9-2022

On the Cryptographic Deniability of the Signal Protocol

Nihal Vatandas
The Graduate Center, City University of New York

On the Cryptographic Deniability of the Signal Protocol

by

Nihal Vatandas

A dissertation submitted to the Graduate Faculty in Computer Science in partial fulfillment of the requirements for the degree of Doctor of Philosophy, The City University of New York

2022

On the Cryptographic Deniability of the Signal Protocol

by

Nihal Vatandas

This manuscript has been read and accepted by the Graduate Faculty in Computer Science in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

Date	**Rosario Gennaro** Chair of Examining Committee
Date	**Ping Ji** Executive Officer

Supervisory Committee:

Nelly Fazio

Hugo Krawczyk

William E. Skeith

The City University of New York

Abstract

On the Cryptographic Deniability of the Signal Protocol

by

Nihal Vatandas

Advisor: Professor Rosario Gennaro

Offline deniability is the ability to *a posteriori* deny having participated in a particular communication session. This property has been widely assumed for the Signal messaging application, yet no formal proof has appeared in the literature. In this work, we present the first formal study of the offline deniability of the Signal protocol. Our analysis shows that building a deniability proof for Signal is non-trivial and requires strong assumptions on the underlying mathematical groups where the protocol is run.

To do so, we study various implicitly authenticated key exchange protocols, including MQV, HMQV, and 3DH/X3DH, the latter being the core key agreement protocol in Signal. We first present examples of mathematical groups where running MQV results in a provably non-deniable interaction. While the concrete attack applies only to MQV, it also exemplifies the problems in attempting to prove the deniability of other implicitly authenticated protocols, such as 3DH. In particular, it shows that the intuition that the minimal transcript produced by these protocols suffices for ensuring deniability does not hold. We then provide a characterization of the groups where deniability holds, defined in terms of a knowledge assumption that extends the Knowledge of Exponent Assumption (KEA).

We conclude our research by presenting additional results. First, we prove a general theorem that links the deniability of a communication session to the deniability of the key agreement protocol starting the session. This allows us to extend our results on the deniability of 3DH/X3DH to the entire Signal communication session.

We show how our Knowledge of Diffie-Hellman Assumptions (KDH) knowledge assumption family can be used to establish a deniability proof for other implicitly authenticated Diffie-Hellman protocols, specifically the OAKE family [80].

By examining the deniability of the implicitly authenticated AKE protocols augmented with a confirmation step, we also demonstrate a counterintuitive result. Although such a modification requires protocol users to exchange additional information during the session, deniability may be established for these protocols under weaker assumptions (compared to the implicitly authenticated version).

Lastly, we discussed our observations on various attack scenarios that undermine offline deniability with the assistance of third-party services and why these attacks should be put in a different category than offline deniability.

Acknowledgments

I would like to express my gratitude to my advisor, Prof. Rosario Gennaro, for giving me the opportunity to pursue research in the CAISS Lab and for his support over the years. I'm very grateful to him for his efforts, patience, and numerous discussions we had, which allowed me to develop and complete my doctoral research. I greatly benefited from his perspective and productive discussions on our research problems. I consider myself extremely fortunate, and I will greatly miss being his student.

I owe many thanks to Hugo Krawczyk for the very efficient collaboration and his support. His significant contributions, insightful and detailed comments, and suggestions greatly improved this work. It was a great pleasure for me to work with him.

I would like to thank Nelly Fazio, Hugo Krawczyk again, and William E. Skeith for their willingness to serve on my dissertation's exam committee and for their guidance and contributions that helped this dissertation take its final form.

During my PhD years, I had many friends in the CAISS office, including Marios Georgiou, Bertrand Ithurburn, Sima Jafarikhah, Kelsey Melissaris, and Matteo Campanelli, who created a pleasant working atmosphere. I'm especially grateful for the friendship of Marios and Bertrand, as well as many fruitful discussions and research collaborations. I will always remember and miss the enjoyable moments we shared together in the office and on lunch breaks.

I'd like to thank my friends Serra, Başak, Cavidan, Ferda, Hilal, Hande, Berfin, Dilan,

and Eslem for enriching my life and being my family in New York, far away from home.

My deepest gratitude is reserved for my parents, my brother, and my grandfather. They were my sole source of strength at the times when I needed courage, patience, and endurance. I would never have found the strength to complete my degree without the support of my dearest mother from the very beginning to the very end.

Note: This dissertation is the extended version of the study that has been presented and published at the International Conference on Applied Cryptography and Network Security in 2020 [76].

Contents

List of Figures

Chapter 1

Introduction

1.1 Deniable Communications

In digital communications, we authenticate ourselves to other parties, usually through the use of a secret key that is linked to our identity. Depending on the application, this could be a public/private key pair such as a signature key, an encryption key, or it could be a pre-shared symmetric key confirming our identity to our interlocutor.

Cryptographic deniability refers to the ability to deny our involvement in a particular communication session. This is accomplished not by obscuring our digital presence, which exists as a result of our usage of identity keys, but rather by rendering our presence ineligible for use as proof of our participation in a particular communication session. This is usually possible by designing protocols in such a way that potential evidence of our interaction might be easily fabricated as well from the information available to others.

Informally, deniable communications ensure that two parties communicating online will be unable to prove to a third party that any communication actually happened. The source of threat against the deniability of a user could be an outsider eavesdropping on the conversation as well as the user's peer in the communication. Deniability should be maintained in the

latter case, too, even if one of the peers violates the protocol's instructions in order to generate a proof associated with the other party's identity.

Given the fact that the source of threat to our deniability could be our peer, to whom we are authenticating, the concept of deniability may appear as a conflicting term. After all, the protocol is set to prove our identity, while the generated output cannot be linked to our identity. Indeed, a deniable protocol assures a user Alice that she is speaking with Bob (Bob authenticates), but this assurance should be unique to Alice and not "transferable" to a third party. Thus Bob's messages should not be convincing to anyone other than Alice. [1]

A key observation is that communications authenticated via a shared symmetric key are intrinsically deniable: any authenticated message held by one of the parties could have been produced by either party (since they both know the key) and therefore is not a proof of the origin of the message. In other words, when Alice receives a message authenticated by Bob, she knows it comes from Bob because she did not send it, and only Alice and Bob know the authentication key. On the other hand, a third party will not be able to distinguish if the message was authenticated by Alice or Bob.

Deniable authentication is based on a similar concept in the asymmetric public key setting. Authenticated messages might have been produced by either side and are therefore useless as proof for a third party to implicate a specific participant.

[1]The assurance that cryptographic deniability offers lies at the algorithmic level, i.e., a deniable protocol does not generate any output that may be used to incriminate the user. Traffic analysis of digital communications may lead to IP to be tracked down to the origin of a message. A deniable authentication running at the IP layer (with the necessary caution paid to the upper layer protocols), for example, prohibits the device's IP address from being associated with the user's identity, i.e. the user's secret key.

1.2 Offline vs Online Deniability

The definition of deniability considers a *judge* whose role is to assess if the presented evidence for a protocol session allegedly between Alice and Bob corresponds to an actual execution of the protocol, or if it is possibly a fabricated "evidence." Different types of deniability for a protocol are conceivable depending on the judge's intervention to the communication session.

Consider a scenario where Bob is trying to prove to the judge that he is talking to Alice. Offline deniability involves a judge who is not present during the communication. Instead, the judge attempts to decide whether Alice engaged in a communication with Bob, upon reading evidence presented by Bob after the session ends. Alternatively, we can consider a judge who participates in the protocol and actively collaborates with Bob during the run of the protocol. In this case, the judge can pick some protocol values on behalf of Bob, which will eventually convince the judge that Bob is receiving information authenticated by Alice. A protocol is online deniable if such a collusion between Bob and the judge is possible while still keeping Bob's identity secret key hidden from the judge. Note that, otherwise, it would be impossible to distinguish between Bob and the judge and therefore deniability would be irrelevant.

Obviously, online deniability is a stronger notion, and indeed not easy to achieve [78, 75, 31]. Notice that an online judge that is able to obtain proof of communication during the run of a protocol may be unable to convince anyone else either, as the "proof" that is convincing for the judge may be nontransferable. That is, a protocol which fails to satisfy online deniability may leave convincing evidence for merely an online colluding party available at the time of protocol run, but it may still be hard to convince anyone else by that "one-time proof" after the communication ends. On the other hand, failure to maintain offline deniability may leave traces that convince everyone (possibly with the assistance of Bob)

after the communication ends. Therefore maintaining offline deniability is of more practical relevance.

1.3 Deniable Authenticated Key Exchange

Communication sessions are usually split in an online authenticated key exchange (AKE) protocol, used by the parties to authenticate each other using their own private/public keys and establish a secret session key. This is followed by a communication session which is protected (authenticated and possibly encrypted) usually by means of symmetric-key techniques using the session key established by the AKE. To ensure the session's deniability, it is always necessary that the session key generated by AKE can be "denied" by either party (*deniable authenticated key exchange*).

Additionally, the deniability of the AKE alone may be sufficient to assure the deniability of the entire session, depending how message authentication is handled in the communication session that follows the AKE. In case the messages are authenticated using a symmetric-key approach that is deniable (since not linked to the peers' identities), the AKE protocol will be the sole focus of attention for deniability.

There are two parties to an authenticated key exchange protocol, Alice and Bob, associated with identity keys (sk_A, pk_A) and (sk_B, pk_B), respectively. The parties run according to the specified protocol, one acting as the initiator and the other as responder and eventually the protocol returns a shared session key K to each user. By the end of a successful execution of the protocol, Alice makes sure that she is talking with Bob, who holds the secret key associated to pk_B, and that Bob is the only other party who can compute the session key K. Similarly Bob has the same assurance.

When Alice and Bob communicate through a deniable AKE, the protocol is supposed to establish a shared secret between two as above, this time without leaving any cryptographic

evidence of communication that can convince a third party. That is, by the end of the protocol Alice is convinced that she is talking with Bob and she cannot prove to anyone else that she communicated with Bob. Also, Alice is guaranteed that Bob, even he is malicious, cannot prove to anyone else that a communication took place.

Deniability for an AKE protocol must be investigated independently for each party, i.e., for the initiator and responder. A protocol may be deniable with respect to only one of these parties. Say, the initiator Alice (or the responder Bob) might be able to deny the session with Bob (resp. Alice), whereas Bob does not have this guarantee. In that case, the protocol is said to be *deniable with respect to initiator (resp. responder).* When deniability assurance holds with respect to each parties, the protocol is said to be a deniable key exchange protocol.

Deniable key exchange is formally defined in [28] based on the simulation paradigm. Informally, the definition says that an AKE protocol is (offline) deniable with respect to Alice if the view of Bob can be simulated by an efficient simulator that does not possess Alice's identity secret key and is allowed to interact with Bob to create the simulation. A successful simulation is the one which cannot be distinguished from the real view of Bob by any efficient distinguisher (judge). Bob's view includes both the protocol's transcript and the associated session key.

The definition is comparable to that of standard zero-knowledge. Similarly, we can interpret this statement to mean that Bob's view does not convey significantly more information to a judge/distinguisher than a simulation produced without Alice's identity secret key. Therefore, Bob's view cannot be used to incriminate Alice, since any information Bob presents to the judge could have been produced by the simulator as well.

The fact that a participant's view includes the session key K implies that deniability of AKE may easily extend to the communication session secured by AKE regardless of how the communication protocol uses K. Common practice is to derive hash or encryption keys from K and employ symmetric-key methods to secure the communication phase. A deniable

session key K ensures deniability of the subsequent operations as long as the identity keys of participants are not involved in the subsequent computations. (See Section 3.3 for further discussion.)

Involving the session key in the view is a critical point and some misconceptions on this point are present in the literature [74, 75]. Evaluating an AKE's deniability solely on the basis of its transcript simulatability may indeed result in inaccurate findings. This approach is consistent with the definition of "deniable authentication protocol" rather than "deniable key exchange."

Simulating only the transcript suggests that the transcript is devoid of any confirmation of the participants' identity, allowing the transcript to be denied. However, the ability to compute K could be a sign of participation in the protocol, too. One might assume that since the ability to compute the session key is symmetric, it cannot be used to incriminate a protocol participant. This viewpoint, however, is incorrect. In Section 1.6, we will look at how implicitly authenticated key exchange protocols allow a party to renounce this ability by executing the protocol dishonestly. Only one party will be capable of computing the session key K in this case, which may work against that party's deniability.

1.4 Restrictions for Deniability Simulation

The idea behind the deniability definition is that because the outputs of a deniable protocol can be generated using a simulator, the outputs of a claimed protocol execution should not be regarded as proof that the protocol was actually performed. In this respect, the simulation is a method of producing "false evidence" to claim the occurrence of a session, rendering the actual protocol outputs ineffective as proof. As a result, the simulation must be executable in the real world.

The fact that deniability simulation must happen in the real world leaves the deniability

simulator devoid of some common tools used in simulation-based proofs. This condition specifically relates to updating publicly visible values that are also accessible to third parties (i.e., distinguishers). Some instances of this situation include programming random oracles in RO model, or selecting common reference string–possibly combined with a trapdoor–in CRS model. Because a RO function or a CRS value is fixed in reality, simulations involving their modification are not applicable.

Another aspect to consider is about rewinding. The rewinding technique is known to cause complications with efficiently completing the ZK simulations in concurrent execution settings. The same concern applies for concurrent executions of deniable protocols, too. Additionally, [33, 78] raised issues about the use of rewinding in deniability simulations. The key problem is that a deniability attacker could produce its messages using a third-party service during the protocol session. When the adversary is rewound, it may be required to rewind the third-party as well in order to maintain desired control over the adversary.

Furthermore, Walfish [78] described an attack strategy based on the exploitation of online third-party services. The attack is intended to demonstrate that protocols requiring rewinding in deniability simulations are inherently non-deniable. Briefly, Walfish's argument appears to be relevant to online deniability, rather than offline deniability. We address this issue in greater detail in Chapter 2.

1.5 Implicitly Authenticated Key Exchange

Diffie-Hellman (DH) key exchange protocol is usually combined with authentication of communicating parties in order to thwart active adversary attacks. Authentication is usually carried out by use of cryptographic primitives, such as signatures, encryption, MAC or hash functions.

A well-known practice in key agreement is for parties to exchange the necessary DH

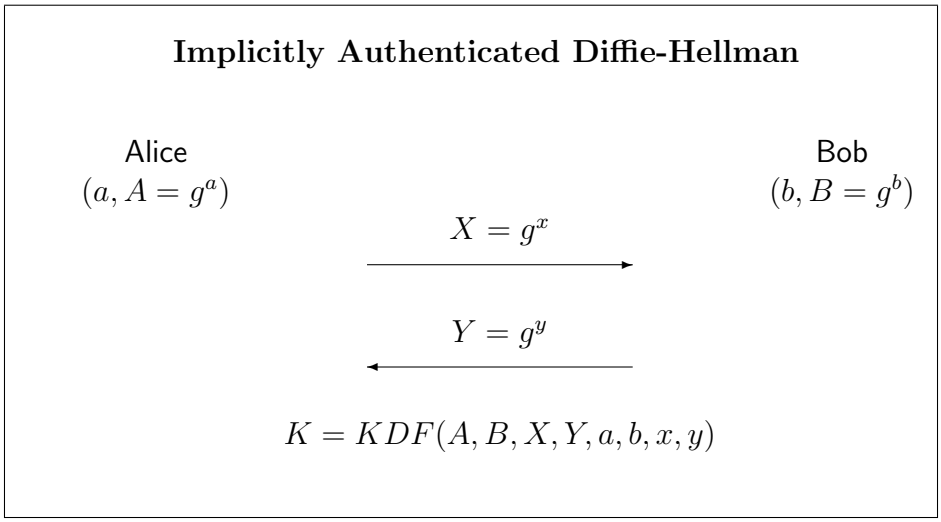

Figure 1.1: A generic representation of implicitly authenticated DH protocols with session key K

components and authentication values, followed by a hash (or MAC) of the transcript at the final stage to ensure that their peer is able to compute the right session key as well. This practice is referred to as *key confirmation.*

A design choice motivated by increasing the efficiency in practice is to defer the confirmation step to the communication session that follows the key agreement phase. That is, after exchanging the values required to compute the session key, each party computes the session key K and prepares to use it to secure the upcoming communication phase, such as generating an encryption key. Once Bob sends a message encrypted by the key derived from K, Alice knows that Bob correctly computed the session key K. Authentication takes place implicitly since Alice knows that the only person who is able to compute the session key beside herself, is Bob. This convinces Alice that the message is coming from Bob and also Bob computed the key correctly. This line of design of KE protocols in which authentication comes from the ability to compute the key is called *implicitly authenticated key exchange* (iAKE).

Implicitly authenticated DH protocols have gained widespread adoption due to their

efficiency in communication cost, which is identical to that of plain (unauthenticated) DH protocol. This type of protocols are extremely simple, as they do not need encryption or a MAC value for authentication purposes. Each user exchanges only a DH component in the protocol message, which is a randomly generated value often used as an ephemeral public key. See examples in Figure 1.1. Due to the absence of any value associated with the users' identities in the protocol transcript, these protocols "appear" convincingly deniable.

Implicit authentication, on the other hand, may result in incrimination that is harder to identify at first glance. A party's ability to compute the session key can be used to incriminate them as well, since authentication relies on this ability. Though this ability is symmetric (both Alice and Bob can compute K), it can be used to incriminate one side through a dishonest twist by the other party. The lack of key confirmation may allow such a dishonest act to go unnoticed from the attention of the victim.

Alice, for example, might choose her ephemeral key pair $(x, X = g^x)$ in such a way that she can prove she doesn't know the secret associated with her ephemeral public key X. Setting the public key X to hash of some value will be strong enough evidence for Alice to convince a judge that she doesn't know her ephemeral secret $x = \log X$. In this situation, Alice deliberately gave up the ability to compute the session key K, and the ability now resides entirely (and uniquely) with Bob. Once Bob sends a message using K, Alice may bring it to the judge as evidence that the message is originated with Bob, as Bob is the only one who can compute K. Obviously, Alice must also demonstrate to the judge that K is the real session key associated with their key exchange transcript in order to establish such a claim (K must be recognizable by the judge).

1.6 Obtaining Incriminating Evidence from AKE Protocols

Incrimination refers to the act of one party to a communication session acting intentionally (and sometimes dishonestly) and demonstrating that its peer has participated in the protocol. Incrimination may occur when a party provides publishable evidence of their participation, such as a signature from Alice, which immediately demonstrating her identity. Alternatively, the evidence may be acquired by demonstrating that Alice performed a computation. In general, we can describe an incriminating evidence as an information obtained from Alice that cannot be simulated without Alice's identity secret key.

Let us walk through the examples of various AKE protocols, to understand how an (incriminating) evidence of involvement can be gained from a participant to a protocol, which may not always be immediately apparent. In all of the scenarios, we are assuming the presence of an offline judge.

Example 1.6.1 (**ISO-9796**)**.** Figure 1.2 shows ISO-9796, a Diffie-Hellman protocol authenticated by signature. Alice and Bob use their signature key pairs as their long-term (identity) keys. Each party provides an ephemeral DH component (X and Y), followed by a signature of both ephemeral keys and the identity of their respective peer. This is an example of a non-deniable AKE because the signatures connect the identities of the communicating parties, providing publicly verifiable evidence that Alice and Bob ran (or attempted to run) a session. Any eavesdropper can gather evidence for this session run without the need for Alice or Bob's assistance.

Example 1.6.2 (**SIGMA-I**)**.** On the other hand, the adoption of signatures does not completely eliminate the possibility of deniability. A communication session may allow participants to sign session values other than their peers' identities while yet maintaining a

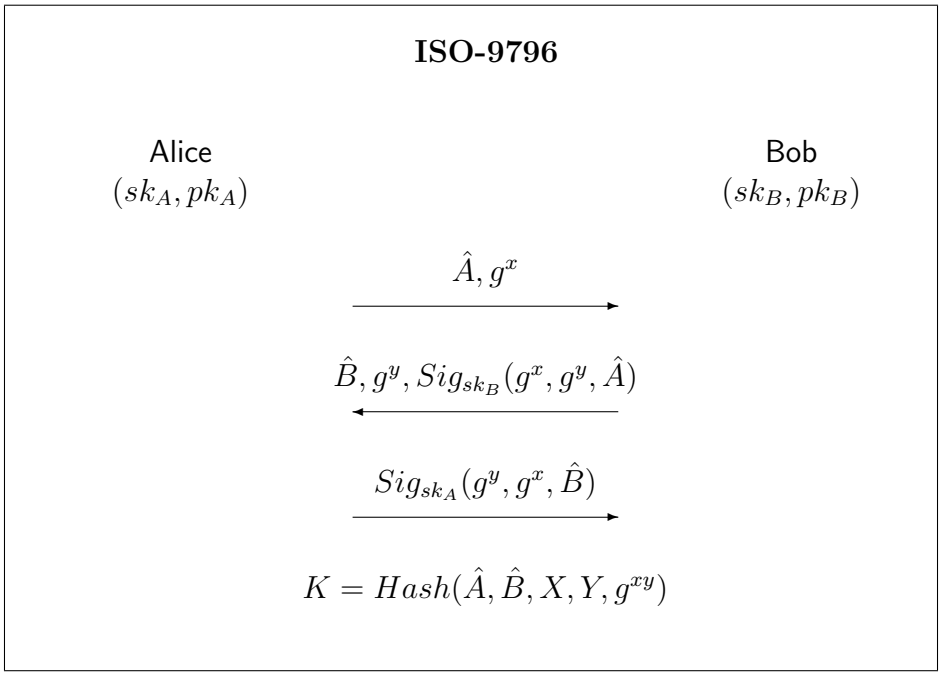

Figure 1.2: ISO-9796 Protocol with session key K

limited degree of deniability on their part.

See Figure 1.3, the SIGMA-I protocol [51], for an illustration of this approach. The protocol uses both signatures and MACs for authentication. Notice that neither party signs any value tied to the identity of the other party. In this example, Bob's signature in the protocol communication indicates that Bob was alive and had run the protocol, even though there is no evidence of the receiver's identity. This gives Bob the ability to deny both the receiver and the content of communications protected under the resulting session key.

The level of deniability that SIGMA-I provides for Bob falls under the category of partial deniability[28]. This is a weaker definition and on the technical side it differs from the original definition in that the simulator is permitted to have an additional oracle access with the simulated party. Specifically, the simulator is given access to an oracle that keeps Alice's secret key and performs the protocol honestly in either initiator or responder role (when deniability for Alice is evaluated). The inclusion of the signatures in the simulation is made

possible thanks to the integration of this extra oracle.

