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

Ran Canetti* Hugo Krawczyk†

August 26, 2002

Abstract

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

*IBM T.J. Watson Research Center, Yorktown Heights, New York 10598. Email: canetti@watson.ibm.com.

†EE Department, Technion, Haifa, Israel. Email: hugo@ee.technion.ac.il. Supported by Irwin and Bethea Green & Detroit Chapter Career Development Chair.

An abridged version of this paper appears in the proceedings of Crypto'2002.

Check for updates of this work at http://eprint.iacr.org.

Contents

1 Introduction

The Internet Key-Exchange (IKE) protocol [10] specifies the key exchange mechanisms used to establish secret shared keys for use in the Internet Protocol Security (IPsec) standards [13]. 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 [12], SKEME [14] and Oakley [20] 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 [4], the security of IKE's signature-based mode has not been cryptographically analyzed so far. (But see [18] where the IKE protocol is scrutinized under an automated protocol analyzer.) This later mode originates with a variant of the STS protocol [7] 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 [15, 17]. This mechanism forms the basis for a larger family of protocols referred to as SIGMA ("SIGn-and-MAc") [17] 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 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 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 [7, 19]) that are considered as prime examples of key-exchange protocols offering identity protection, are not secure in general (under certain circumstances they fail to ensure an authenticated binding between peers to the session and the exchanged secret key[2]) 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, 21, 4]) 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

[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]For example, the basic STS protocol from [7] fails if the attacker can register under its name the public key of the attacked party (e.g., if proof of posession is not required for PK registrarion). This failure happens regardless of the encryption function used by the STS protocol, and also when a MAC function is used to replace (or on top of) the encryption function. The variant in which the DH key g^{xy} is signed [12, 19] is insecure if the signature scheme allows for message recovery (such as in the case of RSA signature) even if proof of posession is required for PK registration. See [17].

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 smart card 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).[3] In this paper we refer to this general setting as the *"post-specified peer"* model.

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 treatment of this anonymous uni-directional model of authentication is presented in [21].

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 [11] and proven in [4], 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 SIGMA protocols (in particular, IKE) 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 [17] 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 [4] 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 [4] 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.

[3] 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.

An important point to stress is that the present security model and definition (even if relaxed with respect to [4]) 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 [4]) 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.

We also show how to formalize 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, it suffices for implementing secure channels, both in the UC formalization of [5] and in the formalization of [4].

Paper's organization. In Section 2 we describe the adaptation of the security model of [4] 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 a proof of security of the Σ_0 protocol in the model from Section 2. 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). Finally, Appendix A holds the modeling and analysis of protocol Σ_0 within the UC framework.

2 The security model

Here we present the adaptation of the security model for key-exchange protocols from [4] to the setting of post-specified peers as described above. We start by providing an overview of the model in [4] (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 [4]

Following the work of [2, 1], Canetti and Krawczyk [4] 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 [4] 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 [4] 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 (in particular, it may obtain the long term secret information at a party). Protocols that ensure that expired sessions are protected even in case of party corruption are said to enjoy "perfect forward secrecy" [19] (this is a central property of the KE protocols analyzed here).

For defining the security of a KE protocol, [4] 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, [4] 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 [4] 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 [4]) *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 [4] 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 [4] 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 [4] 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 [4] 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 [4]. 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 [4] (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 [4]; in the latter, d is uniquely associated with (and identifies) a specific party. We keep the convention from [4] 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 [4] 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, 4] 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 [4] 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 [4]). 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 [4] 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 [7], 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 [4]. On the one hand, any SK-secure protocol according to [4] 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 [4]. The IKE protocols studied here (in particular, protocols Σ_0 and Σ_1 presented in the following sections) constitute such examples (see Remark 1 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 [4] 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 [4] 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 [4] 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 4 *SK-security in the post-specified peer setting implies secure channels in the formulation of [4] (either with pre-specified or post-specified secure-channel peers).*

3 The basic SIGMA protocol: Σ_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.

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 ($I \to R$): s, g^x

Response message ($R \to I$): $s, g^y, ID_r, \text{SIG}_r(\text{"1"}, s, g^x, g^y), \text{MAC}_{k_1}(\text{"1"}, s, ID_r)$

Finish message ($I \to R$): $s, ID_i, \text{SIG}_i(\text{"0"}, s, g^y, g^x), \text{MAC}_{k_1}(\text{"0"}, s, ID_i)$

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 $x \stackrel{R}{\leftarrow} Z_q$ 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 $y \stackrel{R}{\leftarrow} Z_q$, 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 $k_1 = \text{PRF}_{g^{xy}}(1)$ where g^{xy} is computed by ID_r as $(g^x)^y$. Finally, the value $k_0 = \text{PRF}_{g^{xy}}(0)$ 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 $k_1 = \text{PRF}_{g^{xy}}(1)$ (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 $k_0 = \text{PRF}_{g^{xy}}(0)$. 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.

Figure 1: The basic SIGMA protocol

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

- The notation $I \to R$ and $R \to I$ 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.

- In the above description of Σ_0 the session identifiers serve a dual functionality: they serve to identify sessions and direct incoming messages to these sessions, but they also serve as "freshness guarantees" against replay attacks. In reality, the two functionalities may be implemented via different mechanisms. (In particular, in order to prevent replay, the second functionality requires uniqueness of the session identifiers throughout the life time of long-term keys. In contrast, if one is interested only in directing incoming messages to the correct session then it may suffice to have identifiers that repeat once old sessions are completed.) Nonetheless, for simplicity we choose to "overload" session id's with the two functionalities.

- 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 [8]. See Section 5.6.

- As we will see in Section 5 the above protocol can be simplified by eliminating some of its elements (e.g., the 0/1 tags under the MAC and signatures, and the signing of the peer's DH exponent can be eliminated without compromising security). However, this is not necessarily recommended. One benefit of these elements is in simplifying analysis, the other is making security of the protocol more robust to changes (yet, anyone making such changes needs to verify that the security and analysis of the protocol are preserved – we show several such cases in Section 5).

Remark 1 As mentioned in Section 2 it is illustrative to note that protocol Σ_0 is not secure in the original (pre-specified) model of [4]. 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 [4] 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 [4] 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}).

Remark 2 A stronger property of security can be achieved if we add to Σ_0 a fourth message in which the responder sends an "ack" message authenticated under MAC_{k_1}. In this case, the initiator does not complete the session until it gets (and verifies) this fourth message. The resultant protocol has the property that when a party completes the session it has a guarantee that the peer (either if it completed or not the session) *already* has a consistent view of who the session's peer is. In the SIGMA and IKE protocols this is not the case (in these protocols this consistency is ensured only when both peers complete the session – a condition that suffices for guaranteeing the secure channels application). The above "peer consistency" property is stronger than the guarantees of SK-security from [4] and may be significant in some scenarios.

4 Proof of Protocol Σ_0

4.1 The Statements

We start by formulating the Decisional Diffie-Hellman (DDH) assumption which is the 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 5 *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 pseudo-random functions) under the standard security notions in the cryptographic literature.

Theorem 6 (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.

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.

Remark on $\mathcal{A}$. We assume wlog that a Σ_0-attacker always chooses a test session and queries it, and does not expose the test session or its matching session before expiration. (That is, we do not consider superfluous attackers that halt without querying a test session, or invalid attackers that expose the test session.)

Remark (on the term "negligible"). We use the term 'negligible' to represent any function (in the security parameter) that diminishes asymptotically faster than any polynomial fraction. (The attacker is assumed to be polynomial-time in the security parameter of the protocol.) We note that the analysis presented here can be used to obtain more quantified security bounds via a concrete security treatment. This requires assuming explicit "(ε, t) bounds" on the security of the different cryptographic primitives used throughout the analysis, and then representing our results as a function of these particular values. Completing these details given our analysis is standard; we choose not to do this explicitly for the sake of simplified presentation.

4.2 Proof of Property P1

Proof: Let $\mathcal{A}$ be a Σ_0 attacker, and let ID_i and ID_r be two uncorrupted parties that complete matching sessions (ID_i, s, ID_r) and (ID_r, s, ID_i). We want to prove that regardless of $\mathcal{A}$'s operations both sessions output the same session key. Clearly, it suffices to show that both compute the same DH value g^{xy} (from which the session key k_0 is deterministically derived). Let us denote by u_i the DH exponent sent in the start message by ID_i where $u_i = g^{x_i}$ with x_i chosen by ID_i, and let v_i denote the DH exponent that ID_i received in the response message of session s (since ID_i completes the session s then it necessarily receives such a response message). Similarly, let u_r be the DH exponent received by ID_r in the incoming start message for session s, and by by v_r the DH exponent sent by ID_r in its response message where $v_r = g^{x_r}$ with x_r chosen by ID_r.

The signature produced by ID_r during session s is $\text{SIG}_r(\text{“1”}, s, u_r, v_r)$, while the signature that ID_i verifies in the response message is $\text{SIG}_r(\text{“1”}, s, u_i, v_i)$. Since the first signature is the only one that ID_r ever produces with the value s as the session id, then it must be that either all arguments to the first and second signature are the same, or a valid signature containing the second (and different) pair (u_i, v_i) was produced by the attacker even though ID_r did not generated such a signature. If the later case happens with non-negligible probability then we can use the attacker $\mathcal{A}$ under a simulation of protocol Σ_0 to produce a forger for the signature scheme SIG (note that ID_r is not corrupted so the forgery would be a real forgery against the scheme). Since we assume SIG to be a secure signature scheme this event must have negligible probability. Therefore, we get that except for such a negligible probability, $u_r = u_i$ and $v_r = v_i$.

Now the DH key computed by ID_i is $v_i^{x_i} = v_r^{x_i} = (g^{x_r})^{x_i} = g^{x_i x_r}$, while the DH key computed by ID_r is $u_r^{x_r} = u_i^{x_r} = (g^{x_i})^{x_r} = g^{x_i x_r}$. And therefore both compute the same session key.

(Note that we have only used the uniqueness of s and ID_r's signature in this argument, and have not used the MAC or the tags “0” or “1”.) $\square$

4.3 Proof of Property P2

4.3.1 Proof plan

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 (Lemma 7) and MAC (Lemma 11).

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 Section 4.3.2 we define this simulation process. However, 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).

4.3.2 The simulators

We define a simulator $\mathcal{S} = \mathcal{S}(\mathcal{A})$ that on parameters n (number of parties) and κ (security parameter) and a given Σ_0 attacker $\mathcal{A}$, simulates a run of protocol Σ_0 against attacker $\mathcal{A}$. Simulator $\mathcal{S}$ starts by choosing the initialization information for each of the n parties (private signature keys and their corresponding public verification keys). Then upon any activation by $\mathcal{A}$ the simulator $\mathcal{S}$ performs the Σ_0 operations on behalf of the parties and provides to $\mathcal{A}$ with the outgoing messages and public outputs generated in each session.

If at any point $\mathcal{A}$ corrupts a party, $\mathcal{S}$ hands out to $\mathcal{A}$ all the internal information of that party (including private signature key, session state for incomplete sessions, and session keys for unexpired sessions) and $\mathcal{S}$ stops operating that party (which is now under full control of $\mathcal{A}$). Upon a state-reveal query against a specific (incomplete) session, $\mathcal{S}$ provides $\mathcal{A}$ with the internal state information for that session; similarly, if $\mathcal{A}$ performs a session-key query against a (complete and unexpired) session then $\mathcal{S}$ provides $\mathcal{A}$ with the corresponding secret key output by that session. Note that at any point in its run $\mathcal{S}$ has full information to answer all of $\mathcal{A}$'s queries or perform the protocol actions on behalf of the uncorrupted parties. When $\mathcal{A}$ chooses a test session and performs its test query, $\mathcal{S}$ responds with the value of the session key as output by the test session. When $\mathcal{A}$ stops, $\mathcal{S}$ stops too with the same output (0 or 1) as $\mathcal{A}$.

We introduce several variants of the above simulator $\mathcal{S}$ which by now we generically denote by $\hat{\mathcal{S}}$ (we will describe specific variants later). An $\hat{\mathcal{S}}$ simulator is similar to $\mathcal{S}$ except for the following differences.

1. Let m be an a-priori upper bound on the number of sessions that $\mathcal{A}$ initiates (i.e., sessions for which $\mathcal{A}$ issues an initiation activation upon which a party outputs a start message) during its run with security parameter κ and n parties. At the beginning of its run $\hat{\mathcal{S}}$ chooses the following values: a number t chosen uniformly between 1 and m, an identity R_0 randomly chosen among the identities of the n parties in the protocol, two elements $x, y \in Z_q$, and two

values k_0 and k_1 of the same length as the output of the PRF functions. (The specification of the ways in which k_0 and k_1 are chosen will determine the different variants of simulators $\hat{\mathcal{S}}$ that we will define later.)

