

# More Efficient Digital Signatures with Tight Multi-User Security<sup>\*</sup>

Denis Diemert, Kai Gellert, Tibor Jager, and Lin Lyu

Bergische Universität Wuppertal, Germany

{denis.diemert, kai.gellert, tibor.jager, lin.lyu}@uni-wuppertal.de

**Abstract.** We construct the currently most efficient signature schemes with tight *multi-user* security against *adaptive* corruptions. It is the first *generic* construction of such schemes, based on lossy identification schemes (Abdalla et al.; JoC 2016), and the first to achieve *strong* existential unforgeability. It also has significantly more compact signatures than the previously most efficient construction by Gjøsteen and Jager (CRYPTO 2018). When instantiated based on the decisional Diffie–Hellman assumption, a signature consists of only three exponents.

We propose a new variant of the generic construction of signatures from *sequential OR-proofs* by Abe, Ohkubo, and Suzuki (ASIACRYPT 2002) and Fischlin, Harasser, and Janson (EUROCRYPT 2020). In comparison to Fischlin et al., who focus on constructing signatures in the non-programmable random oracle model (NPROM), we aim to achieve tight security against adaptive corruptions, maximize efficiency, and to directly achieve *strong* existential unforgeability (also in the NPROM). This yields a slightly different construction and we use slightly different and additional properties of the lossy identification scheme.

Signatures with tight multi-user security against adaptive corruptions are a commonly-used standard building block for tightly-secure authenticated key exchange protocols. We also show how our construction improves the efficiency of all existing tightly-secure AKE protocols.

## 1 Introduction

The commonly accepted standard security goal for digital signatures is *existential unforgeability under adaptive chosen message attacks* (EUF-CMA). This security model considers a *single-user* setting, in the sense that the adversary has access to a single public key and its goal is to forge a signature with respect to this key. A stronger security notion is EUF-CMA-security in the *multi-user* setting with *adaptive corruptions* (MU-EUF-CMA<sup>corr</sup>). In this security model, the adversary has access to multiple public keys, and it is allowed to adaptively *corrupt* certain users, and thus obtain their secret keys. The goal of the adversary is to forge a signature with respect to the public key of an *uncorrupted* user. A straightforward argument, which essentially guesses the user for which the

---

<sup>\*</sup> Supported by the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme, grant agreement 802823.

adversary creates a forgery at the beginning of the security experiment, shows that EUF-CMA security implies MU-EUF-CMA<sup>corr</sup> security. However, this guessing incurs a linear security loss in the number of users, and thus cannot achieve *tight* MU-EUF-CMA<sup>corr</sup> security.

The question how tightly MU-EUF-CMA<sup>corr</sup>-secure signatures can be constructed, and how efficient these constructions can be, is interesting for different reasons. Most importantly, MU-EUF-CMA<sup>corr</sup> security seems to reflect the security requirements of many applications that use digital signatures as building blocks more directly than EUF-CMA security. This holds in particular for many constructions of authenticated key exchange protocols (AKE) that use signing keys as long-term keys to authenticate protocol messages. Standard AKE security models, such as the well-known Bellare–Rogaway [8] or the Canetti–Krawczyk [10] model and their countless variants and refinements, allow for adaptive corruption of users, which then translates to adaptive corruptions of secret keys. Therefore Bader et al. [5] introduced the notion of MU-EUF-CMA<sup>corr</sup> as a building block to construct the first tightly-secure AKE protocol. This security model was subsequently used to construct more efficient tightly-secure AKE protocols [23, 28, 34], or to prove tight security of real-world protocols [14, 15]. Note that tight security is particularly interesting for AKE, due to the pervasive and large-scale use of such protocols in practice (e.g., the TLS Handshake is an AKE protocol). Furthermore, we consider the goal of understanding if, how, and how efficiently strong security notions for digital signatures such as MU-EUF-CMA<sup>corr</sup> can be achieved with tight security proofs also as a general and foundational research question in cryptography.

*The difficulty of constructing tightly MU-EUF-CMA<sup>corr</sup>-secure signatures.* The already mentioned straightforward reduction showing that EUF-CMA security implies MU-EUF-CMA<sup>corr</sup> security guesses the user for which the adversary creates a forgery. Note that this user must not be corrupted by a successful adversary. Hence, the reduction can define this user’s public key as the public key obtained from the EUF-CMA experiment. The keys of all users are generated by the reduction itself, such that it knows all corresponding secret keys. On the one hand, this enables the reduction to respond to all corruption queries made by the adversary, provided that it has guessed correctly. On the other hand, this makes the reduction lossy, since it may fail if the reduction did not guess correctly.

A reduction proving MU-EUF-CMA<sup>corr</sup> security *tightly* (under some complexity assumption) has to avoid such a guessing argument. However, note that this implies that the reduction must satisfy the following two properties simultaneously:

1. It has to know the secret keys of all users, in order to be able to respond to a corruption query for any user, without the need to guess uncorrupted users.
2. At the same time, the reduction has to be able to extract a solution to the underlying assumed-to-be-hard computational problem, while knowing the secret key of the corresponding instance of the signature scheme.

Since these two properties seem to contradict each other, one might think that tight  $\text{MU-EUF-CMA}^{\text{corr}}$  security is impossible to achieve. Indeed, one can even prove formally that  $\text{MU-EUF-CMA}^{\text{corr}}$  security is not tightly achievable [6] (under non-interactive assumptions<sup>1</sup>), however, this impossibility result holds only for signature schemes satisfying certain properties. While most schemes indeed satisfy these properties, and thus seem not able to achieve tight  $\text{MU-EUF-CMA}^{\text{corr}}$  security, there are some constructions that circumvent this impossibility result.

*Known constructions of tightly  $\text{MU-EUF-CMA}^{\text{corr}}$ -secure signatures.* To our best knowledge, there are only a few schemes with tight  $\text{MU-EUF-CMA}^{\text{corr}}$  security under non-interactive hardness assumptions (cf. Table 1). Bader et al. (BHJKL) [5] describe a scheme with constant security loss (“fully-tight”), but it uses the tree-based scheme from [27] as a building block and therefore has rather large signatures. The scheme is proven secure in the standard model, using pairings. Bader et al. also describe a second scheme with constant-size signatures, which is also based on pairings and in the standard model, but which has a linear security loss in the security parameter (“almost-tight”) and has linear-sized public keys. The currently most efficient tightly  $\text{MU-EUF-CMA}^{\text{corr}}$ -secure scheme is due to Gjøsteen and Jager (GJ) [23]. It has constant-size signatures and keys, as well as a constant security loss, in the random oracle model. The security proof requires “programming” of the random oracle in the sense of [19].

*Strong existential unforgeability.* Currently there exists no signature scheme with tight multi-user security under adaptive corruptions that achieves *strong* existential unforgeability. Here “strong” unforgeability refers to a security model where the adversary is considered to successfully break the security of a signature scheme, even if it outputs a *new* signature for a message for which it has already received a signature in the security experiment. Hence, strong unforgeability essentially guarantees that signatures additionally are “non-malleable”, in the sense that an adversary is not able to efficiently derive a new valid signature  $\sigma^*$  for a message  $m$  when it is already given another valid signature  $\sigma$  for  $m$ , where  $\sigma \neq \sigma^*$ .

Strong unforgeability is particularly useful for the construction of authenticated key exchange protocols where partnering is defined over “*matching conversations*”, as introduced by Bellare and Rogaway [8]. Intuitively, matching conversations formalize “authentication” for AKE protocols, by requiring that a communicating party must “accept” a protocol session (and thus derive a key for use in a higher-layer application protocol) only if there exists a unique partner oracle to which it has a matching conversation, that is, which has sent and received exactly the same sequence of messages that the accepting oracle has received and sent.

Consider for instance the “signed Diffie–Hellman” AKE protocol. Standard existential unforgeability of the signature scheme is not sufficient to achieve

<sup>1</sup> One can always prove tight  $\text{MU-EUF-CMA}^{\text{corr}}$  security under the interactive assumption that the scheme is  $\text{MU-EUF-CMA}^{\text{corr}}$  secure.

**Table 1.** Comparison of existing tightly-secure signature schemes in the multi-user setting with adaptive corruptions. “BHJKL 1” refers to the generic construction from [5] instantiated with the scheme from [27], “BHJKL 2” is the new scheme constructed in [5].  $|\sigma|$  indicates the size of a signature and  $|pk|$  the size of public keys, where  $|\mathbb{G}|$  is the size of an element of the underlying group  $\mathbb{G}$ ,  $|q|$  is the size of the binary representation of an integer in the discrete interval  $[0, q - 1]$ , where  $q$  is order of  $\mathbb{G}$ , and  $\lambda$  is the security parameter. The column “Setting” indicates whether pairings / the Programmable Random Oracle (PRO) model / the Non-Programmable Random Oracle (NPRO) model is used. The column “sEUF” refers to whether the scheme is proven *strongly* existentially unforgeable.

Scheme	$ \sigma $	$ pk $	Loss	Assumption	Setting	sEUF
BHJKL 1 [5, 27]	$\mathcal{O}(\lambda) \mathbb{G} $	$\mathcal{O}(1) \mathbb{G} $	$\mathcal{O}(1)$	DLIN	Pairings	–
BHJKL 2 [5]	$3 \mathbb{G} $	$\mathcal{O}(\lambda) \mathbb{G} $	$\mathcal{O}(\lambda)$	SXDH	Pairings	–
GJ [23]	$2 \mathbb{G}  + 2\lambda + 4 q $	$2 \mathbb{G} $	$\mathcal{O}(1)$	DDH	PRO	–
Ours	$3 q $	$4 \mathbb{G} $	$\mathcal{O}(1)$	Lossy ID	NPRO	✓

security in the sense of matching conversations, because this security notion does not guarantee that signatures are non-malleable. Hence, an adversary might, for instance, be able to efficiently re-randomize probabilistic signatures, and thus always be able to break matching conversations efficiently. This is a commonly overlooked mistake in many security proofs for AKE protocols [33]. Therefore Bader et al. [5] need to construct a more complex protocol that additionally requires strongly-unforgeable one-time signatures to achieve security in the sense of matching conversations. Gjøsteen and Jager [23] had to rely on the weaker partnering notion defined by Li and Schäge [33] in order to deal with potential malleability of signatures.

Hence, strongly-unforgeable digital signatures are particularly desirable in the context of AKE protocols, in order to achieve the strong notion of “matching conversation” security from [8].

*Our contributions.* We construct strongly MU-EUF-CMA<sup>corr</sup>-secure digital signature schemes, based on *lossy identification schemes* as defined by Abdalla et al. [2, 3] and *sequential OR-proofs* as considered by Abe et al. [4] and Fischlin et al. [18]. This construction provides the following properties:

- It is the first *generic* construction of MU-EUF-CMA<sup>corr</sup>-secure digital signatures, which can be instantiated from any concrete hardness assumption that gives rise to suitable lossy identification schemes. This includes instantiations from the decisional Diffie–Hellman (DDH) assumption, and the  $\phi$ -Hiding assumption.
- It is the first construction of MU-EUF-CMA<sup>corr</sup>-secure digital signatures that achieves *strong* existential unforgeability. Here we use “uniqueness” of the lossy identification scheme in the sense of [2, 3].
- When instantiated under the DDH assumption, a signature consists of only three elements of  $\mathbb{Z}_q$ , where  $q$  is the order of the underlying algebraic group.

For comparison, Schnorr signatures [36] and ECDSA [16], for instance, have signatures consisting of two elements of  $\mathbb{Z}_q$ , but do not enjoy tight security proofs (not even in the single-user setting) [17, 20–22, 35, 37]. In case of Schnorr signatures [36], security can be based on the weaker discrete logarithm assumption, though. Katz-Wang signatures [29] also consist of two  $\mathbb{Z}_q$ -elements and have tight security in the single-user setting, but not in the multi-user setting with adaptive corruptions.

- Similar to the work by Fischlin et al. [18], the proof does not rely on *programming* a random oracle, but holds in the *non-programmable* random oracle model [19]. This yields the first efficient and tightly multi-user secure signature scheme that does not require a programmable random oracle.

Our construction is almost identical to the construction based on sequential OR-proofs (as opposed to “parallel” OR-proofs in the sense of [13]), which was originally described by Abe et al. [4]. Fischlin, Harasser, and Janson [18] formally analyzed this construction and showed that it implies EUF-CMA-secure digital signatures based on lossy identification schemes. Their main focus is to achieve security in the non-programmable random oracle model [19], since the classical construction of signatures from lossy identification schemes [2, 3] requires a programmable random oracle.

We observe that this approach also gives rise to tightly-secure signatures in a multi-user model with adaptive corruptions, by slightly modifying the construction. Due to the fact that the reduction is always in possession of a correctly distributed secret key for all users, it can both (i) respond to signing-queries and (ii) respond to corruption-queries without the need to guess in the  $\text{MU-EUF-CMA}^{\text{corr}}$  security experiment.

Also, our security proof is based on slightly different and additional properties of the lossy identification scheme. We use that a sequential OR-proof is *perfectly* witness indistinguishable when both instances of the lossy identification scheme are in non-lossy mode. This enables us to argue that the adversary receives no information about the random bit  $b$  chosen by the key generation algorithm of one user, such that the probability that the adversary creates a forgery with respect to  $sk_{1-b}$  is  $1/2$ . This enables us then to construct a distinguisher for the lossy identification scheme with only constant security loss.

Another difference to the proof by Fischlin et al. [18] is that we directly achieve *strong* unforgeability by leveraging *uniqueness* of lossy identification schemes, as defined by Abdalla et al. [2, 3]. Also, their construction does not yet leverage “commitment-recoverability” of a lossy identification scheme, such that their DDH-based instantiation consists of four elements of  $\mathbb{Z}_q$ .

