SIGMA: the 'SIGn-and-MAc' Approach to Authenticated Diffie-Hellman and its Use in the IKE Protocols *

Hugo Krawczyk†

June 12, 2003

Abstract

We present the SIGMA family of key-exchange protocols and the "SIGn-and-MAc" approach to authenticated Diffie-Hellman underlying its design. The SIGMA protocols provide perfect forward secrecy via a Diffie-Hellman exchange authenticated with digital signatures, and are specifically designed to ensure sound cryptographic key exchange while supporting a variety of features and trade-offs required in practical scenarios (such as optional identity protection and reduced number of protocol rounds). As a consequence, the SIGMA protocols are very well suited for use in actual applications and for standardized key exchange. In particular, SIGMA serves as the cryptographic basis for the signature-based modes of the standardized Internet Key Exchange (IKE) protocol (versions 1 and 2).

This paper describes the design rationale behind the SIGMA approach and protocols, and points out to many subtleties surrounding the design of secure key-exchange protocols in general, and identity-protecting protocols in particular. We motivate the design of SIGMA by comparing it to other protocols, most notable the STS protocol and its variants. In particular, it is shown how SIGMA solves some of the security shortcomings found in previous protocols.

*A shortened version of this paper appears in the proceedings of CRYPTO'03. For further information related to the SIGMA protocols see `http://www.ee.technion.ac.il/~hugo/sigma.html`

†EE Department, Technion, Haifa, Israel, and IBM T.J. Watson Research Center. Email: hugo@ee.technion.ac.il

Contents

1 Introduction

In this paper we describe the SIGMA family of key-exchange protocols, with emphasis on its design features and rationale. The SIGMA protocols introduce a general approach to building authenticated Diffie-Hellman protocols using a *careful* combination of digital signatures and a MAC (message authentication) function. We call this the *"SIGn-and-MAc" approach* which is also the reason for the SIGMA acronym.

SIGMA serves as the cryptographic basis for the Internet Key Exchange (IKE) protocol [14, 19] standardized to provide key-exchange functionality to the IPsec suite of security protocols [20]. More precisely, SIGMA is the basis for the signature-based authenticated key exchange in IKE [14], which is the most commonly used mode of public-key authentication in IKE, and the basis for the only mode of public-key authentication in IKEv2 [19].

This paper provides the first systematic description of the development and rationale of the SIGMA protocols. The presentation is intended to motivate the design choices in the protocol by comparing and contrasting it to alternative protocols, and by learning from the strong and weak aspects of previous protocols. It also explains how the different variants of the SIGMA protocol follow from a common design core. In particular, it explains the security basis on which the signature-based modes of IKE, and its current revision IKEv2, are based. The presentation is informal and emphasizes rationale and intuition rather than rigorous analysis. A formal analysis of the SIGMA protocol has been presented in [8] where it is shown that the basic SIGMA design and its variants are secure under a complexity-theoretic model of security. While this rigorous analysis is essential for gaining confidence in the security design of SIGMA, it does not provide an explicit understanding of the design process that led to these protocols, and the numerous subtleties surrounding this design. Providing such an understanding is a main goal of this paper which will hopefully be beneficial to cryptographers and security protocol designers (as well as for those engineering security solutions based on these protocols).

The basic guiding requirements behind the design of SIGMA are (a) to provide a secure key-exchange protocol based on the Diffie-Hellman exchange (for ensuring "perfect forward secrecy"), (b) the use of digital signatures as the means for public-key authentication of the protocol, and (c) to provide the option to protect the identities of the protocol peers from being learned by an attacker in the network. These were three basic requirements put forth by the IPsec working group for its preferred key-exchange protocol. The natural candidate for satisfying these requirements is the well-known STS key-exchange protocol due to Diffie, van Oorschot and Wiener [11]. We show, however, that this protocol and some of its variants (including a variant adopted into Photuris [17], a predecessor of IKE as the key-exchange protocol for IPsec) suffer from security shortcomings that make them unsuited for some practical scenarios, in particular in the wide Internet setting for which the IPsec protocols are designed. Still, the design of SIGMA is strongly based on that of STS: both the strengths of the STS design principles (very well articulated in [11]) as well as the weaknesses of some of the STS protocol choices have strongly influenced the SIGMA design.

One point that is particularly important for understanding the design of SIGMA (and other key-exchange protocols) is the central role that the requirement for identity protection has in this design. As it turns out, the identity protection functionality conflicts with the essential requirement of peer authentication. The result is that both requirements (authentication and identity protection) can be satisfied simultaneously, at least to some extent, but their co-existence introduces significant subtleties both in the design of the protocol and its analysis. In order to highlight this issue we compare SIGMA to another authenticated Diffie-Hellman design, a variant of the ISO protocol [15], that has been shown to be secure [7] but which is not well-suited to support identity protection (Section 4). As we will see SIGMA provides a satisfactory and flexible solution to this problem by supporting identity protection as an optional feature of the protocols, while keeping the number of communication rounds and cryptographic operations to a minimum. As a result SIGMA can suit the identity protection scenarios as well as those that do not require this functionality. We thus believe that SIGMA is well suited as a "general purpose" authenticated Diffie-Hellman protocol that can serve a wide range of applications and security scenarios.

History of the SIGMA protocols. The SIGMA approach was introduced by the author in 1995 [22] to the IPsec working group as a possible replacement for the Photuris key-exchange protocol [17] developed at the time by that working group. Photuris used a variant of the STS protocol that we showed [22] to be flawed through the attack presented in Section 3.3. In particular, this demonstrated that the Photuris key exchange, when used with optional identity protection and RSA signatures (or any signature scheme allowing for message recovery), was open to the same form of attack that originally motivated the design of STS (see Section 3.1). Eventually, the Photuris protocol was replaced with the Internet Key Exchange (IKE) protocol which adopted SIGMA (unnamed at the time) into its two signature-based authentication modes: main mode (that provides identity protection) and aggressive mode (which does not support identity protection). The IKE protocol was standardized in 1999, and a revised version (IKEv2) is currently under way [19] (the latter also uses the SIGMA protocol as its cryptographic key exchange).

Related work. There is a vast amount of work that deals with the design and analysis of key-exchange (and authentication) protocols and which is relevant to the subject of this paper. Chapter 12 of [29] provides many pointers to such works, and additional papers can be found in the more recent security and cryptography literature. There have been a few works that provided analysis and critique of the IKE protocol (e.g., [12, 31]). Yet, these works mainly discuss issues related to functionality and complexity trade-offs rather than analyzing the core cryptographic design of the key exchange protocols. A formal analysis of the IKE protocols has been carried by Meadows [28] using automated analysis tools. In addition, as we have already mentioned, [8] provides a formal analysis of the SIGMA protocols (and its IKE variants) based on the complexity-theoretic approach to the analysis of key-exchange protocols initiated in [2]. A BAN-logic analysis of the STS protocols is presented in [34], and attacks on these protocols that enhance those reported in [22] are presented in [5] (we elaborate on these attacks in Section 3.3). Finally, we mention the SKEME protocols [23] which served as the basis for the cryptographic structure of IKE and its non-signature modes of authentication, but did not include a signature-based solution as in SIGMA.

A final remark. One clear conclusion from this work is that in order to achieve provable security for key-exchange protocols one does not have to abandon simplicity and practicality (as exemplified by the ISO and SIGMA protocols from Sections 4 and 5, respectively). Yet, converging to these protocols and proving them secure has been non-trivial (especially considering the large number of flaws constantly found in authentication and key-exchange protocols). The reason for this is the enormous number of subtleties surrounding the definition and design of secure key-exchange protocols. Telling apart secure from insecure protocol can hardly be done by immediate inspection, or using simple intuition, as illustrated in Figure 1. Therefore, understanding the rationale for design choices in secure protocols is of utmost importance as well as it is understanding the shortcomings of other protocols. Hopefully, this work will help to shed some light on these issues.

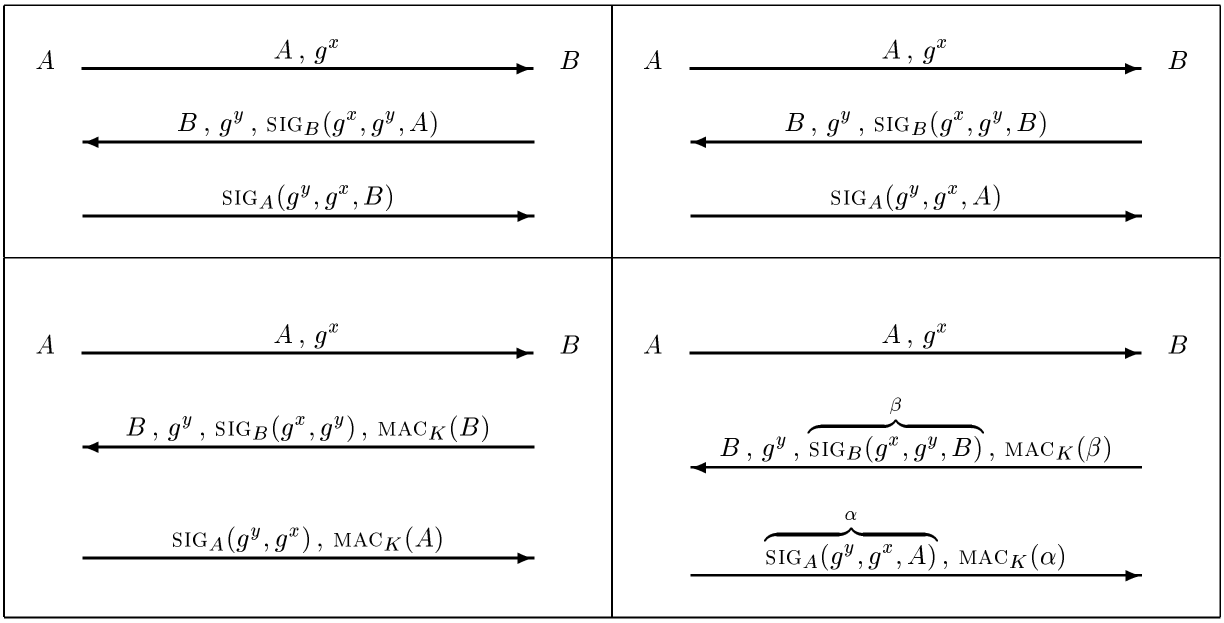

Figure 1: Test your intuition: which of these four authenticated Diffie-Hellman exchanges constitute secure key-exchange protocols (without identity protection)? Answers are provided within the paper. The notation SIG_X represents a signature by participant X; MAC_K represents a message authentication function computed using a key K derived from the Diffie-Hellman key g^{xy}. The output session key in all cases is derived from g^{xy} independently of K.

