Deniable Internet Key Exchange*

Andrew C. Yao † Frances F. Yao‡ Yunlei Zhao§ Bin Zhu¶

Abstract

In this work, we develop a family of protocols for deniable Internet Key-Exchange (IKE) with the following properties:

- Highly practical efficiency, and conceptual simplicity and clarity.
- Forward and concurrent (non-malleable) deniability against adversaries with arbitrary auxiliary inputs.
- Provable security in the Canetti-Krawczyk post-specified-peer model, and maintenance of essential security properties not captured by the Canetti-Krawczyk security model.
- Compatibility with the widely deployed and standardized SIGMA (i.e., the basis of IKEv2) and (H)MQV protocols, when parties possess DL public-keys.

In view of the wide deployment and use of IKE and increasing awareness of privacy protection (especially for E-commerce over Internet), this work is naturally of practical interest.

1 Introduction

Key-exchange (KE) is a traditional area of cryptography. The Internet Key-Exchange (IKE) protocols [14, 16], pioneered by Photuris [15], SKEME [18], Oakley [24] and SIGMA [19], specify key exchange mechanisms used to establish shared keys for use in the Internet Protocol Security (IPsec) standards [17]. The IPsec and IKE are intended to protect messages communicated *in the IP layer*, i.e., "layer 3" of ISO-OSI, that processes the transmission of messages *using the network addresses possibly without knowing end-user peers' identities*. IKE and IPsec can in turn be used to offer confidentiality, authentication and privacy for communication protocols in the higher layers of ISO-OSI (note that many communication protocols including many authentication protocols which are invoked by *end-users with explicit peer's identity information* work at "layer 7" of ISO-OSI, i.e., the application layer).

The standard of IKE has gone through two generations (versions). The first generation IKEv1 [14] uses public-key encryption as the authentication mechanism (with SKEME serving as the basis in part), and IKEv2 [16] uses public-key signature as the authentication mechanism (with SIGMA serving as the basis in part).

Several salient features of SIGMA helped make it adopted as the basis of IKEv2: (1) it works in the post-specified-peer ID model; (2) it provides identity concealment and plausible deniability; etc.

Desirability for IKE with post-specified-peer ID [7]. Note that IPsec and IKE are intended to protect communication protocols in the IP layer that processes the transmission of messages *using network addresses (rather than end-uses' IDs)*. This brings forth an issue: the peer's identity may be not available at the onset of the IKE, and may be learned by the party only after the protocol run evolves (*even learning from the last message in a run of the protocol*). More specifically, in IKE a party may be activated to exchange a key with an "address" of a peer without knowing the identity of the peer. This

*The work described in this paper was supported in part by the National Basic Research Program of China Grant (973) 2007CB807900, 2007CB807901.

†Institute for Theoretical Computer Science, Tsinghua University, Beijing, China. andrewcyao@tsinghua.edu.cn

‡Department of Computer Science, City University of Hong Kong. csfyao@cityu.edu.hk

§Corresponding author. Software School, Fudan University, Shanghai 200433, China. ylzhao@fudan.edu.cn

¶Microsoft Research Asia, Beijing, China. binzhu@microsoft.com

is quite common in practical applications, in particular for IKE. For example, a key-exchange session may take place with any one of a set of servers sitting behind a (url/ip) address specified in the session activation. Furthermore, even for protocols running in the higher layers (e.g, the application layer), a party may respond to a key-exchange request coming from a peer who, for the sake of preserving its privacy, is not willing to reveal its identity over the network and, sometimes, not even to the responder before the latter has authenticated itself (e.g., a roaming mobile user connecting from a temporary address, or a smart-card that authenticates the legitimacy of the card-reader before disclosing its own identity). Such a general setting for IKE is referred to as "post-specified peer" model [7].

Desirability for deniable IKE [11]. Deniable communication is an essential privacy property, and 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 many of these interactions now happen over digital media (email, instant messaging, web transactions, virtual private networks), it is of critical importance to provide these communications with "off-the-record" or deniability capability: the sender of a message should be able to deny, e.g., in court, that he or she has sent that message. In general, we can just run a deniable authentication protocol for each message to be sent when deniability of the message is desired. However, the beauty of using deniable key-exchange is that if the key-exchange protocol is deniable, then all transactions using the session-key produced by the key-exchange protocol will be automatically deniable. In addition, offering deniability at the IP layer is desirable because it enables various privacy services to be offered at the higher layers with uncompromised quality. Note that a privacy problem at the IP layer can cause irreparable privacy damage at the application layer. For example, an identity connected to an IP address, if not deniable, certainly nullifies an anonymous property offered by a fancy cryptographic protocol running at the application level. A *variant* of the SIGMA protocol of [19] (i.e., the underlying basis of public-key IKEv2) has been proved to satisfy some plausible partial deniability assuming key-aware derivation (typically, with random oracle assumption of hash functions) [11].

On the desirability of forward and concurrent deniability. In the traditional definition of deniable authentication, sender Alice wants, in protecting her privacy, to prevent (possibly malicious) receiver Bob from proving to a third party that he has received a message from Alice by requiring that the protocol is actually a (computational) zero-knowledge (ZK) protocol in the sense that Bob's view can be simulated. Furthermore, we hope the ZK property holds against malicious Bob interacting with Alice *concurrently* in many sessions, i.e., the concurrent ZK (CZK) property [12].

However, as clarified in [10], there are scenarios in which deniability is actually a concern to the receiver's privacy. Consider the following example presented in [10]. Alice and Bob are involved in some shady transaction, like drug-dealing or money laundering. Alice wants to make sure that her communications to Bob cannot be later linked to her, so she uses deniable authentication. Bob thinks that such communications are indeed deniable, stores all the messages in his hard disk. Later the operation is busted by the police and Alice and Bob end up in jail, and Bob's computer is seized. Alice is offered a sweet deal in exchange for her cooperation in linking Bob to the crime (Bob is claiming that the messages in his hard disk are not coming from Alice, that he never talked to her, actually does not even know her, all messages are just simulations!). Alice produces some piece of secret information (her secret-key for example) that indeed shows that the transcripts in Bob's hard disk are actually authentic and not simulated. Bob ends up in jail, cursing himself for dropping out of crypto class in graduate school.

