

# Deniable Authentication and Key Exchange\*

Mario Di Raimondo  
Dipartimento di Matematica ed Informatica  
Università di Catania, Italy  
diraimondo@dmi.unict.it

Rosario Gennaro, Hugo Krawczyk  
IBM T.J.Watson Research Center, USA  
rosario@watson.ibm.com,  
hugo@ee.technion.ac.il

## ABSTRACT

We extend the definitional work of Dwork, Naor and Sahai from deniable authentication to deniable key-exchange protocols. We then use these definitions to prove the deniability features of SKEME and SIGMA, two natural and efficient protocols which serve as basis for the Internet Key Exchange (IKE) protocol.

SKEME is an encryption-based protocol for which we prove full deniability based on the plaintext awareness of the underlying encryption scheme. Interestingly SKEME's deniability is possibly the first "natural" application which essentially requires plaintext awareness (until now this notion has been mainly used as a tool for proving chosen-ciphertext security).

SIGMA, on the other hand, uses non-repudiable signatures for authentication and hence cannot be proven to be fully deniable. Yet we are able to prove a weaker, but meaningful, "partial deniability" property: a party may not be able to deny that it was "alive" at some point in time but can fully deny the contents of its communications and the identity of its interlocutors.

We remark that the deniability of SKEME and SIGMA holds in a *concurrent* setting and does not essentially rely on the random oracle model.

## Categories and Subject Descriptors

C.2.2 [Computer-Communication Networks]: Network Protocols – *Applications*; D.4.6 [Operating Systems]: Security and Protection – *Authentication*

## General Terms

Security, Theory

## Keywords

Authentication, deniability, key exchange

\*A full version of this paper appears on the IACR Eprint Archive at <http://eprint.iacr.org/2006/280>

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CCS'06, October 30–November 3, 2006, Alexandria, Virginia, USA.  
Copyright 2006 ACM 1-59593-518-5/06/0010 ...\$5.00.

## 1. INTRODUCTION

Privacy of communications has been the main object of study in cryptography for centuries. Its classical goal: prevent unauthorized parties from accessing secret or confidential information. In this setting the focus is on establishing authenticated and secret communication (a.k.a. "secure channels") with authorized peers. The intent is to defend the communications from an "unauthorized third party" in the form of an eavesdropper or active attacker; there is no attempt at preventing an authorized peer from disclosing information it receives legitimately. Today, with the transfer of our personal, social, economic and political lives to digital form, privacy has become a much wider and central notion. Modern cryptography recognized these issues early on through anonymity-related notions [13, 14], mix networks [12], undeniable signatures [15], private information retrieval [17], and more. More recently, we have seen a huge increase in the treatment of broader privacy issues in the crypto/security community (e.g. [36, 24, 16]). This paper focuses on an essential aspect of privacy: the deniability of every-day digital communications.

Deniable communication has always been a central concern in personal and business communications, with off-the-record communication serving as an essential social and political tool. Given that much of these interactions now happen over digital media (email, instant messaging, web transactions, virtual private networks) it is of central importance to provide these communications with "off-the-record" or deniability capabilities: the author or sender of a message should be able to deny, e.g., in court, that he or she sent that message. (Needless to say, there are special applications where non-repudiation is essential, but this is not the case for most of our communications.) One of the challenges of deniability in the digital world is that deniability is at odds, at least without careful design, with remote authentication. That is, Alice needs to be able to get a digital proof that she is talking to Bob but that proof should not leave any trace that will convince a third party that Bob talked (or said something specific) to Alice. This should be the case even if Alice herself is trying to prove the existence of the conversation to such third party! Thus, while in the traditional "secure channels" setting one is not defending against misbehavior by the (authorized) peer to the communication, in the deniability setting the potential attackers include the authorized (identified and authenticated) peer.

The first to formally treat the deniable authentication problem were Dwork, Naor and Sahai in [25], followed by a series of papers including [39, 32, 21]. On the more ap-

plied front, a practical (and widespread) protocol that set “plausible deniability” as a desirable property (though, not as an essential goal) is the IKE protocol [29]; in particular, this motivated the design of the SKEME protocol [34] that became part of IKEv1. More recently, [7, 22] treat explicitly off-the-record communications in the setting of instant-messaging communications.

Our present paper is motivated by the above works. One missing link in these works is the formal treatment of deniability for key-exchange (KE) protocols. Note that when using symmetric shared keys to authenticate/encrypt information then deniability is easy to achieve at least *as long as the secret key cannot be tied, via third-party verifiable proofs, to the identities of the peers*. However, when the symmetric keys are established via a KE protocol, which in turn uses public key techniques for authentication (as it is common in today’s communications), then the weak link for deniability becomes this KE protocol. If its authentication mechanisms leave a “proof of communication” then deniability is lost.

The following example is useful to illustrate how a KE protocol can indeed leave such a “proof of communication”. Here is a simple 3-message variant of an ISO Diffie-Hellman KE protocol [30, 9]:

- (1)  $A \rightarrow B: g^x$
- (2)  $B \rightarrow A: g^y, sig_B(g^x, g^y, id_A)$
- (3)  $A \rightarrow B: sig_A(g^y, g^x, id_B)$

The protocol makes use of digital signatures as the most natural (and scalable) tool for remote authentication. In addition, as part of the essential features of a secure KE protocol the identities of the communicating parties (i.e.,  $id_A$  and  $id_B$ ) are tied to the exchanged key via these signatures. However, by signing the peer’s identity each of the parties of the protocol leaves an undeniable proof of communication between these parties. Note that in this case the non-deniability of the protocol is particularly serious: not only can  $A$  prove that  $B$  talked to her, but even an eavesdropper that obtains these signatures will be able to provide such a proof. Unfortunately, if one omits these identities in the signatures the protocol becomes completely insecure [26].

This example serves to stress the conflict between authentication and deniability; and, in the case of KE, the conflict between deniability and the binding between identities and the exchanged key so central to KE security. One of our central contributions is in showing that carefully designed protocols may provide for sound KE security as well as for significant levels of deniability. Fortunately, we show this to be the case for some well-known and practical KE protocols.

**DEFINING DENIABILITY.** The notion of *deniability* in public key authentication is formalized by Dwork, Naor and Sahai [25] using the *simulation* paradigm. Informally, a protocol is deniable if the view of any receiver (or verifier) can be simulated by a machine that does not know the secret key of the sender (or prover). The idea behind this natural and appealing definition is that the transcript of the protocol owned by the receiver, cannot be used to trace this conversation back to a specific sender, since the receiver could have produced it via the simulator machine.

Thus, the basic property of a deniable protocol follows the notion of *zero-knowledge* [27]. However, for deniability one needs a stronger simulator than in the case of ZK: while in ZK the simulator is basically a “mental experiment”, for deniability, as pointed out by Pass [39], the simulator must

be a real machine that works in the real world (ruling out some typical ZK simulators in the common reference string (CRS) and random oracle models). Another challenge in dealing with ZK-based proofs is that those usually make use of “rewinding”. As pointed out in [25], this technique significantly limits the applicability of the proof in a real-life concurrent-executions setting.

