Universally Composable Notions of Key Exchange and Secure Channels[⋆] (Extended Abstract)

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

[1] IBM T.J. Watson Research Center,
canetti@watson.ibm.com.
[2] EE Department, Technion,
hugo@ee.technion.ac.il

Abstract. Recently, Canetti and Krawczyk (Eurocrypt'2001) formulated a notion of security for key-exchange (KE) protocols, called SK-security, and showed that this notion suffices for constructing secure channels. However, their model and proofs do not suffice for proving more general composability properties of SK-secure KE protocols.

We show that while the notion of SK-security is strictly weaker than a fully-idealized notion of key exchange security, it is sufficiently robust for providing secure composition with arbitrary protocols. In particular, SK-security guarantees the security of the key for *any application* that desires to set-up secret keys between pairs of parties. We also provide new definitions of secure-channels protocols with similarly strong composability properties, and show that SK-security suffices for obtaining these definitions.

To obtain these results we use the recently proposed framework of "universally composable (UC) security." We also use a new tool, called "non-information oracles," which will probably find applications beyond the present case. These tools allow us to bridge between seemingly limited indistinguishability-based definitions such as SK-security and more powerful, simulation-based definitions, such as UC security, where general composition theorems can be proven. Furthermore, based on such composition theorems we reduce the analysis of a full-fledged multi-session key-exchange protocol to the (simpler) analysis of individual, stand-alone, key-exchange sessions.

Keywords: Key Exchange, Cryptographic Protocols, Proofs of Security, Composition of protocols.

1 Introduction

Authenticated Key-Exchange protocols (KE, for short) are protocols by which two parties that communicate over an adversarially controlled network can generate a common secret key. These protocols are essential for enabling the use of

[⋆] Full version appears in [CK02].

[⋆⋆] Supported by Irwin and Bethea Green & Detroit Chapter Career Development Chair.

shared-key cryptography to protect transmitted data. As such they are a central piece for building secure communications, and are perhaps the most commonly used cryptographic protocols. (Popular examples include SSH, SSL, IPSec. Many others exist.)

Capturing the security requirements from a key exchange protocol has proven to be non-trivial. On the one hand, a definition should be strong enough to guarantee the desired functionality within the protocol settings under consideration. On the other hand, it should not be overly strong and should not impose unnecessary restrictions on key-exchange protocols. Moreover, it should be simple and easy to work with as much as possible.

Numerous works studying the cryptographic security for key-exchange protocols have been carried out in the past, and some quite different definitional approaches were proposed. A very partial list includes [B$^+$91, DOW92, BR93, BJM97, BCK98, S99, CK01,] See [MOV96, Chapter 12] for more background information. Most recently, Canetti and Krawczyk [CK01], building on several prior works (most notably, [BR93,BCK98]) have proposed a definition that has several attractive properties. It is simple and permissive, and yet it was shown to suffice for the quintessential application of key-exchange, namely providing keys to symmetric encryption and authentication algorithms in order to obtain secure communication channels. In other words, the [CK01] notion of security, called SK-security, guarantees that *composing* a key exchange protocol with symmetric encryption and authentication suffices for the specific purpose of providing secure channels.

This specific composability property of SK-security is indeed an important one. However, we would like to be able to guarantee more general composability properties of key-exchange protocols. Specifically, we would like to be able to guarantee that a key exchange protocol remains secure for *any* application protocol that may wish to set-up secret keys between pairs of parties, and even when the key-exchange protocols runs concurrently with an arbitrary set of other protocols. In addition, we would like to have definitions of the task of providing secure channels with similar composability properties.

In order to provide such strong composability properties one needs a general framework for representing and arguing about arbitrary protocols. We use the recently proposed framework of [C01]. This framework allows formulating definitions of security of practically any cryptographic task. Furthermore, it is shown that protocols proven secure in this framework maintain their security under a very general composition operation, called universal composition. Following [C01], we refer to notions of security formulated in this framework as universally composable (UC).

Our main result is a universally composable notion of security for key exchange protocols that is *equivalent* to SK-security. This allows us to combine the relative simplicity and permissiveness of SK-security with the strong composability properties of the UC framework. We also provide a UC definition of secure channels and demonstrate that our notion of UC key exchange suffices for realizing UC secure channels.

An additional advantage of the new definitions is that they treat key-exchange and secure-channel protocols as protocols for a *single session* between two parties (i.e., a single exchange of a key, or a single pairwise communication session). In contrast, previous works treated such protocol as multi-party protocols where a single instance of the protocol handles multiple pairwise sessions in a multi-party network. On top of being conceptually simpler, the single-session treatment simplifies the analysis of protocols.

Obtaining these definitions and especially proving equivalence with SK-security proves to be non-trivial, and required some new definitional techniques that may well be useful elsewhere. Let us elaborate.