2. $\hat{\mathcal{S}}$ performs a usual simulation of $\mathcal{A}$ like $\mathcal{S}$ does except that it takes two types of special actions:

   (a) the actions related to the t-th session initiated by $\mathcal{A}$ as described in step 3 below; and

   (b) stopping its run upon the occurrence of any of the *"abort events"* that we list below, in which case $\hat{\mathcal{S}}$ stops with output 0.

3. Let the t-th session initiated by $\mathcal{A}$ be (I_0, s_0). The following actions take place as long as an abort event does not happen. The start message of session (I_0, s_0) is generated by $\hat{\mathcal{S}}$ using the value x chosen in step 1 (i.e., the start message output by (I_0, s_0) is s_0, g^x). In case that session (R_0, s_0) is activated by $\mathcal{A}$ with R_0 as responder then $\hat{\mathcal{S}}$ outputs a response message on behalf of R_0 using the exponent g^y computed using the value y chosen in step 1. Also the MAC computation for this message uses the key k_1 chosen by $\hat{\mathcal{S}}$ in step 1. If a response message is delivered to session (I_0, s_0) then the MAC verification operation for this message uses also the key k_1. Similarly, if a finish message is delivered to (R_0, s_0) then the MAC verification also uses key k_1. If any of the sessions (I_0, s_0) or (R_0, s_0) complete then the secret session key is set to k_0 as chosen by $\hat{\mathcal{S}}$ in step 1.

4. If $\mathcal{A}$ chooses (I_0, s_0) or (R_0, s_0) as its test session then the response to the test query by $\hat{\mathcal{S}}$ is k_0.

5. If $\mathcal{A}$ ends its run (without $\hat{\mathcal{S}}$ having aborted) then $\hat{\mathcal{S}}$ outputs the same bit as $\mathcal{A}$ outputs.

Now we define the abort events upon which $\hat{\mathcal{S}}$ stops its run and outputs 0. The choice of these particular events is related to some "bad events" in the cryptographic and probabilistic analysis of the protocol. Specifically, these events have the property that if $\mathcal{A}$ happens to choose one of the sessions (I_0, s_0) or (R_0, s_0) as the test session then these events will not happen (see Lemma 7). On the other hand, the lack of these events in a run between $\hat{\mathcal{S}}$ and $\mathcal{A}$ guarantees a "matching" between g^x and g^y under the (I_0, s_0) and (R_0, s_0) sessions which allows $\hat{\mathcal{S}}$ to carry the actions as defined in step 3 above.

Abort events: If any of the following events happen $\hat{\mathcal{S}}$ stops its run and outputs 0 (recall that we denote by (I_0, s_0) the t-th session initiated by $\mathcal{A}$, and by R_0 the identity randomly chosen by $\hat{\mathcal{S}}$ in Step 1 above):

- $\mathcal{A}$ corrupts I_0 or R_0 before (I_0, s_0) is completed (this includes the case that one of these parties is already corrupted at the time when the t-th session is initiated).
- $\mathcal{A}$ issues a state-reveal query against (I_0, s_0) or (R_0, s_0)
- Session (R_0, s_0) is initiated as responder before (I_0, s_0) sent its start message; or (R_0, s_0) is initiated as responder with a start message containing a DH exponent which is different than the DH exponent in the start message output by (I_0, s_0).
- The response message received by (I_0, s_0) arrives before (R_0, s_0) was activated as responder, or this response message has a different DH exponent than the DH exponent appearing in the response message output by session (R_0, s_0)

- Session (I_0, s_0) aborts.
- $\mathcal{A}$ chooses a test session other than (I_0, s_0) or (R_0, s_0), or it chooses one of these but the session completes with a peer different than R_0, I_0, respectively.
- $\mathcal{A}$ completes the game without having chosen a test session[4], or $\mathcal{A}$ stops before having initiated t sessions.

The $\hat{\mathcal{S}}$ variants. We introduce five variants of $\hat{\mathcal{S}}$ which differ by the way k_0 and k_1 are defined. We use the notation *random*() to represent a random (and independent) choice of a string of some appropriate length; also, in following definitions of $\hat{\mathcal{S}}$, x and y refer to the values chosen by the simulator in step 1 above.

$\hat{\mathcal{S}}$-REAL: $k_0 \leftarrow \text{PRF}_{g^{xy}}(0), \quad k_1 \leftarrow \text{PRF}_{g^{xy}}(1)$

$\hat{\mathcal{S}}$-RPRF: $k_0 \leftarrow \text{PRF}_k(0), \quad k_1 \leftarrow \text{PRF}_k(1), \quad k \leftarrow random()$

$\hat{\mathcal{S}}$-ALLR: $k_0 \leftarrow random(), \quad k_1 \leftarrow random()$

$\hat{\mathcal{S}}$-HYBR: $k_0 \leftarrow random(), \quad k_1 \leftarrow \text{PRF}_k(1), \quad k \leftarrow random()$

$\hat{\mathcal{S}}$-RAND: $k_0 \leftarrow random(), \quad k_1 \leftarrow \text{PRF}_{g^{xy}}(1)$

Note: in $\hat{\mathcal{S}}$-ALLR the values of k_0 and k_1 are independent, and in $\hat{\mathcal{S}}$-HYBR the values of k_0 and k are independent. The names of the simulators stand for: "real", "random prf", "all random", "hybrid", and "random", respectively. For any of the above simulators $\hat{\mathcal{S}}$ the notation $\hat{\mathcal{S}}(\mathcal{A})$ represents the distribution of runs of $\hat{\mathcal{S}}$ when interacting with $\mathcal{A}$ as the Σ_0-attacker.

Intuitively, the choice of t in step 1 of $\hat{\mathcal{S}}$ can be seen as an attempt by the simulator to guess the test session to be chosen by $\mathcal{A}$; when this guess succeeds (i.e., either (I_0, s_0) is chosen by $\mathcal{A}$ as the test session with peer R_0 or (R_0, s_0) is chosen as test session with peer I_0) then $\hat{\mathcal{S}}$-REAL corresponds to a real execution of $\mathcal{A}$ while $\hat{\mathcal{S}}$-RAND corresponds to a run of $\mathcal{A}$ where the test query is answered with a random key. The other simulators are used as intermediate games to prove that for any attacker $\mathcal{A}$, the outputs of $\hat{\mathcal{S}}$-REAL$(\mathcal{A})$ and $\hat{\mathcal{S}}$-RAND$(\mathcal{A})$ are "indistinguishable" (in the sense of the following definition of $\approx$).

Definition ($\mathcal{D} \approx \mathcal{D}'$). If $\mathcal{D}$ and $\mathcal{D}'$ are two probabilistic algorithms that output 0 or 1, then we write that $\mathcal{D} \approx \mathcal{D}'$ if and only if $|Prob(\mathcal{D} \text{ outputs } 1) - Prob(\mathcal{D}' \text{ outputs } 1)|$ is negligible.

The above "guess" of the test session by $\hat{\mathcal{S}}$ is a central element in our proofs and it motivates the following definition.

Definition (GUESS event). Let $\hat{\mathcal{S}}$ be one of the simulators defined above and $\mathcal{A}$ be a Σ_0-attacker. We say that a GUESS event happens in a run of $\hat{\mathcal{S}}(\mathcal{A})$ if the following conditions are satisfied:

1. $\mathcal{A}$ initiates at least t sessions in this run where t is the parameter chosen by $\hat{\mathcal{S}}$ in step 1 of its run (we denote by I_0 the initiator of this session and by s_0 the session id);
2. If R_0 denotes the random party chosen by $\hat{\mathcal{S}}$ in step 1 of its run then either
   (a) $\mathcal{A}$ chooses (I_0, s_0) as its test session and this session completes with peer R_0;
   or

[4]Note that we have assumed that in a regular run $\mathcal{A}$ always chooses a test session but under the changes introduced by $\hat{\mathcal{S}}$ behavior $\mathcal{A}$ could, in principle, never choose a test session.

(b) $\mathcal{A}$ chooses (R_0, s_0) as its test session and this session completes with peer I_0.

Plan of the proof of P2. In the next subsection we provide the detailed proof of P2. The plan is to show that $\hat{\mathcal{S}}$-REAL $\approx$ $\hat{\mathcal{S}}$-RAND via the indistinguishability of each pair of consecutive simulators in the above list (see Lemma 15), and then prove (Theorem 16) that $|P_{\text{REAL}}(\mathcal{A}) - P_{\text{RAND}}(\mathcal{A})| = poly * |Prob(\hat{\mathcal{S}}\text{-REAL outputs 1}) - Prob(\hat{\mathcal{S}}\text{-RAND outputs 1})|$ where "*poly*" is a quantity that is polynomial in the number of parties and number of sessions in the protocol (specifically, "*poly*" is the product of these numbers – see Remark 3).

4.3.3 Detailed Proof of P2

The following lemma is concerned with the actions of $\mathcal{A}$ and not directly with the behavior of the above simulators. However, this lemma will be instrumental later in claiming that under a GUESS event the above simulators do not abort their run (see Lemma 9). It is important to note that this lemma only uses the security of the underlying signature scheme; this is possible by the use of the tags "0" and "1" in Σ_0. If these tags are not used the lemma is still valid but requires a more involved argument that uses the security of DDH, PRF, and MAC in addition to the security of the signatures. These more involved arguments are presented in Section 5.1.

Lemma 7 *For all Σ_0-attackers $\mathcal{A}$, the following holds except for negligible probability.*

(a) *Consider a regular run by $\mathcal{A}$ in which $\mathcal{A}$ chooses a test session with output (P, s, Q) where P is the initiator. Then:*

1. *P and Q are never corrupted before expiration of the test session*
2. *Sessions (P, s) and (Q, s) are never revealed by $\mathcal{A}$*
3. *(Q, s) is initiated as responder with the start message sent by (P, s)*
4. *(P, s) receives a response message after (Q, s) was activated as responder, and this message carries the same DH exponent as in the response message output by (Q, s)*
5. *Session (P, s) does not abort.*

(b) *Consider a regular run by $\mathcal{A}$ in which $\mathcal{A}$ chooses a test session with output (Q, s, P) where Q is the responder. Then:*

1. *P and Q are never corrupted before expiration of the test session*
2. *Sessions (P, s) and (Q, s) are never revealed by $\mathcal{A}$*
3. *(Q, s) is initiated as responder with the start message sent by (P, s)*
4. *(P, s) receives a response message after (Q, s) was activated as responder, and this message carries the same DH exponent as in the response message output by (Q, s)*
5. *Session (P, s) does not abort.*

Proof: Proof of (a):

1. $\mathcal{A}$ is not allowed to corrupt the peers to the test session and we have assumed (wlog) that it does not do that.

2. (P, s) cannot be revealed by $\mathcal{A}$ since $\mathcal{A}$ is not allowed to expose the test session. As for (Q, s), a state-reveal query can be done only against incomplete sessions (since upon completion sessions erase their state). However, while incomplete, (Q, s) is the matching session to the test session so $\mathcal{A}$ cannot issue a state-reveal query against it

3. Since (P, s, Q) completes, it means that P received a response message with identity Q in it. In particular, it means that P verified the signature $\text{SIG}_Q(\text{"1"}, s, g^x, g^y)$ under Q's public key and where g^x was the value included by (P, s) in its start message. Since the above signature by Q is the only one Q could have generated as responder under session s, then we have that Q indeed was activated as responder of s under the DH exponent g^x as output in the start message by (P, s).

   If, however, it happens with non-negligible probability that such a signature was verified by P under Q's public key but Q did not produce it then we can use this non-negligible event to build a forger against SIG_Q. This is in contradiction to the assumed security of the signature scheme. (Note that by the first item above Q cannot be corrupted at the point that P verified the above signature, so Q's private key was not available to the attacker at the time of forgery.)

4. (P, s) completes with output (P, s, Q) so it must have received a response message which included Q as the identity. Moreover, P verified the signature in the response message under Q's public key, namely $\text{SIG}_Q(\text{"1"}, s, g^x, g^y)$. If (Q, s) was not activated as a responder then Q would have never generated a signature $\text{SIG}_Q(\text{"1"}, s, ...)$, so the above signature is a forgery. If Q generated such a signature then we have that g^y included under that signature was the DH exponent in the response message generated by (Q, s), and since P verified it using the DH exponent it received in the response message then we have that either this is the same exponent generated and sent by Q or the signature is a forgery. If any of the above "forgery events" happen with non-negligible probability then we can use attacker $\mathcal{A}$ to build a forger against SIG_Q that succeeds with such non-negligible probability.

5. Clearly. session (P, s) does not abort since it completes.

Proof of (b) : Omitted. Similar to (a). □

We now start proving the indistinguishability of the above defined $\hat{\mathcal{S}}$ simulators.