**DENIABILITY FOR KEY EXCHANGE PROTOCOLS.** In a KE protocol, two parties engage in a protocol whose result is a (session) key  $K$  which only the two of them know, and they are assured to be sharing  $K$  with each other. They will use  $K$  to encrypt and authenticate messages in the session, using a symmetric-key authentication mechanism that is deniable *provided that the key cannot be traced to either party*.

Thus, the most important component for the deniability of electronic communications is the deniability of KE protocols. If the parties can deny having exchanged a key with the other party, then the rest of the communication can also be denied.

**OUR CONTRIBUTIONS.** After recalling the definition of deniable authentication from [25] we propose a definition of deniable key exchange protocols, which still adheres to the simulation paradigm. The extension is not trivial as we are moving to a protocol which outputs a secret value (rather than the simple accept/reject bit of an authentication protocol). In particular KE deniability requires the simulation not only of the entire transcript, but of the output key as well, since the key will be passed to an arbitrary security protocol after the exchange phase is completed.

We then study the deniability of SKEME [34] and SIGMA [35], two practical KE protocols which form the cryptographic core of IKE, the Internet Key Exchange of IPSEC [29, 33].

SKEME uses encryption to authenticate the parties. For SKEME we show the strong form of deniability guaranteed by our definition. The analysis has several interesting features. We first abstract out a basic message authentication mechanism from the key exchange protocol. This authentication scheme was proven unforgeable in [1] if the encryption scheme used inside is CCA2-secure [23]. Our first result is to show that CCA2-security is *not* sufficient for deniability by showing that there is a CCA2-secure encryption scheme for which the above protocol is not deniable. We then show that deniability holds if the encryption scheme is *plaintext-aware* (either in the standard model [4, 20] or in the random oracle model [6, 2]). Interestingly this makes deniability one of the first applications to essentially require plaintext awareness; so far plaintext awareness was mainly used as a tool to prove CCA2 security.

The case of SIGMA is more involved. Since SIGMA uses signature-based authentication, deniability (defined as full simulation) cannot be achieved. However, in spite of its use of signatures (and in contrast to the above ISO example) SIGMA offers some valuable deniability properties. We capture these properties via a modified notion of *partial deniability* for key-exchange protocols, in which a party can deny the identity of the parties he or she exchanges keys with, as well as the content of the subsequent communications protected by those keys. We show the 4-message variant of SIGMA (known as SIGMA-R) to be partially deniable, under a “key awareness” notion, which can be plausibly assumed to hold for “natural” MAC and hashing functions (and formally shown to hold in the random oracle model).

**CONCURRENCY** An important feature is that our proof of deniability for SKEME and SIGMA holds in a *concurrent* setting, where the adversary can open and schedule sessions in an arbitrary way [25]. This is of utmost importance for practical applications run in an open network like the Internet. In addition, our notions and proofs, while meaningful in the random oracle model, do *not* essentially depend on it.

**RELATED WORK.** As we said earlier deniable authentication has been studied from both the theoretical [25, 39, 32, 21] and the practical [34, 29, 8, 38, 7, 22] points of view. For the case of key exchange protocols, where both parties have registered public keys there are other approaches to deniable authentication, such as: *Designated Verifier Proofs* [31] and *Ring Signatures* [40]. Our approach is different in that our goal is to prove deniability for real-life protocols used in practice (which do not use the above tools).

Formalisms of KE security has been extensively studied [1, 41, 9, 11] yet none of those studies considered deniability explicitly and/or formally. Informal treatment of deniability issues for KE protocols can be found in [34, 7, 8, 38, 22].

## 2. DENIABLE KEY EXCHANGE

Our presentation and definition of the notions of deniability in this section follows essentially the approach and definition from Dwork, Naor and Sahai [25]. We first recall their definition of deniable authentication that we use as the basis for formalizing deniability of key exchange protocols. (Throughout the paper we will use the standard polynomial-time complexity notions, such as indistinguishability, negligible probabilities, etc.; in particular, all algorithms and machines are probabilistic polynomial-time.)

**Deniable Authentication.** We assume the author is familiar with the notion of a secure (unforgeable) message authentication protocol from [25]. Intuitively an authentication protocol is deniable if a (possibly dishonest) receiver cannot convince a third party (let’s call it, the *judge*) that a given sender  $S$  authenticated a given message  $m$ . This notion was formalized in [25] using a zero-knowledge formalism. The idea is that an authentication protocol is deniable if the receiver’s *view* of the protocol can be *simulated* by an efficient machine (called the *simulator*) that does *not* know the secret key of the sender  $S$ . In other words, the receiver interacting with the simulator obtains views with the same distribution than when interacting with the real sender  $S$ . Thus, when the receiver (the attacker in this setting) brings such a view to the judge, this view will not be a convincing evidence of the interaction with  $S$  since the same view could have been generated by the receiver alone by running the simulator.

Consider an adversary  $\mathcal{M}$  (for “malicious”) acting as a receiver on input  $\text{pk}$ . The adversary may also have some auxiliary input  $\text{aux}$  taken from a distribution  $AUX$  of such inputs. This auxiliary input models some extra information that the adversary might have gathered in some other form; for example, if  $\mathcal{M}$  has been eavesdropping on correctly executed protocols between other parties and  $S$ ,  $AUX$  will consist of legal transcripts of runs of the authentication protocol.

The adversary  $\mathcal{M}$  starts an arbitrary number of executions of the authentication protocol with  $S$  with public key  $\text{pk}$ , choosing the input messages for these executions. These executions can be arbitrarily scheduled and interleaved by

$\mathcal{M}$ . The adversary’s *view* of this interaction is then defined as the transcript of the full interaction between  $\mathcal{M}$  and  $S$ , together with the internal coin tosses of  $\mathcal{M}$ . We denote this as  $\text{View}_{\mathcal{M}}^S(\text{pk}, \text{aux})$ .

**DEFINITION 1.** [25] *We say that  $(AKG, S, R)$  is concurrently deniable with respect to the class  $AUX$  of auxiliary inputs if for any adversary  $\mathcal{M}$ , acting as the receiver on input  $\text{pk}$  and any auxiliary input  $\text{aux} \in AUX$ , there exists a simulator  $SIM_{\mathcal{M}}^S$  that, running on the same inputs, produces a simulated view which is indistinguishable from the real one.*

We stress that the the simulator has *all* the same inputs as  $\mathcal{M}$ , including its random coins (alternatively assume that the simulator provides these coins to  $\mathcal{M}$ ).

**Remark (Off-line vs. on-line judges.)** In the above definition, the real and simulated views are required to be indistinguishable: i.e. no efficient machine (a *distinguisher*) can tell them apart. In the deniability context (also in the case of KE protocols) the *distinguisher* represents the role of the *judge* which needs to decide if the transcript presented by  $\mathcal{M}$  corresponds to a real execution of the protocol with  $S$  or to a simulated view of such run. This distinguisher is presented with the transcript as well as with the inputs of  $\mathcal{M}$  including the auxiliary input  $\text{aux}$  (which are also the inputs on which  $SIM$  is run). Hence, this formulation corresponds to the situation in which the transcript is generated without the judge intervention, i.e., the judge is invoked *a posteriori* to decide if the message  $m$  was really authenticated by  $S$  or not. One could also contemplate a stronger setting in which the judge is allowed to interact with the adversary *before* the authentication protocol takes place or even *during* the run of the protocol between  $\mathcal{M}$  and  $S$ . In the latter case, there is little one can do to provide deniability with respect to this “on-line” judge since he is a direct witness of the conversation between  $S$  and  $R$ . In the case of interaction between the judge and adversary  $\mathcal{M}$  before the (alleged) run between  $\mathcal{M}$  and  $S$ , but not during the run, the above definition per se is not sufficient to ensure deniability. Yet, some protocols will achieve some form of deniability also in this case. We will not formalize this stronger notion here but when presenting specific protocols we will discuss the extent to which they achieve this stronger notion.

