On Deniable Authentication against Malicious Verifiers

Rune Fiedler[1] and Roman Langrehr[2]

[1] Technische Universität Darmstadt
rune.fiedler@cryptoplexity.de
[2] ETH Zurich
roman.langrehr@inf.ethz.ch

Abstract. Deniable authentication allows Alice to authenticate a message to Bob, while retaining deniability towards third parties. In particular, not even Bob can convince a third party that Alice authenticated that message. Clearly, in this setting Bob should not be considered trustworthy. Furthermore, deniable authentication is necessary for deniable key exchange, as explicitly desired by Signal and off-the-record (OTR) messaging.
In this work we focus on (publicly verifiable) designated verifier signatures (DVS), which are a widely used primitive to achieve deniable authentication. We propose a definition of deniability against malicious verifiers for DVS. We give a construction that achieves this notion in the random oracle (RO) model. Moreover, we show that our notion is not achievable in the standard model with a concrete attack; thereby giving a non-contrived example of the RO heuristic failing.
All previous protocols that claim to achieve deniable authentication against malicious verifiers (like Signal's initial handshake protocols X3DH and PQXDH) rely on the Extended Knowledge of Diffie–Hellman (EKDH) assumption. We show that this assumption is broken and that these protocols do not achieve deniability against malicious verifiers.

Keywords. Deniability, Random oracle model, Rogue key attacks

1 Introduction

In an offline world, speaking convinces the immediate listener of the authenticity of the spoken word but not necessarily a third party, to whom the listener relays the conversation later. In fact, this can be the reason why somebody *dares* to speak in the first place, e.g., in case of health issues, whistleblowing, or moral but illegal protests. We call this property *deniable authentication*, where one party can authenticate a message to the intended recipient and deny its contents to all other parties.

Formalizing deniable authentication in the digital realm has been a subject of research for more than three decades, e.g., see works by Dolev, Dwork, and Naor [19], Dwork, Naor, and Sahai [20], and Di Raimondo and Gennaro [16]. Several primitives provide deniable authentication, including: 1) designated

verifier signatures, introduced by Jakobsson, Sako, and Impagliazzo [30], where a signer *designates* a signature to a specific verifier, who cannot transfer this authentication to a third party; 2) ring signatures, introduced by Rivest, Shamir, and Tauman [43], which guarantee that 1 *out of* n ring members authenticates a message; 3) split KEMs, introduced by Brendel, Fischlin, Günther, Janson, and Stebila [9], dubbed authenticated KEMs (AKEMs) by Alwen, Blanchet, Hauck, Kiltz, Lipp, and Riepel [1], which allow decapsulation of an encapsulated shared secret in a manner that deniably authenticates the encapsulator.

Deniable authentication has since been transferred to the protocol layer. In particular, Di Raimondo, Gennaro, and Krawczyk [17] have introduced deniability for key exchange. Chat applications and messaging protocols are particularly interested in deniability. Off-the-record (OTR) messaging prominently discussed deniability [6, 54, 55]. Signal, inspired by OTR, also aims for deniability of its initial handshake protocol. Previously, they deployed X3DH [39], which Vatandas, Gennaro, Ithurburn, and Krawczyk [56] showed to be deniable under their novel Extended Knowledge of Diffie-Hellman Assumption (EKDHA). Looking ahead, in Section 5.2 we show that the EKDHA is broken. For the post-quantum replacement protocol PQXDH [33], Fiedler and Janson [23] have proven slightly worse deniability properties that additionally require plaintext awareness of the used KEM. Several post-quantum replacement protocols for Signal's initial handshake were proposed [24, 8, 18, 12] and all proved some form of deniability. We refer the reader to [23, Appendix A] for a more detailed discussion of different notions of deniability for key exchange.

1.1 Malicious verifier keys

In the setting of deniable authentication, Alice wants to convince Bob, without Bob being able to convey his conviction to Charlie. If Alice was able to trust Bob not to transfer his belief to Charlie (and if Alice was assured that Bob cannot be coerced into cooperating with Charlie), then we would not need special precautions to ensure deniability in the first place. Hence, while Alice wants to authenticate a message to Bob, she does not trust him. As a logical consequence, Alice should not trust Bob to generate his keys honestly. We apply this idea to designated verifier signatures (DVS), which we consider to be the most basic primitive out of those mentioned above[3].

A DVS that retains unforgeability and deniability[4] against malicious verifier keys has several uses. A whistleblower can deniably authenticate her message, even if the receiving journalist is malicious, compromised, or negligent. In more extreme cases, law enforcement might pose as a journalist to gather incriminating material on a whistleblower for a court trial. Given that regimes around the world are prosecuting whistleblowers, this does not appear far-fetched. Hence, we cannot trust a verifier to generate keys honestly.

[3] Split KEMs and AKEMs additionally ensure confidentiality of the shared secret, and ring signatures are more generic since they can deal with more than two users.

[4] In the rest of the paper we refer to the deniability property as "source hiding", cp. Definition 17.

Furthermore, in a chat application users interact with many other users to which they have different levels of trust. And even if one of the chat partners turns out to be malicious, users expect deniability in all chats, including the chat with the malicious peer, and confidentiality in all remaining chats. We study security notions for DVS that provide these assurances.

1.2 Related concepts and works

The concept of verifier keys being maliciously generated has already appeared several times in the literature. Closest to our field of study, Shim [48] and Zhang, Au, Yang, and Susilo [58] have studied multi-DVS against malicious verifier keys, dubbed *rogue key attacks*. An unforgeable signature scheme can exhibit unexpected behavior against maliciously generated keys, such as one signature verifying messages under two distinct public keys, one signature verifying two distinct messages, or a signature not guaranteeing that the signer knows the message. These Beyond UnForgeability Features (BUFF) were defined and compiled by Cremers, Düzlü, Fiedler, Fischlin, and Janson [14]. Looking at KEMs, Cremers, Dax, and Medinger [13] studied whether shared secrets, public keys, and ciphertexts may be bound to each other (e.g. a shared secret may only stem from decapsulating one particular ciphertext). One of their settings allows maliciously generated public keys as well.

1.3 Our contributions

We propose new security definitions for DVS that provide unforgeability and deniability against malicious verifiers. Similar notions for unforgeability were already considered by previous works [48, 58], but the notion for deniability is novel. We also propose a construction satisfying our new definitions in the random oracle model. Furthermore, we show that our new deniability notion cannot be achieved in the standard model by giving a concrete attack. In particular, this shows that instantiating our random oracle construction with a concrete hash function results in an insecure scheme. To the best of our knowledge, the only way to achieve a similar security notion so far relied on the EKDH assumption[5], which we show to be broken.

Model. Recall that the standard notion of deniability for DVS guarantees that a valid signature generated with the signer's secret key (and the verifier's public key) is indistinguishable from a simulated signature, that was generated with the signer's public key and the verifier's secret key.

To capture that verifiers might generate their keys maliciously, we aim for a security notion where the adversary chooses the verifier's public key. In this setting, there might not be a matching secret key for the verifier's public key.

[5] The deniability of Signal's initial handshake protocols X3DH and PQXDH rely on the EKDH assumption.

Thus, we need a different approach to simulate signatures. Clearly, the simulation should require some secret that only the (potentially malicious) verifier possesses, since otherwise unforgeability could no longer hold. Moreover, everybody should be convinced that the verifier actually possesses this secret: Otherwise, if the verifier could convince a judge of not possessing the secret, the judge would be convinced that a signature must have been generated by the purported signer. In this work, we propose to use the *random coins* used to generate the verifier's public key (without making any restrictions on *how* the verifier generated this public key) as such a secret. Clearly, a classical verifier (without quantum capabilities) must have known these random coins when generating the public key and it is impossible to convince a third party that he deleted these random coins, since the verifier can always keep a copy of the random coins.

More concretely, we propose an extractor-based definition where the simulated signature is generated by an extractor that can depend on the adversary and is given the adversary's random coins. Following [41], the extractor should only have capabilities that a verifier actually has in the real world. In particular, this excludes knowing the trapdoor to a common reference string (crs) or the ability to program a random oracle (RO). The extractor might still simulate the adversary in its head and change the crs or program the RO in these simulations, but it has to output a signature that is indistinguishable in the *real world*, where the extractor does not posses these capabilities.

The security notion guarantees that no adversary can tell apart real signatures from those generated by the extractor for an honestly generated signer public key and a verifier public key and message chosen by the adversary.

Identities. Unfortunately, the model described so far is very unrealistic, because in the real world it is fairly easy to generate a public key without knowing the random coins used to generate it. One way to achieve this is to copy the public key of another user.

We avoid this attack by introducing identities, which are meaningful unique identifiers of each user, e.g., an e-mail address, a phone number, or a username in a chat application. Verifier keys are generated with respect to an identity and the signer signs messages with respect to a verifier public key and a verifier identity. Signing only has to output valid signatures (for which correctness and unforgeability hold) if the verifier identity matches the one used to generate the verifier public key.

Looking ahead, our ROM construction prevents these key copying attacks by outputting a dummy value when the identity the signer intended to sign for does not match the identity used for generating the verifier public key.

Auxiliary input. Of course, there might be different ways to obtain random data without knowing the random coins in the real world. For example, in the real world a verifier has access to public keys or signatures of other, unrelated deployed cryptosystems. A malicious verifier could use such data to generate his own public key.

If the adversary sampled this data on its own, it would know the random coins and therefore also the extractor would get access to these random coins, which is not realistic. Therefore, we have to provide the adversary (and thus also the extractor) with auxiliary input (*common auxiliary input model*[6]).

Of course, we now have to avoid that the auxiliary input just contains a verifier public key for our DVS. Otherwise, a malicious verifier could just use the public key from the auxiliary input. An extractor would then have to forge a signature for a signer and verifier public key for which it knows no secret. The existence of such an extractor would contradict the unforgeability of the signature scheme.

We avoid this by using a (trusted) setup for our DVS that generates public parameters that are given as input to all other algorithms of the DVS. The auxiliary input is not allowed to depend on the public parameters. This also shows that our security notion cannot be achieved in the plain model (without trusted setup) and requires at least the uniform random string (urs) model, where the public parameters are uniformly random bits.

Finally, an adversary can realistically also get some inputs depending on the public parameters, such as signer public keys, verifier public keys for different identities (and possibly also the corresponding secret keys, modeling users who are negligent in keeping their secret keys safe), and signatures. As a first step in this line of research we keep this to a minimum and propose a notion we call *simplified malicious source hiding* where the adversary gets access to an oracle for generating verifier public keys.

Prior work [17] (and subsequently [56, 23]) use auxiliary inputs to model data that a receiver is able to obtain that is *helpful* to simulate a protocol transcript. This is very different to our use of auxiliary inputs, where the auxiliary input is only to the *disadvantage* of the simulator. Formally, the difference is that our security game asks for security to hold for *all* auxiliary inputs $\mathsf{aux} \in \{0,1\}^*$. In particular, this includes the case where aux is the empty string and thus our security notion is stronger than the version without any auxiliary input. In contrast, e.g. [17], only demand security for *some* auxiliary inputs (all auxiliary inputs that form a valid protocol transcript) and this can be weaker than the variant without any auxiliary input.

We discuss possible restrictions on the auxiliary input in Section 1.4.

Limitations. The approach of using the random coins as the verifier's secret still has a few limitations.

1. The verifier might not know the random coins if he generated the key with the help of someone else (for example with multi-party computation). In this

[6] Here, "common" refers to the adversary and the extractor getting the same auxiliary input. If we allow the extractor to get a different (existentially quantified) auxiliary input, it would be much easier to achieve our security notion, but that model is not realistic: For example, if the adversary's auxiliary input contains a public key of a different cryptosystem, the simulator's auxiliary input can contain the secret key. This is not realistic, because it is not feasible for the real receiver to retrieve the secret key and the simulator should model the real receiver's capabilities.

case the knowledge of the random coins might be spread among several people (or devices). On the one hand, this seems unavoidable (unless we assume a trusted party or trusted device generating the verifier's key). On the other hand, we think this is not a severe limitation: If several users collaborate to generate the verifier public key, it might be hard to convince a judge that they did not also collaborate to produce the randomness.
2. Quantum capabilities (or other physical processes) might enable the verifier to produce a verifier key and prove that he does not know the corresponding random coins.

New security definitions. We present our new security notions in Section 3 and show separations from existing security notions (that do not capture maliciously generated keys) in Appendix A.

Non-solutions. Before describing our solutions, we describe a few natural approaches that fail to achieve deniability against malicious verifiers.

Knowledge-type assumptions. Since our security notion involves an extractor, one might be tempted to use knowledge-type assumptions like the knowledge of exponent assumption [15]. However, most of these assumptions imply extractable one-way functions (EOWF), which cannot be secure in the common auxiliary input model [5].

Vatandas et al. [56] propose the Knowledge of Diffie-Hellman (KDH) and extended KDH (EKDH) assumption, which are knowledge-type assumptions that do not imply EOWF. They conjecture that these assumptions are hard in the common auxiliary input model. Vatandas et al. and Fiedler and Janson [56, 23] use this assumption to achieve deniability notions for the Signal's initial handshake protocols X3DH and PQXDH. We show that this assumption is broken by giving a relatively simple attack in Section 5.2. Moreover, the attack can also be extended to these protocols, showing that that they do not achieve deniability in a model that allows maliciously generated public keys. We give more details on this attack in Appendix B.

NIZKPoKs. A standard technique to deal with maliciously generated public keys is to equip them with a proof of knowledge of the secret key. Informally speaking, this restricts an adversary to public keys from the image of the key generation algorithm[7] and allows the reduction to extract a secret key for adversarially generated public keys.

To see why this approach does not work in our setting, we have to dig into the definitions of knowledge soundness for non-interactive zero-knowledge proofs of knowledge (NIZKPoKs). There are two incomparable notions in the literature.

[7] Note that such a NIZKPoK does not guarantee anything about the *distribution* of the public key, thus it might still follow a very different distribution than honestly generated public keys.

The first variant of knowledge soundness allows for extraction of a witness with a trapdoor to the common reference string (crs) of the NIZKPoK. Such an extraction is not helpful for our security notion, because giving the extractor such a trapdoor is unrealistic (as explained in the section on the model).

Another variant of knowledge soundness allows for extraction of a witness from the randomness used to generate the NIZKPoK. This notion is typically considered for succinct non-interactive arguments of knowledge (SNARKs), where the previous notion is unachievable. This form of extraction would work well with our simplified malicious source hiding notion, but since SNARKs (or any NIZKPoK satisfying this type of extractability) cannot be secure in the common auxiliary input model [5], we cannot use them.

A solution in the ROM. The starting point of our solution is to equip the public key with a NIZKPoK of the secret key, where the NIZKPoK satisfies the crs trapdoor based extraction. Assume a regular DVS in which we equip all public keys with a NIZKPoK of the secret key. As explained above, this solution alone will not work, because the extractor has no access to a trapdoor for the crs.

However, the extractor can generate its own crs along with a trapdoor and simulate internally the adversary on this new crs. Assume for the moment the adversary generates the DVS verifier public key independently of the crs of the NIZKPoK. The extractor could then use the trapdoor to extract the verifier's secret key and use it to simulate the signature.

The missing ingredient for this approach is a tool to *enforce* that the adversary generates the DVS verifier public key independently of the crs. Our solution for this problem is to pick the crs as the hash of the public key, the verifier's identity and a random string s that is part of the public parameters of the scheme. Clearly, the adversary then has to query the ROM on these values to generate a valid NIZKPoK for the verifier public key. We include s in the hash, because then this ROM query cannot be made during the generation of the auxiliary input – which happens independently of s – and has to be done by the adversary. Since we model identities, this query cannot be made during the generation of any of the verifier keys we provide to the adversary (since they use a different identity).

This idea is reminiscent of the Fiat–Shamir transform [22], that compiles a Σ-protocol to a NIZK by choosing the challenge as a hash of the commit message. This ensures a random challenge generated independently of the commit message.

Finally, the underlying DVS has to satisfy source hiding for all valid verifier public/secret key pairs (instead of an honestly generated one), since the NIZKPoK does not give us any guarantees about the *distribution* of the public keys. We show that a DVS satisfying this notion can be built from OR-proofs.

Impossibility result. We also show in this work that our simplified malicious source hiding definition cannot be achieved by DVS in the standard model, assuming that indistinguishability obfuscation (iO) [4] exists. With the recent progress on constructing iO [29, 28, 42], it seems very plausible that iO exists. The result is reminiscent of [5]. However, we cannot use their result directly, because

our simplified malicious source hiding definition does not imply extractable one-way functions (as far as we know).

Recall that if the auxiliary input would contain a valid verifier public key (without the secret key or any hints on how it was generated), the adversary could simply use this public key in the security game. The extractor, who gets the adversary's random coins, then gets no useful information how that public key was generated. Hence, simulating a signature is as hard for the extractor as breaking unforgeability of the DVS.

However, verifier public keys can depend on the public parameters, which are not available during generation of the auxiliary input. Hence, we cannot directly include a valid verifier public key in the auxiliary input. Though, we can include an obfuscated program that takes the public parameters as input and outputs a valid verifier public key for the public parameters, which yields essentially the same problem as before. We present this argument in detail in Section 5.1.

This attack also applies to our ROM based construction, when we replace the random oracle with a concrete hash function. Thus, the wide-spread heuristic of replacing a random oracle with a suitable hash function fails here. To see where exactly the heuristic fails, note that in our ROM based construction it is necessary to make a random oracle query to generate a verifier public key (to obtain the crs for the NIZKPoK). However, an obfuscated circuit cannot access the random oracle, therefore this attack does not work for our ROM based scheme. However, if we replace the ROM with an efficiently computable function, the obfuscated circuit can evaluate this function, therefore the attack works.

There have been examples of constructions in the random oracle that become insecure when we replace the random oracle by any concrete hash function prior to this work [11]. However, these constructions have been criticized for being "contrived counter-examples" and they did not stop cryptographers from instantiating ROM schemes with concrete hash functions in practice. Our example for a failure of the ROM heuristic is not contrived. In fact, the authors of this work had the idea for the ROM based construction before becoming aware of this impossibility result and realized the failing of the ROM heuristic only when they tried to resolve the alleged contradiction between the construction and the impossibility result. The authors did not intend to construct a scheme where the ROM heuristic fails.

Concurrently to this work, Khovratovich, Rothblum, and Soukhanov also present a non-contrived counter-example of the random oracle heuristic [32]. Their counter-example works very different to ours on a technical level. Previous works obtained non-contrived counter examples only for the ideal cipher model [7].

Our impossibility result also rules out the existence of ring signatures that retain anonymity (among the *entire* ring) if at least one public key in the ring was generated maliciously, since such a ring signature would imply a simplified malicious source hiding DVS. Moreover, our impossibility result makes no use of the oracle returning verifier public keys, and therefore applies also to a weaker notion of security which can be achieved without identities.

We want to emphasize here that this impossibility result crucially relies on the public verifiability of DVS. In a privately verifiable DVS, the extractor could generate a signature that does not verify (and therefore does not contradict unforgeability), but is still indistinguishable from a real signature to the adversary.

1.4 Open problems

A compelling question is whether deniability against malicious verifiers can be achieved in a weaker model, that circumvents our impossibility result while still being realistic. A natural restriction is to allow only *efficiently sampleable* auxiliary inputs. However, this does not help to circumvent our impossibility result, because the auxiliary input for our result is efficiently sampleable. Another realistic variant would be to restrict the auxiliary input to contain at least certain information that a receiver can realistically obtain (for example data that is published in a public bulletin board) and that can help the simulator. However, that also does not help to circumvent our impossibility result, since this result holds as long as the auxiliary input can also contain a suitable obfuscated circuit.

The authors of this work are convinced that restricting the auxiliary input to circumvent our impossibility result does not result in a security notion that meaningfully captures deniability with malicious verifiers: The reason is that such a result would have to restrict the auxiliary input to *not* contain a suitable obfuscated circuit. However, a saboteur (e.g. a government that wants to make end-to-end encrypted messaging less attractive) might sample a bit string aux such that the DVS is not malicious source hiding with respect to this aux. Assume the saboteur publishes aux on her website and deletes the random coins used. Now deniability for every other user (except the saboteur herself) is lost, because the verifier could retrieve aux from the saboteur's website and use it to generate his key. Even worse, such an auxiliary input might be generated and published without any malicious intent. In fact, there are proposed schemes that require as setup to generate an obfuscated circuit that is suitable for our impossibility result and to delete the random coins used to generate it (for example, the constructions from selectively secure universal samplers in [25]).

A more promising approach to circumvent our impossibility result is to consider privately verifiable DVS, since our impossibility result crucially relies on the public verifiability of the DVS. Privately verifiable DVS are sufficient for many applications that require deniable authentication, for example messaging. If this can be answered positively, it would also be desirable to achieve a more realistic notion that gives the adversary all values it can realistically obtain without generating them itself, such as signer public keys and signatures.

An interesting new research direction is to find new ways of deploying random oracle based schemes, that also work for our DVS in the ROM. For example, an approach could be to use trusted hardware that implements a pseudo-random function with a hard-coded key. Every user of the system then needs such a device with the same key, while the key is not publicly known.

Furthermore, our simplified malicious source hiding definition is not meaningful against quantum adversaries. We leave it as an open question to model source hiding against malicious verifiers in a (post-)quantum setting.

Finally, we leave open to transfer our notion (or a stronger, more realistic version) of deniability against malicious verifiers to more complex primitives and the protocol level, e.g., for authenticated KEMs and key exchange.

2 Preliminaries

2.1 Notation

We denote the output y of a deterministic algorithm A on input x as $y \leftarrow A(x)$, and $y \xleftarrow{\$} A(x)$ for a probabilistic algorithm or $y \leftarrow A(x;r)$ with explicit randomness r. We write $x \xleftarrow{\$} X$ for (uniformly) sampling from a set or distribution X, and $\mathsf{Img}(A)$ for the image of an algorithm A. We write $\mathbb{E}_{x \xleftarrow{\$} X} f(x)$ for the expected value of the random variable $f(x)$ with $x \xleftarrow{\$} X$. We append an element e to an ordered list L with $L \xleftarrow{+} e$. To evaluate a boolean expression e to 1 or 0 we write $[\![e]\!]$.

2.2 Rényi entropy

The collision entropy (Rényi entropy of order $\alpha = 2$) of a discrete random variable X, that can take n different values, each with probability p_i for $i \in [n]$, is defined as $\mathsf{H}_2(X) := -\log\left(\sum_{i=1}^{n} p_i^2\right)$. The min-entropy (the limit of the Rényi entropy for $\alpha \to \infty$) is $\mathsf{H}_\infty(X) := -\log \max_{i\in[n]} p_i$.

Fact 1. $\mathsf{H}_2(X) \leq 2\mathsf{H}_\infty(X)$.

Proof.

$$\mathsf{H}_2(X) = -\log\left(\sum_{i=1}^{n} p_i^2\right) \geq -\log(\max_{i\in[n]} p_i^2) = -2\log(\max_{i\in[n]} p_i) = 2\mathsf{H}_\infty(X)$$

□

2.3 NP relations

For an NP-relation $\mathcal{R}$ we define the associated language as $\mathcal{L}_\mathcal{R} := \{x \mid \exists w : (x, w) \in \mathcal{R}\}$.