Lemma 8 *For all Σ_0-attackers $\mathcal{A}$, $\hat{\mathcal{S}}\text{-RAND}(\mathcal{A}) \approx \hat{\mathcal{S}}\text{-HYBR}(\mathcal{A})$*

Proof: We show that if for an attacker $\mathcal{A}$ there is a non-negligible difference (say ε) between $Prob(\hat{\mathcal{S}}\text{-RAND}(\mathcal{A})$ outputs 1) and $Prob(\hat{\mathcal{S}}\text{-HYBR}(\mathcal{A})$ outputs 1) then we can build a distinguisher for the DDH Assumption (which succeeds with non-negligible advantage ε). We show such a distinguisher $\mathcal{D}$.

Let (g^x, g^y, z) be a DDH triple input to $\mathcal{D}$. The distinguisher $\mathcal{D}$ follows the action of a $\hat{\mathcal{S}}$-RAND simulator (including abort actions) except for the following changes:

Whenever the session (I_0, s_0) chosen by $\hat{\mathcal{S}}$-RAND is initiated then $\mathcal{D}$ replaces the DH value in the start message of (I_0, s_0) with the value g^x from the DDH triple. That is, $\mathcal{D}$ produces s_0, g^x as the start message from (I_0, s_0). (Note that $\mathcal{D}$ controls I_0 which we may assume to be uncorrupted or otherwise $\hat{\mathcal{S}}$-RAND, and $\mathcal{D}$, would have aborted; same holds for $\mathcal{D}$ controlling R_0). Later, if (R_0, s_0) ever issues a response message (this means that (R_0, s_0) was activated via the start message (s_0, g^x), since otherwise $\hat{\mathcal{S}}$-RAND aborts), then $\mathcal{D}$ generates a response message from (R_0, s_0) of the form $s_0, g^y, R_0, \text{SIG}_{R_0}(\text{"1"}, s_0, g^x, g^y), \text{MAC}_{k_1}(\text{"1"}, s_0, R_0)$, where g^y is the second element in the DDH triple, and $k_1 = \text{PRF}_z(1)$ where z is the third element in this triple. If (I_0, s_0) receives a response message with a DH exponent different than g^y (i.e., the second element in the DDH triple) then $\mathcal{D}$ aborts (as $\hat{\mathcal{S}}$-RAND would do). Otherwise, I_0 verifies the signature included in the incoming response message under the public key of the sender (as it appears in the response message), and checks the MAC under key $k_1 = \text{PRF}_z(1)$, where z is the third element in the DDH triple. If the verification fails (I_0, s_0) aborts the session and $\mathcal{D}$ aborts its run (as $\hat{\mathcal{S}}$-RAND does). Otherwise, $\mathcal{D}$ makes (I_0, s_0) output a finish message of the form: $s_0, I_0, \text{SIG}_{I_0}(\text{"0"}, s, g^y, g^x), \text{MAC}_{k_1}(\text{"0"}, s_0, I_0)$, where $k_1 = \text{PRF}_z(1)$ with z being the third element in the DDH triple. On incoming finish message to (R_0, s_0) all actions are as in a regular run of $\hat{\mathcal{S}}$-RAND but the MAC in the message is verified using key $k_1 = \text{PRF}_z(1)$. All other actions of $\hat{\mathcal{S}}$-RAND, including the completion of sessions (I_0, s_0) and (R_0, s_0) follow the regular specifications of a run of $\mathcal{A}$ under $\hat{\mathcal{S}}$-RAND (in particular, if any of the sessions (I_0, s_0) or (R_0, s_0) complete then they output the random key k_0 chosen by $\hat{\mathcal{S}}$-RAND as the session key for these sessions).

We now argue that in case that $z = g^{xy}$ the probability distribution of runs of the distinguisher $\mathcal{D}$ under attacker $\mathcal{A}$ is the same as the distribution of runs of $\hat{\mathcal{S}}$-RAND under $\mathcal{A}$. First note that for sessions other than (I_0, s_0) and (R_0, s_0) the actions of $\mathcal{D}$ do not differ from those of $\hat{\mathcal{S}}$-RAND. As for sessions (I_0, s_0) and (R_0, s_0), the values g^x and g^y used in the start and response messages of these sessions are distributed identically as in a regular run of the Σ_0 protocol, namely, they are chosen independently and uniformly over the group generated by g. (Such are the specifications of Σ_0 and such is the way g^x and g^y are chosen under the DDH assumption.)

Moreover, since any event that brings to an association of g^x to a different DH exponent than g^y (and of g^y to a different exponent than g^x) causes an abort action by $\hat{\mathcal{S}}$-RAND (and then abort by $\mathcal{D}$) then all MAC computations in (I_0, s_0) and (R_0, s_0) that are visible to $\mathcal{A}$ are done under the key $z = g^{xy}$ as specified by the protocol and by $\hat{\mathcal{S}}$-RAND. Finally, the state of sessions (I_0, s_0) and (R_0, s_0) is never visible to $\mathcal{A}$ (state-reveal queries against these sessions or corruption of I_0 or R_0 lead to abort by $\mathcal{D}$) therefore the differences in these states between the run of $\hat{\mathcal{S}}$-RAND and the run of $\mathcal{D}$ do not influence the view of $\mathcal{A}$. (Note that such differences in the session state do exist: under the run by $\mathcal{D}$ the powers x and y of the DH exponents g^x and g^y do not appear in the state of (I_0, s_0) and of (R_0, s_0) while in a regular run of $\hat{\mathcal{S}}$-RAND they do appear. However, this would be visible to $\mathcal{A}$ only via state-reveal queries which lead to abort by $\mathcal{D}$ and $\hat{\mathcal{S}}$-RAND.) Note that all actions of $\mathcal{D}$ after (I_0, s_0) sends its finish message do not deviate from the regular actions of $\hat{\mathcal{S}}$-RAND and, by the above arguments, also the view of $\mathcal{A}$ at that point (and then after that) is the same as in a run under $\hat{\mathcal{S}}$-RAND. We therefore have that: $Prob(\hat{\mathcal{S}}\text{-RAND}(\mathcal{A})$ outputs $1) = Prob(\mathcal{D}(\mathcal{A})$ outputs $1 : z = g^{xy})$.

In the case of z being a random element g^r, a similar argument as above shows that the runs of $\mathcal{D}(\mathcal{A})$ are distributed exactly as the runs of $\hat{\mathcal{S}}$-HYBR$(\mathcal{A})$, that is: $Prob(\hat{\mathcal{S}}\text{-HYBR}(\mathcal{A})$ outputs $1) =$ $Prob(\mathcal{D}(\mathcal{A})$ outputs $1 : z = random)$

Now, by the DDH assumption it must be that for all $\mathcal{A}$, $Prob(\mathcal{D}(\mathcal{A})$ outputs $1 : z = g^{xy})$ equals $Prob(\mathcal{D}(\mathcal{A})$ outputs$1 : z = random)$ up to a negligible difference, and therefore we get that for all attackers $\mathcal{A}$, $\hat{\mathcal{S}}\text{-RAND}(\mathcal{A}) \approx \hat{\mathcal{S}}\text{-HYBR}(\mathcal{A})$.

□

The next Lemma shows that for any attacker $\mathcal{A}$, $\hat{\mathcal{S}}$-RAND$(\mathcal{A})$ has a significant probability of guessing the test session. This property will "propagate" through our later proofs to all the other $\hat{\mathcal{S}}$ simulators.

Lemma 9 *For any Σ_0-attacker $\mathcal{A}$, the probability of a* GUESS *event under a run of $\hat{\mathcal{S}}$-*RAND*$(\mathcal{A})$ is at least $1/(m \cdot n)$ (recall that m is the number of sessions initiated by $\mathcal{A}$ and n is the number of parties in the protocol – also see Remark 3).*

Proof: Let $\mathcal{S}(\mathcal{A})$ be a regular simulator of protocol Σ_0 under an attacker $\mathcal{A}$. Since $\mathcal{A}$ always selects a test session then if one chooses a random session (I_0, s_0) and random peer R_0 the probability that a run of $\mathcal{S}(\mathcal{A})$ ends with ("0", I_0, s_0, R_0) or ("1", R_0, s_0, I_0) as the output of the test session is at least $1/(m \cdot n)$.

Let $\hat{\mathcal{S}}$-RAND$'$ be a simulator that acts exactly as $\hat{\mathcal{S}}$-RAND except that it does not stop (neither outputs 0) in the case of abort events. Note that under $\hat{\mathcal{S}}$-RAND$'(\mathcal{A})$ the answer to the session-key query is a random key while under $\mathcal{S}(\mathcal{A})$ it is the real session key. However, this difference does not influence the way $\mathcal{A}$ chooses the test session (which obviously happens before the session-key query is answered). Therefore the probability of a GUESS event under $\hat{\mathcal{S}}$-RAND'$(\mathcal{A})$ is exactly the same as the probability a GUESS event under $\mathcal{S}(\mathcal{A})$, and then at least $1/(m \cdot n)$.

Now consider a fixed set of coins for $\hat{\mathcal{S}}$-RAND$'$ and for $\mathcal{A}$ that brings $\hat{\mathcal{S}}$-RAND$'(\mathcal{A})$ to a GUESS event. If now we look back at a regular run of $\hat{\mathcal{S}}$-RAND$(\mathcal{A})$ (i.e., with abort actions) with the same set of coins, the run still will produce a GUESS event since by Lemma 7 none of the abort events happen with respect to the test session (which in this case is either (I_0, s_0) or (R_0, s_0)) and therefore no abort event happens under $\hat{\mathcal{S}}$-RAND$(\mathcal{A})$ either. That is, under this set of coins the run of $\hat{\mathcal{S}}$-RAND$(\mathcal{A})$ is not different than the run of $\hat{\mathcal{S}}$-RAND$'(\mathcal{A})$.

Thus each set of coins that bring $\hat{\mathcal{S}}$-RAND$'(\mathcal{A})$ to GUESS will also bring $\hat{\mathcal{S}}$-RAND$(\mathcal{A})$ to a GUESS and thus $Prob$(GUESS under $\hat{\mathcal{S}}$-RAND$(\mathcal{A})$) $\geq Prob$(GUESS under $\hat{\mathcal{S}}$-RAND'$(\mathcal{A})$) $\geq 1/(m \cdot n)$. □

Lemma 10 *For any Σ_0-attacker $\mathcal{A}$, the probability of a* GUESS *event under a run of $\hat{\mathcal{S}}$-*HYBR*$(\mathcal{A})$ is, up to a negligible difference, the same as the probability of a* GUESS *event under a run of $\hat{\mathcal{S}}$-*RAND*$(\mathcal{A})$.*

Proof: If there is a non-negligible difference between the GUESS probability in a run under $\hat{\mathcal{S}}$-RAND and under $\hat{\mathcal{S}}$-HYBR, then we can build a distinguisher $\mathcal{D}'$ for DDH. Let $\mathcal{D}$ be the distinguisher from the proof of Lemma 8. On input a DDH triple (g^x, g^y, z) the distinguisher $\mathcal{D}$' runs $\mathcal{D}$ except that $\mathcal{D}$' outputs 1 if and only if in the run of $\mathcal{D}$ a guess event happens (in any other case it outputs 0). Following the proof of Lemma 8 we get that in case that $z = g^{xy}$ the distinguisher $\mathcal{D}$' outputs 1 with the probability of a GUESS event under $\hat{\mathcal{S}}$-RAND, while if $z = random$, it outputs 1 with the probability of a GUESS event under $\hat{\mathcal{S}}$-HYBR. □

The following is a central lemma in our analysis; it shows that when a GUESS event happens then one of the sessions (I_0, s_0) and (R_0, s_0) is the test session and the other is its matching session. Therefore, in this case the attacker is not allowed to expose any of these two session (until expiration). This property is used in an essential way to establish the value of P_{RAND} and P_{REAL} (Lemmas 13 and 14). It is interesting to note that the proof of Lemma 11 uses the security of the MAC and PRF families but not the security of the signatures (or the DDH assumption). However, when proving protocol Σ_0 without the "0" and "1" tags (see Section 5.1) the proof is more complex and involves the unforgeability of signature as well.

Lemma 11 *For all Σ_0-attackers $\mathcal{A}$, if a* GUESS *event happens under a run of $\hat{\mathcal{S}}$-*HYBR*$(\mathcal{A})$ then the following properties hold (except for negligible probability):*
(i) if (I_0, s_0) was chosen by $\mathcal{A}$ as the test session then (R_0, s_0) (either if completed or not) is its matching session; (ii) if (R_0, s_0) was chosen by $\mathcal{A}$ as the test session then (I_0, s_0) is its matching session.