**Deniable Key Exchange.** We now extend the above definition of deniability to the setting of (authenticated) key-exchange (KE) protocols. We first present a simplified definition of a KE protocol (for formal definitions of KE security see [5, 41, 9, 10]). Then we define the deniability property in a concurrent-execution setting.

In a key exchange protocol, two parties, say  $A$  and  $B$ , are associated with public keys  $\text{pk}_A$  and  $\text{pk}_B$  respectively, for which they each own the matching secret key  $\text{sk}_A$  and  $\text{sk}_B$ . We assume that public/secret keys are generated according to a key generation algorithm  $KG$  which is part of the specification of the KE protocol, and these are used in the authentication steps of the KE protocol. The protocol specifies the interaction between  $A$  and  $B$  (one acting as “initiator” and the other as “responder”) and its result is either a (session) key  $K$  or “error”. The basic security requirement in a KE protocol is that if  $A$  outputs a session key  $K$  and associates it to peer  $B$  then the only party that may possibly know  $K$  is  $B$ ; and if  $B$  outputs the same session key then

it associates it to peer A. Note however that this security guarantee is provided only for sessions (i.e., runs of the KE protocol) in which both peers are uncorrupted.

In great *contrast*, the deniability guarantee of a KE session is most relevant when one of the peers is dishonest. The goal is to prevent either A or B from proving to a judge that they exchanged a key with a specific party, and to prevent a proof of what the contents of a communication protected with that key were. Once again we model deniability via simulation along the lines of Definition 1 but with some important differences, arising mainly from the fact that not only is the KE protocol itself that needs deniability but also the communications that use the resultant session key.

Let  $\Sigma$  be a key-exchange protocol defined by a key generation algorithm KG and interactive machines  $\Sigma_I, \Sigma_R$  specifying the roles of the (honest) initiator (the party who sends the first message) and responder, respectively. Both  $\Sigma_I$  and  $\Sigma_R$  run on input their own secret and public keys and, typically, also on input the identity and public key of a specified peer. Each run of the KE protocol by a party is called a session. Upon completion, the protocol outputs either error (e.g., an authentication operation failed) or outputs a session key.

Consider an adversary  $\mathcal{M}$  which runs on input an arbitrary number of public keys  $\vec{pk} = (pk_1, \dots, pk_\ell)$ , randomly chosen according to KG and associated with the honest users in the network.  $\mathcal{M}$  also has an auxiliary input  $aux$  as in Definition 1. The adversary initiates an arbitrary number of executions of  $\Sigma$  with the honest parties, some as an initiator, others as a responder. The executions are concurrent, i.e. scheduled and interleaved arbitrarily by the adversary. The view of  $\mathcal{M}$  consists of its internal coin tosses, the transcript of the entire interaction *and the session keys computed in all the protocols in which  $\mathcal{M}$  participated* (if the session does not complete, the session key is defined as an error value). We denote this view as  $\text{View}_{\mathcal{M}}(\vec{pk}, aux)$ .

The definition below follows the traditional approach of simulation of the adversary’s view. However, it is important to highlight an element in this definition that differentiates it essentially from deniability in the message authentication setting of the previous subsection: the inclusion of the computed secret key to the adversary’s view. Recall that the goal of deniability in a KE protocol is not only to prevent a (adversarial) party  $\mathcal{M}$  from proving that another (honest) party B talked to  $\mathcal{M}$  but also to prevent  $\mathcal{M}$  from proving to a third party the contents of a communication in which B participated. Since KE protocols are typically run in order to agree on a session key which is later used to authenticate further communication, *it is essential that not only the communication during the KE session be simulatable but also the value of the session key should be part of the output of the simulation*. In this case, when an attacker brings to a “judge” evidence against B in the form of a key exchange or subsequent authenticated information, the simulatability of the exchange and the resultant key will make this evidence worthless.

**DEFINITION 2.** *We say that  $(KG, \Sigma_I, \Sigma_R)$  is a concurrently deniable key exchange protocol with respect to the class  $AUX$  of auxiliary inputs if for any adversary  $\mathcal{M}$ , for any input of public keys  $\vec{pk} = (pk_1, \dots, pk_\ell)$  and any auxiliary input  $aux \in AUX$ , there exists a simulator  $SIM_{\mathcal{M}}$  that, running on the same inputs as  $\mathcal{M}$ , produces a simulated view which is indistinguishable from the real view of  $\mathcal{M}$ .*

**Remarks on the definition.** First note that impersonation is not the issue here: when  $\mathcal{M}$  is interacting with B she is not trying to impersonate A but rather, she is trying to obtain a proof that she herself interacted with B and established a key with him. The goal of deniability is to prevent  $\mathcal{M}$  from proving to a third party that this was the case. Thus, when  $\mathcal{M}$  interacts with B we can assume (wlog) that she will do so using a public key  $pk_{\mathcal{M}}$  (which may or may not be associated with  $\mathcal{M}$ ’s identity). Indeed, since our definition guarantees that even when the attacker  $\mathcal{M}$  runs the key generation algorithm to generate a public key (thus knowing the corresponding secret key) she cannot prove that B talked to her, then  $\mathcal{M}$  can certainly not be able to prove that B talked to any *other* party A (in particular, this implies deniability with respect to eavesdroppers). In addition, while  $\mathcal{M}$  may decide to reveal her secret key  $sk_{\mathcal{M}}$  to help in proving that B talked to her, she does not have to do so (actually  $\mathcal{M}$  may use a public key for which she does not know a corresponding private key!).

### 3. DENIABILITY OF SKEME

In this section we prove the deniability of the key exchange protocol SKEME [34] which uses public key encryption as a means of authentication. First we abstract out the authentication protocol which is at the core of SKEME and prove that it is deniable if the encryption scheme used is plaintext-aware according to the definition in [4]. Then we use this result to prove the deniability of the SKEME key exchange protocol<sup>1</sup>. We also show that chosen-ciphertext security is not sufficient to prove deniability. In the following we denote an encryption scheme  $\mathcal{E} = (\text{gen}, \text{enc}, \text{dec})$  which are the key generation, encryption and decryption algorithms, respectively. We assume the reader is familiar with the notions of security against adaptive chosen-ciphertext attack [23] and plaintext-awareness [4], though we informally recall them below.

#### 3.1 Encryption-based Deniable Authentication

We first study the authentication protocol based on public-key encryption which is at the core of SKEME [34] (related protocols are studied in [23, 25]).

