Security Analysis of IKE's Signature-based Key-Exchange Protocol *

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

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

Abstract. We present a security analysis of the Diffie-Hellman key-exchange protocol authenticated with digital signatures used by the Internet Key Exchange (IKE) standard. The analysis is based on an adaptation of the key-exchange model from [Canetti and Krawczyk, Eurocrypt'01] to the setting where peers identities are not necessarily known or disclosed from the start of the protocol. This is a common practical setting, including the case of IKE and other protocols that provide confidentiality of identities over the network. The formal study of this "post-specified peer" model is a further contribution of this paper.

1 Introduction

The Internet Key-Exchange (IKE) protocol [11] specifies the key exchange mechanisms used to establish secret shared keys for use in the Internet Protocol Security (IPsec) standards [14]. IKE provides several key-exchange mechanisms, some based on public keys and others based on long-term shared keys. Its design emerged from the Photuris [13], SKEME [15] and Oakley [21] protocols. All the IKE key-exchange options support Diffie-Hellman exchanges but differ in the way authentication is provided. For authentication based on public-key techniques two modes are supported: one based on public-key encryption and the other based on digital signatures.

While the encryption-based modes of IKE are studied in [5], the security of IKE's signature-based mode has not been cryptographically analyzed so far. (But see [19] where the IKE protocol is scrutinized under an automated protocol analyzer.) This later mode originates with a variant of the STS protocol [8] adopted into Photuris. However, this STS variant, in which the DH key is signed, is actually insecure and was eventually replaced in IKE with the "sign-and-mac" mechanism proposed in [16, 18]. This mechanism forms the basis for a larger family of protocols referred to as SIGMA ("SIGn-and-MAc") [18] from which the IKE signature modes are particular cases.

The main goal of the current paper is to provide cryptographic analysis of IKE, and the underlying SIGMA protocols. The practical interest in this analysis work is natural given the wide deployment and use of IKE and the fact

* This proceedings version lacks most proof details essential for the results presented here; for a complete version see [4].

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

that authentication via signatures is the most common mode of public-key authentication used in the context of IKE.[1] Yet, the more basic importance of this analytical work is in contributing to a further development of a theory that supports the analysis of complex and more functional protocols as required in real-world applications. Let us discuss two such issues, that are directly relevant to the design of IKE. One such issue (not dealt with in previous formal analysis work of key-exchange protocols) is the requirement for *identity concealment*. That is, the ability to protect the identities of the peers to a key-exchange session from eavesdroppers in the network (and, in some case, from active attackers as well). While this requirement may be perceived at first glance as having minor effects on the protocols, it actually poses significant challenges in terms of design and analysis. One piece of evidence pointing out to this difficulty is the fact that the STS protocol and its variants (see [8, 20]) that are considered as prime examples of key-exchange protocols offering identity protection, turned out to be insecure (they fail to ensure an authenticated binding between peers to the session and the exchanged secret key). The general reason behind this difficulty is the conflicting character of the authentication and identity-concealment requirements.

Another issue arising in the context of IKE is the possible unavailability of the peer identity at the onset of the protocol. In previous analytical work (such as [2, 22, 5]) the peer identities are assumed to be specified and given at the onset of a session activation, and the task of the protocol is to guarantee that it is this particular *pre-specified peer* the one which the key is agreed. In contrast, in IKE a party may be activated to exchange a key with an "address" of a peer but without a specified identity for that peer. This is a common case in practical situations. For example, the key-exchange session may take place with any one of a set of servers sitting behind a (url/ip) address specified in the session activation. Or, a party may respond to a request for a key exchange coming from a peer that is not willing to reveal its identity over the network and, sometimes, not even to the responder before the latter has authenticated itself (e.g., a roaming mobile user connecting from a temporary address, or a smartcard that authenticates the legitimacy of the card-reader before disclosing its own identity). So, how do the parties know who they are authenticating? The point is that each party learns the peer's identity *during* the protocol. A secure protocol in this setting will detect impersonation and will ensure that the learned identity is authentic (informally, if Alice completes a session with the view "I exchanged the session key k with Bob", then it is guaranteed that no other party than Bob learns k, and if Bob completes the session then it associates the key k with Alice).[2] In this paper we refer to this general setting as the *"post-specified peer"* model.

---

[1] In particular, recent suggestions in the IPsec working group for variants of the key-exchange protocols in IKE fall also under the family of protocols analyzed here.

[2] The issue of whether a party may agree to establish a session with the particular peer whose identity is learned during the key-exchange process is an orthogonal issue taken care by a separate "policy engine" run by the party.

Remark. Note the crucial difference between this "post-specified peer" model and the "anonymous" model of protocols such as SSL where the server's identity is publicly known from the start of the protocol while the client's identity remains undisclosed even when the key exchange finishes. In the anonymous case, the client does not authenticate at all to the server; authentication happens only in the other direction: the server authenticates to the client. A formal treatment of this anonymous uni-directional model of authentication is proposed in [22].

The combination of the requirement for identity protection and the "post-specified peer" setting puts additional constraints on the design of protocols. For example, the natural and simple Diffie-Hellman protocol authenticated with digital signatures defined by ISO [12] and proven in [5], is not suitable for providing identity protection in the post-specified peer model. This is so since this protocol instructs each party to sign the peer identity, which in turn implies that the parties must know the peer identities before a session key is generated. In a setting where the peer identities are not known in advance, these identities must be sent over the network, in the clear, thus forfeiting identity concealment. As we will see in Section 3, the IKE and SIGMA protocols use a significantly different approach to authentication. In particular, parties never sign other parties identities; instead a MAC-based mechanism is added to "compensate" for the unsigned peer's identity. (See [18] for more information on the rationale behind the design of the SIGMA protocols.)