Bridging two approaches to defining security. The definition of SK-security follows a definitional approach which is often called "security by indistinguishability". This approach proceeds roughly as follows. In order to define the security of some cryptographic task, formulate two *games,* G_0 and G_1, in which an adversary interacts with the protocol under consideration. The protocol is considered secure if no feasible adversary can distinguish between the case where it interacts in game G_1 and the case where it interacts in game G_2. This definitional approach was first used to define semantic security of encryption schemes [GM84] and was used in many definitions since. It was first applied to the context of key-exchange protocols in [BR93].

In contrast, definitions in the UC framework follow a different definitional approach, which is often referred to as "security by emulation of an ideal process". This approach proceeds roughly as follows. In order to define the security of some cryptographic task, formulate an "ideal process" that captures the desired functionality of the task. Typically, this ideal process involves adding an idealized "trusted party" to the system and having the trusted party compute the desired output for the parties. A protocol is considered secure if for any adversary that attacks the protocol there exists another adversary (a "simulator") that causes essentially the same effect on the system by interacting with the ideal process for the task. This implies that the real protocol is essentially as secure as the ideal process. This approach is used to formulate an alternative (and equivalent) notion of semantic security of encryption schemes (see e.g., [G01]). In addition, it is used for capturing security of tasks such as zero-knowledge [GMRa89] and general cryptographic protocols (e.g., [GL90,MR91,B91,C00,PSW00,DM00,C01]).

Typical advantages of definitions that take the first approach are relative simplicity and permissiveness. On the other hand, definitions that take the second approach usually appear to capture the security requirements in a more "convincing" way. More importantly, the second approach (and in particular the UC framework) enables demonstrating the general "secure composability" properties discussed above.

One case where definitions that follow the two approaches were shown to be equivalent is the case of semantically secure encryption against chosen plaintext attacks [GM84,G01]. However, in most other cases the two approaches seem to result in distinct notions of security, where the emulation approach typically results in a strictly more restrictive definition. One example is the case of Zero-

Knowledge versus Witness-Indistinguishable protocols (see, e.g., [FS90,G01]). Another example is key exchange protocols: There are protocols that are SK-secure but do not satisfy the emulation-based definitions of [BCK98,S99]. Interestingly, these include natural protocols that do not exhibit any exploitable security weaknesses. A quintessential example is the original two-round Diffie-Hellman protocol in the case of ideally authenticated communication.

Indeed, an initial attempt at formalizing a UC definition of secure key-exchange protocols results in a definition that is even more restrictive than [BCK98,S99], and thus strictly more restrictive than SK-security. We thus *relax* the UC notion so that it becomes *equivalent* to SK-security. To do that, we modify the "ideal process" to provide the simulator with additional information that the simulator can use to complete its simulation. This information, which we call a "non-information oracle", has the property that it is "computationally independent" from the exchanged key and therefore does not over-weaken the notion of security. In fact, we show that the resultant, relaxed definition is *equivalent* to SK-security. See the overview section (Section 2.4) for a sketch of our relaxation technique via non-information oracles.

On defining and realizing secure channels. Another contribution of this work is a universally composable definition of secure channels. This provides a notion of secure channels with strong composability guarantees. As in the case of key-exchange protocols, we first formulate the intuitively-immediate version of UC secure channels. We demonstrate that SK-secure key-exchange can be combined with any message authentication function and a certain class of encryption mechanisms in order to realize secure channels. This provides further assurance in the adequacy of the notion of SK-security.

However, it turns out that some natural encryption mechanisms result in protocols that do *not* realize this notion of UC secure channels. Here we encounter a similar problem as in the case of UC-secure key exchange: Some of these "insecure" protocols do not seem to have any exploitable security weakness. We remark that here the problem stems from a different source, namely the use of (symmetric) encryption in a setting where the adversary adaptively corrupts parties. In particular, the problem persists even if we use strong UC key exchange protocols. We formulate a relaxed version of UC secure channels (called weak UC secure-channels) based on a variant of non-information oracles. We demonstrate that SK-secure key-exchange, combined with any encryption scheme (that is semantically secure against chosen plaintext attacks) and any message authentication function, results in a weak UC secure-channels protocol.

Organization. Section 2 provides an overview of the definition of SK-security, the UC framework, and our results. For lack of space, we do not include a detailed technical exposition of our results in this extended abstract. For full details, the reader is referred to [CK02].

2 Overview

Sections 2.1 and 2.2 provide some background on the definitions of [CK01] and the UC framework, respectively. Later sections present our contributions. In Section 2.3 we describe the methodology for reducing the analysis of a multi-session protocol to the analysis of a simplified single-session protocol. Section 2.4 overviews our results regarding universally composable key exchange protocols. Finally, Section 2.5 overviews our results regarding the definition and construction of UC secure channels. Throughout, the presentation remains high-level and ignores many important details, such as the syntax of key-exchange protocols, session-id's, and many others. Full details are presented in [CK02].