Definition 1 (Hard on average). *We say that the search problem for an NP-relation $\mathcal{R}$ is (t,ε)-hard on average if there exists an efficient sampler* Samp *that outputs statements $x \in \mathcal{L}_\mathcal{R}$ such that for any adversary $\mathcal{A}$ running in time at most t we have $\Pr[(x,w) \in \mathcal{R}] \leq \varepsilon$ where the probability is taken over $x \xleftarrow{\$} \mathsf{Samp}()$ and $w \xleftarrow{\$} \mathcal{A}(x)$.*

2.4 NIZKs

We recall the definition of non-interactive zero knowledge (NIZK) proofs and NIZK proofs of knowledge (NIZKPoKs).

Definition 2. *A NIZK* $\mathsf{NIZK} = (\mathsf{Gen}_{\mathsf{NIZK}}, \mathsf{Prove}, \mathsf{Vrfy}_{\mathsf{NIZK}})$ *for a NP-relation* $\mathcal{R}$ *with associated language* $\mathcal{L}_{\mathcal{R}} = \{x \mid \exists w : (x, w) \in \mathcal{R}\}$ *consists of three algorithms:*

- $\mathsf{Gen}_{\mathsf{NIZK}}() \xrightarrow{\$} \mathsf{crs}$*: outputs at common reference string (crs)* crs*. We say that the NIZK is in the uniform random string (urs) model, if* $\mathsf{Gen}_{\mathsf{NIZK}}$ *only outputs its random coins. In this case we call the output* urs*.*
- $\mathsf{Prove}(\mathsf{crs}, x, w) \xrightarrow{\$} \pi$*: receives a crs* crs *and a statement witness pair* $(x, w) \in \mathcal{R}$ *as input, and outputs a proof* π*.*
- $\mathsf{Vrfy}_{\mathsf{NIZK}}(\mathsf{crs}, x, \pi) \rightarrow$ true/false*: receives a common references string* crs*, a statement* x*, and a proof* π *as input and outputs* true *or* false*, indicating a valid or invalid proof.*

We require the following properties from the NIZK:

Definition 3 (Completeness). *A NIZK* $\mathsf{NIZK} = (\mathsf{Gen}_{\mathsf{NIZK}}, \mathsf{Prove}, \mathsf{Vrfy}_{\mathsf{NIZK}})$ *is* δ*-complete if for all* $(x, w) \in \mathcal{R}$*,* $\mathsf{crs} \xleftarrow{\$} \mathsf{Gen}_{\mathsf{NIZK}}()$*, and* $\pi \xleftarrow{\$} \mathsf{Prove}(\mathsf{crs}, x, w)$ *it holds that*

$$\Pr[\mathsf{Vrfy}_{\mathsf{NIZK}}(\mathsf{crs}, x, \pi)] \geq 1 - \delta,$$

where the probability is taken over the randomness used to sample crs *and* π*.*

Definition 4 (Zero-knowledge). *A NIZK* $\mathsf{NIZK} = (\mathsf{Gen}_{\mathsf{NIZK}}, \mathsf{Prove}, \mathsf{Vrfy}_{\mathsf{NIZK}})$ *is* $(t, t_{\mathsf{Sim}_{\mathsf{zk}}}, q, \varepsilon)$*-zero-knowledge if there exist algorithms*

- $\mathsf{TDGen}_{\mathsf{zk}}() \xrightarrow{\$} (\mathsf{crs}, \mathsf{td}_{\mathsf{zk}})$*: outputs a crs* crs *and a trapdoor* $\mathsf{td}_{\mathsf{zk}}$ *and*
- $\mathsf{Sim}_{\mathsf{zk}}(\mathsf{crs}, \mathsf{td}_{\mathsf{zk}}, x) \xrightarrow{\$} \pi$*: that receives a crs* crs *with the corresponding trapdoor* $\mathsf{td}_{\mathsf{zk}}$ *and a statement* x *as input and outputs a proof* π

with combined runtime $t_{\mathsf{Sim}_{\mathsf{zk}}}$ *such that for all adversaries* $\mathcal{A}$ *running in time* t *and making at most* q *queries to the* Prove *oracle we have*

$$\mathsf{Adv}^{\mathsf{zk}}_{\mathsf{NIZK}}(\mathcal{A}) := \left| \Pr[\mathcal{G}^{\mathsf{zk}}_{\mathsf{NIZK}}(\mathcal{A}) = 1] - \frac{1}{2} \right| \leq \varepsilon$$

where the experiment $\mathcal{G}^{\mathsf{zk}}_{\mathsf{NIZK}}(\mathcal{A})$ *is defined in Figure 1.*

Definition 5 (Soundness). *A NIZK* $\mathsf{NIZK} = (\mathsf{Gen}_{\mathsf{NIZK}}, \mathsf{Prove}, \mathsf{Vrfy}_{\mathsf{NIZK}})$ *is* $(t_{\mathsf{s}}, \varepsilon_{\mathsf{s}})$*-sound iff for all adversaries* $\mathcal{A}$ *running in time* t_{s}*, for* $\mathsf{crs} \leftarrow \mathsf{Gen}_{\mathsf{NIZK}}()$ *and* $(x, \pi) \xleftarrow{\$} \mathcal{A}(\mathsf{crs})$

$$\Pr[\mathsf{Vrfy}_{\mathsf{NIZK}}(\mathsf{crs}, x, \pi) = 1 \wedge x \notin \mathcal{L}_{\mathcal{R}}] \leq \varepsilon_{\mathsf{s}}.$$

The probability is taken over the random coins of $\mathsf{Gen}_{\mathsf{NIZK}}$ *and* $\mathcal{A}$*.*

If a NIZK fulfills the following property, which is a strengthening of regular soundness, we call it a NIZKPoK.

$\underline{\mathcal{G}^{\mathsf{zk}}_{\mathsf{NIZK}}(\mathcal{A})}$:

1 $b \xleftarrow{\$} \{0,1\}$
2 **if** $b = 0$ **then**
3 $\quad \mathsf{crs} \xleftarrow{\$} \mathsf{Gen}_{\mathsf{NIZK}}()$
4 **else**
5 $\quad (\mathsf{crs}, \mathsf{td}_{\mathsf{zk}}) \xleftarrow{\$} \mathsf{TDGen}_{\mathsf{zk}}()$
6 $b' \xleftarrow{\$} \mathcal{A}^{\textsc{Prove}}(\mathsf{crs})$
7 **return** $[\![b' = b]\!]$

$\underline{\textsc{Prove}(x, w)}$:

8 **if** $(x, w) \notin \mathcal{R}$ **then**
9 $\quad$ **return** $\bot$
10 **else if** $b = 0$
11 $\quad$ **return** $\mathsf{Prove}(\mathsf{crs}, x, w)$
12 **else**
13 $\quad$ **return** $\mathsf{Sim}_{\mathsf{zk}}(\mathsf{crs}, \mathsf{td}, x)$

Fig. 1. The zero-knowledge experiment for a NIZK $\mathsf{NIZK} = (\mathsf{Gen}_{\mathsf{NIZK}}, \mathsf{Prove}, \mathsf{Vrfy}_{\mathsf{NIZK}})$.

$\underline{\mathcal{G}^{\mathsf{pok}}_{\mathsf{NIZKPoK}}(\mathcal{A})}$:

1 $(\mathsf{crs}, \mathsf{td}_{\mathsf{ks}}) \xleftarrow{\$} \mathsf{TDGen}_{\mathsf{ks}}()$
2 $(x^\star, \pi^\star) \xleftarrow{\$} \mathcal{A}(\mathsf{crs})$
3 $w^\star \xleftarrow{\$} \mathsf{Extr}(\mathsf{crs}, \mathsf{td}_{\mathsf{ks}}, x^\star, \pi^\star)$
4 **return** $[\![\mathsf{Vrfy}_{\mathsf{NIZK}}(\mathsf{crs}, x, \pi) \wedge (x, w) \notin \mathcal{R}]\!]$

Fig. 2. The proof of knowledge experiment for NIZKPoK with $\mathsf{TDGen}_{\mathsf{ks}}$ and Extr.

Definition 6 (Knowledge soundness). *A NIZK* $\mathsf{NIZK} = (\mathsf{Gen}_{\mathsf{NIZK}}, \mathsf{Prove}, \mathsf{Vrfy}_{\mathsf{NIZK}})$ *is* $(t_{\mathsf{td}}, \varepsilon_{\mathsf{td}}, t_{\mathsf{ks}}, \varepsilon_{\mathsf{ks}})$*-knowledge sound iff there exists algorithms* $\mathsf{TDGen}_{\mathsf{ks}}$ *and* Extr *where*

- $\mathsf{TDGen}_{\mathsf{ks}}() \xrightarrow{\$} (\mathsf{crs}, \mathsf{td}_{\mathsf{ks}})$*: outputs a crs* crs *and an extraction trapdoor* $\mathsf{td}_{\mathsf{ks}}$ *and*
- $\mathsf{Extr}(\mathsf{crs}, \mathsf{td}_{\mathsf{ks}}, x, \mathsf{Prove}) \xrightarrow{\$} w$*: receives a crs* crs *with the corresponding extraction trapdoor* $\mathsf{td}_{\mathsf{ks}}$ *and a statement* x *with a proof* Prove *as input and outputs a witness* w*.*

We also require for all adversaries $\mathcal{A}$ *running in time* t_{ks}

$$\mathsf{Adv}^{\mathsf{pok}}_{\mathsf{NIZKPoK}}(\mathcal{A}) := \Pr[\mathcal{G}^{\mathsf{pok}}_{\mathsf{NIZKPoK}}(\mathcal{A}) = 1] \leq \varepsilon_{\mathsf{ks}}.$$

where $\mathcal{G}^{\mathsf{pok}}_{\mathsf{NIZKPoK}}(\mathcal{A})$ *is defined in Figure 2, and that for all adversaries* $\mathcal{B}$ *running in time* t_{td} *we have*

$$\left| \Pr_{\mathsf{crs} \xleftarrow{\$} \mathsf{Gen}_{\mathsf{NIZK}}()}[\mathcal{B}(\mathsf{crs}) = 1] - \Pr_{(\mathsf{crs}, \mathsf{td}_{\mathsf{ks}}) \xleftarrow{\$} \mathsf{TDGen}_{\mathsf{ks}}()}[\mathcal{B}(\mathsf{crs}) = 1] \right| \leq \varepsilon_{\mathsf{td}}.$$

We also recall the notion of witness indistinguishability [21], which is a relaxation of zero-knowledge. A NIZK or NIZKPoK, that does not satisfy zero-knowledge but witness indistinguishability, is called a NIWI or NIWIPoK, respectively.

Definition 7 (Witness indistinguishable). *A NIWI* $\mathsf{NIWI} = (\mathsf{Gen}_{\mathsf{NIZK}}, \mathsf{Prove}, \mathsf{Vrfy}_{\mathsf{NIZK}})$ *for relation* $\mathcal{R}$ *is* (t, ε)-witness indistinguishable *if for every* $\mathsf{crs} \xleftarrow{\$} \mathsf{Gen}_{\mathsf{NIZK}}()$ *and every adversary running in time at most* t *and for all* $(x, w), (x, w') \in \mathcal{R}$

$$|\Pr[\mathcal{A}(\mathsf{crs}, \mathsf{Prove}(\mathsf{crs}, x, w)) \Rightarrow 1] - \Pr[\mathcal{A}(\mathsf{crs}, \mathsf{Prove}(\mathsf{crs}, x, w')) \Rightarrow 1]| \leq \varepsilon$$

where the probability is taken over $\mathsf{crs} \xleftarrow{\$} \mathsf{Gen}_{\mathsf{NIZK}}()$ *and the randomness of* $\mathcal{A}$ *and* Prove.

2.5 Signatures

Definition 8 (Signature scheme). *A* signature scheme $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Ver})$ *for a message space* $\mathcal{M}$ *and signature space* $\mathcal{S}$ *consists of three algorithms* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Ver})$ *with the following syntax*

- $\mathsf{KeyGen}() \xrightarrow{\$} (\mathsf{vk}, \mathsf{sk})$*: outputs a verification key* vk *and a signing key* sk.
- $\mathsf{Sign}(\mathsf{sk}, m) \xrightarrow{\$} \sigma$*: receives a signing key* sk *and a message* $m \in \mathcal{M}$ *as input, and outputs a signature* $\sigma \in \mathcal{S}$.
- $\mathsf{Ver}(\mathsf{vk}, m, \sigma) \to$ true/false*: receives a verification key* vk*, a message* $m \in \mathcal{M}$ *and a signature* $\sigma \in \mathcal{S}$ *as input, and outputs* true *(indicating a valid signature) or* false *(indicating an invalid signature).*

A signature scheme $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Ver})$ *is* δ-correct*, if for every message* $m \in \mathcal{M}$*, we have*

$$|\Pr[\mathsf{Ver}(\mathsf{vk}, m, \sigma) \mid (\mathsf{vk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}(), \sigma \xleftarrow{\$} \mathsf{Sign}(\mathsf{sk}, m)]| \geq 1 - \delta.$$

Definition 9 (EUF-CMA). *We call a signature scheme* $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Ver})$ (t, ε, q)*-EUF-CMA secure, if for every adversary* $\mathcal{A}$ *running in time at most* t *making at most* q *queries to the* Sign *oracle we have that*

$$\Pr[\mathcal{G}_{\Sigma}^{\mathsf{euf\text{-}cma}}(\mathcal{A}) = 1] \leq \varepsilon,$$

where $\mathcal{G}_{\Sigma}^{\mathsf{euf\text{-}cma}}(\mathcal{A})$ *is given in Figure 3.*

$\underline{\mathcal{G}_{\Sigma}^{\mathsf{euf\text{-}cma}}(\mathcal{A})}$:

1 $Q \leftarrow \emptyset$
2 $(\mathsf{vk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}()$
3 $(m^\star, \sigma^\star) \xleftarrow{\$} \mathcal{A}^{\textsc{Sign}}(\mathsf{vk})$
4 **return** $[\![\mathsf{m}^\star \notin Q \wedge \mathsf{Ver}(\mathsf{vk}, m^\star, \sigma^\star)]\!]$

$\underline{\textsc{Sign}(m)}$:

5 $\sigma \xleftarrow{\$} \mathsf{Sign}(\mathsf{sk}, m)$
6 $Q \leftarrow Q \cup \{m\}$
7 **return** σ

Fig. 3. Game for EUF-CMA security of a signature scheme $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Ver})$.

2.6 Puncturable PRFs

We recall the definition of puncturable pseudorandom functions (puncturable PRFs) from [46].

Definition 10 (Puncturable PRF). *A* puncturable PRF *for domain* $\mathcal{D}$ *and co-domain* $\mathcal{C}$ *consists of three polynomial time algorithms* $\mathsf{PPRF} = (\mathsf{KeyGen}, \mathsf{Puncture}, \mathsf{Eval})$ *where*

- $\mathsf{KeyGen}() \xrightarrow{\$} K$*: returns a PRF key* K*.*
- $\mathsf{Puncture}(K, x) \xrightarrow{\$} K_x$*: takes an unpunctured PRF key* K *and a puncturing point* $x \in \mathcal{D}$ *as input and outputs a punctured PRF key* K_x*.*
- $\mathsf{Eval}(K, x) \to y$*: takes a PRF key* K *and* x *as input, and outputs a value* $y \in \mathcal{C}$*.*

Definition 11 (Functionality preserving). *A puncturable PRF* PPRF *is* $(t_{\mathsf{fp}}, \varepsilon_{\mathsf{fp}})$-functionality preserving *iff for every adversary* $\mathcal{A}$ *running in time at most* t_{fp}*,*

- *for* $x \xleftarrow{\$} \mathcal{A}()$ *with* $x \in \mathcal{D}$*,*
- $K \xleftarrow{\$} \mathsf{KeyGen}()$*, and*
- $K_x \xleftarrow{\$} \mathsf{Puncture}(K, x)$*, we have*

$$\Pr[\forall x' \in \mathcal{D} \setminus \{x\} : \mathsf{Eval}(K, x') = \mathsf{Eval}(K_x, x')] \geq 1 - \varepsilon_{\mathsf{fp}}.$$

Definition 12 (Pseudorandomness at punctured points). *A puncturable PRF* PPRF *is* $(t_{\mathsf{pr}}, \varepsilon_{\mathsf{pr}})$-pseudorandom at punctured points *iff for every adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *running in time at most* t_{pr}

- *for* $(x, \mathsf{st}) \xleftarrow{\$} \mathcal{A}_1()$ *with* $x \in \mathcal{D}$
- $K \xleftarrow{\$} \mathsf{KeyGen}()$*,*
- $K_x \xleftarrow{\$} \mathsf{Puncture}(K, x)$*, and*
- $y \xleftarrow{\$} \mathcal{C}$*, we have*

$$\mathsf{Adv}^{\mathsf{pprf}}_{\mathsf{PPRF}}(\mathcal{A}) := |\Pr[\mathcal{A}_2(K_x, \mathsf{Eval}(K, x), \mathsf{st})] - \Pr[\mathcal{A}_2(K_x, y, \mathsf{st})]| \leq \varepsilon_{\mathsf{pr}},$$

where the probability is taken over the randomness used to sample K*,* K_x*,* y*, and the randomness of* $\mathcal{A}_1$ *and* $\mathcal{A}_2$*.*

2.7 Indistinguishability obfuscation

We also recall the definition of indistinguishability obfuscation (iO), first conceived by [4]. We use the definition of [25].

Definition 13 (iO). *An* indistinguishability obfuscator *is a PPT algorithm* iO *where*

- $\mathsf{iO}(C) \xrightarrow{\$} C'$*: receives as input a circuit* C *and outputs another circuit* C'*.*

An indistinguishability obfuscator iO *is* δ-correct *iff for every circuit* C *with* n *input bits and every* $x \in \{0,1\}^n$*, for* $C' \xleftarrow{\$} \mathsf{iO}(C)$ *we have* $\Pr[C'(x) = C(x)] \geq 1 - \delta$*.*

Every indistinguishability obfuscator should satisfy the following property:

Definition 14 (Security). *An indistinguishability obfuscator* iO *is* $(t_{\mathsf{io}}, \varepsilon_{\mathsf{io}})$-secure *iff for every adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *with combined runtime at most* t_{io}*, where* $\mathcal{A}_1$ *takes no input and outputs* (C_0, C_1, st) *such that there exists an* $\varepsilon \in [0, 1]$ *with*

$$\Pr[|C_0| = |C_1| \wedge \forall x : C_0(x) = C_1(x)] \geq 1 - \varepsilon,$$

where the probability is taken over the randomness of $\mathcal{A}_1$*, and for* $b \xleftarrow{\$} \{0,1\}$ *it holds that*

$$\mathsf{Adv}^{\mathsf{io}}_{\mathsf{iO}}(\mathcal{A}) := 2\left|\Pr[\mathcal{A}_2(\mathsf{st}, \mathsf{iO}(C_b)) = b] - \frac{1}{2}\right| \leq \varepsilon_{\mathsf{io}} + \varepsilon,$$

where the probability is taken over the randomness of b*,* $\mathcal{A}_1$*, and* $\mathcal{A}_2$*.*

3 Designated verifier signatures

Designated Verifier Signature (DVS) were introduced by Jakobsson, Sako, and Impagliazzo [30]. In contrast to plain signatures, here a signature is computed for a *designated verifier*. The signature convinces only the designated verifier of the authenticity of the message but no third party. To this end, we require that the designated verifier can *simulate* a signature. Hence, a third party cannot tell if the signer authenticated the message or if the designated verifier simulated the signature.

In this work we consider only *publicly verifiable* DVS. If the verifier's secret key is needed for verification, the scheme is called *privately verifiable* [38] or a *strong* DVS [36, 37][8].

Definition 15 (DVS). *A* designated verifier signature scheme *(DVS) is a tuple of algorithms* $\mathsf{DVS} = (\mathsf{Setup}, \mathsf{SKGen}, \mathsf{VKGen}, \mathsf{Sign}, \mathsf{Vrfy}, \mathsf{Sim})$ *along with a message space* $\mathcal{M}$ *and identity space* $\mathcal{IDS}$*.*

- $\mathsf{Setup}() \xrightarrow{\$} \mathsf{pp}$*: A probabilistic setup algorithm that outputs public parameters.*
- $\mathsf{SKGen}(\mathsf{pp}) \xrightarrow{\$} (\mathsf{pk}_S, \mathsf{sk}_S)$*: A probabilistic key generation algorithm that outputs a public-/secret-key pair for the signer.*
- $\mathsf{VKGen}(\mathsf{pp}, \mathsf{id}_D) \xrightarrow{\$} (\mathsf{pk}_D, \mathsf{sk}_D)$*: A probabilistic key generation algorithm that takes as input an identity and outputs a public-/secret-key pair for the verifier.*
- $\mathsf{Sign}(\mathsf{pp}, \mathsf{sk}_S, \mathsf{pk}_D, \mathsf{id}_D, m) \xrightarrow{\$} \sigma/\perp$*: A probabilistic signing algorithm that takes as input a signer secret key* sk_S*, a designated verifier public key* pk_D *and an identity* $\mathsf{id}_D \in \mathcal{IDS}$*, and a message* $m \in \mathcal{M}$*, and produces a signature* σ *or a dedicated error symbol* $\perp$*.*
- $\mathsf{Vrfy}(\mathsf{pp}, \mathsf{pk}_S, \mathsf{pk}_D, \mathsf{id}_D, m, \sigma) \to$ true/false*: A deterministic verification algorithm that checks a message* m *and signature* σ *against a signer public key* pk_S *and verifier public key* pk_D *with identity* id_D*.*
- $\mathsf{Sim}(\mathsf{pp}, \mathsf{pk}_S, \mathsf{sk}_D, \mathsf{id}_D, m) \xrightarrow{\$} \sigma/\perp$*: A probabilistic signature simulation algorithm that takes as input a signer public key* pk_S*, a designated verifier secret key* sk_D *and identity* $\mathsf{id}_D \in \mathcal{IDS}$*, and a message* $m \in \mathcal{M}$*, and produces a signature* σ *or a dedicated error symbol* $\perp$*.*

[8] Other work considers a scheme *strong* if it achieves *privacy of signer's identity* [35], formalizing prose definitions of prior work [30, 45]. This property is only achievable for privately verifiable schemes.