We present a notion of security for key exchange protocols that is appropriate for the post-specified peer setting. This notion is a simple relaxation of the key-exchange security model of [5] that suitably reflects the needs of the "post-specified" model as well as allows for a treatment of identity concealment. After presenting the adaptation of the security definition of [5] to our setting, we develop a detailed security proof for the basic protocol (denoted Σ_0) underlying the signature-based modes of IKE. This is a somewhat simplified variant that reflects the core cryptographic logic of the protocol and which already presents many of the technical issues and subtleties that need to be dealt with in the analysis. One prime example of such subtleties is the fact that the IKE protocols use the exchanged Diffie-Hellman key not only to derive a session key at the end of the session but also to derive keys used *inside* the key-exchange protocol itself to provide essential authentication functionality and for identity encryption. After analyzing and providing a detailed proof of this simplified protocol, we show how to extend the proof to deal with richer-functionality variants including the IKE protocols. The resultant analysis approach and techniques are applicable to other protocols, in particular other identity-concealing protocols and those that use the DH key during the session establishment protocol.

The security properties guaranteed by our analysis consider a strong realistic adversarial setting where the attacker has full control of the communication lines, and may corrupt session and parties at will (in an adaptive fashion). In particular, this security model and definition (even if relaxed with respect to [5]) guarantees that session keys derived in the protocol are secure for use in conjunction with symmetric encryption and authentication functions for implementing "secure channels" (as defined in [5]) that protect communications over realistic adversarially-controlled networks. Deriving such keys is the quintessential application of key-exchange protocols in general, and the fundamental requirement from the IKE protocols.

In the full version of this paper [4] we show how to embed the post-specified peer model in the framework of universally composable (UC) security [3]. Specifically, we formulate a UC notion of post-specified secure key exchange and show that protocol Σ_0 presented here satisfies this notion. The UC notion ensures strong composability guarantees with other protocols. In particular, we show that it suffices for implementing secure channels, both in the UC formalization of [6] and in the formalization of [5].

Paper's organization. In Section 2 we describe the adaptation of the security model of [5] to the post-specified peer setting, and establish the notion of security for key-exchange used throughout this paper. In Section 3 we describe Σ_0, the basic SIGMA protocol underlying all the other variants including the IKE signature-based protocols. In Section 4 we present the formal proof of the Σ_0 protocol in the model from Section 2 (due to space constraints we omit most of the lengthy technical details of this proof from this proceedings version – see [4] for a full version). In Section 5 we treat several variants of the basic protocol and extend the analysis from Section 4 to these cases. In particular, the two signature authentication variants of IKE are analyzed here (Section 5.2 and 5.4).

2 The security model

Here we present the adaptation of the security model for key-exchange protocols from [5] to the setting of post-specified peers as described above. We start by providing an overview of the model in [5] (refer to that paper for the full details). Then we describe the relaxation of the security definition required to support the post-specified setting.

2.1 The SK-security definition from [5]

Following the work of [2, 1], Canetti and Krawczyk [5] model key-exchange (KE) protocols as multi-party protocols where each party runs one or more copies of the protocol. Each activation of the protocol at a party results in a *local* procedure, called a *session*, that locally instantiates a run of the protocol and produces outgoing messages and processes incoming messages. In the case of key-exchange, a session is intended to agree on a "session key" with one other party (the "peer" to the session) and involves the exchange of messages with that party. Sessions can run concurrently and incoming messages are directed to its corresponding session via a session identifier. The activation of a KE session at a party has three input parameters (P, s, Q): the local party at which the session is activated, a unique session identifier, and the identity of the intended peer to the session. (There is also a fourth input field, specifying whether the party is the initiator or the responder in the exchange; however this field has no bearing on the security requirements and is thus ignored in this overview.) A party can be activated as initiator (e.g., by an application calling the KE procedure) or as a responder (upon an incoming key-exchange initiation message arriving from another party). The output of a KE session at a party P consists of a public triple (P, s, Q) that identifies the session, and of a secret value called the *session key*. Sessions can also be "aborted" without producing a session key value, in which case a special symbol is output instead of the session key. Sessions maintain a local state that is erased when the session *completes* (i.e., when the session produces output). Each party may have additional state, such as a long-term signature key, which is accessed by different sessions and which is not part of any particular session state.

The attacker model in [5] follows the *unauthenticated-links model* (UM) of [1] where the attacker is a (probabilistic) polynomial-time machine with full control of the communication lines between parties and free to intercept, delay, drop, inject or change all messages sent over these lines (i.e., a full-fledge "man-in-the-middle" attacker). The attacker can also schedule session activations at will and sees the output of sessions except for the values of session keys. In addition, the attacker can have access to secret information via *session exposure* attacks of three types: session-state reveal, session-key queries, and party corruption. The first type of attack is directed at a single session while still incomplete (i.e., before producing output) and its result is that the attacker learns the session state for that particular session (which does not include long-term secret information, such as private signature keys, shared by all sessions at the party). A session-key query can be performed against an individual session after completion and the result is that the attacker learns the corresponding session-key (this models leakage on the session key either via usage of the key by applications, cryptanalysis, break-ins, known-key attacks, etc.). Finally, party corruption means that the attacker learns *all* information in the memory of that party (including session states and session-key information and also long-term secrets); in addition, from the moment a party is corrupted all its actions are totally controlled by the attacker. (We stress that all attacker's actions can be decided by the attacker in a fully adaptive way, i.e., as a function of its current view).

In the model of [5] sessions can be *expired*. From the time a session is expired the attacker is not allowed to perform a session-key query or a state-reveal attack against the session, but is allowed to corrupt the party that holds the session. Protocols that ensure that expired sessions are protected even in case of party corruption are said to enjoy "perfect forward secrecy" [20] (this is a central property of the KE protocols analyzed here).

For defining the security of a KE protocol, [5] follows the indistinguishability style of definitions as used in [2] where the "success" of an attacker is measured via its ability to distinguish the real values of session keys from independent random values. In order to be considered successful the attacker should be able to distinguish session-key values for sessions that were *not exposed* by any of the above three types of attacks. (Indeed, the attacker could always succeed in its distinguishing task by exposing the corresponding session and learning the session key.) Moreover, [5] prohibits attackers from exposing the "matching session" either, where two sessions (P, s, Q) and (P', s', Q') are called *matching* if $s = s'$, $P = Q'$ and $Q = P'$ (this restriction of the attacker is needed since the matching session contains the session key as well).