Let  $pk$  be the sender’s S public key and  $sk$  the related secret key, for a public key encryption scheme  $\mathcal{E}$ . The authentication protocol  $\lambda_{\mathcal{E}}$  on input a message  $m$  works as follows:

1. The receiver R chooses a random key  $k$  and encrypts it for S as  $c = \text{enc}(pk, k)$ . R sends  $c$  to S.
2. S decrypts  $k' = \text{dec}(sk, c)$  (if the decryption fails, the sender chooses a random key  $k'$ ). S computes  $t = \text{MAC}_{k'}(m)$  and sends  $t$  to R.
3. R accepts if  $t = \text{MAC}_k(m)$ .

The receiver’s belief that S is really authenticating  $m$  comes from the fact that she is the only one able to decrypt  $k$ . Indeed this authentication protocol is proven secure in [1] if  $\mathcal{E}$  is secure against adaptive chosen-ciphertext attack.

The protocol  $\lambda_{\mathcal{E}}$  is perfectly deniable against an honest receiver, since the simulator  $SIM_R$  can easily produce valid

<sup>1</sup>Interestingly, this appears to be the first “essential” application of plaintext-awareness; before, this notion was used mainly as a tool to prove chosen-ciphertext security.

transcripts on his own. What happens against a dishonest receiver  $\mathcal{M}$ ? To formally prove the deniability we need to create a simulator  $SIM_{\mathcal{M}}$  that is able to produce valid transcripts interacting with the malicious receiver  $\mathcal{M}$ . When the simulator receives the encrypted challenge  $c = \text{enc}(\text{pk}, k)$ , how are we going to simulate the reply  $t = \text{MAC}_k(m)$ ? The simulator doesn't know the sender's secret key  $sk$ !

The scheme can be made deniable by adding a challenge-response sub-protocol (in a way similar to the authentication protocol in Dwork *et al.* [25]). Basically, instead of replying with the MAC  $t$ , the sender replies with a commitment to  $t$ . Upon receiving such commitment, the receiver reveals the key  $k$  encrypted in the first message. If this key equals the one that he decrypted, the sender opens the commitment, revealing the MAC  $t$ .

This variant is still unforgeable assuming that  $\mathcal{E}$  is secure against adaptive chosen ciphertext attack. Deniability follows from a standard black-box ZK simulation, which however includes a "rewinding" step for the malicious receiver. This rewinding step limits the applicability of the protocol in a concurrent setting: in this case the proof of security guarantees deniability only if a logarithmic (in the security parameter) number of sessions are concurrently executed [25].

We are interested in the deniability of the original protocol  $\lambda_{\mathcal{E}}$ , which is the authentication core of SKEME. Our first result is to show security against adaptive chosen ciphertext attack is *not* sufficient to make  $\lambda_{\mathcal{E}}$  is not deniable.

**THEOREM 3.** *There exists a encryption scheme  $\mathcal{E}$ , which is secure against adaptive chosen ciphertext attack and such that  $\lambda_{\mathcal{E}}$  is not deniable.*

In contrast we show next that *concurrent deniability* for  $\lambda_{\mathcal{E}}$  can be proven by making a stronger assumption on the underlying encryption scheme, namely (*PA-2*) *plaintext awareness* [4].

Intuitively, we say that  $\mathcal{E}$  is *PA-1 plaintext-aware* if for any adversary  $\mathbf{C}$  that on input a public key  $\text{pk}$  outputs a valid ciphertext  $c$  there is a "companion" machine  $\mathbf{C}^*$  that outputs the matching plaintext. Think of  $\mathbf{C}^*$  as the *alter ego* of  $\mathbf{C}$ : the definition basically implies that if  $\mathbf{C}$  produces a valid ciphertext it must "know" the corresponding plaintext. A strengthening of this notion, *PA-2*, accommodates the fact that an attacker may have access to a set of ciphertexts computed under public key  $\text{pk}$  but not produced by the attacker itself; for example, these ciphertexts could have been generated by other, honest, parties in the system communicating with the owner of this public key. Hence, the definition of *PA-1* is strengthened so that the above adversary  $\mathbf{C}$  is also given as input a list of valid ciphertexts for which it does not know the corresponding plaintexts, (thus modeling the fact that the attacker may copy ciphertexts from the network and replay them). In a *PA-2* scheme, and the companion machine  $\mathbf{C}^*$  is defined to yield the corresponding plaintext only for valid ciphertexts produced by  $\mathbf{C}$  which are not in the above list. The resultant notion is called *PA-2 plaintext awareness*.

When proving the deniability property of protocol  $\lambda_{\mathcal{E}}$ , the *PA-1* notion is therefore too weak to represent the common situation in which multiple copies of the protocol are run concurrently since in this case the attacker does have access to valid ciphertexts created by other parties. In particular, in the case of protocol  $\lambda_{\mathcal{E}}$  the attacker may "replay" such

ciphertexts in a communication with the owner of  $\text{pk}$  without knowing the encrypted plaintext. Our result, therefore, depends on the encryption function being *PA-2*. In this case we will use the auxiliary input  $AUX$  to represent a list of valid transcripts of protocol  $\lambda_{\mathcal{E}}$  (with  $\text{pk}$  as the sender's public key) gathered by the attacker in the network.

We denote by  $TR(\text{pk})$  the set of legal transcripts of the protocol  $\lambda_{\mathcal{E}}$  under public key  $\text{pk}$  (i.e., those transcripts that contain legal ciphertexts under  $\text{pk}$ ) and let the auxiliary input  $aux$ , with respect to which the deniability of  $\lambda_{\mathcal{E}}$  is established, be a list of transcripts sampled from  $TR(\text{pk})$  and for which the adversary  $\mathcal{M}$  has no information on the plaintexts (and randomness) used to generate the ciphertexts included in these transcripts. In other words, we assume these to be transcripts generated by *honest parties* interacting with  $\mathbf{S}$ .

**THEOREM 4.** *If the encryption scheme  $\mathcal{E}$  is PA-2 plaintext-aware and semantically secure, then the protocol  $\lambda_{\mathcal{E}}$  is concurrently deniable with respect to the class of auxiliary inputs  $TR(\text{pk})$ .*

### 3.2 The SKEME Key Exchange Protocol

The SKEME key exchange from [34] is described in Figure 1. It consists of two parallel executions of the authentication protocol  $\lambda_{\mathcal{E}}$  applied to the messages  $(g^x, g^y)$  and  $(g^y, g^x)$  where  $\mathbf{A}$  and  $\mathbf{B}$  act as the sender in one and the receiver in the other. Yet, deniability for SKEME does not immediately follow, from the simulatability of  $\lambda_{\mathcal{E}}$ . Recall that for a key exchange, we must simulate not only the transcript, but also the output key.

**THEOREM 5.** *If  $\mathcal{E}$  is a PA-2 and IND-CPA secure encryption scheme, then SKEME is a concurrently deniable key exchange protocol.*

Notice that Theorem 5 holds for the regular case in which the judge (or distinguisher) is not present during the run of the KE protocol nor it provides inputs to the protocol, but rather is presented with a transcript *a posteriori*. The full version discusses how SKEME can be made deniable in the case in which the adversary cooperates with the judge before the protocol takes place.

### 3.3 On Plaintext-Aware Encryptions

Above we used the notion of plaintext-awareness in the standard model (i.e. without random oracle) from [4]. Dent in [20] shows that the Cramer-Shoup scheme [18] is *PA-2* plaintext-aware under a non-black-box type of assumption known as "Knowledge of exponent assumption" (KEA1) [19, 28, 3], which we recall below.