A DVS scheme DVS *is* δ-correct*, if, for any* $\mathsf{pp} \xleftarrow{\$} \mathsf{Setup}()$*, any* $\mathsf{id}_D \in \mathcal{IDS}$*, and honestly generated key pairs* $(\mathsf{pk}_S, \mathsf{sk}_S) \xleftarrow{\$} \mathsf{SKGen}(\mathsf{pp}), (\mathsf{pk}_D, \mathsf{sk}_D) \xleftarrow{\$} \mathsf{VKGen}(\mathsf{pp}, \mathsf{id}_D)$ *and every message* $m \in \mathcal{M}$*, it holds that*

$$\Pr[\mathsf{Vrfy}(\mathsf{pk}_S, \mathsf{pk}_D, \mathsf{id}_D, m, \mathsf{Sign}(\mathsf{sk}_S, \mathsf{pk}_D, m))] = 1 - \delta.$$

We add the concept of identities (associated with verifier keys but not with signer keys) to achieve our novel security notions against malicious verifiers. Note that we allow several key pairs per identity. Looking ahead, in Section 3.2 we argue why verifier identities are *necessary* to achieve our novel security notions. Hence, we require separate algorithms to generate signer and verifier keys, following Laguillaumie and Vergnaud [35] and Brendel, Fiedler, Günther, Janson, and Stebila [8].

We consider two security notions: *Unforgeability* (conceptually similar to unforgeability of (standard) signature schemes) and *source hiding* (a signature does not divulge if it was created by the signer or the designated verifier). In the following, we adapt these properties to our syntax including identities and introduce new, stronger versions allowing verifiers to generate their keys maliciously.

3.1 Unforgeability

Similar to unforgeability of (standard) signatures, the adversary has to create a signature for a fresh message while having access to a signing oracle. The challenger generates the sender key pair, whereas the adversary can obtain verifier public keys and verifier key pairs for identities of its choice with the VKeys oracle, accounting for the introduction of identities. Note that we allow the adversary to decide which one of the verifier public keys to challenge, further strengthening the definition. The adversary can query for signatures designated for any honestly generated verifier public key and identity (including the lines with gray background) as previously done by [34, 8]; or, for our novel notion *malicious unforgeability*, designated for any verifier public key and identity without restriction. In particular, this models a designated verifier possibly not knowing its own secret key ("suicide attack" [30]). A similar notion has been discussed by [48, 58] for multi-designated verifier signatures under the name *unforgeability under rogue key attacks*. Prior work [27, 40, 2, 3] provides the adversary with a simulation oracle as well; we opt against it do keep a clearer distinction between unforgeability and source hiding.

Our novel malicious unforgeability is important in a setting where a signer cannot assume that verifier keys were honestly generated. Looking ahead, Theorem 13 shows that signing for a maliciously generated verifier key may even leak the signer secret key if the scheme does not achieve malicious unforgeability.

Definition 16 (Unforgeability of DVS). *A designated verifier signature scheme* DVS *is* $(t, \varepsilon, q_S, q_{VK})$-malicious unforgeable *(resp. unforgeable) if, for any*

$\mathcal{G}^{\mathsf{mal\text{-}uf}\,\boxed{\mathsf{uf}}}_{\mathsf{DVS}}(\mathcal{A})$:

1 $Q_S, \mathcal{L}_V, Q_R \leftarrow \emptyset$
2 $\mathsf{pp} \xleftarrow{\$} \mathsf{Setup}()$
3 $(\mathsf{pk}_S, \mathsf{sk}_S) \xleftarrow{\$} \mathsf{SKGen}(\mathsf{pp})$
4 $(m^*, \sigma^*, \mathsf{id}^*, \mathsf{pk}^*_D) \xleftarrow{\$} \mathcal{A}^{\textsc{Sign},\textsc{VKeys}}(\mathsf{pp}, \mathsf{pk}_S)$
5 **if** $\exists \mathsf{sk}^*_D : (\mathsf{id}^*, \mathsf{pk}^*_D, \mathsf{sk}^*_D) \in \mathcal{L}_V$
6 $\quad d_1 \leftarrow \mathsf{Vrfy}(\mathsf{pp}, \mathsf{pk}_S, \mathsf{sk}^*_D, \mathsf{id}^*, m^*, \sigma^*)$
7 $\quad d_2 \leftarrow (\mathsf{id}^*, \mathsf{pk}^*_D, m^*) \notin Q_S$
8 $\quad d_3 \leftarrow (\mathsf{id}^*, \mathsf{pk}^*_D) \notin Q_R$
9 $\quad$ **return** $[\![d_1 \wedge\ d_2 \wedge\ d_3]\!]$
10 **else**
11 $\quad$ **return** 0

$\textsc{Sign}(\mathsf{pk}, \mathsf{id}, m)$:

12 **if** $\neg\exists \mathsf{sk} : (\mathsf{id}, \mathsf{pk}, \mathsf{sk}) \in \mathcal{L}_V$
//only accept honestly generated verifier keys
13 $\quad$ **return** $\perp$
14 $Q_S \leftarrow Q_S \cup \{(\mathsf{id}, \mathsf{pk}, m)\}$
15 $\sigma \xleftarrow{\$} \mathsf{Sign}(\mathsf{pp}, \mathsf{sk}_S, \mathsf{pk}, \mathsf{id}, m)$
16 **return** σ

$\textsc{VKeys}(\mathsf{id}, \mathsf{s})$:

17 $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{VKGen}(\mathsf{pp}, \mathsf{id})$
18 $\mathcal{L}_V \leftarrow \mathcal{L}_V \cup \{(\mathsf{id}, \mathsf{pk}, \mathsf{sk})\}$
19 **if** $\mathsf{s} = \text{true}$
20 $\quad Q_R \leftarrow Q_R \cup \{(\mathsf{id}, \mathsf{pk})\}$
21 $\quad$ **return** $(\mathsf{pk}, \mathsf{sk})$
22 **else**
23 $\quad$ **return** pk

Fig. 4. Unforgeability limited to honestly generated verifier keys (including the parts with gray background) or allowing malicious verifier keys (without gray background) of a designated verifier signature (cp. Definition 16).

adversary $\mathcal{A}$ with running time at most t, making at most q_S queries to the Sign *oracle and at most q_{VK} queries to the* VKeys *oracle we have that*

$$\mathsf{Adv}^{\mathsf{mal\text{-}uf}\,\boxed{\mathsf{uf}}}_{\mathsf{DVS}}(\mathcal{A}) := \Pr\left[\mathcal{G}^{\mathsf{mal\text{-}uf}\,\boxed{\mathsf{uf}}}_{\mathsf{DVS}}(\mathcal{A}) = 1\right] \leq \varepsilon,$$

where $\mathcal{G}^{\mathsf{mal\text{-}uf}\,\boxed{\mathsf{uf}}}_{\mathsf{DVS}}(\mathcal{A})$ is as defined in Figure 4 and lines with gray background are considered only for (honest) unforgeability.

Remark 1. Note that $(t, \varepsilon, q_S, 1)$-malicious unforgeability implies $(t, \varepsilon \cdot q_{VK}, q_S, q_{VK})$-malicious unforgeability with a reduction that guesses the adversary's output verifier key and generates the remaining verifier keys itself. For (honest) unforgeability, this reduction additionally requires the scheme to be source hiding so that the reduction can answer the Sign queries to other designated verifiers with Sim.

3.2 Source hiding

Furthermore, we require that the DVS scheme is *source hiding*, i.e. an adversary cannot tell whether a signature was created by the sender (with Sign) or the designated verifier (with Sim). We follow the formalization of Brendel et al. [8], who discuss that the same property has been around with different names (designated verifier property, non-transferability, untransferability, source deniable, off-the-record). We incorporate verifier identities into the notion of [8]: The adversary chooses the verifier identity and the game honestly generates the verifier key pair and gives it to the adversary. Prior work has proposed definitions

which are in some aspects weaker: The adversary does not get the secret keys [38, 37, 44, 10, 57, 50, 52] (but instead a signing oracle [10, 57]), the adversary is limited to a single challenge [38, 53, 37, 10, 57, 51, 35], or the challenge message is randomly chosen [38, 53, 37, 51].

While this provides a guarantee against a third party who can coerce users into giving up their secret keys, it does not protect against verifiers who generate their keys maliciously.

Definition 17 (Source hiding of DVS). *A designated verifier signature scheme* DVS *is* (t, ε, q_C)-source hiding *if, for any adversary* $\mathcal{A}$ *with running time at most* t *and making at most* q_C *queries to the* CHALL *oracle, we have that*

$$\mathsf{Adv}^{\mathsf{sh}}_{\mathsf{DVS}}(\mathcal{A}) := 2 \cdot \left| \Pr\left[\mathcal{G}^{\mathsf{sh}}_{\mathsf{DVS}}(\mathcal{A}) = 1\right] - \frac{1}{2} \right| \leq \varepsilon,$$

where $\mathcal{G}^{\mathsf{sh}}_{\mathsf{DVS}}(\mathcal{A})$ *is defined in Figure 5.*

We define the following strengthening of source hiding that all-quantifies over the verifier key pairs instead of generating them honestly. Looking ahead, we will use this notion as a tool for building a simplified malicious source hiding DVS in the ROM in Section 4. Note that this notion is still too weak to meaningfully capture security against maliciously generated verifier keys, since it does not enforce that verifiers *know* their secret keys.

Definition 18 (All verifier key source hiding of DVS). *A designated verifier signature scheme* DVS *(with associated identity space* $\mathcal{IDS}$*) is* (t, ε, q_C)-all verifier key source hiding *for the NP-relation* $\mathcal{R}$ *(where* $((\mathsf{id}, \mathsf{pk}_D), \mathsf{sk}_D) \in \mathcal{R}$ *for all* $\mathsf{pp} \in \mathsf{Img}(\mathsf{Setup}()), \mathsf{id} \in \mathcal{IDS}, (\mathsf{pk}_D, \mathsf{sk}_D) \in \mathsf{Img}(\mathsf{VKGen}(\mathsf{pp}, \mathsf{id}))$*) iff, for all adversaries* $\mathcal{A}$ *with running time at most* t *that make at most* q_C *queries to the* CHALL *oracle, we have for all auxiliary input* $\mathsf{aux} \in \{0,1\}$ *and for all* $((\mathsf{id}_D, \mathsf{pk}_D), \mathsf{sk}_D) \in \mathcal{R}$

$$\mathsf{Adv}^{\mathsf{all\text{-}sh}}_{\mathsf{DVS},\mathsf{id}_D,\mathsf{pk}_D,\mathsf{sk}_D,\mathsf{aux}}(\mathcal{A}) := 2 \cdot \left| \Pr\left[\mathcal{G}^{\mathsf{all\text{-}sh}}_{\mathsf{DVS},\mathsf{id}_D,\mathsf{pk}_D,\mathsf{sk}_D,\mathsf{aux}}(\mathcal{A}) = 1\right] - \frac{1}{2} \right| \leq \varepsilon,$$

where $\mathcal{G}^{\mathsf{all\text{-}sh}}_{\mathsf{DVS},\mathsf{id}_D,\mathsf{pk}_D,\mathsf{sk}_D,\mathsf{aux}}(\mathcal{A})$ *is defined in Figure 5.*

Remark 2. In the definition of all verifier key source hiding, the auxiliary input can be omitted when the adversary is non-uniform, because it can have the auxiliary input hard-coded. Since we use in this work the concrete security model (and not the asymptotic security model), all adversaries are automatically non-uniform. We make the auxiliary input nevertheless explicit in our definitions, so that they can be translated easily to the asymptotic security model.

Finally, we introduce our novel *simplified malicious source hiding* definition. Intuitively, this notion models that the source of a signature remains hidden, even if the verifier generated his keys maliciously. We prefix it *simplified* since this notion alone is not yet practically meaningful, but already unachievable in

$\underline{\mathcal{G}^{\mathsf{sh}}_{\mathsf{DVS}}(\mathcal{A})}$:

1 $\mathsf{pp} \xleftarrow{\$} \mathsf{Setup}()$
2 $\mathsf{id}_D \leftarrow \bot$
3 $(\mathsf{pk}_S, \mathsf{sk}_S) \xleftarrow{\$} \mathsf{SKGen}(\mathsf{pp})$
4 $b \xleftarrow{\$} \{0,1\}$
5 $b' \xleftarrow{\$} \mathcal{A}^{\textsc{Chall},\textsc{VKey}}(\mathsf{pp}, \mathsf{pk}_S, \mathsf{sk}_S)$
6 **return** $[\![b' = b]\!]$

$\underline{\textsc{VKey}(\mathsf{id})}$:

7 **if** $\mathsf{id}_D \neq \bot$
//one-time oracle: set verifier identity at first query
8 **return** $\bot$
9 $(\mathsf{pk}_D, \mathsf{sk}_D) \xleftarrow{\$} \mathsf{VKGen}(\mathsf{pp}, \mathsf{id})$
10 $\mathsf{id}_D \leftarrow \mathsf{id}$
11 **return** $(\mathsf{pk}_D, \mathsf{sk}_D)$

$\underline{\textsc{Chall}(m)}$:

12 **if** $\mathsf{id}_D = \bot$
13 **return** $\bot$
14 **if** $b = 0$
15 $\sigma \xleftarrow{\$} \mathsf{Sign}(\mathsf{pp}, \mathsf{sk}_S, \mathsf{pk}_D, \mathsf{id}_D, m)$
16 **else**
17 $\sigma \xleftarrow{\$} \mathsf{Sim}(\mathsf{pp}, \mathsf{pk}_S, \mathsf{sk}_D, \mathsf{id}_D, m)$
18 **return** σ

$\underline{\mathcal{G}^{\mathsf{all\text{-}sh}}_{\mathsf{DVS},\mathsf{id}_D,\mathsf{pk}_D,\mathsf{sk}_D,\mathsf{aux}}(\mathcal{A})}$:

1 $\mathsf{pp} \xleftarrow{\$} \mathsf{Setup}()$
2 $(\mathsf{pk}_S, \mathsf{sk}_S) \xleftarrow{\$} \mathsf{SKGen}(\mathsf{pp})$
3 $b \xleftarrow{\$} \{0,1\}$
4 $b' \xleftarrow{\$} \mathcal{A}^{\textsc{Chall}}(\mathsf{pp}, \mathsf{pk}_S, \mathsf{sk}_S, \mathsf{aux})$
5 **return** $[\![b' = b]\!]$

$\underline{\textsc{Chall}(m)}$:

6 **if** $b = 0$
7 $\sigma \xleftarrow{\$} \mathsf{Sign}(\mathsf{pp}, \mathsf{sk}_S, \mathsf{pk}_D, \mathsf{id}_D, m)$
8 **else**
9 $\sigma \xleftarrow{\$} \mathsf{Sim}(\mathsf{pp}, \mathsf{pk}_S, \mathsf{sk}_D, \mathsf{id}_D, m)$
10 **return** σ

$\underline{\mathcal{G}^{\mathsf{s\text{-}mal\text{-}sh}}_{\mathsf{DVS},\mathsf{aux}}(\mathcal{A},\mathcal{D})}$:

1 $\mathsf{pp} \xleftarrow{\$} \mathsf{Setup}()$
2 $(\mathsf{pk}^*_S, \mathsf{sk}^*_S) \xleftarrow{\$} \mathsf{SKGen}(\mathsf{pp})$
3 $\mathcal{L}_V, R \leftarrow \varepsilon$
//save adversary's verifier keys and randomness
4 $b \xleftarrow{\$} \{0,1\}$
5 $\mathsf{st} \leftarrow \mathcal{A}^{\textsc{Chall},\textsc{VKeys},\textsc{Random}}(\mathsf{pp}, \mathsf{pk}^*_S, \mathsf{aux})$
6 $b' \xleftarrow{\$} \mathcal{D}(\mathsf{st}, \mathsf{sk}^*_S)$
7 **return** $[\![b' = b]\!]$

$\underline{\textsc{VKeys}(\mathsf{id})}$:

8 $(\mathsf{pk}_D, \mathsf{sk}_D) \xleftarrow{\$} \mathsf{VKGen}(\mathsf{pp}, \mathsf{id})$
9 $\mathcal{L}_V \xleftarrow{+} (\mathsf{pk}, \mathsf{id})$ //append
10 **return** pk_D

$\underline{\textsc{Chall}(\mathsf{pk}, \mathsf{id}, m)}$: //only one query

11 **if** $(\mathsf{pk}, \mathsf{id}) \in \mathcal{L}_V$ //ensure malicious verifier
12 **return** $\bot$
13 **if** $b = 0$
14 $\sigma \xleftarrow{\$} \mathsf{Sign}(\mathsf{pp}, \mathsf{sk}^*_S, \mathsf{pk}, \mathsf{id}, m)$
15 **else**
16 $\sigma \xleftarrow{\$} \mathcal{E}_{\mathcal{A}}(\mathsf{pp}, \mathsf{pk}^*_S, \mathsf{aux}, R, \mathcal{L}_V, \mathsf{pk}, \mathsf{id}, m)$
17 **return** σ

$\underline{\textsc{Random}(n)}$:

18 $r \xleftarrow{\$} \{0,1\}^n$
19 $R \xleftarrow{+} r$ //append
20 **return** r

Fig. 5. Source hiding (top), all-verifier source hiding (middle), and simplified malicious source hiding (bottom) of a designated verifier signature (cp. Definitions 17 to 19).

the standard model as we will show in Section 5.1. In the big picture, we have an adversary $\mathcal{A}$ who can choose verifier identity and public key arbitrarily, the challenge oracle which executes Sign or extracts a signature from the adversary (thereby modeling that some malicious verifier could have produced a signature), and a distinguisher, who has to make a guess based on the adversary's output. This also addresses discussions of prior work [30, 58] of a designated verifier generating his public key in a way that he arguably does not know his own secret key and thereby undermines source hiding.

Let us go through the design choices that led to this definition. The adversary gets access to some auxiliary data aux, modeling that the adversary has access to arbitrary data generated independently of the public parameters of the scheme. Note that the auxiliary input is sampled independently of the public parameters; otherwise, the auxiliary input could include a public verifier key without the secret key or any hint how it has been generated. Simulating a signature for that key is then as hard as breaking the unforgeability of the signature. Hence, a scheme without public parameters cannot satisfy simplified malicious source hiding. Furthermore, the public parameters for this scheme must not be reused for another scheme (without re-proving security), because this might give an adversary additional auxiliary input that depends on the public parameters.

Design choices for identities. Without verifier identities, an adversary can copy any verifier public key off the internet, pass it off as its own, and then argue that it could not have generated the signature itself, thereby undermining simplified malicious source hiding. To defend against this type of attack we add the concept of identities to verifier keys. A DVS scheme has to ensure that a verifier public key cannot be used with a different identity, thus precluding this attack.

Design choices for the adversary. Unlike for (honest) source hiding the adversary $\mathcal{A}$ does not get the challenge signer secret key; otherwise, the extractor would get it as input as well and could trivially execute Sign, rendering the notion meaningless. We model the randomness of the adversary explicitly with the Random oracle. We limit the adversary to one query to the Chall oracle. We allow the adversary to obtain verifier public keys of other users via the VKeys oracle. For a more realistic notion the adversary should have access to more key (pairs) and honestly generated signatures. Looking ahead, in Section 5.1 we show that this notion is already unachievable in the standard model.

Design choices for the challenge oracle. The challenge oracle answers for verifier keys controlled by the adversary with the extractor. The extractor gets the same arguments as the adversary, and additionally all oracle responses that the adversary has obtained so far.

Design choices for the distinguisher. The distinguisher cannot get access to the challenge oracle since it learns the challenge signer secret key.

Definition 19 (Simplified malicious source hiding of DVS). *A designated verifier signature scheme* DVS *is* $(t_{\mathcal{A}}, t_{\mathcal{E}}, t_{\mathcal{D}}, \varepsilon, q_{VK})$-simplified malicious source

hiding *iff for all adversaries* $\mathcal{A}$ *with running time at most* $t_{\mathcal{A}}$ *and making at most* q_{VK} *queries to the* VKeys *oracle there exists an extractor* $\mathcal{E}_{\mathcal{A}}$ *running in time at most* $t_{\mathcal{E}}$ *such that for any distinguisher running in time at most* $t_{\mathcal{D}}$ *and for all auxiliary inputs* $\mathsf{aux} \in \{0,1\}^*$ *we have*

$$\mathsf{Adv}^{\mathsf{s\text{-}mal\text{-}sh}}_{\mathsf{DVS},\mathsf{aux}}(\mathcal{A},\mathcal{D}) := 2 \cdot \left| \Pr\left[\mathcal{G}^{\mathsf{s\text{-}mal\text{-}sh}}_{\mathsf{DVS},\mathsf{aux}}(\mathcal{A},\mathcal{D}) = 1\right] - \frac{1}{2} \right| \leq \varepsilon,$$

where $\mathcal{G}^{\mathsf{s\text{-}mal\text{-}sh}}_{\mathsf{DVS},\mathsf{aux}}(\mathcal{A},\mathcal{D})$ *is defined in Figure 5.*

Remark 3. In the definition of simplified malicious source hiding, the auxiliary input *cannot* be omitted, even for non-uniform adversaries. The reason is that the extractor can depend in an arbitrary way on the adversary (and thus everything that is hardcoded in the adversary), but the extractor cannot depend on the auxiliary input. For example, if $\mathcal{A}$ has a hardcoded public key, the extractor $\mathcal{E}_{\mathcal{A}}$ is allowed to have a corresponding secret key hardcoded (if it exists). However, if $\mathcal{A}$ gets a public key as part of the auxiliary input, the extractor only gets access to the public key (since it gets the same auxiliary input), but no corresponding secret key.

4 Candidate construction

4.1 A DVS from NIWIPoKs

The starting point for our construction is the following simple construction of a DVS satisfying all verifier key source hiding for an arbitrary relation $\mathcal{R}_V$[9] with a (t,ε)-hard on average search problem for instance sampler Samp (c.f. Definition 1).

The construction makes use of an ordinary signature scheme $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Ver})$ and a NIWIPoK NIWI for the relation

$$\mathcal{R}_N = \{((\mathsf{id}, \mathsf{vk}, m, x), (\sigma, w)) \mid \mathsf{Ver}(\mathsf{vk}, \mathsf{id}||m, \sigma) \vee (x, w) \in \mathcal{R}_V\}$$

and is given in Figure 6.

Theorem 2 (Correctness). *If* Σ *is* δ_1*-correct and* NIWI *is* δ_2*-correct, then* $\mathsf{DVS}[\mathsf{Samp}, \Sigma, \mathsf{NIWI}]$ *as given in Figure 6 is* $\delta_1 + \delta_2$*-correct.*

Proof. The signature $\sigma \xleftarrow{\$} \mathsf{Sign}(\mathsf{sk}, \mathsf{id}_D||m)$ verifies with probability at least $1-\delta_1$ and in this case $(\sigma, \perp)$ is a valid witness for the statement $(\mathsf{id}_D, \mathsf{vk}, m, \mathsf{pk}_D)$ in $\mathcal{R}_N$. Since NIWI is δ_2-correct, this results in $\mathsf{DVS}[\mathsf{Samp}, \Sigma, \mathsf{NIWI}]$ being $\delta_1 + \delta_2$-correct. □

Theorem 3 (Malicious unforgeability). *If* Σ *is* $(t, q_S, \varepsilon_\Sigma)$*-EUF-CMA secure, the search problem with instance sampler* Samp *for* $\mathcal{R}_V$ *is* $(t, \varepsilon_{\mathcal{R}_V})$*-hard on average, and* NIWI *is* $(t, \varepsilon_{\mathsf{td}}, t, \varepsilon_{\mathsf{ks}})$*-knowledge sound with algorithms* $\mathsf{TDGen}_{\mathsf{ks}}$ *and* Extr*, then* $\mathsf{DVS}[\mathsf{Samp}, \Sigma, \mathsf{NIWI}]$ *as given in Figure 6 is* $(t', \varepsilon, q_S, q_{VK})$*-malicious unforgeable for arbitrary* q_{VK}*,* $t' \approx t$*, and* $\varepsilon \leq \varepsilon_{\mathsf{td}} + \varepsilon_{\mathsf{ks}} + \varepsilon_\Sigma + \varepsilon_{\mathcal{R}_V} \cdot q_{VK}$*.*

[9] Since we are aiming for all verifier key source hiding we can ignore identities for this scheme.