In particular, Table 2 shows that our scheme does not only improve the overall performance of all the presented protocols, but it also enables the protocols by GJ and LLGW to catch up to the communication complexity of JKRS. This means that when instantiated with our signature scheme, the constructions by GJ, LLGW, and JKRS achieve the same communication complexity. This observation suggests that especially constructions that exchange two or more signatures will benefit from an instantiation with our new signature scheme.

**Table 2.** Comparison of existing tightly-secure AKE protocols when instantiated with parameters for “128-bit security” (i.e.,  $\lambda = 128$ ). The columns **Comm.** count the values exchanged during execution of the protocol with an abstract signature scheme, when instantiated with the GJ signature scheme [23], and when instantiated with our DDH-based signature scheme respectively.  $\mathbb{G}$  is the number of group elements,  $H$  the number of hashes or MACs, Sig. the number of signatures,  $\mathbb{Z}_q$  the number of exponents, and “other” the amount of additional data in bits (nonces are  $2\lambda$ -bit strings). The columns **Bytes** contain the total amount of data in bytes when instantiating  $\mathbb{G}$  with the NIST P256 curve.

Protocol	Comm. ( $\mathbb{G}, H, \text{Sig.}, \text{other}$ )	With GJ Sigs.		With our scheme	
		Comm. ( $\mathbb{G}, H, \mathbb{Z}_q, \text{other}$ )	Bytes	Comm. ( $\mathbb{G}, H, \mathbb{Z}_q, \text{other}$ )	Bytes
GJ [23]	(2, 1, 2, 0)	(6, 1, 8, $4\lambda$ )	544	(2, 1, 6, 0)	288
TLS 1.3 [14, 15]	(2, 2, 2, 512)	(6, 2, 8, $4\lambda + 512$ )	640	(2, 2, 6, 512)	384
SIGMA-I [14, 32]	(2, 2, 2, 512)	(6, 2, 8, $4\lambda + 512$ )	640	(2, 2, 6, 512)	384
LLGW [34]	(3, 0, 2, 0)	(7, 0, 8, $4\lambda$ )	544	(3, 0, 6, 0)	288
JKRS [28]	(5, 1, 1, 0)	(7, 1, 4, $2\lambda$ )	416	(5, 1, 3, 0)	288

*Applications to tightly-secure AKE protocols.* Since tightly MU-EUF-CMA<sup>corr</sup>-secure signatures are commonly used to construct tightly-secure AKE protocols, let us consider the impact of our scheme on the performance of known protocols. Since the performance gain obtained by the signature scheme has already been discussed, we focus here only on the *communication complexity* of the considered protocols, that is, the number of bits exchanged when running the protocol. Table 2 shows the impact of our signature schemes on known AKE protocols with tight security proofs. We compare instantiations with the signature scheme by Gjsteen and Jager [23] to instantiations with our signature scheme. Note that the Gjsteen–Jager scheme is also based on the DDH assumption, and so are the considered protocols (except for TLS 1.3 and Sigma, which are based on the *strong* Diffie–Hellman assumption).

We omit the protocol by Bader et al. [5], since it is more of a standard-model feasibility result, which does not aim for maximal efficiency. Their protocol has a communication complexity of  $O(\lambda)$  group elements when instantiated with constant security loss, and 14 group elements plus 4 exponents when instantiated with their “almost-tight” signature scheme with a security loss of  $O(\lambda)$ . Cohn-Gordon et al. [12] construct a protocol which entirely avoids signatures and aims to achieve tightness, however, they achieve only a *linear* security loss and also show that this is optimal for the class of protocols they consider.

*Outline.* The remainder of this work is organized as follows. In the next section, we introduce standard definitions for signatures and their security. In Section 3, we recall lossy identification schemes and their security properties. The generic construction of our signature scheme from any lossy identification scheme alongside a security proof is presented in Section 4. We conclude our work with a detailed discussion on possible instantiations of our scheme in Section 5.

## 2 Preliminaries

For strings  $a$  and  $b$ , we denote the concatenation of these strings by  $a \parallel b$ . For an integer  $n \in \mathbb{N}$ , we denote the set of integers ranging from 1 to  $n$  by  $[n] := \{1, \dots, n\}$ . For a set  $X = \{x_1, x_2, \dots\}$ , we use  $(v_i)_{i \in X}$  as a shorthand for the tuple  $(v_{x_1}, v_{x_2}, \dots)$ . We denote the operation of assigning a value  $y$  to a variable  $x$  by  $x := y$ . If  $S$  is a finite set, we denote by  $x \stackrel{\$}{\leftarrow} S$  the operation of sampling a value uniformly at random from set  $S$  and assigning it to variable  $x$ .

### 2.1 Digital Signatures

We recall the standard definition of a *digital signature scheme* by Goldwasser, Micali, and Rivest [24] and its standard security notion.

**Definition 1.** A digital signature scheme is a triple of algorithms  $\text{Sig} = (\text{Gen}, \text{Sign}, \text{Vrfy})$  such that

1.  $\text{Gen}$  is the randomized key generation algorithm generating a public (verification) key  $pk$  and a secret (signing) key  $sk$ .
2.  $\text{Sign}(sk, m)$  is the randomized signing algorithm outputting a signature  $\sigma$  on input of a message  $m \in M$  and a signing key  $sk$ .
3.  $\text{Vrfy}(pk, m, \sigma)$  is the deterministic verification algorithm outputting either 0 or 1.

We say that a digital signature scheme  $\text{Sig}$  is  $\rho$ -correct if for  $(pk, sk) \stackrel{\$}{\leftarrow} \text{Gen}$ , and any  $m \in M$ , it holds that

$$\Pr[\text{Vrfy}(pk, m, \text{Sign}(sk, m)) = 1] \geq \rho.$$

And we say  $\text{Sig}$  is perfectly correct if it is 1-correct.

**Definition 2.** Let  $\text{Sig} = (\text{Gen}, \text{Sign}, \text{Vrfy})$  be a signature scheme and let  $N \in \mathbb{N}$  be the number of users. Consider the following experiment  $\text{Exp}_{\text{Sig}, N}^{\text{MU-sEUF-CMA}^{\text{corr}}}(\mathcal{A})$  played between a challenger and an adversary  $\mathcal{A}$ :

1. The challenger generates a key pair  $(pk^{(i)}, sk^{(i)}) \stackrel{\$}{\leftarrow} \text{Gen}$  for each user  $i \in [N]$ , initializes the set of corrupted users  $Q^{\text{corr}} := \emptyset$ , and  $N$  sets of chosen-message queries  $Q^{(1)}, \dots, Q^{(N)} := \emptyset$  issued by the adversary. Subsequently, it hands  $(pk^{(i)})_{i \in [N]}$  to  $\mathcal{A}$  as input.
2. The adversary may adaptively issue signature queries  $(i, m) \in [N] \times M$  to the challenger. The challenger replies to each query with a signature  $\sigma \stackrel{\$}{\leftarrow} \text{Sign}(sk^{(i)}, m)$  and adds  $(m, \sigma)$  to  $Q^{(i)}$ . Moreover, the adversary may adaptively issue corrupt queries  $\text{Corrupt}(i)$  for some  $i \in [N]$ . In this case, the challenger adds  $i$  to  $Q^{\text{corr}}$  and forwards  $sk^{(i)}$  to the adversary. We call each user  $i \in Q^{\text{corr}}$  corrupted.
3. Finally, the adversary outputs a tuple  $(i^*, m^*, \sigma^*)$ . The challenger checks whether  $\text{Vrfy}(pk^{(i^*)}, m^*, \sigma^*) = 1$ ,  $i^* \notin Q^{\text{corr}}$  and  $(m^*, \sigma^*) \notin Q^{(i^*)}$ . If all of these conditions hold, the experiment outputs 1 and 0 otherwise.

We denote the advantage of an adversary  $\mathcal{A}$  in breaking the strong existential unforgeability under an adaptive chosen-message attack in the multi-user setting with adaptive corruptions ( $\text{MU-sEUF-CMA}^{\text{corr}}$ ) for  $\text{Sig}$  by

$$\text{Adv}_{\text{Sig}, N}^{\text{MU-sEUF-CMA}^{\text{corr}}}(\mathcal{A}) := \Pr \left[ \text{Exp}_{\text{Sig}, N}^{\text{MU-sEUF-CMA}^{\text{corr}}}(\mathcal{A}) = 1 \right]$$

where  $\text{Exp}_{\text{Sig}, N}^{\text{MU-sEUF-CMA}^{\text{corr}}}(\mathcal{A})$  is as defined as above.

### 3 Lossy Identification Schemes

We adapt the definitions of a *lossy identification scheme* [2, 3, 30].

**Definition 3.** A lossy identification scheme is a five-tuple  $\text{LID} = (\text{LID.Gen}, \text{LID.LossyGen}, \text{LID.Prove}, \text{LID.Vrfy}, \text{LID.Sim})$  of probabilistic polynomial-time algorithms with the following properties.

- $\text{LID.Gen}$  is the normal key generation algorithm. It outputs a public verification key  $pk$  and a secret key  $sk$ .
- $\text{LID.LossyGen}$  is a lossy key generation algorithm that takes the security parameter and outputs a lossy verification key  $pk$ .
- $\text{LID.Prove}$  is the prover algorithm that is split into two algorithms:
  - $(\text{cmt}, \text{st}) \xleftarrow{\$} \text{LID.Prove}_1(sk)$  is a probabilistic algorithm that takes as input the secret key and returns a commitment  $\text{cmt}$  and a state  $\text{st}$ .
  - $\text{resp} \leftarrow \text{LID.Prove}_2(sk, \text{cmt}, \text{ch}, \text{st})$  is a deterministic algorithm<sup>2</sup> that takes as input a secret key  $sk$ , a commitment  $\text{cmt}$ , a challenge  $\text{ch}$ , a state  $\text{st}$ , and returns a response  $\text{resp}$ .
- $\text{LID.Vrfy}(pk, \text{cmt}, \text{ch}, \text{resp})$  is a deterministic verification algorithm that takes a public key, and a conversation transcript (i.e., a commitment, a challenge, and a response) as input and outputs a bit, where 1 indicates that the proof is “accepted” and 0 that it is “rejected”.
- $\text{cmt} \leftarrow \text{LID.Sim}(pk, \text{ch}, \text{resp})$  is a deterministic algorithm that takes a public key  $pk$ , a challenge  $\text{ch}$ , and a response  $\text{resp}$  as inputs and outputs a commitment  $\text{cmt}$ .

We assume that a public key  $pk$  implicitly defines two sets, the set of challenges  $\text{CSet}$  and the set of responses  $\text{RSet}$ .

**Definition 4.** Let  $\text{LID} = (\text{LID.Gen}, \text{LID.LossyGen}, \text{LID.Prove}, \text{LID.Vrfy}, \text{LID.Sim})$  be defined as above. We call  $\text{LID}$  lossy when the following properties hold:

<sup>2</sup> All known instantiations of lossy identification schemes have a deterministic  $\text{LID.Prove}_2$  algorithm. However, if a new instantiation requires randomness, then it can be “forwarded” from  $\text{LID.Prove}_1$  in the state variable  $\text{st}$ . Therefore the requirement that  $\text{LID.Prove}_2$  is deterministic is without loss of generality, and only made to simplify our security analysis.

- Completeness of normal keys. We call LID  $\rho$ -complete, if

$$\Pr \left[ \begin{array}{l} (pk, sk) \xleftarrow{\$} \text{LID.Gen} \\ \text{LID.Vrfy}(pk, \text{cmt}, \text{ch}, \text{resp}) = 1 : \begin{array}{l} (\text{cmt}, \text{st}) \xleftarrow{\$} \text{LID.Prove}_1(sk) \\ \text{ch} \xleftarrow{\$} \text{CSet} \\ \text{resp} \xleftarrow{\$} \text{LID.Prove}_2(sk, \text{cmt}, \text{ch}, \text{st}) \end{array} \end{array} \right] \geq \rho.$$

We call LID perfectly-complete, if it is 1-complete.

- Simulatability of transcripts. We call LID  $\varepsilon_s$ -simulatable if for  $(pk, sk) \xleftarrow{\$} \text{LID.Gen}$ ,  $(\text{ch}, \text{resp}) \xleftarrow{\$} \text{CSet} \times \text{RSet}$ , the distribution of the transcript  $(\text{cmt}, \text{ch}, \text{resp})$  where  $\text{cmt} \leftarrow \text{LID.Sim}(pk, \text{ch}, \text{resp})$  is statistically indistinguishable from honestly generated transcript (with a statistical distance up to  $\varepsilon_s$ ) and we have that  $\text{LID.Vrfy}(pk, \text{cmt}, \text{ch}, \text{resp}) = 1$ . If  $\varepsilon_s = 0$ , we call LID perfectly simulatable.

Note that this simulatability property is different from the original definition in [2] where the simulator simulates the whole transcript.

- Indistinguishability of keys. This definition is a generalization of the standard key indistinguishability definition of a lossy identification scheme extended to  $N$  instances. For any integer  $N > 0$ , we define the advantage of an adversary  $\mathcal{A}$  breaking the  $N$ -key-indistinguishability of LID as  $\text{Adv}_{\text{LID}, N}^{\text{MU-IND-KEY}}(\mathcal{A}) :=$

$$\left| \Pr \left[ \mathcal{A}(pk^{(1)}, \dots, pk^{(N)}) = 1 \right] - \Pr \left[ \mathcal{A}(pk'^{(1)}, \dots, pk'^{(N)}) = 1 \right] \right|,$$

where  $(pk^{(i)}, sk^{(i)}) \xleftarrow{\$} \text{LID.Gen}$  and  $pk'^{(i)} \xleftarrow{\$} \text{LID.LossyGen}$  for all  $i \in [N]$ .