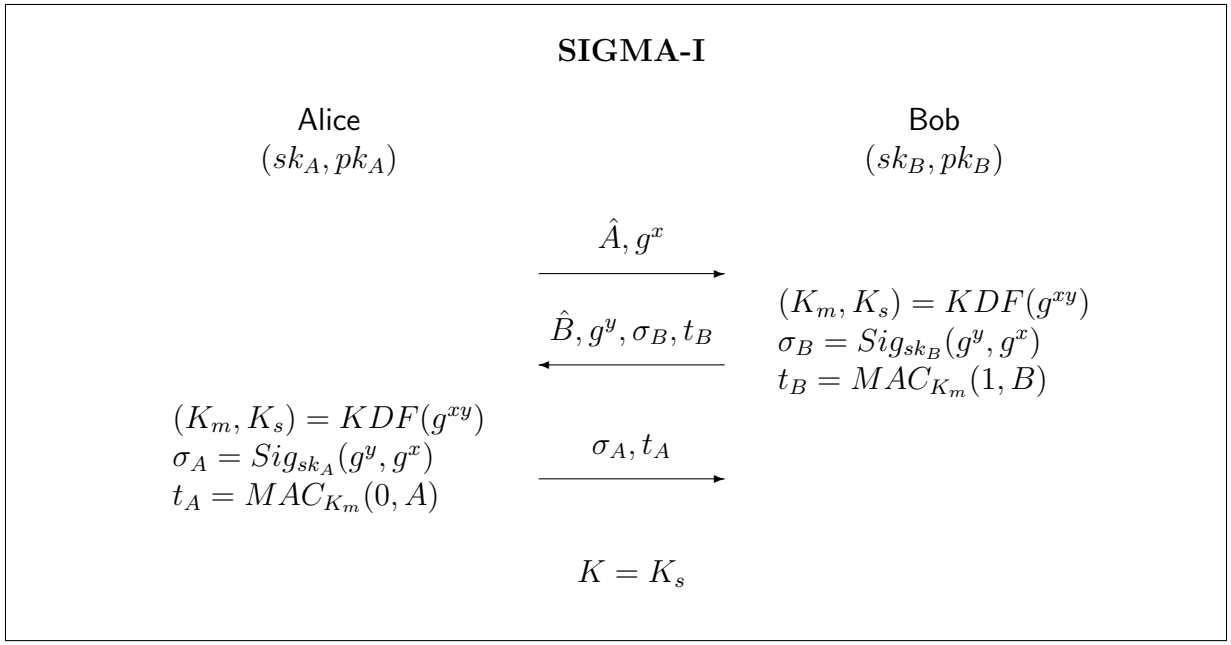

Figure 1.3: SIGMA-I protocol with session key K

SIGMA-I fulfills partial deniability only for Bob, the responder. Additionally, there is a four-round variant of SIGMA (named SIGMA-R) which is partially deniable for both parties. We will examine the three-round variant, SIGMA-I, in order to illustrate how the partial deniability fails for the initiator, which is not quite easy to see.

SIGMA-I is not deniable for Alice, the initiator. This fact manifests itself in the following ways in simulation and in practice, respectively.

In the simulation setting, the additional oracle of Alice can be used in either role (initiator or responder), since signatures in SIGMA-I includes no indicator of the role of the signer. However, oracle in neither role is sufficient for obtaining signatures on the correct values. When Alice's oracle acts as the responder, we recognize that an initial commit from Alice to her ephemeral key is necessary.[2] When the oracle assumes the responder role, SIM is unable

[2]Satisfying this criterion leads in the four-round version, SIGMA-R, that is partially deniable for both parties.

to compute the MAC required to retrieve Alice's signature.

In practice, this situation leads to an incriminating strategy as follows. Bob, the malicious responder, may select g^y based on g^x that he received from Alice. This causes the signatures to reflect the role of the signer: Even though Alice only signs some random looking values (g^y, g^x), the connection between the values and their order reveals that $Sig_{sk_A}(g^y, g^x)$ is created by the initiator.

The core mechanism enabling partial deniability in SIGMA is the flexibility to substitute an initiator's signature for the responder's signature (or vice versa). However, with Bob's strategy outlined above, this feature is removed, hence compromising the initiator's deniability.

Notice that such a strategy only damages initiator's deniability, since the responder in the protocol signs any value g^x received from the initiator without knowing the identity of the initiator. On the other side, the initiator signs the value g^y after the responder's identity is verified. As a result of Bob's strategy, Alice lost the chance to pretend that she acted as responder and signed without knowing who she was talking to.

This is an interesting example that demonstrates how a malicious party might exploit use of a random value by encoding information in it. We will see that the threat to Signal's deniability is caused by a similar strategy by the attacker.

Example 1.6.3 (**SKEME**)**.** SKEME [50], seen in Figure 1.4, is another Diffie-Hellman based protocol, where authentication is carried out by encryption and MAC.

In contrast to the earlier examples, SKEME's transcript only contains the types of messages that both participants can compute. This is insufficient by itself to prove deniability in the strong form of [28]. However, based on this characteristic, we can conclude that the protocol is deniable when both parties act honestly. Due to the fact that both parties are capable of producing any message that could be used as evidence, one party cannot frame

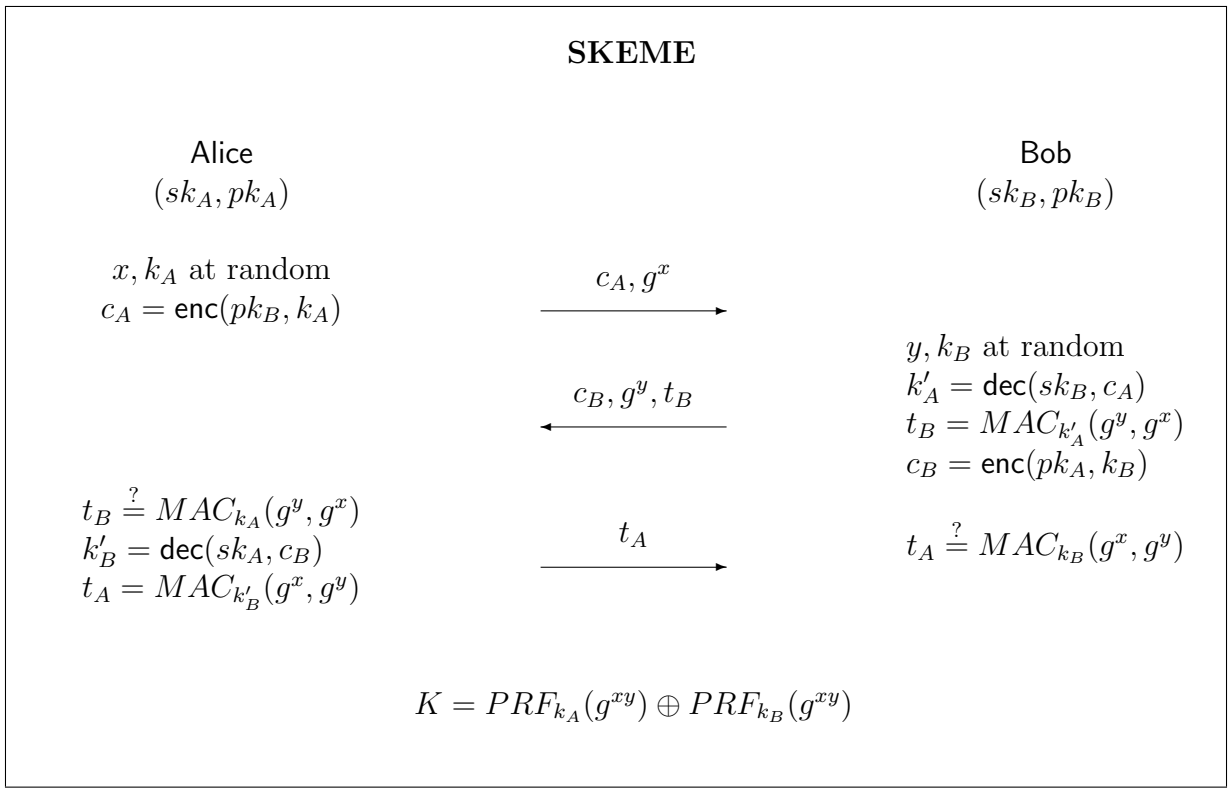

Figure 1.4: SKEME protocol with session key K

the other.

SKEME's security is reliant on the CCA2 security of the encryption used in its construction, even though CCA2 does not suffice to provide deniability protection by itself. Specifically, SKEME is deniable against malicious players when the encryption scheme is chosen to be "plaintext-aware" (PA2)[28].

DiRaimondo et. al. [28] demonstrates how to obtain incriminating evidence from SKEME using a CCA2 encryption.

The encryption scheme they use is specifically tailored for the counterexample, but it highlights a surprising aspect of CCA2 security. Their encryption scheme makes use of an unusual type of public key pair. To encrypt a message m, one merely requires the first half of the public key and uses it to do standard encryption. Similarly for decryption, only first half of the secret key is necessary to decrypt. However the decryption algorithm may optionally deliver the secret information (which is part of the secret key) associated with the unused fraction of the public key. This option can be triggered by anyone doing a simple bit flip on the ciphertext, so it is fairly simple to activate. Enabling this feature allows a party to obtain unsimulatable information from the decryptor indicating that the encrypted message m is the output of the decryption algorithm, and hence was generated by the party holding the decryption (secret) key.

Due to the fact that the additional secret embedded in the public key does not compromise the confidentiality of the message m, the encryption scheme preserves its CCA2 security. On the other hand, it is surprising to discover that a proof of running the decryption algorithm without violating the security conditions is possible. This hard-to-see gap was closed by the plaintext awareness property, which prohibits the ciphertext from being modified in such a way that it decrypts to a message initially unknown to the encryptor.

Example 1.6.4 (**iAKE Protocols**)**.** The transcript of DH-based iAKE protocols are made as minimal as possible; each party sends one DH component only (see Figure 1.1). In the

absence of additional cryptographic primitives such as encryption, MACs, or hashes, the protocol appears to be unproblematic from the deniability aspect: Nothing in the communications conveys information about the participants' identity, and so such protocols must be deniable.

This perception is challenged when faced with the formal definition of deniable AKE. According to the definition, the session key must be simulated as well as the transcript, yet given the simulator's state of knowledge, this may be impossible (depending on the specified key computation).

When simulation is impossible (and the definition is precise), we expect this scenario to manifest itself in practice as an attack strategy. For such an incriminating attack, a potential source of threat with iAKE protocols may be an adversary exploiting the random value's use, as with the SIGMA-I example above. Unlike in the previous example, the adversary will not encode specific information into the random value, but will instead exploit it to demonstrate his lack of knowledge.

The threat scenario follows from the fact that in case of iAKE protocols, the authentication comes from parties to have the ability to compute the session key. This ability resides only on the two communicating parties Alice and Bob. When one of the parties acts dishonestly and purposefully abdicates this ability, the circumstance may become incriminating for the other party.

Let us continue the discussion with 3DH, the predecessor and synchronous version of Signal's X3DH protocol. The reason for choosing 3DH is to keep the example simple by removing the complexity of involving the server. Note that the discussion also applies with minor changes to other iAKE protocols we addressed.

The session key of 3DH is $K = H(g^{ay}||g^{xy}||g^{bx})$, double bars indicating the concatenation. The main authentication mechanism of 3DH is to derive keys from Y^a and X^b.

Assume that an adversary, Bob, receives $X = g^x$ from Alice, who is willing to start a

conversation. Let Bob choose his ephemeral key Y as the hash of some value, such as a fragment from daily newspaper NYT: $Y = H(NYT)$. After receiving Y from Bob, Alice computes the session key K, encrypts some message with it and sends the ciphertext to Bob.

Notice that Bob may take the ciphertext and claim to the judge that the encrypted message originated with Alice. Bob tells the judge that "This is the encryption key K, which I am unable to calculate. This message originated from Alice, who is the only one capable of computing this key." Y and NYT serve as evidence that he does not know $y = \log Y$ and is incapable of computing the key K.

There are a few points to be noted here. First, for Bob's claim to have any convincing value, the judge must recognize the session key, i.e., recognize that K corresponds to the session with public keys A, B, X, Y. When analyzing the initiator's deniability in 3DH, the bottleneck is g^{ay} in the key computation. If g^{ay} is recognizable then the judge may make Alice disclose $x = \log X$ so that K become recognizable. Second, we addressed in Section 3.4 that, there are group settings where it is possible that Bob is unable to compute yet capable of recognizing g^{ay} value. In that case the judge may not even need to know Alice's ephemeral secret x. Thirdly, unlike the previous examples, this attack is distinguished by the fact that deniability is affected by the algebraic properties of the underlying group.

In Section 3.4, we addressed a group setting in which MQV is provably non-deniable, yet the analysis remains inconclusive in a different group setting without an additional assumption or tool.

As we discussed throughout this work, there is a gap in the protocol settings, that enables such an incriminating method to be used. In other words, neither the specifications of the key derivation functions nor the algebraic group properties of the protocols we studied in this work makes such an attack impossible. Indeed, to the best of our knowledge, no well-known attribute of the hashes or groups is sufficient in and of itself to rule out the threat posed by this attack. Our approach is to describe this gap in terms of the hash and group structure

jointly.

In Section 3.5, we have identified the characteristics of the hash (used in the key computation) and the algebraic groups (for Diffie-Hellman operation) that enable such an attack to produce incriminating evidence against a protocol user. We then prove that this attack is the only barrier to the deniability for the iAKE protocols we examined. Following that we concluded that these iAKE protocols are resistant to this attack, and therefore deniable, provided that such characteristics do not exist in their configuration (i.e., in their choice of group and hash functions).

1.7 The Signal Communication Protocol

Signal protocol is running on a key exchange followed by a double ratcheting mechanism, which regularly injects fresh entropy to the keys to ensure forward secrecy. In Section 3.3, we showed that having a deniable KE is sufficient to claim deniability of the entire session that follows it.

Deniability was one of the main goals in the design of the Signal protocol [20], which has become the de-facto standard in the area of secure messaging protocols (it is now widely deployed including in protocols such as WhatsApp and Facebook messenger). Yet a formal proof of its deniability property has been lacking.

In fact, in [75, 74] it has already been shown that online deniability does not hold for Signal. On the other hand its offline deniability is widely believed to hold.

In this study, we address the reasons why a proof for the offline deniability of Signal has been difficult to construct and analyze ways to overcome this problem, offering the first formal study of the offline deniability of the Signal protocol.

For this, we focus on the offline deniability properties of a particular family of AKE protocols, namely, *implicitly authenticated* protocols (which, in particular, form the basis of

Signal, see below). These are characterized by the property that *the transcript of the protocol is independent of the private keys of the peers*. That is, anyone can generate the transcript messages. Authentication is provided *implicitly* by involving the peers' private keys in the derivation of the session key. Implicitly authenticated protocols seem to be perfectly suited to provide deniability. In their minimal form, all the peers exchange are Diffie-Hellman (DH) values $X = g^x, Y = g^y$ with the key being computed as a function of X, Y, the public keys of the parties and their corresponding private keys. There is little one can learn about the participants from these transcripts, hence, *intuitively,* they "must be deniable."

The above intuition, however, has been insufficient to prove these protocols deniable, in particular due to the need to simulate the session key. Thus, the question of deniability of implicitly authenticated protocols has not been settled to this day. This is not just a challenging theoretical question, but one of practical significance. Indeed, prime examples of this family are MQV [59, 54], HMQV [52], and 3DH [56] (see Section 3.1 for the description of these protocols). Implicitly authenticated protocols are particularly attractive due to their higher efficiency (since no additional signatures or encryptions are computed/sent together with the basic DH ephemeral keys). Very importantly, 3DH is the basis of the X3DH AKE underlying the Signal protocol and a main source of "intuition" regarding Signal's deniability properties.

1.8 Our Contribution

We make progress in the study of deniability of implicitly authenticated key exchange protocols, and Signal, in several ways:

• We demonstrate the insufficiency of implicit authentication as a property to ensure deniability. We present settings, in terms of properties of groups and assumptions, where the original MQV protocol [59] (that does not use a random oracle for key derivation) fails denia-

bility. We discuss how the counter-example built around MQV illuminates the difficulties one encounters in attempting to prove deniability for any of the other implicitly authenticated AKEs we consider, including 3DH.

- Using the above result, we are able to *characterize the non-deniability* of MQV in terms of the feasibility of the following problem: Given a random group element A, sample Y in G (with any efficiently-samplable distribution), so that it is hard to compute the Diffie-Hellman value of A and Y, denoted $DH(A, Y)$, even for the party sampling Y while it is easy (for anyone) to decide correctness of $DH(A, Y)$. We show that if such a Y can be feasibly sampled in G, then MQV is non-deniable over G and an adversary can *always prove* that he communicated with a particular honest peer.

- We show that in groups where the above condition does not hold (namely, there are efficient ways to sample Y, given A, so that it is infeasible to compute and decide $DH(A, Y)$), deniability holds for the studied protocols. Formally, we state a property, referred to as the *Knowledge of Diffie-Hellman (KDH)* assumption, so that HMQV (or MQV with random-oracle key derivation) and 3DH are deniable in groups where this assumption holds. While KDH is a strong assumption in the tradition of other knowledge assumptions, our treatment shows it to be necessary for formally proving deniability of these protocols.

- We show a connection between KDH and the more established Knowledge of Exponent assumption (KEA) [22, 5] via an additional, but more natural assumption we call *Knowledge of Discrete Logarithm (KDL)*. In particular, we get that deniability of the above protocols holds in groups where both KEA and KDL hold. It is an interesting question to find additional properties that imply KDH, hence implying deniability of the above protocols.

- To validate the definition of deniable AKE we prove a general theorem showing that any two-party protocol, whose transcript can be generated from a shared symmetric key and public information, is deniable (namely, simulatable without access to the shared key) if the

symmetric key is the product of a deniable AKE.

• As a corollary of the above theorem, we get a proof of deniability of the full Signal protocol under the assumption that its underlying AKE, 3DH, is deniable; in particular, this is the case under the KDH assumption.

• We analysed deniability of the protocols studied here when augmented with explicit authentication. We showed that HMQV-C (HMQV with confirmation) is deniable with respect to the initiator is achievable under the assumption KEA, which is weaker than the KDH assumption that is required for the plain HMQV.

• We examined the deniability properties of OAKE protocols by Yao et. al. [80], a family of implicitly authenticated Diffie-Hellman KE. We proved that without registration of long-term keys, the OAKE protocol is deniable under KDH, and T-OAKE is deniable under EKDH assumption.

1.9 Organization

In Chapter 2 we presented related work in the literature. In Section 3.1 we introduce some preliminaries, including the implicitly-authenticated key exchange protocols MQV, HMQV and 3DH. In Section 3.2 we present the definition of Deniable Authenticate Key Exchange and in Section 3.3 we prove how this notion extends to the deniability of communication sessions that use a key computed through a deniable AKE. In Section 3.4 we show an important negative result: A group setting where the MQV protocol is provably non-deniable, in particular showing that the intuition that implicitly authenticated protocols must be deniable does not hold. In Section 3.5 we use the MQV example from previous section to derive a general characterization of non-deniability. In Section 3.6 we present our main results regarding the provable deniability of the protocols we study, in particular introducing the KDH assumption that underlies these results. In Section 3.7 we show that the deniability of

3DH implies the deniability of X3DH and the full Signal protocol. In Section 3.8 we show how to remove the assumption of proof of possession at key registration that was used in some of the results in the paper. In Section 3.9 we present the KDL assumption and prove that together with KEA it implies KDH, hence basing our deniability results on these assumptions too. In Section 4.1 we present deniability analysis of OAKE protocol family due to [80] based on the KDH assumptions. In Section 4.2 we examine deniability properties of the iAKE protocol combined with a key confirmation step. In Section 4.3 we present attack scenarios demonstrating the impact of obfuscation and time-based cryptography on deniability. Finally, in Chapter 5 we conclude with the future research directions on the subject.

Chapter 2

Deniable Key Exchange in the Literature

Deniability as a privacy notion has been studied since the 1990s, with the phrase referred to also as 'plausible deniability' in the early works. Initial research concentrated on the concept of 'deniable authentication' and its application to authentication protocols. Dwork et al. [33] provided the first formal definition of deniable authentication, and among the early studies on the subject are [32, 49, 40].

Offline deniable key exchange was defined in [28] based on the work on deniable authentication in [33]. Definitions of deniable AKE that offer composability and online deniability were presented in [31, 78].

Informal discussions on deniable key exchange began to appear in [49, 12, 55, 13, 11, 26, 54], in which exploration of the concept is insightful and instructive in identifying the various degrees of deniability that KE protocols might achieve.

Borisov et al. [11] presented a prominent work whose major objective is to provide deniability and forward secrecy for instant messaging protocols. They developed a key exchange protocol based on Diffie-Hellman to enable *Off-the-Record* communication, hence

the name OTR protocol. As an instant messaging protocol, OTR begins with a synchronous key agreement protocol, in which the parties are required to be online during the process. It then moves on to a communication phase in which encrypted messages from two parties are exchanged. OTR authentication is based on two factors. Users hold identity (long-term) public keys and authenticate one another using signatures during the key agreement step. Messages are authenticated in the communication that follows the AKE using regularly refreshed MAC tags. OTR establishes deniability via two key mechanisms: signing random-looking values that are independent of the peer's identity during the key agreement phase and exposing the keys for timed-out MAC values during the communication session As explained in Section 3.3, disclosing MAC keys is not necessary for deniability if the prior AKE is deniable and the subsequent communication session does not involve identity of the communicating parties. Due to the inclusion of signatures in the transcript, OTR aims for at most partial deniability. The protocol contains significant security flaws, which were highlighted in [26] with suggested fixes (However, proposed fixes entailed a loss of deniability). Despite its weaknesses, OTR made a significant contribution to bringing deniability to the forefront of privacy concerns.

DiRaimondo et al. [28] introduced a formal definition of deniable key exchange based on the simulation paradigm, extending the definition of deniable authentication by [33]. This is the primary definition upon which we base our assessment of the offline deniability of KE protocols in this work. The authors presented formalized the partial deniability for KE, too, where the simulator is equipped with an additional oracle access. In order to evaluate partial deniability for Alice against Bob, the simulator can interact with Alice but in this interaction, the simulator must use a public key different than Bob's public key. In this way, Alice's interactions with other parties in the network can be shown to be useful for fabricating evidence of interaction with Bob. Obviously, such a strengthening is necessary for the simulator in order to provide the signatures in the simulation. Additionally, it retains

a useful privacy feature, as it does not reveal the identity of the interlocutor.

Yao et. al. presented a series of studies on key exchange protocols [79, 80, 82, 81], where deniability was one of the desired characteristics, along with efficiency as the major objective. As part of these studies, a new family of implicitly authenticated KE protocols, named OAKE family, were introduced.

The authors examined the deniability properties of OAKE family in terms of *honest player (HP-) deniability*, a new deniability definition for KE, introduced by the authors [80]. HP-deniability is weaker than the standard deniable key exchange definition by [28] and it considers a basic form of deniability, namely that if both peers are honest during the protocol run, they leave no trace of the communication. This notion considers the source of threat as an outsider, such as an eavesdropper on a session run or an active adversary conducting a man-in-the-middle attack to gather evidence that a session between two honest parties occurred.

A protocol failing to fulfil this level of deniability allows an outsider to obtain an (incriminating) evidence that A and/or B ran the protocol even when A and B acted honestly.

According to HP-deniability, the view (of the adversary) to be simulated is composed of the transcript, the session key and "all the pre-computed and stored session states" that the adversary can acquire using state-reveal and session-reveal queries [80]. As a result, the simulator of HP-deniability is able to compute easily any session value and the session key derived from ephemeral secrets x and y. In that respect, 3DH and X3DH satisfy the HP-deniability requirement, however MQV and HMQV do not due to the g^{ab} factor in the session key.

The evaluation of the OAKE family based on the standard offline deniability definition was an open problem. In Section 4.1, we provided an analysis of OAKE protocols on the basis of KDH assumptions.

Similar to HP-deniability, there are alternative, relaxed definitions of deniability that

work for less expansive purposes [35, 21]. These definitions are tailored to offer deniability for a particular type of information, for example, the protocol does not leave a provable trace for the content of a conversation while it allows to prove that a communication between A and B took place. Similar variations are available to deny source/destination of a conversation, the time of the protocol execution, the identity of the peer, and so forth. Typically, these relaxations grant the simulator access to some appropriately defined oracles, where the oracle represents the "deniability loss" with respect to the standard strict definition of simulatability.