2.1 On the Definitions of [CK01]

We very briefly sketch the [CK01] definition of SK-security and its applicability to secure channels (refer to that paper for a precise description). Following [BCK98], two models of computation are first defined: the unauthenticated-links model (UM) and the authenticated-links model (AM). In both models the communication is public, asynchronous, and without guaranteed delivery of messages. In both models the protocols are message-driven (i.e., a party is activated with an incoming message, performs some internal computation, possibly generates outgoing messages, and then waits for the next activation). Furthermore, in both models the adversary may adaptively corrupt parties, or individual key-exchange sessions within parties, and obtain their internal states. In the UM the adversary can arbitrarily modify messages before delivering them. In the AM the adversary can deliver only messages that were sent by parties, and must deliver them unmodified.

Key-exchange protocols are treated as multiparty protocols where multiple pairwise exchanges are handled within a single instance of the protocol. That is, each instance of a key exchange protocol (running within some party) consists of multiple KE-sessions, where each KE-session is an invocation of some subroutine which handles a single exchange of a key with a given peer and a given session ID. To define SK-security, the following game between an adversary and parties running the protocol is formulated (following [BR93]). The adversary is allowed to invoke multiple KE-sessions within parties to exchange keys with each other. It can then deliver messages and corrupt parties (and expose KE-sessions within parties) as in the UM (resp., AM). When the adversary chooses to, it announces a specific KE-session to be the "test session". Once the session completes, a random bit b is chosen and the adversary receives a "test value." If $b = 0$ then the test value is the key generated in this session. If $b = 1$ then the test value is a random value. The adversary can then continue the usual interaction except that it is not allowed to expose the test session or the "matching" session held by the test session's partner. At the end of its run the adversary outputs a guess (a single bit). The adversary wins if it manages to guess the value of b.

A key exchange protocol is secure in the UM (resp., AM) if no adversary can cause two partners of an exchange to output different values of the session key, and in addition no adversary can win in the above game with probability non-negligibly more than one half.

A secure channels protocol is defined to be a protocol which is a secure network authenticator and also a secure network encryptor. The definition of secure network authenticators follows that of [BCK98]: A protocol α is a secure network authenticator if any protocol π in the AM, when composed with protocol α (i.e., when each sending of a message by π is replaced with an activation of protocol α) results in a composed protocol that has essentially the same functionality as π — but in the UM.

The definition of network encryptors follows the style of definition by indistinguishability. That is, first a game between an adversary and parties running the protocol is formulated. The game captures the requirement that the adversary should be unable to distinguish between encryptions of two adversarially chosen test-messages in some session, even after the adversary sees encryptions of messages of its choice and decryptions of ciphertexts of its choice (except for decryptions that result in the test-message itself). A network encryptor is deemed secure if no adversary can win the game with non-negligible probability.

Consider the following generic protocol for realizing secure channels, given a key-exchange protocol, an encryption scheme and a message authentication function: In order to set-up a secure channel, the two partners first run a key-exchange protocol and obtain a key. Then, the sender encrypts each message and then sends the ciphertext together with a tag computed by applying the message authentication function to the ciphertext. Encryption and authentication are done using different portions of the obtained key (or via keys derived from the exchanged session key). Verification and decryption are done analogously. It is shown that if the key exchange protocol is secure, the encryption scheme is semantically secure against chosen plaintext attacks, and the message authentication function is secure, then this protocol is a secure secure-channels protocol. (A counter-based mechanism is added to avoid replay of messages.)

2.2 On Universally Composable Definitions

Providing meaningful security guarantees under composition with arbitrary protocols requires using an appropriate framework for representing and arguing about such protocols. Our treatment is based in a recently proposed such general framework [C01]. This framework builds on known definitions for function evaluation and general tasks [GL90,MR91,B91,C00,DM00,PSW00], and allows defining the security properties of practically any cryptographic task. Most importantly, in this framework security of protocols is maintained under a general composition operation with an unbounded number of copies of arbitrary protocols running concurrently in the system. This composition operation is called universal composition. Similarly, definitions of security in this framework are called universally composable (UC). We briefly summarize the relevant properties of this framework. See more details in [C01,CK02].

As in other general definitions, the security requirements of a given task (i.e., the functionality expected from a protocol that carries out the task) are captured via a set of instructions for a "trusted party" that obtains the inputs of the participants and provides them with the desired outputs (in one or more iterations). Informally, a protocol securely carries out a given task if running the protocol amounts to "emulating" an ideal process where the parties hand their inputs to a trusted party with the appropriate functionality and obtain their outputs from it, without any other interaction. We call the algorithm run by the trusted party an ideal functionality.