- Lossiness. Consider the following security experiment  $\text{Exp}_{\text{LID}}^{\text{IMPERSONATE}}(\mathcal{A})$  described below, played between a challenger and an adversary  $\mathcal{A}$ :

1. The challenger generates a lossy verification key  $pk \xleftarrow{\$} \text{LID.LossyGen}$  and sends it to the adversary  $\mathcal{A}$ .
2. The adversary  $\mathcal{A}$  may now compute a commitment  $\text{cmt}$  and send it to the challenger. The challenger responds with a random challenge  $\text{ch} \xleftarrow{\$} \text{CSet}$ .
3. Eventually, the adversary  $\mathcal{A}$  outputs a response  $\text{resp}$ . The challenger outputs  $\text{LID.Vrfy}(pk, \text{cmt}, \text{ch}, \text{resp})$ .

We call LID  $\varepsilon_\ell$ -lossy if no computationally unrestricted adversary  $\mathcal{A}$  wins the above security game with probability

$$\Pr[\text{Exp}_{\text{LID}}^{\text{IMPERSONATE}}(\mathcal{A}) = 1] \geq \varepsilon_\ell.$$

Below are two more properties for lossy identification schemes defined in [2,3].

**Definition 5.** Let  $pk \xleftarrow{\$} \text{LID.LossyGen}$  be a lossy public key and let  $(\text{cmt}, \text{ch}, \text{resp})$  be any transcript which makes  $\text{LID.Vrfy}(pk, \text{cmt}, \text{ch}, \text{resp}) = 1$ . We say LID is  $\varepsilon_u$ -unique with respect to lossy keys if the probability that there exists  $\text{resp}' \neq \text{resp}$  such that  $\text{LID.Vrfy}(pk, \text{cmt}, \text{ch}, \text{resp}') = 1$  is at most  $\varepsilon_u$ , and perfectly unique with respect to lossy keys if  $\varepsilon_u = 0$ .

**Definition 6.** Let  $(pk, sk) \xleftarrow{\$} \text{LID.Gen}$  be any honestly generated key pair and  $\mathcal{C}(sk) := \{\text{LID.Prove}_1(sk)\}$  be the set of commitments associated to  $sk$ . We define the min-entropy with respect to LID as

$$\alpha := -\log_2 \left( \max_{sk, \text{cmt} \in \mathcal{C}(sk)} \Pr[\text{LID.Prove}_1(sk) = \text{cmt}] \right)$$

Below is another property for lossy identification schemes defined in [30].

**Definition 7.** *A lossy identification scheme LID is commitment-recoverable if the algorithm  $\text{LID.Vrfy}(pk, \text{cmt}, \text{ch}, \text{resp})$  first recomputes a commitment  $\text{cmt}' = \text{LID.Sim}(pk, \text{ch}, \text{resp})$  and then outputs 1 if and only if  $\text{cmt}' = \text{cmt}$ .*

Below, we define a new property for lossy identification schemes which requires that the  $\text{LID.Sim}$  algorithm is injective with respect to the input challenge.

**Definition 8.** *A lossy identification scheme LID has an injective simulator if for any  $(pk, sk) \xleftarrow{\$} \text{LID.Gen}$ , any response  $\text{resp} \in \text{RSet}$ , any  $\text{ch} \neq \text{ch}'$ , it holds that  $\text{LID.Sim}(pk, \text{ch}, \text{resp}) \neq \text{LID.Sim}(pk, \text{ch}', \text{resp})$ .*

In Section 5 we give a detailed discussion which of the existing lossy identification schemes [1–3, 26, 29] satisfy which of the above properties.

## 4 Construction and Security of our Signature Scheme

Let  $\text{LID} = (\text{LID.Gen}, \text{LID.LossyGen}, \text{LID.Prove}, \text{LID.Vrfy}, \text{LID.Sim})$  be a lossy identification scheme and let  $\text{H}: \{0, 1\}^* \rightarrow \text{CSet}$  be a hash function mapping finite-length bitstrings to the set of challenges  $\text{CSet}$ . Consider the following digital signature scheme  $\text{Sig} = (\text{Gen}, \text{Sign}, \text{Vrfy})$ .

**Key generation.** The key generation algorithm  $\text{Gen}$  samples a bit  $b \xleftarrow{\$} \{0, 1\}$  and two independent key pairs  $(pk_0, sk_0) \xleftarrow{\$} \text{LID.Gen}$  and  $(pk_1, sk_1) \xleftarrow{\$} \text{LID.Gen}$ . Then it sets

$$pk := (pk_0, pk_1) \quad \text{and} \quad sk := (b, sk_b)$$

Note that the secret key consists only of  $sk_b$  and the other key  $sk_{1-b}$  is discarded.

**Signing.** The signing algorithm  $\text{Sign}$  takes as input  $sk = (b, sk_b)$  and a message  $m \in \{0, 1\}^*$ . Then it proceeds as follows.

1. It first computes  $(\text{cmt}_b, \text{st}_b) \xleftarrow{\$} \text{LID.Prove}_1(sk_b)$  and sets

$$\text{ch}_{1-b} := \text{H}(m, \text{cmt}_b)$$

Note that the  $\text{ch}_{1-b}$  is derived from  $\text{cmt}_b$  and  $m$ .

2. It generates the simulated transcript by choosing  $\text{resp}_{1-b} \xleftarrow{\$} \text{RSet}$  and

$$\text{cmt}_{1-b} := \text{LID.Sim}(pk_{1-b}, \text{ch}_{1-b}, \text{resp}_{1-b})$$

using the simulator.

3. Finally, it computes

$$\text{ch}_b := \text{H}(m, \text{cmt}_{1-b}) \quad \text{and} \quad \text{resp}_b := \text{LID.Prove}_2(sk_b, \text{ch}_b, \text{cmt}_b, \text{st}_b)$$

and outputs the signature  $\sigma := (\text{ch}_0, \text{resp}_0, \text{resp}_1)$ . Note that  $\text{ch}_1$  is not included in the signature.

**Verification.** The verification algorithm  $\text{Vrfy}$  takes as input a public key  $pk = (pk_0, pk_1)$ , a message  $m \in \{0, 1\}^*$ , and a signature  $\sigma = (\text{ch}_0, \text{resp}_0, \text{resp}_1)$ . It first recovers

$$\text{cmt}_0 := \text{LID.Sim}(pk_0, \text{ch}_0, \text{resp}_0)$$

From  $\text{cmt}_0$  it can then compute

$$\text{ch}_1 := \text{H}(m, \text{cmt}_0)$$

and then recover

$$\text{cmt}_1 := \text{LID.Sim}(pk_1, \text{ch}_1, \text{resp}_1)$$

Finally, the reduction outputs 1 if and only if  $\text{ch}_0 = \text{H}(m, \text{cmt}_1)$ .

One can easily verify that the above construction  $\text{Sig}$  is perfectly correct if  $\text{LID}$  is commitment-recoverable and perfectly complete. Also, note that, even though algorithm  $\text{LID.Vrfy}$  is not used in algorithm  $\text{Vrfy}$ , we have that  $\text{Vrfy}(pk, m, \sigma) = 1$  implies that  $\text{LID.Vrfy}(pk_j, \text{cmt}_j, \text{ch}_j, \text{resp}_j) = 1$  for both  $j \in \{0, 1\}$ . This is directly implied by our definition of the lossy identification scheme's simulatability of transcripts.

**Theorem 9.** *If  $\text{H}$  is modeled as a random oracle and  $\text{LID}$  is commitment-recoverable, perfectly simulatable,  $\varepsilon_\ell$ -lossy,  $\varepsilon_u$ -unique, has  $\alpha$ -bit min-entropy and has an injective simulator, then for each adversary  $\mathcal{A}$  with running time  $t_{\mathcal{A}}$  breaking the  $\text{MU-sEUF-CMA}^{\text{corr}}$  security of the above signature scheme  $\text{Sig}$ , we can construct an adversary  $\mathcal{B}$  with running time  $t_{\mathcal{B}} \approx t_{\mathcal{A}}$  such that*

$$\text{Adv}_{\text{Sig}, N}^{\text{MU-sEUF-CMA}^{\text{corr}}}(\mathcal{A}) \leq 4 \cdot \text{Adv}_{\text{LID}, N}^{\text{MU-IND-KEY}}(\mathcal{B}) + \frac{2q_S q_H}{2^\alpha} + \frac{2}{|\text{CSet}|} + 2\varepsilon_u + 2Nq_H^2\varepsilon_\ell,$$

where  $q_S$  is the number of signing queries and  $q_H$  is the number of hash queries.

*Proof.* We prove Theorem 9 through a sequence of games. See Table 3 for an intuitive overview of our proof. In the sequel, let  $X_i$  denote the event that the experiment outputs 1 in Game  $i$ .

*Game 0.* This is the original security experiment  $\text{Exp}_{\text{Sig}, N}^{\text{MU-sEUF-CMA}^{\text{corr}}}(\mathcal{A})$ . In this experiment, adversary  $\mathcal{A}$  is provided with oracles  $\text{Sign}$  and  $\text{Corrupt}$  from the security experiment, as well as a hash oracle  $\text{H}$  since we are working in the random oracle model. In the following, it will be useful to specify the implementation of this game explicitly:

- The game initializes the chosen-message sets  $\mathcal{Q}^{(1)}, \dots, \mathcal{Q}^{(N)} := \emptyset$ , the set of corrupted users  $\mathcal{Q}^{\text{corr}} := \emptyset$  and an implementation of the random oracle  $\mathcal{L} := \emptyset$ . It then runs the signature key generation algorithm  $\text{Gen}$   $N$  times to get the key pair  $(pk^{(i)}, sk^{(i)})$  for each  $i \in [N]$ . More precisely, the game samples a bit  $b^{(i)} \xleftarrow{\$} \{0, 1\}$  and two independent key pairs  $(pk_0^{(i)}, sk_0^{(i)}) \xleftarrow{\$} \text{LID.Gen}$  and  $(pk_1^{(i)}, sk_1^{(i)}) \xleftarrow{\$} \text{LID.Gen}$ . Then it sets  $pk^{(i)} := (pk_0^{(i)}, pk_1^{(i)})$  and stores  $(pk^{(i)}, b^{(i)}, sk_0^{(i)}, sk_1^{(i)})$ . Finally, it runs adversary  $\mathcal{A}$  with input  $(pk^{(i)})_{i \in [N]}$ . In the following proof, to simplify the notation, we will use  $pk_b^{(i)}, pk_{1-b}^{(i)}, sk_b^{(i)}, sk_{1-b}^{(i)}$  to denote  $pk_{b^{(i)}}^{(i)}, pk_{1-b^{(i)}}^{(i)}, sk_{b^{(i)}}^{(i)}, sk_{1-b^{(i)}}^{(i)}$  respectively.

**Table 3.** Overview of the sequence of games used in the proof of Theorem 9.

Game #	Changes	Remark
0	–	The MU-sEUF-CMA <sup>corr</sup> game.
1	We rule out repeating commitments <code>cmt</code> .	This ensures that every signing query makes fresh hash queries.
2	We ensure the two hash queries in the final verification have been made before.	We will need this in Game 4.
3	We exclude the case where $(\text{cmt}_0^*, \text{cmt}_1^*)$ is re-used from a signing query.	The adversary does not use “implicit” knowledge of the secret bit $b^{(i^*)}$ .
4	The adversary can only win if hash query “ $(1 - b^{(i^*)})$ ” is made first.	$b^{(i^*)}$ is perfectly hidden, preparation to achieve statistically small winning probability.
5	We make all “ $(1 - b^{(i)})$ ” public keys lossy.	This game has statistically small winning probability for any adversary.

- $H(x)$ . When the adversary or the simulation of the experiment make a hash oracle query for some  $x \in \{0, 1\}^*$ , the game checks whether  $(x, y) \in \mathcal{L}$  for some  $y \in \text{CSet}$ . If it does, the game returns  $y$ . Otherwise the game selects  $y \xleftarrow{\$} \text{CSet}$ , logs  $(x, y)$  into set  $\mathcal{L}$  and returns  $y$ .
- $\text{Sign}(i, m)$ . When the adversary queries the signing oracle with user  $i$  and message  $m$ , the game first sets  $b := b^{(i)}$ , then computes

$$(\text{cmt}_b, \text{st}_b) \xleftarrow{\$} \text{LID.Prove}_1(sk_b^{(i)})$$

and sets  $\text{ch}_{1-b} := H(m, \text{cmt}_b)$  by making a hash query. Then, the game chooses  $\text{resp}_{1-b} \xleftarrow{\$} \text{RSet}$  and uses the simulator to compute  $\text{cmt}_{1-b} := \text{LID.Sim}(pk_{1-b}^{(i)}, \text{ch}_{1-b}, \text{resp}_{1-b})$ . Finally, the game queries hash oracle to get  $\text{ch}_b := H(m, \text{cmt}_{1-b})$  and then uses  $\text{LID.Prove}_2$  to compute

$$\text{resp}_b := \text{LID.Prove}_2(sk_b^{(i)}, \text{ch}_b, \text{cmt}_b, \text{st}_b).$$

The game outputs signature  $\sigma := (\text{ch}_0, \text{resp}_0, \text{resp}_1)$  to  $\mathcal{A}$  and logs the pair  $(m, \sigma)$  in set  $\mathcal{Q}^{(i)}$ .

- $\text{Corrupt}(i)$ . When the adversary  $\mathcal{A}$  queries the `Corrupt` oracle for the secret key of user  $i$ , the game returns  $sk^{(i)} := (b^{(i)}, sk_b^{(i)})$  to the adversary and logs  $i$  in the set  $\mathcal{Q}^{\text{corr}}$ .
- Finally, when adversary  $\mathcal{A}$  outputs a forgery attempt  $(i^*, m^*, \sigma^*)$ , the game outputs 1 if and only if  $\text{Vrfy}(pk^{(i^*)}, m^*, \sigma^*) = 1$ ,  $i^* \notin \mathcal{Q}^{\text{corr}}$ , and  $(m^*, \sigma^*) \notin \mathcal{Q}^{(i^*)}$  hold. More precisely, for  $\sigma^* = (\text{ch}_0^*, \text{resp}_0^*, \text{resp}_1^*)$ , the game recovers  $\text{cmt}_0^* := \text{LID.Sim}(pk_0^{(i^*)}, \text{ch}_0^*, \text{resp}_0^*)$  and queries the hash oracle to get  $\text{ch}_1^* := H(m^*, \text{cmt}_0^*)$ . Then it recovers  $\text{cmt}_1^* := \text{LID.Sim}(pk_1^{(i^*)}, \text{ch}_1^*, \text{resp}_1^*)$  and queries the hash oracle to get  $\text{ch}^* := H(m^*, \text{cmt}_1^*)$ . Finally, the game outputs 1 if and only if  $\text{ch}_0^* = \text{ch}^*$ ,  $i^* \notin \mathcal{Q}^{\text{corr}}$  and  $(m^*, \sigma^*) \notin \mathcal{Q}^{(i^*)}$ .