$\underline{\mathsf{Setup}()}$:
1 **return** $\mathsf{crs} \xleftarrow{\$} \mathsf{Gen}_{\mathsf{NIWI}}()$

$\underline{\mathsf{SKGen}(\mathsf{crs})}$:
2 $(\mathsf{vk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}()$
3 **return** $(\mathsf{vk}, (\mathsf{vk}, \mathsf{sk}))$

$\underline{\mathsf{VKGen}(\mathsf{crs}, \mathsf{id}_D)}$:
4 **return** $(x, w) \xleftarrow{\$} \mathsf{Samp}()$

$\underline{\mathsf{Sign}(\mathsf{crs}, \mathsf{sk}_S = (\mathsf{vk}, \mathsf{sk}), \mathsf{id}_D, \mathsf{pk}_D, m)}$:
5 $\sigma \xleftarrow{\$} \mathsf{Sign}(\mathsf{sk}, \mathsf{id}_D || m)$
6 **return** $\pi \xleftarrow{\$} \mathsf{Prove}(\mathsf{crs}, (\mathsf{id}_D, \mathsf{vk}, m, \mathsf{pk}_D), (\sigma, \bot))$

$\underline{\mathsf{Sim}(\mathsf{pp}, \mathsf{pk}_S, \mathsf{sk}_D, \mathsf{id}_D, m)}$:
7 **return** $\pi \xleftarrow{\$} \mathsf{Prove}(\mathsf{crs}, (\mathsf{id}_D, \mathsf{vk}, m, \mathsf{pk}_D), (\bot, \mathsf{sk}_D))$

$\underline{\mathsf{Vrfy}(\mathsf{crs}, \mathsf{pk}_S, \mathsf{id}_D, \mathsf{pk}_D, m, \pi)}$:
8 **return** $\mathsf{Vrfy}_{\mathsf{NIWI}}(\mathsf{crs}, (\mathsf{id}_D, \mathsf{pk}_S, m, \mathsf{pk}_D), \pi)$

Fig. 6. Our construction of a DVS $\mathsf{DVS}[\mathsf{Samp}, \Sigma, \mathsf{NIWI}] = (\mathsf{Setup}, \mathsf{SKGen}, \mathsf{VKGen}, \mathsf{Sign}, \mathsf{Sim}, \mathsf{Vrfy})$ that satisfies all verifier key source hiding built from an instance sampler Samp for an NP relation, an ordinary signature scheme $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Ver})$, and a NIWIPoK $\mathsf{NIWI} = (\mathsf{Gen}_{\mathsf{NIWI}}, \mathsf{Prove}, \mathsf{Vrfy}_{\mathsf{NIWI}})$.

Proof. The proof uses a hybrid argument. Let G_0 be the real malicious unforgeability game for $\mathsf{DVS}[\mathsf{Samp}, \Sigma, \mathsf{NIWI}]$ and let G_1 be the same game, except that the crs for NIWI is sampled as $(\mathsf{crs}, \mathsf{td}_{\mathsf{ks}}) \xleftarrow{\$} \mathsf{TDGen}_{\mathsf{ks}}()$ instead of using $\mathsf{crs} \xleftarrow{\$} \mathsf{Gen}_{\mathsf{NIWI}}()$.

Lemma 1 ($\mathsf{G}_0 \rightsquigarrow \mathsf{G}_1$). $\left|\Pr[\mathsf{G}_0^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}_1^{\mathcal{A}} \Rightarrow 1]\right| \leq \varepsilon_{\mathsf{td}}$

The proof is a straightforward reduction to the first condition of knowledge soundness.

Lemma 2 (G_1). $\Pr[\mathsf{G}_1^{\mathcal{A}} \Rightarrow 1] \leq \varepsilon_{\mathsf{ks}} + \varepsilon_{\Sigma} + \varepsilon_{\mathcal{R}_V} \cdot q_{VK}$

Proof. We give a reduction to the EUF-CMA security of Σ. The reduction generates $(\mathsf{crs}, \mathsf{td}_{\mathsf{ks}}) \xleftarrow{\$} \mathsf{TDGen}_{\mathsf{ks}}()$ and sends crs along with the public key pk_S obtained from the signature scheme to the verifier. The reduction simulates all VKeys$(\mathsf{id}, \mathsf{s})$ queries as in the real game. It simulates Sign$(\mathsf{pk}, \mathsf{id}, m)$ queries by querying Sign$(\mathsf{id}||m)$ for the underlying signature scheme to get σ, computing $\pi \xleftarrow{\$} \mathsf{Prove}(\mathsf{crs}, (\mathsf{id}_D, \mathsf{vk}, m, \mathsf{pk}_D), (\sigma, \bot))$ and sending π to the adversary. We omit Vrfy oracle queries here, because they are unnecessary for a DVS, since the adversary can verify the signatures itself. This simulates the view of the adversary in the malicious unforgeability game perfectly.

When the adversary outputs $(m^*, \pi^*, \mathsf{id}^*)$ where id^* has been queried to VKeys before and answered with pk_D^* and $\mathsf{Vrfy}_{\mathsf{NIWI}}(\mathsf{crs}, (\mathsf{id}^*, \mathsf{pk}_S, m^*, \mathsf{pk}_D^*), \pi^*) =$ true, the reduction runs $(\sigma, w) \xleftarrow{\$} \mathsf{Extr}(\mathsf{crs}, \mathsf{td}_{\mathsf{ks}}, (\mathsf{id}^*, \mathsf{pk}_S, m^*, \mathsf{pk}_D^*), \pi^*)$ and submits σ as signature for $\mathsf{id}^*||m^*$ for the underlying signature scheme. When $(\mathsf{id}^*, m^*) \notin Q$, the reduction did not query for a signature for $\mathsf{id}^*||m^*$ before.

The signature produced by the reduction is guaranteed to verify if the adversary was successful and none of the following two bad events occurred:

- $\mathsf{Bad}_1 := (\sigma, w) \notin \mathcal{R}_N$
- $\mathsf{Bad}_2 := (\sigma, w) \in \mathcal{R}_N \wedge \mathsf{Ver}(\mathsf{pk}_S, \mathsf{id}^*||m^*, \sigma) = \text{false}$

We next bound the probability of these two bad events.

Claim 4. $\Pr[\mathsf{Bad}_1] \leq \varepsilon_{\mathsf{ks}}$

Proof. We describe a simple reduction that wins the second part of the knowledge soundness game if Bad_1 occurs. The reduction receives crs and simulates all other parts of the malicious unforgeability game itself. When the adversary outputs $(m^*, \pi^*, \mathsf{id}^*)$, it submits $((\mathsf{id}^*, \mathsf{pk}_S, m^*, \mathsf{pk}_D^*), \pi^*)$ (where pk_S is the signer public key generate initially and pk_D^* is the verifier public key generated for id^*) to the unforgeability game. Clearly, the adversary's view is exactly as in G_1 and the reduction wins the knowledge soundness game if Bad_1 occurs. □

Claim 5. $\Pr[\mathsf{Bad}_2] \leq \varepsilon_{\mathcal{R}_V} \cdot q_{VK}$

Proof. We give a simple reduction that solves the search problem for $\mathcal{R}_V$ on average, if Bad_2 occurs. The reduction is given a random instance x and simulates the game G_1 for the adversary honestly, except that it answers the i-th query to the VKeys oracle (where $i \xleftarrow{\$} [q_{VK}]$) with x as public key.

When the adversary outputs $(m^*, \pi^*, \mathsf{id}^*)$ and id^* has been queried in the i-th VKeys query before (which happens with probability $1/q_{VK}$) and $\mathsf{Vrfy}_{\mathsf{NIWI}}(\mathsf{crs}, (\mathsf{id}^*, \mathsf{pk}_S, m^*, x), \pi^*) = \text{true}$, the reduction runs $(\sigma, w) \xleftarrow{\$} \mathsf{Extr}(\mathsf{crs}, \mathsf{td}_{\mathsf{ks}}, (\mathsf{id}^*, \mathsf{pk}_S, m^*, x), \pi^*)$ and sends w to its game.

Clearly, the adversary's view is exactly as in G_1 and the reduction solves the search problem game if Bad_2 occurs, since Bad_2 implies $(x, w) \in \mathcal{R}_V$. □

Combining Claims 4 and 5 with the analysis of the reduction to the EUF-CMA security of Σ yields Lemma 2. □

Theorem 3 follows from combining Lemmata 1 and 2. □

Theorem 6 (All verifier key source hiding). *If* NIWI *is* (t, ε)*-witness indistinguishable, then* $\mathsf{DVS}[\mathsf{Samp}, \Sigma, \mathsf{NIWI}]$ *as given in Figure 6 is* (t', ε', q_C)*-all verifier key source hiding for arbitrary* q_C*,* $t' \approx t$*, and* $\varepsilon' \leq q_C \cdot \varepsilon$*.*

Proof. We describe a reduction for one Chall query. The argument can be extended to many Chall queries with a standard hybrid argument.

The reduction has the game parameters $\mathsf{id}_D, \mathsf{pk}_D, \mathsf{sk}_D, \mathsf{aux}$ hardcoded. It gets crs from the witness indistinguishability game and samples $(\mathsf{pk}_S, \mathsf{sk}_S) \xleftarrow{\$} \mathsf{SKGen}(\mathsf{crs})$ and runs the adversary on $\mathsf{pp}, \mathsf{pk}_S, \mathsf{sk}_S, \mathsf{aux}$. When the adversary makes a challenge query on message m, the reduction answers this with a proof for the statement $(\mathsf{id}_D, \mathsf{vk}, m, \mathsf{pk}_D)$ generated via the witness indistinguishability game with witnesses $(\sigma, \bot)$ and $(\bot, \mathsf{sk}_D)$, where $\sigma \xleftarrow{\$} \mathsf{Sign}(\mathsf{sk}, \mathsf{id}_D || m)$. The reduction outputs the same bit that the adversary outputs.

If the witness indistinguishability game uses the witness $(\sigma, \bot)$, the reduction perfectly simulates the all verifier key source hiding game for $b = 0$. If the witness $(\bot, \mathsf{sk}_D)$ is used, the reduction perfectly simulates the game for $b = 1$.

Hence, we get $\varepsilon' \leq \varepsilon$ for a single Chall query, or $\varepsilon' \leq q_C \cdot \varepsilon$ for multiple challenge queries with a hybrid argument. □

4.2 Construction in the random oracle model.

In the random oracle model each algorithm (including the adversary and the extractor) has oracle access to a uniformly random function $f : \{0,1\}^* \to \{0,1\}^\ell$

that models a cryptographic hash function. For our definitions with auxiliary input, we allow the auxiliary input to depend on $q_{\mathcal{H},\mathsf{aux}}$ function values of f.

In this section we present a generic transformation of an all verifier key source hiding DVS $\mathsf{DVS} = (\mathsf{Setup}, \mathsf{SKGen}, \mathsf{VKGen}, \mathsf{Sign}, \mathsf{Vrfy})$ and a NIZKPoK in the uniform random string (urs) model for the relation

$$\mathcal{R} = \{((\mathsf{pp}, \mathsf{id}_D, \mathsf{pk}_D), r) \mid \exists \mathsf{sk}_D : (\mathsf{pk}_D, \mathsf{sk}_D) \leftarrow \mathsf{VKGen}(\mathsf{pp}, \mathsf{id}_D; r)\}$$

to a simplified malicious source hiding DVS in the random oracle model (ROM). The result is in contrast to our impossibility result presented in Section 5 which works for any construction in the standard model. This shows that replacing the random oracle with any concrete hash functions (which has to be done to deploy such a scheme in the real world, where a random oracle is not available) yields an insecure construction.

If the underlying DVS is unforgeable or malicious unforgeable, then the resulting DVS achieves the same notion of unforgeability. The construction is given in Figure 7.

$\underline{\mathsf{Setup}'():}$

1 $\mathsf{pp} \xleftarrow{\$} \mathsf{Setup}()$
2 $s \xleftarrow{\$} \{0,1\}^\lambda$
3 **return** $\mathsf{pp}' = (\mathsf{pp}, s)$

$\underline{\mathsf{SKGen}'(\mathsf{pp}' = (\mathsf{pp}, s)):}$

4 **return** $\mathsf{SKGen}(\mathsf{pp})$

$\underline{\mathsf{VKGen}'(\mathsf{pp}' = (\mathsf{pp}, s), \mathsf{id}_D):}$

5 $(\mathsf{pk}_D, \mathsf{sk}_D) \leftarrow \mathsf{VKGen}(\mathsf{pp}, \mathsf{id}_D; r)$
6 $\mathsf{urs} \leftarrow \mathcal{H}(s, \mathsf{id}_D, \mathsf{pk}_D)$
7 $\pi \xleftarrow{\$} \mathsf{Prove}(\mathsf{urs}, (\mathsf{pp}, \mathsf{id}_D, \mathsf{pk}_D), r)$
8 **return** $(\mathsf{pk}'_D = (\mathsf{pk}_D, \pi), (\mathsf{pk}_D, \mathsf{sk}_D))$

$\underline{\mathsf{Vrfy}'(\mathsf{pp}, \mathsf{pk}_S, \mathsf{pk}'_D = (\mathsf{pk}_D, \pi), \mathsf{id}_D, m, \sigma):}$

9 **return** $\mathsf{Vrfy}(\mathsf{pp}, \mathsf{pk}_S, \mathsf{pk}_D, \mathsf{id}_D, m, \sigma)$

$\underline{\mathsf{Sign}'(\mathsf{pp}' = (\mathsf{pp}, s), \mathsf{sk}_S, \mathsf{id}_D, \mathsf{pk}'_D, m):}$

10 $(\mathsf{pk}_D, \pi) \leftarrow \mathsf{pk}'_D$
11 $\mathsf{urs} \leftarrow \mathcal{H}(s, \mathsf{id}_D, \mathsf{pk}_D)$
12 **if** $\mathsf{Vrfy}_{\mathsf{NIZK}}(\mathsf{urs}, (\mathsf{pp}, \mathsf{id}_D, \mathsf{pk}_D), \pi)$
13 **return** $\mathsf{Sign}(\mathsf{pp}, \mathsf{sk}_S, \mathsf{id}_D, \mathsf{pk}_D, m)$
14 **else**
15 **return** $\perp$

$\underline{\mathsf{Sim}'(\mathsf{pp}, \mathsf{pk}_S, \mathsf{sk}'_D = (\mathsf{pk}_D, \mathsf{sk}_D), \mathsf{id}_D, m):}$

16 $\mathsf{urs} \leftarrow \mathcal{H}(s, \mathsf{id}_D, \mathsf{pk}_D)$
17 **if** $\mathsf{Vrfy}_{\mathsf{NIZK}}(\mathsf{urs}, (\mathsf{pp}, \mathsf{id}_D, \mathsf{pk}_D), \pi)$
18 **return** $\mathsf{Sim}(\mathsf{pp}, \mathsf{pk}_S, \mathsf{sk}_D, \mathsf{id}_D, m)$
19 **else**
20 **return** $\perp$

Fig. 7. Our construction of a simplified malicious source hiding DVS $\mathsf{DVS}'[\mathsf{DVS}, \mathsf{NIZK}] = (\mathsf{Setup}', \mathsf{SKGen}', \mathsf{VKGen}', \mathsf{Sign}', \mathsf{Sim}', \mathsf{Vrfy}')$ built from an all verifier key source hiding DVS $\mathsf{DVS} = (\mathsf{Setup}, \mathsf{SKGen}, \mathsf{VKGen}, \mathsf{Sign}, \mathsf{Sim}, \mathsf{Vrfy})$, a NIZKPOK $\mathsf{NIZK} = (\mathsf{Gen}_{\mathsf{NIZK}}, \mathsf{Prove}, \mathsf{Vrfy}_{\mathsf{NIZK}})$ in the urs model with a uniform reference string of length ℓ and a hash function $\mathcal{H} : \{0,1\}^\lambda \times \mathcal{IDS} \times \mathcal{VPKS} \to \{0,1\}^\ell$, where $\mathcal{IDS}$ is the identity space and $\mathcal{VPKS}$ the public key space of the verifiers, modeled as a random oracle.

Theorem 7 (Correctness). *If* DVS *is* δ*-correct and* NIZK *is* δ'*-correct, then* $\mathsf{DVS}'[\mathsf{DVS}, \mathsf{NIZK}]$ *as given in Figure 7 is* $(\delta + \delta')$*-correct.*

Proof. Let $\mathsf{id}_D \in \mathcal{IDS}$ and $\mathsf{pp} \xleftarrow{\$} \mathsf{Setup}()$, $(\mathsf{pk}_S, \mathsf{sk}_S) \xleftarrow{\$} \mathsf{SKGen}(\mathsf{pp})$, $(\mathsf{pk}_D, \mathsf{sk}_D) \leftarrow \mathsf{VKGen}(\mathsf{pp}, \mathsf{id}_D; r)$, and $\pi \xleftarrow{\$} \mathsf{Prove}(\mathsf{urs}, (\mathsf{pp}, \mathsf{id}_D, \mathsf{pk}_D), r)$. Since NIZK is δ'-correct,

with probability $1-\delta'$ we have $\mathsf{Vrfy}_{\mathsf{NIZK}}(\mathsf{urs},(\mathsf{pp},\mathsf{id}_D,\mathsf{pk}_D),\pi)=\text{true}$ and in this case $\mathsf{Sign}'(\mathsf{pp},\mathsf{sk}_S,\mathsf{id}_D,\mathsf{pk}'_D=(\mathsf{pk}_D,\pi),m)$ returns $\sigma \xleftarrow{\$} \mathsf{Sign}(\mathsf{pp},\mathsf{sk}_S,\mathsf{id}_D,\mathsf{pk}_D,m)$. Since DVS is δ-correct, $\mathsf{Vrfy}'(\mathsf{pp},\mathsf{pk}_S,\mathsf{pk}'_D=(\mathsf{pk}_D,\pi),m,\sigma)=\mathsf{Vrfy}(\mathsf{pp},\mathsf{pk}_S,\mathsf{pk}_D,m,\sigma)=\text{true}$ with probability $1-\delta$. □

Theorem 8 ((Malicious) unforgeability). *If* DVS *is δ-correct, $(t,\varepsilon,q_S,q_{VK})$-(malicious) unforgeable with $\varepsilon<1/2$, and $(t,\varepsilon_{\mathsf{sh}},1)$-source hiding with $4\varepsilon_{\mathsf{sh}}+2\delta\le 1-\varepsilon$ and* NIZK *is $(t_{\mathsf{zk}},t_{\mathsf{Sim}_{\mathsf{zk}}},\varepsilon_{\mathsf{zk}})$-zero-knowledge, then* $\mathsf{DVS}'[\mathsf{DVS},\mathsf{NIZK}]$ *as given in Figure 7 is $(t',\varepsilon',q_S,q_{VK})$-(malicious) unforgeable against adversaries making at most $q_{\mathcal{H}}$ ROM queries with $t'\approx t$ and $\varepsilon'\le\varepsilon+q_{VK}\varepsilon_{\mathsf{zk}}+q_{\mathcal{H}}\sqrt{2\varepsilon+4\varepsilon_{\mathsf{sh}}+2\delta}$.*

Proof. Assume there is an adversary $\mathcal{A}$ that breaks the $(t',\varepsilon',q_S,q_V,q_{VK})$-(malicious) unforgeability of $\mathsf{DVS}'[\mathsf{DVS},\mathsf{NIZK}]$. Let $\mathsf{TDGen}_{\mathsf{zk}}$ and Extr be the algorithms for the zero-knowledge property of NIZK. The proof proceeds via a hybrid argument. Let G_0 be the real (malicious) unforgeability game for $\mathsf{DVS}'[\mathsf{DVS},\mathsf{NIZK}]$ and G_1 the same game with the following modification: If the adversary makes a VKeys query on id and that query is answered with (pk,π) and the adversary has made a ROM query on $(s,\mathsf{id},\mathsf{pk})$ before, the game aborts and outputs a random bit.

Lemma 3 ($\mathsf{G}_0 \rightsquigarrow \mathsf{G}_1$). $\left|\Pr[\mathsf{G}_0^{\mathcal{A}}\Rightarrow 1]-\Pr[\mathsf{G}_1^{\mathcal{A}}\Rightarrow 1]\right|\le q_{\mathcal{H}}\sqrt{2\varepsilon+4\varepsilon_{\mathsf{sh}}+2\delta}$

Proof. Consider the reduction $\mathcal{B}$ against the (malicious) unforgeability of DVS that proceeds as follows: The adversary gets $(\mathsf{pp},\mathsf{pk}_S)$ and makes VKeys query on identities, that are chosen exactly how $\mathcal{A}$ chooses the identities for its VKeys queries, to get for each query a public key pk and then sample $(\mathsf{pk}',\mathsf{sk}')\xleftarrow{\$}\mathsf{VKGen}(\mathsf{pp},\mathsf{id})$. If $\mathsf{pk}'\neq\mathsf{pk}$ for all queries, the adversary aborts. Otherwise, it computes a signature via $\sigma\xleftarrow{\$}\mathsf{Sim}(\mathsf{pp},\mathsf{pk}_S,\mathsf{sk}',\mathsf{id},m)$ for any fixed message m. The reduction then outputs $(m,\sigma,\mathsf{id}_D,\mathsf{pk})$.

We will show next that $\Pr[\mathsf{Vrfy}(\mathsf{pp},\mathsf{pk}_S,\mathsf{pk},\mathsf{id},m,\sigma)=\text{true}]\ge 1/2-2\varepsilon_{\mathsf{sh}}-\delta$. Consider for this the reduction $\mathcal{B}'$ that plays the source hiding game for DVS as follows: $\mathcal{B}'$ receives $\mathsf{pp},\mathsf{pk}_S$ (and sk_S which is ignored by $\mathcal{B}'$) and makes one VKeys query on id to obtain pk_D (and sk_D, which is again ignored) and then a Chall query on m to obtain σ. The inputs id and m are chosen exactly how $\mathcal{B}$ chooses them. $\mathcal{B}'$ outputs $b'=0$ if $\mathsf{Vrfy}(\mathsf{pp},\mathsf{pk}_S,\mathsf{pk}_D,\mathsf{id},m,\sigma)=\text{true}$ and $b'=1$ otherwise. The reduction $\mathcal{B}'$ only fails when Sign produced a signature that does not verify or if Sim produced a signature that did verify. The first event occurs with probability at most δ. Let E be the second event, i.e., the signature computed by Sim verifies. Then $1-\delta-\Pr[E]\le 1/2+2\varepsilon_{\mathsf{sh}}$. Rearranging gives the desired lower bound on $\Pr[E]$.