In order to allow proving the composition theorem, the notion of emulation in this framework is considerably stronger than previous ones. Traditionally, the model of computation includes the parties running the protocol and an adversary, $\mathcal{A}$. "Emulating an ideal process" means that for any adversary $\mathcal{A}$ there should exist an "ideal process adversary" (or, simulator) $\mathcal{S}$ that results in similar distribution on the outputs for the parties. Here an additional adversarial entity, called the environment $\mathcal{Z}$, is introduced. The environment generates the inputs to all parties, reads all outputs, and in addition interacts with the adversary in an arbitrary way throughout the computation. (Arbitrary interaction between $\mathcal{Z}$ and $\mathcal{A}$ is essential for proving the universal composition theorem.) A protocol is said to securely realize a given ideal functionality $\mathcal{F}$ if for any adversary $\mathcal{A}$ there exists an "ideal-process adversary" $\mathcal{S}$, such that *no environment* $\mathcal{Z}$ can tell whether it is interacting with $\mathcal{A}$ and parties running the protocol, or with $\mathcal{S}$ and parties that interact with $\mathcal{F}$ in the ideal process. (In a sense, here $\mathcal{Z}$ serves as an "interactive distinguisher" between a run of the protocol and the ideal process with access to $\mathcal{F}$. See [c01] for more motivating discussion on the role of the environment.)

The following universal composition theorem is proven in [c01]. Consider a protocol π that operates in a *hybrid* model of computation where parties can communicate as usual, and in addition have ideal access to an unbounded number of copies of some ideal functionality $\mathcal{F}$. (This model is called the $\mathcal{F}$-hybrid model.) Let ρ be a protocol that securely realizes $\mathcal{F}$ as sketched above, and let π^ρ be the "composed protocol". That is, π^ρ is identical to π with the exception that each interaction with some copy of $\mathcal{F}$ is replaced with a call to (or an invocation of) an appropriate instance of ρ. Similarly, ρ-outputs are treated as values provided by the appropriate copy of $\mathcal{F}$. Then, π and π^ρ have essentially the same input/output behavior. In particular, if π securely realizes some ideal functionality $\mathcal{G}$ given ideal access to $\mathcal{F}$ then π^ρ securely realizes $\mathcal{G}$ from scratch.

We also make use of an additional composition operation, called universal composition with joint state (JUC), proposed in [cr02]. This operation is similar to universal composition, with the important difference that multiple instances of the subroutine protocol, ρ, may have some amount of joint state. (In contrast, if universal composition is used then each instance of ρ has its own separate local state.) This becomes useful in the case of key-exchange protocols where multiple protocol instances (sessions) have access to the same long-term authentication module (realized, for instance, via a signature scheme that uses the same signature key for authenticating multiple sessions of the key-exchange protocol run under a party; in this case the signature key represents a joint state).

Extensions to the UC model. As a preliminary step for our study, we cast the unauthenticated-links model (UM) and the authenticated-links model (AM) of [CK01] in the UC framework. This is done by casting these models as "hybrid models" with access to the appropriate ideal functionality. In both cases the ideal functionality is $\mathcal{F}_{auth}$ from [C01], which allows an ideally authenticated transmission of a single message. In the AM the parties have access to an unlimited number of copies of $\mathcal{F}_{auth}$. In the UM each party can send only a single message to each other party using $\mathcal{F}_{auth}$. We also extend the UC framework to accommodate the session-state corruption operation of [CK01] that allows the adversary to obtain the internal data of individual sessions within parties.

2.3 Single-Session vs. Multi-session Protocols

In contrast to previous works, we treat key exchange and secure channel protocols as protocols where each instance handles a single pairwise session (i.e., a single exchange of a key or a single pairwise communication session). This results in greater conceptual and analytical simplicity. However, it requires taking care of the following two issues.

Multi-session extensions. In order to be able to compare the definitions here to the definitions of [CK01], we define the multi-session extension of a (single session) key exchange protocol π to be the protocol $\hat{\pi}$ that handles multiple exchanges of a key, where each exchange consists of running an instance of the original protocol π. The multi-session extension of a (single session) secure session protocol is defined analogously. This way, we are able to state and prove results of the sort "A single-session protocol π is secure according to some UC definition if and only if the multi-session extension $\hat{\pi}$ is secure according to [CK01]".

The long-term authentication module. In typical key exchange protocols multiple pairwise sessions use the same instance of a long-term authentication mechanism. (For instance, this mechanism may be a long-term shared key between parties, or a public-key infrastructure based either on digital signatures or asymmetric encryption.) Thus, pairwise key-exchange and secure channels sessions are not completely disjoint from each other. Still, the "main bulk" of the state of each such pairwise session is disjoint from all other sessions and can be treated separately. In order to do that, we proceed as follows.