As is customary, the ability of the attacker to distinguish between real and random values of the session key is formalized via the notion of a *test session* that the attacker is free to choose among all complete sessions in the protocol. When the attacker chooses the test session it is provided with a value v which is chosen as follows: a random bit b is tossed, if $b = 0$ then v is the real value of the output session key, otherwise v is a random value chosen under the same distribution of session-keys produced by the protocol but independent of the value of the real session key. After receiving v the attacker may continue with the regular actions against the protocol; at the end of its run the attacker outputs a bit b'. The attacker *succeeds* in its attack if (1) the test session is not exposed, and (2) the probability that $b = b'$ is significantly larger than $1/2$. We note that in the model of [5] the attacker is allowed to corrupt a peer to the test session once the test session expires at that peer (this captures perfect forward secrecy). The resultant security notion for KE protocols is called SK-security and is stated as follows:

Definition 1. (SK-security [5]) *An attacker with the above capabilities is called an* SK-attacker*. A key-exchange protocol π is called* SK-secure *if for all SK-attackers $\mathcal{A}$ running against π it holds:*

1. *If two uncorrupted parties complete matching sessions in a run of protocol π under attacker $\mathcal{A}$ then, except for a negligible probability, the session key output in these sessions is the same.*
2. *$\mathcal{A}$ succeeds (in its test-session distinguishing attack) with probability not more that $1/2$ plus a negligible fraction.*

(The term 'negligible' represents any function (in the security parameter) that diminishes asymptotically faster than any polynomial fraction, or a small specific quantity in a concrete security treatment).

Remark. In [5] there are two additional notions that play a central role in the analysis of KE protocols: the "authenticated-links model" (AM) and "authenticators" [1]. While these notions could have been used in our analysis too, they would have required their re-formulation to adapt to the post-specified peer setting treated here. We have chosen to save definitional complexity and develop our protocol analysis in the current paper directly in the UM model.

2.2 Adapting SK-security to the post-specified peer setting

The model of [5] makes a significant assumption: *a party that is activated with a new session knows already at activation the identity of the intended peer to the session.* That is, the authentication process in [5] is directed to verify that the "intended peer" is the party we are actually talking to. In contrast, in the "post-specified setting" analyzed here (in particular in the setting of the IKE protocol) the information of who the other party is does not necessarily exist at the session initiation stage. It may be learned by the parties only after the protocol run evolves.

Adapting the security model from [5] to the post-specified peer setting requires: (A) generalizing the formalism of key-exchange protocols to allow for unspecified peers at the start of the protocol; and (B) relaxing the security definition to accept protocols where the peer of a session may be decided (or learned) only after a session evolves (possibly not earlier than the last protocol message as is the case of IKE). Fortunately this adaptation requires only small technical changes which we describe next; all the other definitional elements remain unchanged from [5]. In particular, we keep the UM model and most of the key-exchange formalism unchanged (including full adversarial control of the communication lines and the three types of session exposure: session-state reveal, session-key queries, and party corruption).

(A) Session activation and identification. Instead of activating sessions with input a triple (P, s, Q) as in [5] (where P is the identity of the local party, s a session identifier, and Q the identity of the intended peer for the session), in the post-specified case a session at a party P is activated with a triple (P, s, d) where d represents a "destination address" that may have no implications regarding the peer's identity sitting behind this address, and is used only as information for delivery of messages related to this session. This may be, for example, a temporary address used by arbitrary parties, or an address that may identify a set of parties, etc. Note that the above (P, s, d) formalism represents a generalization of the formalism from [5]; in the latter, d is uniquely associated with (and identifies) a specific party. We keep the convention from [5] that session id's are assumed to be unique among all the session id's used by party P (this is a simple abstraction of the practice where parties provide unique session id's for their own local sessions; we can see the identifier s as a concatenation of these local identifiers – see [5] for more discussion on this topic). We use the pair of entity identity and session-id (P, s) to *uniquely name sessions* for the purpose of attacker actions (as well as for identification of sessions for the purpose of protocol analysis). The output of a session (P, s) consists of a public triple (P, s, Q) where Q is the peer to the session, and the secret value of the session key. When the session produces such an output it is called *completed* and its state is erased (only the session output persists after the session completes and until the session expires). Sessions can abort without producing a session-key output in which case the session is referred to as *aborted* (and not completed).

(B) SK security and matching sessions. The formalism used in [2, 5] to define the security of key-exchange protocols via a test session is preserved in our work. The significant (and necessary) change here is in the definition of "matching sessions" which in turn influences the limitations on the attacker's actions against the "test session" and its peers (recall, that the attacker is allowed to attack any session except for the test-session and its matching session). In [5] the matching session of a (complete) session (P, s, Q) within party P is defined as (Q, s, P) (running within Q). This is well-defined in the pre-specified setting where both peer identities are fixed from the start of the session. In our case, however, the peer of a session may only be decided (or learned) just before the completion of that session. In particular, a session (P, s) may complete with peer Q, while the session (Q, s) may not have completed and therefore its peer is not determined. In this case, corrupting Q or learning the state of (Q, s) could obviously provide the attacker with information about the session key output by (P, s, Q). We thus introduce the following modified definition of matching session.

Definition 2. *Let (P, s) be a* completed *session with public output (P, s, Q). The session (Q, s) is called the* matching session *of (P, s) if either*

1. *(Q, s) is not completed; or*
2. *(Q, s) is completed and its public output is (Q, s, P).*

Note that by this definition only completed sessions have a matching session; in particular the "matching" relation defined above is not symmetric (except if the matching session is completed too — in which case the above definition of matching session coincides with the definition in [5]). Also, note that if Q is uncorrupted then the matching session of (P, s) is unique.