The above example shows that deniability is not just a concern for the sender, but also for the receiver. What we would like to happen is that if the sender acts honestly during the protocols, she should not be able at a later stage to claim the messages are authentic. This property is called *forward deniability*, as it has some affinity to the notion of forward secrecy. In particular, Di Raimondo and Gennaro have shown that computational ZK does not guarantee forward deniability [10], but statistical/perfect (concurrent) ZK guarantees forward (concurrent) deniability.

Another typical example where forward deniability is important is for electronic elections [10]. There, while it is important that both parties, the voting authority and the voter, authenticate each other (for the authority to know that the voter has the right to vote, for the voter to know that his or her vote

will be counted), it is also mandatory to prevent either party from walking away with a non-repudiable proof of what the actual vote was (the message being authenticated). This application, in particular, shows the importance of forward deniability: if the voter (sender) is authenticating his or her vote to the authority, not only the latter should not be able to prove to a third party how the voter voted, but even more importantly, the voter should not be able to do so either at a later stage, to prevent coercion and vote-selling.

1.1 Our contributions

In this work, we present a family of protocols of deniable IKE (in accordance with the main model and the aggressive model of IKE in IPsec) with the following properties:

- Highly practical efficiency, and conceptual simplicity and clarity.
- Forward and concurrent (non-malleable) deniability against adversaries with arbitrary auxiliary inputs.
- Provable security in the Canetti-Krawczyk post-specified-peer model [7], and maintenance of essential security properties not captured by the Canetti-Krawczyk security model.
- Compatibility with the widely deployed and standardized SIGMA [19] (i.e., the basis of IKEv2) and (H)MQV [21, 20] protocols, when parties possess DL public-keys.

The provable security of our protocols relies on the random oracle (RO) assumption and a (very reasonable and essentially weaker) variant of *knowledge-of-exponent assumption*(KEA), called *fresh-challenge KEA with arbitrary auxiliary input*. We note here that the RO and/or KEA assumptions are common in the stringent situation of deniable key-exchange and authentication [4, 3, 10, 20, 9, 11, 26], and actually most known KEA-based deniable authentication and key-exchange protocols are not secure against adversaries with arbitrary auxiliary input. More details can be found in Section 3.

Of independent interest of this work are the clarifications and revision of KEA assumption with arbitrary auxiliary input, and clarifications and observations on the independent Stinson-Wu key-exchange (SW-KE) protocol of [26].

2 Preliminaries

Important note on presentation. *Although our deniable IKE protocols and their security properties can be rigorously formalized and analyzed (that can be found in the full version), we have chosen to present this work here in a somewhat informal and intuitive way to highlight basic ideas of our approach.*

Secure key-exchange in the post-specified-peer model [7]. Let $\langle A, B\rangle$ be a key-exchange protocol. Each activation of the protocol at a party results in a local procedure, called a *session*. Sessions can run concurrently and each session is identified by the party's ID and a (unique) session identifier, e.g., the KE session (A, sid) run by A with session identifier sid. The output of the activation of the KE session (A, sid) is $((A, sid, B), sk)$, where (A, sid, B) is the public output, and sk is the secret session-key (sessions can be aborted without producing the secret session-key sk, and in this case a special symbol is output). A session maintains a local state, e.g., the exponent x of a Diffie-Hellman (DH) component g^x, that is erased when the session completes (i.e., when the party gets the session-key). Each party may have additional state, such as a long-term signature key, which is accessed by different sessions and which is not part of any particular session state. If the session (A, sid) is a completed session with output $((A, sid, B), sk)$, then the session (B, sid) run at the side of B is called the *matching session* of (A, sid) if either (B, sid) is not completed or (B, sid) is finished with (B, sid, A) as its public output.

The attacker $\mathscr{A}$ is a probabilistic polynomial-time (PPT) algorithm that controls the *unauthenticated* communication channels and can schedule session activations and message transmissions at will and sees the outputs of sessions except for the values of session keys. In addition, the attacker can have access to secret information via *session exposure* attacks of three types: session-state reveal, session-key queries,

and party corruption. Session-state reveal is directed at an incomplete session and its result is that the attacker learns the session state, e.g., x of g^x, for that particular session. A session-key query can be performed against a completed session and the result is that the attacker learns the corresponding session-key. Finally, party corruption means that the attacker learns *all* information in the memory of that corrupted party; in addition, from the moment a party is corrupted all its actions are totally controlled by the attacker. Sessions can expire, and from the time a session has expired the attacker is not allowed to perform a session-key query or a state-reveal attack against the expired session, but is still allowed to corrupt the party that holds the session. (Protocols, ensuring that expired sessions are protected even in case of party corruption, are said to enjoy *perfect forward secrecy*.)

A KE protocol is secure, called *satisfying SK-security*, if the following two conditions hold: (1) If two uncorrupted parties complete matching sessions in a run of the key-exchange protocol, under the attacker $\mathscr{A}$, the session key output in the matching sessions is the same; (2) For any session, the attacker cannot distinguish session-key output of the session from an independent random value, provided the session and its matching sessions are *not exposed*.

The SK-security guarantees some essential security properties of key-exchange protocols, in the following sense: a party that completes a session has the guarantee that: (1) if the peer to the session is uncorrupted then the session-key is unknown to anyone except this peer; (2) if the peer completes a matching session then the two parties have the same shared key. These properties are guaranteed even if the attacker has corrupted other parties in the system and/or the attacker has learned the secret information from other sessions. Furthermore, as shown in [6, 7], the SK-security ensures secure communication channels via the shared session-keys.

Forward and concurrent deniable key-exchange. A key-exchange protocol $\langle A, B\rangle$ is called concurrently forward deniable for A, if for any PPT malicious $\mathscr{A}$ running *concurrently* with instances of A in any polynomially many sessions (in any name in each session), there exists a PPT simulator S that, on public system parameters and public-keys, outputs a simulated transcript that is identical to (or statistically indistinguishable from) the real view of $\mathscr{A}$ in its real concurrent executions (i.e., concurrent perfect/statistical zero-knowledge). *We stress that the real view of $\mathscr{A}$ consists of all secret session-key outputs of all sessions, which means that the deniability of key-exchange implies deniability of subsequent communications via the shared session-keys.* Also, the public-keys used by $\mathscr{A}$ in various names/IDs are not guaranteed to be well-generated (note that the names/identities claimed by $\mathscr{A}$ in different sessions are not necessarily the same), though the association of player's ID and its public-key is certified by trusted CA. That is, $\mathscr{A}$ may not know corresponding secret-keys of the various public-keys used by it.