This shows that the reduction $\mathcal{B}$ wins the (malicious) unforgeability game with probability $\Pr[\mathsf{pk}=\mathsf{pk}']\cdot(1/2-2\varepsilon_{\mathsf{sh}}-\delta)\le\varepsilon$ which shows $\Pr[\mathsf{pk}=\mathsf{pk}']\le\varepsilon/(1/2-2\varepsilon_{\mathsf{sh}}-\delta)=2\varepsilon/(1-4\varepsilon_{\mathsf{sh}}-2\delta)\le 2\varepsilon+4\varepsilon_{\mathsf{sh}}+2\delta$. See Lemma 11 in Appendix C for an explanation of the last inequality.

Thus we have $2\varepsilon+4\varepsilon_{\mathsf{sh}}+2\delta\ge\Pr[\mathsf{pk}=\mathsf{pk}']=2^{-\mathsf{H}_2(\mathsf{pk})}$ which can be rearranged to $\mathsf{H}_2(\mathsf{pk})\ge-\log(2\varepsilon+4\varepsilon_{\mathsf{sh}}+2\delta)$. Thus we also have $\mathsf{H}_\infty(\mathsf{pk})\ge$

$\frac{-1}{2}\log(2\varepsilon + 4\varepsilon_{\mathsf{sh}} + 2\delta)$ by Fact 1, which implies that any public key pk' used by $\mathcal{A}$ in a ROM query matches a public key pk generated in a VKeys oracle query at most with probability $2^{-\mathsf{H}_\infty(\mathsf{pk})} \leq 2^{\frac{1}{2}\log(2\varepsilon+4\varepsilon_{\mathsf{sh}}+2\delta)} = \sqrt{2\varepsilon + 4\varepsilon_{\mathsf{sh}} + 2\delta}$. □

The next game, G_2, is defined like G_1, except that all VKeys(id) queries are now answered by generating the urs with known trapdoor and simulating the random oracle accordingly. In particular, by

- sampling $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{VKGen}(\mathsf{pp}, \mathsf{id})$, programming the random oracle query on $(s, \mathsf{id}, \mathsf{pk})$ with urs sampled via $(\mathsf{urs}, \mathsf{td}_{\mathsf{zk}}) \xleftarrow{\$} \mathsf{TDGen}_{\mathsf{zk}}()$ (or, as in G_1, aborting if the adversary already queried the random oracle on $(s, \mathsf{id}, \mathsf{pk})$),
- sampling $\pi \xleftarrow{\$} \mathsf{Sim}_{\mathsf{zk}}(\mathsf{urs}, \mathsf{td}_{\mathsf{zk}}, (\mathsf{pp}, \mathsf{id}_D, \mathsf{pk}_D))$, and
- returning $((\mathsf{pk}, \pi), \mathsf{sk})$.

Lemma 4 ($\mathsf{G}_1 \rightsquigarrow \mathsf{G}_2$). $\left|\Pr[\mathsf{G}_1^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}_2^{\mathcal{A}} \Rightarrow 1]\right| \leq q_{VK}\varepsilon_{\mathsf{zk}}$

Proof. We give a reduction to the zero-knowledge property of NIZK for the i-th query. The transition from G_1 to G_2 follows then from a standard hybrid argument. The reduction samples $\mathsf{pp}' = (\mathsf{pp}, s)$ and $(\mathsf{pk}_S, \mathsf{sk}_S) \xleftarrow{\$} \mathsf{SKGen}(\mathsf{pp})$ and sends them to $\mathcal{A}$. When the adversary makes its i-th VKeys query on id, the reduction samples $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{VKGen}(\mathsf{pp}, \mathsf{id})$. If the adversary has already made a random oracle query on $(s, \mathsf{id}, \mathsf{pk})$, the reduction aborts and outputs a random bit. Otherwise, the reduction uses urs from the zero-knowledge game (i.e., its original input) and programs the random oracle to return urs on $(s, \mathsf{id}, \mathsf{pk})$. The reduction queries the Prove oracle on $((\mathsf{pp}, \mathsf{id}, \mathsf{pk}), \mathsf{sk})$ to retrieve π and sends (pk, π) to the adversary. The reduction answers Sign and Vrfy queries as in G_1.

When the reduction does not abort, the reduction simulates the game G_1 if the zero-knowledge game for NIZK is played with $b = 0$ and otherwise the reduction simulates G_2. If the reduction does abort, the game G_1 and G_2 both abort, too, and thus the reduction has simulated them correctly. □

Lemma 5 (G_2). $\Pr[\mathsf{G}_2^{\mathcal{A}} \Rightarrow 1] \leq \varepsilon$

Proof. For this proof we assume without loss of generality that $\mathcal{A}$ queries the ROM on $(s, \mathsf{id}, \mathsf{pk})$ before making a sign query on Sign($\mathsf{pk}, \mathsf{id}, \cdot$).

We give a reduction to the (malicious) unforgeability of DVS. Initially, the reduction receives $(\mathsf{pp}, \mathsf{pk}_S)$, samples $s \xleftarrow{\$} \{0,1\}^\lambda$ and sends $(\mathsf{pp}' := (\mathsf{pp}, s), \mathsf{pk}_S)$ to the adversary. If the adversary makes a VKeys(id) query, the reduction forwards the query to obtain pk, aborts if the adversary already made a ROM query on $(s, \mathsf{id}, \mathsf{pk})$ and otherwise programs this query to urs sampled via $(\mathsf{urs}, \mathsf{td}_{\mathsf{zk}}) \xleftarrow{\$} \mathsf{TDGen}_{\mathsf{zk}}()$ and computes $\pi \xleftarrow{\$} \mathsf{Sim}_{\mathsf{zk}}(\mathsf{urs}, \mathsf{td}_{\mathsf{zk}}, (\mathsf{pp}, \mathsf{id}_D, \mathsf{pk}_D))$. The reduction sends the answer (pk, π) to the adversary.

The reduction perfectly simulates the game G_2 for the adversary and thus wins the (malicious) unforgeability game for DVS exactly when the adversary wins G_2. □

Combining Lemma 3 – Lemma 5 yields Theorem 8. □

Remark 4. The argument in Lemma 3 for the min-entropy in the verifier public keys makes our reduction quite loose. However, we can obtain a tight reduction by appending random bits to the verifier public keys to increase their min-entropy.

$\underline{\mathcal{A}()}$:
1 $X \xleftarrow{\$} \mathcal{D}$
2 **repeat**
3 $\quad X' \xleftarrow{\$} \mathcal{D} \mid [h(\cdot) = h(X)]$
4 **until** $g(X') = g(X)$
5 **return** X'

Fig. 8. Algorithm for the split resampling Lemma.

The simplified malicious source hiding proof makes use of the following Lemma, which is a special case of [26, Lemma 6], which in turn relies on the "Bucket Lemma" by [31].

Let $\mathcal{D}$ be a distribution on a set $\mathcal{X}$ and $g : \mathcal{X} \to \mathcal{Y}$ and $h : \mathcal{X} \to \mathcal{Z}$ functions where $\mathcal{Y}$ is a finite set and $\mathcal{Z}$ is any set. Moreover, for $X \in \mathcal{X}$ let $\mathcal{D} \mid [h(\cdot) = h(X)]$ be the distribution that samples $X' \xleftarrow{\$} \mathcal{D}$ conditioned on $h(X') = h(X)$.

Lemma 6 (Split resampling Lemma [26, Lemma 6]). *Let T^{rep} be the number of iterations of the loop in lines 2–4 of algorithm $\mathcal{A}()$ given in Figure 8. Then for all $\gamma \in (0, 1]$ we have*

$$\Pr\left[T^{\mathsf{rep}} \leq 2\ln\left(\frac{2}{\gamma}\right)\frac{|\mathcal{Y}|}{\gamma}\right] \geq 1 - \gamma.$$

Theorem 9 (Simplified malicious source hiding). *If* DVS *is $(t, \varepsilon_{\mathsf{sh}}, 1)$-all verifier key source hiding and* NIZK *is $(t_{\mathsf{td}}, \varepsilon_{\mathsf{td}}, t_{\mathsf{ks}}, \varepsilon_{\mathsf{ks}})$-knowledge sound, then* $\mathsf{DVS}'[\mathsf{DVS}, \mathsf{NIZK}]$ *as given in Figure 7 is $(t_{\mathcal{A}}, t_{\mathcal{E}}, t_{\mathcal{D}}, \varepsilon, q_{VK})$-simplified malicious source hiding with $t_{\mathcal{E}} \approx 2\ln(2/\gamma)(q_{\mathcal{H}} + 1)/\gamma \cdot t_{\mathcal{A}}$, $t_{\mathsf{td}} \approx t_{\mathsf{ks}} \approx t \approx t_{\mathcal{A}} + t_{\mathcal{D}}$, and $\varepsilon \leq \varepsilon_{\mathsf{ks}} + 2\ln(2/\gamma)(q_{\mathcal{H}} + 1)\varepsilon_{\mathsf{td}} + q_{\mathcal{H},\mathsf{aux}}/2^{\lambda}$, where $q_{\mathcal{H},\mathsf{aux}}$ is the number of random oracle queries that* aux *depends on. The parameters $\lambda \in \mathbb{N}$ and $\gamma \in (0, 1]$ can be selected arbitrarily to achieve the desired security properties.*

Proof. Let $\mathcal{A}$ be an adversary against the simplified malicious source hiding property of $\mathsf{DVS}'[\mathsf{DVS}, \mathsf{NIZK}]$. We first observe that aux does not depend on any random oracle query that starts with s, where s is part of the public parameters, except with probability $q_{\mathcal{H},\mathsf{aux}}/2^{\lambda}$, because s is a uniformly random λ bit string generated independently of aux. We assume in the following that aux does not depend on such a query.

We next describe the extractor $\mathcal{E}_{\mathcal{A}}$. The extractor is given $\mathsf{pp}' = (\mathsf{pp}, s), \mathsf{pk}^*_S$, $\mathsf{aux}, R, \mathcal{L}_V, \mathsf{pk}' = (\mathsf{pk}, \pi), \mathsf{id}, m$, where R are the random coins consumed by the adversary so far. If $\mathsf{Vrfy}_{\mathsf{NIZK}}(\mathsf{urs}, (\mathsf{pp}, \mathsf{id}, \mathsf{pk}), \pi) = \text{false}$ for $\mathsf{urs} \leftarrow \mathcal{H}(s, \mathsf{id}, \mathsf{pk})$, the extractor outputs $\perp$.

Otherwise, the extractor runs $\mathcal{A}$ several times, while changing some query answers. We assume without loss of generality that $\mathcal{A}$ queries the RO for each value at most once and makes a query on $(s, \widetilde{\mathsf{id}}, \widetilde{\mathsf{pk}})$ before the $\textsc{Chall}(\widetilde{\mathsf{pk}}' = (\widetilde{\mathsf{pk}}, \widetilde{\pi}), \widetilde{\mathsf{id}}, \widetilde{m})$ query. We also assume without loss of generality that the number of random coins consumed and the number of random oracle queries by $\mathcal{A}$ until a $\textsc{Chall}$ query are fixed values.

The extractor runs $\mathcal{A}$ using the random coins R and answers all queries as described before, except when $\mathcal{A}$ makes the random oracle query on $(s, \mathsf{id}, \mathsf{pk})$. This ROM query is answered by sampling $(\mathsf{urs}, \mathsf{td}_{\mathsf{ks}}) \xleftarrow{\$} \mathsf{TDGen}_{\mathsf{ks}}()$ and returning urs. We call this the programmed ROM query. After that query, all further ROM queries are answered with fresh uniformly random bitstrings.

Note that the run of $\mathcal{A}$ until the programmed ROM query is exactly as the run in the real world, but after the programmed ROM query it can differ.

If this results in a run where $\mathcal{A}$ makes the Chall query on arguments $\widetilde{\mathsf{pk}}' = (\mathsf{pk}, \widetilde{\pi}), \mathsf{id}, \widetilde{m}$ with $\mathsf{Vrfy}_{\mathsf{NIZK}}(\mathsf{urs}, (\mathsf{pp}, \mathsf{id}, \mathsf{pk}), \widetilde{\pi}) = \text{true}$, we call the run successful and the extractor computes

- $r \xleftarrow{\$} \mathsf{Extr}(\mathsf{urs}, \mathsf{td}_{\mathsf{ks}}, (\mathsf{pp}, \mathsf{id}, \mathsf{pk}), \widetilde{\pi})$,
- $(\widehat{\mathsf{pk}}, \widehat{\mathsf{sk}}) \leftarrow \mathsf{VKGen}(\mathsf{pp}, \mathsf{id}; r)$,
- $\sigma \xleftarrow{\$} \mathsf{Sim}(\mathsf{pp}, \mathsf{pk}_S, \mathsf{id}, \widehat{\mathsf{sk}}, m)$,

and returns σ. Otherwise, the adversary runs $\mathcal{A}$ again, up to a maximum of $\kappa := 2\ln(2/\gamma)(q_{\mathcal{H}}+1)/\gamma$ runs. Here, $\gamma \in (0, 1]$ is a parameter we can choose for a trade-off between runtime and advantage of the reduction. If none of these runs leads to the above case, the extractor outputs $\perp$.

We analyze next the probability that the adversary does not receive a valid proof for pk in all κ runs. For this analysis we consider an extractor $\mathcal{E}'_{\mathcal{A}}$ that uses in the programmed ROM query a uniformly random sampled urs (instead of urs being sampled via $(\mathsf{urs}, \mathsf{td}_{\mathsf{ks}}) \xleftarrow{\$} \mathsf{TDGen}_{\mathsf{ks}}()$). The knowledge soundness of NIZK guarantees that the probability of a run being successful can change at most with probability $\varepsilon_{\mathsf{td}}$.

Next, we describe another, hypothetical extractor $\mathcal{E}''_{\mathcal{A}}$ that also runs $\mathcal{A}$ several times, but simulates these runs less efficiently. Then, we use Lemma 6 to bound the number of runs the hypothetical extractor needs. Finally, we show that the extractor $\mathcal{E}'_{\mathcal{A}}$ needs at most as many runs as the hypothetical extractor $\mathcal{E}''_{\mathcal{A}}$.

The hypothetical extractor $\mathcal{E}''_{\mathcal{A}}$ works exactly like the extractor $\mathcal{E}'_{\mathcal{A}}$, except that the modified random oracle query is not always the query on $(s, \mathsf{id}, \mathsf{pk})$, but always the query on $(s, \widetilde{\mathsf{id}}, \widetilde{\mathsf{pk}})$ where $\widetilde{\mathsf{pk}}' = (\widetilde{\mathsf{pk}}, \widetilde{\pi}), \widetilde{\mathsf{id}}, \widetilde{m}$ are the arguments of the Chall query of $\mathcal{A}$ in the simulated run. Of course, it might not be known at the time when $\mathcal{A}$ makes the ROM query on $(s, \widetilde{\mathsf{id}}, \widetilde{\mathsf{pk}})$ that this is the ROM query used for the Chall query, but the hypothetical extractor can guess the correct query and run $\mathcal{A}$ again when that guess was incorrect.

Now, let us use Lemma 6 to bound the number of runs the hypothetical extractor needs. Here, we count as runs only those where the hypothetical extractor programmed the correct random oracle query. Therefore, we set $\mathcal{D}$ to be the distribution that outputs a transcript of a single run of $\mathcal{A}$. That transcript includes the random coins used by $\mathcal{A}$ and all oracle calls with their respective inputs and outputs and the output of $\mathcal{A}$. Let X be the transcript of the real-world run of the adversary. The function h maps such a transcript to a truncated transcript that includes all of the randomness and all queries that happen before the ROM query on $(s, \widetilde{\mathsf{id}}, \widetilde{\mathsf{pk}})$, where $\widetilde{\mathsf{pk}}' = (\widetilde{\mathsf{pk}}, \widetilde{\pi}), \widetilde{\mathsf{id}}, \widetilde{m}$ are the arguments of the Chall query, (excluding this ROM query). Now in each run

of $\mathcal{A}$ by the hypothetical extractor, the resulting transcript X' is distributed like a fresh transcript conditioned on $h(X) = h(X')$. The function g maps a such a transcript to the index (among all ROM queries) of the ROM query on $(s, \widetilde{\mathsf{id}}, \widetilde{\mathsf{pk}})$, where $\widetilde{\mathsf{pk}}' = (\widetilde{\mathsf{pk}}, \widetilde{\pi}), \widetilde{\mathsf{id}}, \widetilde{m}$ are the arguments of the CHALL query if $\mathsf{Vrfy}_{\mathsf{NIZK}}(\mathsf{urs}, (\mathsf{pp}, \widetilde{\mathsf{id}}, \widetilde{\mathsf{pk}}), \widetilde{\pi}) = \text{true}$. Otherwise, g maps the transcript to a special failure symbol. Note that if a run produces a transcript X' with $g(X) = g(X')$, the extractor successfully obtains a valid proof under a urs where it knows the extraction trapdoor. Now Lemma 6 tells us that with probability at least $1 - \gamma$, the hypothetical extractor obtains such a successful run among the first $\kappa = 2\ln(2/\gamma)(q_{\mathcal{H}} + 1)/\gamma$ runs.

Next, we show that the probability of being successful for a run of the adversary performed by the extractor $\mathcal{E}'_{\mathcal{A}}$ is at least as high as for runs performed by the hypothetical extractor $\mathcal{E}''_{\mathcal{A}}$. Let X' be a transcript of a run of $\mathcal{A}$ by $\mathcal{E}''_{\mathcal{A}}$ with $g(X) = g(X')$. Then, the programmed ROM query has the same index as in the real run X. Since all queries before the programmed ROM query are identical and the same randomness is used, the programmed ROM query must be on $(s, \mathsf{id}, \mathsf{pk})$. Since this is also the ROM query that $\mathcal{E}'_{\mathcal{A}}$ programs, the probability that a run by $\mathcal{E}'_{\mathcal{A}}$ results in the transcript X' is at least as high as for a run by $\mathcal{E}''_{\mathcal{A}}$.

If one of the runs of $\mathcal{A}$ by the real extractor $\mathcal{E}_{\mathcal{A}}$ is successful, the extractor will be able to extract with probability at least $1 - \varepsilon_{\mathsf{ks}}$ a random string r such that $(\widehat{\mathsf{pk}}, \widehat{\mathsf{sk}}) \leftarrow \mathsf{VKGen}(\mathsf{pp}, \mathsf{id}; r)$ satisfies $\widehat{\mathsf{pk}} = \mathsf{pk}$, because otherwise we can use the extractor to win the knowledge soundness game of NIZK. In particular, $((\mathsf{id}, \mathsf{pk}), \widehat{\mathsf{sk}}) \in \mathcal{R}_V$, where $\mathcal{R}_V$ is the relation for the all verifier key source hiding game.

The all verifier key source hiding notion guarantees now that the adversary $\mathcal{A}$ and the distinguisher $\mathcal{D}$ cannot distinguish between a signature generated via $\mathsf{Sign}(\mathsf{pp}, \mathsf{sk}_S, \mathsf{pk}, \mathsf{id}, m)$ and a signature generated via $\mathsf{Sim}(\mathsf{pp}, \mathsf{pk}_S, \mathsf{id}, \widehat{\mathsf{sk}}, m)$ (as produced by $\mathcal{E}_{\mathcal{A}}$). □

5 Impossibility results

In this section we show that it is impossible to achieve simplified malicious source hiding in the standard model and that the EKDH assumption is broken. We apply a similar technique to disprove deniability of X3DH in Appendix B. For these results we give an efficiently sampleable distribution AuxPrep of auxiliary inputs such that security does not hold for a randomly sampled auxiliary input. This trivially rules out security in our stronger model with all-quantified auxiliary inputs by taking the auxiliary input from the image of AuxPrep that maximizes the advantage of the adversary.

5.1 Impossibility of simplified malicious source hiding DVS

We show that a DVS can not be simultaneously unforgeable and simplified malicious source hiding in the common auxiliary-input model, where the adversary

$C_K(\mathsf{pp}, \mathsf{id} \in \mathcal{IDS})$:

1 $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{VKGen}(\mathsf{pp}, \mathsf{id}; \mathsf{Eval}(K, \mathsf{id}))$
//where K is a PPRF key
2 **return** pk

$\mathsf{AuxPrep}()$:

3 $K \xleftarrow{\$} \mathsf{KeyGen}()$
4 **return** $\mathsf{iO}(C_K)$

$\mathcal{A}^{\textsc{Chall},\textsc{Random},\textsc{VKeys}}(\mathsf{pp}, \mathsf{pk}_S^*, \mathsf{aux} = C')$:

5 $m \xleftarrow{\$} \mathcal{M}$
6 $\mathsf{pk} \leftarrow C'(\mathsf{pp}, \mathsf{id}^\star)$
7 $\sigma \xleftarrow{\$} \textsc{Chall}(\mathsf{pk}, \mathsf{id}^\star, m)$
8 **return** $\mathsf{st} := (\mathsf{pp}, \mathsf{pk}_S^*, \mathsf{pk}, m, \sigma)$

$\mathcal{D}(\mathsf{st} = (\mathsf{pp}, \mathsf{pk}_S^*, \mathsf{pk}, m, \sigma), \mathsf{sk}_S^*)$:

9 **if** $\mathsf{Vrfy}(\mathsf{pp}, \mathsf{pk}_S^*, \mathsf{pk}, m, \sigma)$
10 **return** 0
11 **else**
12 **return** 1

Fig. 9. On the left-hand side, the circuit C_K and $\mathsf{AuxPrep}$, which obfuscates this circuit. On the right-hand side, the adversary $(\mathcal{A}, \mathcal{D})$ that breaks simplified malicious source hiding using auxiliary input. The identity $\mathsf{id}^\star \in \mathcal{IDS}$ is a fixed identity.

and the extractor get access to the same common auxiliary input. The result assumes the existence of an indistinguishability obfuscator and a puncturable PRF and is on a high level similar to [5].

However, we cannot use the result of [5] in a black-box way, because it is unclear if simplified malicious source hiding DVS imply extractable one-way function. This is in contrast to other "extractable primitives" like SNARKs and, e.g., the knowledge of exponent assumption, which all imply extractable one-way functions [5].

The result does not extend to privately verifiable DVS, because it crucially relies on being able to verify a signature without a secret key.