Proof: (i) Since we assume a GUESS event then if (I_0, s_0) is chosen by $\mathcal{A}$ as the test session then the peer to the session is R_0. By definition of matching session, as long as (R_0, s_0) is incomplete it is matching to (I_0, s_0). If (R_0, s_0) is complete and its output is (R_0, s_0, ID) then by definition (R_0, s_0) matches (I_0, s_0) if and only if $ID = I_0$. We want to prove that if (R_0, s_0) completes then $ID = I_0$.

Assume that (R_0, s_0) completes with peer ID. This means that in the finish message received by (R_0, s_0) before the session completed, R_0 verified the value $\text{MAC}_{k_1}(\text{"0"}, s_0, ID)$ under $k_1 = \text{PRF}_k(1)$ where k is a random key chosen by $\hat{\mathcal{S}}$-HYBR and never provided to the attacker. At this point there could have been two examples of MAC_{k_1} output in the protocol (and no other use of k)[5], namely, $\text{MAC}_{k_1}(\text{"1"}, s_0, R_0)$ and $\text{MAC}_{k_1}(\text{"0"}, s_0, I_0)$. Therefore, if based on this information the attacker has non-negligible probability of producing $\text{MAC}_{k_1}(\text{"0"}, s_0, ID)$ for $ID \neq I_0$ then we can build, based on $\hat{\mathcal{S}}$-HYBR, a forger to the MAC function under key $k_1 = \text{PRF}_k(1)$, where k is a random independent key. This forger can then be turned into a distinguisher to the PRF function, or into a forger against the MAC function (with random keys). Since we assume these functions to be secure then the probability that (R_0, s_0) ends with peer $ID \neq I_0$ is negligible.

(ii) Since we assume a GUESS event then if (R_0, s_0) is chosen by $\mathcal{A}$ as the test session then the peer to the session is I_0. By definition of matching session, as long as (I_0, s_0) is incomplete it is matching to (R_0, s_0). If (I_0, s_0) is complete and its output is (I_0, s_0, ID) then by definition (I_0, s_0) matches (R_0, s_0) if and only if $ID = R_0$. Thus, we want to prove that if (I_0, s_0) completes then its peer $ID = I_0$.

Assume that (I_0, s_0) completes with peer ID. This means that in the response message received by (I_0, s_0) before the session completed, I_0 verified the value $\text{MAC}_{k_1}(\text{"1"}, s_0, ID)$ under $k_1 = \text{PRF}_k(1)$ where k is a random key chosen by $\hat{\mathcal{S}}$-HYBR and never provided to the attacker. At this point there could have been a single example of MAC_{k_1} use in the protocol (and no other use of k), namely, $\text{MAC}_{k_1}(\text{"1"}, s_0, R_0)$. Therefore, if based on this information the attacker has non-negligible probability of producing $\text{MAC}_{k_1}(\text{"1"}, s_0, ID)$ for $ID \neq R_0$ then, as in (i) above, we can build a forger for the MAC function or a distinguisher for the PRF family.

□

Lemma 12 *Lemma 11 holds for $\hat{\mathcal{S}}$-RAND as well.*

Proof: If in a run of a simulator $\hat{\mathcal{S}}$ the properties (i) and (ii) from Lemma 11 hold then we say that a MATCH event happened. Lemma 11 proves that under a run of $\hat{\mathcal{S}}$-HYBR, $Prob(\text{MATCH} : \text{GUESS}) \approx 1$ (i.e., 1 up to a negligible difference). Here we want to prove the same property under a run of $\hat{\mathcal{S}}$-RAND.

For this we build a DDH distinguisher $\mathcal{D}'$ as follows. $\mathcal{D}'$ runs $\mathcal{D}$ from the proof of Lemma 8 except that $\mathcal{D}'$ outputs 1 if and only if in the run of $\mathcal{D}$ a GUESS <u>and</u> a MATCH event happen (we will consider runs of $\mathcal{D}'$ both under $\hat{\mathcal{S}}$-HYBR and $\hat{\mathcal{S}}$-RAND). We have that:

$$
\begin{aligned}
&|Prob(\mathcal{D}' \text{ outputs } 1 : z = random) - Prob(\mathcal{D}' \text{ outputs } 1 : z = g^{xy})| = (\text{by Lemma 8}) \\
&= |Prob(\mathcal{D}' \text{ outputs 1 under } \hat{\mathcal{S}}\text{-HYBR}) - Prob(\mathcal{D}' \text{ outputs 1 under } \hat{\mathcal{S}}\text{-RAND})| = (\text{by def of } \mathcal{D}') \\
&= |Prob(\text{GUESS } \textit{and} \text{ MATCH under } \hat{\mathcal{S}}\text{-HYBR}) - Prob(\text{GUESS } \textit{and} \text{ MATCH under } \hat{\mathcal{S}}\text{-HYBR})| = \\
&= |Prob(\text{MATCH under } \hat{\mathcal{S}}\text{-HYBR} : \text{GUESS under } \hat{\mathcal{S}}\text{-HYBR}) Prob(\text{GUESS under } \hat{\mathcal{S}}\text{-HYBR}) -
\end{aligned}
$$

[5] Recall that the session key value from test session (I_0, s_0) provided to the attacker by $\hat{\mathcal{S}}$-HYBR is a random value independent from k.

$-Prob(\text{MATCH under } \hat{\mathcal{S}}\text{-RAND} : \text{GUESS under } \hat{\mathcal{S}}\text{-RAND})Prob(\text{GUESS under } \hat{\mathcal{S}}\text{-RAND})| \approx$ (Lemmas 10,11)
$\approx Prob(\text{GUESS under } \hat{\mathcal{S}}\text{-RAND})|1 - Prob(\text{MATCH under } \hat{\mathcal{S}}\text{-RAND} : \text{GUESS under } \hat{\mathcal{S}}\text{-RAND})| \geq$ (Lemma 9)
$\geq 1/(m \cdot n)Prob(\text{not MATCH under } \hat{\mathcal{S}}\text{-RAND} : \text{GUESS under } \hat{\mathcal{S}}\text{-RAND})$

That is, we have that up to a negligible probability:

$$Prob(\text{not MATCH under } \hat{\mathcal{S}}\text{-RAND} : \text{GUESS under } \hat{\mathcal{S}}\text{-RAND}) \leq$$
$$\leq (m \cdot n)|Prob(\mathcal{D}' \text{ outputs } 1 : z = random) - Prob(\mathcal{D}' \text{ outputs } 1 : z = g^{xy})|$$

Since by the DDH assumption the later expression is negligible then we have that under $\hat{\mathcal{S}}$-RAND: $Prob$(not MATCH : GUESS) is negligible, thus proving the lemma.

□

Lemma 13 *For all Σ_0-attackers $\mathcal{A}$, $P_{\text{RAND}}(\mathcal{A}) = Prob(\hat{\mathcal{S}}\text{-RAND}(\mathcal{A})$ outputs $1 :$ GUESS event)*

Proof: By the same argument as in the proof of Lemma 9 we get that under a run of $\hat{\mathcal{S}}$-RAND a GUESS event implies that $\hat{\mathcal{S}}$-RAND does not abort. Thus the run of $\mathcal{A}$ under $\hat{\mathcal{S}}$-RAND in the case of a GUESS event is the same as a regular run of $\mathcal{A}$ except that the secret key output by the sessions (I_0, s_0) and (R_0, s_0) (if completed) is not the real key but a random key k_0 chosen independently of the actual exchange in these sessions. In particular, this means that the value of the test session key provided to $\mathcal{A}$ under $\hat{\mathcal{S}}$-RAND is this random value k_0. On the other hand, the only other session that outputs k_0 is, by virtue of Lemma 12, a matching session to the test session so this value is never revealed to $\mathcal{A}$, and thus it makes no difference to $\mathcal{A}$'s view.

In summary, we have that in case of a GUESS event the output of $\hat{\mathcal{S}}$-RAND is exactly the output of $\mathcal{A}$ in a run where the test query is answered with a random key. Or, in other words:

$$Prob(\hat{\mathcal{S}}\text{-RAND}(\mathcal{A}) \text{ outputs } 1 : \text{GUESS } event) =$$
$$= Prob(\mathcal{A} \text{ outputs } 1 \text{ under a } \hat{\mathcal{S}}\text{-RAND run} : \text{GUESS } event) =$$
$$= Prob(\mathcal{A} \text{ outputs } 1 \text{ under a regular run with test query answered with a random key}) =$$
$$= P_{\text{RAND}}(\mathcal{A})$$

□

Lemma 14 *For all Σ_0-attackers $\mathcal{A}$, $P_{\text{REAL}}(\mathcal{A}) = Prob(\hat{\mathcal{S}}\text{-REAL}(\mathcal{A})$ outputs $1 :$ GUESS event)*

Proof: This is analogous to Lemma 13. The proof is similar too and it involves the proof of Lemmas 8 to 12 with the role of $\hat{\mathcal{S}}$-RAND replaced with $\hat{\mathcal{S}}$-REAL and the role of $\hat{\mathcal{S}}$-HYBR replaced with $\hat{\mathcal{S}}$-RPRF. The proofs of these lemmas require just minor and straightforward adaptations to the above simulators and are omitted. □

Lemma 15 *For all Σ_0-attackers $\mathcal{A}$, $\hat{\mathcal{S}}\text{-REAL}(\mathcal{A}) \approx \hat{\mathcal{S}}\text{-RAND}(\mathcal{A})$*

Proof: We prove that $\hat{\mathcal{S}}$-REAL $\approx$ $\hat{\mathcal{S}}$-RPRF $\approx$ $\hat{\mathcal{S}}$-ALLR $\approx$ $\hat{\mathcal{S}}$-HYBR $\approx$ $\hat{\mathcal{S}}$-RAND (for all Σ_0-attackers).

The indistinguishability of $\hat{\mathcal{S}}$-HYBR and $\hat{\mathcal{S}}$-RAND is proven in Lemma 8. The proof of $\hat{\mathcal{S}}$-REAL $\approx$ $\hat{\mathcal{S}}$-RPRF is similar; the only difference being that k_0 in $\hat{\mathcal{S}}$-RPRF is computed via the pseudorandom function rather than chosen at random. However, this does not change the validity of the argument in the proof of Lemma 8.

For proving $\hat{\mathcal{S}}$-RPRF $\approx \hat{\mathcal{S}}$-ALLR one uses the following standard argument based on the security of the pseudorandom function family PRF. Let $\mathcal{A}$ be a Σ_0-attacker; based on $\mathcal{A}$ we build a distinguisher $\mathcal{D}$ against the family PRF as follows. The distinguisher $\mathcal{D}$ has oracle access to a function F (which may have been selected truly randomly or as a random member of PRF); $\mathcal{D}$ works exactly as a $\hat{\mathcal{S}}$-RPRF($\mathcal{A}$) simulator, except that for computing k_0 and k_1 it uses the oracle F rather than a randomly selected function from the PRF family. It is clear that if F itself is implemented via a random member of PRF then the actions of $\mathcal{D}^F$ are identical to those of $\hat{\mathcal{S}}$-RPRF($\mathcal{A}$). On the other hand, if F is a truly random function the actions of $\mathcal{D}$ are identical to those of $\hat{\mathcal{S}}$-ALLR. Therefore, we have that

$$\begin{aligned}&|Prob(\hat{\mathcal{S}}\text{-RPRF}(\mathcal{A}) \text{ outputs } 1) - Prob(\hat{\mathcal{S}}\text{-ALLR}(\mathcal{A}) \text{ outputs } 1)| = \\ &= |Prob(\mathcal{D}^F \text{ outputs } 1 : F \text{ is pseudorandom}) - Prob(\mathcal{D}^F \text{ outputs } 1 : F \text{ is random})|\end{aligned}$$

Since the PRF family is secure then the last difference is negligible and therefore also $|Prob(\hat{\mathcal{S}}\text{-RPRF}(\mathcal{A}) \text{ outputs } 1) - Prob(\hat{\mathcal{S}}\text{-ALLR}(\mathcal{A}) \text{ outputs } 1)|$ is negligible, i.e., $\hat{\mathcal{S}}$-RPRF $\approx \hat{\mathcal{S}}$-ALLR.

For proving $\hat{\mathcal{S}}$-ALLR $\approx \hat{\mathcal{S}}$-HYBR one uses a similar argument as in the previous case where the oracle F replaces PRF in the choice of k_1 while k_0 is chosen at random and independently. □

We are finally able to complete the proof of property P2 for protocol Σ_0.

