edu/cryptolib/1999/99-12.html.

A A Comparison with [34]

Section 1.1 mentioned the work by Shoup [34] on definitions and analysis of KE protocols. This appendix further expands on some of the differences between that work and ours.

We start with a short summary of the relevant parts of [34]. Shoup's definitions are based on the simulatability approach of [2] with some significant modifications. Three levels of security are presented: *Static security* (i.e., security against adversaries that corrupt parties only at the onset of the computation), *adaptive security* (where the adversary obtains only the long-term information of a newly corrupted party) and *strongly adaptive security* where the adversary obtains all the private information of corrupted parties. (Oddly, strongly adaptive security does not imply adaptive security.) In addition, two definitions based on the indistinguishability approach of Bellare and Rogaway [5] are presented. The first is aimed at capturing security without perfect forward secrecy (PFS), and is shown to be equivalent to the static variant of the simulation-based definition. The second is aimed at capturing security with PFS, and is claimed to be equivalent to the adaptive variant of the simulation-based definition. Sufficiency of the definitions to constructing secure-channel protocols is informally argued, but is not proved nor rigorously claimed.

While the first variant of the indistinguishability-based definition is roughly equivalent to the non-PFS variant presented here (modulo the general differences mentioned below), the second variant is strictly weaker than our PFS formulation of SK-security. Specifically, the definition in [34] accepts as secure protocols that do not erase sensitive ephemeral data (e.g. protocol DHKE-1 in [34]), while the definition here treats these protocols as insecure.

There are several other technical and methodological differences between the two works that we mention next. (a) A major methodological difference is our use of the authenticated-links model and authenticators as a simplifying analysis tool. While our formalization of security does not mandate the use of this methodology we carefully build our definitions to accomodate the use of this tool. (b) Shoup allows the adversary a more general attack than session-key query, namely an *application attack* that reveals an arbitrary function of the key. Our modeling does not define this explicit attack as it turns out to be unnecessary for guaranteeing secure channels. (c) Here we consider an additional adversarial behavior that is not treated in [34]. Specifically, we protect against adversaries that obtain the internal state of corrupted sessions (even without fully corrupting the corresponding parties) by requiring that such exposure will not compromise

other protocol sessions run by the same parties. This protection is not guaranteed by some protocols suggested in [34] (e.g., protocol DHKE). (d) The treatment of the interaction with the certificate authority (CA). In [34] the interaction with the CA is an integral part of every KE protocol, whereas here this interaction with the CA is treated as a separate protocol. We make this choice for further modularity and ease of proof. Yet, as we already remarked in Section 2.2, the CA protocol needs to be taken into consideration with any full specification and analysis of actual KE protocols. (e) The treatment of the session-id's. In [34] the session-id's are artificially given to the parties by the model which results, in our view, in a more cumbersome formalization of the security conditions. In contrast, here we adopt a more natural approach where the session-id's are generated by the calling protocol and security is guaranteed only when these session-id's satisfy some minimal (and easy to implement) conditions. In particular, this formalism can be satisfied by letting the parties jointly generate the session-id (as is common in practice).

Overall, we believe that the approaches in this work and in [34] are not "mutually exclusive" and both can be useful depending on a particular setting or even taste. However, for [34] to be truly useful, and for a full comparison and assessment to be possible, many of the missing definition and proof details in that work will need to be completed. Especially, rigorous proofs of protocols and a definition of secure channels is needed to assess the sufficiency of these protocols for providing the basic secure-channels functionality.