First, we restrict attention to (single session) key exchange protocols that have an explicitly specified long-term authentication module. This module represents the part of the key-exchange protocol that handles the long-lived information used to bind each generated key to an identity of a party in the network. Typically, this part consists of a standard cryptographic primitive with a well-defined interface. Next, we analyze key exchange protocols under the assumption that the functionality of the long-term authentication module is ideally provided. (That is, we work in a hybrid model with access to the appropriate ideal functionality.) This in particular means that in a setting where multiple instances of a key-exchange protocols are being used, each such instance uses its own separate copy of the idealized long-term authentications module. We then use universal composition with joint state (see above) to replace all copies of the idealized long-term authentications module *single instance* of a protocol that realizes this module.

For concreteness, we further specify the functionality of the long-term authentication module, when based on digital signatures, and describe the use of universal composition with joint state for this case. (Here we basically use the modeling and results of [CR02].) Similar treatment can be done for other types of long-term authentication. We stress, however, that the results in this work are general and apply regardless of the specific long-term authentication module in use.

2.4 UC Key-Exchange

In order to establish the relationship between the notion of SK-security and UC notions, we first rephrase the definition of SK-security in the UC framework. This is done as follows. We formulate a specific environment machine, $\mathcal{Z}_{test}$, which carries out the game of the definition of SK-security. That is, $\mathcal{Z}_{test}$ expects to interact with a protocol $\hat{\pi}$ which is the multi-session extension of some key-exchange protocol π. Whenever the adversary $\mathcal{A}$ asks $\mathcal{Z}_{test}$ to invoke a session within some party to exchange a key with another party, $\mathcal{Z}_{test}$ does so. When the adversary asks to obtain the session key generated in some session, $\mathcal{Z}_{test}$ reveals the key to $\mathcal{A}$. When $\mathcal{A}$ announces a test session, $\mathcal{Z}_{test}$ flips a coin b. If $b = 0$ then $\mathcal{Z}_{test}$ hands $\mathcal{A}$ the real session key of that session. If $b = 1$ then $\mathcal{A}$ gets a random value. $\mathcal{Z}_{test}$ outputs 1 if $\mathcal{A}$ managed to guess b. (If in any session the partners output different values for the key then $\mathcal{Z}_{test}$ lets $\mathcal{A}$ determine the output.) A (single session) protocol π is said to be SK-secure if no adversary $\mathcal{A}$ can skew the output of $\mathcal{Z}_{test}$ non-negligibly away from fifty-fifty, when interacting with $\hat{\pi}$, the multi-session extension of π. In all, $\mathcal{Z}_{test}$ is designed so that this formulation of SK-security remains only a rewording of the formulation in [CK01]. We later refer to this notion of security as security by indistinguishability.

We then turn to defining UC-secure key exchange. This is done by formulating an ideal functionality that captures the security requirements from a single exchange of a key between a pair of parties. We first formulate a functionality, $\mathcal{F}_{ke}$, that simply waits to receive requests from two uncorrupted parties to exchange a key with each other, and then privately sends a randomly chosen value to both parties, and halts. (If one of the partners to an exchange is corrupted then it gets to determine the value of the key.) We first show that known protocols satisfy this notion:

Theorem 1. *The ISO 9798-3 Diffie-Hellman key exchange protocol authenticated via digital signatures (see* [CK01]*) securely realizes functionality $\mathcal{F}_{ke}$ in the* UM*, under the Decisional Diffie-Hellman assumption, and assuming security of the signature scheme in use.*

Next we show that any protocol that securely realizes $\mathcal{F}_{ke}$ is secure by indistinguishability:

Theorem 2. *Any protocol that securely realizes $\mathcal{F}_{ke}$ is secure by indistinguishability. This holds both in the* UM *and in the* AM*, and both with and without forward secrecy.*

The converse, however, is not true. Specifically, we show that, surprisingly, the "classic" two-move Diffie-Hellman protocol does not securely realize $\mathcal{F}_{ke}$ in the AM, whereas this protocol is secure by indistinguishability in the AM. (Other examples are also given.) Moreover, the proof of "insecurity" of this protocol does not point out to any exploitable security weakness of this protocol. Rather, it seems to point to a technical "loophole" in the UC definition. Specifically, the problem arises when a party gets corrupted, and the real-life adversary expects to see an internal state of the party. This information needs to match other information, such as the value of the session key and the messages sent by the party in the past. Mimicking such an activity in the ideal process is problematic, since the simulator needs to "commit" to messages sent by the party before knowing the value of the key, which is randomly chosen (by $\mathcal{F}_{ke}$) only later.