Other tools to achieve deniability include designated verifier proofs [44] and ring signatures [60, 66]. However these primitives do not provide deniability in the strong sense because they incriminate both participants collectively. Designated verifier proofs refers to a proof that only one party can verify, ensuring that this proof cannot be passed on to anyone else. This property is achieved by use of an OR proof in [44]. For example, Alice proves to Bob that she knows a certain value x by proving the statement "either I know x or Bob knows his secret key." Such a proof is non-transferable, but it leaves an evidence that Alice and Bob were involved in the relevant protocol. Likewise, the ring signatures enable members of a group to sign a message without it being tracked back to the original signer. Again, the usage of a sign will incriminate both Alice and Bob, although no party should be according to the standard deniability definition.

Pass [61] stresses the differences between a deniability simulator and a zero-knowledge one (as defined in [37]). This work is significant for pointing out the conditions that should be avoided in deniability simulations, owing to the requirement that simulations be executable in real life. Pass showed that there are cases where ZK implies deniability in the standard model. However, once stepping out of the standard model, this implication is not guaranteed anymore, i.e., in CRS or ROM models. The meaning of a simulation created relying on the additional abilities granted by CRS or ROM is unclear in terms of deniability, since the

simulation is supposed to be a way of 'fabricating an evidence' under the conditions where a real protocol session is run. Therefore one should not use the simulation techniques such as programming a random oracle or choosing its own CRS value for deniability analyses.

Rewinding is another simulation technique that should be used with caution in deniability-related scenarios. The reasons, according to previous investigations in the literature, are twofold. A well-known issue with rewinding is with the concurrent executions of a protocol, rewinding may result in exceeding the probabilistic polynomial time bound on the simulation depending on the interleaving of sessions. Or, in the case where the adversary makes use of a third party service to choose its responses in the protocol, rewinding may not be sufficient to reset the adversary: the third party service may require to be reset, too, to control adversary's responses in the desired way.

Walfish [78] drew attention to a second aspect, which is directly related to deniability. Bob, a protocol participant, may utilize an online third-party service, such as a public bulletin board, where anyone may post data for public view and receive a unique value in return for that data. Bob can use this service to publish everything he receives from Alice during a session and then generate his protocol message using the unique value. Thus, Bob leaves a public and irreversible record of his interaction with Alice. The claim is that Bob may later prove his conversation with Alice to a judge by showing that his messages were generated specifically as a response to Alice's messages in the protocol. Also, rewinding Bob does not remove the traces of this session on the public board.

We believe that attack strategies of this type, carried out with the assistance of an online third party during the session, fall under the category of online deniability attacks. When a third party is present to witness the interaction, the meaning and implications of rewinding will be more involved and complex than they are in an offline context. However, this issue of avoiding rewinding due to the risk of leaving an irrefutable trace through an online party appears to be outside the scope of offline deniability. As a side note, none of the deniability

proofs in our work makes use of rewinding.

In [78], Walfish objected to the use of the random oracle model or the KEA assumption in deniability simulations, claiming that exploiting RO queries or extracting exponents (via KEA) inherently implies that only protocol participants are capable of performing these tasks, i.e., querying the random oracle or computing the exponentiations. He asserted that these tasks may be performed by a trusted external party, in which case the aforementioned benefits would not be realized. In this regard, we remark that our analysis does not make use of viewing RO queries of the adversary. A second point to highlight is that outsourcing some computations to an external party would not be practicable for all computations. It is reasonable to assume that Bob requested assistance from a friend in computing, for example, the exponent coefficient in HMQV, $d = RO(A, B, X, Y)$ which is computed using only public information. However, Bob is unlikely to pass the keying material for the session key to do the same, as this would compromise the security. The way we use the KDH extraction assumptions is similar to the latter example. Disclosing the value we extract would compromise the security of the protocol. (If Bob outsources computations at the cost of compromising his security, then this scenario will be pertinent to online deniability case.) Thus, prohibiting the use of extraction or RO in general would be excessively restrictive for the offline deniability and would not always correspond to real-world cases.

Dodis et al [31] (is part of the work presented in [78]) formalized the notion of online deniability for authentication protocols. This is a strong definition, which withstands the attacks involving a judge cooperating with the adversary during the protocol run, such as the attack mentioned above involving an online bulletin board.

They also showed that this notion is not possible to achieve in PKI model with adaptive corruption. In order to achieve online deniability along the entire communication, the authors suggest use of a key exchange, which satisfies a weaker notion named “deniability with incriminating abort” followed by a symmetric-key setting for conversation. Incrimination

occurs when one protocol participant proves that its peer has participated in the protocol. The guarantee that this new notion provides is that if the KE protocol is completed without abort, it is deniable, otherwise adversary obtains an unsimulatable information, which permits incriminating either party.

They provided a 4-round KE protocol satisfying this weaker notion against semi-adaptive corruption. Their primary tool to achieve this type of deniability is Dual Receiver Encryption used among two communicating parties.

Unger et al. [74, 75] presented a series of AKE protocols, with the primary motivation being a strong type of deniability that incorporates both offline and online forms. In [75], the authors demonstrate with an attack why Signal is not online deniable.

However, their interpretation of offline deniability is incorrect; they focus exclusively on transcript simulatability, omitting the requirement for session key simulatability. Therefore, they label 3DH (actually any iAKE protocol with the same protocol communication) as a deniable AKE with "unrestricted offline deniability," and X3DH as having partial offline deniability due to the use of signatures for storing ephemeral keys on the server. Section 3.4 of our work illustrates why such an incorrect use of the offline deniability definition results in inaccurate analysis.

Recent line of research on Signal [15, 14, 42, 30] aims to develop a post-quantum secure version of the X3DH protocol. This research focuses on abstracting X3DH and reconstructing it using standard post-quantum secure primitives. All of these works share the objective of developing a key exchange protocol with the same round complexity as X3DH, asynchronous communication eligibility, deniability, and forward secrecy.

Using post-quantum KEM and two-party ring signatures, Brendel et al. [14] constructed an AKE with the desired properties. The new protocol replaces DH operations in X3DH with KEM and additionally transmits a signed transcript using a ring signature for the two parties.

The deniability property that the resulting AKE satisfies is newly defined. The authors propose a game-based deniability definition against semi-honest adversaries: the definition places the judge against a challenge oracle in an indistinguishability game. The oracle is similar to a real-or-random oracle in spirit: based on a random challenge bit, the oracle either returns a real conversation (transcript and session key) between two *honest* peers or returns a simulation created using all public information and the secret key of one of the participants. (Simulation requires the secret key of one party.)

This defines a weak form of deniability since protocol participants are required to follow the protocol honestly. Therefore, the definition does not encompass the threat that posed by a malicious peer, which is the primary attack strategy we focused on in our work: that an adversary can break from the protocol to frame their peer. As this definition aims for a basic form of deniability, the proof of deniability for their protocol doesn't require any non-standard extraction assumptions. A stronger aspect of the definition is that the judge is equipped with the private and public keys of all protocol users.

With the same goal of building a post-quantum secure version of Signal, Hashimoto et. al. [42] independently presented a key exchange protocol around the same time period, which was also built on KEM and ring signatures (the second protocol in the article). This protocol and the one from Brendel et. al.[14] are quite similar in design. The primary distinction is that the former one is masking the signature with a pseudorandom value rather than transmitting it in the clear in the round message.

Hashimoto et al. analyzed the deniability of their protocol in accordance with the definition by DiRaimondo et al. [28] with a slight modification: they omitted the auxiliary information from the judge's and participants' view. The authors demonstrated that the protocol is only deniable in this sense against semi-honest adversaries, and that the proof requires no knowledge extraction assumptions. To achieve deniability against malicious adversaries, the authors added an NIZK proof for well-formedness of the signature keys: an adversary must

now generate the signature verification key honestly and demonstrate knowledge of the corresponding secret key via NIZK proof. This modification achieves deniability against malicious adversaries under the assumption that the KEMs satisfy plaintext-awareness property.

An interesting point raised in this work is that the modified protocol (which is deniable against malicious adversaries) is not known to be secure against quantum adversaries. This is due to the fact that the knowledge extractor that comes with plaintext-awareness is required to re-run the adversary's algorithm fixing the randomness fed to it. In the quantum setting, invoking an algorithm fixing the randomness is not well-defined, nor is rewinding an algorithm (without disturbing the randomness). Consequently, the presented proof only applies to classical adversaries.

Dobson and Galbraith [30] built a post-quantum candidate for X3DH based on the Supersingular Isogeny Diffie Hellman (SIDH) problem. The authors demonstrated the deniability of their protocol based on the KDH assumptions presented in our study, claiming that the KDH family extends to SIDH problem setting.

Gunn et al. [38] drew attention to the interaction between attestation and deniability in their study. The authors showed that trusted execution environments (TEE) supporting remote attestation (RA) can be used to circumvent the deniability of any authenticated protocol as well as to restore it. This mechanism of TEE and RA is a combination of hardware and software augmented to a device in order to execute critical codes securely isolated from other software on the same device. The system provides a log of all operations that are performed. The RA then generates records of the executed operations that are signed using the hardware vendor's signature key (which is hard-coded to the device). These statements can then be verified by any party who receives the signature verification keys from the vendor. The attested statements could take the form of "The protocol X output Y" or "Bobs private ephemeral key y was protected by the TEE so that he could not forge messages falsely claiming to be from Alice."

The following example illustrates Gunn et al.'s primary attack against deniability. Assuming that both Alice and Bob are using a deniable messaging protocol, Bob is using a device that is equipped with a TEE, supporting RA. An adversary contacts Bob, receives attestations, and verifies them using the verification keys of the hardware vendor. Through the trust placed in the vendor, the adversary becomes convinced that the protocol on Bob's machine exchanges messages with Alice.

There are several essential points to emphasize about the attack. First, the entire attestation mechanism is executed on Bob's device over the course of his conversation with the adversary. Consequently, Alice is unable to detect this process. Second, the adversary is convinced even if it does not trust Bob. The trust placed in the TEE vendor. Third, the generated output by the TEE and RA service is transferable. Finally, the attack is applicable to any protocol run on the TEE.

This attack goes in parallel to the bulletin board attack described above from [31]. A similar attack strategy for keeping indisputable records of round messages at the time of the session was also discussed in [75] (Section A.2). The use of a trusted third-party service to provide publicly or selectively verifiable records to circumvent offline deniability is a common element of these attacks. (We also presented comparable attack strategies in Section 4.3.) As stated previously, we consider these types of attacks to fall under the category of online deniability attacks. The reader can find online deniable AKE protocols that can resist these attacks in [31, 78, 75].

A full specification of the Signal protocol can be found in [73]. As mentioned above it uses the Extended Triple Diffie-Hellman (X3DH) key agreement protocol [57] (built on the 3DH AKE [56]) followed by a communication session which include a *ratcheting* mechanism to refresh keys [62]. Security analyses of these protocols that do not include deniability can be found in [20, 1].

Chapter 3

Deniability Analysis of iAKE Protocols

3.1 Implicitly Authenticated Key Exchange

In this section, we recall two examples of implicitly authenticated key exchange protocols, namely HMQV [52] (and its predecessor MQV [59, 54]) and 3DH as used in Signal [56].

3.1.1 Preliminaries

In the following, we denote with G a cyclic group of prime order q generated by g. For every element $X \in G$ there exist an integer $x \in \mathbb{Z}_q$ such that $X = g^x$. We say that x is the discrete log of X with respect to g and denote it with $x = \log_g X$. Given two elements $A = g^a$ and $B = g^b$ we denote with $DH(A, B) = g^{ab} = A^b = B^a$, the Diffie-Hellman transform of A, B, [29].

With $a \leftarrow S$ we denote the process of sampling a uniformly at random in the set S.

The following definition states that computing the discrete log is hard.

Definition 3.1.1. Let G be a cyclic group of prime order q generated by g. We say that

the (T, ϵ) Discrete Log Assumption holds in G if for every probabilistic Turing Machine $\mathcal{A}$ running in time T we have that

$$Prob[x \leftarrow \mathbb{Z}_q \; ; \; \mathcal{A}(g^x) = x] \leq \epsilon$$

The following definition states that computing the Diffie-Hellman transform is hard.

Definition 3.1.2. Let G be a cyclic group of prime order q generated by g. We say that the (T, ϵ) Computational Diffie-Hellman (CDH) Assumption holds in G if for every probabilistic Turing Machine $\mathcal{A}$ running in time T we have that

$$Prob[A, B \leftarrow G \; ; \; \mathcal{A}(A, B) = DH(A, B)] \leq \epsilon$$

Consider the set $G^3 = G \times G \times G$ and the following two probability distributions over it:

$$\mathcal{R}_G = \{(g^a, g^b, g^c) \text{ for } a, b, c \leftarrow [0..q]\}$$

and

$$\mathcal{DH}_G = \{(g^a, g^b, g^{ab}) \text{ for } a, b, \leftarrow [0..q]\}$$

We use these distributions in the following definition:

Definition 3.1.3. We say that the (T, ϵ) Decisional Diffie-Hellman (DDH) Assumption holds over $G = \langle g \rangle$ if the two distributions $\mathcal{R}_G$ and $\mathcal{DH}_G$ are (T, ϵ)-indistinguishable.

For Definition 3.1.3, we use Goldwasser and Micali's classical definition of computational indistinguishability [36].

Definition 3.1.4. Let $\mathcal{X}, \mathcal{Y}$ be two probability distributions over A. Given a circuit D, the

distinguisher, consider the following quantities

$$\delta_{D,\mathcal{X}} = Prob_{x\in\mathcal{X}}[D(x) = 1]$$

$$\delta_{D,\mathcal{Y}} = Prob_{y\in\mathcal{Y}}[D(y) = 1]$$

We say that the probability distributions $\mathcal{X}$ and $\mathcal{Y}$ are (T, ϵ)-*indistinguishable* if for every probabilistic Turing Machine D running in time T we have that $|\delta_{D,\mathcal{X}} - \delta_{D,\mathcal{Y}}| \leq \epsilon$.

3.1.2 MQV and HMQV Protocols

The MQV protocol was introduced in [59] and further specified in [54]. A formal analysis of its security properties was presented by Krawczyk in [52] where he also presented an improved version called HMQV to address some of MQV's weaknesses uncovered by the analysis. In Figure 3.1, we describe both protocols.

Note that in MQV the session key is defined as the group element $\tilde{K}$ (with the use of a hash function left as optional), while HMQV mandates the use of a hash function H to derive the session key from the secret value $\tilde{K}$ shared by Alice and Bob.

3.1.3 Triple Diffie-Hellman

The Triple Diffie-Hellman (3DH) protocol creates a shared secret key between two parties who authenticate each other via public keys [56], with deniability being one of its claimed features (but never formally proven until now).

3DH is the key exchange that underlies secure communication in the Signal messaging application. We postpone discussions about how 3DH is used inside Signal to Section 3.7. Here, we simply describe 3DH as a basic key exchange protocol where both parties are alive and communicating with each other.

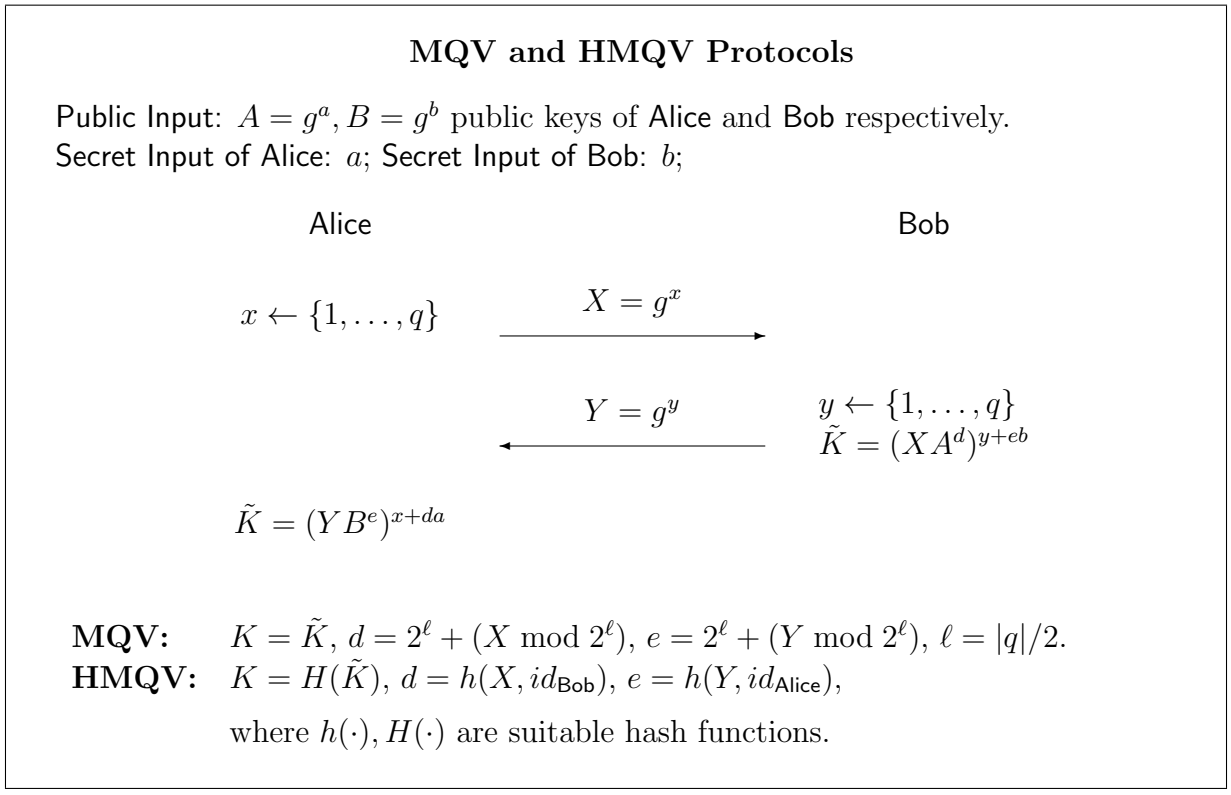

Figure 3.1: Computation of the session key K in MQV and HMQV protocols

Figure 3.2 describes the protocol. As for the case of MQV Alice and Bob have long-term public keys $A = g^a, B = g^b$ and exchange ephemeral public keys X, Y. What changes is how the session key is computed. The "three Diffie-Hellman" part refers to three separate Diffie-Hellman operators concatenated together and then passed to a hash function to derive a key.

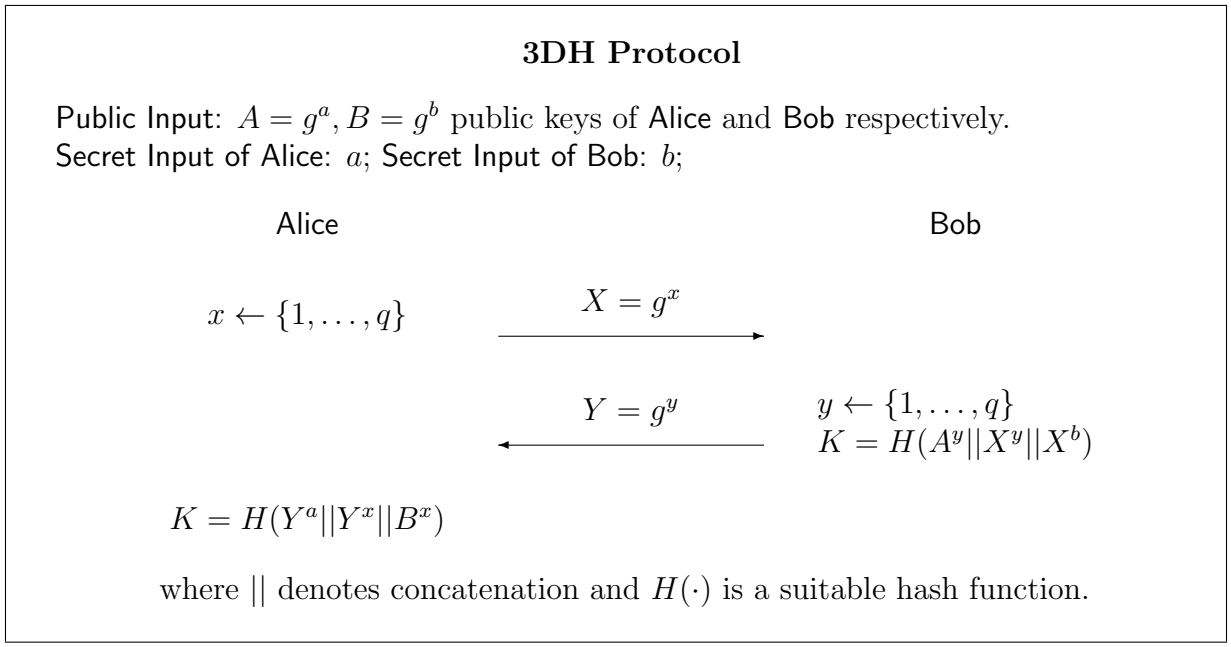

Figure 3.2: Triple Diffie-Hellman key exchange protocol

3.1.4 Key Registration

The protocols described above assume that participants have long-term public keys that are associated to their identity via some form of Public Key Infrastructure (PKI). Certification authorities often require that at the time of *key registration*, participants prove knowledge of their secret key. If this is done via an extractable proof of knowledge, e.g., via a Schnorr proof [68], proving deniability can be simplified by assuming the simulator can extract the private key of the participants. Some of our proofs (e.g., for Theorems 3.6.2, 3.6.4 and 3.7.2) are simplified by assuming such a "Key Registration" setting. However, we later show

(Section 3.8) that in all these cases, a slight strengthening of our assumptions gets rid of the need to assume private-key extractability.

3.2 Deniable Key Exchange

We recall the definition of deniable Authenticated Key Exchange (AKE in the rest) from [28, 31, 78].

An AKE protocol works with two parties, A and B, who want to agree on a secret key K. Each party has a long-term public/secret key pair which is associated to the party through a trusted registry. These key pairs are generated via a key generation phase. For notation purposes, A has public key pk_A and secret key sk_A – B has pk_B and sk_B.

One of A and B acts as the initiator and the other acts as the responder. The protocol results in session key K. Informally, security for AKE requires that if an honest party A outputs key K and associates it to an honest party B, then no party other than B may know anything about K. Additional security properties can be enforced, such as perfect forward secrecy (past session keys remain secure, even if long-term keys are compromised), security against state compromise (ephemeral values are revealed to the adversary), etc. A formal treatment of AKE security can be found in [6, 71, 17, 19].

Informally we say that an AKE is *deniable* if it prevents A or B from proving to a third party (which we will call the *judge*) that an AKE occurred with a particular party. A weaker goal is to be able to deny just the contents of communication protected by the AKE's resulting key.

Recall that a KE protocol involves two communicating parties: the initiator, who sends the first message, and the responder. Let Σ denote an AKE protocol with key generation algorithm KG and interactive machines Σ_I and Σ_R, which respectively denote the roles of the initiator and responder. Both Σ_I and Σ_R take as input their own secret and public keys.

In most cases, they also take in the identity and public key of their peer, but other AKE protocols specify that the parties learn this information during the session [18]. The term *session* denotes an individual run of a KE protocol. When the protocol finishes, it outputs either an error or a session key.