It is clear that  $\Pr[X_0] = \text{Adv}_{\text{Sig}, N}^{\text{MU-sEUF-CMA}^{\text{corr}}}(\mathcal{A})$ .

*Game 1.* Game 1 is the same with Game 0 except with one change. Denote with  $\text{cmtColl}$  the event that there exists a signing query  $\text{Sign}(i, m)$  such that at least one of the two hash queries  $H(m, \text{cmt}_{b^{(i)}})$  and  $H(m, \text{cmt}_{1-b^{(i)}})$  made in this signing query has been made before.

Game 1 outputs 0 when  $\text{cmtColl}$  happens. In other words,  $X_1$  happens if and only if  $X_0 \wedge \neg \text{cmtColl}$  happens. We can prove the following lemma.

**Lemma 10.**

$$\Pr[X_1] \geq \Pr[X_0] - \frac{2q_S q_H}{2^\alpha}$$

where  $q_S$  is the number of signing queries made by  $\mathcal{A}$  and  $q_H$  is the number of hash queries made in Game 0.

*Proof.* To prove Lemma 10, we divide the event  $\text{cmtColl}$  into two subevents.

- There exists a signing query  $\text{Sign}(i, m)$  such that  $H(m, \text{cmt}_{b^{(i)}})$  has been made before. If this happens, then  $\text{cmt}_{b^{(i)}}$  is the output of  $\text{LID.Prove}_1(sk^{(i)})$  for any signing query. Since LID has  $\alpha$ -bit min-entropy (cf. Definition 6), the probability that this happens is at most  $q_S q_H / 2^\alpha$  by a union bound.
- There exists a signing query  $\text{Sign}(i, m)$  such that  $H(m, \text{cmt}_{1-b^{(i)}})$  has been made before. Note that  $\text{cmt}_{1-b^{(i)}}$  is the output of

$$\text{LID.Sim}(pk_{1-b}^{(i)}, \text{ch}_{1-b^{(i)}}, \text{resp}_{1-b^{(i)}})$$

where  $\text{ch}_{1-b^{(i)}} = H(m, \text{cmt}_{b^{(i)}})$ . Since LID.Sim is deterministic, we know that  $\text{cmt}_{1-b^{(i)}}$  is determined by  $pk_{1-b}^{(i)}, m, \text{cmt}_{b^{(i)}}$  and  $\text{resp}_{1-b^{(i)}}$ . Furthermore, since LID.Sim is injective with respect to challenges (cf. Definition 8), we know that the entropy of  $\text{cmt}_{1-b^{(i)}}$  in any fixed signing query is at least the entropy of  $\text{cmt}_{b^{(i)}}$  in that query. Thus, we obtain that the probability that this subevent happens is at most  $q_S q_H / 2^\alpha$ .

Thus, we have that  $\Pr[\text{cmtColl}] \leq 2q_S q_H / 2^\alpha$  and Lemma 10 follows.  $\square$

*Remark 11.* Note that, from Game 1 on, the hash queries  $H(m, \text{cmt}_{b^{(i)}})$  and  $H(m, \text{cmt}_{1-b^{(i)}})$  are not made before any signing query  $\text{Sign}(i, m)$  if the game finally outputs 1. This implies that each signing query uses independent and uniformly random  $\text{ch}_{1-b^{(i)}}$  and  $\text{ch}_{b^{(i)}}$ , and they are not known to the adversary at that time.

*Game 2.* Game 2 differs from Game 1 only in the way the game checks the winning condition. More precisely, Game 1 issues two hash queries  $H(m^*, \text{cmt}_0^*)$  and  $H(m^*, \text{cmt}_1^*)$  to check the validity of a forgery attempt  $(i^*, m^*, \sigma^*)$ . In the following, we call the former  $H(m^*, \text{cmt}_0^*)$  a “0-query” and the latter  $H(m^*, \text{cmt}_1^*)$  a “1-query”. Let **Both** denote the event that both a 0-query and a 1-query have been made by the signing oracle or by the adversary before submitting the forgery attempt  $(i^*, m^*, \sigma^*)$ .

Game 2 outputs 0 if event **Both** does *not* happen. In other words,  $X_2$  happens if and only if  $X_1 \wedge \text{Both}$  happens. We can prove the following lemma.

**Lemma 12.**  $\Pr[X_2] \geq \Pr[X_1] - 2/|\text{CSet}|$ .

*Proof.* We know that  $\Pr[X_1] = \Pr[X_1 \wedge \neg\text{Both}] + \Pr[X_2]$ . We will prove that  $\Pr[X_1 \wedge \neg\text{Both}] \leq 2/|\text{CSet}|$  and the lemma follows. Note that

$$\Pr[X_1 \wedge \neg\text{Both}] \leq \Pr[X_1 \wedge \text{1-query is never made}] + \Pr[X_1 \wedge \text{0-query is never made}]$$

- $X_1 \wedge \text{1-query is never made}$ : Event  $X_1$  implies that  $\text{Vrfy}(pk^{(i^*)}, m^*, \sigma^*) = 1$ . This further implies that the value  $\text{ch}_0^*$  (chosen by the adversary) equals the 1-query hash result  $\text{ch}^* = \text{H}(m^*, \text{cmt}_1^*)$ , which is a random element in  $\text{CSet}$ . Since the 1-query is never made at this time, the adversary has no knowledge about this value, which yields

$$\Pr[X_1 \wedge \text{1-query is never made}] \leq \frac{1}{|\text{CSet}|}.$$

- $X_1 \wedge \text{0-query is never made}$ : The 0-query value  $\text{ch}_1^* = \text{H}(m^*, \text{cmt}_0^*)$  is used to recover  $\text{cmt}_1^* = \text{LID.Sim}(pk_1^{(i^*)}, \text{ch}_1^*, \text{resp}_1^*)$ . Since the 0-query is not made at that time, the adversary has no knowledge about  $\text{ch}_1^*$  except that it is a random element in  $\text{CSet}$ . Together with the fact that algorithm  $\text{LID.Sim}$  is injective (cf. Definition 8), the adversary only knows that  $\text{cmt}_1^*$  is uniformly distributed over a set of size  $|\text{CSet}|$ . To make the verification pass, the adversary would need to select  $\text{ch}_0^*$  which equals to  $\text{H}(m^*, \text{cmt}_1^*)$ . However, there are  $|\text{CSet}|$  possible values for  $\text{cmt}_1^*$  so that this can happen with probability at most  $1/|\text{CSet}|$ . Thus,

$$\Pr[X_1 \wedge \text{0-query is never made}] \leq \frac{1}{|\text{CSet}|}.$$

Putting both together, we have  $\Pr[X_1 \wedge \neg\text{Both}] \leq 2/|\text{CSet}|$ . □

*Game 3.* Game 3 is exactly the same as Game 2, except for one change. We denote by  $\text{ImplicitUsage}$  the event that the first 0-query and the first 1-query are made in a signing query  $\text{Sign}(i^*, m^*)$ , and the pair  $(\text{cmt}_0^*, \text{cmt}_1^*)$  equals to the pair  $(\text{cmt}_0, \text{cmt}_1)$ , which is generated in this signing query. Game 3 outputs 0 if event  $\text{ImplicitUsage}$  happens.

Hence,  $X_3$  happens if and only if  $X_2 \wedge \neg\text{ImplicitUsage}$  happens. We prove the following lemma.

**Lemma 13.** *We can construct an adversary  $\mathcal{B}$  with running time  $t_{\mathcal{B}} \approx t_{\mathcal{A}}$  such that*

$$\Pr[X_3] \geq \Pr[X_2] - 2 \cdot \text{Adv}_{\text{LID}, N}^{\text{MU-IND-KEY}}(\mathcal{B}) - 2\varepsilon_u.$$

*Remark 14.* Note that this proof can be potentially simplified if we define the uniqueness property of LID with respect to normal public keys. However, this would introduce a non-standard LID property compared to the standard LID definition by Abdalla et al. [2, 3].

*Proof.* We know that  $\Pr[X_2] = \Pr[X_2 \wedge \text{ImplicitUsage}] + \Pr[X_3]$ . We will prove that  $\Pr[X_2 \wedge \text{ImplicitUsage}] \leq 2\text{Adv}_{\text{LID}, N}^{\text{MU-IND-KEY}}(\mathcal{B}) + 2\varepsilon_u$  such that the above lemma follows.

Note that  $\text{ImplicitUsage}$  implies that

$$\text{ch}_j = \text{H}(m^*, \text{cmt}_{1-j}) = \text{H}(m^*, \text{cmt}_{1-j}^*) = \text{ch}_j^*$$

for  $j \in \{0, 1\}$ . Together with the fact that  $(m^*, \sigma^*) \notin \mathcal{Q}^{(i^*)}$ , we must have that  $(\text{resp}_0^*, \text{resp}_1^*) \neq (\text{resp}_0, \text{resp}_1)$ . Then two subcases are possible.

- $X_2 \wedge \text{ImplicitUsage} \wedge (\text{resp}_{1-b(i^*)}^* = \text{resp}_{1-b(i^*)}) \wedge (\text{resp}_{b(i^*)}^* \neq \text{resp}_{b(i^*)})$ . This subcase intuitively implies that the adversary successfully guesses the bit  $b(i^*)$ , since the adversary has to choose  $\text{resp}_0^*, \text{resp}_1^*$  such that  $\text{resp}_{1-b(i^*)}^*$  is equal and  $\text{resp}_{b(i^*)}^*$  is unequal. However, in Game 2, the secret bit  $b(i^*)$  is perfectly hidden to the adversary due to the following facts.
  - The public key  $pk^{(i^*)}$  is independent of  $b(i^*)$ .
  - User  $i^*$  is not corrupted (or otherwise the forgery is invalid, anyway), so the bit  $b(i^*)$  is not leaked through corruptions.
  - The signature  $\sigma$  returned by oracle  $\text{Sign}(i^*, m)$  is independent of bit  $b(i^*)$ . The reason is that  $X_2$  implies that  $\text{cmtColl}$  does not happen. As shown in Remark 11, each  $\text{Sign}(i^*, m)$  query will use uniformly random  $\text{ch}_{1-b(i^*)}$  and  $\text{ch}_{b(i^*)}$ . Thus, the signature essentially contains the two transcripts

$$(\text{cmt}_{b(i^*)}, \text{ch}_{b(i^*)}, \text{resp}_{b(i^*)}) \quad \text{and} \quad (\text{cmt}_{1-b(i^*)}, \text{ch}_{1-b(i^*)}, \text{resp}_{1-b(i^*)})$$

Note that the  $b(i^*)$  transcript is an “honestly generated” transcript and the  $(1-b(i^*))$  transcript is a “simulated” transcript with uniformly random  $\text{ch}_{1-b(i^*)}$  and  $\text{resp}_{1-b(i^*)}$ . Due to the perfect simulatability of LID, we know that these two transcripts are perfectly identically distributed.

Thus,  $\mathcal{A}$  gains no information about  $b(i^*)$  through signatures.

In summary, we conclude that this subcase happens with probability

$$\frac{1}{2} \Pr[X_2 \wedge \text{ImplicitUsage}].$$

- $X_2 \wedge \text{ImplicitUsage} \wedge (\text{resp}_{1-b(i^*)}^* \neq \text{resp}_{1-b(i^*)})$ . For this subcase, we can prove the following claim.

*Claim.* We can construct an adversary  $\mathcal{B}$  with running time  $t_{\mathcal{B}} \approx t_{\mathcal{A}}$  such that

$$\Pr[X_2 \wedge \text{ImplicitUsage} \wedge (\text{resp}_{1-b(i^*)}^* \neq \text{resp}_{1-b(i^*)})] \leq \text{Adv}_{\text{LID}, N}^{\text{MU-IND-KEY}}(\mathcal{B}) + \varepsilon_u.$$

*Proof.* To prove this claim, we define a new intermediate game Game 2', which is exactly the same as Game 2, except that we choose a lossy public key  $pk_{1-b}^{(i)} \xleftarrow{\$} \text{LID.LossyGen}$  for every user  $i \in [N]$  in Game 2'. We can build an adversary  $\mathcal{B}$  with running time  $t_{\mathcal{B}} \approx t_{\mathcal{A}}$  such that

$$\left| \begin{array}{l} \Pr[X_2 \wedge \text{ImplicitUsage} \wedge (\text{resp}_{1-b(i^*)}^* \neq \text{resp}_{1-b(i^*)})] \\ - \Pr[X_{2'} \wedge \text{ImplicitUsage} \wedge (\text{resp}_{1-b(i^*)}^* \neq \text{resp}_{1-b(i^*)})] \end{array} \right| \leq \text{Adv}_{\text{LID}, N}^{\text{MU-IND-KEY}}(\mathcal{B}) \quad (1)$$

The construction of  $\mathcal{B}$  using  $\mathcal{A}$  is straightforward. It receives  $(pk'_i)_{i \in [N]}$ , which is either generated by algorithm  $\text{LID.Gen}$  or by  $\text{LID.LossyGen}$ . Then, it simulates Game 2 for the adversary  $\mathcal{A}$  and sets  $pk_{1-b}^{(i)} := pk'_i$  for all  $i \in [N]$ . Note that, in Game 2, the secret key  $sk_{1-b}^{(i)}$  is not used for any user  $i$ . So  $\mathcal{B}$  is able to simulate the game perfectly. Finally,  $\mathcal{B}$  outputs 1 if and only if  $\mathcal{A}$  wins and  $\text{ImplicitUsage} \wedge (\text{resp}_{1-b(i^*)}^* \neq \text{resp}_{1-b(i^*)})$  happens. It is clear that  $\mathcal{B}$  perfectly simulates Game 2 if it receives normal public keys and  $\mathcal{B}$  perfectly simulates Game 2' if it receives lossy public keys. Thus, Equation (1) follows.