With the above discussion in mind, we relax the ideal key exchange functionality as follows. We define a special type of probabilistic interactive Turing machine, called a non-information oracle. Essentially, a non-information oracle has the property that its local output is computationally independent from its communication with the outside world. Now, when the functionality is asked to hand a key to a pair of uncorrupted parties, it invokes the non-information oracle $\mathcal{N}$, and lets $\mathcal{N}$ interact with the simulator. The key provided to the parties is set to be the local output of $\mathcal{N}$. When the adversary corrupts one of the partners to the session, it is given the internal state of $\mathcal{N}$.

On the one hand, we are guaranteed that the additional information provided to the simulator (i.e., to the adversary in the ideal process) does not compromise the security of the session key as long as both partners of the session remain uncorrupted. (This follows from the fact that $\mathcal{N}$ is a non-information oracle.) On the other hand, when the simulator corrupts a partner, it obtains some additional information (the internal state of $\mathcal{N}$). With an adequate choice of $\mathcal{N}$, this information allows the simulator to complete its task (which is to mimic the behavior of the real-life adversary, $\mathcal{A}$).

We call the relaxed ideal key-exchange functionality, parameterized by a non-information oracle $\mathcal{N}$, $\mathcal{F}_{wke}^{\mathcal{N}}$. A protocol π is called weakly UC secure if there exists a non-information oracle $\mathcal{N}$ such that π securely realizes $\mathcal{F}_{wke}^{\mathcal{N}}$.

Let us exemplify the use of non-information oracles by sketching a proof for the security of the classic two-move Diffie-Hellman protocol. (Let 2DH *denote this protocol.) Assume that a prime p and a generator g of a large subgroup of Z_p^* of prime order are given. Recall that the protocol instructs the initiator I to choose $x \overset{R}{\leftarrow} Z_q$ and send $\alpha = g^x$ to the responder R, who chooses $y \overset{R}{\leftarrow} Z_q$ and sends $\beta = g^y$ to I. Both parties locally output g^{xy} (and erase x and y if forward secrecy is desired). Simulating this interaction with access to $\mathcal{F}_{ke}$ (which only chooses a random session key and gives it to the parties) is not possible. Let us informally reason why this is the case. The simulator has to first come up with values α' and β' as the messages sent by the parties. Next, when, say, I gets* *corrupted before receiving R's message, the simulator learns the random value k that $\mathcal{F}_{ke}$ chose to be the session key, and has to come up with a value x' such that $g^{x'} = \alpha$ and $\beta'^{x'} = k$. However, since α', β' were chosen independently of k, such a value x' exists only with negligible probability $1/q$.*

To solve this problem we define the following non-information oracle, $\mathcal{N}$. Upon receiving p, q, g as defined above, $\mathcal{N}$ chooses $x, y \overset{R}{\leftarrow} Z_q$, sends out a message containing $\alpha = g^x, \beta = g^y$, locally outputs $k = g^{xy}$ and halts. It is easy to see that, under the Decisional Diffie-Hellman assumption, the local output of $\mathcal{N}$ is computationally indistinguishable from random even given the communication of $\mathcal{N}$ with the outside world. Now, having access to $\mathcal{F}^{\mathcal{N}}_{wke}$, we can simulate protocol 2DH *using the following strategy. Recall that, in order to provide I and R with a session key in the ideal process, functionality $\mathcal{F}^{\mathcal{N}}_{wke}$ first runs $\mathcal{N}$, lets the simulator obtain the messages sent by $\mathcal{N}$, and sets the session key to be the local output of $\mathcal{N}$. In our case, this means that the simulator obtains $\alpha = g^x, \beta = g^y$, while the session key is set to $k = g^{xy}$. Therefore, all the simulator has to do is say that the messages sent by I and R are α and β, respectively. Now, if either I or R is corrupted, the simulator receives from $\mathcal{F}^{\mathcal{N}}_{wke}$ the internal state of $\mathcal{N}$, which contains x, y.*

We show that a key-exchange protocol is secure by indistinguishability *if and only if* it is weakly UC secure:

Theorem 3. *A key-exchange protocol π is secure by indistinguishability if and only if there exists a non-information oracle $\mathcal{N}$ such that π securely realizes $\mathcal{F}^{\mathcal{N}}_{wke}$. This holds both in the* UM *and in the* AM*, and both with and without forward secrecy.*

Theorem 3 provides a *characterization* of the composability properties of security by indistinguishability: Using a key exchange protocol that is secure by indistinguishability is essentially the same as using an ideal key-exchange functionality that provides the adversary with some randomized information that is computationally independent from the exchanged key.

2.5 UC Secure Channels

The main application of key-exchange protocols is for providing secure channels. Indeed, [CK01] provide a definition of secure-channels protocols, and demonstrate that SK-secure key exchange suffices for realizing their definition of secure channels. (See more details in Section 2.1.)

However, the secure-channels notion of [CK01] does not provide any secure composability guarantees. For example, there is no guarantee that a secure-channels protocol remains secure when used within general "application protocols" that assume "idealized secure channels" between pairs of parties.