Let  $G$  be a cyclic group of prime order  $q$ , and  $g, h$  both both generators of  $G$ . We assume that for every algorithm  $\mathcal{M}$  that on input  $G, q, g, h$  outputs  $(y, z)$  where  $y = g^x$  and  $z = h^x$  for some  $x \in Z_q$ , there exists an algorithm  $\mathcal{M}^*$  which outputs  $x$ . In other words, the only way for  $\mathcal{M}$  to output a pair of elements  $y, z$  that have the same discrete log  $x$  with respect to two different basis  $g, h$  is to know  $x$ .

**RANDOM ORACLE MODEL SCHEMES.** Typical instantiations of SKEME, such as in IKE, use encryption schemes like OAEP which are plaintext-aware in the random oracle model. It is not hard to see that our proof of deniability will also hold for such schemes. Indeed the basic tool to simulate the authentication protocol  $\lambda_{\mathcal{E}}$  is a "plaintext extractor" for

correct ciphertexts, which is guaranteed to exist by the definition of plaintext-awareness in the random oracle model.

Note that the “programmability feature” of the random oracle is *not* used in the above argument (only the ability to “see” where the adversary queries the oracle.) Thus our simulator is a valid deniability simulator (a simulator that “programs” the random oracle, does not guarantee deniability [39]).

## 4. PARTIAL DENIABILITY OF SIGMA

The SIGMA key-exchange protocol was proposed in [35] and proven secure in [11]. It forms the cryptographic basis for the Internet Key Exchange (IKE) protocol (specifically, the signature-based mode in IKEv1 [29] and the public key mode in IKEv2 [33]). Its basic goal is to provide a secure Diffie-Hellman exchange authenticated using digital signatures.

While deniability was not an explicit design goal of SIGMA (in IKEv1 deniability was offered via the encryption-based mode based on SKEME), we show that the protocol provides some significant level of deniability even though it does not achieve the full deniability of SKEME.

The 4-round SIGMA protocol is presented in Figure 2: each party signs its own DH exponential as well as the peer’s exponential and computes a MAC on its own identity and a value 0/1 depending on whether the party the initiator or responder; this serves to prevent reflection attacks<sup>2</sup>. We stress that if one adds a differentiator, such as this bit 0/1, to the signatures then the proof of deniability (for the responder) as presented below will not work. Indeed, the proof uses the fact that the signatures produced by the parties do not include evidence of whether the signature was produced in the role of initiator and responder.

Another aspect of SIGMA that will be of importance in our analysis in Section 4.2 relates to the way the MAC key  $K_m$  and session key  $K_s$  are derived from the DH value  $g^{xy}$ . This key derivation function (KDF) typically consists of three components: (i) computing the DH value  $g^{xy}$ , (ii) extracting a key  $K$  from  $g^{xy}$  via a hashing operation (implemented via SHA-1, a universal hashing scheme, etc.), (iii) using a PRF with key  $K$  to derive the two values  $K_m, K_s$  (e.g., the first is set to  $\text{PRF}_K(0)$  and the latter to  $\text{PRF}_K(1)$ ). We do not specify the way these components are implemented but in the analysis we will need to assume certain properties of these functions. Finally, we point out that in some variations of SIGMA, the third and fourth message are encrypted (to provide *identity protection*): the deniability of the protocol is preserved (the proof is just a straightforward adaptation of the proof of Theorem 7).

### 4.1 Partial Deniability

The challenge in creating a deniable key-exchange protocol that uses digital signatures is that the sole fact that a signature was generated, even if on random inputs, may provide significant information (e.g., that the signer was “alive” or active). Yet, even in this case the range of deniability may vary widely: from no deniability in the case of a protocol that signs the peer’s identity (as the ISO protocol

<sup>2</sup>Alternatively, as in the case of IKE, initiator and responder can use different MAC keys (both derived from  $g^{xy}$ ) in generating their MAC values. The deniability of the protocol is preserved with either technique.

discussed in the introduction) to a protocol that only signs self-generated random information. Here we investigate the position of SIGMA in this “deniability range”. We’ll see that the provision of not signing the peer’s identity (but rather MACing one’s own identity), needed to achieve identity protection, also provides the basis for deniability.

Let’s consider first the role of the initiator, or A. The core observation is that A will sign  $g^y$  irrespective of who generated it. For example, consider an attacker  $\mathcal{M}$  that encodes in the exponent  $y$  (used to generate the value  $g^y$  sent to A) its own identity (e.g., choosing  $y$  to be a signature by this attacker). The fact that A will sign  $g^y$  says nothing about whether A talked or not to  $\mathcal{M}$  (or even if she was willing to talk to  $\mathcal{M}$ ) since A will sign  $g^y$  regardless of who sent it. In other words, A’s transcripts are *peer-independent*. The case of the responder B is similar to the above and its transcript is also “peer-independent”<sup>3</sup>. This peer-independence property provides a limited, yet meaningful and significant form of deniability.

To formalize it we resort to a weaker notion of deniability for message authentication introduced by Dwork, Naor and Sahai [25]. This definition from [25] still follows the simulation paradigm but allows the simulator to interact with an oracle representing the real sender (or prover) on random messages rather than the input one. This captures the fact that whatever is learned from the authentication protocol is *independent* of the authenticated message. Or in other words, the attacker can prove that he talked to a specific party, but can’t prove which message the party authenticated.

Roughly speaking, we will say that a key-exchange protocol is *partially deniable for the initiator* if the runs of the (honest) initiator A with a given responder B are indistinguishable from runs with any other responder B’. In the formal definition we provide the initiator’s simulator  $SIM_I$ , simulating an initiator A and acting on input  $(B, \text{pk}_B)$ , with one of two oracles: an oracle that acts as the real initiator A running with peer B’ where  $B' \neq B$ , or an oracle that acts as A running as the responder with peer B’. Similarly, we define partial deniability for the responder. Then we will say that the protocol is *partially deniable* if it is deniable for the initiator and responder. This is formalized in Definition 6 below.

NOTATION. Let  $\Sigma$  be a key-exchange protocol with interactive machines  $\Sigma_I, \Sigma_R$  defining the roles of the (honest) initiator and responder, respectively. We use the symbol  $\Sigma_I^C(D, \text{pk}_D)$  to denote the interactive machine implementing a honest party C as the initiator in a run of protocol  $\Sigma$  with peer D and peer’s public key  $\text{pk}_D$ . Implicit in this notation is that  $\Sigma_I^C$  has as input the secret and public keys of C. The responder machine  $\Sigma_R^C(D, \text{pk}_D)$  is defined analogously.

DEFINITION 6. Let  $\Sigma = (\Sigma_I, \Sigma_R)$  be a key-exchange protocol. We say that  $\Sigma_I$  is partially deniable with respect to an I-oracle (resp. R-oracle) if for any adversary  $\mathcal{M}$  and any (honest) party C, the interaction between C as initiator with  $\mathcal{M}$  as responder can be simulated (as in Definition 2) by a simulator  $SIM_I$  that is given oracle access to  $\Sigma_I^C(D, \text{pk}_D)$  (resp.  $\Sigma_R^C(D, \text{pk}_D)$ ) where  $\text{pk}_D$  is a public key chosen independently of  $\mathcal{M}$ ’s public key  $\text{pk}_M$ .

