On the Cryptographic Deniability of the Signal Protocol

Nihal Vatandas[1], Rosario Gennaro[1], Bertrand Ithurburn[1], and Hugo Krawczyk[2]

[1] Center for Algorithms and Interactive Scientific Software. The City College of New York. Email: nihal.vatandas@gmail.com (contact author), rosario@ccny.cuny.edu, bithurburn@gmail.com

[2] Algorand Foundation. Email: hugokraw@gmail.com

Abstract. *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 paper, we present what we believe is 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 the paper by showing two additional positive results. The first is 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.

1 Introduction

The ever growing privacy concerns and vulnerabilities behind online interactions have made the *deniability of communications* a prime privacy requirement. The goal is to ensure that the transcript of an online communication between two peers cannot be used for proving to a third party that the transcript corresponds to a communication between the two. This property should hold even if one of

the parties is willing to deviate from the protocol just to generate a proof that reveals the identity of the peer to a third party.

A well-known fact is that communications protected by a shared symmetric key do not allow one of the peers to frame the other: Any information Bob claims to have been authenticated by Alice with a shared key could have been produced by Bob himself since he also knows the key. However, usually shared keys are generated via an online authenticated key exchange (AKE) protocol, where the parties authenticate using their own private/public keys. Then, the question is whether the shared key computed in that protocol can itself be denied. This gives rise to the notion of *deniable authenticated key exchange*. As above, the transcript of the AKE protocol should leave no proof of communication that can convince a third party. Bob should not be able to convince a *judge* that Alice communicated with him, *even if Bob is malicious* and departs from the prescribed AKE protocol just to generate such proof.

Deniability of AKE was first formally defined in [16] in terms of simulatability (following an approach set forth by [19] in the context of deniable authentication). Roughly, an AKE protocol π between peers Alice and Bob is *deniable for Alice* if the communication transcript generated in an interaction between Alice and Bob in executions of π can be simulated by Bob without Alice's participation. That is, one requires the existence of a simulator that in interactions with Bob generates transcripts that are computationally indistinguishable from real transcripts between Alice and Bob. Here, Alice's actions are those of an honest AKE participant but Bob can deviate from the protocol in arbitrary ways (if that helps him frame Alice as discussed above). Moreover, and crucially, the simulator needs to *output a session key* under a distribution that is indistinguishable from that of the session key in a real protocol between Alice and (possibly malicious) Bob.

We note that deniability of an AKE protocol may be one-sided, namely, it may be deniable for the initiator but not for the responder, or vice versa. Different flavors of AKE deniability are obtained by considering the type of judge to which a proof is presented. The basic case is that of *offline deniability* that assumes a judge who is not present during the communication between the peers. In this case, the judge examines a transcript presented by Bob, after the alleged communication took place, to decide whether Alice knowingly communicated with Bob. If, in contrast, we let the judge be an active participant in the protocol, ensuring deniability is harder, or even impossible. In particular, if Bob provides his private keys to the judge, the latter can interact with Alice masquerading as Bob and get convinced directly of Alice's participation. The notion of *online deniability* [47] refers to a judge who participates in the protocol but is not given Bob's private keys. The offline case is of more practical relevance since a protocol that is *not deniable* in this sense can leave proofs of communication that *anyone* can verify later (possibly with Bob's help).

Deniability has been a consideration for AKE protocols for (at least) the last 25 years since the design of the Internet Key Exchange (IKEv1) protocol [24, 29] through the influential *off-the-record* protocol [5] to today's privacy

conscious Signal messaging protocol [44]. Yet, designing deniable AKE protocols has proven to be a delicate matter and is still an active area of research (see the related work section below).

1.1 The Signal Protocol

Deniability was the central focus in the design of the Signal protocol [35, 44]. The latter has become the de-facto standard in the area of secure messaging protocols, having been adopted in popular applications such as WhatsApp and Facebook messenger. However in [45, 46] it has already been shown that online deniability does not hold for Signal. Still, offline deniability is widely believed to hold, and yet no formal proof has ever appeared in the literature.

In this paper, 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 [32,36], HMQV [31], and 3DH [34] (see Section 2 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.

1.2 Our Contributions

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 [36] (that does not use a random oracle for key derivation) fails deniability. 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) [2, 12] 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.

In the full version of the paper we prove deniability of the protocols studied here when augmented with explicit authentication.

1.3 On our use of knowledge extraction

The assumptions we use to prove our deniability 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 [4].