ADVERSARY. Let $\mathcal{M}$ denote an adversary that runs on an arbitrary number of randomly chosen public keys $\vec{\mathsf{pk}} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_l)$ generated by KG. The algorithm associates the keys to honest users in the network. $\mathcal{M}$'s input also includes some arbitrary auxiliary information $aux \in AUX$. The adversary runs Σ with an arbitrary number of honest users. Sometimes $\mathcal{M}$ acts as the initiator, and other times $\mathcal{M}$ acts as the responder. The individual sessions run in a concurrent setting, so $\mathcal{M}$ may schedule and interleave them arbitrarily.

VIEW. We define the view of the adversary $\mathcal{M}$ as its internal coin tosses, the transcript of the entire interaction, and the session keys from each session that $\mathcal{M}$ participates either as an initiator or responder. Sessions that do not produce keys result in a key defined by an error value. We denote this view by $\mathsf{View}_{\mathcal{M}}(\vec{\mathsf{pk}}, aux)$.

SIMULATOR. In order to demonstrate deniability with respect to initiator (*resp.*, responder), the simulator takes the role of the initiator I (*resp.*, responder R) and imitates I (*resp.*, R) without having the long-term secret key sk_I (*resp.*, sk_R).

As input, the simulator receives some random coins, the public keys $\vec{\mathsf{pk}}$ of all parties and any auxiliary input aux available to the adversary. It generates $\mathsf{Sim}_{\mathcal{M}}(\vec{\mathsf{pk}}, aux)$ by interacting with the adversary $\mathcal{M}$ as its peer. $\mathsf{Sim}_{\mathcal{M}}(\vec{\mathsf{pk}}, aux)$ includes the transcript and the resulting shared key of the session.

The simulator provides the inputs to the adversary $\mathcal{M}$ prior to the protocol execution and observes all communication $\mathcal{M}$ has with its environment (such as AKE sessions $\mathcal{M}$ holds with other honest parties and the random oracle queries). The random oracle (RO) queries made by the adversary are visible to the simulator. However, the simulator might

not be able to freely tamper with the RO input-output pairs (program the RO), because the judge is granted access to the random oracles, too. Therefore the RO queries involved in the simulation are expected to be consistent with the possible queries made by the judge and other honest parties running a session with the adversary.

Definition 3.2.1 ([28])**.** An AKE protocol $(\mathsf{KG}, \Sigma_I, \Sigma_R)$ is $(T_{\mathcal{M}}, T_{\mathcal{S}}, T_{\mathcal{D}}, \epsilon)$ *concurrently deniable* with respect to the class of auxiliary inputs AUX if for any adversary $\mathcal{M}$ running in time $T_{\mathcal{M}}$, on input honest parties' public keys $\vec{\mathsf{pk}} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_l)$ generated by KG and any auxiliary input $aux \in AUX$, there exists a simulator SIM running in time $T_{\mathcal{S}}$ on the same inputs as $\mathcal{M}$ which produces a simulated view $\mathcal{S}im(\mathsf{pk}, aux)$ such that the following two distributions are $(T_{\mathcal{D}}, \epsilon)$-indistinguishable:

$$\mathcal{R}eal(aux) = [(\mathsf{sk}, \mathsf{pk})_{I,R} \leftarrow \mathsf{KG}; (aux, \vec{\mathsf{pk}}, \mathsf{View}_{\mathcal{M}}(\vec{\mathsf{pk}}, aux))]$$
$$\mathcal{S}im(aux) = [(\mathsf{sk}, \mathsf{pk})_{I,R} \leftarrow \mathsf{KG}; (aux, \vec{\mathsf{pk}}, \mathsf{Sim}_{\mathcal{M}}(\vec{\mathsf{pk}}, aux))]$$

The definition follows the usual simulation paradigm which guarantees that the judge cannot decide if anything that the adversary presents (the view) is the result of an interaction with a real party or the product of a simulation. As pointed out by Pass in [61], the simulation requirements for deniability are stronger than for Zero-Knowledge simulation as the simulation is not just a "thought experiment" but it needs to run in the real world; for example, random oracle programmability is not allowed since the distinguisher (the judge in the deniability setting) has access to a (real-world) pre-defined hash function. The same holds for trapdoored common reference strings.

Why is the session key included in the view. We are interested in the deniability of the full communication protected by the session key, not just deniability of the key exchange run. Limiting deniability to the key exchange transcript only, would allow for situations where Bob could not prove Alice's participation in a key exchange session but could do so

once the session key is used (we show such an example in the context of the non-deniability of MQV in Section 3.4.1). Fortunately, by simply including the session key in the view that the deniability simulator needs to produce, one guarantees that *any* application using the session key (and otherwise public information) will be as deniable as the key exchange itself (namely, the joint transcript of the key exchange session and the ensuing application can be simulated). This important consequence of the above definition is formalized and proven in Theorem 3.3.2.

3.3 Deniable Sessions

As we noted above the definition of deniability of an AKE explicitly includes the session key in the view in order for us to claim that any deniability is preserved in any subsequent use of the key in the session that follows the AKE. We now formally prove this statement[1]. First we define deniability for an arbitrary interactive protocol between two parties, and then we show that any communication structured as an AKE, followed by messages where the two parties only use the session key (but not their long-term secret keys) is deniable.

Consider an adversary $\mathcal{M}$ that on input $\vec{\mathsf{pk}}$ interacts with the parties holding the public keys. The adversary also may have auxiliary input aux drawn from distribution AUX.

$\mathcal{M}$ initiates several concurrent interactions with the parties and we define the adversary's *view* of the interaction as $\mathcal{M}$'s coin tosses together with the transcript of the full interactions. We denote the view as $\mathsf{View}(\vec{\mathsf{pk}}, aux)$.

Definition 3.3.1. We say that an interactive protocol $\mathcal{P}$ $(\mathsf{KG}, \mathsf{I}, \mathsf{R})$ is $(T_{\mathcal{M}}, T_{\mathcal{S}}, T_{\mathcal{D}}, \epsilon)$-*concurrently deniable* with respect to the class of auxiliary inputs AUX if for any adversary $\mathcal{M}$ running in time $T_{\mathcal{M}}$, on input honest parties' public keys $\vec{\mathsf{pk}} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_l)$ generated by KG and

[1]This was claimed informally and without proof in [28]; here, we use this result in essential way in Section 3.7 to show how deniability of 3DH carries to deniability of the whole Signal protocol.

any auxiliary input $aux \in AUX$, there exists a simulator $SIM_{\mathcal{M}}$ running in time $T_{\mathcal{S}}$ on the same inputs as $\mathcal{M}$, such that the following two distributions are $(T_{\mathcal{D}}, \epsilon)$-indistinguishable

$$\mathcal{R}eal(aux) = [(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathsf{KG}; (aux, \vec{\mathsf{pk}}, \mathsf{View}(\vec{\mathsf{pk}}, aux))]$$
$$\mathcal{S}im(aux) = [(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathsf{KG}; (aux, \vec{\mathsf{pk}}, SIM(\vec{\mathsf{pk}}, aux))]$$

We now define a *session*. Consider two parties Alice and Bob with associated public keys pk_A, pk_B. They also hold private keys sk_A, sk_B respectively, and also additional inputs x_A, x_B. We say that an interactive protocol P between Alice and Bob is a *session* if

- $P = [P_1, P_2]$ where P_1 is an AKE. Let K be the session key resulting from the execution of P_1
- every message sent by Alice (resp. Bob) in P_2 is a function only of the transcript so far, the private input x_A (resp. x_B), and the session key K, but *not* of the private keys sk_A (resp. sk_B)

Theorem 3.3.2. *Let $P = [P_1, P_2]$ be a session, where P_1 is a deniable authenticated key exchange according to Definition 3.2.1, which includes the session key in the view. Then P is deniable according to Definition 3.3.1.*

Proof. Based on Definition 3.3.1, a deniability simulator for P is required to simulate the transcript between the parties only, i.e. $\vec{\mathsf{tr}}_{P_1}$ and $\vec{\mathsf{tr}}_{P_2}$, which denote the transcript of P_1 and P_2, respectively.

Since P_1 is $(T_{\mathcal{M}_1}, T_{\mathcal{S}_1}, T_{\mathcal{D}_1}, \epsilon_1)$-deniable, we know that for any adversary $\mathcal{M}_1$ running in time $T_{\mathcal{M}_1}$, there exists a simulator running in $T_{\mathcal{S}_1}$ which outputs a simulation of transcript and session key: $\{\vec{\mathsf{tr}}^*_{P_1}, K^*\}$.

For all distinguishers $\mathcal{D}_1$ running in time $T_{\mathcal{D}_1}$,

$$|Pr[\mathcal{D}_1(\vec{\mathsf{pk}}, \vec{\mathsf{tr}}^*_{P_1}, K^*, coins_{\mathcal{M}_1}) = 1] - Pr[\mathcal{D}_1(\vec{\mathsf{pk}}, \vec{\mathsf{tr}}_{P_1}, K, coins_{\mathcal{M}_1}) = 1]| \leq \epsilon_1.$$

Assume, for contradiction, that P is not deniable. So there is an adversary $\mathcal{M}$ for protocol P running in time $T_{\mathcal{M}}$ such that for every simulator $\mathcal{S}$ that runs in $T_{\mathcal{S}}$ and outputs $\vec{\mathsf{tr}}^*_P$, there exists a distinguisher $\mathcal{D}$ running in $T_{\mathcal{D}}$ such that

$$|Pr[\mathcal{D}(\vec{\mathsf{pk}}, \vec{\mathsf{tr}}^*_P, coins_{\mathcal{M}}) = 1] - Pr[\mathcal{D}(\vec{\mathsf{pk}}, \vec{\mathsf{tr}}_P, coins_{\mathcal{M}}) = 1]| > \epsilon.$$

Let us run $\mathcal{M}$ on P. This induces an adversary $\mathcal{M}_1$ on P_1 for which we should have a "good" simulator $\mathcal{S}_1$ which outputs a simulated transcript and session key $\{\vec{\mathsf{tr}}^*_{P_1}, K^*\}$.

We now "extend" $\mathcal{S}_1$ to a full simulator $\mathcal{S}$ for P. Using the simulated session key K^*, the simulator $\mathcal{S}$ will simulate the honest party's message: recall that those messages are a function of only the transcript so far, the additional inputs x_A (or x_B) and the session key[2]. For this simulator we have a distinguisher $\mathcal{D}$ which distinguishes the simulated transcript from the real one.

We now can build a distinguisher $\mathcal{D}_1$ for $\mathcal{S}_1$. Given $\{\vec{\mathsf{pk}}, \vec{\mathsf{tr}}_{P_1}, K, coins_{\mathcal{M}_1}\}$ it needs to decide if it is the real view of the AKE P_1 or the output of $\mathcal{S}_1$.

The first thing that $\mathcal{D}_1$ does is to extend these view to a full view for P, the same way in which $\mathcal{S}$ does it. Note that if $\mathcal{D}_1$ runs on input the real view of P_1 then it obtains the real view of P. But if $\mathcal{D}_1$ runs on input the simulated view of P_1 created by $\mathcal{S}_1$ then it obtains the simulated view of $\mathcal{S}$. Therefore it can call $\mathcal{D}$ on its input and distinguish with the same

[2] Here, we assume that the auxiliary inputs x_A and x_B are not tied to the identity of the party (as opposed to sk_A and sk_B) and therefore can be given to the simulator.

probability that $\mathcal{D}$ does.

3.4 Negative examples

In this section, we are going to examine the difficulty in simulating an execution of implicitly authenticated key exchange protocols such as MQV, HMQV and 3DH. We will show a strategy that (on certain groups) allows an adversary to prove that an interaction took place. This negative result will then point the way to what type of assumption about the underlying group we need to make to guarantee deniability in a provable way.

We focus on MQV, but the issues we raise here will lead to an understanding of deniability conditions for HMQV and 3DH. Consider the MQV protocol in Figure 3.1 and let us try to prove that the protocol is deniable for Alice. In other words we need to construct a simulator SIM who plays the role of Alice while talking to Bob. SIM is given the public key of Alice, $A = g^a$, but not the corresponding secret key, a.

SIM runs Bob to simulate the conversation and observes all of Bob's communication in his environment. SIM starts by providing the random coins r, and, if available, the auxiliary information to Bob. SIM then chooses a random x and sends out $X = g^x$ to Bob. In return it receives a group member $Y \in G$ from Bob. SIM's final output must be indistinguishable from Bob's view (r, X, Y, K) and the only thing that SIM does not know is K. Recall that in MQV

$$K = g^{xy} g^{ayd} g^{xbe} g^{abde}$$

where e, d are values computed from the transcript.

When Bob is honestly executing the protocol, the simulator is easy to construct. If Bob follows the protocol and computes Y as g^y for $y \leftarrow \mathbb{Z}_q$, the simulator can do exactly the same thing and compute the session key $K = g^{xy} A^{yd} (X A^d)^{be}$. Here, we assume that the

simulator gets Bob's private key[3].

However, a malicious Bob can deviate from the protocol at will and having Bob's random coins provides SIM no information about how Y is actually sampled. In the simulation above, SIM can compute two of the DH values $g^{xy} = Y^x$ and $(XA^d)^{be}$ since b and x are known. But $DH(A, Y) = g^{ay}$ cannot be computed because neither a (secret key of Alice) nor $y = \log_g Y$ is known to SIM (maybe not even to Bob).

The only option for SIM would be putting a random string as the simulated key, hoping that a random value is indistinguishable from the actual key. Such strategy would work if we could invoke the DDH assumption to claim the two distributions are indistinguishable. However, a random string does not necessarily substitute for g^{ay}, because though a is uniformly selected, DDH does not apply for an adversarially chosen Y.

3.4.1 When MQV is provably *non-deniable*

The discussion above shows that an adversarially chosen Y is a barrier to prove deniability for MQV. We now prove that over some groups, it is actually impossible to prove that MQV is deniable, because there is a strategy for Bob to prove that he interacted with Alice.

Assume we are running the protocol over a cyclic group G setting where:

1. The DDH problem is easy

2. The following experiment succeeds with negligible probability for every efficient adversary Adv and any efficiently samplable distribution $\mathcal{Y}$

   - Adv runs on input $A, B \in G$ chosen uniformly at random and outputs $X \in G$
   - Adv receives $Y \in G$ chosen according to $\mathcal{Y}$

[3] For showing the failure of (proofs of) deniability, assuming the simulator gets b makes our negative result stronger as it implies that *even if* we allow key registration we do not know how to simulate, and in some cases simulation is actually impossible.

- *Adv* succeeds if it outputs $K_A = (XA^d)^y(XA^d)^{be} = DH(XA^d, Y)DH(XA^d, B^e)$ where d, e are defined as in the MQV protocol

We note that (2) follows from the KCI security of MQV, namely, the values x and b do not suffice to compute the session key. Point (1) holds, for example, if G is a bilinear group.

On assuming that the DDH problem is easy. Before we proceed with our counter-example, let us discuss our assumption that the DDH problem is easy in G. In this case, the security of MQV cannot be proven in the sense of the session key being indistinguishable from a random group element. Does this mean that our counter-example shows non-deniability of a protocol that is insecure as a key-exchange scheme? Is there value in doing so? There are several answers to this point:

- First, one can consider a weaker security notion for key exchange where the goal is for the session key to be unpredictable. We illustrate the utility of such notion in the "bearer token" example below. Furthermore, an unpredictable key with sufficient (computational) entropy, but not necessarily indistinguishable from random, can be converted into a strong key using a randomness extractor. MQV could conceivably satisfy such property, be secure as a key exchange, and still be non-deniable.

- Deniability is an orthogonal property to that of security. Our counter-example is designed to illustrate the difficulties of proving deniability for MQV and similar protocols, and demonstrating the *failure of the intuition* that the sole lack of explicit authentication methods (such as digital signatures) is sufficient to assume that deniability holds.

- Additionally, there are so-called *trapdoor DDH* groups [24, 69], where the DDH problem is conjectured to be hard unless one is in possession of a trapdoor. In this case, the protocol is secure for anybody who does not possess the trapdoor but non-deniable for a judge who holds the trapdoor.

The counter-example (incriminating Alice). We show a strategy that incriminates Alice over groups where the DDH problem is easy. A malicious Bob samples Y uniformly at random in the group but in a way in which he can demonstrate that he does not know $y = \log_g Y$, for example by hashing a publicly known string (e.g. today's copy of the NY Times) into a group element via a sufficiently complicated hash function (which could be modeled as a random oracle[4]).

We now prove by contradiction that there cannot be a simulator. If there is a simulator SIM, let K_S be the key provided by the simulator, while $K = (XA^d)^y(XA^e)^b = DH(XA^d, Y)DH(XA^e, B)$ is the real key. We assume that Bob is willing to reveal b to the judge in order to prove that an interaction took place.

The knowledge of b allows the judge to compute $z = DH(XA^d, B^e) = (XA^d)^{be}$. Since the DDH is easy, the judge can decide if $K = K_S$ by checking if $K_S \cdot z^{-1} = DH(XA^d, Y)$. Therefore, anything other than the authentic key is detected by the judge. So the only possible successful simulator is the one that outputs $K_S = K$. But such simulator contradicts assumption (2) above and the security of MQV.

So all that Bob needs to do to be able to prove Alice communicated with him is to choose a value $Y = g^y$ for which Bob could not possibly know y (as described above) and obtain the (unhashed) MQV key computed by Alice on the quadruple (A, X, B, Y). As an example of an application that would disclose this value to Bob (without Bob being able to compute it by himself), consider a key exchange protocol whose session key is used as a bearer token (cf., RFC 6750 [46]) that a user needs to present for obtaining access to some controlled resource (e.g., a non-public webpage, a printing service, etc.). Here, the user Alice would run MQV with the server Bob to obtain the bearer token in the form of an MQV key which Alice later submits to Bob for gaining access to the resource. Server Bob could choose Y

[4] It is not necessary to model this hash as a random oracle, as long as we assume that computing g^{ay} is hard when $A = g^a$ is sampled uniformly at random and $Y = g^y$ is sampled according to the procedure used by Bob.

as above, receive the token (=key) from Alice and use this key to prove she communicated with him (note that the computation of the key is specific to Bob's public key thus proving Alice's intention to have this communication with Bob). In other words, this example shows that the non-deniability of MQV can be actually exploited in practice. We further note that the above bearer-token application illustrates how a key exchange like MQV that outputs an unpredictable value (rather than a key that is indistinguishable from a random string) can have significant value in practice.

3DH without hashing. It is not hard to see that a similar reasoning applies to an "unhashed" version of 3DH where the session key is set as $K = DH(A, Y)||DH(X, Y)||DH(X, B)$. Therefore such a version of 3DH would also be provably non-deniable under the above conditions.

3.4.2 Does the random oracle help?

In HMQV and 3DH the session key is computed by hashing the secret shared value, i.e.

$$K = H[DH(XA^d, Y)DH(XA^d, B^e)]$$

in HMQV and

$$K = H[DH(A, Y)||DH(X, Y)||DH(X, B)]$$

in 3DH. If we model H as a random oracle, would this help in solving the problems described in the previous section?

The question is still how can the simulator provide a session key which is indistinguishable from the real one. In this case, one would hope that the use of the random oracle will allow the simulator to identify the key. Assume Bob is the malicious party and can deviate from the honest execution of the protocol. Every time Bob makes a random oracle query, SIM sees

it (even though it is not allowed to choose the answer for it [61]). In particular, if Bob computes the real session key K in HMQV that matches A, B, X, Y, then he must have queried $DH(XA^d, Y)DH(XA^d, B^e)$ (resp. $DH(A, Y)||DH(X, Y)||DH(X, B)$ for 3DH) to the random oracle.

Note that even if Bob queries the RO on these values, it is not clear how the simulator can identify the correct query that corresponds to the computation of the session key. Indeed, the simulator SIM is able to compute g^{bx} and g^{xy}, but cannot compute g^{ay} since a and y are not known. If Y is uniformly distributed and the DDH holds, SIM cannot *provably* detect which query corresponds to the session key[5].

The only option for the simulator is to choose a random value as the key, but this is distinguishable from the real view if Bob presents to the judge the correct input to the random oracle (e.g. Bob knows $y = \log_g Y$ and can convince the judge that the session key was computed using the correct input).

Note that if this is the case, Bob still cannot convince the judge that he spoke to Alice. In fact if Bob knows y, then the entire transcript could be his own creation without ever interacting with Alice.

In other words, we have one of two possible cases: either Bob does not know y (and the input to the random oracle) in which case the simulator should be able to put a random key in the view, or Bob knows y (and the correct input to the random oracle) in which case the simulator should be able to identify the correct key from the knowledge of Bob. The problem is that we do not know which case we are in, and therefore we cannot complete the simulation successfully.

The way out of this problem is described in the next section and relies on an appropriate

[5] One could use a Gap-DDH Assumption, which states that the CDH Assumption holds even in the presence of an oracle that decides the DDH. Then such oracle could be provided to the simulator to detect the query. Yet this simulator would not be a legitimate *deniability* simulator unless the oracle could be implemented in real-life.

"knowledge extraction" from Bob, which will address also the issues related to the counter-example from Section 3.4.1.

3.5 A characterization for non-deniability

In this section, we show that the sampling strategy shown above to make MQV non-deniable is essentially the *only* strategy that can achieve so. That is, we prove that if an adversary is able to "frame" one of the parties in the MQV protocol and prove that an interaction took place, then we have a way to sample a group element Y in G in a way that it is hard to compute $DH(A, Y)$ for a fixed group element A but it is easy to detect that $DH(A, Y)$ is correct.

The consequence is that if we assume that such a task is computationally infeasible then we can conclude (albeit non-constructively, see below) that the MQV protocol is deniable. Details follows.

Non-deniable AKE. First we define what a non-deniable or *incriminating* AKE is, as the logical negation of deniability.

We call a key exchange protocol (KG, I, R) as $(T_{\mathcal{M}}, T_{\mathsf{SIM}}, T_{\mathsf{J}}, \varepsilon_{\mathsf{J}})$-*incriminating* if there is an adversary $\mathcal{M}$ running in time $T_{\mathcal{M}}$ such that for all simulators SIM running in time T_{SIM}, there exists a judge J running in time T_{J} which distinguishes the uniformly selected samples from the following distributions with probability at least ε_{J}.

$$\mathcal{R}eal = \{\mathsf{View}_{\mathcal{M}}(pk_i)\}_{(sk_i,pk_i)\leftarrow KG}$$

$$\mathcal{S}im = \{\mathsf{SIM}_{\mathcal{M}}(pk_i)\}_{(sk_i,pk_i)\leftarrow KG}$$

$$|Pr_{x\in\mathcal{R}eal}[\mathsf{J}(x) = 1] - Pr_{x\in\mathcal{S}im}[\mathsf{J}(x) = 1]| \geq \varepsilon_{\mathsf{J}}$$

$\mathsf{View}_{\mathcal{M}}$ includes the public keys, the transcript, the session key and random coins r given to $\mathcal{M}$. (sk_i, pk_i) denotes long-term key pairs of parties for $i \in \{I, R\}$ (I for initializer, R for responder).

3.5.1 Bad Sampler

We now define a particular type of sampling algorithm for G which we call a *Bad Sampler*. We will prove that the existence of a bad sampler is equivalent to MQV being incriminating.

We say that a sampling algorithm for G is $(T_{\mathsf{Samp}}, T_{\mathsf{Solv}}, T_{\mathsf{D}}, \varepsilon_{\mathsf{Solv}}, \varepsilon_{\mathsf{D}})$-Bad if the following conditions are satisfied:

There exists a sampling algorithm Sample which satisfies the following

1. Sample takes as input A (uniformly picked from G) and the random coins r to generate $Y = \mathsf{Sample}(A, r)$ running in time $\leq T_{\mathsf{Samp}}$.