Definition 3. (SK-security in the post-specified setting) *SK-security in the post-specified peer setting is defined identically as in Definition 1 but with the notion of matching sessions re-formulated via Definition 2.*

Notes on the definition: 1. We argue that the combination of the two matching conditions in Definition 2 above results in a sound definition of SK-security. In particular, it is sufficient to preserve the proof from [5] that SK-security guarantees secure channels (see below). On the other hand, none of the two matching conditions in isolation induces a satisfactory definition of security. In particular, defining the session (Q, s) to always be the matching session of (P, s) without requiring that the determined peer is correct (in condition (2)) would result in an over-restriction of the actions of the attacker against the test session to the point that such a definition would allow weak protocols to be called secure. An example of such an insecure protocol is obtained by modifying protocol Σ_0 from Section 3 by deleting from it the MAC applied to the parties identities. This modified protocol can be shown to succumb to a key-identity mis-binding (or "unknown key share") attack as in [8], yet it would be considered secure without the conditioning on the output of session (Q, s) as formulated in (2). On the other hand, condition (2) alone is too permissive for the attacker, thus resulting in a too strong definition that would exclude many natural protocols. Specifically, if we eliminate (1) then an attacker could perform a state-reveal query against (Q, s) and reveal the secret key (e.g., g^{xy}) when this information is still in the session's state memory. This would allow the attacker a strategy in which it chooses (P, s, Q) at the test session and forces (Q, s) to be incomplete, and then learn the test session key through a state-reveal attack against (Q, s).

2. The above definition of secure key-exchange in the post-specified peer setting implies a strict relaxation of the SK-security definition in [5]. On the one hand, any SK-secure protocol according to [5] is also post-specified secure provided that we take care of the following formalities. First, we use the "address field" d in the input to the session to specify the identity of a party. Then, before completing a session, the protocol checks that the identity to be output is the same as the identity specified in the "address field" (if not, the session is aborted). On the other hand, there are protocols that are secure according to Definition 3 in the post-specified model but are not secure in the pre-specified setting of [5]. The IKE protocols studied here (in particular, protocols Σ_0 and Σ_1 presented in the following sections) constitute such examples (see the remark at the end of Section 3).

3. A natural question is whether the relaxation of SK-security adopted here is adequate. One strong evidence supporting the appropriateness of the definition is the fact that the proof in [5] that SK-security implies secure channels applies also for SK-security in the post-specified peer setting (Definition 3). One technical issue that arises when applying the notion of secure channels from [5] in our context is that this notion is formulated in the "pre-specified peer" model. Yet, one can use a post-specified SK-secure KE protocol also in this setting. All is needed is that each peer verifies, before completing a KE session, that the authenticated peer (i.e., the identity to be output as the session's peer) is the same as the identity specified in the activation of the secure channels protocol. If this verification fails, then the party aborts the KE session and the secure-channels session. Alternatively, one can easily adapt the model of secure channels in [5] to the post-specified peer setting. Also in this case an SK-secure KE protocol in the post-specified model suffices for constructing (post-specified) secure channels. In all we have:

Theorem 1. *SK-security in the post-specified peer setting implies secure channels in the formulation of [5] (either with pre-specified or post-specified secure-channel peers).*

3 The basic SIGMA protocol: $\boldsymbol{\Sigma_0}$

Here we provide a description of a key-exchange protocol, denoted Σ_0, that represents a simplified version of the signature-mode of IKE. The protocol contains most of the core cryptographic elements and properties found in the full-fledge IKE and SIGMA protocols. In the next section we provide a proof of this basic protocol, and in the subsequent section we will treat some variants and the changes they require in the security analysis. These variants will include the actual IKE protocols (see Sections 5.2 and 5.4). The Σ_0 protocol is presented in Figure 1. Further notes and clarifications on the protocol follow.

Notes on the description and actions of the protocol

– For simplicity we describe the protocol under a specific type of Diffie-Hellman groups, namely, a sub-group of Z_p^* of prime order. However, the protocol and

Protocol Σ_0

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. The protocol also uses a message authentication family MAC, and a pseudorandom function family PRF.

The protocol messages

Start message ($I \rightarrow R$): s, g^x
Response message ($R \rightarrow I$): $s, g^y, ID_r, \mathrm{SIG}_r(\text{“1”}, s, g^x, g^y), \mathrm{MAC}_{k_1}(\text{“1”}, s, ID_r)$
Finish message ($I \rightarrow R$): $s, ID_i, \mathrm{SIG}_i(\text{“0”}, s, g^y, g^x), \mathrm{MAC}_{k_1}(\text{“0”}, s, ID_i)$

The protocol actions

1. The start message is sent by the initiator ID_i upon activation with session-id s (after checking that no previous session at ID_i was initiated with identifier s); the DH exponent g^x is computed with $x \stackrel{\mathrm{R}}{\leftarrow} Z_q$ and x is stored in the state of session (ID_i, s).
2. When a start message with session-id s is delivered to a party ID_r the (if session-id s did not exist before at ID_r) ID_r activates a local session s (as responder). It now generates the response message where the DH exponent g^y is computed with $y \stackrel{\mathrm{R}}{\leftarrow} Z_q$, the signature SIG_r is computed under the signature key of ID_r, and the value g^x placed under the signature is the DH exponent received by ID_r in the incoming start message. The MAC_{k_1} value is produced with $k_1 = \mathrm{PRF}_{g^{xy}}(1)$ where g^{xy} is computed by ID_r as $(g^x)^y$. Finally, the value $k_0 = \mathrm{PRF}_{g^{xy}}(0)$ is computed and kept in memory, and the values y and g^{xy} are erased.
3. Upon receiving a (first) response message with session-id s, ID_i retrieves the public key of the party whose identity ID_r appears in this message and uses this key to verify the signature on the quadruple $(\text{“1”}, s, g^x, g^y)$ where g^x is the value sent by ID_r in the start message, and g^y the value received in this response message. ID_i also checks the received MAC under key $k_1 = \mathrm{PRF}_{g^{xy}}(1)$ (where g^{xy} is computed as $(g^y)^x$) and on the identity ID_r as it appears in the response message. If any of these verification steps fails the session is aborted and a session output of “aborted (ID_i, s)” is generated; the session state is erased. If verification succeeds then ID_i completes the session with public output (ID_i, s, ID_r) and secret session key k_0 computed as $k_0 = \mathrm{PRF}_{g^{xy}}(0)$. The finish message is sent and the session state erased.
4. Upon receiving the finish message of session s, ID_r verifies the signature (under the public key of party ID_i and with g^y being the DH value that ID_r sent in the response message), and verifies the MAC under key k_1 computed in step 2. If any of the verifications steps fails the session is aborted (with the “aborted (ID_r, s)” output), otherwise ID_r completes the session with public output (ID_r, s, ID_i) and secret session key k_0. The session state is erased.