Theorem 10. *Assume there exist a* $(t_{\mathsf{io}}, \varepsilon_{\mathsf{io}})$*-secure and* δ*-correct iO scheme* iO *and a* $(t_{\mathsf{fp}}, \varepsilon_{\mathsf{fp}})$*-functionality preserving and* $(t_{\mathsf{pr}}, \varepsilon_{\mathsf{pr}})$*-pseudorandom puncturable PRF, then there exists no DVS (in the standard model) that is simultaneously* $(t_{\mathsf{unf}}, \varepsilon_{\mathsf{unf}}, 0, 0)$*-unforgeable and* $(t_{\mathsf{msh}}, t_{\mathcal{E}}, t_{\mathcal{D}}, \varepsilon_{\mathsf{msh}}, 0)$*-simplified malicious source hiding with* $\varepsilon_{\mathsf{unf}} + \varepsilon_{\mathsf{msh}} + \varepsilon_{\mathsf{io}} + \varepsilon_{\mathsf{pr}} + \delta + \varepsilon_{\mathsf{fp}} < 1$ *when* t_{msh} *and* $t_{\mathcal{D}}$ *are large enough for the simple adversary in Figure 9 and* $t_{\mathsf{unf}}, t_{\mathsf{io}}, t_{\mathsf{fp}}, t_{\mathsf{pr}} \approx t_{\mathcal{E}}$*.*

Proof. Assume a DVS that is simplified malicious source hiding in the common auxiliary input model. We show how to break unforgeability for such a DVS. Let $\mathsf{id}^\star \in \mathcal{IDS}$ be a fixed identity throughout this proof.

Consider the adversary $\mathcal{A}$ given in Figure 9 and let $\mathcal{E}_{\mathcal{A}}$ be an extractor for this adversary such that $\mathsf{Adv}^{\mathsf{s\text{-}mal\text{-}sh}}_{\mathsf{DVS}}(\mathcal{A}, \mathcal{D})$ is negligible. We now use the extractor to construct an adversary $\mathcal{B}$ that wins the unforgeability game with overwhelming probability. The adversary $\mathcal{B}$ is given in Figure 10. Since the adversary $\mathcal{B}$ uses the extractor on differently distributed inputs compared to how the extractor is called in the simplified malicious source hiding game, we define a sequence of games to argue that the extractor on inputs generated by $\mathcal{B}$ is almost as successful as in the simplified malicious source hiding game.

$C_{\mathsf{id}^\star,\mathsf{pk}^\star,K_{\mathsf{id}^\star}}(\mathsf{pp},\mathsf{id}\in\mathcal{IDS})$:

1 **if** $\mathsf{id}=\mathsf{id}^\star$
2 **return** $\mathsf{pk}^\star$
3 **else**
4 $(\mathsf{pk},\mathsf{sk})\xleftarrow{\$}\mathsf{VKGen}(\mathsf{pp},\mathsf{id};\mathsf{Eval}(K_{\mathsf{id}^\star},\mathsf{id}))$
5 **return** pk

$\mathcal{B}^{\textsc{Sign},\textsc{VKeys}}(\mathsf{pp},\mathsf{pk}_S)$:

1 $\mathsf{aux}\xleftarrow{\$}\mathsf{AuxPrep}()$
2 $\mathsf{pk}^\star\xleftarrow{\$}\textsc{VKeys}(\mathsf{id}^\star)$
3 $K\xleftarrow{\$}\mathsf{KeyGen}()$ //PPRF
4 $K_{\mathsf{id}^\star}\xleftarrow{\$}\mathsf{Puncture}(K,\mathsf{id}^\star)$
5 $C'\xleftarrow{\$}\mathsf{iO}(C_{\mathsf{id}^\star,\mathsf{pk}^\star,K_{\mathsf{id}^\star}})$
6 $m^\star\xleftarrow{\$}\mathcal{M}$
7 $\sigma^\star\xleftarrow{\$}\mathcal{E}_{\mathcal{A}}(\mathsf{pp},\mathsf{pk}_S,\mathsf{aux},\emptyset,\mathsf{id}^\star,m^\star,R)$
8 **return** $(m^\star,\sigma^\star,\mathsf{id}^\star)$

Fig. 10. On the left-hand side, the circuit $C_{\mathsf{id}^\star,\mathsf{pk}^\star,K_{\mathsf{id}^\star}}$, which gets obfuscated and then used as auxiliary input from hybrid G_1 onwards. On the right-hand side, the adversary $\mathcal{B}$ that breaks unforgeability. The identity $\mathsf{id}^\star\in\mathcal{IDS}$ is a fixed identity and R is the randomness used to sample $m^\star$.

To argue that the adversary $\mathcal{B}$ wins the unforgeability game with overwhelming probability, we need to argue that $\mathcal{E}_{\mathcal{A}}$ produces a valid signature with overwhelming probability. For this, we introduce two hybrid games.

Let $\mathsf{AuxPrep}$ be the auxiliary input sampler described in Figure 9. The first hybrid game G_0 invokes the extractor exactly as it is invoked in the simplified malicious source hiding game with adversary $\mathcal{A}$ and a (random) auxiliary input $\mathsf{aux}\xleftarrow{\$}\mathsf{AuxPrep}()$. The game outputs 1 iff the signature of the extractor verifies.

Lemma 7 (G_0). $\Pr[\mathsf{G}_0\Rightarrow 1]=1-\mathbb{E}_{\mathsf{aux}\xleftarrow{\$}\mathsf{AuxPrep}()}\mathsf{Adv}^{\mathsf{s\text{-}mal\text{-}sh}}_{\mathsf{DVS},\mathsf{aux}}(\mathcal{A},\mathcal{D})-\delta\geq 1-\varepsilon_{\mathsf{msh}}-\delta$

Proof. Assume that evaluation of the obfuscated circuit gives the correct result. This happens with probability $1-\delta$.

Furthermore, when the game $\mathcal{G}^{\mathsf{s\text{-}mal\text{-}sh}}_{\mathsf{DVS},\mathsf{aux}}(\mathcal{A},\mathcal{D})$ is played with $b=0$ (where the adversary gets an honest signature), the distinguisher always outputs 0 and thus wins the game due to the correctness of DVS. However, if the game is played with $b=1$, the adversary can win at most with probability $\mathsf{Adv}^{\mathsf{s\text{-}mal\text{-}sh}}_{\mathsf{DVS}}(\mathcal{A},\mathcal{D})$, since the advantage of $\mathcal{A}$ is $\mathsf{Adv}^{\mathsf{s\text{-}mal\text{-}sh}}_{\mathsf{DVS}}(\mathcal{A},\mathcal{D})$. Hence, in the case $b=1$ the adversary has to output 0 with probability at least $1-\mathsf{Adv}^{\mathsf{s\text{-}mal\text{-}sh}}_{\mathsf{DVS}}(\mathcal{A},\mathcal{D})$ and in this case the signature of the extractor did verify. □

In the next hybrid game G_1, the auxiliary input is changed to an obfuscation of the circuit $C_{\mathsf{id}^\star,\mathsf{pk}^\star,K_{\mathsf{id}^\star}}$ given in Figure 10 with the following constants:

- $K_{\mathsf{id}^\star}\xleftarrow{\$}\mathsf{Puncture}(K,\mathsf{id}^\star)$ where $K\xleftarrow{\$}\mathsf{KeyGen}()$
- $\mathsf{pk}^\star$ where $(\mathsf{pk}^\star,\mathsf{sk}^\star)\leftarrow\mathsf{VKGen}(\mathsf{pp},\mathsf{id}^\star;\mathsf{Eval}(K,\mathsf{id}^\star))$

Lemma 8 ($\mathsf{G}_0\rightsquigarrow\mathsf{G}_1$). $|\Pr[\mathsf{G}_0\Rightarrow 1]-\Pr[\mathsf{G}_1\Rightarrow 1]|\leq\varepsilon_{\mathsf{io}}+\varepsilon_{\mathsf{fp}}$.

Proof. The obfuscated circuits in in G_0 and G_1 are functionally equivalent, because:

- The keys for all users except $\mathsf{id}^\star$ are generated in the same way, except that the punctured key $K_{\mathsf{id}^\star}$ is used instead of K. Since the PPRF is not evaluated on $\mathsf{id}^\star$ here, this is functionally equivalent (except with probability $\varepsilon_{\mathsf{fp}}$) because the PPRF is $(t_{\mathsf{fp}},\varepsilon_{\mathsf{fp}})$-functionality preserving.

– The key for user $\mathsf{id}^\star$ is computed in the same way as before, but the result is hard-coded in the obfuscated program.

With this, the reduction to iO is straightforward. □

In the next hybrid G_2, we compute $\mathsf{pk}^\star$ as $(\mathsf{pk}^\star, \mathsf{sk}^\star) \xleftarrow{\$} \mathsf{VKGen}(\mathsf{pp}, \mathsf{id}^\star)$ (with true randomness) instead of as $(\mathsf{pk}^\star, \mathsf{sk}^\star) \leftarrow \mathsf{VKGen}(\mathsf{pp}, \mathsf{id}^\star; \mathsf{Eval}(K, \mathsf{id}^\star))$.

Lemma 9 ($\mathsf{G}_1 \rightsquigarrow \mathsf{G}_2$). *There exists a PPT adversary $\mathcal{D}$ with*

$$|\Pr[\mathsf{G}_1 \Rightarrow 1] - \Pr[\mathsf{G}_2 \Rightarrow 1]| \leq \mathsf{Adv}^{\mathsf{pprf}}_{\mathsf{PPRF}}(\mathcal{D}) \leq \varepsilon_{\mathsf{pr}}.$$

Proof. This is a straightforward reduction to the security of PPRF. This security game gives us the punctured key $K_{\mathsf{id}^\star}$, which gets hard-coded in the obfuscated program. The public key $\mathsf{pk}^\star$ is generated with the challenge value obtained from the PPRF security game, which is either $\mathsf{Eval}(K, \mathsf{id}^\star)$ (corresponding to game G_1) or uniformly random (corresponding to game G_2) □

Lemma 10 (G_2). $\Pr[\mathsf{G}_2 \Rightarrow 1] = \mathsf{Adv}^{\mathsf{uf}}_{\mathsf{DVS}}(\mathcal{B}) \leq \varepsilon_{\mathsf{unf}}$

Proof. The adversary $\mathcal{B}$ and the game G_2 invoke the extractor $\mathcal{E}_{\mathcal{A}}$ on identically distributed values. In particular, in both invocations pp, pk_S and $\mathsf{pk}^\star$ (hard-coded in the obfuscated program) are honestly generated. The game G_2 outputs 1 if the signature returned by the obfuscated program verifies, which is exactly when $\mathcal{B}$ wins the unforgeability game. □

Combining Lemmata 7 – 10 yields $\varepsilon_{\mathsf{unf}} \geq 1 - \varepsilon_{\mathsf{msh}} - \varepsilon_{\mathsf{io}} - \varepsilon_{\mathsf{pr}} - \delta - \varepsilon_{\mathsf{fp}}$. □

Remark 5. In our impossibility result, the auxiliary input depends on the specific scheme (concretely, the VKGen algorithm). This dependency on the specific scheme can be easily avoided by obfuscating a universal circuit of sufficient size instead.

Remark 6. This result shows that a publicly verifiable DVS cannot be unforgeable and simplified malicious source hiding in the standard model. Since ring signatures imply publicly verifiable DVS, our impossibility result also rules out unforgeable and simplified malicious source hiding ring signatures in the standard model.

5.2 Disproving the (E)KDH assumption

In this section we show that the (E)KDH assumption with arbitrary auxiliary input does not hold under very mild conditions on the group, that are satisfied by all groups interesting for cryptography (e.g. elliptic curves). The result also implies that the KDH assumption (which is stronger, since the adversary gets more information) is not true.

In the following, let $\mathcal{G} = (\mathbb{G}, g, q)$ be a description of a group $\mathbb{G}$ of prime-order q with generator g and function $\mathsf{DH} : \mathcal{G} \times \mathcal{G} \to \mathcal{G}$ with $(g^a, g^b) \mapsto g^{ab}$.

Definition 20 (Hinted CDH problem). *Let $\mathcal{G}$ be a cyclic group with generator g. We say that the (t, ε, η)-hinted computational Diffie-Hellman (CDH) assumption holds over group $\mathcal{G}$ if for every adversary $\mathcal{A}$ running in time at most t it holds that* $\Pr_{a,b \xleftarrow{\$} \mathbb{Z}_q}[\mathcal{A}(\mathcal{G}, g^a, g^b, \lfloor b/\eta \rfloor) = \mathsf{DH}(g^a, g^b)] \leq \varepsilon$.

The hinted CDH problem with $\eta = 1$ is trivially efficiently solvable in any group where the group operation is efficiently computable. On the other hand, for $\eta = q$ the problem is equivalent to the regular CDH problem, which is hard in groups used in cryptography. Informally speaking, our attack requires that the hardness of the hinted CDH problem increases continuously with increasing η, which is a very mild assumption as we explain below.

Definition 21 (KDH and EKDH Assumptions [56]). *Let $\mathcal{G}$ be a cyclic group with generator g. Let $\mathcal{A}$ be any algorithm running in time $t_{\mathcal{A}}$ which runs on input $\mathcal{G}$, $U \in \mathcal{G}$, and $\mathsf{aux} \in \{0,1\}^*$ and outputs $Z \in \mathcal{G}$. We denote with $Z \leftarrow \mathcal{A}(\mathcal{G}, U, \mathsf{aux}; r)$ the output of running $\mathcal{A}$ on input $\mathcal{G}, U, \mathsf{aux}$ with random coins r.*

We say that the $(t_{\mathcal{A}}, t_{\mathcal{E}}, t_{\mathcal{D}}, \varepsilon_{\mathcal{D}})$-Knowledge of DH (KDH) assumption holds over group $\mathcal{G}$ iff for every $\mathcal{A}$, there exists an extractor $\mathcal{E}_{\mathcal{A}}$ such that for all auxiliary inputs $\mathsf{aux} \in \{0,1\}^*$ and uniformly random $U \xleftarrow{\$} \mathcal{G}$

- *$\mathcal{E}_{\mathcal{A}}$ runs on input $\mathcal{G}, U, \mathsf{aux}$, and r in time $t_{\mathcal{E}}$ and outputs $\hat{Z} \in \mathcal{G}$ or $\perp$*
- *If $\mathcal{E}_{\mathcal{A}}(\mathcal{G}, U, \mathsf{aux}, r) \neq \perp$ then $\hat{Z} = \mathsf{DH}(U, Z)$*
- *For every algorithm C running in time $t_{\mathcal{D}}$, we have that $\Pr[C(\mathcal{G}, U, r, \mathsf{aux}) = \mathsf{DH}(U, Z) \mid \mathcal{E}_{\mathcal{A}}(\mathcal{G}, U, \mathsf{aux}, r) = \perp] \leq \varepsilon_{\mathcal{D}}$, where $Z \leftarrow \mathcal{A}(\mathcal{G}, \mathsf{aux}; r)$ and the probability is taken over the coins of C and the randomness used to sample r.*

We say that the $(t_{\mathcal{A}}, t_{\mathcal{E}}, t_{\mathcal{D}}, \varepsilon_{\mathcal{D}})$-Extended Knowledge of DH (EKDH) assumption holds over group $\mathcal{G}$ if the same holds for an algorithm $\mathcal{A}$ that does not receive U as input.

The original definition of [56] is ambiguous in two aspects, but our attack breaks this assumption in all meaningful interpretations. First, they define security with respect to a class of auxiliary inputs AUX, but never specify for which classes AUX they conjecture this assumption to hold. For simplicity we have stated the definition for $\mathsf{AUX} = \{0,1\}^*$, but in our attack the auxiliary input is extremely simple. It can be described by less than $2\lceil \log q \rceil$ bits, where q is the order of $\mathcal{G}$.

Secondly, it is not clear in their definition what aux is and what is allowed to depend on aux, apart from the condition $\mathsf{aux} \in \mathsf{AUX}$. We interpret this as security has to hold for all $\mathsf{aux} \in \mathsf{AUX}$ and that the extractor (and thus also the adversary) is not allowed to depend on aux. If instead the extractor is allowed to depend on aux, then the auxiliary input would be meaningless because it can be replaced by a string hard-coded in the description of $\mathcal{A}$. The version where aux is sampled uniformly at random from AUX and AUX is an arbitrary set is equivalent to the version where aux is all-quantified, since AUX can be the singleton set that just contains the value aux that maximizes the success probability for the adversary. Finally, the version where aux is sampled uniformly at random from a fixed set AUX is a weaker assumption, but our attack can still break it for a very simple set AUX (in particular from a set we can very efficiently sample from).

The work [56] justifies the KDH assumption by reducing it to the knowledge of exponent assumption (KEA) and another novel assumption called knowledge of discrete logarithm in the common auxiliary input model. That result is

meaningless, because the KEA assumption does not hold in the common auxiliary input model [5]. Curiously, the authors of [56] explicitly mention [5] in their work and claim to bypass their impossibility result with the (E)KDH assumption.

Our proof uses the following general lower bound for a conditional probability.

Fact 11. Let A and C be events with $\Pr[C] > 0$. Then $\Pr[A \mid C] \geq 1 - \frac{1-\Pr[A]}{\Pr[C]}$.

Proof. $\Pr[A \mid C] = \frac{\Pr[A \wedge C]}{\Pr[C]} = \frac{\Pr[A]-\Pr[A \wedge \neg C]}{\Pr[C]} \geq \frac{\Pr[A]-\Pr[\neg C]}{\Pr[C]} = \frac{\Pr[A]-1+\Pr[C]}{\Pr[C]} = 1 - \frac{1-\Pr[A]}{\Pr[C]}$ □

In the following attack we use the auxiliary input to give a group element and a hint about the corresponding exponent to the adversary, who outputs this group element as its public key. Now, both the extractor and the distinguisher (the algorithm C in case the extractor fails) need to solve the same hinted CDH challenge, but with different runtime and advantage.

Theorem 12. *If $\mathcal{G} = (\mathbb{G}, g, q)$ is a group where the (t, ε, η)-hinted CDH assumption holds for $\varepsilon < 1$, but the (t', ε', η)-hinted CDH assumption (for some $t < t'$) does not hold, then the $(t_{\mathcal{A}}, t, t', 1 - (1-\varepsilon')/(1-\varepsilon))$-EKDH assumption does not hold in $\mathcal{G}$.*

Proof. Consider the variant where the auxiliary input is generated by sampling $b \xleftarrow{\$} \mathbb{Z}_q$ and returning $\mathsf{aux} = (g^b, \lfloor b/\eta \rfloor)$; and the adversary $\mathcal{A}$ playing against the EKDH assumption on input $(\mathcal{G}, \mathsf{aux} = (Z = g^b, b' = \lfloor b/\eta \rfloor))$ outputs Z.

The extractor $\mathcal{E}_{\mathcal{A}}$ for this adversary then has to solve the hinted CDH problem on U and Z with parameter η and has time t. Since we assume the (t, ε, η)-hinted CDH assumption, the extractor can output $\mathsf{DH}(U, Z)$ with probability at most ε.

On the other hand, since we assume (t', ε', η)-hinted CDH to not hold, there exists an algorithm $\mathcal{A}_{\mathsf{HCDH}}$ with $\Pr[\mathcal{A}_{\mathsf{HCDH}}(\mathcal{G}, U, Z, b') = \mathsf{DH}(U, Z)] > \varepsilon'$. Let $C(\mathcal{G}, U, r, \mathsf{aux} = (Z, b'))$ be the circuit that ignores r and runs $\mathcal{A}_{\mathsf{HCDH}}(\mathcal{G}, U, Z, b')$.

Then, we get by applying Fact 11

$$\begin{aligned}\Pr[C(\mathcal{G}, U, r, \mathsf{aux} = (Z, b')) = \mathsf{DH}(U, Z) \mid \mathcal{E}_{\mathcal{A}}(\mathcal{G}, U, \mathsf{aux}, r) = \bot] \\ \geq 1 - \frac{1 - \Pr[C(\mathcal{G}, U, r, \mathsf{aux}) = \mathsf{DH}(U, Z)]}{\Pr[\mathcal{E}_{\mathcal{A}}(\mathcal{G}, U, \mathsf{aux}, r) = \bot]} \geq 1 - \frac{1-\varepsilon'}{1-\varepsilon},\end{aligned}$$

where the probability is taken over sampling U, r, aux and the random coins of C and $\mathcal{E}_{\mathcal{A}}$.

This implies that there exists an auxiliary input aux such that

$$\Pr[C(\mathcal{G}, U, r, \mathsf{aux} = (Z, b')) = \mathsf{DH}(U, Z) \mid \mathcal{E}_{\mathcal{A}}(\mathcal{G}, U, \mathsf{aux}, r) = \bot] \geq 1 - \frac{1-\varepsilon'}{1-\varepsilon},$$

where the probability is taken over sampling U, r and the random coins of C and $\mathcal{E}_{\mathcal{A}}$. □

Note that the result of Theorem 12 is meaningful if $\varepsilon < \varepsilon'$. We next argue that in groups typically used in cryptography we find for arbitrary t and t' with

$t < t'$ a suitable hint size η such that the (t, ε, η)-hinted CDH assumption holds, but the (t', ε', η)-hinted CDH assumption does not hold.

For concreteness, consider generic groups (or any group where the best known algorithm to solve the hinted CDH assumption is generic, such as elliptic curves). In these groups it is easy to see that the best algorithm to solve the hinted CDH assumption is to compute the discrete log of the second group element by searching through the space of possible exponents with the baby-step-giant-step algorithm [47, 49], which takes $c\sqrt{q/h}$ steps for some constant c. By setting $t' = c\sqrt{h}$, we ensure that the (t', ε', η)-hinted CDH assumption does not hold for any $\varepsilon' < 1$, but the (t, ε, η)-hinted CDH assumption holds for any $t < t'$ for a suitable $\varepsilon < 1$ which leads to an attack on the EKDH assumption with advantage $\varepsilon_{\mathcal{D}} = 1$.

This analysis also works for groups where faster algorithms than generic ones are available, as long as we can find a hint size η such that time t is just enough to solve the hinted CDH problem with probability 1. In particular, the argument fails if the time t is enough to break CDH without any hint, but this setting is not interesting for cryptography anyway.

Acknowledgments. We thank Nils Fleischhacker and Guilherme Rito for discussions on an early version of this work. R.F. was supported by the German Federal Ministry of Education and Research and the Hessian Ministry of Higher Education, Research, Science and the Arts within their joint support of the National Research Center for Applied Cybersecurity ATHENE.

References

[1] Joël Alwen, Bruno Blanchet, Eduard Hauck, Eike Kiltz, Benjamin Lipp, and Doreen Riepel. "Analysing the HPKE Standard". In: *EUROCRYPT 2021, Part I*. Ed. by Anne Canteaut and François-Xavier Standaert. Vol. 12696. LNCS. Springer, Cham, Oct. 2021, pp. 87–116. DOI: 10.1007/978-3-030-77870-5_4.

[2] Maryam Rajabzadeh Asaar, Mahmoud Salmasizadeh, and Mohammad Reza Aref. *Code-based Strong Designated Verifier Signatures: Security Analysis and a New Construction*. Cryptology ePrint Archive, Report 2016/779. 2016. URL: https://eprint.iacr.org/2016/779.