The key-exchange protocol $\langle A, B\rangle$ is called forward and concurrently deniable, if the concurrent forward deniability property holds for both A and B *simultaneously*. That is, for any PPT man-in-the-middle (MIM) $\mathscr{A}$ concurrently interacting with both instances of A and instances of B, its view can be perfectly simulated. Note that, for simplicity of presentation, w.l.o.g., we assume that the MIM adversary $\mathscr{A}$ interacts concurrently with instances of only two parties with names/IDs: A and B. The formulation can be trivially extended to the general case of many parties.

Forward and concurrent deniable authentication. A forward and concurrent deniable authentication protocol $\langle P, V\rangle$ is an *interactive* protocol between a prover P with public-key PK_P and a verifier V (V is not necessarily of public-key), satisfying the following: (1) It is concurrent perfect zero-knowledge for P (against possibly malicious V^*); (2) It is unforgeable against concurrent adaptive chosen-message attack. That is, a PPT adversary $\mathscr{A}$, who can run concurrently with polynomially many instances of P on inputs it chooses adaptively, still cannot authenticate to an honest V a new message that P did not ever authenticate in the concurrent executions. In a way, forward and concurrent deniable authentication could be viewed as an *interactive* public-key signature system with the additional property of *concurrent perfect zero-knowledge*.

3 Deniable Internet Key-Exchange: the Main Model

In this section, we present the deniable IKE protocol of the main model, revisit and clarify the subtleties of KEA assumptions with concurrent adversaries of arbitrary auxiliary input, clarify the essence of ideas

and analyze security properties.

3.1 The protocol construction

The deniable IKE protocol of the main model is depicted in Figure 1. In the protocol specification, A and B stand for player IDs, x and y are randomly and independently chosen from Z_q. For simplicity of presentation, we do not specify the verification procedure details at each step of the protocol, which can be understood in a natural way (in particular, the non-one and order q of g^x, g^y, g^a and g^b will be checked).

Public system parameters: (p, q, g, H), where p and q are large prime numbers such that $p = 2q + 1$, g is an element of Z_p^* of order q, and H is a hash function.
Player keys: (g^a, a) is the public-key and secret-key of A, and (g^b, b) is the key pair of B, where a and b are randomly chosen from Z_q. The association of player's ID and its public-key is certificated by $Cert$ issued by a trusted CA. We stress that for a malicious player, its public-key can be any element of Z_p^* of order q. That is, *no proof-of-possession (of secret-keys) is required for key-registration and certification.*

Actual implementation:	**Natural language interpretation:**
$A \xrightarrow{\quad sid, X = g^x \quad} B$	$A \xrightarrow{\quad \text{Hi, I wanna have a session with } g^x \quad} B$
$A \xleftarrow{\quad sid, B, Y = g^y, NMZK_{(B,y)} = MAC_{H(sid,g^y,g^x,g^{xy})}(B) \quad} B$	$A \xleftarrow{\quad \text{Okey, } g^y \text{ "is sent by } B\text{" who knows } y \quad} B$
$A \xrightarrow{\quad sid, (A, Cert_A), NMZK_{(a,x)} = MAC_{H(sid,g^x,g^y,Y^a,Y^x)}(A) \quad} B$	$A \xrightarrow{\quad \text{Well, I TRULY know } (a, x) \quad} B$
$A \xleftarrow{\quad sid, Cert_B, NMZK_{(b,y)} = MAC_{H(sid,g^y,g^x,X^b,X^y)}(B) \quad} B$	$A \xleftarrow{\quad \text{I TRULY know } (b, y) \text{ as well} \quad} B$

The session-key output is computed by $KDF(H(g^{xy}), \cdot)$, where KDF is a key-derivation function, e.g., $PRF_{H(g^{xy})}(0)$. In practice, MAC and PRF will be implemented with HMAC of [1].

Figure 1: Deniable IKE (the main model)

3.2 KEA assumption, revisited

The Knowledge-of-Exponent (KEA) assumption. The security analysis of the deniable IKE (depicted in Figure 1) relies on the KEA assumption in the random oracle model. The KEA assumption roughly says that: given a challenge $C = g^c$ for randomly chosen $c \in Z_q$, if a *uniform* PPT algorithm $\mathscr{A}$ can output *valid* (X, Y) of order q each such that $Y = X^c$, then $\mathscr{A}$ must *know* $x = log_g^X$ and x can be efficiently extracted from the *internal states* of $\mathscr{A}$ (i.e., *its codes, random coins and the transcript up to the point outputting* (X, Y)). Note that the KEA assumption is a *non-black-box* assumption by nature [3]. By "uniform" we mean that $\mathscr{A}$ takes no auxiliary input. The KEA assumption was introduced in [8], and has been used in many subsequent works (e.g., [4, 3, 10, 20, 9, 11], etc). In particular, the KEA assumption, as well as the random oracle assumption, plays a critical role for provable knowledge-awareness (PA) of public-key encryption systems, e.g., Cramer-Shoup and OAEP [4, 20, 5], and for provable deniability of authentication and key-exchange [10, 20, 11].

Subtleties of KEA assumption for protocols running concurrently in the public-key model. As the KEA assumption is not a standard hardness assumption (thought it seems to be very reasonable), we should take much care of the employment of the KEA assumption in practice, especially for IKE. When using KEA in the public-key model for provable security, e.g., for provable plain-text

awareness and deniability [4, 10, 20, 9, 11], the challenge C is actually the player's *public-key*. It would be fine w.r.t. a uniform adversary $\mathscr{A}$ without auxiliary input (even with some well-predefined auxiliary input as is done in [20, 9, 11]). But, when $\mathscr{A}$ takes an *arbitrary* auxiliary input and runs protocols *concurrently* in the *public-key* model (which is just the case of IKE), the KEA assumption can be totally meaningless. Specifically, in such cases, as the challenge C is player's public-key, a valid (X, Y) could be just got from adversary's arbitrary auxiliary input (e.g., collecting/eavesdropping over the network from executions of other different protocols prior to any interaction of the protocol at hand) or from up-to-now transcript of concurrent executions of the protocol at hand. In other words, there could be practical ways to break the KEA assumption if the challenge C is public-key and $\mathscr{A}$ is allowed with arbitrary auxiliary input and/or concurrent executions in the public-key model, which is however very natural and practical for cryptographic schemes when they are run concurrently in unpredicted adversarial settings like the Internet (in particular this is the case for IKE).