Now in Game 2', the key  $pk_{1-b}^{(i^*)}$  is lossy. Since  $X_{2'}$  implies that  $\sigma^*$  is a valid signature with respect to  $m^*$ , we know that

$$\text{LID.Vrfy}(pk_{1-b}^{(i^*)}, \text{cmt}_{1-b(i^*)}^*, \text{ch}_{1-b(i^*)}^*, \text{resp}_{1-b(i^*)}^*) = 1.$$

Since the signing oracle  $\text{Sign}(i^*, m^*)$  also outputs valid signature  $\sigma$  for  $m^*$ , we have that

$$\text{LID.Vrfy}(pk_{1-b}^{(i^*)}, \text{cmt}_{1-b(i^*)}, \text{ch}_{1-b(i^*)}, \text{resp}_{1-b(i^*)}) = 1.$$

In this subcase, we have  $(\text{cmt}_{1-b(i^*)}, \text{ch}_{1-b(i^*)}) = (\text{cmt}_{1-b(i^*)}^*, \text{ch}_{1-b(i^*)}^*)$  and  $\text{resp}_{1-b(i^*)} \neq \text{resp}_{1-b(i^*)}^*$ . Due to the uniqueness property of LID with respect to lossy public keys, we must have

$$\Pr[X_{2'} \wedge \text{ImplicitUsage} \wedge (\text{resp}_{1-b(i^*)}^* \neq \text{resp}_{1-b(i^*)})] \leq \varepsilon_u.$$

Applying Equation (1) to the obtained bounds, the claim follows.  $\square$

Putting both subcases together, we obtain that

$$\Pr[X_2 \wedge \text{ImplicitUsage}] \leq \frac{1}{2} \Pr[X_2 \wedge \text{ImplicitUsage}] + \text{Adv}_{\text{LID}, N}^{\text{MU-IND-KEY}}(\mathcal{B}) + \varepsilon_u,$$

which implies that  $\Pr[X_2 \wedge \text{ImplicitUsage}] \leq 2\text{Adv}_{\text{LID}, N}^{\text{MU-IND-KEY}}(\mathcal{B}) + 2\varepsilon_u$ .  $\square$

*Game 4.* Game 4 further modifies the winning condition. We denote **Before** as the event that **Both** happens and the first  $(1-b(i^*))$ -query is made before the first  $b(i^*)$ -query is made. Game 4 outputs 0 if event **Before** does not happen.

Hence,  $X_4$  happens if and only if  $X_3 \wedge \text{Before}$  happens. We can prove the following lemma.

**Lemma 15.**  $\Pr[X_4] \geq 1/2 \cdot \Pr[X_3]$ .

*Proof.* Since we know that event **Both** happens, we can divide  $X_3$  into three subcases.

- Both the first 0-query and the first 1-query are made in *one and the same* signing query  $\text{Sign}(i^*, m^*)$ .  
In this subcase, we have that two hash queries  $\{\text{H}(m^*, \text{cmt}_0^*), \text{H}(m^*, \text{cmt}_1^*)\}$

made by the final verification algorithm have the same input as the two hash queries  $\{H(m^*, \text{cmt}_0), H(m^*, \text{cmt}_1)\}$  made by the signing oracle. We know that  $X_3$  implies that `ImplicitUsage` does not happen, so we must have that  $(\text{cmt}_0^*, \text{cmt}_1^*) = (\text{cmt}_1, \text{cmt}_0)$ . Since the signing algorithm always makes a  $H(m^*, \text{cmt}_{b(i^*)})$  query before  $H(m, \text{cmt}_{1-b(i^*)})$ , we have that event `Before` always happens in this subcase.

- Both the first 0-query and the first 1-query are made in one signing query `Sign`( $i', m^*$ ) for some  $i' \neq i^*$ .  
In this subcase, the  $b(i')$ -query is made first and `Before` happens if and only if  $b(i') = 1 - b(i^*)$ .
- The first 0-query and the first 1-query are not made in exactly one signing query. In other words, they lie in different signing queries or at least one of them is made by the adversary.  
In this subcase, the adversary  $\mathcal{A}$  actually has full control which one is queried first. Suppose the  $\beta$ -query is made first for some implicit bit  $\beta \in \{0, 1\}$  chosen by the adversary. Then, event `Before` happens if and only if  $\beta = b(i^*)$ .

Similar to the proof of Lemma 13, we can show that the bit  $b(i^*)$  is perfectly hidden to the adversary. So if the second or the third subcase happens, the probability that `Before` happens is  $1/2$ . Together with the fact that `Before` always happen in the first subcase, Lemma 15 follows.  $\square$

*Game 5.* In this game, we change the generation of the key  $pk_{1-b}^{(i)}$ . Namely, the key generation in Game 5 is exactly as in Game 4 except that we choose lossy public keys  $pk_{1-b}^{(i)} \stackrel{s}{\leftarrow} \text{LID.LossyGen}$  for every user  $i \in [N]$  in Game 5.

**Lemma 16.** *We can construct an adversary  $\mathcal{B}$  with running time  $t_{\mathcal{B}} \approx t_{\mathcal{A}}$  such that*

$$|\Pr[X_4] - \Pr[X_5]| \leq \text{Adv}_{\text{LID}, N}^{\text{MU-IND-KEY}}(\mathcal{B}).$$

*Proof.* The proof of the lemma is straightforward. We can construct  $\mathcal{B}$  using  $\mathcal{A}$  as a subroutine.  $\mathcal{B}$  receives as input  $(pk'_i)_{i \in [N]}$ , which is either generated by algorithm `LID.Gen` or by `LID.LossyGen`. Then, it simulates Game 5 for the adversary  $\mathcal{A}$  and set  $pk_{1-b}^{(i)} := pk'_i$  for all  $i \in [N]$ .  $\square$

Finally, we can prove the following lemma.

**Lemma 17.**

$$\Pr[X_5] \leq N \cdot q_{\text{H}}^2 \cdot \varepsilon_{\ell}$$

where  $q_{\text{H}}$  is the number of hash queries made in Game 5.

Note that the lossiness of LID guarantees that  $\varepsilon_{\ell}$  is *statistically* negligible (even for computationally *unbounded* adversaries). Hence, the multiplicative term  $N \cdot q_{\text{H}}^2$  does not break the tightness of our signature scheme. It will convenient to prove this claim by reduction.

*Proof.* To prove this lemma, we build an adversary  $\mathcal{B}$  against the lossiness of LID. On getting a lossy public key  $pk \xleftarrow{\$} \text{LID.LossyGen}$ ,  $\mathcal{B}$  uniformly selects  $i' \xleftarrow{\$} [N]$ ,  $j_1 \xleftarrow{\$} [q_H - 1]$  and  $j_2 \xleftarrow{\$} \{j_1 + 1, \dots, q_H\}$ . Then  $\mathcal{B}$  generates all the public keys for  $\mathcal{A}$  according to Game 5 except that it sets  $pk_{1-b}^{(i')} := pk$ . Then  $\mathcal{B}$  invokes  $\mathcal{A}$  and answers all the queries according to Game 5 with the following exceptions.

- In the  $j_1$ -th hash query  $H(m, \text{cmt})$ ,  $\mathcal{B}$  submits  $\text{cmt}$  to its own challenger and get back  $\text{ch} \xleftarrow{\$} \text{CSet}$ .
- In the  $j_2$ -th hash query  $H(m, \text{cmt}')$ ,  $\mathcal{B}$  returns  $\text{ch}$  as response and logs  $((m, \text{cmt}'), \text{ch})$  into the hash list  $\mathcal{L}$ .

After  $\mathcal{A}$  submits the forgery attempt  $(i^*, m^*, \sigma^* = (\text{ch}_0^*, \text{resp}_0^*, \text{resp}_1^*))$ ,  $\mathcal{B}$  checks whether all the following events happen:

- $X_5$  happens,
- $i' = i^*$ ,
- the first  $(1 - b^{(i^*)})$ -query is exactly the  $j_1$ -th hash query,
- the first  $b^{(i^*)}$ -query is exactly the  $j_2$ -th hash query.

If all of these events happen,  $\mathcal{B}$  outputs  $\text{resp}_{1-b^{(i^*)}}^*$  to its own challenger. Otherwise,  $\mathcal{B}$  halts and outputs nothing.

The probability that  $\mathcal{B}$  does not halt is at least  $\Pr[X_5]/(N \cdot q_H^2)$ . We will show that in this case

$$\text{LID.Vrfy}(pk, \text{cmt}, \text{ch}, \text{resp}_{1-b^{(i^*)}}^*) = 1,$$

and hence  $\mathcal{B}$  wins the lossiness game. This is implied by the following facts.

- $i' = i^*$  indicates that  $pk = pk_{1-b}^{(i^*)}$ .
- The  $j_1$ -th hash query is the first  $(1 - b^{(i^*)})$ -query indicates that  $\text{cmt} = \text{cmt}_{1-b^{(i^*)}}^*$ .
- The  $j_2$ -th hash query is the first  $b^{(i^*)}$ -query indicates that  $\text{ch} = \text{ch}_{1-b^{(i^*)}}^*$ .
- $X_3$  happens indicates that  $\text{Vrfy}(pk_{1-b}^{(i^*)}, m^*, \sigma^*) = 1$ , which further indicates that

$$\text{LID.Vrfy}(pk_{1-b}^{(i^*)}, \text{cmt}_{1-b^{(i^*)}}^*, \text{ch}_{1-b^{(i^*)}}^*, \text{resp}_{1-b^{(i^*)}}^*) = 1.$$

Thus, we have that  $\Pr[X_5]/(N \cdot q_H^2) \leq \Pr[\mathcal{B} \text{ wins}] \leq \varepsilon_\ell$  and Lemma 17 follows.  $\square$

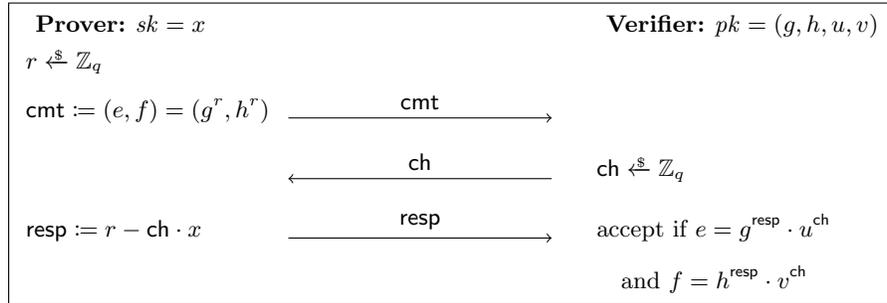
Theorem 9 now follows.  $\square$

## 5 Instantiations of our Scheme

In the previous section we identified the necessary properties of the underlying lossy identification scheme. We now continue to discuss how suitable schemes can be instantiated based on concrete hardness assumptions. The constructions described in this section are derived from [1–3, 29] and are well-known. The purpose of this section is to argue and justify that these constructions indeed satisfy all properties required for our signature scheme.

**5.1 Instantiation based on Decisional Diffie–Hellman**

The well-known DDH-based lossy identification scheme uses the standard Sigma protocol to prove equality of discrete logarithms by Chaum et al. [11] (cf. Figure 1) as foundation, which was used by Katz and Wang [29] to build tightly-secure signatures (in the single-user setting without corruptions).



**Fig. 1.** The DDH-based identification scheme [11].

*The DDH problem.* Let  $(\mathbb{G}, g, q)$  be a cyclic group of prime order  $q$  and generator  $g$ . Further, let  $h \in \mathbb{G}$ . We denote the set of *DDH tuples* in  $\mathbb{G}$  with respect to  $g$  and  $h$  as

$$\text{DDH}(\mathbb{G}, g, h) := \{(u, v) \in \mathbb{G}^2 : \log_g u = \log_h v\}$$

and the set of “*non-DDH tuples*” as

$$\overline{\text{DDH}}(\mathbb{G}, g, h) := \mathbb{G}^2 \setminus \text{DDH}(\mathbb{G}, g, h).$$

**Definition 18.** Let  $(\mathbb{G}, g, q)$  be a cyclic group of prime order  $q$  and generator  $g$ . Further, let  $h \xleftarrow{\$} \mathbb{G}$ . We define the advantage of an algorithm  $\mathcal{B}$  in solving the DDH problem in  $\mathbb{G}$  with respect to  $(g, h)$  as

$$\text{Adv}_{\mathbb{G}, g, h}^{\text{DDH}}(\mathcal{B}) := |\Pr[\mathcal{B}(\mathbb{G}, g, h, u, v) = 1] - \Pr[\mathcal{B}(\mathbb{G}, g, h, \bar{u}, \bar{v}) = 1]|$$

where  $(u, v) \xleftarrow{\$} \text{DDH}(\mathbb{G}, g, h)$  and  $(\bar{u}, \bar{v}) \xleftarrow{\$} \overline{\text{DDH}}(\mathbb{G}, g, h)$  are chosen uniformly random.