[3] Hafsa Assidi and El Mamoun Souidi. "Strong Designated Verifier Signature Based on the Rank Metric". In: *Information Security Theory and Practice - 13th IFIP WG 11.2 International Conference, WISTP 2019, Paris, France, December 11-12, 2019, Proceedings*. Ed. by Maryline Laurent and Thanassis Giannetsos. Vol. 12024. Lecture Notes in Computer Science. Springer, 2019, pp. 85–102. DOI: 10.1007/978-3-030-41702-4_6.

[4] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. "On the (Im)possibility of Obfuscating Programs". In: *CRYPTO 2001*. Ed. by Joe Kilian. Vol. 2139. LNCS. Springer, Berlin, Heidelberg, Aug. 2001, pp. 1–18. DOI: 10.1007/3-540-44647-8_1.

[5] Nir Bitansky, Ran Canetti, Omer Paneth, and Alon Rosen. "On the existence of extractable one-way functions". In: *46th ACM STOC*. Ed. by David B. Shmoys. ACM Press, 2014, pp. 505–514. DOI: 10.1145/2591796.2591859.

[6] Nikita Borisov, Ian Goldberg, and Eric A. 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 2004, Washington, DC, USA, October 28, 2004*. Ed. by Vijay Atluri, Paul F. Syverson, and Sabrina De Capitani di Vimercati. ACM, 2004, pp. 77–84. DOI: 10.1145/1029179.1029200.

[7] Pedro Branco, Nico Döttling, and Jesko Dujmovic. "Rate-1 Incompressible Encryption from Standard Assumptions". In: *TCC 2022, Part II*. Ed. by Eike Kiltz and Vinod Vaikuntanathan. Vol. 13748. LNCS. Springer, Cham, Nov. 2022, pp. 33–69. DOI: 10.1007/978-3-031-22365-5_2.

[8] Jacqueline Brendel, Rune Fiedler, Felix Günther, Christian Janson, and Douglas Stebila. "Post-quantum Asynchronous Deniable Key Exchange and the Signal Handshake". In: *PKC 2022, Part II*. Ed. by Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe. Vol. 13178. LNCS. Springer, Cham, Mar. 2022, pp. 3–34. DOI: 10.1007/978-3-030-97131-1_1.

[9] Jacqueline Brendel, Marc Fischlin, Felix Günther, Christian Janson, and Douglas Stebila. "Towards Post-Quantum Security for Signal's X3DH Handshake". In: *SAC 2020*. Ed. by Orr Dunkelman, Michael J. Jacobson Jr., and Colin O'Flynn. Vol. 12804. LNCS. Springer, Cham, Oct. 2020, pp. 404–430. DOI: 10.1007/978-3-030-81652-0_16.

[10] Jie Cai, Han Jiang, Pingyuan Zhang, Zhihua Zheng, Hao Wang, Guangshi Lü, and Qiuliang Xu. "ID-Based Strong Designated Verifier Signature over $\mathcal{R}$-SIS Assumption". In: *Security and Communication Networks* 2019 (2019), 9678095:1–9678095:8. DOI: 10.1155/2019/9678095.

[11] Ran Canetti, Oded Goldreich, and Shai Halevi. "The Random Oracle Methodology, Revisited (Preliminary Version)". In: *30th ACM STOC*. ACM Press, May 1998, pp. 209–218. DOI: 10.1145/276698.276741.

[12] Daniel Collins, Loïs Huguenin-Dumittan, Ngoc Khanh Nguyen, Nicolas Rolin, and Serge Vaudenay. "K-Waay: Fast and Deniable Post-Quantum X3DH without Ring Signatures". In: *USENIX Security 2024*. Ed. by Davide Balzarotti and Wenyuan Xu. USENIX Association, Aug. 2024.

[13] Cas Cremers, Alexander Dax, and Aurora Naska. "Formal Analysis of SPDM: Security Protocol and Data Model version 1.2". In: *USENIX Security 2023*. Ed. by Joseph A. Calandrino and Carmela Troncoso. USENIX Association, Aug. 2023, pp. 6611–6628.

[14] Cas Cremers, Samed Düzlü, Rune Fiedler, Marc Fischlin, and Christian Janson. "BUFFing signature schemes beyond unforgeability and the case of post-quantum signatures". In: *2021 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2021, pp. 1696–1714. DOI: 10.1109/SP40001.2021.00093.

[15] Ivan Damgård. "Towards Practical Public Key Systems Secure Against Chosen Ciphertext Attacks". In: *CRYPTO'91*. Ed. by Joan Feigenbaum. Vol. 576. LNCS. Springer, Berlin, Heidelberg, Aug. 1992, pp. 445–456. DOI: 10.1007/3-540-46766-1_36.

[16] Mario Di Raimondo and Rosario Gennaro. "New Approaches for Deniable Authentication". In: *ACM CCS 2005*. Ed. by Vijayalakshmi Atluri, Catherine Meadows, and Ari Juels. ACM Press, Nov. 2005, pp. 112–121. DOI: 10.1145/1102120.1102137.

[17] Mario Di Raimondo, Rosario Gennaro, and Hugo Krawczyk. "Deniable authentication and key exchange". In: *ACM CCS 2006*. Ed. by Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati. ACM Press, 2006, pp. 400–409. DOI: 10.1145/1180405.1180454.

[18] Samuel Dobson and Steven D. Galbraith. "Post-Quantum Signal Key Agreement from SIDH". In: *Post-Quantum Cryptography - 13th International Workshop, PQCrypto 2022*. Ed. by Jung Hee Cheon and Thomas Johansson. Springer, Cham, Sept. 2022, pp. 422–450. DOI: 10.1007/978-3-031-17234-2_20.

[19] Danny Dolev, Cynthia Dwork, and Moni Naor. "Non-Malleable Cryptography (Extended Abstract)". In: *23rd ACM STOC*. ACM Press, May 1991, pp. 542–552. DOI: 10.1145/103418.103474.

[20] Cynthia Dwork, Moni Naor, and Amit Sahai. "Concurrent Zero-Knowledge". In: *30th ACM STOC*. ACM Press, May 1998, pp. 409–418. DOI: 10.1145/276698.276853.

[21] Uriel Feige and Adi Shamir. "Zero Knowledge Proofs of Knowledge in Two Rounds". In: *CRYPTO'89*. Ed. by Gilles Brassard. Vol. 435. LNCS. Springer, New York, Aug. 1990, pp. 526–544. DOI: 10.1007/0-387-34805-0_46.

[22] Amos Fiat and Adi Shamir. "How to Prove Yourself: Practical Solutions to Identification and Signature Problems". In: *CRYPTO'86*. Ed. by Andrew M. Odlyzko. Vol. 263. LNCS. Springer, Berlin, Heidelberg, Aug. 1987, pp. 186–194. DOI: 10.1007/3-540-47721-7_12.

[23] Rune Fiedler and Christian Janson. "A Deniability Analysis of Signal's Initial Handshake PQXDH". In: *PoPETs* 2024.4 (Oct. 2024), pp. 907–928. DOI: 10.56553/popets-2024-0148.

[24] Keitaro Hashimoto, Shuichi Katsumata, Kris Kwiatkowski, and Thomas Prest. "An Efficient and Generic Construction for Signal's Handshake (X3DH): Post-Quantum, State Leakage Secure, and Deniable". In: *PKC 2021, Part II*. Ed. by Juan Garay. Vol. 12711. LNCS. Springer, Cham, May 2021, pp. 410–440. DOI: 10.1007/978-3-030-75248-4_15.

[25] Dennis Hofheinz, Tibor Jager, Dakshita Khurana, Amit Sahai, Brent Waters, and Mark Zhandry. "How to Generate and Use Universal Samplers". In: *ASIACRYPT 2016, Part II*. Ed. by Jung Hee Cheon and Tsuyoshi Takagi. Vol. 10032. LNCS. Springer, Berlin, Heidelberg, Dec. 2016, pp. 715–744. DOI: 10.1007/978-3-662-53890-6_24.

[26] Dennis Hofheinz, Julia Kastner, and Karen Klein. "The Power of Undirected Rewindings for Adaptive Security". In: *CRYPTO 2023, Part II*. Ed. by Helena Handschuh and Anna Lysyanskaya. Vol. 14082. LNCS. Springer, Cham, Aug. 2023, pp. 725–758. DOI: 10.1007/978-3-031-38545-2_24.

[27] Qiong Huang, Guomin Yang, Duncan S. Wong, and Willy Susilo. *Efficient Strong Designated Verifier Signature Schemes without Random Oracles or Delegatability*. Cryptology ePrint Archive, Report 2009/518. 2009. URL: https://eprint.iacr.org/2009/518.

[28] Aayush Jain, Huijia Lin, and Amit Sahai. "Indistinguishability Obfuscation from LPN over $\mathbb{F}_p$, DLIN, and PRGs in NC^0". In: *EUROCRYPT 2022, Part I*. Ed. by Orr Dunkelman and Stefan Dziembowski. Vol. 13275. LNCS. Springer, Cham, 2022, pp. 670–699. DOI: 10.1007/978-3-031-06944-4_23.

[29] Aayush Jain, Huijia Lin, and Amit Sahai. "Indistinguishability obfuscation from well-founded assumptions". In: *53rd ACM STOC*. Ed. by Samir Khuller and Virginia Vassilevska Williams. ACM Press, June 2021, pp. 60–73. DOI: 10.1145/3406325.3451093.

[30] Markus Jakobsson, Kazue Sako, and Russell Impagliazzo. "Designated Verifier Proofs and Their Applications". In: *EUROCRYPT'96*. Ed. by Ueli M. Maurer. Vol. 1070. LNCS. Springer, Berlin, Heidelberg, May 1996, pp. 143–154. DOI: 10.1007/3-540-68339-9_13.

[31] Julia Kastner, Julian Loss, and Jiayu Xu. "The Abe-Okamoto Partially Blind Signature Scheme Revisited". In: *ASIACRYPT 2022, Part IV*. Ed. by Shweta Agrawal and Dongdai Lin. Vol. 13794. LNCS. Springer, Cham, Dec. 2022, pp. 279–309. DOI: 10.1007/978-3-031-22972-5_10.

[32] Dmitry Khovratovich, Ron D. Rothblum, and Lev Soukhanov. *How to Prove False Statements: Practical Attacks on Fiat-Shamir*. Cryptology ePrint Archive, Paper 2025/118. 2025. URL: https://eprint.iacr.org/2025/118.

[33] Ehren Kret and Rolfe Schmidt. *The PQXDH key agreement protocol*. https://signal.org/docs/specifications/pqxdh/. September 2023.

[34] Caroline Kudla and Kenneth G. Paterson. "Non-interactive Designated Verifier Proofs and Undeniable Signatures". In: *10th IMA International Conference on Cryptography and Coding*. Ed. by Nigel P. Smart. Vol. 3796. LNCS. Springer, Berlin, Heidelberg, Dec. 2005, pp. 136–154. DOI: 10.1007/11586821_10.

[35] Fabien Laguillaumie and Damien Vergnaud. "Designated Verifier Signatures: Anonymity and Efficient Construction from Any Bilinear Map". In: *SCN 04*. Ed. by Carlo Blundo and Stelvio Cimato. Vol. 3352. LNCS. Springer, Berlin, Heidelberg, Sept. 2005, pp. 105–119. DOI: 10.1007/978-3-540-30598-9_8.

[36] Yong Li, Helger Lipmaa, and Dingyi Pei. "On Delegatability of Four Designated Verifier Signatures". In: *ICICS 05*. Ed. by Sihan Qing, Wenbo Mao, Javier López, and Guilin Wang. Vol. 3783. LNCS. Springer, Berlin, Heidelberg, Dec. 2005, pp. 61–71. DOI: 10.1007/11602897_6.

[37] Yong Li, Willy Susilo, Yi Mu, and Dingyi Pei. "Designated Verifier Signature: Definition, Framework and New Constructions". In: *Ubiquitous Intelligence and Computing, 4th International Conference, UIC 2007, Hong Kong, China, July 11-13, 2007, Proceedings*. Ed. by Jadwiga Indulska, Jianhua Ma, Laurence Tianruo Yang, Theo Ungerer, and Jiannong Cao. Vol. 4611. Lecture Notes in Computer Science. Springer, 2007, pp. 1191–1200. DOI: 10.1007/978-3-540-73549-6_116.

[38] Helger Lipmaa, Guilin Wang, and Feng Bao. "Designated Verifier Signature Schemes: Attacks, New Security Notions and a New Construction". In: *ICALP 2005*. Ed. by Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung. Vol. 3580. LNCS. Springer, Berlin, Heidelberg, July 2005, pp. 459–471. DOI: 10.1007/11523468_38.

[39] Moxie Marlinspike and Trevor Perrin. *The X3DH key agreement protocol*. https://signal.org/docs/specifications/x3dh/. November 2016.

[40] Geontae Noh and Ik Rae Jeong. "Strong designated verifier signature scheme from lattices in the standard model". In: *Security and Communication Networks* 9.18 (2016), pp. 6202–6214. DOI: 10.1002/SEC.1766.

[41] Rafael Pass. "On Deniability in the Common Reference String and Random Oracle Model". In: *CRYPTO 2003*. Ed. by Dan Boneh. Vol. 2729. LNCS. Springer, Berlin, Heidelberg, Aug. 2003, pp. 316–337. DOI: 10.1007/978-3-540-45146-4_19.

[42] Seyoon Ragavan, Neekon Vafa, and Vinod Vaikuntanathan. "Indistinguishability Obfuscation from Bilinear Maps and LPN Variants". In: *TCC 2024, Part IV*. Ed. by Elette Boyle and Mohammad Mahmoody. Vol. 15367. LNCS. Springer, Cham, Dec. 2024, pp. 3–36. DOI: 10.1007/978-3-031-78023-3_1.

[43] Ronald L. Rivest, Adi Shamir, and Yael Tauman. "How to Leak a Secret". In: *ASIACRYPT 2001*. Ed. by Colin Boyd. Vol. 2248. LNCS. Springer, Berlin, Heidelberg, Dec. 2001, pp. 552–565. DOI: 10.1007/3-540-45682-1_32.

[44] Michal Rjaško and Martin Stanek. *On Designated Verifier Signature Schemes*. Cryptology ePrint Archive, Report 2010/191. 2010. URL: https://eprint.iacr.org/2010/191.

[45] Shahrokh Saeednia, Steve Kremer, and Olivier Markowitch. "An Efficient Strong Designated Verifier Signature Scheme". In: *ICISC 03*. Ed. by Jong In Lim and Dong Hoon Lee. Vol. 2971. LNCS. Springer, Berlin, Heidelberg, Nov. 2004, pp. 40–54. DOI: 10.1007/978-3-540-24691-6_4.

[46] Amit Sahai and Brent Waters. "How to use indistinguishability obfuscation: deniable encryption, and more". In: *46th ACM STOC*. Ed. by David B. Shmoys. ACM Press, 2014, pp. 475–484. DOI: 10.1145/2591796.2591825.

[47] Daniel Shanks. "Class number, a theory of factorization, and genera". In: *1969 Number Theory Institute*. Ed. by Donald J. Lewis. Vol. 20. Proceedings of Symposia in Pure Mathematics. Providence, Rhode Island: American Mathematical Society, 1971, pp. 415–440. ISBN: 0–8218–1420–6. DOI: 10.1090/pspum/020/0316385.

[48] Kyung-Ah Shim. "Rogue-key attacks on the multi-designated verifiers signature scheme". In: *Inf. Process. Lett.* 107.2 (2008), pp. 83–86. DOI: 10.1016/J.IPL.2007.11.021.

[49] Victor Shoup. "Lower Bounds for Discrete Logarithms and Related Problems". In: *EUROCRYPT'97*. Ed. by Walter Fumy. Vol. 1233. LNCS. Springer, Berlin, Heidelberg, May 1997, pp. 256–266. DOI: 10.1007/3-540-69053-0_18.

[50] Xi Sun, Haibo Tian, and Yumin Wang. "Toward Quantum-Resistant Strong Designated Verifier Signature from Isogenies". In: *2012 Fourth International Conference on Intelligent Networking and Collaborative Systems, INCoS 2012, Bucharest, Romania, September 19-21, 2012*. Ed. by Fatos Xhafa, Leonard Barolli, Florin Pop, Xiaofeng Chen, and Valentin Cristea. IEEE, 2012, pp. 292–296. DOI: 10.1109/INCOS.2012.70.

[51] P. Thanalakshmi, R. Anitha, N. Anbazhagan, Chulho Park, Gyanendra Prasad Joshi, and Changho Seo. "A Hash-Based Quantum-Resistant Designated Verifier Signature Scheme". In: *Mathematics* 10.10 (2022). ISSN: 2227-7390. DOI: 10.3390/math10101642.

[52] Haibo Tian, Xiaofeng Chen, and Jin Li. "A Short Non-delegatable Strong Designated Verifier Signature". In: *ACISP 12*. Ed. by Willy Susilo, Yi Mu, and Jennifer Seberry. Vol. 7372. LNCS. Springer, Berlin, Heidelberg, July 2012, pp. 261–279. DOI: 10.1007/978-3-642-31448-3_20.

[53] Raylin Tso, Takeshi Okamoto, and Eiji Okamoto. "Practical Strong Designated Verifier Signature Schemes Based on Double Discrete Logarithms". In: *Information Security and Cryptology, First SKLOIS Conference, CISC 2005, Beijing, China, December 15-17, 2005, Proceedings*. Ed. by Dengguo Feng, Dongdai Lin, and Moti Yung. Vol. 3822. Lecture Notes in Computer Science. Springer, 2005, pp. 113–127. DOI: 10.1007/11599548_10.

[54] Nik Unger and Ian Goldberg. "Deniable Key Exchanges for Secure Messaging". In: *ACM CCS 2015*. Ed. by Indrajit Ray, Ninghui Li, and Christopher Kruegel. ACM Press, Oct. 2015, pp. 1211–1223. DOI: 10.1145/2810103.2813616.

[55] Nik Unger and Ian Goldberg. "Improved Strongly Deniable Authenticated Key Exchanges for Secure Messaging". In: *PoPETs* 2018.1 (Jan. 2018), pp. 21–66. DOI: 10.1515/popets-2018-0003.

[56] Nihal Vatandas, Rosario Gennaro, Bertrand Ithurburn, and Hugo Krawczyk. "On the Cryptographic Deniability of the Signal Protocol". In: *ACNS 20International Conference on Applied Cryptography and Network Security, Part II*. Ed. by Mauro Conti, Jianying Zhou, Emiliano Casalicchio, and Angelo Spognardi. Vol. 12147. LNCS. Springer, Cham, Oct. 2020, pp. 188–209. DOI: 10.1007/978-3-030-57878-7_10.

[57] Fenghe Wang, Yupu Hu, and Baocang Wang. “Lattice-based strong designate verifier signature and its applications”. In: *Malaysian Journal of Computer Science* 25 (1 2012), pp. 11–22.

[58] Yunmei Zhang, Man Ho Au, Guomin Yang, and Willy Susilo. “(Strong) Multi-Designated Verifiers Signatures Secure against Rogue Key Attack”. In: *Network and System Security - 6th International Conference, NSS 2012, Wuyishan, Fujian, China, November 21-23, 2012. Proceedings*. Ed. by Li Xu, Elisa Bertino, and Yi Mu. Vol. 7645. Lecture Notes in Computer Science. Springer, 2012, pp. 334–347. DOI: 10.1007/978-3-642-34601-9_25.

$\underline{\mathsf{Setup}'()}$:
1 **return** $\mathsf{Setup}()$

$\underline{\mathsf{SKGen}'(\mathsf{pp})}$:
2 **return** $\mathsf{SKGen}(\mathsf{pp})$

$\underline{\mathsf{VKGen}'(\mathsf{pp}, \mathsf{id}_D)}$:
3 $(\mathsf{pk}_D, \mathsf{sk}_D) \xleftarrow{\$} \mathsf{VKGen}(\mathsf{pp}, \mathsf{id}_D)$
4 **return** $(\mathsf{pk}_D||0, \mathsf{sk}_D)$

$\underline{\mathsf{Sign}'(\mathsf{pp}, \mathsf{sk}_S, \mathsf{pk}_D||a, \mathsf{id}_D, m)}$:
5 $\sigma \xleftarrow{\$} \mathsf{Sign}(\mathsf{pp}, \mathsf{sk}_S, \mathsf{pk}_D, \mathsf{id}_D, m)$
6 **if** $a = 1$ **then return** $\sigma||\mathsf{sk}_S$
7 **return** σ

$\underline{\mathsf{Sim}'(\mathsf{pp}, \mathsf{pk}_S, \mathsf{sk}_D, \mathsf{id}_D, m)}$:
8 **return** $\mathsf{Sim}(\mathsf{pp}, \mathsf{pk}_S, \mathsf{sk}_D, \mathsf{id}_D, m)$

$\underline{\mathsf{Vrfy}'(\mathsf{pp}, \mathsf{pk}_S, \mathsf{pk}_D, \mathsf{id}_D, m, \sigma||b)}$:
9 **return** $\mathsf{Vrfy}(\mathsf{pp}, \mathsf{pk}_S, \mathsf{pk}_D, \mathsf{id}_D, m, \sigma)$

Fig. 11. A DVS scheme $\mathsf{DVS}' = (\mathsf{Setup}', \mathsf{SKGen}', \mathsf{VKGen}', \mathsf{Sign}', \mathsf{Sim}', \mathsf{Vrfy}')$ based on a DVS scheme $\mathsf{DVS} = (\mathsf{Setup}, \mathsf{SKGen}, \mathsf{VKGen}, \mathsf{Sign}, \mathsf{Sim}, \mathsf{Vrfy})$ that is unforgeable and source hiding. Separates uf and mal-uf (see Theorem 13), as well as sh and s-mal-sh (see Theorem 14).

A Separations of security notions for DVS

Here, we separate our new security notions for DVS against malicious verifiers from the existing ones against honest verifiers. While malicious unforgeability is strictly stronger than (honest) unforgeability), source hiding and simplified malicious source hiding are actually incomparable.

DVS unforgeability. We begin with separating unforgeability from malicious unforgeability.

Theorem 13. *Assume an $(t, \varepsilon, q_S, q_V, q_{VK})$-unforgeable DVS scheme exists, then there exists a DVS scheme that is $(t, \varepsilon, q_S, q_V, q_{VK})$-unforgeable and not $(t', \varepsilon', q'_S, q'_V, q'_{VK})$-malicious unforgeable for arbitrary t', q'_V, q'_{VK}, and $q'_S \geq 1$ and $\varepsilon' < 1$.*

Proof. Given an unforgeable DVS scheme DVS, we build DVS' as shown in Figure 11: VKGen' appends a zero bit to the public key. Sign' checks the extra bit of the verifier's public key: If it is 1, it appends sk_S to the signature. All other algorithms ignore the extra bit and work the same way as DVS.

This scheme retains unforgeability of DVS since the adversary cannot trigger the special condition in Sign' within the Sign oracle with honestly generated verifier keys.