Organization In Section 2 we informally discuss security requirements for key-exchange protocols in general and for SIGMA in particular, and present specific requirements related to identity protection. Section 3 presents the STS protocol and its variants, and analyzes the strengths and weaknesses of these protocols. Section 4 discusses the ISO protocol as a further motivation for the design of SIGMA (in particular, this discussion serves to stress the role of identity protection in the design of SIGMA). Finally, Section 5 presents the SIGMA protocols together with their design rationale and security properties. In particular, Section 5.4 discusses the SIGMA variants used in the IKE protocols. Additional material is presented in the appendices. In Appendix A we expand on the security definition that underlies the analysis of the SIGMA protocols in [8]. In particular, this appendix includes a simplified (and somewhat informal) definition of key-exchange security. Appendix B presents a "full fledge" instantiation of SIGMA which includes some of the elements omitted in the simplified presentation of Section 5 but which are crucial for a full secure implementation of the protocols. Appendix C discusses key derivation issues and presents the specific key derivation technique designed for, and used in, the IKE protocols. This technique is of independent interest since it is applicable to the derivation of keys in other key-exchange protocols; in particular, it includes a mechanism for "extracting randomness" from Diffie-Hellman keys using pseudorandom functions.

2 Preliminaries: On the Security of Key-Exchange Protocols

Note: this section is important for understanding the design goals of SIGMA; yet, the impatient reader may skip it in a first reading (but see the notation paragraph at the end of the section).

In this paper we present an informal exposition of the design rationale behind the development of the SIGMA protocols. This exposition is intended to serve crypto protocol designers and security engineers to better understand the subtle design and analytical issues arising in the context of key-exchange (KE for short) protocols in general, and in the design of SIGMA in particular. This exposition, however, is not a replacement for a formal analysis of the protocol. A serious analysis work requires a formal mathematical treatment of the underlying security model and protocol goals. This essential piece of work for providing confidence in the security of the SIGMA protocols is presented in a companion paper [8]. The interested reader should consult that work for the formal foundations of security on which SIGMA is based (see also Appendix A). Yet, before going on to present the SIGMA protocols and some of its precursors we discuss informally some of the salient aspects of the analytical setting under which we study and judge KE protocols. This presentation will also provide a basis for the discussion of some of the techniques, strengths and weaknesses showing up in the protocols studied in later sections.

We start by noting that *there is no ultimate security model.* Security definitions may differ depending on the underlying mathematical methodology, the intended application setting, the consideration of different properties as more or less important, etc. The discussion below focuses on the core security properties of KE protocols as required in most common settings. These requirements stem from the the quintessential application of KE protocols as suppliers of shared keys to pairs of parties which later use these keys to secure (via integrity and secrecy protection) their pairwise communications. In addition, we deal with some more specific design goals of SIGMA motivated by requirements put forth by the IPsec working group: the use of the Diffie-Hellman exchange as the basic technique for providing "perfect forward secrecy", the use of digital signatures for authenticating the exchange, and the (possibly optional) provision of "identity protection".

2.1 Overview of the security model and requirements

In spite of being a central (and "obvious") functionality in many cryptographic and security applications, the notion of a "secure key-exchange protocol" remains a very complex notion to formalize correctly. Here we state very informally some basic requirements from KE protocols that we will use as a basis for later discussion of security issues arising in the design of KE protocols. These requirements are in no way a replacement for a formal treatment carried in [8], but are consistent (at least at the intuitive level) with the notion of security in that work. (See also Appendix A.)

Authentication Each party to a KE execution (referred to as a *session*) needs to be able to uniquely *verify the identity* of the peer with which the session key is exchanged.

Consistency If two honest parties establish a common session key then both need to have a consistent view of who the peers to the session are. Namely, if a party A establishes a key K and believes the peer to the exchange to be B, then if B establishes the session key K then it needs to believe that the peer to the exchange is A; and vice-versa.

Secrecy If a session is established between two honest peers then no third party should be able to learn any information about the resultant session key (in particular, no such third party, watching or interfering with the protocol run, should be able to distinguish the session key from a random key).

While the "authentication" and "secrecy" requirements are very natural and broadly accepted, the requirement of "consistency" is much trickier and many times overlooked. In Section 3.1 we exemplify this type of failure through an attack first discovered in [11]. This attack, to which we refer as an "identity misbinding attack", applies to many seemingly natural and intuitive protocols. Avoiding this form of attack and guaranteeing a consistent binding between a session key and the peers to the session is a central element in the design of SIGMA.

The adversarial model. An important point to observe is that the above requirements are not absolute but exist only in relation to a well-defined attack model. Here we summarize the adversarial model from [8]. We consider an active ("man-in-the-middle") attacker with full control of the communication links between parties. This attacker can intercept messages, delay or prevent their delivery, modify them at will, inject its own messages, interleave messages from different sessions, etc. This adversary can also schedule the activation of parties to initiate and respond to KE sessions.

Parties hold long-term private information that they use to authenticate their identities to other parties. (In the context of this paper we can concretely think of this long-term authentication material as secret digital signature keys.) We also assume the existence of a trusted certification authority, or any other trusted mechanism (manual distribution, web of trust, etc), for *faithfully* binding identities with public keys (i.e., it is assumed that the trusted party correctly verifies the identity of the registrant of a public key before issuing a certificate that binds this identity with this public key) . Each party has its own computing environment which may or may not be controlled by the attacker. If the attacker gains access to the secret long-term authentication information at a party then we consider this party fully controlled by the attacker, and we call that party corrupted. We make no attempt at protecting session keys produced by a party after corruption (since in this case the attacker can fully impersonate that party), but we will be interested in protecting session keys produced (and erased from memory) before the party corruption happened. This protection of past session keys in spite of the compromise of long-term secrets is known as perfect forward secrecy (PFS) and is a property of all the (secure) protocols discussed in this paper.

We also consider the level of protection that a KE protocol can provide when the attacker gains some session-specific information such as learning information on the secret internal state of a session (e.g. the exponent x used by a party to produce an ephemeral DH exponential g^x) or learning the value of a (past or current) session key. Note that in the case that such information leakage happens (either via break-ins, mishandling of secret ephemeral information, cryptanalysis, etc) then no guarantee on the security of the exposed session can be made. Yet, in this case we require that any adverse security consequence from such a compromise will affect the exposed sessions only, with no implications on the security of other sessions. These security requirements are very significant and take care of avoiding well known type of attacks such as known-key attacks and replay attacks (see [29]), and emphasize the need for key independence between different sessions.

Note: The above attack model differentiates between an attack that compromises a long-term secret and one that exposes an ephemeral state or secret. While in some environments gaining access to a local state of a session is as hard, or easy, as gaining access to the party's long-term secrets, in other cases, however, the level of protection of these two forms of information may be very different. For example, the secret signature key may be well protected in a special hardware device while ephemeral DH pairs (x, g^x) may be produced off-line and stored overnight in less protected environments. Our model thus follows the important security principle that the exposure of ephemeral security information will have more limited consequences than the compromise of sensitive long-term secrets.

Security analysis. The analysis of protocols under this model is carried on the basis of the generic properties assumed from the cryptographic primitives used in the protocol (e.g., digital signatures, MAC, etc.), rather than based on the properties of specific algorithms (i.e. specific instantiations of these primitives). This algorithm independence (or generic security) principle is important in case that specific crypto algorithms need to be replaced (for better security or improved performance), and is needed to support different combinations of individually secure algorithms. We say that we *prove the security* of a key-exchange protocol in the above model, if we can show how to transform any adversarial action that violates any of the postulated security properties of the protocol into an explicit algorithm that breaks one of the cryptographic primitives used in the protocol. This ensures that as long as these primitives (and their implementation) are not broken then the protocol satisfies the defined properties.

2.1.1 Discussion: sufficiency of the above security requirements.

One important question is whether the above security requirements (and more precisely the formal security requirements from [8]), under which we judge the security of protocols in this work, are necessary and/or sufficient to guarantee "key-exchange security". Necessity is easy to show through natural examples in which the removal of any one of the above required properties results in explicit and clearly harmful attacks against the security of the exchanged key (either by compromising the secrecy of the key or by producing an inconsistent binding between the key and the identities of the holders of that key). Sufficiency, however, is harder to argue. We subscribe to the approach put forth in [7] (and followed by [8]) by which a minimal set of requirements for a KE protocol must ensure the security of the quintessential application of KE protocols, namely, the provision of "secure channels" (i.e., the sharing of a key between peers that subsequently use this key for protecting the secrecy and integrity of the information transmitted between them). It is shown in [8] that their definition (outlined here) is indeed sufficient (and actually minimalistic) for providing secure channels.

Also important to stress is that this definitional approach dispenses of some requirements that some authors (e.g., [27]) consider vital for a sound definition of security. One important example is the aliveness requirement, namely, if A completes a session with peer B then A has a proof that B was "alive" during the execution of the protocol (e.g., by obtaining B's unique authentication on some nonce freshly generated by A). This property is *not* guaranteed by our (or [8]) definition of security. Moreover, some natural key-transport protocols (e.g., the ENC protocol formally specified in [7]) are useful key-exchange protocols that guarantee secure channels yet do not provide a proof of aliveness. The only possible negative aspect of a KE protocol that lacks the aliveness guarantee is that a party may establish a session with a peer that did not establish the corresponding session (and possibly was not even operational at the time); this results in a form of "denial of service" for the former party but not a compromise of data transmitted and protected under the key. However, DoS attacks with similar effects are possible even if aliveness guarantees are provided, for example by the attacker preventing the arrival of the last protocol message to its destination.

A related (and stronger) property not guaranteed by our basic definition of security is peer awareness. Roughly speaking, a protocol provides peer awareness for A if when A completes a session with peer B, A has a guarantee that (not only is B alive but) B has initiated a corresponding session with peer A. Adding aliveness and peer awareness guarantees to a KE that lacks these properties is often very simple, yet it may come at a cost (e.g., it may add messages to the exchange or complicate other mechanisms such as identity protection). Therefore, it is best to leave these properties as optional rather than labeling as "insecure" any protocol that lacks them.[1]

All the protocols discussed in this paper provide aliveness proofs to both parties but only the ISO protocol and the 4-message SIGMA-I with added ACK (Section 5.2) provide peer awareness to both parties. In particular, the IKE protocols (Section 5.4) do not provide peer awareness to one of the peers. As said, this property can be added, when required, at the possible expense of extra messages or other costs.

2.2 Identity protection

As discussed in Section 2.1, key-exchange protocols require strong mutual authentication and therefore they must be designed to communicate the identity of each participant in the protocol to its session peer. This implies that the identities must be transmitted as part of the protocol. Yet some applications require to prevent the disclosure of these identities over the network. This may be the case in settings where the identity (for the purpose of authentication) of a party is not directly derivable from the routing address that must appear in the clear in the protocol messages.

A common example is the case of mobile devices wishing to prevent an attacker from correlating their (changing) location with the logical identity of the device (or user). Note that such an application may not just need to hide these identities from passive observers in the network but may require to conceal the identity even from active attackers. In this case the sole encryption of the sender's identity is not sufficient and it is required that the peer to the session proves its own identity before the encrypted identity is transmitted. Many other examples of applications requiring identity protection exist. One is the case of the IKE protocol in which lack of protection of the responder's identity in the key-exchange would open this identity to trivial "identity-probing attacks" from any machine in the Internet. That is, if I want to know the (logical) identity of a machine sitting at a given IP address all I need to do is to initiate an IKE exchange with that IP address and receive back, as part of the key exchange, the responder's identity (which may, for example, be included under a public-key certificate sent by the responder). To avoid this form of attack, the IKEv2 protocol [19] specifies that a responder to a key-exchange will not reveal its identity until the initiator of the exchange communicates and authenticates its own identity. In this way, the responder may use its own local security policy to determine if it is willing to engage in a key-exchange with that (authenticated) initiator and, in particular, if it is willing to reveal its identity to that initiator. Yet another example from a different set of applications is described in [32]; in this case, a smart-card engages in a key-exchange with a card-reader but the card will not reveal its identity until the reader has proven its identity (which, in particular, serves to prove that the latter is a legitimate card-reader).