We note that the security proofs of PA properties in [4, 9] are critically based on the assumption that adversaries have no arbitrary (rather than well pre-specified) auxiliary input. In particular, the plaintext awareness properties of the public-key encryption schemes considered in [4, 9] will not hold w.r.t adversaries with arbitrary auxiliary input. Furthermore, it implies that the deniability of the PKE-based message-authenticator of [2] and the SKEME key-exchange protocol of [18], as proven in [11], will also not hold w.r.t adversaries with arbitrary auxiliary input when the underlying public-key encryption scheme is implemented by the Cramer-Shoup encryption scheme. (Note that the Cramer-Shoup encryption scheme is currently the only known PA2-secure public-key encryption scheme based on KEA but without random oracles [9].) This is the case even for the variants of the PKE-based message-authenticator and SKEME key-exchange protocols considered in [11], where fresh nonce is sent on the top to frustrate cooperating judge attack. We note that security proof (for the principal security of HMQV against revealing ephemeral exponent) in [20], that uses both the KEA assumption and the random oracle assumption, will also not work when adversaries have arbitrary auxiliary inputs.

Fresh challenge KEA assumption with arbitrary auxiliary input. According to the above clarifications, when using the KEA assumption for protocols running in the public-key model, we allow the algorithm $\mathscr{A}$ to take arbitrary auxiliary input, but require that the challenge C is fresh in the sense that each C is answered by $\mathscr{A}$ at most once. This implies that C cannot be player's public-key. Such a revised KEA assumption is called *fresh challenge KEA assumption with arbitrary auxiliary input*, which is necessary and reasonable for protocols running concurrently in the public-key model.

Note that, as clarified, the arbitrary auxiliary input captures any information collected over the network from executions of arbitrary different protocols prior to any interaction of the protocol at hand. When using the KEA assumption for provable security of protocols running concurrently in the public-key model, the protocol should be designed that the transcript of the concurrent execution of the protocol at hand should also not help $\mathscr{A}$ break the KEA assumption, which is left in the security analysis of the protocol at hand and should be the responsibility and care of the protocol designer.

3.3 Discussion and analysis: security features and properties

Some notable features of our deniable IKE protocol. We first briefly point out some notable features of our deniable IKE protocol.

- Peer's ID and public-key concealment: messages from one party do not include its peers's ID and public-key. This perfectly matches the requirement of working in the post-specified-peer model and the principle of privacy-preserving.

- Concurrent non-malleable (and even maybe universal composable) zero-knowledge (CNMZK/UCZK) authentic binding: party's ID (e.g., A), its DH-component (e.g., g^x), its proofs of knowledge of its secret-key (i.e., a or b) and the discrete logarithm (i.e., x or y) of its DH key component (i.e., X or Y), and the session-specific information (e.g., (sid, g^x, g^y)) are authentically bounded in a CNMZK/UCZK way, through $NMZK_{(B,y)}$, $NMZK_{(a,x)}$ and $NMZK_{(b,y)}$. $NMZK_{(B,y)}$ denotes that it is a non-malleable zero-knowledge proof that the second round message is bounded to responder

B who knows y. $NMZK_{(a,x)}$ (resp., $NMZK_{(b,y)}$) denotes that it is a NMZK proof of knowledge of (a, x) of initiator A (resp., (b, y) of responder B.) This essentially implies that messages of one session cannot be malleated into other successful sessions, or messages of one successful session cannot be malleated from other sessions, other than just simply copying which however is not deemed to be a harmful activity (more details can be found in the following analysis).

- All authentic messages from B, i.e., $NMZK_{(B,y)}$ and $NMZK_{(b,y)}$, can be computed merely from the discrete logarithm, i.e., x, of it's peer's DH key component X (without knowing (a, b, y)!), and A does have proved the knowledge of x via $NMZK_{(a,x)}$; The authentic message $NMZK_{(a,x)}$ can be computed merely from the discrete logarithm, i.e., y, of it's peer's DH key component Y (without knowing (a, b, x)!), , and B does have proved the knowledge of y via $NMZK_{(B,y)}$ and $NMZK_{(b,y)}$. This is typically fair for the privacy concerns for both A and B.

Security in the post-specified-peer model. We show that, under fresh-challenge KEA assumption with arbitrary auxiliary input and the random oracle assumption of H and computational Diffie-Hellman (CDH) assumption, the protocol depicted in Figure 1 satisfies SK-security in the post-specified-peer model of [7], which implies secure communication channels via the shared session-keys and perfect forward security. The security analysis follows, in a very essential way, the security analysis of the variant of SIGMA (combining MAC and signatures) presented in [7], which is quite complicated and lengthy and is deferred to the full version. Below, we provide some high-level hints of the proof. Our IKE protocol actually could correspond to a variant of SIGMA in which the MAC is combined into the signature, e.g., combining $SIG(sid, g^x, g^y)$ and $MAC(sid, A)$ into $SIG(MAC(sid, g^x, g^y, A))$. This variant of SIGMA is still secure in the post-specified-peer model [7]. In view of our protocol, we note that the first three rounds of our IKE protocol actually amount to a deniable authentication (i.e., interactive deniable public-key signature) with A as the signer, and the last three rounds amount to a deniable authentication with B as the signer (more details are deferred to the full version).

Forward and concurrent (non-malleable) deniability. We show that, under the fresh challenge KEA assumption with arbitrary auxiliary input and the RO assumption of H and the CDH assumption, the protocol depicted in Figure 1 is forward and concurrently (non-malleable) deniable. For any PPT MIM-adversary $\mathscr{A}$ concurrently interacting with instances of A and B with possible different IDs/names and public-keys in different sessions, we briefly hint at the security proof.