Theorem 16 *Protocol Σ_0 satisfies condition P2 of SK-security: for all Σ_0-attacker $\mathcal{A}$, $|P_{\text{REAL}}(\mathcal{A}) - P_{\text{RAND}}(\mathcal{A})|$ is negligible.*

Proof:

$$\begin{aligned}&Prob(\hat{\mathcal{S}}\text{-RAND}(\mathcal{A}) \text{ outputs } 1) = \\ &= Prob(\hat{\mathcal{S}}\text{-RAND}(\mathcal{A}) \text{ outputs } 1 : \text{GUESS } \textit{event})\ Prob(\text{GUESS under } \hat{\mathcal{S}}\text{-RAND}) + \\ &\quad + Prob(\hat{\mathcal{S}}\text{-RAND}(\mathcal{A}) \text{ outputs } 1 : \textit{not} \text{ GUESS } \textit{event})\ Prob(\textit{not} \text{ GUESS }) = \\ &= Prob(\hat{\mathcal{S}}\text{-RAND}(\mathcal{A}) \text{ outputs } 1 : \text{GUESS } \textit{event})\ Prob(\text{GUESS under } \hat{\mathcal{S}}\text{-RAND}) \geq \\ &\geq Prob(\hat{\mathcal{S}}\text{-RAND}(\mathcal{A}) \text{ outputs } 1 : \text{GUESS } \textit{event})\ 1/(m \cdot n)\end{aligned}$$

The second equality is due to the fact that if a GUESS event does *not* happen then necessarily $\hat{\mathcal{S}}$-RAND outputs 0; while the last inequality is from Lemma 9.

Similarly (using the analogous of Lemma 9 in the case of $\hat{\mathcal{S}}$-REAL) we have that $Prob(\hat{\mathcal{S}}\text{-REAL}(\mathcal{A}) \text{ outputs } 1) \geq Prob(\hat{\mathcal{S}}\text{-REAL}(\mathcal{A}) \text{ outputs } 1 : \text{GUESS } \textit{event})\ 1/(m \cdot n)$

From Lemma 13 and Lemma 14 we have that:

$$\begin{aligned}&|P_{\text{REAL}}(\mathcal{A}) - P_{\text{RAND}}(\mathcal{A})| = \\ &= |Prob(\hat{\mathcal{S}}\text{-REAL}(\mathcal{A}) \text{ outputs } 1 : \text{GUESS } \textit{event}) - Prob(\hat{\mathcal{S}}\text{-RAND}(\mathcal{A}) \text{ outputs } 1 : \text{GUESS } \textit{event})| \leq \\ &\leq (m \cdot n)|Prob(\hat{\mathcal{S}}\text{-REAL}(\mathcal{A}) \text{ outputs } 1) - Prob(\hat{\mathcal{S}}\text{-RAND}(\mathcal{A}) \text{ outputs } 1)|\end{aligned}$$

Since $|Prob(\hat{\mathcal{S}}\text{-REAL}(\mathcal{A}) \text{ outputs } 1) - Prob(\hat{\mathcal{S}}\text{-RAND}(\mathcal{A}) \text{ outputs } 1)|$ is negligible (Lemma 15) so is $|P_{\text{REAL}}(\mathcal{A}) - P_{\text{RAND}}(\mathcal{A})|$. □

Remark 3 When doing an exact quantification of the above analysis of the security of protocol Σ_0, one can see that the main "degradation factor" of the security of the protocol with respect to the security of the underlying cryptographic functions, is the factor $m \cdot n$, namely the number of sessions initiated by $\mathcal{A}$ in its run times the number of parties in the protocol. If one thinks of key-exchange protocols that run over the Internet then the numbers for m and n may be huge and then this factor may seem as a prohibitive loss of security. However, for any given attacker that breaks the protocol the real meaning of m and n is not the maximal potential number of sessions or parties in the Internet but just the minimal number of sessions and parties necessary to develop the attack. These numbers are usually very small (single-digit numbers in currently known attacks). Therefore, for such attacks the security of the protocol is related to the security of the underlying cryptographic functions by a very small (usually constant) degradation factor.

5 Variants and Discussions

At this point we have a full analysis of protocol Σ_0. 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.)

The lemmas where we have used these tags as part of the proof arguments are Lemma 7 and Lemma 11. Here we show how to modify these arguments in order for these lemmas to hold also for $\Sigma_0{}'$.

Proof of Lemma 7, part (a)(3). In this case we used the tag "1" included under the signature of Q to argue that $\text{SIG}_Q(\text{"1"}, s, g^x, g^y)$ received by P in the response message of session s is the only signature that Q could have produced under session s and then the response message must have come from Q. However, if we omit "1" from this signature then this claim is not necessarily correct. In this case the signature received by P in the response message is $\text{SIG}_Q(s, g^x, g^y)$ which could have been taken from a finish message sent by Q in a session (Q, s) where Q was activated under session-id s as initiator! In the later case, however, we know that before sending the finish message with the above signature (Q, s) should have received a valid response message which included a legal signature $\text{SIG}_E(s, g^y, g^x)$ from some party E, as well as a corresponding $\text{MAC}_{k_1}(s, E)$ for $k_1 = \text{PRF}_{g^{xy}}(1)$. Since we know that g^y was chosen by Q itself (it appears as last element in the signature) and g^x was chosen by P, then no uncorrupted party could have chosen g^x except for negligible (collision) probability. Moreover, E created $\text{MAC}_{k_1}(s, E)$ for k_1 computed under random g^x, g^y not chosen by E (nor could x and y be found by $\mathcal{A}$ via session reveals since by parts (a)(2) of the lemma none of the sessions (P, s), (Q, s) could be revealed).

From this we have that if the event in which E produces a valid response message for session (Q, s) happens with non-negligible probability then we can build a DDH distinguisher in a similar way to the proof of Lemma 8; this distinguisher just needs to guess the sessions (Q, s) and (P, s) where this attack by E happens. In the guessed sessions the distinguisher uses z from the DDH triple to compute $k_1 = \text{PRF}_z(1)$. Now if $z = g^{xy}$ then the probability of forgery by E is non-negligible as assumed above. On the other hand, if the probability of forgery is non-negligible with $z = random$ then we can build a breaker to the MAC or PRF as done in the proof of Lemma 11. So either we contradict the DDH assumption, or the security of the MAC or PRF functions.

(Note that this proof involves signature security considerations as well as DDH, MAC and PRF; the proof of this property in the case of Σ_0 used a signature-only argument; this shows the simplifying effect for the analysis that the use of a responder's tag has.)

Proof of Lemma 7, part (a)(4). This requires changes to the argument in Lemma 7 which are very similar to the case of part (a)(3) proved above.

Proof of Lemma 11, part (i). The proof of this part of Lemma 11 for protocol Σ_0 used in an essential way the tags "0" and "1" included under the MAC; otherwise the attacker could have replayed in the finish message the value $\text{MAC}_{k_1}(s_0, R_0)$ taken from the response message by R_0. However, we will show that the proof can be adapted to the case of $\Sigma_0{}'$ where the tags are not included. First note that the current argument in the proof of Lemma 11, part (i) already shows (even without the tags) that it is not feasible for the attacker to make (R_0, s_0) complete with ID other than I_0 or R_0 (since the only available MAC values are on I_0 and R_0).

Thus, if (I_0, s_0) completes with peer $R_0 \neq I_0$ then we need to use a signature-based argument to show that replaying $\text{MAC}_{k_1}(s_0, R_0)$ in the finish message does not help. Indeed, this finish message (in order to be valid with $ID = R_0$) will also have to carry a signature $\text{SIG}_{R_0}(s_0, g^y, g^x)$ where g^x is the DH exponent received by (R_0, s_0) in the start message. On the other hand, the only signature produced by R_0 in session s_0 is $\text{SIG}_{R_0}(s_0, g^x, g^y)$, where g^y was chosen by R_0 itself after receiving g^x and independently of this value. Therefore, except for a negligible collision probability, $g^x \neq g^y$ and $\text{SIG}_{R_0}(s_0, g^y, g^x)$ was never produced by R_0. Thus, if such a valid signature appears in the finish message received by (R_0, s_0) then we have a forgery event against SIG_{R_0} which can only happen with negligible probability or otherwise we have a forgery algorithm against the signature scheme.

Proof of Lemma 11, part (ii). No change required, the presence of the tags was not used in the proof argument of this part for protocol Σ_0.

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 $\text{SIG}_r(\text{“1”}, s, g^x, g^y), \text{MAC}_{k_1}(\text{“1”}, s, ID_r)$, as in Σ_0, but rather sends the value $\text{SIG}_r(\text{MAC}_{k_1}(s, g^x, g^y, ID_r))$. Similarly, the pair of signature-mac is replaced in the finish message by the value $\text{SIG}_i(\text{MAC}_{k_1}(s, g^y, g^x, ID_i))$. 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.[6]

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 17 *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 $\text{SIG}(\text{MAC}_{k_1}(M'))$ even after seeing $\text{SIG}(\text{MAC}_{k_1}(M))$.*

[6]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).

Indeed, if the attacker can do that then either $\text{MAC}_{k_1}(M') \neq \text{MAC}_{k_1}(M)$ with significant probability and this results in a signature forgery strategy, or $\text{MAC}_{k_1}(M') = \text{MAC}_{k_1}(M)$ 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 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 6. 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 6. The main change regards the formulation of the hybrid simulators in Section 4.3.2. Specifically, to the specification of how these simulators choose k_0 and k_1 we now add the choice of a third key k_2. In the case of $\hat{\mathcal{S}}$-ALLR k_2 is chosen at random and independently of k_0 and k_1; in all other cases k_2 is chosen by applying the PRF to the value 2 and with the same key used to derive k_1 (e.g. $\hat{\mathcal{S}}$-REAL will choose $k_2 = \text{PRF}_{g^{xy}}(2)$). The proofs of lemmas in Section 4.3.3 now need to be augmented with an extra simulation of the encryption function under a random key. Any deviation from the attacker's advantage from the non-encryption case results in a break of DDH (i.e., a construction of a distinguisher to the DDH assumption) or a break to the PRF family (i.e., a construction of a distinguisher against this family).

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 (as under $\hat{\mathcal{S}}$-ALLR). 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. Using the indistinguishability between $\hat{\mathcal{S}}$-ALLR and $\hat{\mathcal{S}}$-REAL (re-proven as sketched before for the case of encrypted Σ_0) we get a guarantee of semantic security also under the real runs of the protocol (as represented by $\hat{\mathcal{S}}$-REAL).

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 [4, 16]). 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. The protocol is:

$I \to R$: s, g^x

$R \to I$: s, g^y

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

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

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. In particular, this requires some changes to the definition of the "abort events" related to the $\hat{\mathcal{S}}$-simulators from Section 4.3.2 and the statement of Lemma 7. 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 (for example, in Lemma 7 item 4) cannot be argued directly but via a binding of these exponents to the session id.

5.6 Hashing g^{xy}: the HDH assumption

We mentioned in Section 3 that it is advisable in practice to hash the DH value g^{xy} to the length of the PRF's key from which further keys are derived. In particular, this may result in better security of the resultant hashed bits relative to the initial plain string g^{xy}. The use of all of g^{xy} as if they were all perfectly random is generally justified by the DDH assumption (see Assumption 5). However, while this assumption is considered "standard" these days, it actually constitutes a very strong conjecture about the strength of the DH key g^{xy}: namely, that *all* bits in this key are simultaneously indistinguishable from random for an observer of g^x and g^y. Currently, there is no evidence against this strong conjecture, yet, whenever possible, it is best to rely on weaker assumptions. A possible weakening of DDH is to assume the indistinguishability of the distributions Q_0 and Q_1 defined in Assumption 5 when the values g^{xy} and g^z are replaced with $h(g^{xy})$ and $h(g^z)$, respectively; where h is a randomly chosen element from a family of hash functions (such as a cryptographic hash function family or universal hash functions). This approach was recently taken in [8] where this weaker assumption is referred to as the "Hashed Diffie-Hellman Assumption (HDH)". We point out that the IKE protocols use a key derivation technique from g^{xy} based on this approach with the "hashing" implemented via a family of pseudorandom functions. A more common practice is to just use a single (idealized) hash function H (such as SHA-1) to hash the DH key.

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., "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`.

[5] 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`.

[6] 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.

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

[8] Gennaro, R., Krawczyk H., and Rabin, T., "Hashed Diffie-Hellman: A Hierarchy of Diffie-Hellman Assumptions", manuscript, Feb 2002.

[9] O. Goldreich, "Foundations of Cryptography: Basic Tools", Cambridge Press, 2001.

[10] D. Harkins and D. Carrel, ed., "The Internet Key Exchange (IKE)", *RFC 2409,* Nov. 1998.

[11] ISO/IEC IS 9798-3, "Entity authentication mechanisms — Part 3: Entity authentication using asymmetric techniques", 1993.

[12] Karn, P., and Simpson W.A., "The Photuris Session Key Management Protocol", draft-ietf-ipsec-photuris-03.txt, Sept. 1995.

[13] S. Kent and R. Atkinson, "Security Architecture for the Internet Protocol", *Request for Comments 2401*, Nov. 1998.