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 [4] 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 [25] 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 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, 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 [4].

1.4 Related work

Readers are referred to [3,7,9,43] for the formal definition of secure AKE. These formal treatments provide robust justifications for the security of AKE but do not deal with the issue of deniability. Informal discussions of deniability began to appear in [5,6,15,24,30,33]. Offline Deniable AKE were defined in [16] based on the work on deniable authentication in [19]. Definitions of deniable AKE that offer composability and online deniability were presented in [18, 47]. Provably offline deniable AKEs are presented in [16, 48], while online deniable ones are presented in [18, 45–47]. None of these protocols is implicitly authenticated. Although the offline deniability of protocols such as HMQV and 3DH is widely conjectured, we are not aware of any formal proof according to the definitions above for any implicitly authenticated scheme.

There are alternative, relaxed definitions of deniability that work for less expansive purposes [11, 20, 49]. These definitions include: content deniability, context deniability, source deniability, destination deniability, two notions of time deniability, peer deniability, and reasonable deniability. Each of these alternative definitions involve giving the simulator access to an oracle to perform the simulation. The oracle represent the "deniability loss" with respect to the standard strict definition of simulatability.

Other works in the context of deniability include [14, 28, 37], and we remark on the work by Pass [38] which stresses the important differences between a

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

deniability simulator and a zero-knowledge one (as defined in [22]). Tools to achieve deniability include: designated verifier proof and ring signatures. Designated verifier proofs refers to a proof that only one party can verify, ensuring that this proof cannot be passed on to anyone else [26]. A ring signature describes a signature made by a member of a group that cannot be traced back to that particular member [37, 40]. The "minimalistic" implicitly-authenticated schemes studied here do not involve any of these (relatively costly) techniques

A full specification of the Signal protocol can be found in [44]. As mentioned above it uses the Extended Triple Diffie-Hellman (X3DH) key agreement protocol [35] (built on the 3DH AKE [34]) followed by a communication session which include a *ratcheting* mechanism to refresh keys [39]. Security analyses of these protocols that do not include deniability can be found in [1, 10]. The deniability of X3DH and 3DH is often stated or taken for granted, but always without a formal definition or proof.

The MQV protocol was presented in [32, 36]. It inspired several other protocols including HMQV in [31] and the OAKE family of AKEs by Yao and Zhao [49]. No formal deniability proof of these protocols has appeared anywhere. In [49], one protocol in the OAKE family is claimed to have the weaker notion of reasonable deniability. Our full deniability results extend to the OAKE family as well.

1.5 Organization

In Section 2 we introduce some preliminaries, including the implicitly-authenticated key exchange protocols MQV, HMQV and 3DH. In Section 3 we present the definition of Deniable Authenticate Key Exchange and in Section 4 we prove how this notion extends to the deniability of communication sessions that use a key computed through a deniable AKE. In Section 5 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 6 we use the MQV example from previous section to derive a general characterization of non-deniability. In Section 7 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 8 we show that the deniability of 3DH implies the deniability of X3DH and the full Signal protocol. In Section 9 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. Finally, in Appendix A we present the KDL assumption and prove that together with KEA it implies KDH, hence basing our deniability results on these assumptions too.

2 Implicitly Authenticated Key Exchange

In this section, we recall two examples of implicitly authenticated key exchange protocols, namely HMQV [31] (and its predecessor MQV [32, 36]) and 3DH as used in Signal [34].

2.1 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, [17].

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 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 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. *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, we use Goldwasser and Micali's classical definition of computational indistinguishability [21].

Definition 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 $|\delta_{D,\mathcal{X}} - \delta_{D,\mathcal{Y}}| \leq \epsilon$.*

2.2 MQV and HMQV Protocols

The MQV protocol was introduced in [36] and further specified in [32]. A formal analysis of its security properties was presented by Krawczyk in [31] where he also presented an improved version called HMQV to address some of MQV's weaknesses uncovered by the analysis. In Figure 1, we describe both protocols.