From the viewpoint of party A, for the *valid* second-round message $(sid, M, g^y, NMZK_{(M,y)} = MAC_{H(sid,g^y,g^x,g^{xy})}(M))$ of the (A, sid)-session, received from $\mathscr{A}$ in the name of M of public-key g^m w.r.t a fresh challenge g^x generated and sent by the simulator (in the name of A) at the first-round, if g^y was not generated by $\mathscr{A}$ and $M \neq B$ and $\mathscr{A}$ did not query the random oracle with (sid, g^y, g^x, g^{xy}), there are essentially two possible ways for $\mathscr{A}$ to compute out such message:

1. $NMZK_{(M,y)}$ is malleated by $\mathscr{A}$ from some second message sent by honest B (who generated g^y and sent $NMZK_{(B,y)}$), by breaking the security of the underlying MAC. Note that the session-ID sid is unique to (A, sid) and its matching session (B, sid), and the order of (g^y, g^x) or (g^x, g^y) indicates the role of player (i.e., whether responder B or initiator A).

2. Compute g^{xy} from g^x and g^y, and guess successfully the answer of the random oracle of H on (sid, g^y, g^x, g^{xy}).

By the random-oracle of H and the security of the underlying MAC and the CDH assumption, $\mathscr{A}$ must query the random oracle with (sid, g^y, g^x, g^{xy}), which just contains a valid answer (g^y, g^{xy}) w.r.t. the freshly generated challenge of g^x. By the KEA assumption, the simulator (who programs the random oracle) can extract y, and then finish the (A, sid)-session perfectly. Note that in the third-round message $g^{y(x+a \mod q)} = g^{xy} \cdot g^{ay}$, which can be computed from the extracted value y by the simulator without knowing a (and even x).

Now, from the viewpoint of B, in the session (B, sid), the simulator works as follows: after receiving the first-round message g^x from $\mathscr{A}$, it perfectly computes the second-round message in the name of B by

generating g^y. Note that the second-round message, in particular $NMZK_{B,y}$ perfectly hides g^{xy} due to the random-oracle of H. For the third-round message sent by $\mathscr{A}$ in the name of M with public-key g^m, similar to above arguments, $\mathscr{A}$ must have queries the random oracle with $(sid, g^x, g^y, Y^m, Y^x)$. Then, by the KEA assumption w.r.t. fresh challenge $Y = g^y$, the value x can be extracted. Note that the fourth-message can then be perfectly computed from the extracted value x by the simulator without knowing b (and even y).

On the dual roles of (sid, g^x, g^y)**.** We remark that including $(sid, X = g^x, Y = g^y)$ or $(sid, Y = g^y, X = g^x)$ into the input of H plays dual roles: On the one hand, the uniqueness of sid provides freshness to each session in order to prevent *replaying* attacks; the order of (g^x, g^y) or (g^y, g^x) indicates the role of player (i.e., whether the initiator A or the responder B), which is necessary for resistance against *reflection* attack. On the other hand, the inclusion of (sid, g^x, g^y) or (sid, g^y, g^x) plays a critical role in the above proof of concurrent deniability (concurrent ZK). Specifically, as mentioned, the KEA assumption is a *non-black-box* assumption, and the CZK simulator needs to extract x (resp., y), directly in a straight-line way, from (the codes, random coins, and partial transcript up to now) of $\mathscr{A}$ just at the point where it queries the random oracle H with valid (Y, Y^x) (resp., (X, X^y)) w.r.t. a freshly-generated g^x (resp., g^y). But, if the input to H does not include (sid, g^x, g^y) or (sid, g^y, g^x), the CZK simulator cannot realize whether random-oracle queries made by $\mathscr{A}$ include valid Y^x (resp., X^y) to a fresh challenge g^x (resp., g^y). *Important note: $\mathscr{A}$ may query the oracle with g^{xy} after receiving g^x but much earlier before sending the second-round message of that session that contains $Y = g^y$ (and so the CZK-simulator cannot check the validness of g^{xy} before receiving Y).* This means that the CZK simulator has to rewind $\mathscr{A}$ whenever it needs to extracts y after receiving the second-round message in question (including Y) from $\mathscr{A}$ (as the value y may not be extracted from a later state of $\mathscr{A}$). But, such rewindings cause exponential blow-up in the simulation time of the CZK-simulator [12]. Furthermore, even if y could still be extracted from a later state of $\mathscr{A}$, the CZK-simulator has to compute polynomially many exponentiations in order to check whether $\mathscr{A}$ ever queried with Y^x in the history of transcript, which causes very high inefficiency of the CZK simulation. In a way, (sid, X, Y) or (sid, Y, X) serves as a pointer to help the CZK-simulator realize *immediately* whenever the random-oracle queries made by $\mathscr{A}$ includes valid Y^x or X^y.

According to above discussions and clarifications, we have the following theorem:

Theorem 3.1 (informal) *Under fresh-challenge KEA assumption with arbitrary auxiliary input and the random oracle assumption of H and CDH assumption, the IKE protocol depicted in Figure 1 satisfies SK-security in the post-specified-peer model, and is forward and concurrently deniable.*

But, as noted in [20, 23], there are some important security properties of key-exchange that are not captured by the security model of [2, 7]. We analyze those properties case by case below.

Resistance against cutting-last-message attack. We remark that the cutting-last-message attack, suffered by IKEv2 and SIGMA, does not work on our IKE protocol (the cutting-last-message attack on IKEv2 and SIGMA is briefly recalled in Appendix A). Specifically, after receiving the second-round message $(B, Y = g^y, NMZK_{(B,y)})$ from B, the MIM-adversary $\mathscr{A}$ cannot compute and send to A the message of $(M, Y = g^y, NMZK_{(M,y)})$ in the name of $M \neq B$, due to the security of MAC and random oracle assumption of H and CDH assumption. In other words, if A receives a valid second-round message $(M, Y' = g^{y'}, NMZK_{(M,y')})$ from $\mathscr{A}$, it implies Y' is generated by M (i.e., $\mathscr{A}$) who knows y'. In this case, the third-round message from A will be w.r.t. Y' generated by M (not w.r.t. Y generated by B), and cannot be forwarded by $\mathscr{A}$ to B in order to fool B to believe it will share a session-key with A, as $Y \neq Y'$.