**A DDH-based LID Scheme.** Let  $(\mathbb{G}, g, q)$  be a cyclic group of prime order  $q$  and generator  $g$  and let  $h \in \mathbb{G}$ . We define the lossy identification scheme  $\text{LID} = (\text{LID.Gen}, \text{LID.LossyGen}, \text{LID.Prove}, \text{LID.Vrfy}, \text{LID.Sim})$  based on the protocol presented above as follows:

**Key generation.** The algorithm  $\text{LID.Gen}$  chooses a value  $x \xleftarrow{\$} \mathbb{Z}_q$  uniformly at random. It sets  $pk := (g, h, u, v) = (g, h, g^x, h^x)$  and  $sk := x$ , and outputs  $(pk, sk)$ .

**Lossy key generation.** The algorithm  $\text{LID.LossyGen}$  chooses two group elements  $u, v \xleftarrow{\$} \mathbb{G}$  uniformly and independently at random. It outputs  $pk := (g, h, u, v)$ .

**Proving.** The algorithm  $\text{LID.Prove}$  is split up into the following two algorithms:

1. The algorithm  $\text{LID.Prove}_1$  takes as input a secret key  $sk = x$ , chooses a random value  $r \xleftarrow{\$} \mathbb{Z}_q$ , and computes a commitment  $\text{cmt} := (e, f) = (g^r, h^r)$ , where  $g, h$  are the value of the  $pk$  corresponding to  $sk$ . It outputs  $(\text{cmt}, \text{st})$  with  $\text{st} := r$ .
2. The algorithm  $\text{LID.Prove}_2$  takes as input a secret key  $sk = x$ , a commitment  $\text{cmt} = (e, f)$ , a challenge  $\text{ch} \in \mathbb{Z}_q$ , a state  $\text{st} = r$ , and outputs a response  $\text{resp} := r - \text{ch} \cdot x$ .

**Verification.** The verification algorithm  $\text{LID.Vrfy}$  takes as input a public key  $pk = (g, h, u, v)$ , a commitment  $\text{cmt} = (e, f)$ , a challenge  $\text{ch} \in \mathbb{Z}_q$ , and a response  $\text{resp} \in \mathbb{Z}_q$ . It outputs 1 if and only if  $e = g^{\text{resp}} \cdot u^{\text{ch}}$  and  $f = h^{\text{resp}} \cdot v^{\text{ch}}$ .

**Simulation.** The simulation algorithm  $\text{LID.Sim}$  takes as input a public key  $pk = (g, h, u, v)$ , a challenge  $\text{ch} \in \mathbb{Z}_q$ , and a response  $\text{resp} \in \mathbb{Z}_q$ . It outputs a commitment  $\text{cmt} = (e, f) = (g^{\text{resp}} \cdot u^{\text{ch}}, h^{\text{resp}} \cdot v^{\text{ch}})$ .

*Remark 19.* Note that an honest public key generated with  $\text{LID.Gen}$  is of the form  $pk = (g, h, u, v)$  such that  $(u, v) \in \text{DDH}(\mathbb{G}, g, h)$ , whereas a lossy public key generated with  $\text{LID.LossyGen}$  is of the form  $pk = (g, h, u, v)$  such that  $(u, v) \notin \text{DDH}(\mathbb{G}, g, h)$  with high probability.

**Theorem 20.** *The scheme LID defined above is lossy with*

$$\rho = 1, \quad \varepsilon_s = 0, \quad \varepsilon_\ell \leq 1/q,$$

and from any efficient adversary  $\mathcal{A}$  we can construct an efficient adversary  $\mathcal{B}$  such that

$$\text{Adv}_{\text{LID}, N}^{\text{MU-IND-KEY}}(\mathcal{A}) \leq \text{Adv}_{\mathbb{G}, g}^{\text{DDH}}(\mathcal{B}).$$

Furthermore, LID is perfectly unique with respect to lossy keys (i.e.,  $\varepsilon_u = 0$ ), LID has  $\alpha$ -bit min-entropy with  $\alpha = \log_2(q)$ , LID is commitment-recoverable, and LID has an injective simulator.

The proof of this theorem is rather standard and implicitly contained in the aforementioned prior works. For completeness, we provide a sketch in Appendix A.

**Concrete instantiation.** We can now use the DDH-based lossy identification scheme to describe an explicit instantiation of our signature scheme based on the DDH assumption, in order to assess its concrete performance. Let  $\mathbb{G}$  be a group of prime order  $p$  with generator  $g$ , let  $h \xleftarrow{\$} \mathbb{G}$  be a random generator and let  $\text{H}: \{0, 1\}^* \rightarrow \mathbb{Z}_p$  be a hash function. We construct a digital signature scheme  $\text{Sig} = (\text{Gen}, \text{Sign}, \text{Vrfy})$  as follows.

**Key generation.** The key generation  $\text{Gen}$  algorithm samples  $x_0, x_1 \xleftarrow{\$} \mathbb{Z}_p$ ,  $b \xleftarrow{\$} \{0, 1\}$ . Then it sets

$$pk := (u_0, v_0, u_1, v_1) = (g^{x_0}, h^{x_0}, g^{x_1}, h^{x_1}) \quad \text{and} \quad sk := (b, x_b).$$

**Signing.** The signing algorithm  $\text{Sign}$  takes as input  $sk = (b, x_b)$  and a message  $m \in \{0, 1\}^*$ . Then it proceeds as follows.

1. It first chooses a random value  $r \xleftarrow{\$} \mathbb{Z}_p$ , and sets  $(e_b, f_b) := (g^r, h^r)$  and

$$\text{ch}_{1-b} := \text{H}(m, e_b, f_b).$$

2. Then it samples a value  $\text{resp}_{1-b} \xleftarrow{\$} \mathbb{Z}_p$  and computes

$$e_{1-b} = g^{\text{resp}_{1-b}} u_{1-b}^{\text{ch}_{1-b}} \quad \text{and} \quad f_{1-b} = h^{\text{resp}_{1-b}} v_{1-b}^{\text{ch}_{1-b}}.$$

3. Finally, it computes

$$\text{ch}_b := \text{H}(m, e_{1-b}, f_{1-b}) \quad \text{and} \quad \text{resp}_b := r - \text{ch}_b \cdot x_b$$

and outputs the signature  $\sigma := (\text{ch}_0, \text{resp}_0, \text{resp}_1) \in \mathbb{Z}_p^3$ .

**Verification.** The verification algorithm takes as input a public key  $pk := (u_0, v_0, u_1, v_1)$ , a message  $m \in \{0, 1\}^*$ , and a signature  $\sigma = (\text{ch}_0, \text{resp}_0, \text{resp}_1)$ . It first computes

$$e_0 = g^{\text{resp}_0} u_0^{\text{ch}_0} \quad \text{and} \quad f_0 = h^{\text{resp}_0} v_0^{\text{ch}_0}.$$

From  $(e_0, f_0)$  it is then able to compute

$$\text{ch}_1 := \text{H}(m, e_0, f_0)$$

and then

$$e_1 = g^{\text{resp}_1} \cdot u_1^{\text{ch}_1} \quad \text{and} \quad f_1 = h^{\text{resp}_1} \cdot v_1^{\text{ch}_1}.$$

Finally, the algorithm outputs 1 if and only if

$$\text{ch}_0 = \text{H}(m, e_1, f_1).$$

Note that public keys are  $pk \in \mathbb{G}^4$ , secret keys are  $sk \in \{0, 1\} \times \mathbb{Z}_p$ , and signatures are  $\sigma \in \mathbb{Z}_p^3$ .

## 5.2 Instantiation from the $\phi$ -Hiding Assumption

Another possible instantiation is based on the Guillou–Quisquater (GQ) identification scheme [25], which proves that an element  $U = S^e \bmod N$  is an  $e$ -th residue (cf. Figure 2). Abdalla et al. [1] describe a lossy version of the GQ scheme, based on the  $\phi$ -hiding assumption. We observe that we can build a lossy identification scheme on a weaker assumption, which is implied by  $\phi$ -hiding.

In order to achieve tightness in a multi-user setting, we will need a common setup, which is shared across all users. This setup consists of a public tuple  $(N, e)$  where  $N = p \cdot q$  is the product of two large random primes and  $e$  a uniformly random prime of length  $\ell_e \leq \lambda/4$  that divides  $p-1$ . The factors  $p$  and  $q$  need to remain secret, so we assume that  $(N, e)$  either was generated by a trusted party, or by running a secure multi-party computation protocol with multiple parties.

**The Guillou–Quisquater LID Scheme.** We define the lossy identification scheme  $\text{LID} = (\text{LID.Gen}, \text{LID.LossyGen}, \text{LID.Prove}, \text{LID.Vrfy}, \text{LID.Sim})$  based on the protocol presented above as follows:

**Common setup.** The common system parameters are a tuple  $(N, e)$  where  $N = p \cdot q$  is the product of two distinct primes  $p, q$  of length  $\lambda/2$  and  $e$  is random prime of length  $\ell_e \leq \lambda/4$  such that  $e$  divides  $p - 1$ .

Note that the parameters  $(N, e)$  are always in “lossy mode”, and not switched from an “injective” pair  $(N, e)$  where  $e$  is coprime to  $\phi(N) = (p - 1)(q - 1)$  to “lossy” in the security proof, as common in other works.

**Key generation.** The algorithm  $\text{LID.Gen}$  samples  $S \xleftarrow{\$} \mathbb{Z}_N^*$  and computes  $U = S^e$ . It sets  $pk = (N, e, U)$  and  $sk = (N, e, S)$ , where  $(N, e)$  are from the common parameters.

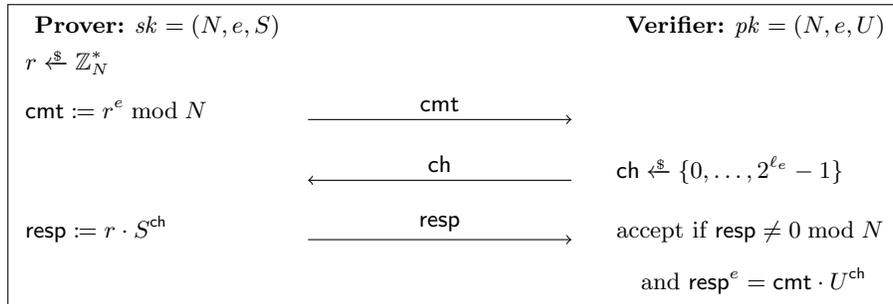
**Lossy key generation.** The lossy key generation algorithm  $\text{LID.LossyGen}$  samples  $U$  uniformly at random from the  $e$ -th non-residues modulo  $N$ .<sup>3</sup>

**Proving.** The algorithm  $\text{LID.Prove}$  is split up into the following two algorithms:

1. The algorithm  $\text{LID.Prove}_1$  takes as input a secret key  $sk = (N, e, S)$ , chooses a random value  $r \xleftarrow{\$} \mathbb{Z}_N^*$ , and computes a commitment  $\text{cmt} := r^e \bmod N$ . It outputs  $(\text{cmt}, \text{st})$  with  $\text{st} := r$ .
2. The algorithm  $\text{LID.Prove}_2$  takes as input a secret key  $sk = (N, e, S)$ , a commitment  $\text{cmt}$ , a challenge  $\text{ch} \in \{0, \dots, 2^{\ell_e} - 1\}$ , a state  $\text{st} = r$ , and outputs a response  $\text{resp} := r \cdot S^{\text{ch}} \bmod N$ .

**Verification.** The verification algorithm  $\text{LID.Vrfy}$  takes as input a public key  $pk = (N, e, U)$ , a commitment  $\text{cmt}$ , a challenge  $\text{ch}$ , and a response  $\text{resp}$ . It outputs 1 if and only if  $\text{resp} \neq 0 \bmod N$  and  $\text{resp}^e = \text{cmt} \cdot U^{\text{ch}}$ .

**Simulation.** The simulation algorithm  $\text{LID.Sim}$  takes as input a public key  $pk = (N, e, U)$ , a challenge  $\text{ch}$ , and a response  $\text{resp}$ . It outputs a commitment  $\text{cmt} = \text{resp}^e / U^{\text{ch}}$ .



**Fig. 2.** The Guillou–Quisquater identification scheme [25].

<sup>3</sup> This is indeed efficiently possible as  $U \xleftarrow{\$} \mathbb{Z}_N^*$  is not an  $e$ -th residue with probability  $1 - 1/e$  and we can efficiently check whether a given  $U$  is an  $e$ -th residue when the factorization of  $N$  is known [1].

**Theorem 21.** *The scheme LID defined above is lossy with*

$$\rho = 1, \quad \varepsilon_s = 0, \quad \varepsilon_\ell \leq 1/2^{\ell_e},$$

and from any efficient adversary  $\mathcal{A}$  we can construct an efficient adversary  $\mathcal{B}$  such that

$$\text{Adv}_{\text{LID},n}^{\text{MU-IND-KEY}}(\mathcal{A}) \leq \text{Adv}^{n\text{-HR}}(\mathcal{B}).$$

Furthermore, LID is perfectly unique with respect to lossy keys (i.e.,  $\varepsilon_u = 0$ ), LID has  $\alpha$ -bit min-entropy with  $\alpha \geq \lambda - 2$ , LID is commitment-recoverable, and LID has an injective simulator.

The above theorem has been proven in [1] for most of its statements. What is left is a proof for  $n$ -key-indistinguishability, which we provide in Appendix B.

### 5.3 On Instantiations of Lossy ID Schemes from Other Assumptions

There also exist lossy identification schemes based on the decisional short discrete logarithm problem, the ring LWE problem, and the subset sum problem (all due to Abdalla et al. [2,3]). However, they do not directly translate to a tight multi-user signature scheme that is existentially unforgeable with adaptive corruptions.

Our security proof requires tight multi-instance security of the underlying hardness assumption. While, for example, the DDH-based scheme satisfies this via its self-reducibility property, it is not obvious how schemes based on, for example, lattices or subset sum achieve this notion in a *tight* manner.