We formulate universally composable notions of secure channels. Such notions carry strong composability guarantees with any application protocol and with any number of other protocols that may be running concurrently in the system. In addition, in contrast with [CK01], here we treat a secure channel protocol as a single session protocol (i.e., a protocol that handles only a single communication session between two parties). The extension to the multi-session case is obtained via the general composition theorems.

Formulating a UC notion of secure channels requires formulating an ideal functionality that captures the security requirements from a secure channels protocol. We first formulate an ideal functionality, $\mathcal{F}_{sc}$, that captures these requirements in a straightforward way: Upon receiving a request by two peers to establish a secure channel between them, functionality $\mathcal{F}_{sc}$ lets the adversary know that the channel was established. From now on, whenever one of the peers asks $\mathcal{F}_{sc}$ to deliver a message m to the other peer, $\mathcal{F}_{sc}$ privately sends m to the other peer, and lets the adversary know that a message of $|m|$ bits was sent on the channel. As soon as one of the parties requests to terminate the channel, $\mathcal{F}_{sc}$ no longer transmits information on the channel. A protocol that securely realizes the functionality $\mathcal{F}_{sc}$ is called a UC secure channels protocol.

We wish to show that any weak UC key-exchange protocol suffices to build UC secure channels. More specifically, recall the generic protocol for realizing secure channels, given a key-exchange protocol, an encryption scheme and a message authentication function: In order to set-up a secure channel, the two partners first run a key-exchange protocol and obtain a key. Then, the sender encrypts each message and then sends the ciphertext together with a tag computed by applying the message authentication function to the ciphertext. Encryption and authentication are done using different portions of the obtained key. Verification and decryption are done analogously. We want to show that if the key exchange protocol is weak UC secure, the encryption scheme is semantically secure against chosen plaintext attacks, and the message authentication function is secure against chosen message attacks, then this protocol constitutes a UC secure channels protocol (i.e., it securely realizes $\mathcal{F}_{sc}$). We prove this result for a special case where the encryption function is of a certain form. That is:

Theorem 4. *Let* MAC *be a secure Message Authentication Code function, and let π be a weakly UC secure key exchange protocol. Then there exist symmetric encryption schemes* ENC *such that the above sketched protocol, based on π,* MAC, *and* ENC, *securely realizes $\mathcal{F}_{sc}$ in the* UM.

Unfortunately, this statement is not true for any semantically secure symmetric encryption scheme. There exist natural encryption protocols that are semantically secure and where the resulting protocol does *not* securely realize $\mathcal{F}_{sc}$, regardless of which message authentication function and which key-exchange protocol are in use. (In fact, most practical encryption protocols are such. This holds even if the key-exchange protocol is a strong UC secure one, i.e. if it securely realizes $\mathcal{F}_{ke}$.) As in the case of key-exchange protocols, some of the protocols that fail to realize $\mathcal{F}_{sc}$ do not seem to have any exploitable security weakness. Rather, the failure to realize $\mathcal{F}_{sc}$ stems from a technical "loophole" in the definition. As there, the problem arises when the real-life adversary adaptively corrupts a party or a session and wishes to see the plaintexts that correspond to the previously transmitted ciphertexts. As there, mimicking such behavior in the ideal process is problematic since the simulator (i.e., the ideal process adversary) has to "commit" to the ciphertext before knowing either the plaintext or the decryption key.

We thus proceed to formulate a relaxed version of the secure channels functionality. Also here we let the relaxed functionality use a non-information oracle in order to provide the simulator with "randomized information on the plaintexts" at the time when these are secretly transmitted to its recipient. More specifically, if the two partners of the secure channel are uncorrupted, then the relaxed functionality, $\mathcal{F}_{wsc}^{\mathcal{N}}$, invokes the non-information oracle $\mathcal{N}$. Whenever one party wishes to send a message m on the channel, $\mathcal{F}_{wsc}^{\mathcal{N}}$ secretly transmits m to the other party, and in addition feeds m to $\mathcal{N}$. The output of $\mathcal{N}$ is then forwarded to the adversary. When the channel or one of its peers is corrupted, $\mathcal{F}_{wsc}^{\mathcal{N}}$ reveals the internal state of $\mathcal{N}$ to the adversary.

Here the security requirement from a non-information oracle is slightly different from the case of key-exchange. Specifically, we require that the messages generated by a non-information oracle $\mathcal{N}$ be "computationally independent" from the messages received by $\mathcal{N}$. That is, we require that an interaction with $\mathcal{N}$ will be indistinguishable from a "modified interaction" where each message sent to $\mathcal{N}$ is replaced with an all-zero string of the same length before it is handed to $\mathcal{N}$.