2. $\forall$ Solve running in T_{Solv}

$$Pr_{\mathsf{Solve},\, A,\, r}[\mathsf{Solve}(A, Y, r) = g^{ay} \mid Y = \mathsf{Sample}(A, r)] \leq \varepsilon_{\mathsf{Solv}}$$

   Probability is over the randomness of Solve, uniform choice of $A = g^a$ and random coins r.

3. There exists a distinguisher D running in time $\leq T_{\mathsf{D}}$ which tells apart g^{ay} from a random group member $\hat{g} \leftarrow G$ for a uniformly chosen A and random coins r.

$$|Pr_{\mathsf{D},\, A,\, r}[\mathsf{D}(A, Y, r, g^{ay}) = 1 \mid Y = \mathsf{Sample}(A, r)] -$$
$$Pr_{\mathsf{D},\, A,\, r}[\mathsf{D}(A, Y, r, \hat{g}) = 1 \mid Y = \mathsf{Sample}(A, r)]| \geq \varepsilon_{\mathsf{D}}$$

3.5.2 Equivalence between Bad Sampling and Incrimination

In the following Theorem, if T is the running time of an algorithm then the notation $\tilde{T}$ means T plus a constant number of exponentiations in G.

Theorem 3.5.1. *If there is a* $(T_{\mathsf{Samp}}, T_{\mathsf{Solv}}, T_{\mathsf{D}}, \varepsilon_{\mathsf{Solv}}, \varepsilon_{\mathsf{D}})$*-bad sampler in* G *then the MQV protocol is* $(\tilde{T}_{\mathcal{M}}, \tilde{T}_{\mathsf{SIM}}, \tilde{T}_{\mathsf{J}}, \varepsilon_{\mathsf{J}})$*-incriminating with* $\varepsilon_{\mathsf{J}} = \varepsilon_{\mathsf{D}}(1 - \varepsilon_{\mathsf{Solv}})$.

Conversely if the MQV protocol run over G *is* $(T_{\mathcal{M}}, T_{\mathsf{SIM}}, T_{\mathsf{J}}, \varepsilon_{\mathsf{J}})$*-incriminating then there exists a* $(\tilde{T}_{\mathsf{Samp}}, \tilde{T}_{\mathsf{Solv}}, \tilde{T}_{\mathsf{D}}, \varepsilon_{\mathsf{Solv}}, \varepsilon_{\mathsf{D}})$*-bad sampler for* G*, with* $\varepsilon_{\mathsf{Solv}} = (1 - \varepsilon_{\mathsf{J}})$ *and* $\varepsilon_{\mathsf{D}} = \varepsilon_{\mathsf{J}}$.

Malicious Initiator. The theorem above proves the equivalence of bad sampling with the non-deniability of MQV for the initiator when interacting with a malicious responder. It is also not hard to see that a similar theorem holds for the case of a malicious initiator who is trying to incriminate the responder. In this case also, the only possible strategy for a malicious initiator will be to run a bad sampler.

Theorem Interpretation. The above theorem, which will guide us towards the proof of deniability in Section 3.6, characterizes the strategy that the adversary needs to follow to be able to incriminate one of the parties: the only way to do it is to be able to sample elements Y in G such that for every element $A \leftarrow G$ it is easy to decide if $DH(A, Y)$ is correct while it is still hard to compute it. If we assume that such "bad" sampling is infeasible, then we immediately have a proof that the protocols are deniable. Yet such proof is a "non-constructive" one, as we are not able to show how the simulator works, but just that it must exist. The significance of such a statement in real-life is not clear, as plausible deniability requires the judge to actually run a simulator to counter Bob's statement that he spoke to Alice. In the absence of such real simulator, there is no way to argue that the conversation was not generated by Alice, even if we assume that bad sampling is impossible.

As before we are stuck on the fact that when we are trying to simulate a malicious Bob we do not know if he did sample $Y = g^y$ with or without knowledge of y (or more precisely

with or without knowledge of $DH(A, Y)$). The above theorem says that if bad sampling is impossible then either Bob must know $DH(A, Y)$ or the value is indistinguishable from random: in either case we would be able to successfully complete the simulation if we knew which case we were in (and in the former be able to "extract" $DH(A, Y)$). But the mere assumption that bad sampling is impossible does not give us this knowledge, and therefore leaves us stuck in the construction of a simulator.

The next section shows how to define an appropriate "knowledge extractor" for Bob, that will allow us to build a simulator.

3.5.3 Proof of Theorem 3.5.1

Proof. The proof we present is for the case in which the adversary plays the role of Bob, the responder. A similar proof can be easily replicated for the case in which the adversary plays the role of Alice (the initiator).

We assume that the long-term key $B = g^b$ of Bob is certified which means that the secret key b is available to the simulator. Also we assume that Bob is willing to reveal b to the judge when trying to incriminate Alice.

Bad Sampler $\implies$ MQV is Incriminating

Here, we assume that we have algorithms Sample and D according to the definition of Bad Sampler. Let Bob* be the malicious responder.

- Bob* runs on input $A = g^a$ and $X = g^x$ the long-term and ephemeral keys of Alice (respectively). He randomizes those keys as follows

$$\hat{A} := A^{d\alpha}.g^u$$

$$\hat{X} := X^{\alpha}.g^{u'}$$

where d is defined as in the MQV protocol (it only depends on X).

- Bob* selects coin tosses r and invokes the bad sampler on the product $Z = \hat{A}\hat{X}$

$$Y = \mathsf{Sample}(Z, r)$$

and sends Y as its response. Let $Y = g^y$ (but we do not know y).

- Let now SIM be a simulator running in time $< \tilde{T}_{\mathsf{Solv}}$, and let S be the output of SIM for the session key which we denote as

$$S = (XA^d)^y (XA^d)^{be} g^\lambda = Kg^\lambda$$

i.e. the session key $K = (XA^d)^y (XA^e)^b$ times an offsetting factor g^λ. Here, e is defined as in the MQV protocol (depends on Y).

- Since SIM runs in time $< \tilde{T}_{\mathsf{Solv}}$ it must be that $\lambda = 0$ with probability at most $\varepsilon_{\mathsf{Solv}}$. In fact if $\lambda = 0$ then we have a solver Solve running in time $< T_{\mathsf{Solv}}$ which computes Z^y which can only happen with probability at most $\varepsilon_{\mathsf{Solv}}$. The solver runs as follows
  - Runs SIM to obtain $S = K = (XA^d)^y (XA^d)^{be}$
  - Computes $\hat{S} = S(XA^d)^{-be} = (XA^d)^y$ – here, we assume that Solve knows b since it has the coin tosses of Bob*
  - Compute $S' = \hat{S}^\alpha \cdot Y^{u+u'} = [A^{d\alpha} g^u X^\alpha g^{u'}]^y = Z^y$

- We now build a judge J running in time $\tilde{T}_{\mathsf{D}}$. Given the output S of SIM it computes $\hat{S} = S(XA^d)^{-be} = (XA^d)^y g^\lambda$ and $S' = \hat{S}^\alpha \cdot Y^{u+u'} = Z^y g^{\alpha\lambda}$ and then runs the bad sampler distinguisher D on it.

  Note that if $\lambda = 0$, then D cannot distinguish, but if $\lambda \neq 0$ the value S' is uniformly

distributed in G. Therefore the distinguisher D is guaranteed to distinguish with probability $> \varepsilon_{\mathsf{D}}$.

Therefore our judge distinguishes with probability $\epsilon_{\mathsf{J}} > \varepsilon_{\mathsf{D}}(1 - \varepsilon_{\mathsf{Solv}})$.

MQV is Incriminating $\implies$ Bad Sampler

Given a bad Bob* and a judge J we need to construct a bad sampler Sample and a distinguisher D.

Construction of Sample

- Let Z be the input given to Sample. The sampler chooses at random $X = g^x$, computes d as in the MQV protocol and solves for A such that $A^d X = Z$
- Sample runs Bob* on input A, X and coin tosses r, and output the Y that Bob* outputs. Note that Sample runs in time $\tilde{T}_{\mathcal{M}}$ where $T_{\mathcal{M}}$ is the running time of the adversary (Bob*).

Let Solve be an algorithm running in time $\tilde{T}_{\mathsf{SIM}}$ which computes $DH(Z, Y)$. Then consider the simulator SIM running in time T_{SIM} which runs Solve to compute $Z^y = (A^d X)^y$ and then compute $K = Z^y (A^d X)^{be}$ (again we assume the simulator knows b). This is a perfect simulation and therefore fools any judge. But we know that for every simulator there is a judge that distinguishes with probability $> \varepsilon_{\mathsf{J}}$ so the algorithm Solve can only succeed with probability $< 1 - \varepsilon_{\mathsf{J}}$.

Finally consider the simulator SIM_R which outputs a random session key R. For this simulator there is a judge J that distinguishes with probability $> \varepsilon_{\mathsf{J}}$. Then we can build our distinguisher D as follows. Given a value W that it is either Z^y or random R' the distinguisher computes $W(A^d X)^{be}$ which is either the correct key or a random value (i.e. the output of SIM_R). If we run J on this value we distinguish with probability $> \varepsilon_{\mathsf{J}}$.

3.6 Deniability Proof

As we discussed in the previous section, the roadblock in the construction of a deniability simulator is that the simulator does not know if a malicious Bob knows the value $DH(A, Y)$ or not, at the moment Bob sends Y out. In the case of MQV, we also showed that the only way a malicious Bob can frame Alice is if he samples a Y for which he does *not* know $DH(A, Y)$, but such value can be efficiently recognized as correct (i.e. distinguished from a random value).

3.6.1 The Case of MQV

The above discussion therefore points out to the natural definition of a "knowledge extractor" which allows us to build a simulator for MQV. If we assume that given a malicious responder Bob, we can either (i) extract the value $DH(A, Y)$ or (ii) assume that $DH(A, Y)$ is indistinguishable from random, then the simulator immediately follows as the output of the extractor will be the simulated key.

We call this the Strong Knowledge of DH (SKDH) Assumption and it is defined below. In the next section we define a weaker version of this assumption which will be sufficient to prove HMQV and 3DH.

Definition 3.6.1. Let G be a cyclic group and AUX a class of auxiliary inputs. Let M be a probabilistic Turing Machine running in time T_M on input (U, aux) where $U \leftarrow G$, and $aux \in AUX$, and outputs $Z \in G$; we denote with $Z = M(U, aux, r)$ the output of running M on input U, aux with coin tosses r. We say that the $(T_M, T_{\hat{M}}, T_D, \varepsilon_D)$-SKDH Assumption holds over group G and class AUX, if for every such M, there exists a companion probabilistic Turing Machine $\hat{M}$ (called the extractor for M) such that: $\hat{M}$ runs on input (U, aux, r) in

time $T_{\hat{M}}$ and outputs $\hat{Z} \in G$ such that the distributions

$$[U, aux, r, DH(U, Z)] \quad \text{and} \quad [U, aux, r, \hat{Z}]$$

are (T_D, ε_D)-indistinguishable.

Remark: Basically the assumption says that for every sampler M of a value Z, there is an extractor that either computes $DH(U, Z)$ or produces an output distribution that is computationally indistinguishable from $DH(U, Z)$ even when given the internal coin tosses of M. The assumption is written generically: when Bob [resp. Alice] is the adversary $U = A$ [resp. $U = B$] the peer's long-term public key, and $Z = Y$ [resp. $Z = X$] the adversary's ephemeral value.

Recall from Section 3.1.4 that we assume key registration as a way for the simulator to extract the private key of the attacker. We note that we can prove MQV deniability without key registration by strengthening the SKDH assumption.

Theorem 3.6.2. *Under the $(T_M, T_{\hat{M}}, T_D, \varepsilon_D)$ SKDH Assumption, MQV with Key Registration is a $(\tilde{T}_M, \tilde{T}_{\hat{M}}, \tilde{T}_D, \varepsilon_D)$ deniable AKE.*

Proof. We prove deniability for the initiator. The proof for the responder is similar.

SIM, on input public key A of Alice (but not her secret key a), interacts with the adversary Bob. Because we assume Key Registration, Bob has proven knowledge of his secret key b during key registration and therefore we can assume that SIM has extracted it.

SIM runs the algorithm of Bob with input A, aux', r (where $aux' = aux||X$) and receives Y as Bob's ephemeral public key. Then it runs the extractor for Bob which is provided by the SKDH assumption (with $U = A$).

Let $\hat{Z}$ be the extractor's output. SIM computes the key using $\hat{Z}$ as $(A^d X)^{be} \hat{Z}^d Y^x$.

For contradiction, assume a judge J running in time T_J distinguishes the key resulting

from a real execution $K_{\text{real}} = DH(A^dX, B^eY)$ from a simulated key $K_{\text{sim}} = (A^dX)^{be}\hat{Z}^dY^x$ with probability ε_J:

$$p = Pr[J(A, B, X, Y, K_{\text{real}}, aux, r) = 1]$$

$$\hat{p} = Pr[J(A, B, X, Y, K_{\text{sim}}, aux, r) = 1]$$

$$|p - \hat{p}| > \varepsilon_J$$

This contradicts the indistinguishability claim of SKDH assumption for the parameters $\varepsilon_J = \varepsilon_D$ and $T_J = (T_D +$ constant number of exponentiations and multiplications).

3.6.2 The Case of HMQV and 3DH

For HMQV and 3DH we can use a weaker assumption. In this case, when the extractor fails to produce $DH(A, Y)$ we do not need to establish that $DH(A, Y)$ is indistinguishable from random, but rather that it is infeasible to compute. This is sufficient because the session key (in both HMQV and 3DH) is the result of a random oracle call over a function of $DH(A, Y)$. Thus if the random oracle is not queried on this value, then the session key can be simulated with a random value.

Before presenting the KDH assumption on which the deniability of HMQV and 3DH will be proven, we motivate it on the basis of the well-known Knowledge of Exponent Assumption (KEA) [22, 5]. Recall that, informally, KEA states that for any algorithm M that on input (g, g^u) outputs a pair (Z, Z^u), there is an algorithm M' that outputs z such that $Z = g^z$. Machine M' runs on the same inputs as M, including same random coins. (Here g is a generator of the group G, u is chosen uniformly and Z can be any group element.) .

Can we base deniability on KEA? At first glance, it would seem that the deniability of HMQV and 3DH will follow from KEA. Indeed, in the case of an adversarial Bob, the simulator SIM needs to learn whether Bob computed the session key K and if so what the

value of K was. To compute K, Bob must query the necessary DH values, e.g., $DH(A, Y)$, from the random oracle RO. If Bob does query these values, SIM can learn K; if it does not, then SIM can replace K with a random value. The question is how does SIM identify which query to the RO equaled $DH(A, Y)$, if any. For this, SIM can resort to the KEA extractor which upon computation of $V = DH(A, Y)$ by Bob outputs $y = \mathsf{dlog}(Y)$, thus allowing SIM to check if indeed $Y = g^y$ and $V = A^y$. So it seems that we are done and KEA is all that is needed here.

But there is a problem. Consider the following scenario. Bob does not query the RO on the required values during the simulated session and does not compute K, so SIM sets the key to a random value K'. Later, Bob provides y and b to the judge who can compute the key K and make the simulation fail by distinguishing K from K'. To prevent this "trivial" simulation failure we need to assume that if it is possible at all to efficiently compute the session key K given the information Bob has, then there exists an extractor that on the same inputs of Bob (including Bob's random coins) produces K. Then SIM can use this extractor to output the key K, and if the extractor does not output K then SIM sets it to a random value K'. In the latter case, it is guaranteed that the judge will not be able to compute the key K and distinguish the simulation. In general terms, what this approach captures is the fact that SIM (acting as the alter ego of Bob), can use Bob in a non-black-box way and extract from it all knowledge needed to complete a successful simulation. In particular, if the distinguisher (judge) can compute the correct key K based on the attacker's (Bob) view, so should SIM.

So to use KEA, we would need to strengthen it by requiring not only that if Bob computed $DH(A, Y)$ then the extractor outputs $y = \mathsf{dlog}(Y)$, but also that if Bob does not output $DH(A, Y)$ then no other machine has a non-negligible probability of outputting $DH(A, Y)$ (or y itself) on Bob's inputs (more precisely, that no machine can succeed with non-negligible probability over the distribution of inputs where Bob failed to output DH(A,Y)).

The following KDH assumption is a generalization of this KEA strengthening.

Definition 3.6.3. Let G be a cyclic group and AUX a class of auxiliary inputs. Let M be a probabilistic Turing Machine running in time T_M which runs on input (U, aux) where $U \leftarrow G$, and $aux \in AUX$, and outputs $Z \in G$; we denote with $Z = M(U, aux, r)$ the output of running M on input U, aux with coin tosses r.

We say that the $(T_M, T_{\hat{M}}, T_G, \varepsilon_G)$ *Knowledge of DH (KDH) Assumption* holds over group G and class AUX, if for every such M, there exists a companion probabilistic Turing Machine $\hat{M}$ (called the extractor for M) that runs on input U, aux, r in time $T_{\hat{M}}$ and outputs $\hat{Z} \in G$ or $\hat{Z} = \perp$ such that

- For all U, aux, r, if $\hat{M}(U, aux, r) = \hat{Z} \neq \perp$ then $\hat{Z} = DH(U, Z)$
- For every probabilistic Turing Machine C running in time T_G we have that

$$Prob[C(U, r, aux) = DH(U, Z) \mid \hat{M}(U, aux, r) = \perp] \leq \varepsilon_G$$

where $Z = M(U, aux, r)$ and the probability is taken over the coin tosses of C and uniform distribution on (U, r).

The first condition[6] says that if the extractor outputs a group element, this element must be $DH(U, Z)$. The second condition says that no machine can succeed to compute $DH(U, Z)$ with non-negligible probability over the distribution of triples (U, aux, r) where $\hat{M}$ outputs $\perp$.

As said, the KDH assumption can be seen as a strengthening of KEA. In Section 3.9 we show how the combination of KEA with another natural knowledge assumption, "knowledge of discrete log" (KDL), implies KDH. Thus, protocols proven deniable under KDH are deniable under KEA+KDL, providing more confidence on the deniability proof.

[6]This can be relaxed to allow a negligible set of (U, r) values where the condition does not hold.

We now show that KDH implies the deniability of HMQV if the simulator can extract the incriminating partys (Bob in our examples) private key, for example via key registration as discussed in Section 3.1.4. In Section 3.8 we remove the need for this extraction condition using a mild generalization of KDH.

Theorem 3.6.4. *Under the $(T_M, T_{\hat{M}}, T_G, \varepsilon_G)$-KDH Assumption, HMQV with Key Registration is a $(\tilde{T}_M, \tilde{T}_{\hat{M}}, \tilde{T}_G, \varepsilon_G)$ deniable AKE in the random oracle model.*

Proof. We consider deniability for the initiator against a possibly malicious responder. The case of deniability for the responder against a possibly malicious initiator is dealt similarly.

We build a simulator SIM which on input the public key A of Alice (but not her secret key a), interacts with a possibly malicious Bob and outputs a view that is indistinguishable from the real one. Again we assume that long-term keys are registered and therefore the simulator knows b, the secret key of Bob.

SIM chooses $x \leftarrow \mathbb{Z}_q$ and computes $X = g^x$. At this point the value d is determined in the HMQV protocol. Bob receives X and outputs $Y = g^y$ (which determines the value e). Note that Bob can be seen as a machine M in the definition of the KDHA: it runs on input $U = A$ and some auxiliary information aux which includes X. Therefore under the KDHA there must be an extractor $\hat{B}$ for Bob.

Remember that the real key of the HMQV protocol is defined as $K = H[(XA^d)^{y+be}]$ where H is a random oracle. The simulator can easily compute $(XA^d)^{be}$ since it knows b. To compute $(XA^d)^y$ it invokes $\hat{B}$.

- If $\hat{B}$ outputs $\hat{Z} = DH(A, Y)$ then SIM sets the key to $H[Y^x \hat{Z}^d (XA^d)^{be}]$, i.e. the real key.
- If $\hat{B}$ outputs $\perp$ then SIM sets the key to $K \leftarrow \{0,1\}^n$ where n is the length of the session key.

Note that in the second case, any judge running in time less than T_G will not be able to compute $DH(A, Y)$ and therefore will not be able to query the random oracle in the preimage of the real key. This immediately yields that for this judge a random key is indistinguishable from the real key.

For the case of 3DH the Key Registration step is not necessary since the value g^{ab} (the Diffie-Hellman transform of the long-term secret keys) is not included in the session key.

Theorem 3.6.5. *Under the $(T_M, T_{\hat{M}}, T_G, \varepsilon_G)$ KDHA, 3DH is a $(\tilde{T}_M, \tilde{T}_{\hat{M}}, \tilde{T}_G, \varepsilon_G)$ deniable AKE in the random oracle model.*

Proof. We consider deniability for the initiator against a possibly malicious responder. The case of deniability for the responder against a possibly malicious initiator is dealt similarly.

We build a simulator SIM which on input the public key A of Alice (but not her secret key a), interacts with a possibly malicious Bob and outputs a view that is indistinguishable from the real one.

SIM chooses $x \leftarrow \mathbb{Z}_q$ and computes $X = g^x$. Bob receives X and outputs $Y = g^y$. Note that Bob can be seen as a machine M in the definition of the KDHA: it runs on input $U = A$ and some auxiliary information aux which includes X. Therefore under the KDHA there must be an extractor $\hat{B}$ for Bob.

Remember that the real key of the 3DH protocol is defined as

$$K = H[DH(A, Y)||DH(X, Y)||DH(B, X)]$$

where H is a random oracle. The simulator can easily compute $DH(X, Y)$ and $DH(B, X)$ since it knows x. To compute $DH(A, Y)$ it invokes $\hat{B}$.

- If $\hat{B}$ outputs $\hat{Z} = DH(A, Y)$ then SIM sets the key to

$$K = H[\hat{Z}||DH(X, Y)||DH(B, X)]$$

i.e. the real key.

- If $\hat{B}$ outputs $\perp$ then SIM sets the key to $K \leftarrow \{0,1\}^n$ where n is the length of the session key.

Note that in the second case, any judge running in time less than T_G will not be able to compute $DH(A, Y)$ and therefore will not be able to query the random oracle in the preimage of the real key. This immediately yields that for this judge a random key is indistinguishable from the real key.

3.7 3DH vs Signal

In this section, we show that the deniability of 3DH (independently of the assumptions on which such deniability can be proven) implies the deniability of X3DH and the full Signal protocol. We do this by invoking Theorem 3.3.2 on the message flow of Signal.

We refer the reader to [20] for a full description of the Signal protocol and its security analysis. Informally we can describe Signal as an initial AKE which establishes a *root key*, followed by a secure session where messages are exchanged. However each message exchange is accompanied by a *ratcheting* step, which generates new session key. These sequence of keys, creates a *key chain* where keys are authenticated by their predecessor in the chain. In a *symmetric ratcheting* step the current chain key K is fed to a KDF function to generate two keys, the new chain key K_1 and the key K_2 used to encrypt/authenticate the message at this round. In a *asymmetric* ratcheting the parties perform a new Diffie-Hellman exchange over two ephemeral keys and feed the result to a KDF together with the current chain key, also outputting K_1, K_2 as above.

Note how in the above description, after the initial AKE which establishes a session key K, the messages exchanged in the protocol do *not* use the long-term secret keys of the

parties. Therefore if the initial AKE is deniable we can apply Theorem 3.3.2 and claim the deniability of Signal.