Fig. 1. The basic SIGMA protocol

subsequent analysis apply to any Diffie-Hellman group for which the DDH assumption holds (see Section 4).

- The notation $I \rightarrow R$ and $R \rightarrow I$ is intended just to indicate the direction between initiator and responser of the messages. The protocol as described here does not specify where the messages are sent to. They can be sent to a pool of messages, to a local broadcast network, to a physical or logical address, etc. The protocol and its analysis accommodate any of these (or other) possibilities. What is important is that the protocol does not make any assumption on who will eventually get a message, how many times, and when (these are all actions decided by the attacker). Also, there is no assumption on the logical connection between the address where a message is delivered and the identity (either ID_i or ID_r) behind that address. This allows us to design the protocol (and prove its security) in the "post-specified peer" model introduced in Section 2.
- ID_i and ID_r represent the real identities of the parties to the exchange. In our model we assume that every party knows the other's party public key before hand. However, one can think of the above identities as full certificates signed by a trusted CA and verified by the recipient. (In this case, the full certificate may be included as the peer's identity under the MAC or just the identity in the certificate – e.g. the "distinguished name"). Our proofs work under this certification-based model as well.
- The strings "0" and "1" are intended to separate between authentication information created by the initiator and responder in the protocol. They serve as "symmetry breakers" in the protocol. However, in the case of Σ_0 these tags are not strictly needed for security; we will see later (Section 5.1) that the protocol is secure even without them. Yet, we include them here for two reasons. First, they simplify analysis; second, they make the protocol's security more robust to changes as we will also discuss later (e.g., they defeat reflection attacks in some of the protocol's variants).
- Recall the uniqueness of session-id's assumed by our model. We use this assumption in order to simplify the model and to accommodate different implementations of this assumption. A typical way to achieve this is to require each party in the exchange to choose a random number (say, s_i and s_r respectively) and then define s to be the concatenation of these values. In this case the values s_i and s_r can be exchanged before the protocol, or s_i can replace s in the start message, and (s_i, s_r) replace s in the response message.
- Parties use the session id's to bind incoming messages to existing (incomplete) sessions. However, only the first message of each type is processed. For example if a response message arrives with session id s at the initiator of session s, then the message is processed only if no previous response message under this session was received. Otherwise the message is discarded. Same for the other message types, or if a message arrives after the session is completed or aborted.
- We note that in this description of Σ_0 session identifiers serve a dual functionality: they serve to identify sessions and bind incoming messages to these sessions, but they also serve as "freshness guarantees" against replay attacks. We choose to "overload" session id's with the two functionalities in order to

simplify presentation. However, actual protocol implementations may use two different elements for these functions: a session identifier for the first functionality above, and random (or non-repeating) nonces for replay protection.

– In practice, it is recommended not to use the plain value g^{xy} of the DH key but a hashed value $H(g^{xy})$ where H is a hash function (e.g. a cryptographic hash function such as SHA or a universal hash function, etc.). This has the effect of producing a number of bits as required to key the PRF, and (depending on the properties of the hash function) may also help to "extracting the security entropy" from the g^{xy} output. If the plain g^{xy} is used, our security results hold under the DDH assumption. Using a hashed value of g^{xy} is secure under the (possibly weaker) HDH assumption [9].

Remark. As mentioned in Section 2 it is illustrative to note that protocol Σ_0 is not secure in the original (pre-specified) model of [5]. In that model an attacker could apply the following strategy: (1) initiate a session (P, s, Q) at P; (2) activate a session (Q, s, Eve) at Q as responder with the start message from (P, s, Q) where Eve is a corrupted party (let g^x be the DH exponent in this message); (3) deliver the response message produced by Q to P (let g^y be the DH exponent in this message). The result is that P completes (P, s, Q) with a session key derived from g^{xy}, while the session (Q, s, Eve) is still incomplete and its state contains the value g^{xy}. Therefore, in the [5] model, the attacker can choose (P, s, Q) as the test session and expose (Q, s, Eve) via a state-reveal attack to learn g^{xy}. This is allowed in [5] since (Q, s, Eve) is not a matching session to the test session (only (Q, s, P) is matching to the test session). In our post-specified model, however, the attacker is not allowed to expose (Q, s) which is incomplete and then by Definition 2 it is matching to the test session (P, s). This restriction of the adversary is needed in the post-specified setting since from the point of view of Q there is no information about who the peer is until the very end of the protocol and then its temporary internal state (before receiving the finish message) is identical whether its session is controlled by the adversary (via Eve as in the above example) or it is a regular run with a honest peer P. What is crucial to note is that protocol Σ_0 (and any SK-secure protocol in the post-specified model) guarantees that *if* Q completes the session (Q, s) then its view of the peer's identity is correct and consistent with the view in the matching session (e.g., in the above example it is guaranteed that if Q completes the session, it outputs P as the peer, and only P can compute the key g^{xy}).

4 Proof of Protocol Σ_0

4.1 The Statements

We start by formulating the Decisional Diffie-Hellman (DDH) assumption which is the standard assumption underlying the security of the DH key exchange against passive attackers. For simplicity, we formulate this assumption for a specific family of DH groups, but analogous assumptions can be formulated for other groups (e.g., based on elliptic curves).

Assumption 2 *Let κ be a security parameter. Let p, q be primes, where q is of length κ bits and $q/p{-}1$, and g be of order q in Z_p^*. Then the probability distributions of quintuples*
$Q_0 = \{\langle p, g, g^x, g^y, g^{xy}\rangle : x, y \stackrel{R}{\leftarrow} Z_q\}$ *and* $Q_1 = \{\langle p, g, g^x, g^y, g^r\rangle : x, y, r \stackrel{R}{\leftarrow} Z_q\}$ *are computationally indistinguishable.*

In addition to the DDH assumption we will assume the security of the other underlying cryptographic primitives in the protocol (digital signatures, message authentication codes, and pseudorandom functions) under the standard security notions in the cryptographic literature.