As it turns out the requirement to support identity protection adds new subtleties to the design of KE protocols; these subtleties arise from the conflicting nature of identity protection and authentication. In particular, *it is not possible* to design a protocol that will protect both peer identities from active attacks. This is easy to see by noting that the first peer to authenticate itself (i.e. to prove its identity to the other party) must disclose its identity to the other party before it can verify the identity of the latter. Therefore the identity of the first-authenticating peer cannot be protected against an active attacker. In other words, KE protocols may protect both identities

[1] We stress that in contrast to the key-exchange setting, the aliveness requirements, and sometimes peer awareness, is essential in "entity authentication" protocols whose sole purpose may be to determine the aliveness of a peer.

from passive attacks and may, at best, protect the identity of one of the peers from disclosure against an active attacker.

This best-possible level of identity protection is indeed achievable by some KE protocols, and in particular is attained by the SIGMA protocols. The underlying design of SIGMA allows for a protocol variant where the initiator of the exchange is protected against active attacks and the responder's identity is protected against passive attacks (we refer to this variant as SIGMA-I), and it also allows for another variant where the responder's id is protected against active attacks and the initiator's against passive attacks only (SIGMA-R). Moreover, providing identity protection has been a main motivating force behind the design of SIGMA which resulted from the requirement put forth by the IPsec working group to support (at least optionally) identity protection in its KE protocol. The SIGMA protocols thus provide the best-possible protection against identity disclosure. The choice of SIGMA-I or SIGMA-R depends on which identity is considered as more sensitive and requires protection against active attacks. On the other hand, SIGMA offers full KE security also in cases where identity protection is not needed. That is, *the core security of SIGMA as a key-exchange protocol does not depend on the hiding of identities*; the latter is a privacy enhancement that the protocol adds, optionally, on top of the core protocol.

A related issue which is typical of settings where identity protection is a concern, but may also appear elsewhere, is that parties to the protocol may not know at the beginning of a session the specific identity of the peer but rather learn this identity as the protocol proceeds. (This is a natural case for the party acting as responder to a key-exchange request, but may also be the case for the initiator of the protocol which may intend to establish a session with one of a set of peers – all of which share the same physical address – rather than with one predefined peer). This adds, in principle, more attack avenues against the protocol and also introduces some delicate formal and design issues (e.g., most existing formalisms of key-exchange protocols do assume that the peer identities are fixed and known from the start of the session). In [8] this more general and realistic setting is formalized under the name of the post-specified peer setting and the SIGMA protocols are shown to be secure in this model. See [8] for the technical details.

Finally we comment on one additional privacy aspect of KE protocols. In some scenarios parties may wish to keep their privacy protected not only against attackers in the network but also to avoid leaving a "provable trace" of their communications in the hands (or disks) of the peers with which they communicate. A protocol such as ISO (see Section 4) in which each party to the protocol signs the peer's identity is particularly susceptible to this privacy concern (since these signatures can serve to prove to a third party the fact that the communication took place). In the SIGMA protocols, however, this proof of communication is avoided to a large extent by not signing the peer's identity, thus providing a better solution to this problem.

Note: some may consider the non-repudiation property of a protocol such as ISO (Section 4) as an advantage. However, we consider that non-repudiation using digital signatures does not belong to the KE protocol realm but as a functionality that needs to be dealt with carefully in specific applications, and with full awareness of the signer to the non-repudiation consequences.

2.3 Further remarks and notation

Denial of Service. Key-exchange protocols (including SIGMA) open opportunities for Denial-of-Service (DoS) attacks since the responder to an exchange is usually required to generate state and/or perform costly computations before it can authenticate the peer to the exchange. This type of attacks cannot be prevented in a strong sense but can be mitigated by using some fast-to-verify measures. One such technique has been proposed by Phil Karn [17] via the use of "cookies" that the responder to a KE protocol uses to verify that the initiator of the exchange is being able to receive messages directed to the IP address from which the exchange was initiated (thus preventing some form of trivial DoS attacks in which the attacker uses forged origin addresses, and also improves the chances to trace back the DoS attack). This and other techniques are orthogonal to the cryptographic details of the KE protocol and then can be adopted into SIGMA. In particular, version 2 of IKE [19] and the JFK protocol [1] incorporate Karn's technique into SIGMA. Other forms of denial of service are possible (and actually unavoidable) such as an active attacker that prevents the completion of sessions, or lets one party complete the session and the other not.

A word of caution. It is important to remark that all the protocols discussed in this paper are presented in their most basic form, showing only their cryptographic core. When used in practice it is essential to preserve this cryptographic core but also to take care of additional elements arising in actual settings. For example, if the protocol negotiates security parameters (such as crypto algorithms) or uses the protocol messages to send additional information then the designers of such full-fledge protocol need to carefully expand the coverage of authentication also to these additional elements. We also (over) simplify the protocol presentation by omitting the explicit use of "session identifiers": such identifiers are needed for the run of a protocol in a multi-session setting in order to match (or "multiplex") incoming protocol messages with open KE sessions. Moreover, the binding of messages to specific session id's is required for core security reasons such as preventing interleaving attacks. Similarly, nonces may need to be included in the protocol to ensure freshness of messages (e.g. to prevent replay attacks). In our presentation, however, these elements are omitted by over-charging the Diffie-Hellman exponentials with the additional functionality of session-id's and nonces. For the level of conceptual discussion in this paper, simplifying the presentation by reducing the number of elements in the protocol is useful (and also in line with the traditional presentation of protocols in the cryptographic literature, in particular with [11]). But when engineering a real-world protocol we recommend to clearly separate the functionality of different elements in the protocol. For illustration purposes, we present a version of a "full fledge" SIGMA protocol in Appendix B.

Notation. All the protocols presented here use the Diffie-Hellman exchange. We use the traditional exponential notation g^x where g is a group generator. However, all the treatment here applies to any group in which the Diffie-Hellman problem is hard. (A bit more precisely, groups in which the so called "Decisional Diffie-Hellman Assumption (DDH)" holds, namely, the infeasibility to distinguish between quadruples of the form (g, g^x, g^y, g^{xy}) and quadruples (g, g^x, g^y, g^z) where x, y, z are random exponents.) We use the acronym DH to denote Diffie-Hellman, and use the word "exponential" to refer to elements such as g^x, and the word "exponent" for x. In the description of our protocols the DH group and generator g are assumed to be fixed and known in advance to the parties or communicated at the onset of the protocol (in the later case, the DH parameters need to be included in the information authenticated by the protocol).
Throughout the paper we will also use the notation $\{\cdots\}_K$ to denote encryption of the information between the brackets under a symmetric encryption function using key K. Other cryptographic primitives used in the paper are a MAC (message authentication code) which is assumed to be unforgeable against chosen message attack by any adversary that is not provided the MAC key, and a digital signature scheme SIG assumed to be secure against chosen message attacks. By $\text{SIG}_A(msg)$ we denote the signature using A's private key on the message msg. The letters A and B denote the parties running a KE protocol, while Eve (or E) denotes the (active) attacker. We also use A, B, E to denote the identities used by these parties in the protocols.

3 The STS Protocols

Here we discuss the STS protocol (and some of its variants) which constitutes one of the most famous and influential protocols used to provide authenticated DH using digital signatures, and of particular appeal to scenarios where identity protection is a concern. The STS protocol, due to Diffie, van Oorschot and Wiener, is presented in [11] where a very instructive description of its design rationale is provided. In particular, this work is the first to observe some of the more intricate subtleties related to the authentication of protocols in general and of the DH exchange in particular. The STS protocol served as the starting point for the SIGMA protocols described in this paper. Both the strengths of the STS design principles as well as the weaknesses of some of the protocol choices have motivated the design of SIGMA. These aspects are important to be understood before presenting SIGMA. We analyze several variants of the protocol proposed in [11, 29, 17].

Remark. The attacks on the STS protocol and its variants presented here originate with the communications by the author to the IPsec working group in 1995 [22]. Since then some of these attacks were recalled elsewhere (e.g. [33]) and enhancements of the attack against the MAC variant have been provided in [5].

3.1 BADH and the identity-misbinding attack: A motivating example

As the motivation for the STS protocol (and later for SIGMA too) we present a proposal for an "authenticated DH protocol" which intuitively provides an authenticated KE solution but is actually flawed. We denote this protocol by BADH ("badly authenticated DH").

$$
\begin{array}{ccc}
A & \xrightarrow{\qquad\qquad\qquad g^x \qquad\qquad\qquad} & B \\
 & \xleftarrow{\qquad g^y\,,\, B\,,\, \text{SIG}_B(g^x, g^y) \qquad} & \\
 & \xrightarrow{\qquad\quad A\,,\, \text{SIG}_A(g^y, g^x) \quad\qquad} &
\end{array}
$$

The output of the protocol is a session key K_s derived from the DH value g^{xy}. (Note: the identity of A may also be sent in the first message, this is immaterial to the discussion here.)

This protocol provides the most natural way to authenticate a DH exchange using digital signatures. Each party sends its DH exponential signed under its private signature key. The inclusion of the peer's exponential under one's signature is required to prove freshness of the signature for avoiding replay attacks (we will discuss more about this aspect in Section 5, in particular the possibility to replace the signature on the peer's exponential with the signature on a peer-generated nonce). One of the important contributions of [11] was to demonstrate that this protocol, even if seemingly natural and intuitively correct, does not satisfy the important *consistency requirement* discussed in Section 2.1. Indeed, [11] present the following attack against the BADH protocol. An active ("person-in-the-middle") attacker, which we denote by Eve (or E), lets the first two messages of the protocol to go unchanged between A and B, and then it replaces the third message from A to B with the following message from Eve to B:

$$
E \xrightarrow{\qquad\quad E\,,\, \text{SIG}_E(g^y, g^x) \quad\qquad} B
$$

The result of the protocol is that A records the exchange of the session key K_s with B, while B records the exchange of the *same* key K_s with Eve. In this case, any subsequent application message arriving to B and authenticated under the key K_s will be interpreted by B as coming from Eve (since from the point of view of B the key K_s represents Eve not A). Note that this attack does not result in a breach of secrecy of the key (since the attacker does not learn, nor influence, the key in any way), but it does result in a severe breach of authenticity since the two parties to the exchange will use the same key with different understandings of who the peer to the exchange is, thus breaking the consistency requirement. To illustrate the possible adverse effects of this attack we use the following example from [11]: imagine B being a bank and A a customer sending to B a monetary element, such as an electronic check or digital cash, encrypted and authenticated under K_s. From the point of view of B this is interpreted as coming from Eve (which we assume to also be a customer of B) and thus the money is considered to belong to Eve rather than to A (hopefully for Eve the money will go to her account!).