<sup>3</sup>However, we will see later that the fact that B sends its signature on  $(g^y, g^x)$  after verifying the sender’s identity may provide some extra information under special circumstances.

We say that  $\Sigma_I$  is partially deniable if it is partially deniable with respect to an  $I$ -oracle or with respect to an  $R$ -oracle. The definition of  $\Sigma_R$  being partially deniable is similar. Finally, we say that  $\Sigma = (\Sigma_I, \Sigma_R)$  is partially deniable if both  $\Sigma_I$  and  $\Sigma_R$  are partially deniable.

**CONCURRENCY.** For ease of presentation the above definition is formulated in terms of a single “stand-alone” execution of the protocol. Adding full concurrency to the definition is straightforward (see Definition 2). More importantly, we note that the proof of partial deniability of SIGMA (Theorem 7) avoids rewinding, and thus holds in the concurrent model.

**AUXILIARY INPUT.** The treatment of auxiliary input, omitted in the above simplified definition, is identical to the case of Definition 2; in particular, the proof of Theorem 7 below holds with respect to any auxiliary input. What is important to stress is that the SIGMA protocol has the salient property of being (partially) deniable even if the judge provides DH values to the attacker to be used in the protocol and for which the attacker does not know the exponent (see the discussion on off-line/on-line judges in Section 2). Indeed, when  $\mathcal{M}$  acts as an initiator then receiving  $g^x$  (but not  $x$ ) from a third party does not allow  $\mathcal{M}$  to produce the correct third protocol message and hence the execution is aborted without the responder generating the authentication information in message 4. In the case that  $\mathcal{M}$  acts as responder then a third-party provided  $g^y$  makes no difference since the initiator authenticates  $g^y$  independently of the peer (and without “inspecting” the  $g^y$  value itself).

## 4.2 Deniability Analysis of SIGMA

We now give an informal overview of the proof that SIGMA (as depicted in Figure 2) is partially deniable as in Definition 6. The main difficulties in the simulation of the protocol are (i) the generation of the signatures on behalf of the simulated parties, and (ii) the computation by the simulator of the session key  $K_s$  and the MAC key  $K_m$ . Note that producing  $K_s$  is a necessary condition for satisfying the definition of deniability and partial deniability (Definitions 2 and 6) while computing (or learning)  $K_m$  is a necessary condition to simulate the generation and verification of the MAC values exchanged in the protocol.

For point (i) the simulator will use the real parties as oracles (as allowed by the definition of partial deniability) to produce the signatures. Specifically, the simulation of a given party  $C$  acting as initiator will use an oracle to the real party  $C$  acting as initiator. The simulation of  $C$  as responder will also use an oracle to the real party  $C$  acting as initiator (rather than as a responder – see discussion in Section 4.3).

For point (ii), since each of the exponentials  $g^x, g^y$  are chosen by either the simulator’s oracles or by the adversary then the corresponding exponents  $x, y$  are not known to the simulator who thus cannot easily compute the values  $g^{xy}$ ,  $K_m$  or  $K_s$ . We solve this problem as follows. If the attacker itself cannot compute its own MAC values then the simulator will be able to succeed without computing or verifying these MAC values (in this case, the simulated party would abort before having to compute the MAC values). However, if the attacker can compute the MAC then, intuitively, the simulator (which has non-black-box access to the attacker) can learn these values as well.

However, what we really need is for the simulator to learn not just the MAC values but the keys  $K_m$  and  $K_s$ . In most natural/practical implementations of the key derivation function of SIGMA one will have the property that for the adversary to compute the right MAC values, it has to compute the right MAC key and to do so it has to compute the PRF key from which also  $K_s$  is computed. This, however, is not a necessary condition that follows just from the regular definitions of MAC and PRF: one may be able to construct artificial functions where the attacker succeeds in computing its own correct MAC value without necessarily having to compute the key  $K_m$  (note that the key  $K_m$  does not have to be random as the attacker can influence it via the choice of the DH exponential). Moreover, one can envision a situation where the attacker can prove that it could have not possibly known how to compute the MAC produced by its peer to an exchange in which case the exchange (and possibly the following communication) may not be deniable. Therefore, our proof of SIGMA will apply to KDF construction where the above artificial situation does not arise. We formalize this via the following “key awareness” assumption.

### The Key-Awareness Assumption for SIGMA’s KDF.

We say that a KDF procedure for SIGMA has the Key-Awareness property if for all deniability attackers  $\mathcal{M}$  against the protocol the following holds. If on exchange of DH values  $X, Y$  an attacker  $\mathcal{M}$  computes a correct value  $\text{MAC}_{K_m}(t)$ , for some input  $t$ , where  $K_m$  is the MAC key derived by the KDF from  $X, Y$ , then there is an “extractor machine”  $\mathcal{M}'$  that on the same inputs of  $\mathcal{M}$  outputs the keys  $K_m$  and  $K_s$ .<sup>4</sup>

In other words, there are no “shortcuts” available for the attacker in computing the MAC without going through the key derivation steps and explicitly computing the MAC key  $K_m$  as well as the companion key  $K_s$ . We note that the above very informal definition of Key-Awareness can be formalized in ways similar to other non-black box “extraction assumptions” such as the “knowledge of exponent” and plaintext awareness assumptions discussed in Section 3.

As we said earlier we expect most natural implementations of key derivation in SIGMA to have the above property, in particular when the hashing, PRF and MAC are implemented using HMAC or CBC-MAC using strong hash and block-cipher functions as in the implementation of SIGMA in IKE. Of course, we cannot prove this, but we have to explicitly assume it.

Next we show that this assumption holds when the hash function  $H$  used to derive the PRF key  $K = H(g^{xy})$  is modelled as a random oracle; in this case, the whole key derivation has the key awareness property. Namely, we show that the simulator can extract the correct keys  $K_m$  and  $K_s$  from the run of the attacker, provided the latter computes correct MAC values. (Interestingly, while intuitive the following argument contains some unexpected subtleties.)

Due to the randomness of  $H$ , if the attacker  $\mathcal{M}$  does not compute  $g^{xy}$  then the key  $K = H(g^{xy})$  (and thus  $K_m$ ) is indistinguishable from random for  $\mathcal{M}$ . In this case, the attacker cannot possibly compute the correct value of a MAC under  $K_m$ . Thus, the simulator (which has access to the or-

<sup>4</sup>In the case in which the protocol uses additional keys, such as directional MAC keys, encryption keys, etc, then we assume that the extractor returns all these keys; alternatively, if all these keys are derived from a single PRF key  $K$  then it suffices that extractor returns this key  $K$ .

acle  $H$ ) can check the inputs to  $H$  provided by  $\mathcal{M}$  and hence learn  $K$ , and with it both  $K_m$  and  $K_s$ . There is however a problem in this argument. The simulator will not be able to compute, in general, the value  $g^{xy}$  by itself; so how will it know which of the inputs to  $H$  was the real value  $g^{xy}$ ? (In particular,  $\mathcal{M}$  may use a key  $K$ , or  $K_m$ , derived from an output of  $H$  on a point different than  $g^{xy}$ ). The solution to this problem uses the fact that the simulator  $SIM$  (shown in the proof of Theorem 7) will have an example of a *correct* value of a MAC, computed under the *correct* key  $K_m$ , produced by a real player! Thus, in this case,  $SIM$  proceeds as follows. It considers each output of  $H$  as a candidate PRF key  $K^*$  and derives from it two candidate keys  $K_s^*$  and  $K_m^*$ . Now,  $SIM$ , uses the candidate key  $K_m^*$  to verify the MAC value received from the real player. If the verification succeeds then  $SIM$  assumes  $K_m^*$  to be the correct MAC key  $K_m$  (and  $K_s^*$  to be the correct session key). It is easy to see that a wrong candidate key  $K_m^*$  will succeed in verifying the real MAC with negligible probability. Indeed, since the distribution of wrong  $K_m^*$  keys is uniform (and independent from  $K_m$ ) then the latter probability is at most as the probability to forge a MAC and hence negligible. (Clearly, if two independent random keys have a high probability of producing, or verifying, the same MAC value then one can forge MAC values computed under a secret random key by simply re-computing the MAC value under another random and independent key.)