Theorem 3. (Main Theorem) *Assuming DDH and the security of the underlying cryptographic functions* SIG, MAC, PRF, *the Σ_0 protocol is SK-secure in the post-specified model, as defined in Section 2.*

Proving the theorem requires proving the two defining properties of SK-secure protocols (we use the term Σ_0-*attacker* to denote an SK-attacker working against the Σ_0 protocol):

P1. If two uncorrupted parties ID_i and ID_r complete matching sessions ((ID_i, s, ID_r) and (ID_r, s, ID_i), respectively) under protocol Σ_0 then, except for a negligible probability, the session key output in these sessions is the same. The proof of this property can be found in [4].

P2. No efficient Σ_0-attacker can distinguish a real response to the test-session query from a random response with non-negligible advantage. More precisely, if for a given Σ_0-attacker we define:

- $P_{\text{REAL}}(\mathcal{A}) = Prob(\mathcal{A}$ outputs 1 when given the real test session key)
- $P_{\text{RAND}}(\mathcal{A}) = Prob(\mathcal{A}$ outputs 1 when given a random test session key)

then we need to prove that for any Σ_0-attacker $\mathcal{A}$: $|P_{\text{REAL}}(\mathcal{A}) - P_{\text{RAND}}(\mathcal{A})|$ is negligible.

4.2 Proof Plan of Property P2.

We prove property P2 by showing that if a Σ_0-attacker $\mathcal{A}$ can win the "real vs. random" game with significant advantage then we can build an attacker against one of the underlying cryptographic primitives used in the protocol: the Diffie-Hellman exchange (DDH assumption), the signature scheme SIG, the MAC scheme MAC, or the pseudorandom family PRF.

More specifically we will show that from any Σ_0-attacker $\mathcal{A}$ that succeeds in distinguishing between a real and a random response to the test-session query we can build a DDH distinguisher D that distinguishes triples g^x, g^y, g^{xy} from random triples g^x, g^y, g^r with the same success advantage as $\mathcal{A}$, or there is an algorithm (that we can construct explicitly) that breaks one of the other underlying cryptographic primitives. This distinguisher D gets as input a triple (g^x, g^y, z) where z is either g^{xy} or g^r for $r \stackrel{R}{\leftarrow} Z_q$. D starts by simulating a run of $\mathcal{A}$ on a virtual instantiation of protocol Σ_0 and uses the values g^x and g^y from the input triple as the DH exponents in the start and response message of one randomly chosen session, say s_0, initiated by $\mathcal{A}$ in this run of protocol Σ_0. The idea is that if $\mathcal{A}$ happens to choose this session s_0 (or the corresponding responder's session) as its test session then D can provide $\mathcal{A}$ with z as the response to the test-session query. In this case, if $\mathcal{A}$ outputs that the response was real then D will decide that $z = g^{xy}$, otherwise D will decide that z is random. One difficulty here is that since D actually changes the regular behavior of the parties in session s_0 (e.g. it uses the value z to derive the key k_1 used in the MAC function) then we still have to show that D has a good probability to guess the right test session, and that the original ability of $\mathcal{A}$ to distinguish between "real" and "random" is not significantly reduced by the simulation changes. Proving this involves showing several properties of the protocol that relate to the authentication elements such as signatures and MAC.

In order to specify the distinguisher D we need to define the above simulation process and the exact rules on how to choose session s_0 and how to change the behavior of the parties to that session. In order to facilitate our analysis we will actually define a sequence of several simulators which differ from each other by the way they choose the keys (k_0 and k_1) used in the processing of the s_0 session. Each of these simulators will define a probability distribution on the runs of attacker $\mathcal{A}$. At one end of the sequence of simulators will be one that corresponds to a "real" run of $\mathcal{A}$ while at the other end the simulation corresponds to a "random" experiment where the session key in session s_0 provided to $\mathcal{A}$ is chosen as a random and independent value k_0. In between, there will be several "hybrid" simulators. We will show that either all the distributions generated by these simulators are computationally indistinguishable, or that a successful distinguisher against DDH or against the PRF family exists. From this we get a proof that the "real" and "random" simulators at the ends of the sequence are actually indistinguishable, and from this that the values P_{RAND} and P_{REAL} differ by at most a negligible quantity (this negligible difference will depend on the quantified security of DDH and of the cryptographic functions).
For a full proof of property P2 see [4].

Detailed proof. The detailed proof of properties P1 and P2 (and thus of the Main Theorem 3) is lengthy and therefore omitted from these proceedings. See [4] for a complete proof.

5 Variants and Discussions

We consider the security of several variants of the protocol and extensions to its functionality. In particular, we extend the analysis to the elements found in the IKE protocols and not included in the basic protocol Σ_0.

5.1 Eliminating the initiator and responder tags in Σ_0

In protocol Σ_0 the initiator and responder include under their signatures and MAC a special tag "0" and "1", respectively. Here we show that protocol $\Sigma_0{}'$ defined identically to Σ_0 except for the lack of these tags is still secure. (We stress that the signature modes of IKE do not use these tags; this is one main reason to provide the analysis here without tags.)
For lack of space the rest of this subsection is omitted from these proceedings (see [4]).

5.2 Putting the MAC under the signature

One seemingly significant difference between protocol Σ_0 and IKE signature-mode is that in the latter the MAC tag is not sent separately but rather it is computed *under* the signature operation. That is, in the response message of IKE the responder does not send $\mathrm{SIG}_r(\text{"1"}, s, g^x, g^y), \mathrm{MAC}_{k_1}(\text{"1"}, s, ID_r)$, as in Σ_0, but rather sends the value $\mathrm{SIG}_r(\mathrm{MAC}_{k_1}(s, g^x, g^y, ID_r))$. Similarly, the pair of signature-mac is replaced in the finish message by the value $\mathrm{SIG}_i(\mathrm{MAC}_{k_1}(s, g^y, g^x, ID_i))$. The reason for this inclusion of the MAC under the signature in IKE is twofold: to save the extra space taken by the MAC tag and to provide a message format consistent with other authentication modes of IKE.[3]

Fortunately, the analysis of the protocol when the MAC goes under the signature is essentially the same as the simplified Σ_0 version analyzed before. The analysis adaptation is straightforward and is based in the following simple fact.