The essence of the attack is that Eve succeeds in convincing the peers to the DH exchange (those that chose the DH exponentials) that the exchange ended successfully, yet the derived key is bound by each of the parties to a different peer. Thus the protocol fails to provide an authenticated binding between the key and the honest identities that generated the key. We will refer to this attack against the consistency requirement of KE protocols as an identity misbinding attack (or just "misbinding attack" for short).[2]

3.2 The basic STS protocol

Having discovered the misbinding attack on the "natural" authenticated DH protocol BADH, Diffie et al. [11] designed the STS protocol intended to solve this problem. The basic STS protocol is:

$$A \xrightarrow{\qquad\qquad g^x \qquad\qquad} B$$

$$\xleftarrow{\quad g^y\,,\ B\,,\ \{\,\mathrm{SIG}_B(g^x, g^y)\,\}_{K_s} \quad}$$

$$\xrightarrow{\quad A\,,\ \{\,\mathrm{SIG}_A(g^y, g^x)\,\}_{K_s} \quad}$$

where the notation $\{\cdots\}_K$ denotes encryption of the information between the brackets under a symmetric encryption function using key K. In the STS protocol the key used for encryption is the same as the one output as the session key produced by the exchange[3].

Is this protocol secure? In particular, is the introduction of the encryption of the signatures sufficient to thwart the identity misbinding attack? This at least has been the intention of STS. The idea was that by using encryption under the DH key the parties to the exchange "prove" knowledge of this key something which the attacker cannot do. Yet, no proof of security of the STS protocol exists (see more on this below). Even more significantly we show here that the misbinding attack applies to this protocol in any scenario where parties can register public keys

[2] This type of attack appears in the context of other authentication and KE protocols. It is sometimes referred to as the "unknown key share attack" [5, 18]. We believe that the name "identity-misbinding attack" better reflects the effect of the attack.

[3] This is a weakness of the protocol since the use of the session key in the protocol leaks information on the key (e.g., the key is not anymore indistinguishable from random). In addition, this can lead to the use of the same key with two different algorithms (one inside the KE protocol, and another when using the exchanged session key in the application that triggered the key exchange), thus violating the basic cryptographic principle of key separation (see, e.g., [23]). These weaknesses are easily solved by deriving different, and computationally independent, keys from the DH value g^{xy}, one used internally in the protocol for encryption and the other as the session key output by the protocol.

without proving knowledge of the corresponding signature key. (We note that while such "proof of possession" is required by some CAs for issuing a certificate, this is not a universal requirement for public key certificates; in particular it is not satisfied in many "out-of-band distribution" scenarios, webs of trust, etc.) In this case Eve can register A's public key as its own and then simply replace A's identity (or certificate) in the third message of STS with her own. B verifies the incoming message and accepts it as coming from Eve. Thus, in this case the STS protocol fails to defend against the misbinding attack. Therefore, for the STS to be secure one must assume that a secure external mechanism for proof of possession of signature keys is enforced. As we will see both the ISO protocol discussed in Section 4 and the SIGMA protocols presented here do not require such a mechanism. Moreover, even under the assumption of external "proof of possession" the above STS protocol has not been proven secure.

Note. In [34] an analysis of the STS protocol based on an extension of BAN logic [6] is presented. However, the modeling of the encryption function in that analysis is as a MAC function. Therefore this analysis holds for the MAC variant of STS presented in the next subsection. However, as we will see, for considering that protocol secure one needs to assume that the CA verifies that the registrant of a public key holds the corresponding private key (proof of possession) and, moreover, that "on-line registration" attacks as discussed in 3.3 are not possible.

What is the reason for this protocol failure? The main reason is to assume that the combination of proof of possession of the session key together with the signature on the DH exponentials provide a sufficient binding between the identities of the (honest) peers participating in the exchange and the resultant key. However, as the above attack shows this is not true in general. Can this shortcoming be corrected? One first observation is that encryption is not the right cryptographic function to use for proving knowledge of a key. Being able to encrypt a certain quantity under a secret key is no proof of the knowledge of that key. Such a "proof of key possession" is not guaranteed by common modes of encryption such as CBC and is explicitly violated by any mode using XOR of a (pseudo) random pad with the plaintext (such as counter or feedback modes, stream ciphers, etc.). To further illustrate this point consider a seemingly stronger variant of the protocol in which not only the signature is encrypted but also the identity (or full certificate) of the signer is encrypted too. In this case the above attack against STS is still viable if the encryption is of the XOR type discussed above. In this case, when A sends the message $\{\, A\,,\ \mathrm{SIG}_A(g^y, g^x)\,\}_{K_s}$, Eve replaces A's identity (or certificate) by just XORing the value $A \oplus E$ in the identity location in the ciphertext. When decrypted by B this identity is read as E's and the signature verified also as E's. Thus we see that even identity encryption does not necessarily prevent the attack. As we will see in the next section replacing the encryption with a MAC function, which is better suited to prove possession of a key, is still insufficient to make the protocol secure.

3.3 Two STS variants: MACed-signature and Photuris

In [11] (see also [29]) a variant of the basic STS protocol is mentioned in which the encryption function in the protocol is replaced with a message authentication (MAC) function. Namely, in this STS variant each party in the protocol applies its signature on the DH exponentials plus it concatenates to it a MAC on the signature using the key K_s. For example, the last message from A to B in this protocol consists of the triple (A, b, c) where $b = \mathrm{SIG}_A(g^y, g^x)$ and $c = \mathrm{MAC}_{K_s}(b)$. In [11] this variant is not motivated as a security enhancement but as an alternative for situations –such as export control restrictions– in which the use of a strong encryption function is not viable. However, considering that a MAC function is more appropriate for "proving knowledge of a key" than an encryption function (as exemplified above) then one could expect that this variant would provide for a more secure protocol. This is actually incorrect too. The above attack on basic STS (where Eve records the public key of A under her name) can be carried exactly in the same way also in this MAC-based variant of the protocol. Same for the case where on top of the signature and identities (or even on top of the MAC) one applies an encryption function of the XOR type.

Moreover, if (as it is common in many application) A and B communicate their public key to each other as part of the KE protocol (i.e., the identities A and B sent in the protocol include their corresponding public-key certificates), then this MAC-ed signature variant is not secure even if the system does ensure that the registrant of a public key knows the corresponding private key! This has been shown by Blake-Wilson and Menezes [5] who present an ingenious *on-line registration attack* against the protocol. In this form of attack, the attacker Eve intercepts the last message from A to B and then registers a public key (for which she knows the private key) that satisfies $\text{SIG}_E(g^y, g^x) = \text{SIG}_A(g^y, g^x)$. Eve then replaces the certificate of A with her own in the intercepted message and forwards it to B (leaving the signature and mac strings unchanged from A's original message). Clearly, B will accept this as a valid message from Eve since both signature and mac will pass verification. In other words, Eve successfully mounted an identity-misbinding attack against the MACed-signature protocol. In [5] it is shown that this on-line registration attack can be performed against natural signature schemes. In particular, it is feasible against RSA signatures provided that the registrant of the public key can choose her own public RSA exponent.[4] While the full practicality of such an attack is debatable, it certainly suffices to show that one cannot prove this protocol to be secure on the basis of generic cryptographic functions, even under the assumption that the CA verifies possession of the private signature key. As a final note on this attack, we point out that this attack is possible even if the protocol is modified in such a way that each peer includes its own identity under the signature (something that can be done to avoid the need for "proof of possession" in the public-key registration stage).

From the above examples we learn that the failure to the misbinding attack is more essentially related to the *insufficiency* of binding the DH key with the signatures. Such a binding (e.g., via a MAC computed on the signature) provides a proof that *someone* knows the session key, but does not prove *who* this *someone* is. As we will see later, the essential binding here needs to be done between the signature and the recipient's identity (the ISO protocol), or between the DH key and the sender's identity (the SIGMA protocol).

Photuris. We finish this section by showing the insecurity of another variant of the STS protocol described in [29] and used as the core cryptographic protocol in Photuris [17] (an early proposal for a KE protocol for IPsec). As the previous variants, this one is also illustrative of the subtleties of designing a good KE protocol. This variant dispenses of the use of encryption or MAC; instead it attempts at binding the DH key to the signatures by including the DH key g^{xy} under the signature:

$$A \xrightarrow{\qquad\qquad\qquad g^x \qquad\qquad\qquad} B$$

$$\xleftarrow{\qquad g^y\ ,\ B\ ,\ \text{SIG}_B(g^x, g^y, g^{xy}) \qquad}$$

$$\xrightarrow{\qquad A\ ,\ \text{SIG}_A(g^y, g^x, g^{xy}) \qquad}$$

An obvious, immediate, complaint about this protocol is that the DH key g^{xy} is included under the signature, and therefore any signature that leaks information on the signed data (for example, any

[4] In this case, Eve uses an RSA public modulus equal to the product of two primes p and q for which computing discrete logarithms is easy (e.g., all factors of $p-1$ and $q-1$ are small), and calculates the private exponent d for which $(hash(g^y, g^x))^d$ equals the signature string sent by A.

signature scheme that provides "message recovery") will leak information on g^{xy}. This problem is relatively easy to fix: derive two values from g^{xy} using a one-way pseudorandom transformation (as in Appendix C); use one value to place under the signature, and the other as the generated session key. A more subtle weakness of the protocol is that it allows, even with the above enhancement, for an an identity misbinding attack whenever the signature scheme provides for message recovery (e.g. RSA). In this case the attacker, Eve, proceeds as follows: it lets the protocol proceed normally between A and B for the first two messages, then it intercepts the last message from A to B and replaces it with the message

$$E \xrightarrow{\quad E\,,\ \mathrm{SIG}_E(g^y, g^x, g^{xy}) \quad} B$$

But how can E sign the key g^{xy} (or a value derived from it) if it does not know g^{xy}? For concreteness assume that $\mathrm{SIG}_A(M) = RSA_A(hash(M))$, for some hash (or encoding) function $hash$. Since Eve knows A's public key it can invert A's signature to retrieve $hash(g^y, g^x, g^{xy})$, and then apply its own signature $RSA_E(hash(g^y, g^x, g^{xy}))$ as required to carry the above attack! (Note that this attack does not depend on any of the details of the public-key registration process; the attacker uses its legitimately generated and registered public key.)

Photuris included the above protocol as an "authentication only" solution, namely, one in which identities are not encrypted. It also offered optional identity protection by applying encryption on top of the above protocol. In the later case the above simple misbinding attack does not work. Yet, even in this case no proof of security for such a protocol is known. The above protocol (without encryption) is also suggested as an STS variant in [29] where it is proposed to explicitly hash the value g^{xy} before including it under the signature.

Remark: In this STS variant [29] the value g^{xy} under the signature is replaced with $h(g^{xy})$ where h is a hash function. This explicit hashing of g^{xy} seems to be intended to protect the value g^{xy} in case that the signature in use reveals its input. While this is not sufficient to defend against our identity misbinding attack, it is interesting to check whether revealing the value $h(g^{xy})$ may be of any use to an eavesdropper (note that in this case the attacker has the significantly simpler task of passively monitoring the protocol's messages rather than actively interfering with the protocol as required to carry the misbinding attack). Certainly, learning $h(g^{xy})$ is sufficient for distinguishing the key g^{xy} from random (even if the hash function acts as an ideal "random oracle"). But can the attacker obtain more than that? To illustrate the subtle ways in which security deficiencies may be exploited, consider the following practical scenario in which the function h is implemented by SHA-1 and the key derivation algorithm defines the session key to be K_s = HMAC-SHA1$_{g^{xy}}(v)$, where v is a non-secret value. The reader can verify (using the definition of HMAC in [24]) that in this case the attacker does not need to find g^{xy} for deriving the session key K_s, but it suffices for her to simply know SHA-1(g^{xy}). Therefore if this later value is revealed by the signature then the security of the protocol is totally lost. Not only this example shows the care required in designing these protocols, but it also points to the the potential weaknesses arising from protocols whose security cannot be claimed in a generic (i.e. algorithm-independent) way.