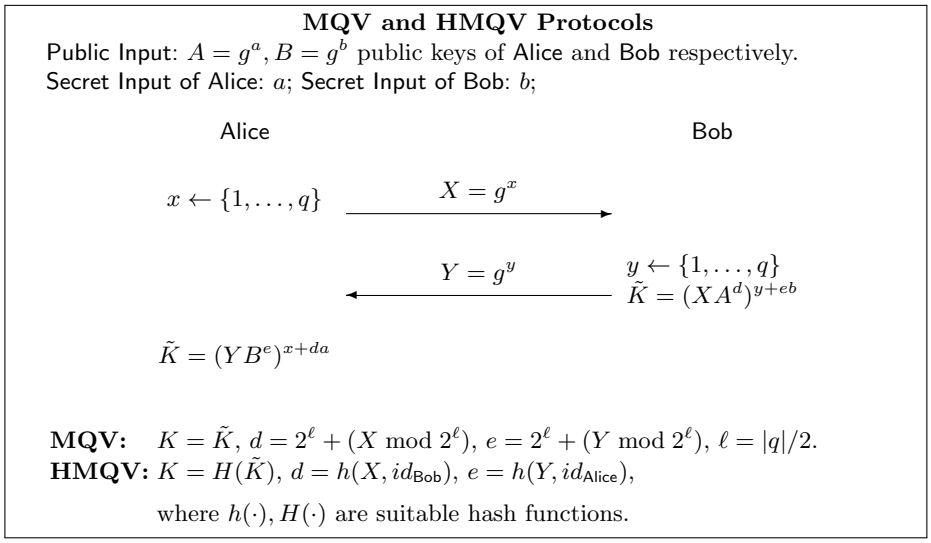

Fig. 1. Computation of the session key K in MQV and HMQV protocols

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.

2.3 Triple Diffie-Hellman

The Triple Diffie-Hellman (3DH) protocol creates a shared secret key between two parties who authenticate each other via public keys [34], 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 8. Here, we simply describe 3DH as a basic key exchange protocol where both parties are alive and communicating with each other.

Figure 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.

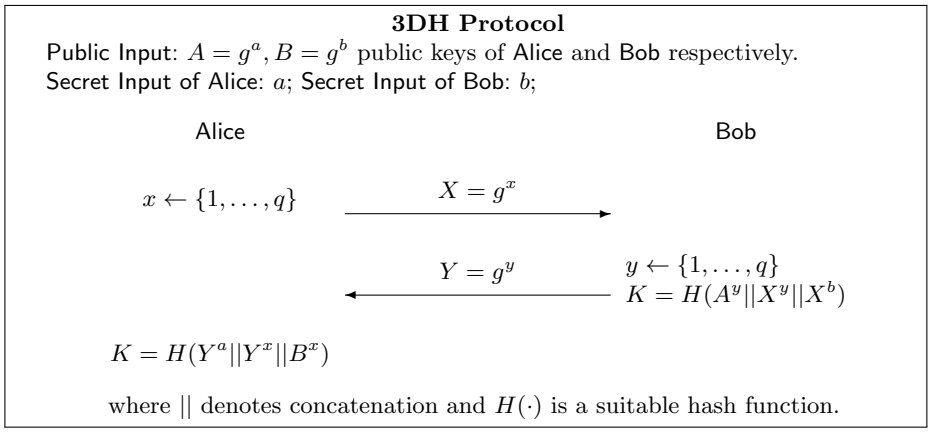

Fig. 2. Triple Diffie-Hellman key exchange protocol

2.4 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 [41], 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, 4 and 6) are simplified by assuming such a "Key Registration" setting. However, we later show (Section 9) that in all these cases, a slight strengthening of our assumptions gets rid of the need to assume private-key extractability.

3 Deniable Key Exchange

We recall the definition of deniable Authenticated Key Exchange (AKE in the rest) from [16, 18, 47].

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 [3, 7, 9, 43].

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 [8]. 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 $\mathbf{pk} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_l)$ 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 $\mathsf{View}_{\mathcal{M}}(\mathbf{pk}, aux)$.

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 $\mathbf{pk}$ of all parties and any auxiliary input aux available to the adversary. It generates $\mathsf{Sim}_{\mathcal{M}}(\mathbf{pk}, aux)$ by interacting with the adversary $\mathcal{M}$ as its peer. $\mathsf{Sim}_{\mathcal{M}}(\mathbf{pk}, aux)$ 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 5. *[16] 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* $\mathbf{pk} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_l)$ *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:*

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

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

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 [38], 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 5.1). 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 1.

4 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[2]. First we define deniability for an arbitrary interactive protocol between two parties, and then we show that any communication

[2] This was claimed informally and without proof in [16]; here, we use this result in essential way in Section 8 to show how deniability of 3DH carries to deniability of the whole Signal protocol.

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 $\mathbf{pk}$ 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 $\mathsf{View}(\mathbf{pk}, aux)$.