[14] 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. `http://www.ee.technion.ac.il/~hugo/skeme-lncs.ps`

[15] Krawczyk, H., *IPsec mailing list archives*, `http://www.vpnc.org/ietf-ipsec/`, April-June 1995.

[16] 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.

[17] Krawczyk, H., "SIGMA: the 'SIGn-and-MAc' Approach to Authenticated Diffie-Hellman Protocols", `http://www.ee.technion.ac.il/~hugo/sigma.html`

[18] 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.

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

[20] Orman, H., "The OAKLEY Key Determination Protocol", *Request for Comments 2412*, Nov. 1998.

[21] V. Shoup, "On Formal Models for Secure Key Exchange", Theory of Cryptography Library, 1999. Available at: http://philby.ucsd.edu/cryptolib/1999/99-12.html.

A On The Universal Composability of Protocol Σ_0 (preliminary version)

An alternative way for defining the security requirements from key exchange protocols is via the framework of universally composable (UC) security [3]. Placing the [4] notion of SK-security within the UC framework was done in [5]. We present a UC definition of secure key exchange in the post-specified peer setting, and show that protocol Σ_0 presented here satisfies this definition. We also argue that the UC notion suffices for realizing secure channels via standard protocols, and that it implies the notion of SK security in the post-specified peer setting (Definition 3 above).

One advantage of working in the UC framework is that it guarantees strong composability guarantees with arbitrary protocols. Another advantage is that the presentation and analysis of protocols can be done in a simplified setting where only a single generation of a key takes place between two parties. Security in a general setting where multiple keys are generated in multiple "pairwise sessions" among many pairs of parties is guaranteed via general composition theorems. This holds even when all "pairwise sessions" use the same instance of the signature scheme. See [5] for more details.

The presentation below assumes familiarity with the UC framework and its use for defining security for key-exchange protocols. It also assumes familiarity with the ideal signature functionality, $\mathcal{F}_{\text{SIG}}$. All this preliminary material can be found in Section 3 in [5]. (For self containment, functionality $\mathcal{F}_{\text{SIG}}$ is presented in Figure 3.) In this section we use I to denote the identity of the initiator, and use R to denote the identity of the responder.

A.1 Universally Composable Key Exchange with Post-Specified Peers

We present a UC notion of secure key exchange in the post-specified peer setting. This is done be presenting an ideal key exchange functionality that is aimed at capturing the fact that the peer identity is not known upon protocol invocation, but it becomes known via the protocol and is part of the output. This functionality, denoted $\mathcal{F}_{\text{POST-KE}}$, is presented in Figure 2. Several remarks on the formulation of $\mathcal{F}_{\text{POST-KE}}$ follow:

Functionality $\mathcal{F}_{\text{POST-KE}}$

$\mathcal{F}_{\text{POST-KE}}$ proceeds as follows, running on security parameter k. The symbols I, R, P below indicate arbitrary identities of parties.

1. Upon receiving a value $(\texttt{Establish-session}, I, s, aux)$ from the first party, where I is the identity of that party, send (s, I, aux) to the adversary. Upon receiving a value $(\texttt{Establish-session}, R, s, aux)$ from the second party, where R is the identity of that party, send (s, R, aux) to the adversary; then, choose a value $\kappa \stackrel{R}{\leftarrow} \{0,1\}^k$ and continue to the next step.
2. (a) Upon receiving a value $(\texttt{Output}, s, I, P, \kappa')$ from the adversary, proceed as follows. If both parties are uncorrupted at this point then ignore (P, κ') and send $(\texttt{Output}, s, R, \kappa)$ to I. If either party is corrupted then send $(\texttt{Output}, s, P, \kappa')$ to I, unless P is an identity of an uncorrupted party (in which case do nothing).

   (b) Upon receiving a value $(\texttt{Output}, s, R, P, \kappa')$ from the adversary, proceed as follows. If both parties are uncorrupted at this point then ignore (P, κ') and send $(\texttt{Output}, s, I, \kappa)$ to R. If either party is corrupted then send $(\texttt{Output}, s, P, \kappa')$ to R, unless P is an identity of an uncorrupted party (in which case do nothing).
3. If the adversary corrupts a party after κ is chosen and before κ is sent to that party, then hand κ to the adversary. Otherwise provide no information to the adversary.

Figure 2: The Post-Specified Peer Key Exchange functionality

1. The interaction takes place among an unbounded number of parties, whose identities are not known to $\mathcal{F}_{\text{POST-KE}}$ in advance. Still, $\mathcal{F}_{\text{POST-KE}}$ interacts only with two parties, whose identities become known when the inputs arrive. Recall that the environment determines the identities of the parties, as well as the inputs of $\mathcal{F}_{\text{POST-KE}}$. In particular, $\mathcal{Z}$ can determine the identities based on information gathered in other protocol executions, etc.
2. The peer identities are not part of the inputs. Nonetheless, they appear as part of the outputs of both parties. This means that the parties learn the peer identities as part of the protocol execution.
3. When one of the two parties is corrupted, the adversary is allowed to set the peer identity in the output of the other party to any arbitrary value, under the condition that this value is not an identity of an existing and uncorrupted party. This last provision makes sure that the adversary cannot "impersonate" other uncorrupted parties. (Technically, we assume that the ideal process allows $\mathcal{F}_{\text{POST-KE}}$ to know the identities of all uncorrupted parties.)

4. Each party has an input field aux, in addition to its own identity and the session identifier. This field represents arbitrary additional information that may help the protocol execution; however it does not play a role in the security requirements. Protocol Σ_0 will use this field to differentiate between the initiator and the responder roles. In addition, this field may be used to incorporate some routing information for message delivery, etc.

5. Functionality $\mathcal{F}_{\text{POST-KE}}$ allows the adversary to learn the session key κ only if it corrupts a party *before* the output message is sent to that party. Once the output message is sent, the adversary does not learn κ, even if the party is corrupted. This reflects the perfect forward secrecy requirement (see [5]). Naturally, the functionality can be relaxed to capture protocols that guarantee only restricted versions of forward secrecy, or no forward secrecy at all.

6. If the initiator is corrupted after it has generated output but before the responder generated output, then $\mathcal{F}_{\text{POST-KE}}$ allows $\mathcal{S}$ to control the output of the responder. We note that protocol Σ_0 actually provides a somewhat stronger guarantee: The protocol guarantees that, in this case, the responder always outputs the same value as the initiator. Still we choose not to enforce this requirement in $\mathcal{F}_{\text{POST-KE}}$, since it is not necessary for the main application, namely realizing secure channels.

7. $\mathcal{F}_{\text{POST-KE}}$ explicitly sends the identities I and R to the adversary. This reflects the fact that identity hiding is not guaranteed. Requiring that the identities of the parties remain unknown to the adversary (unless ofcourse it corrupts one of the parties) can be captured by modifying $\mathcal{F}_{\text{POST-KE}}$ so that the messages to the adversary in Step 1 will not include the identities.

A.2 Protocol Σ_0 securely realizes $\mathcal{F}_{\text{POST-KE}}$

We start by re-formulating protocol Σ_0 in the $\mathcal{F}_{\text{SIG}}$-hybrid model (i.e., in the hybrid model with access to the ideal signature functionality). See Figure 4. For self containment, we also recall the signature functionality, $\mathcal{F}_{\text{SIG}}$, in Figure 3. We then show:

Theorem 18 *Protocol Σ_0 securely realizes $\mathcal{F}_{\text{POST-KE}}$ in the $\mathcal{F}_{\text{SIG}}$-hybrid model.*

Proof: Let $\mathcal{A}$ be an adversary in the $\mathcal{F}_{\text{SIG}}$-hybrid model. We construct an ideal-process adversary (i.e., a simulator) $\mathcal{S}$ such that no environment $\mathcal{Z}$ can tell whether it is interacting with $\mathcal{A}$ and parties running Σ_0 in the $\mathcal{F}_{\text{SIG}}$-hybrid model, or with $\mathcal{S}$ in the ideal process for $\mathcal{F}_{\text{POST-KE}}$. (Recall that the interaction takes place between only two parties, I and R, whose identities are a-priori known only to $\mathcal{Z}$. In fact, we can assume that $\mathcal{Z}$ chooses these identities adaptively during the interaction.)

Simulator $\mathcal{S}$ runs a simulated copy of $\mathcal{A}$, and simulates for $\mathcal{A}$ an interaction with parties running (a single pairwise session of) Σ_0. This is done with the exception that the generated keys and the peer identities are the values received by the parties from $\mathcal{F}_{\text{POST-KE}}$ in the ideal process rather than the values agreed in the simulated protocol execution. More precisely, $\mathcal{S}$ proceeds as follows.

1. **Communication with the environment:** Any input from $\mathcal{Z}$ is forwarded to $\mathcal{A}$. Any output of $\mathcal{A}$ is copied to the output of $\mathcal{S}$ (to be read by $\mathcal{Z}$).

2. **Simulating the initial activation of an uncorrupted I.** When receiving $(s, I, \text{“}init\text{”})$ from $\mathcal{F}_{\text{POST-KE}}$, $\mathcal{S}$ feeds $\mathcal{A}$ with a Start message (s, g^x) sent by I, where x is chosen randomly by $\mathcal{S}$. In addition, $\mathcal{S}$ feeds $\mathcal{A}$ with a message `(signer`, “0” $\circ\, s, I$`)` from $\mathcal{F}_{\text{SIG}}$ (representing the fact that I registered with $\mathcal{F}_{\text{SIG}}$ for the appropriate session identifier).

Functionality $\mathcal{F}_{\text{SIG}}$

$\mathcal{F}_{\text{SIG}}$ proceeds as follows, running with an unbounded number of parties and an adversary.

Set-up: In the first activation, expect to receive a value $(\texttt{signer}, sid)$ from some party S. (Note that S may be a corrupted party.) Then, send $(\texttt{signer}, sid, S)$ to the adversary. From now on, ignore all $(\texttt{signer}, sid)$ values. (That is, the functionality serves a single signer.)

Signature generation: Upon receiving a value $(\texttt{sign}, sid, m)$ from S, hand $(\texttt{sign}, sid, m)$ to the adversary. Upon receiving $(\texttt{signature}, sid, m, \sigma)$ from the adversary, set $s_m = \sigma$, send $(\texttt{signature}, sid, m, \sigma)$ to S, and request the adversary to deliver this message immediately. Save the pair (m, s_m) in memory.

Signature verification: Upon receiving a value $(\texttt{verify}, sid, S', m, \sigma)$ from some party V, do:

1. If $S' = S$ (i.e., if the signer identity in the verification request agrees with the identity of the actual signer) then do: If m was never before signed then let $v = 0$. If m was signed before (i.e., s_m is defined) and $\sigma = s_m$ then let $v = 1$. If m was signed but $s_m \neq \sigma$ then let the adversary decide on the value of v. (That is, hand $(\texttt{verify}, sid, V, S', m, \sigma)$ to the adversary. Upon receiving $\phi \in \{0, 1\}$ from the adversary, let $v = \phi$.)
2. If $S' \neq S$ then do: If S' is uncorrupted then set $v = 0$. Otherwise, let the adversary decide on the value of v, as in Step 1.
3. Once the value of v is set, send $(\texttt{verified}, sid, m, v)$ to V, and request the adversary to deliver this message immediately.

Figure 3: The signature functionality, $\mathcal{F}_{\text{SIG}}$.

3. **Simulating the initial activation of an uncorrupted R.** When receiving $(s, R, \text{"}resp\text{"})$ from $\mathcal{F}_{\text{POST-KE}}$, $\mathcal{S}$ feeds $\mathcal{A}$ with a message $(\texttt{signer}, \text{"1"} \circ s, R)$ from $\mathcal{F}_{\text{SIG}}$ (representing the fact that R registered with $\mathcal{F}_{\text{SIG}}$ for the appropriate session identifier).

4. **Simulating receipt of a Start message by and uncorrupted R.** When $\mathcal{A}$ delivers a Start message (s, α) to R, $\mathcal{S}$ first verifies that in the ideal process it has received a message $(s, R, \text{"}resp\text{"})$ from $\mathcal{F}_{\text{POST-KE}}$ (indicating that R was activated to exchange a key as a responder). Next, $\mathcal{S}$ chooses y randomly, and feeds $\mathcal{A}$ with a Response message $(s, g^y, R, \sigma_r, t_r)$ from R. Here $t_r = \text{MAC}_{k_1}(\text{"1"}, s, R)$, $k_1 = \text{PRF}_{\alpha^y}(1)$, and σ_r is a signature obtained be handing $\mathcal{A}$ the message $(\texttt{sign}, \text{"1"} \circ s, \alpha, g^y)$ in the name of $\mathcal{F}_{\text{SIG}}$, and setting σ_r to the value returned by $\mathcal{A}$.