Lemma 1. *If* SIG *is a secure signature scheme and* MAC *a secure message authentication function then it is infeasible for an attacker to find different messages M and M' such that for a randomly chosen secret* MAC*-key k_1 the attacker can compute $\mathrm{SIG}(\mathrm{MAC}_{k_1}(M'))$ even after seeing $\mathrm{SIG}(\mathrm{MAC}_{k_1}(M))$.*

Indeed, if the attacker can do that then either $\mathrm{MAC}_{k_1}(M') \neq \mathrm{MAC}_{k_1}(M)$ with significant probability and this results in a signature forgery strategy, or $\mathrm{MAC}_{k_1}(M') = \mathrm{MAC}_{k_1}(M)$ with significant probability in which case the attacker has a strategy to break the MAC. (Note that the attacker cannot choose k_1; if it could, the lemma would not hold.)

This lemma implies that all the arguments in our proofs of Section 4 that use the unforgeability of signatures remain valid in this case. More precisely, they are extended through the above lemma to claim that if an attack is successful then either the signature scheme or the MAC are broken (the cases where the weakness comes from the insecurity of either the PRF family or the DDH assumption are treated identically as in the proof of Σ_0).

IKE's aggressive mode. With the above changes, in which the MAC is included under the signature and the "0"/"1" tags are not included, Σ_0 becomes

[3] For example, the IKE mode where authentication is provided by a pre-shared key is obtained from the signature mode by using the same MAC expression but without applying the signature on it (in this case the MAC key is derived from the pre-shared key).

basically the so called "aggressive mode of signature authentication" which is one of the two IKE's protocols based on authentication via digital signatures. One additional difference is that the IKE protocol uses the function PRF itself to implement the MAC function. Since a pseudorandom family is always a secure MAC then this implementation preserves security (in this case the key to the PRF is g^{xy} itself as in the other uses of this function in the protocol; the protocol also makes sure that the input to PRF when used as MAC is different that the inputs used for key derivation).

5.3 Encrypting the identities

Here we consider the augmentation of Σ_0 for providing identity concealment over the network. We present the main ideas behind our treatment, and omit much of the formal and technical issues.

We start by considering the following variant of protocol Σ_0. Before transmitting the response message, the responder computes a key $k_2 = \text{PRF}_{g^{xy}}(2)$ and encrypts under key k_2 the response message excluding s and g^y. That is, the response message is changed to $s, g^y, \text{ENC}_{k_2}(ID_r, \text{SIG}_r(\text{"1"}, s, g^x, g^y), \text{MAC}_{k_1}(\text{"1"}, s, ID_r))$ where ENC is a symmetric-key encryption algorithm. Upon receiving the response message the initiator computes the key k_2 as above, decrypts the incoming message with this key, and then follows with the regular verification operations of Σ_0. If successful, it prepares the finish message as in Σ_0 but sends it encrypted under ENC_{k_2} (only s is sent in the clear). Upon reception of this message the responder decrypts it and follows with the regular operations of Σ_0.

The main goal of this use of encryption is to protect the identities of the peers from disclosure over the network (at least in cases that these identities are not uniquely derivable from the visible (say, IP) address from which communication takes place). We first argue that the addition of encryption preserves the SK-security of the protocol. Then we claim that the encryption provides semantic security of the encrypted information. For the response message semantic security is provided against passive attackers only (indeed, at the point that this encryption is applied by ID_r, the initiator has not yet authenticated to ID_r so this encryption can be decrypted by whoever chose the DH exponent g^x). For information encrypted in the finish message we can provide a stronger guarantee of security, namely, semantic security also against active attackers.

We start by claiming that the modified Σ_0 protocol with encryption as described above satisfies Theorem 3. The basic idea is that if we were encrypting under a random key independent from the Diffie-Hellman exchange then the security of the protocol would be preserved (in particular, since the attacker itself can simulate such an independent encryption on top of Σ_0). However, since we are using an encryption key that is derived from g^{xy} then we need to show that if the encryption helps the attacker in breaking the SK-security of (the encrypted) Σ_0 then we can use this attacker to distinguish g^{xy} from a random value. Technically, this requires an adaptation of the proof of Theorem 3 (see [4] for more details).

In order to show secrecy protection against a passive attacker (note that a passive attacker means an eavesdropper in the network that does not collaborate with the SK-attacker which is active by definition) we consider a run of the protocol where k_2 is chosen randomly. In this case semantic security against a passive attacker follows from the assumption that the encryption function (under a random secret key) is semantically secure against chosen plaintext attacks. Accounting for the fact that k_2 is actually derived from g^{xy} requires an argument similar to the "hybrid simulators" technique in the proof of Theorem 3 (see [4]).

In the case of the finish message, the security guarantee is stronger and the secrecy protection can stand active attackers too (assuming a suitable encryption function secure against active attacks [5, 17]). We can show that for any complete session (ID_i, s, ID_r) that is not exposed by the attacker (i.e., neither this session or its matching session are corrupted), breaking the semantic security of the information transmitted under ENC_{k_2} in the finish message of session (ID_i, s) implies a distinguishing test between k_2 and a random (encryption) key. This in turn can be used to build an attack against the SK-security of the protocol or against one of its underlying cryptographic primitives.

5.4 A four message variant: IKE main mode

Here we study a four-message variant of the Σ_0 protocol. The interest in this protocol is two-fold: on one hand, if encryption is added to it (as discussed below) it allows concealing the responder's identity from active attackers and the initiator's identity from passive attacks. This is in contrast to Σ_0 where the strong active protection is provided to the initiator's identity (see Section 5.3). The other source of interest for this protocol is that it actually represents the core cryptographic skeleton of the so called "main mode with signature authentication" in IKE (which is one of the two signature-based protocols in IKE – see Section 5.2 for a discussion of the other IKE variant).

The four-message protocol, denoted Σ_1, is similar to Σ_0 except that the responder delays its authentication (via SIG_r) to a fourth message.