4 The ISO Protocol

Here we recall the ISO KE protocol [15] which similarly to STS uses digital signatures to authenticate a DH exchange[5]. However, the ISO protocol resolves the problem of key-identity binding demonstrated by the misbinding attack on the BADH protocol (see Section 3.1) differently. The protocol simply adds the identity of the *intended recipient* of the signature to the signed information. Specifically, the protocol is:

$$A \xrightarrow{\quad A\,,\; g^x \quad} B$$

$$\xleftarrow{\quad B\,,\; g^y\,,\; \text{SIG}_B(g^x, g^y, A) \quad}$$

$$\xrightarrow{\quad \text{SIG}_A(g^y, g^x, B) \quad}$$

It is not hard to see that the *specific* identity misbinding attack as described in Section 3.1 is avoided by the inclusion of the identities under the signatures. Yet having seen the many subtleties and protocol weaknesses related to the STS protocols in the previous section it is clear that resolving one specific attack is no guarantee of security. Yet the confidence in this protocol can be based on the analytical work of [7] where it is shown that this is a secure KE protocol (under the security model of that work). It is shown there that any feasible attack in that model against the security of the ISO protocol can be transformed into an efficient cryptanalytical procedure against the DH transform or against the digital signature function scheme in use.

The above version of the ISO protocol is simple and elegant. It uses a minimal number of messages and of cryptographic primitives. It allows for delaying computation of the DH key g^{xy} to the end of the interaction (since the key is not used inside the protocol itself) thus reducing the effect of computation on protocol latency. The protocol is also minimal in the sense that the removal of any of its elements would render the protocol insecure. In particular, as demonstrated by the BADH protocol, the inclusion of the recipient's identity under the signature is crucial for security. It is also interesting to observe that replacing the recipient's identity under the signature with the signer's identity results in an insecure protocol, open to the identity-misbinding attack exactly as in the case of BADH.

Therefore, it seems that we have no reason to look for other DH protocols authenticated with digital signatures. This is indeed true as long as "identity protection" is not a feature to be supported by the protocol. As explained next, in spite of all its other nice properties the ISO protocol does *not* satisfactorily accommodate the settings in which the identities of the participants in the protocol are to be concealed from attackers in the network (especially if such a protection is sought against active attacks).

The limitation of the ISO protocol in providing identity protection comes from the fact that in this protocol each party needs to know the identity of the peer before it can produce its own signature. This means that no party to the protocol (neither A or B) can authenticate the other party before it reveals its own name to that party. This leaves *both* identities open to active attacks.

[5]Strictly speaking, the protocol presented here is a simplification of the protocol in [15]. The latter includes two elements that are redundant and do not contribute significantly to the security of the protocol and are therefore omitted here. These elements are the inclusion of the signer's identity under the signature and an additional MAC value. In contrast to SIGMA, where the additional MAC is essential for security, the MAC in [15] serves only for explicit key confirmation (which adds little to the implicit key confirmation provided in the simplified variant discussed here).

If the only protection sought in the protocol is against passive eavesdroppers then the protocol can be built as a 4-message protocol as follows:

$$
\begin{array}{ccc}
A & \xrightarrow{\qquad\qquad g^x \qquad\qquad} & B \\
 & \xleftarrow{\qquad\qquad g^y\,,\ \{\,B\,\}_{K_e} \qquad\qquad} & \\
 & \xrightarrow{\qquad \{\,A\,,\ \textsc{sig}_A(g^y, g^x, B)\,\}_{K_e} \qquad} & \\
 & \xleftarrow{\qquad \{\,\textsc{sig}_B(g^x, g^y, A)\,\}_{K_e} \qquad} &
\end{array}
$$

where K_e is an encryption key derived from the DH key g^{xy}. We note that with this addition of encryption the ISO protocol looses several of its good properties (in particular, the minimality discussed above and the ability to delay the computation of g^{xy} to the end of the protocol) while it only provides partial protection of identities since both identities are trivially susceptible to active attacks.

Another privacy (or lack of privacy) issue related to the ISO protocol which is worth noting is that by signing the peer's identity each party to the protocol leaves in the hands of the peer a signed (undeniable) trace that the communication took place (see the discussion at the end of Section 2.2).

The SIGMA protocol presented in the next section provides better, and more flexible, support for identity protection with same or less communication and computational cost, and with a full proof of security.

Remark (*an identity-protection variant of the ISO protocol*)**:** We end this section by suggesting an adaptation of the ISO protocol to settings requiring identity protection (of one of the peers) to active attacks. We only sketch the idea behind this protocol. The idea is to run the regular ISO protocol but instead of A sending its real identity in the first message it sends an "alias" computed as $\hat{A} = hash(A, r)$ for a random r. Then B proceeds as in the basic protocol but includes the value $\hat{A}$ under its signature instead of A's identity; it also uses the key g^{xy} to encrypt its own identity and signature. In the third message A reveals its real identity 'A' and the value r used to compute $\hat{A}$. It also sends its signature (with B's identity signed as in the regular ISO protocol). This whole message is privacy-protected with encryption under K_e. The above protocol can be shown to be secure under certain assumptions on the hash function *hash*. Specifically, this function needs to satisfy some "commitment" properties similar to those presented in [25].

We omit further discussion of this protocol and proceed to present the SIGMA protocol that provides a satisfactory and flexible solution to the KE problem suitable also for settings with identity protection requirements, and with less requirements on the underlying cryptographic primitives than the above "alias-based" ISO variant.

5 The SIGMA Protocols

The weaknesses of the STS variants (which provide identity protection but not full security in general) and the unsuitability of the ISO protocol for settings where identity protection is a requirement, motivated our search for a solution that would provide solid security for settings where identity protection may or may not be a requirement. The result is the SIGMA protocols that we present in this section and whose design we explain based on the design lessons learned through the examples presented in previous sections (as well as many other in the literature). SIGMA takes from STS the property that each party can authenticate to the other without needing to know the peer's identity (recall that the lack of this property in the ISO protocol makes that protocol inappropriate to support identity protection). And it takes from ISO the careful binding between identities and keys, but it implements this binding in a very different way. More specifically, SIGMA decouples the authentication of the DH exponentials from the binding of key and identities. The former authentication task is performed using digital signatures while the latter is done by computing a MAC function keyed via g^{xy} (or more precisely, via a key derived from g^{xy}) and applied to the sender's identity. This "SIGn-and-MAc" approach is the essential technique behind the design of these protocols and the reason for the SIGMA acronym.

As pointed out in Section 2.3, we focus on the cryptographic core of the protocol leaving important system and implementation details out of the discussion. In particular, as we will also note below, in the following presentation we overcharge the DH exponentials with the added functionality of session id's and freshness nonces. (A "full fledge" SIGMA instantiation with a more careful treatment of these elements is presented in Appendix B.)

5.1 The basic SIGMA protocol

The most basic form of SIGMA (without identity protection) is the following:

$$
\begin{array}{ccc}
A & \xrightarrow{\qquad\qquad\qquad g^x \qquad\qquad\qquad} & B \\
 & \xleftarrow{\quad g^y\,,\,B\,,\,\mathrm{SIG}_B(g^x,g^y)\,,\,\mathrm{MAC}_{K_m}(B) \quad} & \\
 & \xrightarrow{\quad A\,,\,\mathrm{SIG}_A(g^y,g^x)\,,\,\mathrm{MAC}_{K_m}(A) \quad} &
\end{array}
$$

The output of the protocol is a session key K_s derived from the DH value g^{xy} while the key K_m used as a MAC key in the protocol is also derived from this DH value. It is essential for the protocol security that the keys K_m and K_s be "computationally independent" (namely no information on K_s can be learned from K_m and vice-versa).[6] Note that this basic protocol does not provide identity protection. This will be added on top of the above protocol using encryption (see following sections). The important point is that SIGMA's security is built in a modular way such that its core cryptographic security is guaranteed independently of the encryption of identities. Thus, the same design serves for scenarios requiring identity protection but also for the many cases where such protection is not an issue (or is offered only as an option). We note that the identities A and B transmitted in messages 2 and 3 may be full public-key certificates; in this case the identities included under the MAC may be the certificates themselves or identities bound to these certificates.

[6]We discuss specific ways to derive these values from g^{xy} in Appendix C.

The first basic element in the logic of the protocol is that the DH exponential chosen by each party is protected from modification (or choice) by the attacker via the signature that the party applies to its own exponential. We note that the inclusion of the peer's exponential under the signature is not mandatory and can be replaced with a nonce freshly chosen and communicated by the peer (see Appendix B). Yet, either the peer's exponential (if chosen fresh and anew in each session) or a fresh nonce must be included under the signature; otherwise the following *replay attack* is possible. It would suffice for the attacker to learn the exponent x of a single ephemeral exponential g^x used by a party A in one session for the attacker to be able to impersonate A on a KE with any other party (simply by replaying the values g^x and $\text{SIG}_A(g^x)$). In this case, A's impersonation by the attacker is possible even without learning A's long-term signature key. This violates the security principle (see Section 2.1) by which the exposure of ephemeral secrets belonging to a specific session should not have adverse effects on the security of other sessions.

The second fundamental element in SIGMA's design is the MACing of the sender's identity under a key derived from the DH key. This can be seen as a "proof of possession" of the DH key, but its actual functionality is to *bind* the session key to the identity of each of the protocol participants in a way that ensures the "consistency" requirement of KE protocols. As discussed in Section 2.1, this is a fundamental requirement needed, in particular, to avoid attacks such as the identity misbinding attacks from Section 3. Note that without this MACing the protocol "degenerates" into the BADH protocol from Section 3.1 which is susceptible to this attack. Therefore we can see that all the elements in the protocol are mandatory (up to replacement of the peer's exponential under the signature with a fresh nonce).

We note that the above SIGMA protocol, as well as all the following variants, satisfy all the security guarantees discussed in Section 2.1. In particular, they provide "perfect forward secrecy" due to the use of the Diffie-Hellman exchange. This assumes that DH exponentials are chosen anew and independently for each session, that the exponents x, y used to generate the DH exponentials g^x, g^y are erased as soon as the computation of the key g^{xy} is completed, and that these exponents are not derivable from any other quantity stored in the party's computer after the session terminates (in particular, if x is generated pseudorandomly then the value of past exponents x should not be derivable from the present state of the pseudorandom generator). We note that SIGMA can allow for re-use of DH exponentials by the same party across different sessions. However, in this case the forward secrecy property is lost (or at least confined to hold only after all sessions using the same exponent x are completed and the exponent x erased). In case of re-use of DH exponents one must derive the keys used by the session (e.g. K_m, K_s) in a way that depends on some session-specific non-repeating quantity (such as a nonce or session-id). Also, as discussed before, in this case such a fresh nonce needs to be included under the peer's signature (also see the end of Appendix B for a note on the importance of correctly positioning nonces under the signature). There are other, more theoretical, issues concerning the re-use of DH exponents that are not treated here.