5. **Simulating receipt of a Response message by an uncorrupted I.** When $\mathcal{A}$ delivers a Response message $(s, \beta, P, \sigma_r, t_r)$ to an uncorrupted I, $\mathcal{S}$ proceeds as follows:

   (a) $\mathcal{S}$ verifies that in the simulation I has previously sent a Start message (s, g^x).

   (b) $\mathcal{S}$ mimics the verification process of σ_r, by mimicking the behavior of $\mathcal{F}_{\text{SIG}}$ on input $(\texttt{verify}, \text{"1"} \circ s, I, P, (g^x, \beta), \sigma_r)$ from I. (That is, If $P = R$, then verification succeeds if σ_r was previously generated by $\mathcal{A}$ in response to a request, generated by $\mathcal{S}$ in the name of $\mathcal{F}_{\text{SIG}}$, of the form $(\texttt{sign}, \text{"1"} \circ s, g^x, \beta)$. If $P \neq R$ is an identity of an existing uncorrupted party then verification fails. Otherwise, $\mathcal{S}$ feeds $\mathcal{A}$ with a message $(\texttt{verify}, \text{"1"} \circ s, I, P, (g^x, \beta), \sigma_r)$ in the name of $\mathcal{F}_{\text{SIG}}$, and accepts σ_r if $\mathcal{A}$ says to.)

Protocol Σ_0

Initial information: Primes p, q, $q/p{-}1$, and g of order q in Z_p^*. The players have access to multiple copies of the ideal signature functionality $\mathcal{F}_{\text{SIG}}$. The protocol also uses a message authentication function MAC, and a pseudorandom function family PRF.

The protocol actions

1. Upon activation with input (`Establish-session`, s, I, "*init*") the party learns that it is an initiator with identity I. It then sends the Start message (s, g^x), where the DH exponent g^x is computed with $x \stackrel{R}{\leftarrow} Z_q$ and x is stored in the state of session (I, s).

   In addition, I initializes a copy of $\mathcal{F}_{\text{SIG}}$ with session identifier "0" $\circ\, s$, by sending a message (`signer`, "0" $\circ\, s$) to $\mathcal{F}_{\text{SIG}}$.

2. When activated with input (`Establish-session`, s, R, "*resp*"), the party learns that it is a responder with identity R. It then initializes a copy of $\mathcal{F}_{\text{SIG}}$ with session identifier "1" $\circ\, s$ by sending a message (`signer`, "1" $\circ\, s$) to $\mathcal{F}_{\text{SIG}}$, and waits for delivery of a Start message.

   When a Start message (s, g^x) is delivered, R generates the response message $s, g^y, R, \sigma_r, \text{MAC}_{k_1}(\text{"1"}, s, R)$, where the DH exponent g^y is computed with $y \stackrel{R}{\leftarrow} Z_q$, the signature σ_r is computed by sending (`sign`, "1" $\circ\, s, g^x, g^y$) to $\mathcal{F}_{\text{SIG}}$ and recording the returned value, and $k_1 = \text{PRF}_{g^{xy}}(1)$. (The value g^{xy} is computed by R as $(g^x)^y$.) Next a value $k_0 = \text{PRF}_{g^{xy}}(0)$ is computed and kept in memory, and the values y and g^{xy} are erased.

3. Upon receiving the response message $(s, g^y, R, \sigma_r, t_r)$, I first verifies the signature σ_r by sending (`verify`, "1" $\circ\, s, R, (g^x, g^y), \sigma_r$) to $\mathcal{F}_{\text{SIG}}$. I also verifies that $t_r = \text{MAC}_{k_1}(\text{"1"}, s, R)$, where $k_1 = \text{PRF}_{g^{xy}}(1)$ and g^{xy} is computed as $(g^y)^x$. If any of these verification steps fails the session is aborted, and the session state is erased. If verification succeeds then I sends the finish message $(s, I, \sigma_i, \text{MAC}_{k_1}(\text{"0"}, s, I))$ (where the signature σ_i is computed by sending (`sign`, "0" $\circ\, s, g^y, g^x$) to $\mathcal{F}_{\text{SIG}}$ and recording the obtained value), completes the session with local output (`Output`, s, R, k_0) where $k_0 = \text{PRF}_{g^{xy}}(0)$, and erases the session state.

4. Upon receiving the finish message s, I, σ_i, t_i, R verifies the signature by sending (`verify`, "0" $\circ$ $s, I, (g^y, g^x), \sigma_i$) to $\mathcal{F}_{\text{SIG}}$, where g^y is the DH value received from R in the response message, and verifies that $t_i = \text{MAC}_{k_1}(\text{"0"}, s, I)$. If any of the verifications steps fails the session is aborted, otherwise R completes the session with local output (`Output`, s, I, k_0) where $k_0 = \text{PRF}_{g^{xy}}(0)$, and erases the session state.

Figure 4: The basic SIGMA protocol, in the $\mathcal{F}_{\text{SIG}}$-hybrid model

(c) $\mathcal{S}$ verifies that $t_r = \text{MAC}_{k_1}(\text{"1"}, s, P)$, where $k_1 = \text{PRF}_{\beta^x}(1)$.

(d) If all verifications succeed then $\mathcal{S}$ feeds $\mathcal{A}$ with a Finish message $(s, I, \sigma_i, \text{MAC}_{k_1}(\text{"0"}, s, I))$ sent by I, where the signature σ_i is set to $\mathcal{A}$'s response after being handed (`sign`, "0" $\circ$ s, β, g^x) in the name of $\mathcal{F}_{\text{SIG}}$.

   In addition $\mathcal{S}$ sends, in the ideal-process interaction, the message (`Output`, $s, I, (P, \kappa')$) to $\mathcal{F}_{\text{POST-KE}}$, where $\kappa' = \text{PRF}_{\beta^x}(0)$. Once $\mathcal{F}_{\text{POST-KE}}$ sends the output message to I, $\mathcal{S}$ delivers this message.

6. **Simulating receipt of a Finish message by an uncorrupted R.** When $\mathcal{A}$ delivers a Finish message (s, P, σ_i, t_i) to R, $\mathcal{S}$ proceeds as follows:

(a) $\mathcal{S}$ verifies that in the simulation R has previously received a Start message (s, α) and has sent a Response message $(s, g^y, R, \sigma_r, t_r)$.

(b) $\mathcal{S}$ mimics the verification process of σ_r, by mimicking the behavior of $\mathcal{F}_{\text{SIG}}$ on input $(\texttt{verify}, \text{"0"} \circ s, R, P, (g^y, \alpha), \sigma_i)$ from R. (This is done as in Step 5b.) Next $\mathcal{S}$ verifies that $t_i = \text{MAC}_{k_1}(\text{"0"}, s, P)$, where $k_1 = \text{PRF}_{\alpha^y}(1)$.

(c) If all verifications succeed then $\mathcal{S}$ sends, in the ideal-process interaction, the message $(\texttt{Output}, s, I, (P, \kappa'))$ to $\mathcal{F}_{\text{POST-KE}}$, where $\kappa' = \text{PRF}_{\alpha^y}(0)$. Once $\mathcal{F}_{\text{POST-KE}}$ sends the output message to R, $\mathcal{S}$ delivers this message.

7. **Simulating party corruptions.** If $\mathcal{A}$ corrupts either I or R then $\mathcal{S}$ corrupts the same party in the ideal process and hands $\mathcal{A}$ the internal data of that party. Specifically:

   (a) If I is corrupted after the Start message is sent but before the Response message is received then $\mathcal{S}$ hands $\mathcal{A}$ the secret exponent x from the simulation.

   (b) If R is corrupted after the Response message is sent then but before the Finish message is received then $\mathcal{S}$ hands $\mathcal{A}$ the value k_1 computed in Step 4, together with the session key obtained from $\mathcal{F}_{\text{POST-KE}}$.

   (c) If I is corrupted after the Finish message is sent, or if R is corrupted after the Finish message is received, then all internal state of the corrupted party should be erased and $\mathcal{A}$ obtains nothing.

8. **Simulating $\mathcal{F}_{\text{SIG}}$ for $\mathcal{A}$.** $\mathcal{S}$ simulates $\mathcal{F}_{\text{SIG}}$ for $\mathcal{A}$, in the natural way. That is, whenever $\mathcal{A}$ generates (in the name of a corrupted party) some message to $\mathcal{F}_{\text{SIG}}$, $\mathcal{S}$ responds as $\mathcal{F}_{\text{SIG}}$ would. The communications between $\mathcal{A}$ and the various copies of $\mathcal{F}_{\text{SIG}}$ is also simulated in the obvious way. (It is stressed that $\mathcal{S}$ may need to simulate for $\mathcal{A}$ several different copies of $\mathcal{F}_{\text{SIG}}$.)

Analysis of $\mathcal{S}$. Demonstrating the validity of $\mathcal{S}$, we show that for any environment $\mathcal{Z}$:

$$\text{EXEC}^{\mathcal{F}_{\text{SIG}}}_{\Sigma_0,\mathcal{A},\mathcal{Z}} \approx \text{IDEAL}_{\mathcal{F}_{\text{POST-KE}},\mathcal{S},\mathcal{Z}} \tag{1}$$

This is done as follows. First we define an event EC (for "early corrupt") and demonstrate that, given event EC, the views of $\mathcal{Z}$ in the two interactions are identically distributed. (Essentially EC is the event where one of the parties is corrupted before any of the parties outputs the session key.) We then concentrate on demonstrating (1) under the condition that event EC does not occur. This is done by defining two hybrid distributions, $\mathcal{H}_1$ and $\mathcal{H}_2$, and demonstrating that, given that event EC does not occur, we have $\text{EXEC}^{\mathcal{F}_{\text{SIG}}}_{\Sigma_0,\mathcal{A},\mathcal{Z}} \approx \mathcal{H}_1 \approx \mathcal{H}_2 \approx \text{IDEAL}_{\mathcal{F}_{\text{POST-KE}},\mathcal{S},\mathcal{Z}}$. The leftmost similarity is demonstrated base on the Decisional Diffie-Hellman assumption. The second similarity is demonstrated based on the security of the PRF function family in use. The rightmost similarity is demonstrated base on the security of the MAC function family in use.

The event EC. Consider an interaction of $\mathcal{Z}$ with $\mathcal{A}$ and parties running Σ_0 in the $\mathcal{F}_{\text{SIG}}$-hybrid model. Here event EC is the event where $\mathcal{A}$ corrupts a party before any of the parties generated an output value. In an interaction of $\mathcal{Z}$ with $\mathcal{S}$ in the ideal process for $\mathcal{F}_{\text{POST-KE}}$, event EC is the event where the simulated $\mathcal{A}$ within $\mathcal{S}$ corrupts a party before $\mathcal{S}$ has sent the first $(\texttt{Output}\ldots)$ message to $\mathcal{F}_{\text{POST-KE}}$. We have:

Claim 19 *Conditioned on event EC, the distributions $\text{EXEC}^{\mathcal{F}_{\text{SIG}}}_{\Sigma_0,\mathcal{A},\mathcal{Z}}$ and $\text{IDEAL}_{\mathcal{F}_{\text{POST-KE}},\mathcal{S},\mathcal{Z}}$ are identical.*

Proof: The claim follows by inspecting the codes of Σ_0 and $\mathcal{S}$. Specifically, consider the joint view of $\mathcal{Z}$ and the simulated $\mathcal{A}$ within $\mathcal{S}$ in the ideal process. Since $\mathcal{S}$ perfectly mimics protocol Σ_0 for the simulated $\mathcal{A}$, we have that this joint view is distributed identically to the joint view of $\mathcal{Z}$ and $\mathcal{A}$ in a real interaction with Σ_0, with the only possible exception that the outputs generated by $\mathcal{F}_{\text{POST-KE}}$ may be inconsistent with the rest of the interaction. However, if event EC occurs then, in the ideal process, $\mathcal{S}$ corrupts one of the parties before the `(Output...)` message is sent to $\mathcal{F}_{\text{POST-KE}}$. In this case $\mathcal{F}_{\text{POST-KE}}$ sends to both parties the output values generated by $\mathcal{S}$ and the simulation becomes perfect (i.e., the joint view of $\mathcal{Z}$ and the simulated $\mathcal{A}$ within $\mathcal{S}$ is distributed identically to the joint view of $\mathcal{Z}$ and $\mathcal{A}$ in a real interaction with Σ_0 in the $\mathcal{F}_{\text{SIG}}$-hybrid model). □

In th rest of the proof we assume that event EC occurs with probability that is non-negligible bounded away from 1. (Otherwise we conclude that the two sides of 1 are statistically indistinguishable.)