$I \rightarrow R$: s, g^x
$R \rightarrow I$: s, g^y
$I \rightarrow R$: $s, ID_i, \text{SIG}_i(\text{“0”}, s, g^y, g^x), \text{MAC}_{k_1}(\text{“0”}, s, ID_i)$
$R \rightarrow I$: $s, ID_r, \text{SIG}_r(\text{“1”}, s, g^x, g^y), \text{MAC}_{k_1}(\text{“1”}, s, ID_r)$

The security analysis of Σ_1 is similar to that of Σ_0 as presented in Section 4. It follows the same basic logic and structure of that proof but it requires some changes due to the addition of the fourth message and the fact that the responder authenticates after the initiator. The adaptation, however, of the previous proof to this new protocol is mostly straightforward. The details are omitted. One important point to note is that in this case (as opposed to Σ_0 – see Section 5.1) the use of the tags "0" and "1" is essential for security; at least if one regards reflection attacks (where the attacker impersonates the initiator of the exchange as responder by just replying to each of the initiator's messages with exactly the same message) as a real security threat (see discussion below).

Providing identity concealment in Σ_1 is possible via the encryption of the last two messages of the protocol (under a key $k_2 = \text{PRF}_{g^{xy}}(2)$ as in Section 5.3). In this case, the identity ID_r is protected against active attacks, while ID_i against passive attackers.

IKE's main mode. Protocol Σ_1 with the MAC included under the signature (as in Section 5.2), with encryption of the last two messages (not including the session-id s), and without the "0", "1" tags is essentially the "main mode signature authentication" in IKE. (There are some other secondary differences such as: (i) the session id s equals a pair s_1, s_2, where s_1, s_2 are "cookies" exchanged between the parties in two additional messages preceding the above four-message exchange, and (ii) the MAC function is implemented using $\text{PRF}_{g^{xy}}$). Our analysis here applies to this IKE protocol except for the fact that IKE does not use the "0", "1" tags and thus it is open to reflection attacks. We note that without the use of these tags the protocol can be proven secure in our model if exchanges from a party with itself are considered invalid, or if the initiator verifies, for example, that the incoming DH exponent in the second message differs from the one sent in the initial message. From a practical point of view, these potential reflection attacks have been regarded as no real threats in the context of IKE; in particular based on other details of the IKE specification, such as the way encryption is specified, that make these attacks unrealistic. Yet, the addition of tags as in Σ_1 would have been advisable to close these "design holes" even if currently considered as theoretical threats only.

Note: In case that the MAC goes under the signature (as in IKE and in Section 5.2) then the "0", "1" tags can go under the MAC only. Moreover, in this case one can dispense of these tags and use instead different (and computationally independent) keys k_1 and k'_1 to key the MAC going from ID_i to ID_r and from ID_r to ID_i, respectively.

5.5 Not signing the peer's DH exponent

The protocols as presented before take care of signing each party's own DH exponent as well as the peer's DH exponent. While the former is strictly necessary for security (against "man in the middle" attacks), the later is not essential and is used mainly for simplifying the proofs. If the peer's exponent is not included under the signature then the proofs become more involved since the essential binding between g^x and g^y cannot be argued directly but via a binding of these exponents to the session id.

References

1. M. Bellare, R. Canetti and H. Krawczyk, "A modular approach to the design and analysis of authentication and key-exchange protocols", *30th STOC,* 1998.
2. 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.

3. R. Canetti, "Universally Composable Security: A New paradigm for Cryptographic Protocols", *42nd FOCS,* 2001. Full version available at `http://eprint.iacr.org/2000/067`.
4. Canetti, R., and Krawczyk, H., "Security Analysis of IKE's Signature-based Key-Exchange Protocol", full version. *Cryptology ePrint Archive* (`http://eprint.iacr.org/`), 2002.
5. Canetti, R., and Krawczyk, H., "Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels", *Advances in Cryptology – EUROCRYPT 2001*, Full version in: `http://eprint.iacr.org/2001/040`.
6. Canetti, R., and Krawczyk, H., "Universally Composable Notions of Key Exchange and Secure Channels", *Eurocrypt 02,* 2002. Full version available at `http://eprint.iacr.org/2002/059.`
7. R. Cramer and V. Shoup, "A Practical Public Key Cryptosystem Provable Secure Against Adaptive Chosen Ciphertext Attack", In *Crypto '98*, LNCS No. 1462, pages 13–25, 1998.
8. W. Diffie, P. van Oorschot and M. Wiener, "Authentication and authenticated key exchanges", *Designs, Codes and Cryptography*, 2, 1992, pp. 107–125.
9. Gennaro, R., Krawczyk H., and Rabin, T., "Hashed Diffie-Hellman: A Hierarchy of Diffie-Hellman Assumptions", manuscript, Feb 2002.
10. O. Goldreich, "Foundations of Cryptography: Basic Tools", Cambridge Press, 2001.
11. D. Harkins and D. Carrel, ed., "The Internet Key Exchange (IKE)", *RFC 2409,* Nov. 1998.
12. ISO/IEC IS 9798-3, "Entity authentication mechanisms — Part 3: Entity authentication using asymmetric techniques", 1993.
13. Karn, P., and Simpson W.A., "The Photuris Session Key Management Protocol", draft-ietf-ipsec-photuris-03.txt, Sept. 1995.
14. S. Kent and R. Atkinson, "Security Architecture for the Internet Protocol", *Request for Comments 2401*, Nov. 1998.
15. Krawczyk, H., "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.
16. Krawczyk, H., *IPsec mailing list archives*, `http://www.vpnc.org/ietf-ipsec/`, April-June 1995.
17. Krawczyk, H., "The order of encryption and authentication for protecting communications (Or: how secure is SSL?)", Crypto'2001. Full version in: *Cryptology ePrint Archive* (`http://eprint.iacr.org/`), Report 2001/045.
18. Krawczyk, H., "SIGMA: the 'SIGn-and-MAc' Approach to Authenticated Diffie-Hellman Protocols", `http://www.ee.technion.ac.il/~hugo/sigma.html`
19. Meadows, C., "Analysis of the Internet Key Exchange Protocol Using the NRL Protocol Analyzer", *Proceedings of the 1999 IEEE Symposium on Security and Privacy,* IEEE Computer Society Press, May 1999.
20. A. Menezes, P. Van Oorschot and S. Vanstone, "Handbook of Applied Cryptography," CRC Press, 1996.
21. Orman, H., "The OAKLEY Key Determination Protocol", *Request for Comments 2412*, Nov. 1998.
22. 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.