Another subtlety in the above argument is what is meant by the “right value” of  $g^{xy}$ . What happens if the attacker chooses, say, a value  $Y$  (instead of  $g^y$ ) not in the subgroup generated by  $g$  (as in the Lim-Lee attacks [37])? In this case the value  $g^{xy}$  is not even well-defined. However, since in this case we will have that the value  $X = g^x$  was chosen by a real (and honest) player then the value  $Y^x$  is well defined and it is this value that we actually refer to as the “right  $g^{xy}$ ” (conversely, when  $X$  is the value chosen by the attacker and  $Y = g^y$  is chosen by the honest player then the “real value of  $g^{xy}$ ” refers to  $X^y$ ).

NOTE. We note that the above simulation requires no rewinding and hence it does not break the concurrency of the deniability property. Also worth noting is that as pointed out by Pass [39] one has to be careful when arguing deniability using the random oracle model. Specifically, one cannot use in such an argument the so called “programmability feature” of random oracles. We note that this property is *not* used in the above argument. (We only use the ability to “see” where the adversary queries the oracle.)

Summarizing, we can prove the following Theorem (details of the proof are in the full version):

**THEOREM 7.** *The SIGMA protocol from Figure 2 with a key-aware key derivation is partially deniable according to Definition 6. (In particular, this is the case if one models the hashing of  $g^{xy}$  in SIGMA as a random oracle.)*

### 4.3 Discussion on Partial Deniability

It is important to understand the “real-life” semantics of our notion of partial deniability. The idea behind our definition is that the transcript available to the adversary  $\mathcal{M}$  could have been produced by a party  $C$  when interacting with any other party  $D$  (this is what we refer to as the “peer independence” property). Thus  $C$  can deny having been involved with  $\mathcal{M}$ . A point to notice is the role of  $C$

during this “claimed” interaction with  $D$ . The simpler case is when  $C$  acts as initiator: in this case the information generated by  $C$  is totally independent of the peer’s identity and hence interaction with any specific peer can be denied by the initiator. In the case that  $C$  runs as responder, the “peer independence” property can be formally proven provided that  $C$  also runs as initiator in *other* instances of the protocol. In other words, the authentication information created by  $C$  as responder is simulatable from the information created by  $C$  as initiator. How common is it in real-life protocols that parties run as both initiators and responders? Clearly, this depends on the application. For example, in applications of SIGMA to Instant Messaging (as in [7, 22]) the common case is that end-points to the IM service act as both responders and initiators. In contrast, in a typical client-server configuration of IPSec, parties will run as either initiators or responders but not both (fortunately, in these cases clients usually run as initiators so they are fully protected by deniability). However, as we see next, SIGMA provides some level of deniability also in cases where parties act exclusively as responders and not as initiators.

**DENIABILITY FOR PARTIES WHO ONLY ACT AS RESPONDERS.** For a party,  $B$ , that only acts as a responder (never as an initiator), SIGMA still provides some form of deniability whose significance may depend on the application. Indeed, note that there is nothing explicit in the authentication information sent by the responder that ties it to a specific peer. The problem, however, is that  $B$  will send this information only after verifying the identity of the peer. Let’s consider an example. Say, Charlie, acting as initiator, sends Bob  $g^x$  where  $x$  encodes Charlie’s signature on some value. Bob signs this  $g^x$  and later Charlie brings to court Bob’s signature and the value  $x$  showing that this value was generated by him (Charlie). In itself this proves nothing about Bob having *knowingly* communicated with Charlie:  $g^x$  could have been sent by David, a party with whom Bob was willing to talk. In other words, David could have been collaborating with Charlie in “framing” Bob (or maybe Charlie just broke into David’s computer). In this sense, Bob enjoys deniability even though it acts as responder-only in SIGMA. On the other hand, one can imagine cases where additional “circumstantial evidence” may make it harder for Bob to deny interacting with Charlie, especially since Bob needs to convince that Charlie was using someone with whom Bob was willing to communicate for mounting the attack.

The above considerations apply also to the case of an implementation of SIGMA that adds information under the signatures that make the signatures produced by  $C$  as initiator distinguishable from those generated by  $C$  as responder (the addition of such information is not needed for the security of SIGMA and is certainly not recommended in any setting where deniability is significant).

**DENIABILITY OF SIGMA-I.** We remark that the 3-message variant of SIGMA, called SIGMA-I [35], is partially deniable, according to Definition 6, *only for the responder*. In this case it is the responder’s behavior which is peer-independent since his signature is produced before seeing the identity of the initiator. The initiator, Alice, in SIGMA-I is in a position similar to the responder in the 4-rounds version of SIGMA (Fig. 2, also referred SIGMA-R) since she signs after seeing the identity of the other peer. Unlike the SIGMA-R case, however, the initiator Alice may not be able to claim that her signature on  $(g^x, g^y)$  was performed in her role of

responder. Indeed, a malicious responder could choose his DH value dependent on Alice's DH value, and this is never the case when Alice acts as responder. In other words, the signatures of initiators and responders can be made distinguishable by a dishonest peer in SIGMA-I, something that is not possible in SIGMA-R. Thus SIGMA-R may be somewhat preferable in the deniability setting.

FINAL NOTE. The deniability limitations of SIGMA follow from the use of signatures (essential to the protocol) and the fact that these signatures are applied to a peer-provided value. The latter issue can be avoided if one replaces the peer's DH value under the signature with a freshness value not chosen by the peer, such as a non-repeating counter or a timestamp; however, these values are seldom available, or secure enough, in practical settings. As said earlier, a more essential limitation of partial deniability is that having a signature of a party, even on random information, is sufficient proof to show that the party was "alive". Moreover, it is easy to see that in the case of SIGMA an attacker can encode into its DH value, information that will allow to prove not only that a party was alive but that it was alive after certain time or event (for example, the attacker can encode into the DH value the hashing of today's New York Times). Leaving a proof of such information seems unavoidable in any protocol that needs to include some "freshness guarantee" inside a signature to prevent its replay.