Resistance against key-compromise impersonation (KCI) attack. Informally, security against KCI attack says that [20]: even if the attacker knows the private-key of a party, e.g., the private-key a of A, still cannot compromise the session-key established between A and honest B (i.e., distinguishing it from independent random value), as long as the attacker is not actively controlling or observing the secret choices, i.e., x in this case, made by A for that session. The resistance against KCI attacks of our IKE protocol can be easily checked, from the CDH assumption and the pseudorandomness of the key-derivation function $KDF(H(g^{xy}), \cdot)$.

Resistance against reflection attack. In some scenarios, a party may want to establish a secure channel with itself (i.e., $A = B$ in this case). For example, a mobile user that communicates to its desktop computer, while both the mobile device and the desktop have the same identity in the form of the same signature certificate $Cert$. In a *reflection* attack, an attacker simply copies (authentic) messages from A and sends them back to A as the messages coming from the other copy of A. Such reflection attack fails with our IKE protocol, just by the inclusion of (sid, g^x, g^y) and (sid, g^y, g^x) that indicates the role of initiator and responder even for copies of the same party.

Resistance against unknown key share (UKS) attack. A successful UKS attack is one in which two parties compute the same session key but have different views of who the peer to the exchange was (in other words, the attacker successfully managed to make two different sessions, among different pairs of players, to output the same session-key). The resistance against UKS attack of our IKE protocol is from its NMZK authentic binding nature: player's ID (A or B), its DH key component (g^x or g^y), the NMZK proofs of knowledge of its secret-key (i.e., a or b) and the discrete logarithm (i.e., x or y) of its DH key component (i.e., X or Y), and the specific session information ((sid, g^x, g^y) or (sid, g^y, g^x)), in our IKE protocol, are bounded *in a non-malleable authentic way* through $NMZK_{(B,y)}$ and $NMZK_{(a,x)}$ and ($NMZK_{(b,y)}$, in the sense that they adhere to the specific session and cannot be malleated into/from other sessions. Note that, a successful UKS attack implies that such non-malleable authentic binding is broken, which in turn implies that at least one of the KEA assumption, the security of MAC, RO assumption of H and the CDH assumption, will be broken. We remark that the resistance against UKS attack of our IKE protocol holds even for adversaries who managed to register a public-key that is identical to that of some honest player but with different name, e.g., an adversary can register (M, g^a) to the CA for certification where g^a is actually public-key of A. Note that we do not require player to prove the knowledge of its secret-key during key registration (which is common in practice). Rather, each party is required to prove the knowledge of the secret-key to its peer in the NMZK authentic binding way.

Resistance against leakage of ephemeral DH exponents. That is, the leakage of ephemeral DH exponent, x or y, should not affect the security of sessions where this exponent is not used. In particular, the leakage of ephemeral DH exponents, x or y, of one session should not help the attacker to impersonate A or B in other sessions. This property is from the observation: the NMZK authentic binding of one session, between two honest parties, holds even if the attacker learns some ephemeral DH exponents from other sessions. In particular, note that our IKE implies deniable authentication (i.e., deniable interactive public-key signatures) both for A and for B, and thus an attacker cannot impersonate A or B without knowing the corresponding long-term secret-key (unless the CDH assumption or the security of the underlying MAC is broken).

Protocol variants. One recommendable protocol variant is to put one's own ID (the input of MAC) additionally into the input of H. This variant is still provably secure in the post-specified-peer model, but seemingly provides extra stronger security guarantee in practice. Specifically, even the security of MAC is broken, an adversary is still infeasible to output valid NMZK proofs. For example, given $NMZK_{(B,y)} = MAC_{H(sid,B,g^y,g^x,g^{xy})}(B)$ (in this case), a valid $NMZK_{(M,y)}$ output by an attacker with $M \neq B$ will have to be $MAC_{H(sid,M,g^y,g^x,g^{xy})}(M)$. In other words, with this variant, previous MAC values sent by B with MAC-key $H(sid, B, g^y, g^x, g^{xy})$ provides no extra advantage to an attack in order to forge a valid MAC value with different name $M \neq B$, as in this case the attack has to use MAC-key $H(sid, M, g^y, g^x, g^{xy})$ that is however independent of $H(sid, B, g^y, g^x, g^{xy})$. A further simplification of the above variant is to remove MAC, and only use hash function H to make authentic binding. We caveat that, the provable security of this simplified variant is not directly followed from the security analysis of [7], and we do not have provable security with this variant currently. In practice, we are against such over-simplified variant, as the security will critically rely on the *collision-resistance* and *one-way hiding* of hash function in practice (which we do not want that). In addition, some public session-specific information, e.g., (sid, X, Y) and (sid, Y, X) could also be put into the input of MAC. When our IKE protocol works in the pre-specified-peer setting and/or players do not much care about privacy preserving, peer's ID can also be put into the input of H (and/or the input of MAC).

4 Deniable Internet Key-Exchange: the Aggressive Model

We now present the 3-round variant of our deniable IKE protocol in accordance with the aggressive model of IKE, which is depicted in Figure 2 (page 10). For presentation simplicity, the public system parameters, player's keys and the key-derivation function KDF are omitted here, which remain the same as in the deniable IKE of the main model (depicted in Figure 1).

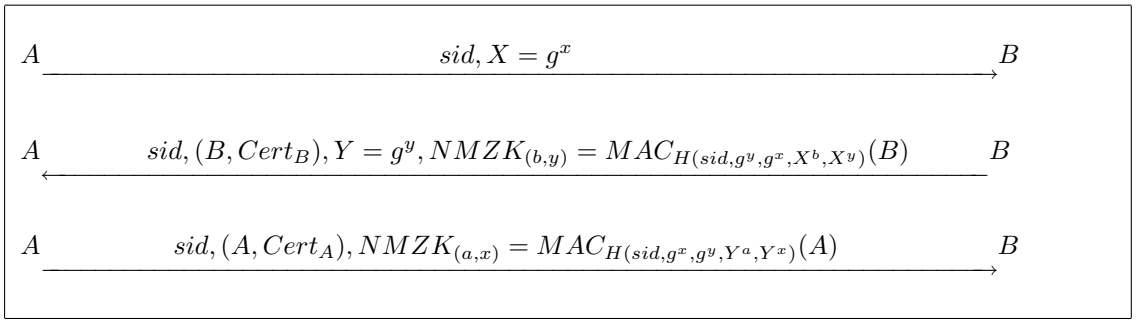


Figure 2: Deniable IKE (the aggressive model)

Most security properties of the deniable IKE of the plain model are essentially inherited by the above deniable IKE of the aggressive model, except for the full deniability for player B. Specifically, player B has only *completed-session deniability*, in the sense that if A^* completes the session then B's deniability will be guaranteed. But if the (possibly malicious) A^* just aborts the session after receiving second-round message, then the deniability for B cannot be proved. Note that player A still has full deniability. We remark that completed-session deniability for responder is still very useful and reasonable. Consider the scenario that A is a client and B is a (bank or shop) server: in this case it is the client A who cares more about its privacy and full deniability does guarantee for it, while the server cares less about the deniability and a completed-session deniability is still deemed to be good enough for it.