THE X3DH PROTOCOL. If the initial AKE protocol in Signal were 3DH we would be done. However to enable asynchronous communication (where Bob, the responder, could be offline at the moment in which Alice, the initiator, sends him a message), the Signal protocol uses the X3DH variant of 3DH. This variant allows Bob to load his ephemeral key Y onto a key distribution server (a *pre-key* in Signal jargon). To prevent impersonation attacks by the server, Bob will *sign* Y with his long term secret key. When Alice wants to talk to Bob she queries the key distribution server for Bob's ephemeral key and runs the 3DH protocol to establish a root chain key K_1 and a message key K_2 used to secure the first message she sends to Bob. At this point Alice and Bob will continue with the ratcheting mechanism described above. We now move to establish the deniability of X3DH.

It is not hard to see that the proof of deniability of 3DH extends to X3DH in the case of the initiator. Indeed, the deniability argument for Alice in X3DH is the same as for the responder in 3DH since here Alice acts on the ephemeral value Y chosen by Bob. In contrast, deniability with respect to Bob in X3DH is complicated by the fact that Bob signs the value Y. But note that Bob places Y and its signature on a public server that anyone can access. Thus, Y is not bound to any specific peer, and cannot be used as a proof that Bob communicated with anyone.

Formally, we can consider Y and its signature as auxiliary information that an adversarial Alice has when initiating the protocol, and can therefore be provided to the simulator as well. While this is the intuition behind the simulation argument, there is another technical twist at this point. In the 3DH simulation of Bob against a malicious Alice, the simulator is allowed to choose $y \leftarrow \mathbb{Z}_q$ and set $Y = g^y$; the knowledge of y helps the simulator in the computation of the correct key. In the X3DH simulation, however, Y is part of the auxiliary input and the simulator has no access to y. Intuitively, because Bob signs Y, the latter can

be seen as another public key associated with him and the simulator cannot be given its secret key.

The problem boils down to the computation of g^{xy}. In the 3DH simulation, we simply computed it through the knowledge of y. Here, we need to extract it from Alice, and this requires an additional assumption that says we can extract *both* g^{xy} and g^{bx}.

Definition 3.7.1. Let G be a cyclic group and AUX a class of auxiliary inputs. Let M be a probabilistic Turing Machine running in time T_M which runs on input (U, W, aux) where $U, W \leftarrow G$, and $aux \in AUX$, and outputs $Z \in G$; we denote with $Z = M(U, W, aux, r)$ the output of running M on input U, W, aux with coin tosses r.

We say that the $(T_M, T_{\hat{M}}, T_G, \varepsilon_G)$ *Knowledge of 2DH (K2DH) Assumption* holds over group G and class AUX, if for every such M, there exists a companion probabilistic Turing Machine $\hat{M}$ (called the extractor for M) such that: $\hat{M}$ runs on input U, W, aux, r in time $T_{\hat{M}}$ and outputs $\hat{Z}_1, \hat{Z}_2 \in G$ or $\perp$ such that

- If $\hat{M}(U, W, aux, r) \neq \perp$ then $\hat{Z}_1 = DH(U, Z)$ and $\hat{Z}_2 = DH(W, Z)$
- If $\hat{M}(U, W, aux, r) = \perp$ then for every probabilistic Turing Machine C running in time T_G we have that

$$Prob[C(U, W, Z, r, aux) \in \{DH(U, Z), DH(W, Z)\} \mid \hat{M}(U, W, aux, r) = \perp] \leq \varepsilon_G$$

where $Z = M(U, W, aux, r)$ and the probability is taken over the coin tosses of C and uniform distribution on (U, W, r).

The reliance of the following theorem on extractability of the private key via Key Registration is removed in Section 3.8.

Theorem 3.7.2. *Under the $(T_M, T_{\hat{M}}, T_G, \varepsilon_G)$ K2DHA, X3DH with Key Registration is a $(\tilde{T}_M, \tilde{T}_{\hat{M}}, \tilde{T}_G, \varepsilon_G)$ deniable AKE in the random oracle model.*

Proof. Deniability for the initiator follows the same structure as Theorem 3.6.5. We now consider deniability for the responder against a possibly malicious initiator.

We build a simulator SIM which on input the long-term public key B of Bob and his signed pre-key Y, sig (but not the matching secret keys b, y), interacts with a possibly malicious Alice and outputs a view that is indistinguishable from the real one. We assume that SIM knows a (the long-term secret key of Alice) due to key registration.

SIM receives X from Alice, and Y, sig from the server. Let $\hat{A}$ be the extractor associated with Alice (running on input $U = B$, $W = Y$ and $Z = X$) guaranteed by the K2DHA.

Remember that the real key of the X3DH protocol is defined as

$$K = H[DH(A, Y)||DH(X, Y)||DH(B, X)]$$

where H is a random oracle. Note that the simulator knows $DH(A, Y)$ since it knows a. The simulator invokes $\hat{A}$. If the extractor outputs $\perp$ at any of the invocations, then the simulator outputs a random session key (which under the K2DHA is indistinguishable from the real one in the random oracle model). Otherwise the output of $\hat{A}$ is $\hat{Z}_1 = DH(U, Z) = DH(B, X)$ and $\hat{Z}_2 = DH(W, Z) = DH(X, Y)$ and therefore the simulator has all the values to compute the correct session key.

3.8 On the need to extract the long-term private keys

The simulation arguments of Theorems 3.6.4 and 3.7.2 assume the ability to extract the incriminating party's (Bob in our examples) private key, for example via key registration. This simplified the proofs and intuition. We note however that such extraction is not essential. Instead, we can generalize our extraction assumptions to prevent Bob from sampling either B or Y in a way that he does not know the discrete logs and yet *both* g^{ab} *and* g^{ay} are

distinguishable from random. Indeed, what happens (in either HMQV, 3DH and X3DH) is that Bob will be able to incriminate Alice if (and only if) he is able to sample either B or Y under the above conditions. Formally, we achieve this by adding one extra "knowledge" assumption about the way parties generate their long-term keys; arguably, this additional assumption is not essentially stronger than the previous ones. Details follow.

Definition 3.8.1. Let G be a cyclic group and AUX a class of auxiliary inputs. Let M be a probabilistic Turing Machine running in time T_M which runs on input $aux \in AUX$, and outputs $Z \in G$; we denote with $Z = M(aux, r)$ the output of running M on input aux with coin tosses r.

We say that the $(T_M, T_{\hat{M}}, T_G, \varepsilon_G)$ *Extended Knowledge of DH (EKDH) Assumption* holds over group G and class AUX, if for every such M, there exists a companion probabilistic Turing Machine $\hat{M}$ (called the extractor for M) such that: $\hat{M}$ runs on input aux, r and an additional input $U \in G$, in time $T_{\hat{M}}$ and outputs $\hat{Z}$ or $\perp$ such that

- If $\hat{M}(U, aux, r) = \hat{Z} \neq \perp$ then $\hat{Z} = DH(U, Z)$
- If $\hat{M}(U, aux, r) = \perp$ then for every probabilistic Turing Machine C running in time T_G we have that

$$Prob[C(U, r, aux) = DH(U, Z) \mid \hat{M}(aux, r) = \perp] \leq \varepsilon_G$$

where $Z = M(aux, r)$ and the probability is taken over the coin tosses of C and uniform distribution on r.

Note the difference between the KDH and the EKDH Assumption. In the latter, the group element $U \in G$ is not known to the machine M, but is fed to the extractor as input. This is because we need to model a machine M that generates Z as its long-term public key *before* seeing any of the keys of the parties it will interact with.

Theorem 3.8.2. *Under the* $(T_M, T_{\hat{M}}, T_G, \varepsilon_G)$ *EKDHA, HMQV protocol is* $(\tilde{T}_M, \tilde{T}_{\hat{M}}, \tilde{T}_G, \varepsilon_G)$ *deniable AKE in the random oracle model, even without registration of long-term public keys.*

Proof. The proof for the deniability with respect to initiator follows closely the proof of Theorem 3.6.4.

For the case of the responder, recall that the session key is defined as $K = H[(XA^d)^{y+be}]$. The simulator computes $Y^{x+ad} = (XA^d)^y$ since it knows y such that $Y = g^y$. It computes $(B^e)^{x+ad}$ by invoking the extractor $\hat{A}$ twice under the EKDHA. Recall that Alice output the public key $A = g^a$, therefore the extractor $\hat{A}$ on input B^{de} will output either B^{ade} or $\perp$. Similarly, after Alice sends $X = g^x$ we can invoke $\hat{A}$ on input B^e to get either B^{ex} or $\perp$. If either output is $\perp$ the simulator outputs a random session key, otherwise it has extracted the correct key.

A similar theorem holds for X3DH.

3.9 Knowledge of Discrete Log Assumption

In Section 3.6.2 we introduced the Knowledge of Diffie-Hellman (KDH) assumption that forms the basis for the proof of deniability of the protocols considered in this paper. As explained there, KDH can be seen as a strengthening of the well-known Knowledge of Exponent Assumption (KEA). Here, we shed further light on the relation between KEA and KDH by introducing another knowledge assumption, Knowledge of Discrete Log (KDL), and showing that KDH is implied by the conjunction of KEA and KDL. In particular, it means that these two assumptions, taken together, suffice for our proofs of deniability. Arguably, the KDL is a natural knowledge assumption very much in the style of KEA and possibly more appealing than KDH.

We start by providing a formal definition of KEA (with auxiliary input) [22, 39, 5].

Definition 3.9.1. Let G be a cyclic group of order q generated by element g and AUX be a class of auxiliary inputs. Let M be a probabilistic Turing Machine that receives as input pairs (g, g^u), $aux \in AUX$ and random coins r, and outputs a pair of elements in G, which we denote by $(V, W) = M(g, g^u, aux, r)$.

We say that $(T, \bar{T}, \varepsilon_{KEA})$-Knowledge of Exponent Assumption (KEA) with AUX holds over group G with respect to generator g if for every machine M as above running in time T, there exists a probabilistic Turing Machine $\bar{M}$ that runs in time $\bar{T}$ on the same inputs and coins as M and outputs an element in Z_p such that the following holds:

$$Prob[M(g, g^u, aux, r) = (V, W = V^u) \text{ and } \bar{M}(g, g^u, aux, r) \neq \mathsf{dlog}_g(V)] < \varepsilon_{KEA},$$

where the probability is over the uniform choice of $u \leftarrow \mathbb{Z}_q$ and the random coins r of M.

Note: When the value of g is clear from the context we omit it as subscript to dlog; and sometimes omit it as input to machines M and $\bar{M}$.

KEA captures the intuition that if a machine, on input (g, g^u), can *sample* a value $Z \in G$ for which it can produce a pair (Z, Z^u) then it has enough "internal knowledge" to also produce $z = \mathsf{dlog}(Z)$. More generally, one can consider machines that sample elements Z in G and ask whether the internal processing that leads to sampling Z has enough information to extract $z = \mathsf{dlog}(Z)$. In more detail, consider a process that samples elements from a cyclic group of order q, generated by an element g. For example, the sampler could use its random coins r to choose an exponent $y \in \mathbb{Z}_q$ and output $Y = g^y$ or it could hash r into a random group element in a way that $y = \mathsf{dlog}(Y)$ is hard to compute. In the first case, examining the internal computation one can extract y while in the latter one cannot. We introduce Knowledge of Discrete Log (KDL) Assumption, which says that, similarly to KEA, if in the process of generating $Y \in G$ there is enough information to extract $y = \mathsf{dlog}(Y)$, then there is an extractor that running on the same coins as the sampler outputs y. Furthermore, such

extractor is "maximal" in the sense that on inputs it fails to output y, other machines will fail too.

We now formalize the KDL Assumption and show that together with KEA, it implies the KDH assumptions, and therefore together they imply deniability of HMQV and 3DH.

Definition 3.9.2. Let G be a finite cyclic group generated by an element g of order q. Consider a probabilistic Turing Machine M with inputs in some set S, such that for any $s \in S$ and random coins $r \in \{0,1\}^\ell$ (for some $\ell \in \mathbb{Z}^+$), M outputs an element $M(s,r) = Y$ in G. We call such a machine a *G-sampler*.

A probabilistic Turing Machine M' is called a *dlog extractor for the G-sampler M*, if for every r, s, $M'(s,r)$ outputs $\mathsf{dlog}(Y)$ or $\perp$.

A machine M' is called a $(T, T', T_{max}, \varepsilon_{max})$-*maximal dlog extractor for a G-sampler M*, if it satisfies the following:

- The running time of M and M' are bounded by T and T', respectively.
- For all $r \in \{0,1\}^\ell$ and $s \in S$, if $M'(s,r) \neq \perp$ then $M'(s,r) = \mathsf{dlog}(M(s,r))$.
- For every probabilistic Turing Machine C running in time T_{max} and for all $s \in S$,

$$Prob_{C,r}[C(s,r) = \mathsf{dlog}(M(s,r))] < Prob_{M',r}[M'(s,r) = \mathsf{dlog}(M(s,r))] + \varepsilon_{max}$$

  where probabilities are taken over coin tosses of the machines C and M' and the uniform distribution over values r.

Definition 3.9.3. Let G be a cyclic group of order q, generated by element g. We say that $(T, T', T_{max}, \varepsilon_{max})$-*Knowledge of Discrete Log* (KDL) Assumption holds over group G, if every *G-sampler* running in time at most T has a $(T, T', T_{max}, \varepsilon_{max})$-maximal dlog extractor.

As a reminder, in the following theorem when t is the running time of an algorithm, the notation $\tilde{t}$ denotes t plus a constant number of exponentiations in G.

Theorem 3.9.4. *Let G be a cyclic group of order q, generated by element g and AUX be a class of auxiliary inputs. If the $(T_{KEA}, \bar{T}_{KEA}, \varepsilon_{KEA})$-KEA with AUX and the $(T, T', T_{KDL}, \varepsilon_{KDL})$-KDL assumptions hold over G, then the $(T, \tilde{T}', \bar{T}_{KEA}$,*
$\varepsilon_{KDL} + \varepsilon_{KEA})$-KDH holds over G and AUX.

Proof. Let M be a probabilistic Turing Machine running on input U, aux, r and outputting $M(U, aux, r) = Z \in G$. We show that the KEA and KDL assumptions imply the existence of a KDH extractor for M as postulated by the KDH assumption.

We view machine M as a G-sampler, which receives a $(U||aux, r)$ input and outputs a group element Z. Let's assume that M runs in time T. By KDL assumption, there is a $(T, T', T_{KDL}, \varepsilon_{KDL})$-maximal dlog extractor M' for M. We build a KDH extractor $\hat{M}$ using M' as follows.

$\underline{\hat{M}(U, aux, r)}$

run $M'(U||aux, r) = \alpha$

if $\alpha = \mathsf{dlog}(Z)$

    return U^{α}

else

    return $\perp$

Note that when M' returns $\mathsf{dlog}(Z)$, $\hat{M}$ returns $DH(U, Z)$ and otherwise it returns $\perp$. Thus, $\hat{M}$ acts as a KDH extractor, whose running time is bounded by T' plus an exponentiation in G, hence $\tilde{T}'$. We need to show that $\hat{M}$ satisfies the condition in KDH. Informally, that if there is a way to compute $DH(U, Z)$ from the inputs (U, aux, r), then $\hat{M}$ will compute it.

Suppose, for contradiction, that there exists a probabilistic Turing Machine C which whose success probability is greater than $\varepsilon = \varepsilon_{KEA} + \varepsilon_{KDL}$ over the set of inputs where

$\hat{M}$ fails, i.e., returns $\perp$. Let Ψ denote the distribution over pairs of (U, r) conditioned on $\hat{M}(U, aux, r) = \perp$. Explicitly, the probability weight assigned to (U_0, r_0) by Ψ is that:

$$\Psi(U_0, r_0) = \frac{Prob_{\hat{M}}[\hat{M}(U_0, aux, r_0) = \perp]}{\Sigma_{(U,r)} Prob_{\hat{M}}[\hat{M}(U, aux, r) = \perp]},$$

where the machine name in the subscript denotes the random coins of the machine. We assume that

$$\underset{C,(U,r)\leftarrow\Psi}{Prob}[C(U, aux, r) = DH(U, Z)] > \varepsilon. \tag{3.1}$$

We define a probabilistic machine μ that runs on a set of inputs (U, aux, r) as defined for machines M and C above. For each such input, μ runs M and C and outputs a pair (Z, γ) where γ is either $DH(U, Z)$ or $\perp$ or another value.

$\underline{\mu(U, aux, r)}$

run $M(U||aux, r) = Z$

run $M'(U||aux, r) = \beta$

if $\beta \neq \perp$,

    set γ to U^{β}

else

    set γ to $C(U, aux, r)$

return (Z, γ)

By KEA, there exists an extractor $\bar{\mu}$ that receives the same input as μ and, except with probability at most ε_{KEA}, it returns $\mathsf{dlog}(Z)$ whenever the output of μ is $(Z, DH(U, Z))$ which happens either when M' successfully extracts ($\beta = \mathsf{dlog}(Z)$, or when M' fails and C succeeds in computing $\gamma = DH(U, Z)$.

We claim that if the assumption (3.1) on C holds, then $\bar{\mu}$ violates the maximality of the dlog extractor M'.

For simplicity in notation, assume that event A indicates $\mu(U, aux, r) = (Z, DH(U, Z))$ and event B indicates $\bar{\mu}(U, aux, r) = \mathsf{dlog}(Z)$. Negation of an event is denoted by $\neg$ symbol.

$$\begin{aligned} Prob\,[B] &\geq Prob\,[B \wedge A] \\ &= Prob\,[B|A]\; Prob\,[A] \\ &= (1 - Prob\,[\neg B|A])\; Prob\,[A] \\ &= Prob\,[A] - Prob\,[\neg B|A]\; Prob[A] \\ &= Prob\,[A] - Prob\,[\neg B \wedge A] \\ &> Prob\,[A] - \varepsilon_{KEA} \end{aligned} \tag{3.2}$$

First, note that the inequality (3.2) is implied by KEA. Using the inequality (3.2) and the definition of μ, we can conclude the following. $\mathcal{U}$ denotes the uniform distribution over the corresponding space and C in the subscript denotes the random coins of the machine C.

$$\begin{aligned} &\underset{(U,r)\leftarrow\mathcal{U}}{Prob}\,[\bar{\mu}(U, aux, r) = \mathsf{dlog}(Z)] \\ &> \underset{(U,r)\leftarrow\mathcal{U}}{Prob}\,[\mu(U, aux, r) = (Z, DH(U, Z))] - \varepsilon_{KEA} \qquad (3.3) \\ &= \underset{r\leftarrow\mathcal{U}}{Prob}\,[M'(U||aux, r) = \mathsf{dlog}(Z)] + \\ &\quad \underset{C,(U,r)\leftarrow\mathcal{U}}{Prob}\,[C(U, aux, r) = DH(U, Z) \;\wedge\; M'(U||aux, r) = \perp] - \varepsilon_{KEA} \qquad (3.4) \\ &> \underset{r\leftarrow\mathcal{U}}{Prob}\,[M'(U||aux, r) = \mathsf{dlog}(Z)] + \\ &\quad \underset{C,(U,r)\leftarrow\mathcal{U}}{Prob}\,[C(U, aux, r) = DH(U, Z) \mid M'(U||aux, r) = \perp] - \varepsilon_{KEA} \qquad (3.5) \\ &= \underset{r\leftarrow\mathcal{U}}{Prob}\,[M'(U||aux, r) = \mathsf{dlog}(Z)] + \end{aligned}$$

$$\underset{C,(U,r)\leftarrow\mathcal{U}}{Prob}\left[C(U,aux,r) = DH(U,Z) \mid \hat{M}(U,aux,r) = \perp\right] - \varepsilon_{KEA} \tag{3.6}$$

$$= \underset{r\leftarrow\mathcal{U}}{Prob}\left[M'(U||aux,r) = \mathsf{dlog}(Z)\right] +$$

$$\underset{C,(U,r)\leftarrow\Psi}{Prob}\left[C(U,aux,r) = DH(U,Z)\right] - \varepsilon_{KEA} \tag{3.7}$$

$$> \underset{r\leftarrow\mathcal{U}}{Prob}\left[M'(U||aux,r) = \mathsf{dlog}(Z)\right] + \varepsilon_{KDL} \tag{3.8}$$

In the above, line (3.3) is the repeat of inequality (3.2). The expression in (3.4) is due to the construction of μ and (3.5) follows from the conditional probability. In the line (3.6), we move from M' to $\hat{M}$ in the conditional part, this is due to the definition of $\hat{M}$. Equation (3.7) comes from the definition of the KDH assumption. Line (3.8) is due to the assumption made in (3.1) and by setting $\varepsilon = \varepsilon_{KEA} + \varepsilon_{KDL}$, as assumed.

Finally, we note that if we restrict the output of machine μ to its first element, namely the output of M, we get that $\bar{\mu}$ acts as a KDL-extractor for M. However, the above inequality shows that $\bar{\mu}$ breaks the assumed maximality of M' as a KDL extractor.

The runtime and success probability of the machine C set the third and fourth parameters of KDH. The bound on the runtime of C comes from the construction of μ:

$$T_{KEA} \leq T + T' + \text{ runtime of } C$$

The success probability that C outputs $DH(U,Z)$ is at most $\varepsilon = \varepsilon_{KEA} + \varepsilon_{KDL}$. Therefore the $(T, \tilde{T}', \bar{T}_{KEA}, \varepsilon_{KDL} + \varepsilon_{KEA})$-KDH holds over group G and AUX.

Chapter 4

Corollaries and Observations

4.1 Deniability Analysis of OAKE Protocol Family

Yao et. al. introduced a family of key exchange protocols named OAKE, in a series of papers [81, 80]. OAKE protocols are implicitly authenticated Diffie-Hellman with the communication complexity is identical to the unauthenticated Diffie-Hellman (See Figure 1.1). Deniability is one of the design aims for OAKE protocols.

The authors aimed for Honest Player deniability for the OAKE family, which is equivalent to the deniable key exchange assuming the protocol participants are acting honest. In this concept, the threat for the deniability comes from outsiders. An outsider adversary may be an eavesdropper on the communication channel, or an active adversary which may gather information from communicating parties by attempting incomplete sessions or interfering with the initiated sessions.

Note that with honest players, deniability simulation for iAKE protocols is quite simple. The session key will be a function of the transcript (A,B,X,Y) and simulator can make use DDH assumption to simulate any of the DH value $g^{ab}, g^{ay}, g^{bx}, g^{xy}$ with a random value. (However in the presence of an adversarial party, simulator cannot have a choice of using

DDH due to the fact that one DH component will be chosen adversarially which leads DDH to be inapplicable.) In case a random oracle is applied at the end to the key material, any random value suffices to simulate the session key. So all the challenge lies in simulating the transcript, if any.

Though, notice that, such a simulation strategy results in a computationally indistinguishable simulation. To ensure forward deniability, the authors sought statistical or perfect simulation (following a finding in [26] that forward deniability is implied by a statistical deniability simulation).

Yao et. al. analyzed the OAKE protocols on the basis of HP-deniability and forward deniability. The question of whether these protocols satisfy regular offline deniability was open. In this section, we are going to show that OAKE protocols are deniable under KDH assumption with the key registration. In the absence of key registration, T-OAKE requires EKDH for deniability while KDH is sufficient for the remaining versions in the random oracle model.

OAKE family includes four versions, all have the same communication. They only differ in the key computation: the coefficients that are used with the DH exponents differ though in every version coefficients are all computable on the public values (public keys and identities). In terms of deniability, main difference stems from the existence of a factor g^{ab} in the session key. We are going to analyse two types of these protocols, one (T-OAKE) with g^{ab} factor and the other (basic OAKE) without g^{ab} in the session key. Their analysis will be sufficient to sketch the proof for the other variants.