The rationale of using non-information oracles in $\mathcal{F}_{wsc}$ is the same as in the case of $\mathcal{F}_{wke}$: the fact that $\mathcal{N}$ is a non-information oracle guarantees that the information gathered by the simulator is essentially independent from the secretly transmitted messages. However, when a party gets corrupted, the simulator received additional information which, for an appropriately chosen non-information oracle, is helpful in completing the simulation.

We say that a protocol π is weak UC secure channels if there exists a non-information oracle $\mathcal{N}$ as defined here such that π securely realizes $\mathcal{F}_{wsc}^{\mathcal{N}}$. We show that the above generic protocol is a weak UC secure channels protocol, as long as the key-exchange protocols is weak UC secure, the encryption scheme is semantically secure against chosen message attacks, and the message authentication function is secure:

Theorem 5. *Let* MAC *be a secure Message Authentication Code function, let* ENC *be a semantically secure symmetric encryption scheme, and let π be a weakly UC secure key exchange protocol. Then there exists a non-information oracle $\mathcal{N}$ for encryption such that the above-sketched protocol, based on π,* MAC*, and* ENC*, securely realizes $\mathcal{F}_{sc}^{\mathcal{N}}$ in the* UM.

Finally, as further assurance in the adequacy of this weaker notion of secure channels, we note that any weak UC secure channels protocol is also secure according to [CK01]. (Recall however that the definition of [CK01] only addresses secure channel protocols where each request to transmit a message from one party to another over the channel results in a single actual protocol message. Consequently, the implication holds only for such protocols.)

References

[B91] D. Beaver, "Secure Multi-party Protocols and Zero-Knowledge Proof Systems Tolerating a Faulty Minority", J. Cryptology (1991) 4: 75–122.

[BCK98] M. Bellare, R. Canetti and H. Krawczyk, "A modular approach to the design and analysis of authentication and key-exchange protocols", *30th STOC,* 1998.

[BR93] 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.

[BR95] M. Bellare and P. Rogaway, "Provably secure session key distribution – the three party case," *Annual Symposium on the Theory of Computing (STOC)*, 1995.

[B+91] 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.)

[BJM97] 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.

[C00] R. Canetti, "Security and Composition of Multiparty Cryptographic Protocols", *Journal of Cryptology,* Winter 2000. On-line version at http://philby.ucsd.edu/cryptolib/1998/98-18.html.

[C01] R. Canetti, "Universally Composable Security: A New paradigm for Cryptographic Protocols", *42nd FOCS,* 2001. Full version available at http://eprint.iacr.org/2000/067.

[CK01] R. Canetti and H. Krawczyk, "Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels", *Eurocrypt 01,* 2001. Full version at http://eprint.iacr.org/2001.

[CK02] R. Canetti and H. Krawczyk, "Universally Composable Notions of Key Exchange and Secure Channels", IACR's Eprint archive, http://eprint.iacr.org/2002.

[CR02] R. Canetti and T. Rabin, "Universal Composition with Join State", available on the Eprint archive, eprint.iacr.org/2002, 2002.

[DH76] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Trans. Info. Theory* IT-22, November 1976, pp. 644–654.

[DOW92] W. Diffie, P. van Oorschot and M. Wiener, "Authentication and authenticated key exchanges", *Designs, Codes and Cryptography*, 2, 1992, pp. 107–125.

[DM00] Y. Dodis and S. Micali, "Secure Computation", *CRYPTO '00,* 2000.

[FS90] U. Feige and A. Shamir. Witness Indistinguishability and Witness Hiding Protocols. In *22nd STOC*, pages 416–426, 1990.

[G01] O. Goldreich, *"Foundations of Cryptography",* Cambridge University Press, 2001. Prelim. version available at `http://philby.ucsd.edu/cryptolib.html`

[GL90] S. Goldwasser, and L. Levin, "Fair Computation of General Functions in Presence of Immoral Majority", *CRYPTO '90, LNCS 537,* Springer-Verlag, 1990.

[GM84] S. Goldwasser and S. Micali, Probabilistic encryption, *JCSS,* Vol. 28, No 2, April 1984, pp. 270–299.

[GMRa89] 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.

[GMRi88] S. Goldwasser, S. Micali, and R.L. Rivest. A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. *SIAM J. Comput.*, April 1988, pages 281–308.

[MOV96] A. Menezes, P. Van Oorschot and S. Vanstone, "Handbook of Applied Cryptography," CRC Press, 1996.

[MR91] S. Micali and P. Rogaway, "Secure Computation", unpublished manuscript, 1992. Preliminary version in *CRYPTO 91.*

[PSW00] B. Pfitzmann, M. Schunter and M. Waidner, "Provably Secure Certified Mail",IBM Research Report RZ 3207 (#93253), IBM Research, Zurich, August 2000.

[S99] 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.