5 Some Observations on the Independent SW-KE Protocol

The Stinson-Wu key-exchange (SW-KE) protocol of [26] is recalled in the left column of Figure 3. (In the description of [26], B is the initiator and A is the responder, but it does not pose any essential matter.) The public system parameters and player's keys (g^a for A and g^b for B) are identical with our deniable IKE protocol. We stress that our deniable IKE and authentication protocols are developed fully independently of [25, 26].

SW-KE does not work in the post-specified-peer model. Note that the computation of the first-round message from A needs B's public-key g^b, and the computation of the second-round message from B needs both A's public-key g^a and B's public-key g^b. We remark that mixing *both* parties' IDs in message computation violates the privacy-preserving principle of IKEv2, i.e., message from one party should not involve it's peer's ID.

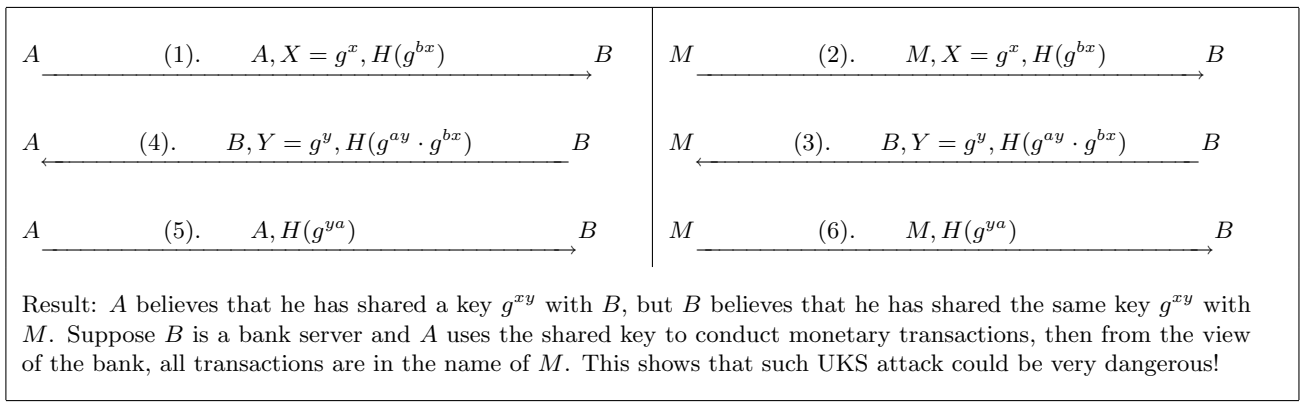
On limited deniability and privacy of SW-KE. B has no deniability over Internet (or against adversary with auxiliary input) even for completed sessions. Specifically, (X, g^{bx}) and/or $(X, H(g^{bx}))$ could just be collected from network, or given by a judge, or eavesdrop earlier execution of the SW-KE protocol. In this case, B cannot deny the second-round message *even for completed sessions*. Similarly, A also has no deniability against adversary with auxiliary input. Specifically, consider (Y, g^{ay}) is collected from network or given by a judge, then A cannot deny the sending of the third-round message $H(g^{ya})$.

With respect to the fairness concern of peer's privacy, note that for our deniable IKE, the authentic information $NMZK_{(a,x)}$ from A can be computed merely from y (without x, a, b), and all the authentic information from B, $NMZK_{(B,y)}$ and $NMZK_{(b,y)}$, can be merely computed from x (without y, a, b). In other words, messages from one party cannot be linked to its ID and its DH-component as well as its

peer's ID. This is thus fair for the concern of privacy for both parties. The SW-KE protocol is clearly not of this case.

Reflection attack. It's easy to see that the SW-KE protocol suffers from reflection attack (discussed in Section 3). Specifically, by running A concurrently in two sessions, and forwarding messages received from A in one session to A in another session, an attacker can complete a session with A in the same name of A.

Dangerous UKS attack! Note that, as in the setup for our deniable IKE, the SW-KE protocol explicitly does not require proof-of-possession during parties' key-registrations with the CA. Now, consider that an attacker $\mathscr{A}$ registers to CA the public-key g^a (i.e., the public-key of the honest A) in the name of M. Then, the message flows of the UKS attack carried by $\mathscr{A}$ is depicted in Figure 3.



Result: A believes that he has shared a key g^{xy} with B, but B believes that he has shared the same key g^{xy} with M. Suppose B is a bank server and A uses the shared key to conduct monetary transactions, then from the view of the bank, all transactions are in the name of M. This shows that such UKS attack could be very dangerous!

Figure 3: The UKS attack against SW-KE

Some fixing suggestions of SW-KE. One suggestion is to combine $H(g^{bx})$ into the third-round as $H(g^{bx} \cdot g^{ay})$. In any case, we should put session-specific information into the input H. Specifically, for A to put (sid, A, B, g^x, g^y) and for B to put (sid, B, A, g^y, g^x). More details are given in the full version.

References

[1] M. Bellare, R. Canetti and H. Krawczyk. Keying Hash Functions for Message Authentication. In *N. Koblitz (Ed.): Advances in Cryptology-Proceedings of CRYPTO 1996, LNCS 1109*, Springer-Verlag, 1996.

[2] M. Bellare, R. Canetti and H. Krawczyk. A Modular Approach to the Design and Analysis of Authentication and Key Exchange Protocols. In *ACM Symposium on Theory of Computing*, pages 419-428, 1998.

[3] M. Bellare and A. Palacio. The Knowledge-of-Exponent Assumptions and 3-Round Zero-Knowledge Protocols. In *M. Franklin (Ed.): Advances in Cryptology-Proceedings of CRYPTO 2004, LNCS 3152*, pages 273-289, Springer-Verlag, 2004.