4.1.1 OAKE and T-OAKE Protocols

OAKE and T-OAKE protocols are pretty similar to 3DH and HMQV, respectively, in terms of deniability characters. As seen in Figure 4.1, OAKE protocol has three DH values in key computation g^{ay}, g^{bx} and g^{xy}, while T-OAKE additionally has g^{ab}. The protocols are

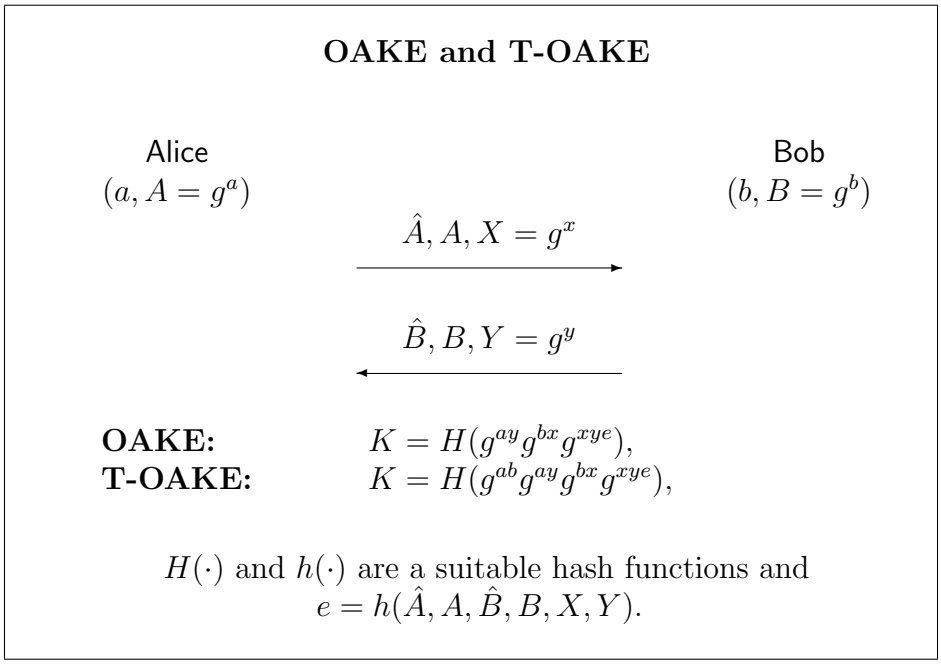

Figure 4.1: OAKE and T-OAKE protocols with session key K

designed in ROM, both functions h and H are random oracles. Regarding key registration, it is noted in [80] that they assume a PKI where no proof-of-knowledge or proof-of-possession is executed by the CA for the long term keys. CA only checks the public keys for sub-group membership, to make sure they are sampled from the correct algebraic group. This operation does not require knowledge of the secret key to perform. As a result, we assume that the deniability simulator is unable to extract the adversary's long-term secret key.

As a side note, both protocols are concurrently deniable with key registration under the KDH assumption in the random oracle model. If we lift the key registration requirement, OAKE is deniable under KDH and T-OAKE is deniable under EKDH assumption in the random oracle model.

The fact that T-OAKE requires EKDH (which is possibly stronger than KDH) is due to the inclusion of g^{ab} in the session key. When we assume no key registration (so that the long-term secret of the adversary cannot be extracted by the simulator) and the session key includes the DH value g^{ab} of long-term keys $A = g^a$ and $B = g^b$, KDH usually is not sufficient

to extract g^{ab}. Because KDH assumes that, following the notation in Definition 3.6.3 KDH, the group member U is available at the time of sampling Z. In order to extract g^{ab}, the parameters U and Z must be set to A and B, (in certain order depending on which party we simulate). However, we cannot assume that Alice's key was available to Bob while Bob is sampling his long-term key, or vice versa. This fact leads us to use EKDH for extracting g^{ab} when key registration is out of the picture.

Theorem 4.1.1. *Under the* $(T_M, T_{\hat{M}}, T_G, \varepsilon_G)$*-KDH Assumption, OAKE is a* $(\tilde{T}_M, \tilde{T}_{\hat{M}}, \tilde{T}_G, \varepsilon_G)$ *deniable AKE in the random oracle model, even without registration of long-term public keys.*

Proof. Consider deniability for the initiator Alice against a malicious responder Bob. We build a simulator which runs on input Alice's public key A, some random coins, auxiliary information aux available to the adversary and eventually generates an output simulating Bob's view, which consists of Bob's random coins, aux, the transcript $\hat{A}, \hat{B}, A, B, X, Y$ and the session key K. Again, the sole challenge for simulation is computing the session key.

Since we assume no key registration SIM is not provided with Bob's long-term secret b.

The simulator SIM activates Bob's machine by inputting aux and random coins r. Then it chooses $x \leftarrow \mathbb{Z}_q$ and computes $X = g^x$. Bob receives id_A, A, X from SIM, and outputs id_B, B, Y. Note that, Bob can be seen as a machine M in the definition of KDH assumption (Definition 3.6.3): it runs on input $U = A$, some auxiliary information $aux||id_A||X$ and coins r. Therefore under KDH, there must be an extractor for Bob.

Remember that the session key of the OAKE protocol is defined as $K = H(g^{ay} g^{xye} g^{bx})$, where H is a random oracle and e is a coefficient computed on the transcript $\hat{A}, \hat{B}, A, B, X, Y$. The simulator can easily compute g^{xye} and g^{bx} with the knowledge of x. In order to compute g^{ay}, it invokes the KDH extractor.

- If the extractor outputs $\hat{Z} = DH(A, Y)$ then SIM sets the key to $K = H(\hat{Z} g^{xye} g^{bx})$, which is the real key.

- If the extractor outputs $\perp$ then SIM sets the key to $K \leftarrow \{0,1\}^n$ where n is the length of the session key.

Note that in the second case, any judge running in time less than T_G will not be able to compute $DH(A, Y)$ and therefore will not be able to query the random oracle in the preimage of the real key. This immediately yields that for this judge a random key is indistinguishable from the real key.

For the deniability with respect to Bob, the proof goes along the same lines. The simulator computes two DH values of the session key g^{ay} and g^{xye} by using y, which it chooses by itself. Then it invokes the KDH extractor to compute the third value g^{bx}, with the parameters $U = B$ and $Z = X$. Because Bob's long-term key B is fixed before X is sampled, the extractor can be executed by setting $U = B$. The simulation of the session key is identical to the proof above.

T-OAKE Protocol. T-OAKE protocol is similar to HMQV in that both protocols involve g^{ab} in the session key, which adds another layer of complexity to the deniability simulation, especially when the adversary's long-term secret is not accessible for extraction during key registration. Note that HMQV with key registration is deniable (Theorem 3.6.4) under KDH, while it is deniable without key registration under EKDH (Theorem 3.8.2). Both results are shown to hold in the random oracle model. The proof for T-OAKE follows closely the proof of Theorem 3.8.2 except for the way the session key is computed.

Theorem 4.1.2. *Under the* $(T_M, T_{\hat{M}}, T_G, \varepsilon_G)$*-EKDH Assumption, T-OAKE is a* $(\tilde{T}_M, \tilde{T}_{\hat{M}}, \tilde{T}_G, \varepsilon_G)$ *deniable AKE in the random oracle model, even without registration of long-term public keys.*

Proof. Recall that the session key of T-OAKE is defined as $K = H(g^{ab} g^{ay} g^{xye} g^{bx})$.

Consider the initiator's deniability, in which the simulator SIM takes on the role of Alice. The simulator is able to compute $g^{xye} g^{bx}$ since it knows x such that $X = g^x$. It computes g^{ab} and g^{ay} by invoking the extractor twice under the EKDH assumption.

The first extraction occurs after Bob outputs his public key $B = g^b$. In accordance with the notation of the EKDH assumption in Definition **??**ekdh, the first extraction is performed on the parameters $Z = B$ and $U = A$. As a result, the extractor returns either A^b or $\perp$.

The second extraction occurs after Bob sends $Y = g^y$. SIM can invoke the EKDH assumption with parameters $Z = Y$ and $U = A$ to obtain either A^y or $\perp$. If either output is $\perp$ the simulator outputs a random session key, otherwise it has extracted the correct key.

Note that KDH assumption suffices for the second extraction, too, since A is fixed by the time Y is sampled by the adversary.

Deniability for the responder follows along the same lines: two extractions are necessary, one for simulating g^{ab} and another for g^{bx}. Similarly, g^{bx} can be obtained by KDH, too.

4.2 Deniability of iAKE protocols combined with Key Confirmation

4.2.1 HMQV with Key Confirmation

Figure 4.2 shows the HMQV protocol with key confirmation, as introduced in [52].

Theorem 4.2.1. *HMQV-C protocol is concurrently deniable with respect to the initiator in the random oracle model under the Knowledge of Exponent Assumption with auxiliary input (Definition 3.9.1) even without registration of long-term public keys.*

Proof. We build a simulator SIM which, on input the public key A of Alice, interacts with a malicious peer Bob, and outputs a simulation of the view of Bob, that is indistinguishable from the real one with respect to any efficient judge.

SIM activates the adversary's machine (Bob) by feeding into random coins r and auxiliary input aux.

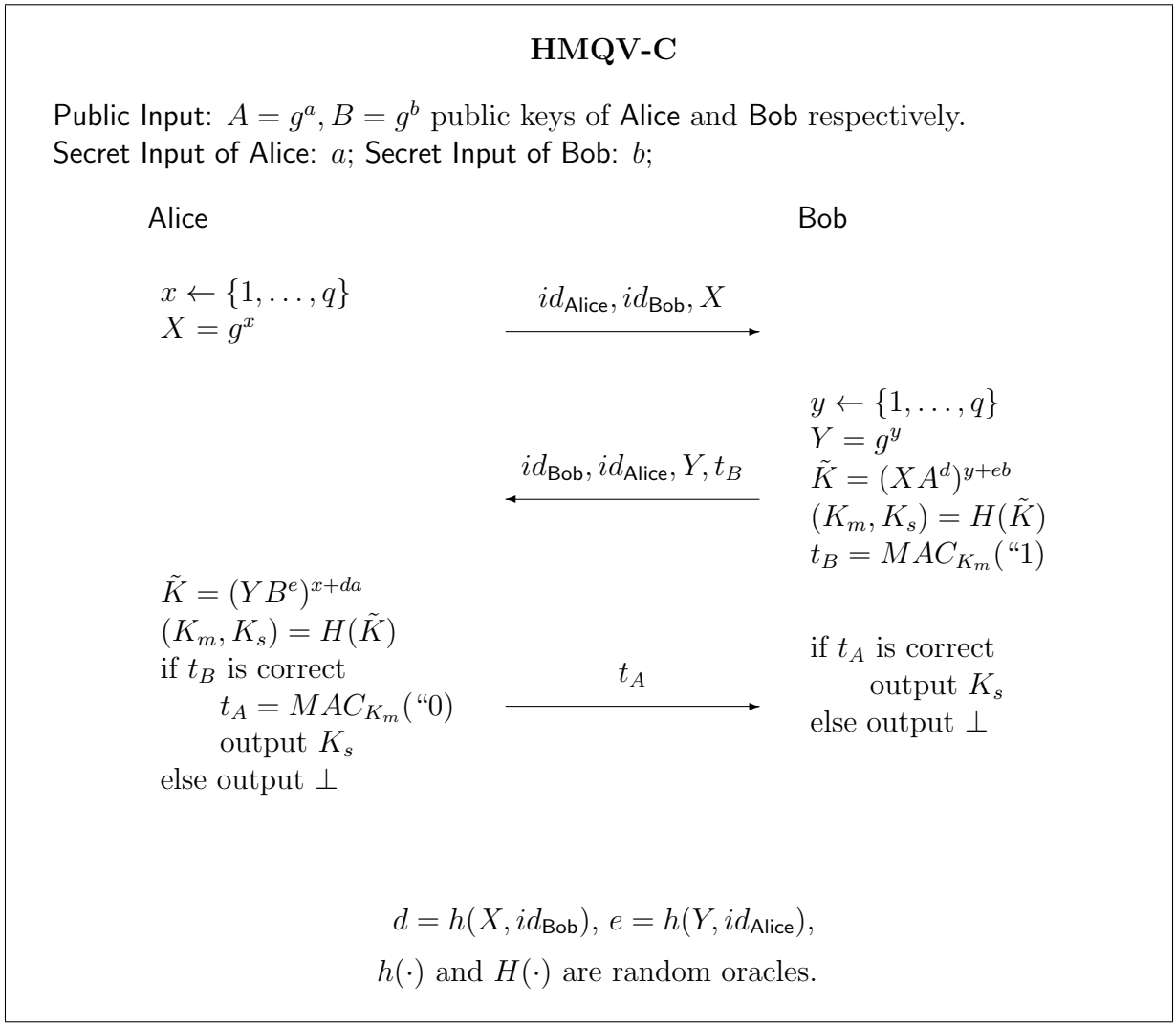

Figure 4.2: HMQV protocol with key confirmation [52].

SIM chooses $x \leftarrow \mathbb{Z}_q$ at random and computes $X = g^x$. After receiving A and X from SIM, Bob may query the random oracle to compute the MAC key and eventually outputs B, Y and t_B. Let us denote the input-output pairs of Bob's RO queries as (α_i, β_i), such that $\beta_i = RO(\alpha_i)$ for the i'th query. The simulator monitors these queries but is not able to modify them (program the oracle).

A careful look at the input-output of Bob makes it clear that we can invoke KEA with aux on Bob in order to extract $(be + y)$, provided that Bob computes it.

Remember that an (honest) responder receives A, X, r, aux and a group generator g as input, queries the random oracle on $\tilde{K} = (A^d X)^{be+y}$ and then outputs B, Y, t_B. We can define a sequence of machines to show that the responder can be viewed as a machine which outputs $g^{be+y}, (A^d X)^{be+y}$ on input $g, (A^d X), r, aux$. Hence, there will be a KEA extractor returning $(be + y)$.

Let us define the sequence of machines as follows. Assume that there is a machine μ_1 enclosing Bob, receiving A, X, aux, r, g as input and invokes another machine μ_2 inside, which is also enclosing Bob. On input $(g, A^d X, r, aux')$, with $aux' = aux||A, X$, μ_2 runs Bob on g, A, X, r, aux. After a series of RO queries $\{(\alpha_i, \beta_i)\}_i$, Bob outputs B, Y and t_B. The machine μ_2 outputs $(B^e Y, (A^d X)^{be+y}, aux'')$, where $aux'' = (B, Y, t_B)$. Finally μ_1 returns B, Y, t_B as specified by the protocol. The runtimes of these machines differ by a constant number of operations. KEA with auxiliary input can be invoked on μ_2 to extract $(be + y)$.

The simulation continues according to Bob's queries and the output of the extractor, following one of the cases below.

If Bob outputs his round message without making any RO query, then SIM aborts the session and sets the session key to $\perp$ in the simulation output **(Case 1)**. Otherwise, SIM runs the KEA extractor which returns γ. SIM verifies the result by checking whether $g^\gamma = B^e Y$. If correct, SIM sets $\tilde{K}$ to $(A^d X)^\gamma$ and follows the specified protocol **(Case 2)**. If $g^\gamma \neq B^e Y$, SIM aborts the session and sets the session key $K_s = \perp$ in the simulation output **(Case 3)**.

At the end of the simulation, SIM returns $(A, B, X, Y, t_A, t_B, K_s, r, aux)$, by setting the value of $t_A = \perp$ whenever it aborts before the final round.

Note that unless Bob queries the RO on $DH(A^d X, B^e Y)$, there is no way that Bob knows the real MAC key K_m (corresponding to A, B, X, Y) beside guessing it right or finding the value within aux.

In order to eliminate the possibility that Bob sends a valid t_B without querying the RO, we may set out the simulator to choose x such that no tuple $(A, B, X, \cdot)$ appears in aux for $X = g^x$.

In Case 1, SIM outputs $(A, B, X, Y, t_A = \perp, t_B, K_s = \perp, r, aux)$. Note that the simulation differs from a real conversation in this case only when t_B is computed correctly. (When t_B is assigned an incorrect value, the initiator is expected to abort, just as SIM does.) The probability that Bob sends out a correct MAC tag t_B without querying the random oracle is exponentially small in the order of the group G. Therefore the probability of failure for the simulation is negligible in Case 1.

In cases 2 and 3, Bob sends B, Y, t_B after querying the random oracle. The KEA extractor for Bob returns γ, which can be verified for correctness by SIM. A valid γ value results in a perfect simulation in Case 2. However, an invalid γ may appear (Case 3) as a consequence of an extractor failure or Bob's not querying the RO on the real Diffie-Hellman value $DH(A^d X, B^e Y)$.

In Case 3, SIM outputs $(A, B, X, Y, t_A = \perp, t_B, K_s = \perp, r, aux)$ and it differs from a real conversation only when Bob sends the correct t_B but KEA extractor fails. Extractor failure happens with probability at most ε_{KEA} and Bob is able to guess the correct t_B with an exponentially small probability in the order of the group G.

We have shown that the judge, as an outsider, can detect the above simulation only with $\varepsilon_{KEA}+$ a negligible probability.

Deniability for Responder in HMQV-C.

For the responder's deniability, KEA in the random oracle model is not sufficient for HMQV-C, regardless of key registration. Consider the deniability simulator in the role of Bob. After receiving A and X from the adversary Alice, SIM is expected to output the MAC t_B. With key registration, SIM needs assistance with computing g^{bx}, without key registration computing both g^{bx} and g^{ab} is challenging. Since the only value Alice outputs at this point is X, KEA or Key-Awareness [28] assumptions are not useful. We find ourselves at the same situation with regular HMQV, therefore we need to resort to KDH assumption (assuming key registration) or EKDH assumption (without key registration).

Since computation of MAC key involves not only ephemeral keys as in SIGMA, but also the long-term keys, auxiliary information which involves transcripts from other sessions in the network may not help either.

Attempts to mitigate the problem by adding a fourth step to the protocol are ineffective. (Though the cost of the fourth round does not appear to be a realistic trade-off for stepping down from KDH to KEA.) This attempt only shifts the center of the problem: Same situation is experienced on the side of the initiator this time. As a result, we can conclude that HMQV-C still requires KDH or EKDH to ensure responder's deniability.

4.3 Observations on Extractability, Obfuscation and Time-Based Cryptography in relation to Deniability

4.3.1 Obfuscation and KDH Assumptions

The KDH assumption family that we use to prove our deniability results require that for any adversary running the key exchange protocol, there exists an extractor that will yield some internal state of the adversary. Here we briefly remark on how our assumptions differ from the notion of *extractable one-way function* introduced in [10].

If F is an extractable one-way function, for every adversary $\mathcal{A}$ that outputs $y = F(x)$, there exists an extractor $\mathcal{E}$ that outputs $x' \in F^{-1}(y)$. The main result in [10] is that in the case in which $\mathcal{A}$ and $\mathcal{E}$ have the same common auxiliary input, then extractable one-way functions do not exists, under the assumption of the existence of indistinguishability obfuscation (iO) for certain classes of circuits. Given recent results [43] that establish the existence of iO for any circuit under reasonable assumptions, then common-auxiliary extractable functions should not be considered as a sound assumption.

We point out that deniability proofs based on knowledge extraction require the adversary and the extractor to have a common auxiliary input (since the deniability simulator must be a "real-life" simulator, which has the same "knowledge" as the adversary[1]). Therefore there seems to be little hope to establish a deniability simulations based on extractable one-way functions. This would seem to doom our approach.

Yet our assumptions are not equivalent to extractable one-way functions and indeed require something weaker. Informally our extractor $\mathcal{E}$ will either output $x' \in F^{-1}(y)$ or fail,

[1] The analogy is allowing the simulator to have the trapdoor associated with a common reference string –while that is OK for zero-knowledge simulation, it does not provide a proof of real-life plausible deniability.

and in the latter case the assumption requires that it is easy to sample a distribution which is computationally indistinguishable from $F^{-1}(y)$. This in turns implies that our assumption are not affected by the negative result in [10].

4.3.2 Obfuscation and Online Deniability Attacks to Signal

Inspecting the interaction between extractability and obfuscation has led us to interesting examples worth considering further in the context of deniability. These examples are motivated by a desire to investigate to what extent the obfuscation and time-based cryptography are useful for staging an attack similar to the "bad sampling strategy" described in Section 3.5.

We observe that, with the introduction of obfuscation into the setting, bad-sampling strategy transforms into an online attack. Both obfuscation and time-lock puzzles (TLP) are useful in staging online attacks for enabling an adversarial participant in a session to prove her/his lack of knowledge of an ephemeral secret. Additionally, these tools eliminate the need for a third party (judge) to be online during the protocol's execution in order to collude with the malicious party. Compared to the known online attack to Signal [75], these examples may demonstrate a stronger impact in terms of number of parties to be convinced and the time period for an assisting party must be online in order to break deniability.

Let us first present the attack scenarios and then discuss why we classify them as online deniability attacks.

For the attack scenarios, consider a third party who offers online services for the purpose of undermining deniability of a particular protocol, such as Signal. Let us call the third party as *Wikileaks*. We assume that Wikileaks has its own digital signature keys.

Example 4.3.1 (**Wikileaks commits and opens the commitment**)**.** Wikileaks commits to a PRF key K everyday and opens the commitments 10 days later. Wikileaks gives access

to $\mathrm{PRF}_K(\cdot)$ publicly, where anyone can query the oracle on a point x and receives $PRF_K(x)$ in return. As a proof of use of this service, Wikileaks signs the result and also provides a NIZK proof that output generated by $\mathrm{PRF}_K(\cdot)$. If a malicious user Alice wants to frame Bob, she takes the newspaper NYT of the day and queries the Wikileaks's PRF oracle, which uses that day's freshly committed key K. Alice uses the response $PRF_K(NYT)$ as her ephemeral (Diffie-Hellman) public key X. After the commitments open Alice can prove that she does not know her ephemeral secret, because she computed the ephemeral key through Wikileaks's PRF service.

Note that in order for this strategy to function as an attack, Alice must have a way to demonstrate that the session key K corresponds to the protocol's transcript (A, B, X, Y).

What distinguishes it from the online attack for Signal in [75]? In the online attack, the judge generates the proof, which convinces only the judge. In Example 1, now that the evidence convinces everyone, i.e., anyone can verify Wikileaks's signature on values and NIZK proof to confirm that the interaction occurred. As long as the third party offers this service, this will be an effective way to circumvent deniability.

Time-Lock Puzzles

Time-lock puzzles (TLP) are cryptographic objects which are designed to keep an information hidden behind the lock of intractability of a hard problem. By tuning the difficulty of the intractable problem, one can ensure that the hidden information will stay safe for a certain amount of time T and it will be released only after a continuous computation power for time T will be exerted on the puzzle to break the lock.

A basic example of TLP due to [67] works as follows.

The puzzle creator Alice wants to encrypt a message M with a TLP for a period of T seconds. Alice first chooses a key K for an appropriate encryption system and encrypts M under K: $C_M = Enc_K(M)$ and then creates a TLP to hide the key K based on squre root

modulo a composite n with an unknown factorization. That is, Alice chooses a composite number n by randomly picking two large primes p and q.

$$n = pq$$

Then she picks a random a modulo n and encrypts K as

$$C_K = K + a^{2^t} \mod n,$$

where t is the number of squarings modulo n that takes at least time T from the solver. Finally Alice outputs the puzzle (n, a, t, C_M, C_K) and removes any other values p, q which allows solving the puzzle in less than time T.