However, this scheme is not malicious unforgeable: The adversary can run $\mathsf{VKGen}'(\mathsf{pp}, \mathsf{id}_D)$ (for some $\mathsf{id}_D \in \mathcal{IDS}$) to obtain $(\mathsf{pk}_D||0, \mathsf{sk}_D)$ and query the signing oracle on the public key and identity $\mathsf{pk}_D||1, \mathsf{id}_D$ for any message. Thereby it learns the signer's secret key and can sign any message from the challenge signer designated for the challenge verifier. □

On the other hand, malicious unforgeability implies (honest) unforgeability since any adversary that breaks (honest) unforgeability also breaks malicious unforgeability.

DVS source hiding. Next, we show that source hiding and simplified malicious source hiding are incomparable. On a technical note, we additionally need to assume that it is not feasible to compute secret keys from public keys (and other public information) in the first separation. Observe that this is a reasonable assumption since otherwise anybody can forge signatures.

Theorem 14. *Assume a (t, ε, q_C)-source hiding DVS scheme exists for which it is not feasible to compute the secret key from public information (including the public key), then there exists a DVS scheme that is (t, ε, q_C)-source hiding and not $(t_{\mathcal{A}}, t_{\mathcal{E}}, t_{\mathcal{D}}, \varepsilon', q_{VK})$-simplified malicious source hiding for arbitrary $t_{\mathcal{A}}, t_{\mathcal{E}}, t_{\mathcal{D}}, q_R$, and $\varepsilon' < 1$.*

Proof. Once more, we use the scheme shown in Figure 11, which appends a 0 to honestly generated public keys and where signing designated for verifier public keys with a trailing 1 leaks the signer secret key.

The scheme retains source hiding of DVS since the adversary cannot trigger the special condition in Sign' within the CHALL oracle with honestly generated verifier keys.

However, the scheme is not simplified malicious source hiding: The adversary can run $\mathsf{VKGen}'(\mathsf{pp}, \mathsf{id}_D)$ (for some $\mathsf{id}_D \in \mathcal{IDS}$) to obtain $(\mathsf{pk}_D||0, \mathsf{sk}_D)$ and query the challenge oracle on the public key and identity $\mathsf{pk}_D||1, \mathsf{id}_D$ for any message. Due to the trailing 1 bit of the public key, the signature now has the signer secret key appended. For $b = 1$, the extractor needs to produce a signature with sk_S appended, even though the adversary does not know sk_S. Hence, if the extractor succeeds, it must be able to compute the secret key sk_S from public information, contradicting our assumption. □

The perhaps surprising separation from simplified malicious source hiding to source hiding can intuitively be explained with the source hiding adversary having access to the challenge signer secret key sk_S and the challenge oracle CHALL at the same time, which neither the adversary nor the distinguisher in the s-mal-sh game have.

Theorem 15. *Assume a $(t_{\mathcal{A}}, t_{\mathcal{E}}, t_{\mathcal{D}}, \varepsilon, q_{VK})$-simplified malicious source hiding DVS scheme exists, then there exists a DVS scheme that is $(t_{\mathcal{A}}, t_{\mathcal{E}}, t_{\mathcal{D}}, \varepsilon - 2^{-\lambda}, q_{VK})$-simplified malicious source hiding and for all possible choices of Sim' not $(t, 3/4, 1)$-source hiding for arbitrary t and for any $\lambda \in \mathbb{N}$.*

Proof. Given a simplified malicious source hiding DVS scheme DVS we construct a new DVS scheme DVS' as described in Figure 12: Key generation appends a randomly sampled bit string r of length λ to the secret key and when calling Sign' with the signer secret key as message the algorithm appends a 1 bit to the signature and otherwise a 0 bit and Vrfy' ignores the extra bit.

The scheme DVS' retains simplified malicious source hiding of DVS. The extractor for DVS' runs the extractor for DVS on its own input and returns the resulting signature with an appended 0 bit. Since the view of the adversary is independent of r, the probability that it made a query where Sign' would append a 1 bit is at most $2^{-\lambda}$.

$\underline{\mathsf{Setup}'()}$:

1 **return** $\mathsf{Setup}()$

$\underline{\mathsf{SKGen}'(\mathsf{pp})}$:

2 $(\mathsf{pk}_S, \mathsf{sk}_S) \xleftarrow{\$} \mathsf{SKGen}(\mathsf{pp})$
3 $r \xleftarrow{\$} \{0,1\}^\lambda$
4 **return** $(\mathsf{pk}_S, (\mathsf{sk}_S, r))$

$\underline{\mathsf{VKGen}'(\mathsf{pp}, \mathsf{id}_D)}$:

5 **return** $\mathsf{VKGen}(\mathsf{pp}, \mathsf{id}_D)$

$\underline{\mathsf{Sign}'(\mathsf{pp}, \mathsf{sk}'_S = (\mathsf{sk}_S, r), \mathsf{pk}_D, \mathsf{id}_D, m)}$:

6 $\sigma \xleftarrow{\$} \mathsf{Sign}(\mathsf{pp}, \mathsf{sk}_S, \mathsf{pk}_D, \mathsf{id}_D, m)$
7 **if** $m = \mathsf{sk}'_S$ **then return** $\sigma||1$
8 **return** $\sigma||0$

$\underline{\mathsf{Vrfy}'(\mathsf{pp}, \mathsf{pk}_S, \mathsf{sk}_D, \mathsf{id}_D, m, \sigma||b)}$:

9 **return** $\mathsf{Vrfy}(\mathsf{pp}, \mathsf{pk}_S, \mathsf{sk}_D, \mathsf{id}_D, m, \sigma)$

Fig. 12. A DVS scheme $\mathsf{DVS}' = (\mathsf{Setup}', \mathsf{SKGen}', \mathsf{VKGen}', \mathsf{Sign}', \mathsf{Vrfy}')$ that is simplified malicious source hiding but not source hiding, based on a DVS scheme $\mathsf{DVS} = (\mathsf{Setup}, \mathsf{SKGen}, \mathsf{VKGen}, \mathsf{Sign}, \mathsf{Sim}, \mathsf{Vrfy})$ that is simplified malicious source hiding (see Theorem 15). In Theorem 15 we show that no algorithm Sim' can exist.

However, the scheme DVS' is not source hiding: The adversary first obtains a verifier key with the VKey oracle[10]. Next, it parses the signer secret key as $(\mathsf{sk}_S, r) \leftarrow \mathsf{sk}'_S$, samples $r^* \xleftarrow{\$} \{0,1\}^\lambda$, and sets $\mathsf{sk}^*_S \leftarrow (\mathsf{sk}_S, r^*)$. Now, the adversary flips a bit $b'' \xleftarrow{\$} \{0,1\}$ and queries the Chall oracle on the messages sk'_S if $b'' = 1$ or sk^*_S if $b'' = 0$. The adversary outputs the guess 0 if the last bit of the signature received from Chall matches b'' and 1 otherwise.

If the game is played with $b = 0$ (where Chall uses $\mathsf{DVS}'.\mathsf{Sign}'$), the adversary always outputs the correct bit. If the game is played with $b = 1$, the inputs to the Sim' algorithm are statistically independent of r and thus sk'_S and sk^* are identically distributed given only the inputs of Sim'. Thus, the last bit of the signature it outputs is statistically independent of b'' and the adversary guesses b'' correctly with probability 3/4. □

B On the deniability of Signal's initial handshake X3DH

In Section 5.2 we show that the EKDH assumption is broken. Here, we use the same strategy to show that Signal's (former) initial handshake protocol X3DH is not deniable if the auxiliary input contains a hint on a public key, and the hinted CDH problem is hard for an algorithm bound to the runtime of the simulator (roughly corresponding to the extractor in our simplified malicious source hiding definition), but not for an algorithm bound to the runtime of the distinguisher.

Let us briefly review the X3DH handshake with the help of Figure 13. All parties have a long-term DH key pair, several semi-static DH key pairs (which are used for several sessions each), and ephemeral DH key pairs (each used only once). Furthermore, parties have a long-term signing key, which they use to authenticate their semi-static public keys. First, Bob builds a so-called pre-key bundle, which is peer independent, and consists of his long-term public keys, his

[10] Keep in mind that this oracle is provided by the sh game, whereas the s-mal-sh game provides a VKeys oracle.

Fig. 13. Signal's initial handshake protocols X3DH and PQXDH, following the presentation of [23, Figure 3] (but without the AEAD and user messages). The KEM with gray background is exclusive to PQXDH.

semi-static public key including the signature, and, optionally, his ephemeral public key. Upon receiving Bob's pre-key bundle, Alice verifies the signature and computes several DH combinations: long-term–semi-static, ephemeral–long-term, ephemeral–semi-static, and, if Bob's pre-key bundle contains an ephemeral key, ephemeral–ephemeral. She uses KDF to derive the session key from these four DH shared secrets and sends her ephemeral public key to Bob, who can compute the session key in the same way. Alice's message can be combined with an AEAD ciphertext of her first user message (such as "Hi Bob, how're you doing?") encrypted under the session key[11]. In this case, Bob decrypts the AEAD ciphertext and aborts if decryption fails. Whether this AEAD ciphertext is part of the initial handshake is ambiguous [39, 33]. Vatandas et al. [56] did not model the AEAD ciphertext as part of the initial handshake[12], while [23] did. We do not consider the AEAD as part of the initial handshake.

Our result contrasts the result of [56], which shows deniability assuming the EKDH assumption without restricting the admissible auxiliary inputs. We embed a hinted CDH challenge as auxiliary input and leverage the difference in runtime between simulator and distinguisher to exclude and ensure breaking the hinted CDH assumption with different parameters. Though, this does not invalidate the results of [23], since they use a specific class of auxiliary inputs, i.e., one pre-key bundle per user and semi-static key[13]. This does not include a hint on a public key and therefore we cannot reduce deniability of X3DH to breaking the hinted CDH assumption for this class of auxiliary inputs.

Our attack relies on embedding a hinted CDH challenge and not on the presence or absence of the AEAD ciphertext in Alice's message. Hence, we expect our attack to transfer even if one considers the AEAD ciphertext as part of Alice's message, assuming the AEAD scheme achieves INT-CTXT.

Our attack also applies to the deniability of PQXDH if the auxiliary inputs lend themselves to building a hinted CDH challenge. For auxiliary inputs that do not facilitate forming a hinted CDH challenge our results do not rule out deniability, as discussed above.

We borrow the deniability definition of Vatandas et al. [56, Definition 5], which adapts the definition of Di Raimondo et al. [17, Definition 2] to concrete security, and we adapt it further for syntactical differences and more detailed treatment of auxiliary inputs. In particular, both prior definitions have quantified the auxiliary input before the simulator. This allows the simulator to arbitrarily depend on the auxiliary input, including on all secret keys corresponding to public keys used in the auxiliary input. Furthermore, both definitions first quantify auxiliary inputs and then sample keys. However, inspection of their proofs suggests that the

[11] More specifically, encrypted under a key derived from the session key. But the details of Signal's key scheduling do not affect our results.

[12] They consider only two different lifetimes for Bob as well; it is not immediately obvious if they dropped semi-static or ephemeral keys, as discussed in [23].

[13] They use the auxiliary input to ensure that a signature is available when faking a transcript. They also have a second setting in which the Fake algorithm learns the signature from an oracle. In this second setting they do not rely on auxiliary inputs.

auxiliary input depends on the sampled keys. Hence, we use a sampling algorithm for auxiliary inputs AuxPrep that depends on the sampled keys. This auxiliary input sampled with AuxPrep, called aux_1 below, can be used to help the simulator. In contrast, we add a second auxiliary input that can help the adversary to the disadvantage of the simulator. This second input, called aux_2, models arbitrary data that the adversary may have access to, following our approach for simplified malicious source hiding of DVS (cp. Definition 19). We will use this aux_2 to embed a public key for the hinted CDH challenge that, allowing us to disprove deniability of X3DH. Though, the basic idea, including that the adversary's view from interacting with an oracle is compared to a simulated view, remains the same.

Definition 22 (Deniable Key Exchange (adapted from [17, 56])). *An AKE protocol* $\Pi = (\mathsf{KGenLT}, \mathsf{KGenSS}, \mathsf{Run})$ *is a* $(t_{\mathcal{A}}, t_{\mathcal{S}}, t_{\mathcal{D}}, \varepsilon_{\mathcal{D}})$*–concurrently deniable key exchange protocol wrt. auxiliary input sampler* AuxPrep *if for any adversary* $\mathcal{A}$ *running in time* $t_{\mathcal{A}}$*,there exists a simulator* SIM *running in time* $t_{\mathcal{S}}$*,such that for the two distributions*

Real():

1 $\mathcal{L}_{\mathsf{pk}} \leftarrow \emptyset; \mathcal{L}_{\mathsf{sk}} \leftarrow \emptyset$
2 **for** $U \in [n_p]$
3 $\quad(\mathsf{ltpk}_U, \mathsf{ltsk}_U) \xleftarrow{\$} \mathsf{KGenLT}()$
4 $\quad$**for** $\mathsf{ssid} \in [n_{ss}]$
5 $\quad\quad(\mathsf{sspk}_U^{\mathsf{ssid}}, \mathsf{sssk}_U^{\mathsf{ssid}}) \xleftarrow{\$} \mathsf{KGenSS}()$
6 $\quad\mathcal{L}_{\mathsf{pk}} \leftarrow \mathcal{L}_{\mathsf{pk}} \cup \{\mathsf{ltpk}_U, \{\mathsf{sspk}_U^{\mathsf{ssid}}\}^{\mathsf{ssid} \in [n_{ss}]}\}$
7 $\quad\mathcal{L}_{\mathsf{sk}} \leftarrow \mathcal{L}_{\mathsf{sk}} \cup \{\mathsf{ltsk}_U, \{\mathsf{sssk}_U^{\mathsf{ssid}}\}^{\mathsf{ssid} \in [n_{ss}]}\}$
8 $\mathsf{aux}_1 \xleftarrow{\$} \mathsf{AuxPrep}(\mathcal{L}_{\mathsf{pk}}, \mathcal{L}_{\mathsf{sk}})$
9 $\mathsf{aux} \leftarrow (\mathsf{aux}_1, \mathsf{aux}_2)$
10 **return** $(\mathsf{aux}, \mathcal{L}_{\mathsf{pk}}, View(\mathcal{A}(\mathcal{L}_{\mathsf{pk}}, \mathsf{aux})))$

Sim():

11 $\mathcal{L}_{\mathsf{pk}} \leftarrow \emptyset; \mathcal{L}_{\mathsf{sk}} \leftarrow \emptyset$
12 **for** $U \in [n_p]$
13 $\quad(\mathsf{ltpk}_U, \mathsf{ltsk}_U) \xleftarrow{\$} \mathsf{KGenLT}()$
14 $\quad$**for** $\mathsf{ssid} \in [n_{ss}]$
15 $\quad\quad(\mathsf{sspk}_U^{\mathsf{ssid}}, \mathsf{sssk}_U^{\mathsf{ssid}}) \xleftarrow{\$} \mathsf{KGenSS}()$
16 $\quad\mathcal{L}_{\mathsf{pk}} \leftarrow \mathcal{L}_{\mathsf{pk}} \cup \{\mathsf{ltpk}_U, \{\mathsf{sspk}_U^{\mathsf{ssid}}\}^{\mathsf{ssid} \in [n_{ss}]}\}$
17 $\quad\mathcal{L}_{\mathsf{sk}} \leftarrow \mathcal{L}_{\mathsf{sk}} \cup \{\mathsf{ltsk}_U, \{\mathsf{sssk}_U^{\mathsf{ssid}}\}^{\mathsf{ssid} \in [n_{ss}]}\}$
18 $\mathsf{aux}_1 \xleftarrow{\$} \mathsf{AuxPrep}(\mathcal{L}_{\mathsf{pk}}, \mathcal{L}_{\mathsf{sk}})$
19 $\mathsf{aux} \leftarrow (\mathsf{aux}_1, \mathsf{aux}_2)$
20 **return** $(\mathsf{aux}, \mathcal{L}_{\mathsf{pk}}, SIM(\mathcal{L}_{\mathsf{pk}}, \mathsf{aux}))$

we have for all probabilistic polynomial time $\mathcal{D}$ *and all* $\mathsf{aux}_2 \in \{0,1\}^*$

$$\left| \Pr_{x \in Real()}[\mathcal{D}(x) = 1] - \Pr_{x \in Sim()}[\mathcal{D}(x) = 1] \right| \leq \varepsilon_{\mathcal{D}}.$$

Equipped with this definition we proceed to show that deniability of X3DH does not hold under our Definition 22 if we leverage the auxiliary input to pass a hinted CDH challenge to the adversary and appropriate parameters.

Theorem 16. *If* $\mathcal{G} = (\mathbb{G}, g, q)$ *is a group where the* (t, ε, η)*-hinted CDH assumption holds for* $\varepsilon < 1$*, but the* (t', ε', η)*-hinted CDH assumption (for some* $t < t'$*) does not hold, and we model* KDF *as random oracle, then the* X3DH *protocol over* $\mathcal{G}$ *as described in Figure 13 is not* $(t_{\mathcal{A}}, t_{\mathcal{S}}, t_{\mathcal{D}}, \varepsilon_{\mathcal{D}})$*-deniable wrt. auxiliary inputs sampled with any* AuxPrep *for* $t_{\mathcal{A}}$ *small,* $t_{\mathcal{S}} \approx t, t_{\mathcal{D}} \approx t'$ *and* $\varepsilon_{\mathcal{D}} \leq \varepsilon' - \varepsilon/2$*.*

Proof. For the real distribution, the adversary $\mathcal{A}(\mathcal{L}_{\mathsf{pk}}, \mathsf{aux} = (\mathsf{aux}_1, (g^z, \lfloor z/\eta \rfloor))$ can set g^z as long-term public key for some party Alice, sample $\mathsf{esk}_A = x \xleftarrow{\$} \mathcal{G}$, and compute $\mathsf{epk}_A \leftarrow g^x$ as an ephemeral public key. Next, it runs a session between Alice with ephemeral public key epk_A and some party Bob, where the

adversary controls Alice and accesses its oracle for Bob. After this one session, the adversary terminates. In the simulated distribution, the simulator has to produce indistinguishable output upon receiving the same input and without oracle access.

The distinguisher now gets the auxiliary input aux, the list of public keys $\mathcal{L}_{\mathsf{pk}}$, the transcript of messages, the session key(s) computed by the oracle, and the randomness of the adversary. The distinguisher can first run the adversary $\mathcal{A}_{\mathsf{HCDH}'}$ against the (t', ε', η)-hinted CDH assumption on input $\mathsf{aux}_{\mathsf{HCDH}} = (\mathsf{sspk}_B, g^z, \lfloor z/\eta \rfloor)$, where sspk_B is Bob's semi-static public key in the session above. The hinted CDH adversary $\mathcal{A}_{\mathsf{HCDH}'}$ outputs $\widetilde{\mathsf{DH}_1}$, and the distinguisher can compute $\mathsf{DH}_2, \mathsf{DH}_3, \mathsf{DH}_4$ with the knowledge of Alice's ephemeral secret key esk_A, which it obtains from the adversary's randomness. Finally, the distinguisher recomputes the session key as $\widetilde{K} \leftarrow \mathsf{KDF}(\widetilde{\mathsf{DH}_1}, \mathsf{DH}_2, \mathsf{DH}_3, \mathsf{DH}_4)$. If $\widetilde{K}$ is identical to the session key provided to the distinguisher as input, it outputs 0, otherwise 1.

The runtime of the distinguisher is dominated by the runtime of the hinted CDH adversary t', since all it otherwise does is getting esk_A from the randomness of $\mathcal{A}$ as well as computing DH and KDF.

If the distinguisher got a sample from the real distribution, it answers correctly iff the hinted CDH adversary $\mathcal{A}_{\mathsf{HCDH}'}$ succeeds, i.e., with probability $> \varepsilon'$. Next, observe that the simulator runs in time $t_{\mathcal{S}} \approx t$ and, by the (t, ε, η)-hinted CDH assumption, can compute $\widetilde{\mathsf{DH}_1}$ with probability $\leq \varepsilon$. If the distinguisher got a sample from the simulated distribution, it answers correctly iff $\mathcal{A}_{\mathsf{HCDH}'}$ succeeds and the simulator fails to compute $\widetilde{\mathsf{DH}_1}$: If both the simulator and $\mathcal{A}_{\mathsf{HCDH}'}$ succeed, then the distinguisher falsely classifies the sample as real, which happens with a probability $\leq \varepsilon$. If only $\mathcal{A}_{\mathsf{HCDH}'}$ succeeds, the distinguisher correctly classifies the sample as simulated: Since the simulator does not know $\widetilde{\mathsf{DH}_1}$ it cannot learn the session key, which is a random oracle output on values including $\widetilde{\mathsf{DH}_1}$. This happens with probability $> \varepsilon' - \varepsilon$ All in all, the distinguisher succeeds with probability $> \varepsilon' - \varepsilon/2$.

The same proof strategy works for a malicious Bob: Embed the hinted CDH challenge as ephemeral, semi-static, or long-term key and sample the other two key pairs honestly (so that the distinguisher can learn those two secrets). If the challenge is embedded as Alice's ephemeral key (or as Bob's semi-static key, respectively), the distinguisher needs to solve three (or two, respectively) hinted CDH challenges, since these keys are used for more than one DH operation. In that case the distinguisher would need to run the hinted CDH adversary $\mathcal{A}_{\mathsf{HCDH}'}$ two or three times. We can account for that with $t_{\mathcal{D}} \approx 3t'$. Similarly, the winning probability now needs to take into account that the hinted CDH adversary is run more than once. To put it in a nutshell, we can embed the hinted public key as any key of Alice or Bob. Embedding the key in Alice's long-term, Bob's long-term, or Bob's ephemeral key results in an attack with the given parameters. When embedding the key in Alice's ephemeral or Bob's semi-static key, we may need to allow for a longer runtime of the distinguisher or worse advantage. □

C Omitted lemma

Lemma 11. *For $\varepsilon \in [0,1), \delta \in [0, 1-\varepsilon]$ we have $\varepsilon/(1-\delta) \leq \varepsilon + \delta$.*

Proof. The inequality $\varepsilon/(1-\delta) \leq \varepsilon + \delta$ can be rearranged to $0 \leq \delta - \varepsilon\delta - \delta^2 = \delta(1-\varepsilon-\delta)$. Clearly the inequality holds in the edge cases $\delta = 0$ and $\delta = 1-\varepsilon$. It also holds for all $\delta \in (0, 1-\varepsilon)$: because the function

$$\begin{aligned} f : \mathbb{R} &\to \mathbb{R} \\ \delta &\mapsto \delta - \varepsilon\delta - \delta^2 \end{aligned}$$

is continuous, there exists a $\delta \in (0, 1-\varepsilon)$ such that $f(\delta) > 0$ (this can be seen for example by observing that the derivative $f'(\delta) = 1 - \varepsilon - 2\delta$ is positive for $\delta = 0$), and $\delta \in \{0, 1-\varepsilon\}$ are the only roots of f. Under these conditions the intermediate value theorem guarantees $f(\delta) > 0$ for $\delta \in (0, 1-\varepsilon)$. □