[4] M. Bellare and A. Palacio. Towards Plaintext-Aware Public-Key Encryption without Random Oracles. In *P. J. Lee (Ed.): Advances in Cryptology-Proceedings of Asiacrypt 2004, LNCS 3329*, pages 48-62, Springer-Verlag, 2004.

[5] M. Bellare and P. Rogaway. Optimal Asymmetric Encryption. In *Advances in Cryptology-Proceedings of EUROCRYPT 1994, LNCS 950*, Springer-Verlag, 1994.

[6] R. Canetti and H. Krawczyk. Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In *Advances in Cryptology-Proceedings of EUROCRYPT 2001, LNCS 2045*, Springer-Verlag, 2001. Available also from Cryptology ePrint Archive, Report No. 2001/040.

[7] R. Canetti and H. Krawczyk. Security Analysis of IKE's Signature-Based Key-Exchange Protocol. In Crypto 2002.

[8] I. Damgaard. Towards Practical Public-Key Systems Secure Against Chosen Ciphertext Attacks. In *J. Feigenbaum (Ed.): Advances in Cryptology-Proceedings of CRYPTO 1991, LNCS 576*, pages 445-456. Springer-Verlag, 1989.

[9] A. Dent. Cramer-Shoup Encryption Scheme is Plantext Aware in the Standard Model. In *Advances in Cryptology-Proceedings of EUROCRYPT 2006, LNCS 4004*, pages 289-307. Springer-Verlag, 2006.

[10] M. Di Raimondo and R. Gennaro. New Approaches for Deniable Authentication. IN proc. of 12nd ACM Conference on Computer and Communications Security (ACM CCS'05), ACM Press, pages 112-121, 2005.

[11] M. Di Raimondo, R. Gennaro and H. Krawczyk. Deniable Authentication and Key Exchange. To appear in ACM CCS'06. Full version appears in Cryptology ePrint Archive Report No. 2006/280.

[12] C. Dwork, M. Naor and A. Sahai. Concurrent Zero-Knowledge. In *ACM Symposium on Theory of Computing*, pages 409-418, 1998.

[13] S. Hada and T. Tanaka. On the Existence of 3-Round Zero-Knowledge Protocols. In *H. Krawczyk (Ed.): Advances in Cryptology-Proceedings of CRYPTO 1998, LNCS 1462* , pages 408-423, Springer-Verlag, 1998.

[14] *D. Harkins and D. Carreal (Ed.): The Internet Key-Exchange (IKE)*, RFC 2409, Nov., 1998.

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

[16] C. Kaufman. Internet Key Exchange (IKEv2) Protocol. The Internet Engineering Task Force: INTERNET-DRAFT, October 2002.

[17] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. Request for Comments 2401, 1998.

[18] H. Krawczyk. SKEME: a Versatile Secure Key Exchange Mechanism for Internet In *proc. of 1996 IEEE Symposium on Network and Distributed System Security (SNDSS'96)*, pages 114-127.

[19] H. Krawczyk. SIGMA: the "SIGn-and-MAc" Approach to Authenticated Diffie-Hellman Protocols. In Crypto 2000.

[20] H. Krawczyk. HMQV: A High-Performance Secure Diffie-Hellman Protocol. In *V. Shoup (Ed.): Advances in Cryptology-Proceedings of CRYPTO 2005, LNCS 3621*, pages 546-566. Springer-Verlag, 2005.

[21] L. Law, A. Menezes, M. Qu, J. Solinas and S. Vanstone. An Efficient Protocol for Authenticated Key Agreement. *Designs, Codes and Cryptography*, 28: 119-134, 2003.

[22] W. Mao. Modern Cryptography: Theory and Practice. Prentice Hall PTR, 2004.

[23] A. Menezes. Another Look at HMQV. Cryptology ePrint Archive, Report No. 2005/205.

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

[25] D. R. Stinson and J. Wu. An Efficient and Secure Two-Flow Zero-Knowledge Identification Protocol. Cryptology ePring Archive, Report No. 2006/337.

[26] D. R. Stinson and J. Wu. A Zero-Knowledge Identification and Key Agreement Protocol. Cryptology ePring Archive, Report No. 2007/116.

A The SIGMA Protocol and Cutting-Last-Message Attack

SIGMA KEY-EXCHANGE PROTOCOL. We briefly recall the SIGMA protocol of [19] (the underlying basis of IKEv2), which is depicted in Figure 4 (page 13).

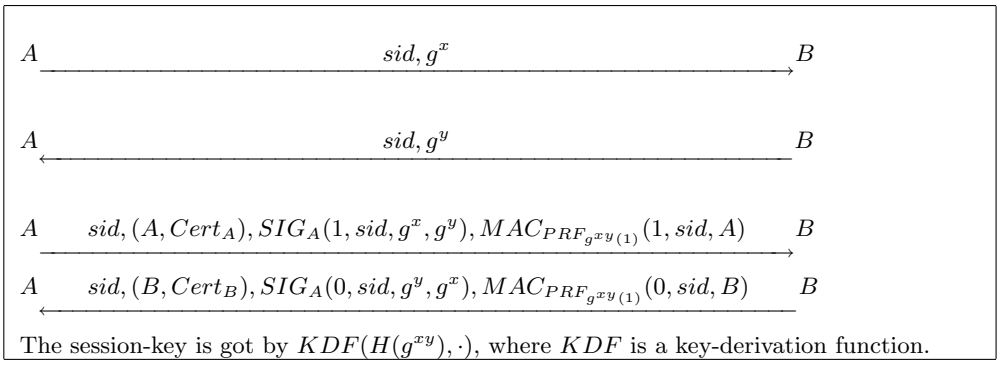


Figure 4: The SIGMA protocol (the main model)

The "cutting-last-message" attack works as follows [22]: A man-in-the-middle $\mathscr{A}$ interacts with A in the name of M, while concurrently interacting with B in the name of A. M just relays messages between A and B, but aborts after receiving the last message from B. Such a simple attack results in authentication failure as follow: B is perfectly fooled to believe it has shared a session key with A, while A only thinks it ever took part in an aborted session with M. The above attack can be simply prevented by adding additional fifth-round of "acknowledgement" from A to B, but increasing the system complexity.