The hybrid distributions. Let $\text{EXEC}|\overline{\text{EC}}$ denote the distribution of $\text{EXEC}^{\mathcal{F}_{\text{SIG}}}_{\Sigma_0,\mathcal{A},\mathcal{Z}}$ conditioned on the event that EC does not occur. Similarly, let $\text{IDEAL}|\overline{\text{EC}}$ denote the distribution of $\text{IDEAL}_{\mathcal{F}_{\text{POST-KE}},\mathcal{S},\mathcal{Z}}$ conditioned on the event that EC does not occur. We define two hybrid distributions $\mathcal{H}_1$ and $\mathcal{H}_2$, and demonstrate that $\text{EXEC}|\overline{\text{EC}} \approx \mathcal{H}_1 \approx \mathcal{H}_2 \approx \text{IDEAL}|\overline{\text{EC}}$. The hybrid distributions are defined as follows:

- $\mathcal{H}_1$ takes the distribution of the output of $\mathcal{Z}$ from a hypothetical interaction which is identical to a real interaction with $\mathcal{A}$ and parties running Σ_0 in the $\mathcal{F}_{\text{SIG}}$-hybrid model (conditioned on the event that EC does not occur), with the following exception. Whenever Σ_0 instructs the initiator (resp., the responder) to evaluate the pseudorandom function PRF with the key β^x (resp., α^y), the parties will now evaluate PRF with an independently chosen random key $r \xleftarrow{R} Z_p$. (Both parties use the same value of r.) That is, we now have that the MAC key is $k_1 = \text{PRF}_r(1)$ and the output key is $\kappa = \text{PRF}_r(0)$.

- $\mathcal{H}_2$ is identical to distribution $\mathcal{H}_1$, with the exception that the parties choose k_1 and κ to be independent and random values in the range of PRF. (Both parties have the same value for k_1; similarly both parties have the same value for κ.)

Abusing notation, we use $\mathcal{H}_1$ and $\mathcal{H}_2$ also to denote the interactions of $\mathcal{Z}$ that lead the the corresponding output distributions. We show:

Claim 20 *Assume that the Decisional Diffie-Hellman assumption holds. Then* $\text{EXEC}|\overline{\text{EC}} \approx \mathcal{H}_1$.

Proof: Assume that there exists an environment $\mathcal{Z}$ and an adversary $\mathcal{A}$ such that $\mathcal{Z}$ distinguishes with non-negligible probability between the interactions $\text{EXEC}|\overline{\text{EC}}$ and $\mathcal{H}_1$. We construct an adversary D that violates the Decisional Diffie-Hellman assumption. That is, D is given g^a, g^b, g^z where $a, b \xleftarrow{R} Z_q$, and can distinguish between the case where $z = ab$ and the case where $z \xleftarrow{R} Z_q$.

Given g^a, g^b, g^z, adversary D runs a copy of $\mathcal{Z}$ on a simulated interaction with $\mathcal{A}$ and parties running Σ_0 in the $\mathcal{F}_{\text{SIG}}$-hybrid model. (D plays for $\mathcal{Z}$ the roles of $\mathcal{A}$ and the parties.) This is done with the following exceptions. First, when the initiator sends the Start message, D sets the g^x value to equal g^a, the first element from its input. Similarly, when the responder sends the Response message, D set the g^y value to equal g^b, the second element from its input. Next, that whenever the responder (resp., the initiator) instructed to run $\text{PRF}_{\alpha^y}()$ (resp., $\text{PRF}_{\beta^x}()$), then D replaces this evaluation with an evaluation of $\text{PRF}_{g^z}()$. Finally, if $\mathcal{Z}$ corrupts a party before the initiator generates output then D aborts and outputs a random bit. Otherwise, D outputs whatever $\mathcal{Z}$ outputs.

Consider first the case where $z \stackrel{R}{\leftarrow} Z_q$, independently from a and b. In this case, the view of the simulated $\mathcal{Z}$, conditioned on the event that D did not abort, is distributed identically to the view of $\mathcal{Z}$ in interaction $\mathcal{H}_1$. This is so since in both interactions the values g^x, g^y, and the key to PRF are independently and randomly chosen. Furthermore, in neither interaction $\mathcal{Z}$ sees the secret exponents x or y.

Next, consider the case where $z = ab$. We claim that in this case the view of simulated $\mathcal{Z}$, conditioned on the event that D did not abort, is distributed identically to the view of $\mathcal{Z}$ in interaction EXEC$|\overline{\text{EC}}$. Indeed, the only potential mismatch between the two interactions is if, in EXEC$|\overline{\text{EC}}$, the initiator accepts a Response message with the real peer identity R and a value β that is different than the value g^y chosen by the responder, or alternatively the responder accepts a Finish message where the value α is different than the value g^x chosen by the initiator. However, such a mismatch cannot occur due to the properties and use of $\mathcal{F}_{\text{SIG}}$. Specifically, the initiator accepts the Response message with peer identity R only if the signature σ_r verifies as a signature on g^x, β with session identifier "1" $\circ\, s$ and signer R, and the only way for this to occur is if the responder R registered as the signer with session identifier "1" $\circ\, s$ and then asked $\mathcal{F}_{\text{SIG}}$ to sign (g^x, β), or in other words $\beta = g^y$. Similarly, the responder accepts the Response message with peer identity I only if the signature σ_i verifies as a signature on g^y, α with session identifier "0" $\circ\, s$ and signer I, and the only way for this to occur is if the initiator I registered as the signer with session identifier "0" $\circ\, s$ and then asked $\mathcal{F}_{\text{SIG}}$ to sign (g^y, α), or in other words $\alpha = g^x$.

We conclude that D distinguishes with non-negligible probability between the case where $z \stackrel{R}{\leftarrow} Z_q$ and the case where $z = ab$. □

Claim 21 *Assume that* PRF *is a secure pseudorandom function family. Then* $\mathcal{H}_1 \approx \mathcal{H}_2$.

Proof: Assume that there exists an environment $\mathcal{Z}$ and an adversary $\mathcal{A}$ such that $\mathcal{Z}$ distinguishes with non-negligible probability between the interaction $\mathcal{H}_1$ and the interaction $\mathcal{H}_2$. We construct an adversary D that breaks the security of the function family PRF. That is, D has access to an oracle function f, and distinguishes with non-negligible probability between the case where $f() = \text{PRF}_r()$ where r is a random value, and the case where f is a random function with the appropriate domain and range.

Adversary D runs a copy of $\mathcal{Z}$ on a simulated interaction with $\mathcal{A}$ and parties running Σ_0 in the $\mathcal{F}_{\text{SIG}}$-hybrid model. (D plays the roles of $\mathcal{A}$ and the parties for $\mathcal{Z}$.) This is done with the exception that whenever a party is instructed to compute k_1 (either in generating the Response message or in generating the Finish message), D instead lets $k_1 = f(1)$. Similarly, whenever a party is instructed to generate the session key κ, D sets the value of the session key to $\kappa = f(0)$. If $\mathcal{Z}$ corrupts a party before the initiator generates output then D aborts and outputs a random bit. Otherwise, D outputs whatever $\mathcal{Z}$ outputs.

It is easy to see that if f is a random function then the view of the simulated $\mathcal{Z}$ (given that D does not abort) is distributed identically to its view in interaction $\mathcal{H}_2$. Similarly, if $f() = \text{PRF}_r()$ and $\mathcal{Z}$ does not abort, then the view of the simulated $\mathcal{Z}$ is distributed identically to its view in interaction $\mathcal{H}_1$. □

Claim 22 *Assume that* MAC *is a secure message authentication function family. Then* $\mathcal{H}_2 \approx$ IDEAL$|\overline{\text{EC}}$.

Proof: The interactions $\mathcal{H}_2$ and IDEAL$|\overline{\text{EC}}$ are identical except for the following two points. First, in IDEAL$|\overline{\text{EC}}$ the peer identity in the output of the initiator is the identity R that appears in the input of the responder; In contrast, in $\mathcal{H}_2$ that peer identity is the identity P that appears in the Response message received by the initiator (and this identity may potentially be different than R). Second, if the responder is uncorrupted and generates output, then in $\text{IDEAL}|\overline{\text{EC}}$ the peer identity in the output of the responder is the identity I that appears in the input of the initiator; In contrast, in $\mathcal{H}_2$ that peer identity is the identity P that appears in the Finish message received by the responder.

We prove the claim by demonstrating that in interaction $\mathcal{H}_2$ the probability that a party outputs a peer identity that is different than the identity in the input of the other party is negligible. This is done via reduction to the security of MAC as a message authentication function against chosen message attacks. Specifically, we construct an adversary D that, given oracle access to a function $\text{MAC}_r()$ where r is a randomly chosen key, generates a message m^* and a tag $t*$ such that $t^* = \text{MAC}_r(m^*)$ and D has not queried the oracle on m^*.

Adversary D runs a copy of $\mathcal{Z}$ on a simulated interaction with $\mathcal{A}$ and parties running Σ_0 in the $\mathcal{F}_{\text{SIG}}$-hybrid model. (D plays the roles of $\mathcal{A}$ and the parties for $\mathcal{Z}$.) Whenever a party is instructed to compute (or verify) $t = \text{MAC}_{k_1}(m)$ for some m, D sets t to the response of its oracle on query m. Finally, if the initiator outputs a peer identity $P \neq R$ then D outputs $m^* = (\text{“1”}, s, P)$ together with the tag t^* in the Response message received by the initiator. Similarly, if the responder outputs a peer identity $P \neq I$ then D outputs $m^* = (\text{“0”}, s, P)$ together with the tag t^* in the Finish message received by the responder. (If $\mathcal{Z}$ corrupts a party before the initiator generates output then D aborts with no output.)

Analyzing D, notice that m^* was never generated by either party, thus it was never before queried by D. (Here we use the fact that the text in the MAC application by the initiator is different than the text in the MAC application by the responder. This is guaranteed by the “0”/“1” field.) Furthermore, $t^* = \text{MAC}_r(m^*)$, otherwise the party would not have generated output. Finally, $\mathcal{Z}$'s view when run by D is identical to its view in interaction $\mathcal{H}_2$. Thus $\mathcal{Z}$ outputs a successful forgery, before corrupting any party, with non-negligible probability. □

This completes the analysis of $\mathcal{S}$ and the proof of the theorem. □

A.3 Obtaining UC Secure Channels

We argue that standard protocols for realizing secure channels in the $\mathcal{F}_{\text{KE}}$-hybrid model exchange remain secure when the key exchange protocol in use securely realizes $\mathcal{F}_{\text{POST-KE}}$. This is done as follows. Recall that in [5] a secure-channels ideal functionality, $\mathcal{F}_{\text{SC}}$, is formulated; next, a protocol is shown that securely realizes $\mathcal{F}_{\text{SC}}$ in the $\mathcal{F}_{\text{KE}}$-hybrid model, where $\mathcal{F}_{\text{KE}}$ is the ideal key exchange functionality in the pre-specified peer setting. (This protocol is the standard protocol that first uses the key exchange functionality to generate a session key, and then encrypts and authenticates each message.)

We claim that a simple modification of this secure channels protocol results in a protocol that securely realizes $\mathcal{F}_{\text{SC}}$ in the $\mathcal{F}_{\text{POST-KE}}$-hybrid model. The modification is as follows: instead of invoking $\mathcal{F}_{\text{KE}}$ with the pre-specified identity of the peer, invoke $\mathcal{F}_{\text{POST-KE}}$ with no peer identity specified; next, when $\mathcal{F}_{\text{POST-KE}}$ returns the actual peer identity, verify that this identity agrees with the desired peer identity, and abort the session if there is a mis-match. Inspecting the proof of [5], it is easy to see that this modified protocol securely realizes $\mathcal{F}_{\text{SC}}$ in the $\mathcal{F}_{\text{POST-KE}}$-hybrid model. We omit further details.

A.4 Securely Realizing $\mathcal{F}_{\text{POST-KE}}$ Implies SK-security

We argue that the UC notion of key exchange in the post-specified peer setting implies the corresponding SK security notion (Definition 3). More precisely, let π be a protocol that securely realizes $\mathcal{F}_{\text{POST-KE}}$, and let $\hat{\pi}$ denote the multi-session extension of π. (See [5, Sec. 3] for a definition of the multi-session extension of a protocol.) We claim that protocol $\hat{\pi}$ is SK-secure in the post-specified peer setting. This is shown following the lines of the treatment in [5] for the pre-specified peer case. First, we formulate a variant of the test environment, $\mathcal{Z}_{\text{TEST}}$, that captures the post-specified peer variant of SK-security. (The only difference is in the way $\mathcal{Z}_{\text{TEST}}$ defines matching sessions.) Next we use essentially the same argument as in [5] to show that no adversary that interacts with parties running $\hat{\pi}$ and (the reformulated) $\mathcal{Z}_{\text{TEST}}$ can skew the output of $\mathcal{Z}_{\text{TEST}}$ more than negligibly away from uniform over $\{0, 1\}$. Also here, we omit further details.