## References

1. Michel Abdalla, Fabrice Ben Hamouda, and David Pointcheval. Tighter reductions for forward-secure signature schemes. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 292–311. Springer, Heidelberg, February / March 2013.
2. Michel Abdalla, Pierre-Alain Fouque, Vadim Lyubashevsky, and Mehdi Tibouchi. Tightly-secure signatures from lossy identification schemes. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 572–590. Springer, Heidelberg, April 2012.
3. Michel Abdalla, Pierre-Alain Fouque, Vadim Lyubashevsky, and Mehdi Tibouchi. Tightly secure signatures from lossy identification schemes. *Journal of Cryptology*, 29(3):597–631, July 2016.
4. Masayuki Abe, Miyako Ohkubo, and Koutarou Suzuki. 1-out-of- $n$  signatures from a variety of keys. In Yuliang Zheng, editor, *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 415–432. Springer, Heidelberg, December 2002.
5. Christoph Bader, Dennis Hofheinz, Tibor Jäger, Eike Kiltz, and Yong Li. Tightly-secure authenticated key exchange. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part I*, volume 9014 of *LNCS*, pages 629–658. Springer, Heidelberg, March 2015.

6. Christoph Bader, Tibor Jager, Yong Li, and Sven Schäge. On the impossibility of tight cryptographic reductions. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 273–304. Springer, Heidelberg, May 2016.
7. Mihir Bellare, Alexandra Boldyreva, and Silvio Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 259–274. Springer, Heidelberg, May 2000.
8. Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 232–249. Springer, Heidelberg, August 1994.
9. Christian Cachin, Silvio Micali, and Markus Stadler. Computationally private information retrieval with polylogarithmic communication. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 402–414. Springer, Heidelberg, May 1999.
10. Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 453–474. Springer, Heidelberg, May 2001.
11. David Chaum, Jan-Hendrik Evertse, and Jeroen van de Graaf. An improved protocol for demonstrating possession of discrete logarithms and some generalizations. In David Chaum and Wyn L. Price, editors, *EUROCRYPT'87*, volume 304 of *LNCS*, pages 127–141. Springer, Heidelberg, April 1988.
12. Katriel Cohn-Gordon, Cas Cremers, Kristian Gjøsteen, Håkon Jacobsen, and Tibor Jager. Highly efficient key exchange protocols with optimal tightness. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 767–797. Springer, Heidelberg, August 2019.
13. Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Yvo Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 174–187. Springer, Heidelberg, August 1994.
14. Hannah Davis and Felix Günther. Tighter proofs for the sigma and TLS 1.3 key exchange protocols. Cryptology ePrint Archive, Report 2020/1029, 2020. <https://eprint.iacr.org/2020/1029>.
15. Denis Diemert and Tibor Jager. On the tight security of TLS 1.3: Theoretically-sound cryptographic parameters for real-world deployments. Cryptology ePrint Archive, Report 2020/726; to appear in the Journal of Cryptology, 2020. <https://eprint.iacr.org/2020/726>.
16. Digital signature standard (DSS). National Institute of Standards and Technology (NIST), FIPS PUB 186-3, U.S. Department of Commerce, 2009. [http://csrc.nist.gov/publications/fips/fips186-3/fips\\_186-3.pdf](http://csrc.nist.gov/publications/fips/fips186-3/fips_186-3.pdf).
17. Manuel Fersch, Eike Kiltz, and Bertram Poettering. On the provable security of (EC)DSA signatures. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 1651–1662. ACM Press, October 2016.
18. Marc Fischlin, Patrick Harasser, and Christian Janson. Signatures from sequential-OR proofs. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part III*, volume 12107 of *LNCS*, pages 212–244. Springer, Heidelberg, May 2020.
19. Marc Fischlin, Anja Lehmann, Thomas Ristenpart, Thomas Shrimpton, Martijn Stam, and Stefano Tessaro. Random oracles with(out) programmability. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 303–320. Springer, Heidelberg, December 2010.

20. Nils Fleischhacker, Tibor Jager, and Dominique Schröder. On tight security proofs for Schnorr signatures. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 512–531. Springer, Heidelberg, December 2014.
21. Nils Fleischhacker, Tibor Jager, and Dominique Schröder. On tight security proofs for Schnorr signatures. *Journal of Cryptology*, 32(2):566–599, April 2019.
22. Sanjam Garg, Raghav Bhaskar, and Satyanarayana V. Lokam. Improved bounds on security reductions for discrete log based signatures. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 93–107. Springer, Heidelberg, August 2008.
23. Kristian Gjøsteen and Tibor Jager. Practical and tightly-secure digital signatures and authenticated key exchange. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 95–125. Springer, Heidelberg, August 2018.
24. Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988.
25. Louis C. Guillou and Jean-Jacques Quisquater. A “paradoxical” indentity-based signature scheme resulting from zero-knowledge. In Shafi Goldwasser, editor, *CRYPTO’88*, volume 403 of *LNCS*, pages 216–231. Springer, Heidelberg, August 1990.
26. Shingo Hasegawa and Shuji Isobe. Lossy identification schemes from decisional RSA. In *International Symposium on Information Theory and its Applications, ISITA 2014, Melbourne, Australia, October 26-29, 2014*, pages 143–147. IEEE, 2014.
27. Dennis Hofheinz and Tibor Jager. Tightly secure signatures and public-key encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 590–607. Springer, Heidelberg, August 2012.
28. Tibor Jager, Eike Kiltz, Doreen Riepel, and Sven Schäge. Tightly-secure authenticated key exchange, revisited. Cryptology ePrint Archive, Report 2020/1279, 2020. <https://eprint.iacr.org/2020/1279>.
29. Jonathan Katz and Nan Wang. Efficiency improvements for signature schemes with tight security reductions. In Sushil Jajodia, Vijayalakshmi Atluri, and Trent Jaeger, editors, *ACM CCS 2003*, pages 155–164. ACM Press, October 2003.
30. Eike Kiltz, Daniel Masny, and Jiaxin Pan. Optimal security proofs for signatures from identification schemes. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 33–61. Springer, Heidelberg, August 2016.
31. Eike Kiltz, Adam O’Neill, and Adam Smith. Instantiability of RSA-OAEP under chosen-plaintext attack. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 295–313. Springer, Heidelberg, August 2010.
32. Hugo Krawczyk. SIGMA: The “SIGn-and-MAC” approach to authenticated Diffie-Hellman and its use in the IKE protocols. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 400–425. Springer, Heidelberg, August 2003.
33. Yong Li and Sven Schäge. No-match attacks and robust partnering definitions: Defining trivial attacks for security protocols is not trivial. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1343–1360. ACM Press, October / November 2017.
34. Xiangyu Liu, Shengli Liu, Dawu Gu, and Jian Weng. Two-pass authenticated key exchange with explicit authentication and tight security. In Shiho Moriai

- and Huaxiong Wang, editors, *ASIACRYPT 2020, Part II*, volume 12492 of *LNCS*, pages 785–814. Springer, Heidelberg, December 2020.
35. Pascal Paillier and Damien Vergnaud. Discrete-log-based signatures may not be equivalent to discrete log. In Bimal K. Roy, editor, *ASIACRYPT 2005*, volume 3788 of *LNCS*, pages 1–20. Springer, Heidelberg, December 2005.
  36. Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 239–252. Springer, Heidelberg, August 1990.
  37. Yannick Seurin. On the exact security of Schnorr-type signatures in the random oracle model. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 554–571. Springer, Heidelberg, April 2012.

## A Proof of Theorem 20

*Random Self-Reducibility of DDH.* It is well-known that the DDH problem is random self-reducible, which we summarize in the following lemma. See [7, Lemma 5.2] for a proof.

**Lemma 22.** *There exists an efficient algorithm ReRand that takes as input  $(g, h)$  and a DDH instance  $(u, v) \in \mathbb{G}^2$  and an integer  $N$ , and outputs  $N$  new DDH instances  $(u^{(i)}, v^{(i)})$  such that*

$$(u, v) \in \text{DDH}(\mathbb{G}, g, h) \iff (u^{(i)}, v^{(i)}) \in \text{DDH}(\mathbb{G}, g, h)$$

for all  $i \in [N]$ . The running time of this algorithm mainly consists of  $\mathcal{O}(N)$  exponentiations in  $\mathbb{G}$ .

*Proof.* To show that LID is lossy, we need to show that it satisfies all properties presented in Definition 4.

**Completeness of normal keys.** We claim that the above scheme is perfectly-complete. To prove this, we show that for any honest transcript it holds that  $\text{LID.Vrfy}(pk, \text{cmt}, \text{ch}, \text{resp}) = 1$ . Let  $(pk, sk) \xleftarrow{\$} \text{LID.Gen}$  be an (honest) key pair and let  $(\text{cmt}, \text{ch}, \text{resp})$  be an honest transcript, that is,  $\text{ch} \xleftarrow{\$} \text{CSet}$ ,  $(\text{cmt}, \text{st}) \xleftarrow{\$} \text{LID.Prove}_1(sk)$  and  $\text{resp} := \text{LID.Prove}_2(sk, \text{cmt}, \text{ch}, \text{st})$ . By definition of the scheme, we have  $pk = (g, h, u, v)$  with  $(u, v) \in \text{DDH}(\mathbb{G}, g, h)$  and  $sk = x$  and  $\text{cmt} = (e, f) = (g^r, h^r)$  and  $\text{resp} = r - \text{ch} \cdot x$ . Further,  $\text{LID.Vrfy}(pk, \text{cmt}, \text{ch}, \text{resp}) = 1$  if and only if  $e = g^{\text{resp}} \cdot u^{\text{ch}}$  and  $f = h^{\text{resp}} \cdot v^{\text{ch}}$ . Observe that

$$g^{\text{resp}} \cdot u^{\text{ch}} = g^{r - \text{ch} \cdot x} \cdot g^{\text{ch} \cdot x} = g^r = e.$$

An analogous equation holds for  $f$  if  $g$  is replaced by  $h$ . Hence,  $\text{LID.Vrfy}$  outputs 1 for every honest transcript.

**Simulatability of transcripts.** We claim that the above scheme is perfectly simulatable. To show this, we need to argue that the two distributions

$$\left\{ \begin{array}{l} (\text{cmt}, \text{st}) \xleftarrow{\$} \text{LID.Prove}_1(sk) \\ (\text{cmt}, \text{ch}, \text{resp}) : \text{ch} \xleftarrow{\$} \mathbb{Z}_q \\ \text{resp} := \text{LID.Prove}_2(sk, \text{ch}, \text{cmt}, \text{st}) \end{array} \right\}$$

and

$$\left\{ \begin{array}{l} \text{ch} \stackrel{\$}{\leftarrow} \mathbb{Z}_q \\ (\text{cmt}, \text{ch}, \text{resp}) : \text{resp} \stackrel{\$}{\leftarrow} \mathbb{Z}_q \\ \text{cmt} := \text{LID.Sim}(pk, \text{ch}, \text{resp}) \end{array} \right\}$$

are identical. Recall that we have  $pk = (g, h, u, v)$  with  $(u, v) \in \text{DDH}(\mathbb{G}, g, h)$ ,  $sk = x$ ,  $\text{cmt} = (e, f) = (g^r, h^r)$  with  $\text{st} = r \stackrel{\$}{\leftarrow} \mathbb{Z}_q$ , and  $\text{resp} = r - \text{ch} \cdot x$  for an honest transcript (i.e., in the former distribution). Thus, we have that  $\text{cmt} = (e, f)$  is uniformly distributed over  $\mathbb{G}^2$ . Consequently, since  $r \stackrel{\$}{\leftarrow} \mathbb{Z}_q$  and  $\text{ch} \stackrel{\$}{\leftarrow} \mathbb{Z}_q$ , we also have that the response  $\text{resp}$  is distributed uniformly and independently (of  $\text{cmt}$  and  $\text{ch}$ ) over  $\mathbb{Z}_q$ .

We will now take a look at the later distribution. Note that  $\text{ch}$  and  $\text{resp}$  are both uniformly random elements over  $\mathbb{Z}_p$ . It remains to show that  $\text{cmt}$  in the simulated transcript is distributed uniformly over  $\mathbb{G}^2$ .

Recall that  $\text{cmt} := \text{LID.Sim}(pk, \text{ch}, \text{resp})$  is defined as  $\text{cmt} := (e, f) = (g^{\text{resp}} \cdot u^{\text{ch}}, h^{\text{resp}} \cdot v^{\text{ch}})$ . Observe that  $\log_g(e) = \text{resp} + \text{ch} \cdot x$  and  $\log_g(f) = \log_g(h) \cdot (\text{resp} + \text{ch} \cdot x)$ . Since  $\text{ch} \stackrel{\$}{\leftarrow} \mathbb{Z}_q$  and  $\text{resp} \stackrel{\$}{\leftarrow} \mathbb{Z}_q$ , we have that both  $\log_g(e)$  and  $\log_g(f)$  are distributed uniformly and independently (of  $\text{ch}$  and  $\text{resp}$ ) over  $\mathbb{Z}_q$  and thus  $(e, f)$  is distributed uniformly over  $\mathbb{G}^2$ . Note that  $e, f$  are not distributed independently of each other (as it is the case in the honest transcript).

**Indistinguishability of keys.** As already remarked above, honest keys contain a DDH tuple, whereas lossy keys contain a non-DDH tuple. Therefore, we claim that for every adversary  $\mathcal{A}$  trying to distinguish honest from lossy keys of LID, we can construct an adversary  $\mathcal{B}$  such that

$$\text{Adv}_{\text{LID}, N}^{\text{MU-IND-KEY}}(\mathcal{A}) \leq \text{Adv}_{\mathbb{G}, g}^{\text{DDH}}(\mathcal{B}).$$

To prove this claim, we give a construction of  $\mathcal{B}$  running  $\mathcal{A}$  as a subroutine. The adversary  $\mathcal{B}$  receives a tuple  $(g, h, u, v)$  such that  $(u, v)$  either is a DDH tuple (i.e.,  $(u, v) \in \text{DDH}(\mathbb{G}, g, h)$ ) or not. Then, it uses the algorithm of Lemma 22 to re-randomize  $(u, v)$  into  $N$  tuples  $(u^{(i)}, v^{(i)})_{i \in [N]} \stackrel{\$}{\leftarrow} \text{ReRand}(g, h, u, v, N)$  such that

$$(u, v) \in \text{DDH}(\mathbb{G}, g, h) \iff \forall i \in [N] : (u^{(i)}, v^{(i)}) \in \text{DDH}(\mathbb{G}, g, h)$$

and hands  $(pk_i = (g, h, u^{(i)}, v^{(i)}))_{i \in [N]}$  to  $\mathcal{A}$  as input. When  $\mathcal{A}$  halts and outputs a bit  $b$ ,  $\mathcal{B}$  halts and outputs  $b$  as well.