Note that, by the design of the TLP it must be infeasible to apply a brute-force search on the puzzle to find out K in time shorter than T.

Example 4.3.2 (**Wikileaks generates a time-lock puzzle**)**.** For the attack scenario, assume there is an agent Wikileaks publicly offering access to the following oracle: Upon an arbitrary value from the user, the oracle generates a pseudorandom string and uses it to choose $y \in G$ and random values (such as p, q, a) to generate a TLP. The oracle encrypts y with the TLP and outputs $Y = g^y$ and the TLP tuple, both signed under Wikileaks' signature. (Assume that a brute force search on Y takes longer time on average than solving the puzzle.)

A malicious party Bob, willing to frame Alice, submits a publicly known text, NYT, to the oracle as input and receives the corresponding Y and TLP tuple. Bob runs the protocol with Alice by submitting Y as his ephemeral public key.

Using NYT as input, Bob is able to prove anyone that he does not know the ephemeral secret $y = \log Y$ starting from the day of the newspaper and the following period of time T.

Bob can also prove after solving the puzzle that the session key used in Alice's messages is the actual key.

Example 2 has the same advantages that Example 1 have over the original online attack. Additionally, this time Wikileaks only needs to be online at the time of generating the puzzle (rather than being online also at the time of opening the commitment).

Example 4.3.3 (**Wikileaks generates an obfuscated oracle**)**.** Imagine that Wikileaks obfuscates the oracle described in Example 2 and presents the obfuscated program to public use. The oracle receives an arbitrary input and returns Y and time-lock puzzle, appended with a signature.

Again, Bob can frame his peer Alice by choosing his ephemeral key as described in Example 2. Unlike to the Example 2, this strategy permanently destroys (online) deniability. It is sufficient to generate such a program once by a trusted party to break deniability once and for all.

Conclusion. In order to claim that the above examples circumvent the deniability of the relevant protocols, we need to place trust on Wikileaks, that Wikileaks will not collude with Bob and reveal the concealed value $y = \log Y$ to him.

We introduced Wikileaks into the picture, because saying that "Bob uses an obfuscated oracle or a time-lock puzzle for choosing his ephemeral key" leads us to the question that who created this oracle, and do we trust the creator that hidden information is not shared with Bob (i.e., creator colludes with Bob)?

Once a trusted party is added into the attack scenario, the setup appears pretty similar to the online attack in [75]. The difference is that with the use of TLP and obfuscation, the requirement for the judge to be online during the session has been relaxed to different degrees.

Chapter 5

Conclusion and Future Directions

Our study shows why a meaningful deniability proof for Signal may be difficult to construct. We presented a constructive counterexample for the MQV protocol consisting of mathematical groups where running MQV results in a provably non-deniable interaction. This MQV counter-example illuminates the core problems one encounters in trying to prove the deniability of 3DH, the AKE underlying Signal.

We showed that a deniability proof for 3DH (and MQV and HMQV) can be built from a strong "knowledge assumption" which seems to be inherently "tight", as it is defined as the logical negation of the strategy required for making this type of AKE non-deniable. On the other hand, however, this assumption is almost tautological as it basically assumes the existence of an extractor which is the required deniability simulator.

Our counter-examples and this problematic assumption and proof, underscores the need to be critically re-examining our belief that Signal can be *formally proven* to be deniable, and accepting instead that we are just assuming it is.

Our research is the first to examine the offline deniability of 2-message iAKE protocols when malicious adversaries are present. The analysis revealed the counter-intuitive result that, despite the fact that the iAKE protocols exchange minimal data between peers, a

provable deniability required much stronger assumptions.

The communication content of the protocols addressed in this work contains only long-term and ephemeral public keys that are transmitted independently. If we attempt to add confirmation step to these protocols or useencryption or signature in order to provide explicit authentication, we will end up with protocols like SKEME, SIGMA and HMQV-C, whose deniability is proven under weaker assumptions than the KDH family; such as key-awareness, plaintext-awareness and KEA. This situation appears to be a consequence of the intricate nature of deniability as a concept, which is orthogonal to authentication.

Many deniability proofs in the literature use a knowledge extractor and produce a perfect or statistically indistinguishable simulation. The question that naturally arises is whether it is possible to achieve deniability under weaker assumptions.

We attempted, in an unpublished effort, to modify these 2-message iAKE protocols with a minimal means of achieving deniability under weaker assumptions. A ZK proof appears to be a natural candidate for augmenting the protocol in order to prevent the attack wherein a party can randomly sample its ephemeral key, in particular without knowing the corresponding secret value. This modification requires peers to demonstrate (in zero-knowledge) that they possess the ephemeral secret value. In order to preserve the 2-message structure, we preferred an NIZK proof. In addition, the resulting protocol can support asynchronous communication because the message of the initiator can be made independent of the identity of the responder.

In the simulation of deniability, we must extract the secret keys corresponding to the ephemeral values without rewinding or programming the random oracle [61]. Therefore, we must assume that only parties with knowledge of the secret value can generate correct NIZK proofs. As with key- or plaintext-awareness, this can be modeled utilizing a companion extractor machine. In other words, the presence of an NIZK proof still necessitates an additional knowledge extraction assumption in order to successfully complete the simulation.

Hashimoto et al. [42] presented a similar specific case in which they introduced a post-quantum secure key exchange protocol that satisfies Signal's essential characteristics, namely asynchronous communication, 2-message round complexity, deniability, and forward secrecy. The protocol is constructed with KEM and ring signatures, and it has been demonstrated to be offline-deniable without any knowledge assumptions against only semi-honest adversaries (who are following the protocol honestly). To achieve deniability against malicious adversaries, they modified this protocol by augmenting an NIZK proof to the protocol. The resultant protocol has been proven to be deniable in this sense, assuming the KEM holds the plaintext-awareness property.

It appears that with the iAKE protocols of the aforementioned Signal properties, knowledge extraction may be intrinsic to achieving deniability against malicious adversaries. Conversely, there are examples of key exchange protocols that provide stronger deniability assurances without relying on knowledge assumptions. These are the online deniable protocols from [31, 78, 74, 75].Nonetheless, these protocols are susceptible to KCI attacks, a fact also highlighted by [42]. Therefore, a security compromise may be inevitable as a result of avoiding knowledge assumptions.

A natural open question in light of the preceding examples is whether a knowledge assumption is required for the iAKE protocol with the aforementioned properties. A stronger statement, such as demonstrating that a non-black box technique is required for offline deniability simulations, could be a viable direction, as it may be possible to prove it using the existing non-black box impossibility results from the literature.

The online attacks that are carried out with the aid of a third-party service are the subject of an interesting discussion regarding Signal's deniability. There are a few instances in the literature that make use of various cryptographic objects, such as [31, 78] presenting an example with append-only bulletin boards, [38] presenting a similar attack utilizing a trusted execution environment and remote attestation service, and [75] presenting an instance in

which the malicious protocol participant selectively discloses any part of the conversation after the session. In Section 4.3.2, we present example attack strategies utilizing obfuscation and time-lock puzzles.

Utilizing a third-party service to generate irrefutable logs of protocol communication in real time is a characteristic shared by all these attacks. The credibility of the logs depends on the trust placed in the third-party service, typically through the use of a signature. Creating non-repudiable protocol records during the session appears to contradict the definition of offline deniability. Offline deniability is achieved by taking advantage of the fact that evidences of participation in a communication session can be fabricated after the fact by exploiting the knowledge/view gap between the judge and protocol participants. In other words, protocol participants are able to cast doubt on the evidences of their involvement by lying to the judge. However, the participants lose the ground for lying when the communication is recorded verbatim with the help of a third party.

Let's compare these attacks to one in which an online judge colludes with Bob to frame Alice during the session by choosing Bob's ephemeral secret. [1] We can see that an online judge who colludes with Bob may end up with non-transferable evidence, whereas the previously described online attacks are capable of producing publicly (or selectively) verifiable and transferable evidence of Alice's participation in the session. Different computational assumptions are required to generate evidences with different characteristics (transferable or publicly verifiable). Understanding these trade-offs may provide us with a more complete view of online deniability.

Finally, we know that online deniability retains composability property due to the definition in the UC framework [78, 31]. The online deniable protocols satisfying this definition are resistant to the aforementioned third-party-assisted attacks. When an offline deniable

[1] This attack has been introduced in Section A.1.2 of [75], showing that Signal is not online deniable.

protocol is combined with arbitrary other protocols, however, offline deniability may not be preserved. Beside composability, the inclusion of auxiliary information in the definition of offline deniability requires careful consideration. An example of this situation manifests itself in the discussion about extractable OWF in Section 4.3. In other words, there are aspects of the offline deniability definition that must be explored in order to comprehend the limitations of offline deniability.

Bibliography

[1] J. Alwen, S. Coretti, and Y. Dodis. The double ratchet: Security notions, proofs, and modularization for the signal protocol. *IACR Cryptology ePrint Archive*, 2018:1037, 2018.

[2] R. Barnes, B. Beurdouche, R. Robert, J. Millican, E. Omara, and K. Cohn-Gordon. The Messaging Layer Security (MLS) Protocol. Internet-Draft draft-ietf-mls-protocol-13, Internet Engineering Task Force, Mar. 2022. Work in Progress.

[3] M. Bellare, R. Canetti, and H. Krawczyk. A modular approach to the design and analysis of authentication and key exchange protocols (extended abstract). In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, pages 419–428, New York, NY, USA, 1998. ACM.

[4] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In *Proceedings of the 18th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '98, pages 26–45, London, UK, UK, 1998. Springer-Verlag.

[5] M. Bellare and A. Palacio. The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, pages 273–289, 2004.

[6] M. Bellare and P. Rogaway. Entity authentication and key distribution. In D. R. Stinson, editor, *Advances in Cryptology — CRYPTO' 93*, pages 232–249, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

[7] M. Bellare and P. Rogaway. Optimal asymmetric encryption – how to encrypt with rsa. In *Advances in Cryptology - Eurocrypt '94 Proceedings*, volume 950 of *Lecture Notes in Computer Science*, 1995.

[8] D. Bernhard, M. Fischlin, and B. Warinschi. Adaptive proofs of knowledge in the random oracle model. In J. Katz, editor, *Public-Key Cryptography – PKC 2015*, pages 629–649, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[9] D. Bernhard, M. Fischlin, and B. Warinschi. On the hardness of proving cca-security of signed elgamal. In C.-M. Cheng, K.-M. Chung, G. Persiano, and B.-Y. Yang, editors, *Public-Key Cryptography – PKC 2016*, pages 47–69, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[10] N. Bitansky, R. Canetti, O. Paneth, and A. Rosen. On the existence of extractable one-way functions. *SIAM J. Comput.*, 45(5):1910–1952, 2016.

[11] N. Borisov, I. Goldberg, and E. Brewer. Off-the-record communication, or, why not to use pgp. In *Proceedings of the 2004 ACM Workshop on Privacy in the Electronic Society*, WPES '04, pages 77–84, New York, NY, USA, 2004. ACM.

[12] C. Boyd, W. Mao, and K. G. Paterson. Deniable authenticated key establishment for internet protocols. In *Proceedings of the 11th International Conference on Security Protocols*, page 255271, Berlin, Heidelberg, 2003. Springer-Verlag.

[13] C. Boyd, W. Mao, and K. G. Paterson. Key agreement using statically keyed authenticators. In M. Jakobsson, M. Yung, and J. Zhou, editors, *Applied Cryptography and Network Security*, pages 248–262, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[14] J. Brendel, R. Fiedler, F. Günther, C. Janson, and D. Stebila. Post-quantum asynchronous deniable key exchange andthesignal handshake. In G. Hanaoka, J. Shikata, and Y. Watanabe, editors, *Public-Key Cryptography – PKC 2022*, pages 3–34, Cham, 2022. Springer International Publishing.

[15] J. Brendel, M. Fischlin, F. Günther, C. Janson, and D. Stebila. Towards post-quantum security for signals x3dh handshake. In *Selected Areas in Cryptography: 27th International Conference, Halifax, NS, Canada (Virtual Event), October 21-23, 2020, Revised Selected Papers*, page 404430, Berlin, Heidelberg, 2020. Springer-Verlag.

[16] R. Canetti, C. Dwork, M. Naor, and R. Ostrovsky. Deniable encryption. In B. S. Kaliski, editor, *Advances in Cryptology — CRYPTO '97*, pages 90–104, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

[17] R. Canetti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In B. Pfitzmann, editor, *Advances in Cryptology — EUROCRYPT 2001*, pages 453–474, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[18] R. Canetti and H. Krawczyk. Security analysis of ike's signature-based key-exchange protocol. In M. Yung, editor, *Advances in Cryptology — CRYPTO 2002*, pages 143–161, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[19] R. Canetti and H. Krawczyk. Universally composable notions of key exchange and secure channels. In L. R. Knudsen, editor, *Advances in Cryptology — EUROCRYPT 2002*, pages 337–351, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[20] K. Cohn-Gordon, C. Cremers, B. Dowling, L. Garratt, and D. Stebila. A formal security analysis of the signal messaging protocol. In *2017 IEEE European Symposium on Security and Privacy (EuroS P)*, pages 451–466, 4 2017.

[21] C. Cremers and M. Feltz. One-round strongly secure key exchange with perfect forward secrecy and deniability. Cryptology ePrint Archive, Report 2011/300, 2011. `https://eprint.iacr.org/2011/300`.

[22] I. Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In J. Feigenbaum, editor, *Advances in Cryptology — CRYPTO '91*, pages 445–456, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg.

[23] A. W. Dent. The cramer-shoup encryption scheme is plaintext aware in the standard model. Cryptology ePrint Archive, Report 2005/261, 2005. `https://eprint.iacr.org/2005/261`.

[24] A. W. Dent and S. D. Galbraith. Hidden pairings and trapdoor ddh groups. In F. Hess, S. Pauli, and M. Pohst, editors, *Algorithmic Number Theory*, pages 436–451, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[25] M. Di Raimondo and R. Gennaro. New approaches for deniable authentication. *Journal of Cryptology*, 22(4):572–615, 10 2009.

[26] M. Di Raimondo, R. Gennaro, and H. Krawczyk. Secure off-the-record messaging. In *Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society*, WPES '05, pages 81–89, New York, NY, USA, 2005. ACM.

[27] M. Di Raimondo, R. Gennaro, and H. Krawczyk. Secure off-the-record messaging. In *Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society*, WPES 05, page 8189, New York, NY, USA, 2005. Association for Computing Machinery.

[28] M. Di Raimondo, R. Gennaro, and H. Krawczyk. Deniable authentication and key exchange. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS '06, pages 400–409, New York, NY, USA, 2006. ACM.

[29] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Trans. Inf. Theor.*, 22(6):644–654, Sept. 2006.

[30] S. Dobson and S. D. Galbraith. Post-quantum signal key agreement with sidh. Cryptology ePrint Archive, Report 2021/1187, 2021. `https://ia.cr/2021/1187`.

[31] Y. Dodis, J. Katz, A. Smith, and S. Walfish. Composability and on-line deniability of authentication. In O. Reingold, editor, *Theory of Cryptography*, pages 146–162, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[32] D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. In *Proceedings of the Twenty-Third Annual ACM Symposium on Theory of Computing*, STOC '91, page 542552, New York, NY, USA, 1991. Association for Computing Machinery.

[33] C. Dwork, M. Naor, and A. Sahai. Concurrent zero-knowledge. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, pages 409–418, New York, NY, USA, 1998. ACM.

[34] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology CRYPTO' 86*, volume 263 of *Lecture Notes in Computer Science*, page 186194, 1987.

[35] M. Fischlin and S. Mazaheri. Notions of deniable message authentication. In *Proceedings of the 14th ACM Workshop on Privacy in the Electronic Society*, WPES '15, pages 55–64, New York, NY, USA, 2015. ACM.

[36] S. Goldwasser and S. Micali. Probabilistic encryption. *JCSS*, 28(2):270–299, 4 1984.

[37] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, Feb. 1989.

[38] L. J. Gunn, R. V. Parra, and N. Asokan. Circumventing cryptographic deniability with remote attestation. Cryptology ePrint Archive, Report 2018/424, 2018. `https://ia.cr/2018/424`.

[39] S. Hada and T. Tanaka. On the existence of 3-round zero-knowledge protocols. *IACR Cryptol. ePrint Arch.*, 1999:9, 1999.

[40] D. Harkins and D. Carrel. The internet key exchange (ike), 1998.

[41] D. Harkins and D. Carrel. The internet key exchange (ike). RFC 2409, RFC Editor, 11 1998.

[42] K. Hashimoto, S. Katsumata, K. Kwiatkowski, and T. Prest. An efficient and generic construction for signal's handshake (x3dh): Post-quantum, state leakage secure, and deniable. Cryptology ePrint Archive, Report 2021/616, 2021. `https://ia.cr/2021/616`.

[43] A. Jain, H. Lin, and A. Sahai. Indistinguishability obfuscation from well-founded assumptions. *IACR Cryptol. ePrint Arch.*, 2020:1003, 2020.

[44] M. Jakobsson, K. Sako, and R. Impagliazzo. Designated verifier proofs and their applications. In U. Maurer, editor, *Advances in Cryptology — EUROCRYPT '96*, pages 143–154, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg.

[45] S. Jiang. Timed encryption and its application. Cryptology ePrint Archive, Report 2010/546, 2010. `https://ia.cr/2010/546`.

[46] M. Jones and D. Hardt. The oauth 2.0 authorization framework: Bearer token usage. RFC 6750, RFC Editor, 10 2012.

[47] J. Katz. Efficient and non-malleable proofs of plaintext knowledge and applications. In E. Biham, editor, *Advances in Cryptology — EUROCRYPT 2003*, pages 211–228, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[48] C. Kaufman. Internet key exchange (ikev2) protocol. RFC 4306, RFC Editor, 12 2005.

[49] H. Krawczyk. Skeme: a versatile secure key exchange mechanism for internet. In *Proceedings of Internet Society Symposium on Network and Distributed Systems Security*, pages 114–127, 2 1996.

[50] H. Krawczyk. Skeme: a versatile secure key exchange mechanism for internet. In *Proceedings of Internet Society Symposium on Network and Distributed Systems Security*, pages 114–127, Feb 1996.

[51] H. Krawczyk. Sigma: The 'sign-and-mac' approach to authenticated diffie-hellman and its use in the ike protocols. In D. Boneh, editor, *Advances in Cryptology - CRYPTO 2003*, pages 400–425, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[52] H. Krawczyk. Hmqv: A high-performance secure diffie-hellman protocol. In V. Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, pages 546–566, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[53] C. Kudla and K. G. Paterson. Modular security proofs for key agreement protocols. In B. Roy, editor, *Advances in Cryptology - ASIACRYPT 2005*, pages 549–565, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[54] L. Law, A. Menezes, M. Qu, J. Solinas, and S. Vanstone. An efficient protocol for authenticated key agreement. *Designs, Codes and Cryptography*, 28(2):119–134, 3 2003.

[55] W. Mao and K. Paterson. On the plausible deniability feature of internet protocols. Manuscript.

[56] M. Marlinspike. Simplifying otr deniability. `https://signal.org/blog/simplifying-otr-deniability/`, 2013.

[57] M. Marlinspike and T. Perrin. The x3dh key agreement protocol, 11 2016. Rev. 1.

[58] T. Matsumoto, Y. Takashima, and H. Imai. On seeking smart public-key distribution systems. *IEE Proc.*, vol.133, pt.F:pp.224–231, 02 2014.

[59] A. Menezes, M. Qu, and S. Vanstone. Some new key agreement protocols providing implicit authentication. In *Workshop on Selected Area in Cryptography (SAC'95)*, pages 22–32, 1995.

[60] M. Naor. Deniable ring authentication. In *Proceedings of the 22Nd Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '02, pages 481–498, Berlin, Heidelberg, 2002. Springer-Verlag.

[61] R. Pass. On deniability in the common reference string and random oracle model. In D. Boneh, editor, *Advances in Cryptology - CRYPTO 2003*, pages 316–337, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[62] T. Perrin and M. Marlinspike. The double ratchet algorithm, 11 2016. Rev. 1.

[63] D. H. Phan and D. Pointcheval. On the security notions for public-key encryption schemes. In C. Blundo and S. Cimato, editors, *Security in Communication Networks*, pages 33–46, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[64] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000.

[65] C. Rackoff and D. R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In J. Feigenbaum, editor, *Advances in Cryptology — CRYPTO '91*, pages 433–444, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg.

[66] R. L. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In *Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*, ASIACRYPT '01, pages 552–565, Berlin, Heidelberg, 2001. Springer-Verlag.

[67] R. L. Rivest, A. Shamir, and D. A. Wagner. Time-lock puzzles and timed-release crypto. Technical report, USA, 1996.

[68] C. Schnorr. Efficient signature generation by smart cards. *J. Cryptology*, 4(3):161–174, 1991.

[69] Y. Seurin. New constructions and applications of trapdoor ddh groups. In K. Kurosawa and G. Hanaoka, editors, *Public-Key Cryptography – PKC 2013*, pages 443–460, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[70] Y. Seurin and J. Treger. A robust and plaintext-aware variant of signed elgamal encryption. In *Proceedings of the 13th International Conference on Topics in Cryptology*, CT-RSA13, page 6883, Berlin, Heidelberg, 2013. Springer-Verlag.

[71] V. Shoup. On formal models for secure key exchange. Technical Report RZ 3120, IBM, 4 1999.

[72] V. Shoup and R. Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. In K. Nyberg, editor, *Advances in Cryptology — EUROCRYPT'98*, pages 1–16, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[73] Signal technical information. `https://signal.org/docs/`.

[74] N. Unger and I. Goldberg. Deniable key exchanges for secure messaging. *Proceedings on 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1211–1223, 2015.

[75] N. Unger and I. Goldberg. Improved strongly deniable authenticated key exchanges for secure messaging. *Proceedings on Privacy Enhancing Technologies*, 2018(1):21–66, 2018.

[76] N. Vatandas, R. Gennaro, B. Ithurburn, and H. Krawczyk. On the cryptographic deniability of the signal protocol. In *Applied Cryptography and Network Security: 18th International Conference, ACNS 2020, Rome, Italy, October 1922, 2020, Proceedings, Part II*, page 188209, Berlin, Heidelberg, 2020. Springer-Verlag.

[77] N. Vatandas, R. Gennaro, B. Ithurburn, and H. Krawczyk. On the deniability of signal communications. Cryptology ePrint Archive, 2020. `https://eprint.iacr.org/2021/642`.

[78] S. Walfish. Enhanced security models for network protocols, 2008. PhD thesis.

[79] A. C. Yao and Y. Zhao. Deniable internet key exchange. In *Proceedings of the 8th International Conference on Applied Cryptography and Network Security*, ACNS'10, pages 329–348, Berlin, Heidelberg, 2010. Springer-Verlag.

[80] A. C. Yao and Y. Zhao. Oake: A new family of implicitly authenticated diffie-hellman protocols. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security*, CCS '13, pages 1113–1128, New York, NY, USA, 2013. ACM.

[81] A. C.-C. Yao and Y. Zhao. Oake: A new family of implicitly authenticated diffie-hellman protocols. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security*, CCS '13, page 11131128, New York, NY, USA, 2013. Association for Computing Machinery.

[82] A. C.-C. Yao and Y. Zhao. Privacy-preserving authenticated key-exchange over internet. *IEEE Transactions on Information Forensics and Security*, 9(1):125–140, 2014.