Definition 6. *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* $\mathbf{pk} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_l)$ *generated by* KG *and 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*

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

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

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 1. *Let* $P = [P_1, P_2]$ *be a session, where* P_1 *is a deniable authenticated key exchange according to Definition 5, which includes the session key in the view. Then* P *is deniable according to Definition 6.*

Proof. Based on Definition 6, a deniability simulator for P is required to simulate the transcript between the parties only, i.e. $\mathbf{tr}_{P_1}$ and $\mathbf{tr}_{P_2}$, 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: $\{\mathbf{tr}^*_{P_1}, K^*\}$.

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

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

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 $\mathbf{tr}_P^*$, there exists a distinguisher $\mathcal{D}$ running in $T_{\mathcal{D}}$ such that

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

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 $\{\mathbf{tr}_{P_1}^*, K^*\}$.

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[3]. 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 $\{\mathbf{pk}, \mathbf{tr}_{P_1}, K, coins_{\mathcal{M}_1}\}$ 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 probability that $\mathcal{D}$ does. □

5 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 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

[3] 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.

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 $K = g^{xy} A^{yd} (XA^d)^{be}$. Here, we assume that the simulator gets Bob's private key[4].

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.

5.1 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 $\mathcal{Y}$
   - 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.

[4] 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.

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 [13, 42], 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[5]).

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.

[5] 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.

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 [27]) 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 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.

5.2 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 [38]). 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[6].

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 "knowledge extraction" from Bob, which will address also the issues related to the counter-example from Section 5.1.

6 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

[6] 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.

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).

6.1 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.

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

6.2 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 2. *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 7, 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.

6.3 Proof of Theorem 2

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 $> \varepsilon_{\mathsf{J}}$.

□

7 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).

7.1 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 7. *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

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

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 2.4 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. *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 $K_{\text{real}} = DH(A^d X, B^e Y)$ from a simulated key $K_{\text{sim}} = (A^d X)^{be} \hat{Z}^d Y^x$ 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).

□

7.2 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) [2,12]. 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 8. *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[7] 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 Appendix A 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.

We now show that KDH implies the deniability of HMQV if the simulator can extract the incriminating party's (Bob in our examples) private key, for example via key registration as discussed in Section 2.4. In Section 9 we remove the need for this extraction condition using a mild generalization of KDH.

Theorem 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.

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

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 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. □

8 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 1 on the message flow of Signal.

We refer the reader to [10] 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 1 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 9. *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 9.

Theorem 6. *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 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. □

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

The simulation arguments of Theorems 4 and 6 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 10. *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 7. *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 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.

Acknowledgment: The authors thank the anonymous reviewer whose excellent comments greatly improved the presentation of this paper.

References

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. 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.
3. 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.
4. 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.
5. 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.
6. 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.

7. 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.
8. 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.
9. 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.
10. 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.
11. 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`.
12. 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.
13. 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.
14. M. Di Raimondo and R. Gennaro. New approaches for deniable authentication. *Journal of Cryptology*, 22(4):572–615, 10 2009.
15. 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.
16. 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.
17. W. Diffie and M. Hellman. New directions in cryptography. *IEEE Trans. Inf. Theor.*, 22(6):644–654, Sept. 2006.
18. 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.
19. 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.
20. 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.
21. S. Goldwasser and S. Micali. Probabilistic encryption. *JCSS*, 28(2):270–299, 4 1984.
22. S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, Feb. 1989.
23. S. Hada and T. Tanaka. On the existence of 3-round zero-knowledge protocols. *IACR Cryptol. ePrint Arch.*, 1999:9, 1999.
24. D. Harkins and D. Carrel. The internet key exchange (ike). RFC 2409, RFC Editor, 11 1998.
25. A. Jain, H. Lin, and A. Sahai. Indistinguishability obfuscation from well-founded assumptions. *IACR Cryptol. ePrint Arch.*, 2020:1003, 2020.

26. 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.
27. M. Jones and D. Hardt. The oauth 2.0 authorization framework: Bearer token usage. RFC 6750, RFC Editor, 10 2012.
28. 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.
29. C. Kaufman. Internet key exchange (ikev2) protocol. RFC 4306, RFC Editor, 12 2005.
30. 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.
31. 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.
32. 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.
33. W. Mao and K. Paterson. On the plausible deniability feature of internet protocols. Manuscript.
34. M. Marlinspike. Simplifying otr deniability. `https://signal.org/blog/simplifying-otr-deniability/`, 2013.
35. M. Marlinspike and T. Perrin. The x3dh key agreement protocol, 11 2016. Rev. 1.
36. 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.
37. 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.
38. 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.
39. T. Perrin and M. Marlinspike. The double ratchet algorithm, 11 2016. Rev. 1.
40. 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.
41. C. Schnorr. Efficient signature generation by smart cards. *J. Cryptology*, 4(3):161–174, 1991.
42. 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.
43. V. Shoup. On formal models for secure key exchange. Technical Report RZ 3120, IBM, 4 1999.
44. Signal technical information. `https://signal.org/docs/`.
45. 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.