As we have stressed earlier in the paper, this informal outline of the design rationale for SIGMA does not constitute a proof of security for the protocol. The formal analysis in which we can base our confidence in the protocol appears in the companion analysis paper [8].

5.2 Protecting identities: SIGMA-I

Recall that SIGMA is designed to serve as a secure key-exchange protocol both in settings that do not require identity protection (in which case the above simple protocol suffices) or those where identity protection is a requirement. The main point behind SIGMA's design that allows for easy addition of identity protection is that the peer's identity is not needed for own authentication. In particular, one of the peers can delay communicating its own identity until it learns the peer's identity in an authenticated form. Specifically, to the basic SIGMA protocol we can add identity protection by simply encrypting identities and signatures using a key K_e derived from g^{xy} (K_e must be computationally independent from the authentication key K_m and the session key K_s):

$$A \quad \xrightarrow{\qquad\qquad\qquad\qquad g^x \qquad\qquad\qquad\qquad} \quad B$$

$$\xleftarrow{\quad g^y\,,\ \{\, B\,,\ \mathrm{SIG}_B(g^x, g^y)\,,\ \mathrm{MAC}_{K_m}(B)\,\}_{K_e} \quad}$$

$$\xrightarrow{\quad \{\, A\,,\ \mathrm{SIG}_A(g^y, g^x)\,,\ \mathrm{MAC}_{K_m}(A)\,\}_{K_e} \quad}$$

This protocol has the property that it protects the identity of the initiator from active attackers and the identity of the responder from passive attackers. Thus, the protocol is suitable for situations where concealing the identity of the initiator is considered of greater importance. A typical example is when the initiator is a mobile client connecting to a remote server. There may be little or no significance in concealing the server's identity but it may be of prime importance to conceal the identity of the mobile device or user. We stress that the encryption function (as applied in the third message) must be resistant to active attacks and therefore must combine some form of integrity protection. Combined secrecy-integrity transforms such as those from [16] can be used, or a conventional mode of encryption (e.g. CBC) can be used with a MAC function computed on top of the ciphertext [3, 26]. Due to the stronger protection of the identity of the **I**nitiator of the protocol we denote this variant by SIGMA-I.

We remark that while this protocol has the minimal number of messages that any KE protocol resistant to replay attacks (and not based on trusted timestamps) can use, it is sometimes desirable to organize the protocol in full round-trips with each pair of message containing a "request message" and a "response message". If so desired, the above protocol can add a fourth message from B to A with a simple ACK authenticated under the authentication key K_m. This ACK message serves to A as a proof that B already established the key and communications protected under the exchanged key K_s can start. It also provides the flexibility for A to either wait for the ACK or start using the session key as soon as it sent the third protocol message. (Depending on B's policy this traffic may be accepted by B if the channel – or "security association" in the language of IKE – was already established by B, or discarded if not, or queued until the key establishment is completed.) Finally, it is worth noting that this ACK-augmented protocol provides the *peer awareness* property discussed in Section 2.1. (This is in contrast to the other variants of SIGMA presented here which do not enjoy this property.)

5.3 A four message variant: SIGMA-R

As seen, SIGMA-I protects the initiator's identity against active attacks and the responder's against passive attacks. Here we present SIGMA-R which provides defense to the responder's identity against active attacks and to the initiator's only against passive attacks. We start by presenting a simplified version of SIGMA-R without encryption:

$$
\begin{array}{ccc}
A & \xrightarrow{\qquad\qquad g^x \qquad\qquad} & B \\
 & \xleftarrow{\qquad\qquad g^y \qquad\qquad} & \\
 & \xrightarrow{\quad A\,,\ \mathrm{SIG}_A(g^y, g^x)\,,\ \mathrm{MAC}_{K_m}(A) \quad} & \\
 & \xleftarrow{\quad B\,,\ \mathrm{SIG}_B(g^x, g^y)\,,\ \mathrm{MAC}_{K_m}(B) \quad} &
\end{array}
$$

The logic of the protocol is similar to that of the basic SIGMA protocol from Section 5.1. The difference is that B delays the sending of its identity and authentication information to the fourth message after it verified A's identity and authentication in message 3. This "similarity" in the logic of the protocol does not mean that its security is implied by that of the 3-message variants. Indeed, the protocol as described above is open to a *reflection attack* that is not possible against the 3-message variant. Due to the full symmetry of the protocol an attacker can simply replay each of the messages sent by A back to A. If A is willing to accept a key exchange with itself then A would successfully complete the protocol.[7] Therefore, to prevent this attack the protocol needs to ensure some "sense of direction" in the authenticated information. This can be done by explicitly adding different "tags" under the MAC for each of the parties (e.g., A would send $\mathrm{MAC}_{K_m}(\text{“0”}, A)$ while B would send $\mathrm{MAC}_{K_m}(\text{“1”}, B)$), or by using different MAC keys in each direction (i.e., instead of deriving a single key K_m from g^{xy} one would derive two keys, K_m and K'_m, where the former is used by A to compute its MAC and the latter by B). Any of these measures are sufficient to prevent the reflection attack and make the protocol secure [8] (another defense is for A to check that the peer's DH exponential is different than her own.)

The full protocol SIGMA-R (with identity protection) is obtained by encrypting the last two messages in the above depicted protocol (and adding a reflection defense as discussed before). A "full fledge" illustration of protocol SIGMA-R is presented in Appendix B.

Remark (*The inter-changeability property of SIGMA*). It is worth noting that the last two messages in the above protocol can be interchanged. Namely, B may proceed as described in SIGMA-R and wait for the reception of A's message (message 3 in the above picture) before sending his last message. But B may also decide to send his last message (signature and mac) immediately after, or together with, message 2 (which results in SIGMA-I). In this way, B may control if he is interested in protecting his own identity from active attacks or if he prefers to favor a faster exchange. The protocol may also allow for messages 3 and 4 to cross in which case the protocol is still secure but both identities may be open to active attacks.

5.4 Further variants and the use of SIGMA in IKE

As seen above the MAC of the sender's identity is essential for SIGMA's security. Here we present a variant of the protocol that differs from the above descriptions by the way the MAC value is placed in the protocol's messages. Specifically, the idea is to include the MAC value under the signature (i.e., as part of the signed information). The interest on this variant is that it saves in

[7] The only damage of this attack seems to be that it forces A to use a key derived from the distribution g^{x^2} rather than g^{xy}. These distributions may be distinguishable depending on the DH groups.

message length by avoiding explicit sending of the MAC value, and more significantly because it is the variant of SIGMA adopted into the IKE protocols (both IKE version 1 [14] and version 2 [19]).

The MAC moved under the signature may cover just the identity of the sender or the whole signed information. For example, in B's message the pair $(\text{SIG}_B(g^x, g^y)\,,\ \text{MAC}_{K_m}(B))$ is replaced with either (i) $\text{SIG}_B(g^x, g^y, \text{MAC}_{K_m}(B))$ or (ii) $\text{SIG}_B(\text{MAC}_{K_m}(g^x, g^y, B))$. In this way the space for an extra MAC outside the signature is saved, and the verification of the MAC is merged with that of the signature. In either case, *as long as the* MAC *covers the identity of the signer* then the same security of the basic SIGMA protocol (as well as SIGMA-I and SIGMA-R) is preserved[8] [8]. Variant (ii) is used in the IKE protocol (version 1) [14] in two of its authentication modes: the signature-based exchange of IKE uses the basic 3-message SIGMA protocol (without identity encryption) as presented in Section 5.1 for its aggressive mode, and it uses the 4-message SIGMA-R in its main mode. (In the later case, the use of SIGMA-R in IKE is preceded by two extra messages for negotiating security parameters.) In IKE the MAC function is implemented via a pseudorandom function which is also used in the protocol for the purpose of key expansion and derivation.[9] IKE version 2 [19] uses variant (i) with SIGMA-R as its single key exchange method authenticated with public keys. In this protocol the peer's DH exponential is not signed; the essential freshness guarantee is provided by signing a nonce chosen by the peer (see Section 5.1).

The SIGMA-R protocol has also been adopted in the JFK protocol [1] which has been proposed in the context of the revision of the IKE protocol. We note that in both [19, 1] protocol SIGMA-R is augmented with mechanisms that provide some defense against Denial-of-Service attacks as discussed in Section 2.3.

Acknowledgment

I wish to thank Ran Canetti for embarking with me on a long and challenging journey towards formalizing and proving the security of key-exchange protocols (in particular, the SIGMA protocols). Special thanks to Dan Harkins for being receptive to my design suggestions when specifying the signature modes of IKE, and to Charlie Kaufman (and the IPsec WG) for incorporating this design into IKEv2. Thanks to Sara Bitan and Paul Van Oorschot for many useful discussions, and to Pau-Chen Cheng for sharing much of his implementation and system experience with me. This research was partially funded by the Irwin and Bethea Green & Detroit Chapter Career Development Chair.

[8] A technicality here is that moving the MAC inside is possible only for MAC functions whose verification is done by recomputation of the MAC value; this is the case for all common MAC functions, in particular when the MAC is implemented via a pseudorandom function as in IKE [14, 19].

[9] This use of a prf –as a MAC – under the signature has been a source of confusion among analysts of the IKE protocol; the prf was sometimes believed to have some other functionality related to the signature. It is important then to realize that its functionality under the signature is simply (and essentially!) that of a MAC covering the signer's identity.

A A Definition of Secure Key-Exchange Protocols

Here we expand on the security definition outlined in Section 2.1. For a full formalism see [8].

Recall that we consider a KE protocol as a protocol specified to run between pairs of parties which wish to establish a secret key known to the participating parties only. The communications environment is a multi-party network where any party may run the protocol with any other parties. Each execution of a protocol run within a party is called a session, and multiple sessions by the same or different parties may run concurrently. When a KE session completes at a party the output of that session is (a) a session identifier that uniquely identifies the session at that party, (b) the name of the identified peer to the session, and (c) the value of a (secret) session key.

Note: We call a KE session completed at the point where the session key is output by the protocol; at this point the local state of a KE session is erased and the session key is passed to the application that requested it (e.g. an application, such as IPsec's ESP [21], intended to authenticate and/or encrypt communications traffic). We also note that while we refer to a session key as a single key, further keys may be produced by the session or be derived from this single session key.

The adversarial model has been summarized in Section 2.1. In essence, the communication links between the parties are controlled by a fully active attacker, which also has control of the scheduling of sessions and message delivery. The attacker may also corrupt any party (in which case the attacker learns the long-term secret information held by that party, and can impersonate it at will), and may expose a session by learning secret data related to that specific session (such as ephemeral session state information or the output session key).

Security of key-exchange protocols. On the basis of the above attacker capabilities we outline the definition of security for a KE protocol. We first recall (see Section 2.1 that we consider a *session* as a local object run at a party. In particular, when two parties A and B interact in a run of the KE protocol they each have a local session corresponding to this run. Sessions are denoted by the name of the party holding the session and a session identifier. The formal treatment in [7, 8] uses the notion of "matching sessions" to denote sessions that are related to the "same exchange". Here we simplify our presentation by implicitly referring to matching sessions as those that have the same session identifier. In practice this requires that parties create session identifiers interactively (before or during the KE run). Specifically, we assume the common practice where (as part of the protocol) A sends to B a value sid_A, B sends to A a value sid_B, and they both define the session identifier as $s = (sid_A, sid_B)$ (for an example see the full-fledge protocol in Appendix B, or the IKE protocols [14, 19] where these session identifiers are called "cookies" and SPIs, respectively). In this case we name the local session at A as session (A, s) and the local session at B as (B, s). Note that each party needs to choose its local session id sid to be unique among all sessions at that party. This suffices to ensure the uniqueness of s at that party. Thus, there is no need for the party to keep a global view of session identifiers at other parties, or to depend on the choice of sid by the peer.