## 5. REFERENCES

- [1] M. Bellare, R. Canetti and H. Krawczyk. *A Modular Approach to the Design and Analysis of Authentication and Key Exchange Protocols*. STOC '98, 419–428, ACM 1998.
- [2] M. Bellare, A. Desai, D. Pointcheval and P. Rogaway. *Relations among Notions of Security for Public-Key Encryption Schemes*. CRYPTO '98, LNCS 1462, 26–45, Springer 1998.
- [3] M. Bellare and A. Palacio. *The Knowledge of Exponent Assumptions and 3-Round Zero-Knowledge Protocols*. CRYPTO '04, LNCS 3152, 273–289, Springer 2004.
- [4] M. Bellare and A. Palacio. *Towards Plaintext-Aware Public-Key Encryption without Random Oracles*. ASIACRYPT '04, LNCS 3329, 48–62, Springer 2004.
- [5] M. Bellare and P. Rogaway. *Entity authentication and key distribution*. CRYPTO '93, LNCS 773, 232–249, Springer 1994.
- [6] M. Bellare and P. Rogaway. *Optimal Asymmetric Encryption*. EUROCRYPT '94, LNCS 950, 92–111, Springer 1994.
- [7] N. Borisov, I. Goldberg and E. Brewer. *Off-the-Record Communication, or, Why Not To Use PGP*. ACM WPES'04, 77–84, ACM 2004.
- [8] C. Boyd, W. Mao and K. Paterson. *Key Agreement using Statically Keyed Authenticators*. ACNS 2004, LNCS 3089, 248–262, Springer 2004.
- [9] R. Canetti and H. Krawczyk. *Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels*. EUROCRYPT '01, LNCS 2045, 453–474, Springer 2001.
- [10] R. Canetti and H. Krawczyk. *Universally Composable Notions of Key Exchange and Secure Channels*. EUROCRYPT '02, LNCS 2332, 337–351, Springer 2002. Full version at [eprint.iacr.org/2002/059](http://eprint.iacr.org/2002/059).
- [11] R. Canetti and H. Krawczyk. *Security Analysis of IKE's Signature-based Key-Exchange Protocol*. CRYPTO '02, LNCS 2442, 143–161, Springer 2002.
- [12] D. Chaum. *Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms*. *Communications of the ACM*, 24(2), February 1981.
- [13] D. Chaum. *Blind Signatures for Untraceable Payments*. CRYPTO '82, 199–203, Plenum 1982.
- [14] D. Chaum. *Security Without Identification: Transaction Systems to Make Big Brother Obsolete*. *Communications of the ACM*, 28(10):1030–1044, October 1985.
- [15] D. Chaum and H. van Antwerpen. *Uneniable Signatures*. CRYPTO '89, LNCS 435, 212–226, Springer 1990.
- [16] S. Chawla, C. Dwork, F. McSherry, A. Smith and H. Wee. *Toward Privacy in Public Databases*. TCC'05, LNCS 3378, 363–385, Springer 2005.
- [17] B. Chor, O. Goldreich, E. Kushilevitz and M. Sudan. *Private Information Retrieval*. FOCS'95, 41–50, IEEE 1995.
- [18] R. Cramer and V. Shoup. *A Practical Public-Key Cryptosystem Secure Against Adaptive Chosen Ciphertexts Attacks*. CRYPTO '98, LNCS 1462, 13–25, Springer 1998.
- [19] I. Damgard. *Towards Practical Public Key Systems Secure Against Chosen Ciphertext Attacks*. CRYPTO '91, LNCS 576, 445–456, Springer 1992.
- [20] A. Dent. *Cramer-Shoup is Plaintext-Aware in the Standard Model*. EUROCRYPT '06, LNCS 4004, 289–307, Springer 2006.
- [21] M. Di Raimondo and R. Gennaro. *New Approaches for Deniable Authentication*. ACM CCS '05, 112–121, ACM 2005.
- [22] M. Di Raimondo, R. Gennaro and H. Krawczyk. *Secure Off-the-Record Messaging*. ACM WPES'05, 81–89, ACM Press, 2005.
- [23] D. Dolev, C. Dwork and M. Naor. *Non-Malleable Cryptography*. SIAM J. Comp., 30(2):391–437, April 2000.
- [24] C. Dwork and K. Nissim. *Privacy-Preserving Datamining on Vertically Partitioned Databases*. CRYPTO '04, LNCS 3152, 528–544, Springer 2004.
- [25] C. Dwork, M. Naor and A. Sahai. *Concurrent Zero-Knowledge*. J. ACM 51(6): 851–898 (2004).
- [26] W. Diffie, P. Van Oorschot and M. Wiener. *Authentication and Authenticate Key Exchange*. *Designs, Codes and Cryptography*, no. 2, 107–125, 1992.
- [27] S. Goldwasser, S. Micali, and C. Rackoff. *The Knowledge Complexity of Interactive Proof-systems*. SIAM J. Comp., 18(1):186–208, February 1989.
- [28] S. Hada and T. Tanaka. *On the Existence of 3-Round Zero-Knowledge Protocols*. CRYPTO '98, LNCS 1462, 408–423, Springer 1998.
- [29] D. Harkins and D. Carrel, eds. *The Internet Key Exchange (IKE)*. RFC 2409, November 1998.
- [30] ISO/IEC IS 9798-3, "Entity authentication mechanisms — Part 3: Entity authentication using asymmetric techniques", 1993.
- [31] M. Jakobsson, K. Sako and R. Impagliazzo. *Designated Verifier Proofs and Their Applications*. EUROCRYPT '96, LNCS 1070, 143–154, Springer 1996.
- [32] J. Katz, *Efficient and Non-Malleable Proofs of Plaintext Knowledge and Applications*. EUROCRYPT '03, LNCS 2656, 211–228, Springer 2003.
- [33] C. Kaufman, ed., Internet Key Exchange (IKEv2) Protocol, draft-ietf-ipsec-ikev2-17.txt, September 2004 (pending RFC).
- [34] H. Krawczyk. *SKEME: a versatile secure key exchange mechanism for Internet*. IEEE SNDSS '96, 114–127, IEEE Press 1996.
- [35] H. Krawczyk. *SIGMA: The 'SiGn-and-Mac' Approach to Authenticated Diffie-Hellman and Its Use in the IKE Protocols*. CRYPTO '03, LNCS 2729, 400–425, Springer 2003. Available at [www.research.ibm.com/security/sigma.ps](http://www.research.ibm.com/security/sigma.ps)
- [36] Y. Lindell and B. Pinkas. *Privacy Preserving Data Mining*. J. of Cryptology, 15(3):177–206, Springer 2002.
- [37] C.H. Lim and P.J. Lee. *A Key Recovery Attack on Discrete Log-based Schemes Using a Prime Order Subgroup*. CRYPTO '97, LNCS 1294, 249–263, Springer 1997.
- [38] W. Mao and K.G. Paterson. *On the Plausible Deniability Feature of Internet Protocols*. Manuscript.
- [39] R. Pass. *On Deniability in the Common Reference String and Random Oracle Model*. CRYPTO '03, LNCS 2729, 316–337, Springer 2003.
- [40] R. Rivest, A. Shamir and Y. Tauman. *How to Leak a Secret*. ASIACRYPT '01, LNCS 2248, 552–565, Springer 2001.
- [41] V. Shoup. *On Formal Models for Secure Key Exchange*. IBM Research Report RZ 3120, April 1999.