46. N. Unger and I. Goldberg. Improved strongly deniable authenticated key exchanges for secure messaging. *Proceedings on Privacy Enhancing Technologies*, 2018(1):21–66, 2018.
47. S. Walfish. Enhanced security models for network protocols, 2008. PhD thesis.
48. 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.
49. 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 & Communications Security*, CCS '13, pages 1113–1128, New York, NY, USA, 2013. ACM.

Appendix A Knowledge of Discrete Log Assumption

In Section 7.2 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) [2, 12, 23].

Definition 11. *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 12. *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 13. *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 8. *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

$$\underset{C,(U,r)\leftarrow\Psi}{Prob}[C(U, aux, r) = DH(U, Z)] > \varepsilon. \tag{1}$$

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 (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.

$$\begin{aligned} Prob\,[B] &\geq Prob\,[B \wedge A] \\ &= Prob\,[B|A]\; Prob\,[A] \\ &= (1 - Prob\,[\neg B|A])\; Prob\,[A] \\ &= Prob\,[A] - Prob\,[\neg B|A]\; Prob[A] \\ &= Prob\,[A] - Prob\,[\neg B \wedge A] \\ &> Prob\,[A] - \varepsilon_{KEA} \end{aligned} \tag{2}$$

First, note that the inequality (2) is implied by KEA. Using the inequality (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.

$$\underset{(U,r)\leftarrow\mathcal{U}}{Prob}\,[\bar{\mu}(U, aux, r) = \mathsf{dlog}(Z)]$$

$$> \underset{(U,r)\leftarrow\mathcal{U}}{Prob}\,[\mu(U, aux, r) = (Z, DH(U, Z))] - \varepsilon_{KEA} \tag{3}$$

$$= \underset{r\leftarrow\mathcal{U}}{Prob}\,[M'(U||aux, r) = \mathsf{dlog}(Z)] + \underset{C,(U,r)\leftarrow\mathcal{U}}{Prob}\,[C(U, aux, r) = DH(U, Z)\ \wedge\ M'(U||aux, r) = \perp] - \varepsilon_{KEA} \tag{4}$$

$$> \underset{r\leftarrow\mathcal{U}}{Prob}\,[M'(U||aux, r) = \mathsf{dlog}(Z)] + \underset{C,(U,r)\leftarrow\mathcal{U}}{Prob}\,[C(U, aux, r) = DH(U, Z) \mid M'(U||aux, r) = \perp] - \varepsilon_{KEA} \tag{5}$$

$$= \underset{r\leftarrow\mathcal{U}}{Prob}\,[M'(U||aux, r) = \mathsf{dlog}(Z)] + \underset{C,(U,r)\leftarrow\mathcal{U}}{Prob}\,[C(U, aux, r) = DH(U, Z) \mid \hat{M}(U, aux, r) = \perp] - \varepsilon_{KEA} \tag{6}$$

$$= \underset{r\leftarrow\mathcal{U}}{Prob}\,[M'(U||aux, r) = \mathsf{dlog}(Z)] + \underset{C,(U,r)\leftarrow\Psi}{Prob}\,[C(U, aux, r) = DH(U, Z)] - \varepsilon_{KEA} \tag{7}$$

$$> \underset{r\leftarrow\mathcal{U}}{Prob}\,[M'(U||aux, r) = \mathsf{dlog}(Z)] + \varepsilon_{KDL} \tag{8}$$

In the above, line (3) is the repeat of inequality (2). The expression in (4) is due to the construction of μ and (5) follows from the conditional probability. In the line (6), we move from M' to $\hat{M}$ in the conditional part, this is due to the definition of $\hat{M}$. Equation (7) comes from the definition of the KDH assumption. Line (8) is due to the assumption made in (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 $(T, \tilde{T}', \tilde{T}_{KEA}, \varepsilon_{KDL} + \varepsilon_{KEA})$-KDH holds over group G and AUX. □