We also use the following notation: if a party P completes a session (P, s) with output (s, Q, k) (denoting the session-id, the peer to the session, and the session key, respectively) then we write $peer(P, s) = Q$ and $sk(P, s) = k$.

Definition. We say that a key-exchange protocol is secure (in the adversarial setting described above) if the following holds. Let (P, s) be a session that completes at an uncorrupted party P with $peer(P, s) = Q$. Then:

1. If Q completes session (Q, s) while P and Q are uncorrupted then

(a) $peer(Q, s) = P$; and

(b) $sk(Q, s) = sk(P, s)$.

2. If the sessions (P, s) and (Q, s) are not exposed then the attacker cannot distinguish $sk(P, s)$ from a random value[10].

This definition is somewhat stronger than the definition of security in [8] which does not guarantee condition 1(a) (and it guarantees 1(b) only in case that 1(a) holds). We use this stronger definition here since it is simpler to state and is satisfied by the SIGMA protocols. (In contrast, some natural KE protocols, such as the *ENC* protocol from [7], satisfy the definition from [8] but not the above stronger variant.)

Proving security. The above security model and definition is aimed at capturing a small set of requirements for key-exchange protocols that when satisfied provide assurance for many other desired security properties and resistance to a large variety of attacks. In particular, they cover in a systematic way different attack scenarios, without necessitating of an exhaustive enumeration of the attacks. Very importantly, this compact mathematical formulation of security allows for security proofs to be carried in this model. In particular, such proofs are provided for a variety of KE protocols in [7] and for the SIGMA protocols in [8]. Following the complexity-theoretic approach to the analysis of KE protocols initiated in [2], these papers show how to relate the security of the protocol to the security of the underlying cryptographic functions. Moreover, this analysis is "constructive" in the sense that *any feasible attack strategy* that breaks the security requirements in the model can be transformed into an explicit efficient algorithm to break one of the cryptographic functions used in the protocol (in the case of SIGMA, for example, this may be an attack against the basic DH transform, or a forgery attack against the signature or MAC schemes used in the protocol). Therefore, as long as the underlying functions are secure so is the KE protocol. Note that this analysis is done on the basis of the *generic* requirements from these cryptographic functions and does not depend on their specific instantiation (we usually refer to this algorithm-independence property as *generic security*).

Finally, we remark that not only the above security formulation provides a strong basis for the analysis of protocols, but actually serves as a design tool too. By understanding the requirements that arise from this security model one can derive clear security principles applicable to the design of specific protocols. For example, these requirements make clear the need for (i) deriving fresh keys for each session; (ii) avoiding the use of the session key *during* the KE run (which in turn requires careful key derivation techniques); (iii) maintaining the (computational) independence between keys of different sessions; (iv) preventing unnoticed replay of old messages; (v) using fresh session identifiers for binding messages to particular sessions; and (vi) the essential role of a careful binding between sessions, identities and keys. In designing a KE protocol all these elements MUST appear in the protocol or otherwise security in the above model cannot be guaranteed. On the other hand, understanding the role of each element in the design simplifies the resultant protocol by avoiding the need to add precautionary "safety margins".

B A "full fledge" Protocol

As "disclaimed" in Section 2.3 our presentation of the key-exchange protocols in this paper shows only the cryptographic skeleton of these protocols. When embedding these protocols in real ap-

[10] More precisely, the probability of the attacker to win the distinguishing game when (P, s) is chosen as the test session is negligibly larger than 1/2 (see [8]).

plications one has to add to these protocols additional information related to the choice (or negotiation) of security parameters, environmental data (such as network protocol information), etc. Very importantly, the protocols should separate the functions of DH exponentials and freshness nonces into different elements (something that our presentation avoids in the name of simplicity and "compatibility" with the presentation in other papers). In addition, protocols need to include session identifiers that serve to match incoming messages with new or existing sessions as well as to identify exchanged keys with their corresponding sessions. For illustration purposes we present in this appendix a more general form of the protocol SIGMA-R (from Section 5.3) in which some of the elements missing in our simplified presentation in Section 5 are shown explicitly.

$$A \xrightarrow{\quad sid_A\,,\; g^x, n_A\,,\; info_A^1 \quad} B$$

$$\xleftarrow{\quad sid_A\,,\; sid_B\,,\; g^y\,,\; n_B\,,\; info_B^1 \quad}$$

$$\xrightarrow{\quad sid_A, sid_B, \{\, info_A^2, A, \text{SIG}_A(n_B, sid_A, g^x, info_A^1, info_A^2), \text{MAC}_{K_m}(A)\,\}_{K_e} \quad}$$

$$\xleftarrow{\quad sid_A, sid_B, \{\, info_B^2, B, \text{SIG}_B(n_A, sid_B, g^y, info_B^1, info_B^2), \text{MAC}_{K'_m}(B)\,\}_{K'_e} \quad}$$

Here sid stands for the session identifier chosen by each party for the ongoing session; the value sid_A chosen by A is returned in the response messages by B, and similarly sid_B is added in messages from A to B (except for the initial message). The nonces n_A and n_B are chosen freshly and anew with each session by A and B, respectively, and they serve to guarantee freshness of the exchanged key and to protect against replay[11]. (We note that some protocols may specify that nonces serve for the dual purpose of freshness guarantee and session identifiers.) The *info* fields represent additional generic information that can be carried in the protocol messages. The letters A and B carried in the messages denote the identities of the participants: they may be addresses, logical names, full public key certificates, etc. The actual security of the protocol depends on a correct binding between these identities, the public keys used to verify the signature, and the internal policy of each party that specifies whether a key-exchange with that party is to be completed or not. The encryption functionality included in this illustrative protocol may be applied optionally (we also note that in the general case the third and fourth messages may carry an *info* field internal to the encryption and another such field in cleartext form). If encryption is applied, and identity protection of the responder B is sought against active attackers, then (at least in the case of message 4) the encryption function must be secure against active attacks, e.g. it may use a regular encryption mode with a MAC function computed on top of it. The keys K_m, K'_m, K_e, K'_e used in the protocol (to key the MAC and encryption functions, respectively) as well as the output session key K_s are all derived from the DH key g^{xy} in a computationally independent way, e.g. by carefully using a pseudorandom function (see Appendix C).

In our above illustration of a full-fledge protocol, we chose to sign the essential fields in the protocol. As a general design rule, however, it is recommended that a party sign the peer's nonce concatenated with the *whole* information sent during the protocol by the signing party. Moreover, if the same signature keys are used for other applications then the information signed in the protocol should also include some "context information" (such as protocol name, message number, etc.). As a final note, it is essential that protocols specify that the signature covers the peer's nonce; it is also

[11]Note that in this full-fledge version the peer' DH exponential is not signed but the peer's nonce is.

important that nonces are positioned in fixed locations in all signatures (e.g., always as the first item under the signature or always at the end). Having the nonce in changing positions may open protocols to attack. As a simple example, consider the basic SIGMA protocol from Section 5.1. In this protocol one uses the peer's DH exponential with the functionality of a nonce. One could change the protocol to specify that in the signature from B to A (second message) the nonce (i.e. the peer's exponential) goes at the beginning of the data to be signed, while in the signature from A to B the peer's nonce goes second. (That is, the second message includes $\textsc{sig}_B(g^x, g^y)$ while the third includes $\textsc{sig}_A(g^x, g^y)$.) This "slight modification" is sufficient to make this protocol insecure: it is open to a serious reflection attack against B (left as an exercise...)

C Key Derivation

Key derivation is a fundamental component of any key-exchange protocol and, in particular, of the SIGMA protocols. Here we discuss two basic issues related to key derivation: (i) how to derive "computationally independent" keys from an initial "seed key"; and (ii) how to compute such a seed key from a Diffie-Hellman value g^{xy}. The former aspect is common to virtually all KE protocols, while the latter is required by Diffie-Hellman exchanges (SIGMA included). The key-derivation design discussed here is recommended for use with SIGMA (and can be applied to other KE protocols as well). On the other hand, SIGMA may remain secure with other key-derivation strategies as long as the cryptographic soundness of the "key independence" principle discussed below is preserved.

C.1 Derivation of multiple keys from a seed key

For simplicity we often think of KE protocols as outputting a (single) session key; however, in practice one usually needs to derive more than one key per session (e.g., a MAC key and an encryption key). Moreover, in some cases (and SIGMA is one of them) not only the KE protocol needs to provide a session key (or keys) to the calling application but it needs to derive keys used internally by the key exchange itself. (Such is the case of the MAC key used in all the SIGMA variants, and the additional encryption key needed to provide identity protection.) A fundamental principle is that all derived keys (whether used internally by the protocol or output by it) need to be computationally independent from each other. Roughly speaking, we need to ensure that given any information on one or more of these derived keys, all other derived keys remain secure. In technical terms this calls for the indistinguishability (usually from the uniform distribution) of any of these keys even when *all* other derived keys are given to the distinguisher.

Once a first key, k (which we call a "seed key"), is exchanged by the parties, all other keys can be derived in a computationally independent way through the use of a pseudorandom generator or a pseudorandom function family.[12] In the former case, the key k is used as a seed to the

[12]See [13] for a formal definition of pseudorandom function families. Informally, the main properties of these families are: (i) Each function in the family is determined by a key k (we usually denote by f_k the resultant function); knowledge of a key k allows to (efficiently and deterministically) compute the function f_k on any input. (ii) For an observer Eve that is not given the key k, the function f_k behaves essentially as a random function; in particular, seeing the function f_k computed on any set of values $v_1, v_2, \ldots, v_n$ (chosen by Eve) is of no help for deducing any information on $f_k(v)$ (other than its length) for any value v not in the above set. Pseudorandom functions are sometimes called "keyed hash functions" (an undefined and abused terminology that should be abandoned).
The most common implementations of pseudorandom functions include HMAC (based on cryptographic hash functions) and CBC-MAC (based on block ciphers – a variable-length input variant is XCBC-MAC [4]).

pseudorandom generator; in the latter, k is used as a key for selecting a specific function in the pseudorandom family. In both cases, a stream of ℓ pseudorandom bits is produced, where ℓ is the total number of key bits required internally and externally by the protocol (that is, ℓ is determined as the sum of lengths of all keys to be derived from k). When using a pseudorandom function to derive new keys from a seed key k, the simplest strategy is to compute the required stream of bits by successive computations: $f_k(1), f_k(2), f_k(3)$, etc. We refer to this usage of the pseudorandom function as "counter mode". A somewhat more conservative approach is to use the pseudorandom function in "feedback mode". In this case, the stream of ℓ key-bits is computed as the concatenation of a sequence of values $t_1, t_2, t_3, \ldots$, where $t_1 = f_k(c, 1)$ and for $i > 1$, $t_i = f_k(t_{i-1}, c, i)$. In this notation, the comma inside the function's argument represents concatenation, c is a "context value" (such as a string identifying a particular protocol or application, one or more nonces exchanged during the protocol, a sequence number related to the current run of the protocol, etc.)[13], and i represents a sequential numeric value (an integer, byte, etc). The "context value" may be useful to bind the current derivation to a particular protocol run or instance. The sequence value i ensures that all inputs to f_k are different. The "feedback value" t_i is used to make all the inputs to the pseudorandom function significantly different. (We note that even when using counter mode, it is advisable to use a context value as in feedback mode, yet this does not resolve the issue of closeness between inputs discussed next.)

This input-variability property is the main advantage of "feedback mode" over "counter mode". In the latter, the consecutive inputs to the function differ by very little (e.g., by a single bit); in contrast in feedback mode consecutive inputs differ very significantly due to the pseudorandom nature of the values t_i. We note that this differentiation between the two modes should be considered a prudent engineering practice rather than an academically founded principle. Indeed, if the function family $\{f_k\}$ behaves as a truly pseudorandom family then both modes are equally secure. On the other extreme, a total break of the family f may make it as easy to find the output values produced by feedback mode as those produced by counter mode. Yet, in practice, where we expect to see progressive cryptanalytical improvements, we recommend "feedback mode" as a more robust strategy for the imaginable case in which newly found weaknesses in the family f make it significantly easier to relate (for example) the values $t_2 = f_k(2)$ and $t_3 = f_k(3)$, than the values $t_2 = f_k(t_1, c, 1)$ and $t_3 = f(t_2, c, 2)$. This consideration has been the basis for our recommendation, and subsequent adoption, of "feedback mode" as the basic key-derivation technique in the IKE protocols (version 1 and 2) and the TLS protocol [10] (the key-derivation specification in IKEv2 is the "cleanest" via the `prf+` construct; IKEv1 and TLS include slight variants of this mechanism).

C.2 Derivation of a seed key from a Diffie-Hellman key

In Diffie-Hellman exchanges, especially those in which the peers share no prior secrets, the above seed key is to be computed from the Diffie-Hellman key g^{xy}. One may be tempted to use g^{xy} itself as a seed to a pseudorandom generator or as a key to a pseudorandom function. For example, using counter mode one would derive a stream of bits $f_{g^{xy}}(1), f_{g^{xy}}(2) \ldots$ However, note that there are two obstacles for this use of g^{xy} as a seed key. First, most pseudorandom generators and pseudorandom functions, do not accept seed keys of arbitrary length, especially given the considerable length of a Diffie-Hellman output.[14] Second, the value g^{xy} is not distributed uniformly over the set of strings

[13]The value c needs to be derivable from public information in the protocol; in particular, it should not include the secret key k.

[14]HMAC is an exception since according to its specification [24] keys longer than a block size (typically, 512 bits) are first hashed. Operationally, this allows the use of long keys in HMAC. Its analytical justification, however, requires

of certain length (as usually required for cryptographic keys), but rather over some mathematical (Diffie-Hellman) group. Moreover, in the case of non-prime order generators (this is the case for the standardized DH groups in IKE) there is explicit information that can be learned about the value g^{xy} from the exponentials g^x, g^y; e.g., the quadratic residuosity of g^{xy} is directly related to (and computable from) that of g^x and g^y. Therefore, we cannot use the value g^{xy} itself as a (seed) key but rather we need to derive a shorter and better randomized (i.e., computationally closer to uniform) key out of g^{xy}.

A well-known mechanism for achieving these two goals uses *strongly-universal hash functions (UH)* [9]. Using the so called "Leftover Hash Lemma" (see [13]; Lemma 3.5.1) one can apply to the DH value g^{xy} a randomly chosen function from a UH family (with a suitable output length) and obtain an output of the required length which is indistinguishable from random provided that the Diffie-Hellman key has sufficient "computational entropy" (the exact quantitative details are omitted from this informal discussion). It is important to note that the "randomness extraction" effect of a UH family holds also if the observer (say the attacker) not only knows g^x, g^y, but also the key that identifies the specific hash function being applied to the g^{xy} value. Thus, the following procedure ensures a secure (pseudorandom) output from any Diffie-Hellman exchange: in addition to a DH group, the protocol specifies a UH family; also, together with the DH exponentials g^x, g^y, the parties exchange (in the clear) random nonces r_1, r_2 from which a random key r is derived (e.g., via concatenation or XORing of r_1 and r_2). Now, both parties compute the (seed) key as $UH_r(g^{xy})$. This seed key is guaranteed to have the length and cryptographic strength required for further key derivation (using any of the techniques discussed in the previous subsection).

The above procedure, however, adds complexity to the specification and implementation of the KE protocol by requiring one more primitive in the form of a UH family. Our proposal (adopted into IKE, versions 1 and 2) is to use the above procedure, but with the pseudorandom function family (needed anyway in the protocol for derivation of keys from the seed key) acting as a UH family. In other words, the seed-key derivation follows the above description but $UH_r(g^{xy})$ is replaced with $f_r(g^{xy})$. Is this secure? Had the key r be secret (i.e. unknown to an attacker) then one could easily argue on the basis of the security properties of a pseudorandom family (specifically, its indistinguishability from random) that the derived seed key is secure. However, the above procedure does not hide r, and therefore one cannot claim the security of the seed key solely based on the standard security properties of a pseudorandom family. Moreover, we can show an explicit (specially tailored) example of a pseudorandom family that when used in the above way produces seed keys that are efficiently distinguishable from random. Yet, we make the *heuristic assumption* that natural (and reasonable) pseudorandom functions have the sufficient statistical (or combinatorial) strength to satisfactorily act as good "randomness extractors" similarly to strong universal hash families. In particular, this seems to be a plausible assumption (given current knowledge) regarding specific families such as HMAC-SHA1 or AES.

We will not expand further on these issues here. A more comprehensive analytical treatment of pseudorandom functions acting as randomness extractors is the subject of on-going work which we hope to publish shortly.

---

of special assumptions on the underlying hash function and therefore the approach suggested here applies also when the pseudorandom function is implemented via HMAC.

References

[1] B. Aiello, S. Bellovin, M. Blaze, R. Canetti, J. Ioannidis, A. Keromytis, O. Reingold, "Efficient, DoS-Resistant Secure Key Exchange for Internet Protocols," *ACM Computers and Communications Security conference (CCS)*, 2002.

[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] S. M. Bellovin, "Problem Areas for the IP Security Protocols", *Proceedings of the Sixth Usenix Unix Security Symposium*, 1996.

[4] Black, J. and P. Rogaway, "CBC MACs for Arbitrary-Length Messages: The Three-Key Constructions," *Advances in Cryptology, - CRYPTO'00*, LNCS Vol. 1880, 2000, p. 197.

[5] S. Blake-Wilson and A. Menezes, "Unknown key-share attacks on the station-to-station (STS) protocol", *Proceedings of PKC '99*, Lecture Notes in Computer Science, 1560 (1999), 154-170.

[6] M. Burrows, M. Abadi and R. Needham, "A logic for authentication," *ACM Trans. Computer Systems* Vol. 8 (Feb. 1990), pp. 18-36

[7] Canetti, R., and Krawczyk, H., "Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels", Eurocrypt'2001, LNCS Vol. 2045. Full version in: *Cryptology ePrint Archive* (`http://eprint.iacr.org/`), Report 2001/040.

[8] Canetti, R., and Krawczyk, H., "Security Analysis of IKE's Signature-based Key-Exchange Protocol", *Crypto 2002.* LNCS Vol. 2442. Full version in: *Cryptology ePrint Archive* (`http://eprint.iacr.org/`), Report 2002/120.

[9] L. Carter and M. Wegman, "Universal hash functions", *J. of Computer and System Sciences*, Vol. 18, 1979, pp. 143–154,

[10] T. Dierks and C. Allen, "The TLS Protocol – Version 1", *Request for Comments 2246*, 1999.

[11] W. Diffie, P. van Oorschot and M. Wiener, "Authentication and authenticated key exchanges", *Designs, Codes and Cryptography*, 2, 1992, pp. 107–125. Available at `http://www.scs.carleton.ca/~paulv/papers/sts-final.ps`.

[12] N. Ferguson and B. Schneier, "A Cryptographic Evaluation of IPSec", `http://www.counterpane.com/ipsec.html`, 1999.

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

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

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

[16] C. Jutla, "Encryption Modes with Almost Free Message Integrity", *Advances in Cryptology – EUROCRYPT 2001 Proceedings*, Lecture Notes in Computer Science, Vol. 2045, Springer-Verlag, B. Pfitzmann, ed, 2001.

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

[18] B. Kaliski, "An unknown key-share attack on the MQV key agreement protocol", *ACM Transactions on Information and System Security (TISSEC).* Vol. 4 No. 3, 2001, pp. 275–288.

[19] C. Kaufman, "Internet Key Exchange (IKEv2) Protocol", draft-ietf-ipsec-ikev2-08.txt, May 2003 (to be published as an RFC).

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

[21] S. Kent and R. Atkinson, "IP Encapsulating Security Payload (ESP)", *Request for Comments 2406*, Nov. 1998.

[22] H. Krawczyk, Communication to IPsec WG, *IPsec mailing list archives*, April-October 1995. `http://www.vpnc.org/ietf-ipsec/`

[23] H. Krawczyk, "SKEME: A Versatile Secure Key Exchange Mechanism for Internet,", *Proceedings of the 1996 Internet Society Symposium on Network and Distributed System Security*, Feb. 1996, pp. 114-127. `http://www.ee.technion.ac.il/~hugo/skeme-lncs.ps`

[24] Krawczyk, H., Bellare, M., and Canetti, R., "HMAC: Keyed-Hashing for Message Authentication", *RFC 2104*, February 1997.

[25] Krawczyk, H., "Blinding of Credit Card Numbers in the SET Protocol", *Proceedings of Financial Cryptography'99*, LNCS Vol. 1648, 1999.

[26] H. Krawczyk, "The order of encryption and authentication for protecting communications (Or: how secure is SSL?)", Crypto'2001, LNCS Vol. 2139. Full version in: *Cryptology ePrint Archive* (`http://eprint.iacr.org/`), Report 2001/045.

[27] G. Lowe, "Some New Attacks upon Security Protocols", *9th IEEE Computer Security Foundations Workshop*, IEEE Press 1996, pp.162-169.

[28] Meadows, C., "Analysis of the Internet Key Exchange Protocol Using the NRL Protocol Analyzer", *Proc. of the 1999 IEEE Symposium on Security and Privacy*, IEEE Computer Society Press, 1999.

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

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

[31] Perlman, R., and Kaufman, C., "Analysis of the IPsec key exchange Standard", *WET-ICE Security Conference*, MIT, 2001.

[32] H. Scherzer, R. Canetti, P.A. Karger, H. Krawczyk, T. Rabin, and D.C. Toll, "Authenticating Mandatory Access Controls and Preserving Privacy for a High-Assurance Smart Card", April 2003.

[33] 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`.

[34] P. van Oorschot, "Extending cryptographic logics of belief to key agreement protocols", *Proceedings, 1st ACM Conference on Computer and Communications Security*, Nov. 1993, Fairfax, Virginia, pp. 232-243,