Observe that by Lemma 22, we have

$$\Pr[\mathcal{B}(g, h, u, v) = 1] \geq \Pr[\mathcal{A}(pk^{(1)}, \dots, pk^{(N)})]$$

with  $(u, v) \stackrel{\$}{\leftarrow} \text{DDH}(\mathbb{G}, g, h)$ ,  $(u^{(i)}, v^{(i)})_{i \in [N]} \stackrel{\$}{\leftarrow} \text{ReRand}(g, h, u, v, N)$ , and  $pk^{(i)} := (g, h, u^{(i)}, v^{(i)})$ . Further, we have

$$\Pr[\mathcal{B}(g, h, \bar{u}, \bar{v}) = 1] \geq \Pr[\mathcal{A}(pk'^{(1)}, \dots, pk'^{(N)})]$$

with  $(\bar{u}, \bar{v}) \stackrel{\$}{\leftarrow} \overline{\text{DDH}(\mathbb{G}, g, h)}$ ,  $(\bar{u}^{(i)}, \bar{v}^{(i)}) \stackrel{\$}{\leftarrow} \text{ReRand}(g, h, \bar{u}, \bar{v})$  for every  $i \in [N]$ , and  $pk'^{(i)} := (g, h, \bar{u}^{(i)}, \bar{v}^{(i)})$ . In conclusion, we have

$$\text{Adv}_{\text{LID}, N}^{\text{MU-IND-KEY}}(\mathcal{A}) \leq \text{Adv}_{\mathbb{G}, g}^{\text{DDH}}(\mathcal{B}).$$

**Lossiness.** We claim that the above scheme LID is  $1/q$ -lossy. To show this, we first recall a classical result showing the soundness of the protocol to “prove DDH tuples” by Chaum et al. presented above. Namely, we claim that if  $\log_g(u) \neq \log_h(v)$  holds for the public key  $pk = (g, h, u, v)$  (i.e.,  $pk$  is a lossy key and  $(u, v) \notin \text{DDH}(\mathbb{G}, g, h)$ ), for any commitment  $\text{cmt}$  there can only be at most one challenge  $\text{ch}$  such that the transcript is valid. We prove this statement by contradiction.

Let  $\mathcal{A}$  be an unbounded adversary that on input of a lossy public key  $pk \xleftarrow{\$} \text{LID.LossyGen}$ , outputs commitment  $\text{cmt} = (e, f)$ . We now show that  $\mathcal{A}$  can only output a correct  $\text{resp}$  for *one*  $\text{ch}$  such that  $\text{LID.Vrfy}(pk, \text{cmt}, \text{ch}, \text{resp}) = 1$ . Suppose that  $\mathcal{A}$  was able to come up with two responses  $\text{resp}_1$  and  $\text{resp}_2$  for two different challenge  $\text{ch}_1 \neq \text{ch}_2$  such that  $\text{LID.Vrfy}(pk, \text{cmt}, \text{ch}_1, \text{resp}_1) = 1$  and  $\text{LID.Vrfy}(pk, \text{cmt}, \text{ch}_2, \text{resp}_2) = 1$  holds. This implies by the definition of  $\text{LID.Vrfy}$  that

$$e = g^{\text{resp}_1} u^{\text{ch}_1} = g^{\text{resp}_2} u^{\text{ch}_2} \quad \text{and} \quad f = h^{\text{resp}_1} v^{\text{ch}_1} = h^{\text{resp}_2} v^{\text{ch}_2}.$$

Equivalently, we get by using the assumption that  $\text{ch}_1 \neq \text{ch}_2$ :

$$\log_g(u) = \frac{(\text{resp}_1 - \text{resp}_2)}{\text{ch}_2 - \text{ch}_1} \quad \text{and} \quad \log_h(v) = \frac{(\text{resp}_1 - \text{resp}_2)}{\text{ch}_2 - \text{ch}_1}.$$

However, this is a contraction to the assumption that  $\log_g(u) \neq \log_h(v)$ . Thus,  $pk$  must be a lossy key.

Using this, we have that for every commitment  $\mathcal{A}$  outputs, there can only be at most one challenge  $\text{ch}$  such that the adversary generated a valid transcript. Note that we have an unbounded adversary and based on  $\text{cmt}$  and  $\text{ch}$  it can compute a response. As there is only one challenge for  $\text{cmt}$  output by  $\mathcal{A}$  and the challenge is chosen uniformly at random, the adversary can only win with a probability of at most  $1/q$ .

**Uniqueness with respect to lossy keys.** Let  $pk = (g, h, u, v)$  with  $(u, v) \notin \text{DDH}(\mathbb{G}, g, h)$  and  $(\text{cmt}, \text{ch}, \text{resp})$  with  $\text{LID.Vrfy}(pk, \text{cmt}, \text{ch}, \text{resp}) = 1$ . Suppose that there is a  $\text{resp}' \neq \text{resp}$  such that  $\text{LID.Vrfy}(pk, \text{cmt}, \text{ch}, \text{resp}') = 1$ . In this case, we have for  $\text{cmt} = (e, f)$  that

$$e = g^{\text{resp}} u^{\text{ch}} = g^{\text{resp}'} u^{\text{ch}} \quad \text{and} \quad f = h^{\text{resp}} v^{\text{ch}} = h^{\text{resp}'} v^{\text{ch}}.$$

However, this implies that

$$g^{\text{resp}} = g^{\text{resp}'} \quad \text{and} \quad h^{\text{resp}} = h^{\text{resp}'},$$

which implies that  $\text{resp} = \text{resp}'$ , contradicting the initial assumption.

**Min-entropy.** For any secret key  $sk$ , the commitment  $\text{cmt} \xleftarrow{\$} \text{LID.Prove}_1(sk)$  equals  $(g^r, h^r)$  for  $r \xleftarrow{\$} \mathbb{Z}_q$ , which is independent of  $sk$ . So the min-entropy of  $\text{cmt}$  is  $\alpha = \log_2(q)$ .

**Commitment-recoverable.** The verification algorithm of LID first recovers a commitment using the simulator and then compares the result with the commitment in the transcript. So LID is commitment-recoverable.

**Injective simulator.** For any normal public key  $pk = (g, h, u, v)$ , any response  $\text{resp}$  and any challenge  $\text{ch} \neq \text{ch}'$ , we have that

$$\begin{aligned} \text{LID.Sim}(pk, \text{ch}, \text{resp}) &= (g^{\text{resp}_u \text{ch}}, h^{\text{resp}_v \text{ch}}), \\ \text{LID.Sim}(pk, \text{ch}', \text{resp}) &= (g^{\text{resp}_u \text{ch}'}, h^{\text{resp}_v \text{ch}'}). \end{aligned}$$

Thus, if the above two pairs are equal, we must have that  $(u^{\text{ch}}, v^{\text{ch}}) = (u^{\text{ch}'}, v^{\text{ch}'})$ . That implies  $\text{ch} = \text{ch}'$ .  $\square$

## B Proof of Theorem 21

The following definition is from [1].

**Definition 23 (RSA modulus generation algorithm).** Let  $\ell_N$  be a positive integer and let  $\text{RSA}_{\ell_N}$  be the set of all tuples  $(N, p_1, p_2)$  such that  $N = p_1 p_2$  is a  $\ell_N$ -bit number and  $p_1, p_2$  are two distinct primes in the set of  $\ell_N/2$ -bit primes  $\mathbb{P}_{\ell_N/2}$ . Let  $R$  be any relation on  $p_1$  and  $p_2$ , define  $\text{RSA}_{\ell_N}[R] := \{(N, p_1, p_2) \in \text{RSA}_{\ell_N} \mid R(p_1, p_2) = 1\}$ .

We can use it to define the  $n$ -fold higher residuosity assumption as well as the  $\phi$ -hiding assumption [1, 9, 31].

**Definition 24 ( $n$ -fold higher residuosity assumption).** Let  $e$  be a random prime of length  $\ell_e \leq \ell_N/4$  and

$$(N, p_1, p_2) \stackrel{\$}{\leftarrow} \text{RSA}_{\ell_N}[p_1 = 1 \pmod{e}]$$

and let  $\text{HR}_N[e] := \{g^e \pmod{N} \mid g \in \mathbb{Z}_N^*\}$  be the set of  $e$ -th residues modulo  $N$ . We define the advantage of any  $\mathcal{A}$  in solving the higher residuosity problem as

$$\text{Adv}^{n\text{-HR}}(\mathcal{A}) := |\Pr[\mathcal{A}(N, e, y_1, \dots, y_n) = 1] - \Pr[\mathcal{A}(N, e, y'_1, \dots, y'_n) = 1]|,$$

where  $y_1, \dots, y_n \stackrel{\$}{\leftarrow} \text{HR}_N[e]$  and  $y'_1, \dots, y'_n \stackrel{\$}{\leftarrow} \mathbb{Z}_N^* \setminus \text{HR}_N[e]$ . The  $e$ -residuosity problem is  $(t, \varepsilon)$ -hard if for any  $\mathcal{A}$  with running time at most  $t$ ,  $\text{Adv}^{n\text{-HR}}(\mathcal{A})$  is at most  $\varepsilon$ .

We prove the following lemma.

**Lemma 25.** For any adversary  $\mathcal{A}$  with running time  $t_{\mathcal{A}}$  against the  $n$ -key-indistinguishability of LID in Figure 2, we can construct an adversary  $\mathcal{B}$  with running time  $t_{\mathcal{B}} \approx t_{\mathcal{A}}$  such that

$$\text{Adv}_{\text{LID}, n}^{\text{MU-IND-KEY}}(\mathcal{A}) \leq \text{Adv}^{n\text{-HR}}(\mathcal{B}).$$

*Proof.* The proof is a straightforward reduction.  $\mathcal{B}$  receives  $(N, e, y_1, \dots, y_n)$  as input and defines the common parameters as  $(N, e)$  and

$$(pk^{(1)}, \dots, pk^{(n)}) = (y_1, \dots, y_n).$$

Note that this defines real keys if the  $y_i$  are  $e$ -th residues, and lossy keys if the  $y_i$  are  $e$ -th non-residues.  $\square$

Finally, we can show that the  $n$ -fold higher residuosity assumption is tightly implied by the  $\phi$ -hiding assumption, for any polynomially-bounded  $n$ .

**Definition 26** ( $\phi$ -hiding assumption [1, 9, 31]). *Let  $c \leq 1/4$  be a constant. For any adversary  $\mathcal{A}$ , define the advantage of  $\mathcal{A}$  in solving the  $\phi$ -hiding problem to be*

$$\text{Adv}^{\phi H}(\mathcal{A}) := |\Pr[\mathcal{A}(N, e) = 1] - \Pr[\mathcal{A}(N', e) = 1]|,$$

where  $e \xleftarrow{\$} \mathbb{P}_{c\ell_N}$ ,  $(N, p_1, p_2) \xleftarrow{\$} \text{RSA}_{\ell_N}[\gcd(e, \phi(N)) = 1]$  and  $(N', p'_1, p'_2) \xleftarrow{\$} \text{RSA}_{\ell_N}[p'_1 = 1 \bmod e]$ . The  $\phi$ -hiding problem is  $(t, \varepsilon)$ -hard if for any  $\mathcal{A}$  with running time at most  $t$ ,  $\text{Adv}^{\phi H}(\mathcal{A})$  is at most  $\varepsilon$ .

**Lemma 27.** *For any adversary  $\mathcal{A}$  with running time  $t_{\mathcal{A}}$  we can construct an adversary  $\mathcal{B}$  with running time  $t_{\mathcal{B}} \approx t_{\mathcal{A}}$  such that*

$$\text{Adv}^{n\text{-HR}}(\mathcal{A}) \leq 2 \cdot \text{Adv}^{\phi H}(\mathcal{B}).$$

*Proof.* First, we have that

$$\begin{aligned} \text{Adv}^{n\text{-HR}}(\mathcal{A}) &= |\Pr[\mathcal{A}(N, e, y_1, \dots, y_n) = 1] - \Pr[\mathcal{A}(N, e, y'_1, \dots, y'_n) = 1]| \\ &\leq |\Pr[\mathcal{A}(N, e, y_1, \dots, y_n) = 1] - \Pr[\mathcal{A}(N', e, y'_1, \dots, y'_n) = 1]| \\ &\quad + |\Pr[\mathcal{A}(N', e, y'_1, \dots, y'_n) = 1] - \Pr[\mathcal{A}(N, e, y'_1, \dots, y'_n) = 1]|, \end{aligned}$$

where  $(N, p_1, p_2) \xleftarrow{\$} \text{RSA}_{\ell_N}[\gcd(e, \phi(N)) = 1]$  and  $(N', p'_1, p'_2) \xleftarrow{\$} \text{RSA}_{\ell_N}[p'_1 = 1 \bmod e]$ . We can prove the following claim.

*Claim.*  $|\Pr[\mathcal{A}(N, e, y_1, \dots, y_n) = 1] - \Pr[\mathcal{A}(N', e, y'_1, \dots, y'_n) = 1]| \leq \text{Adv}^{\phi H}(\mathcal{B})$ .

The proof is again a very straightforward reduction.  $\mathcal{B}$  receives as input  $(N, e)$ . It samples  $x_1, \dots, x_n \xleftarrow{\$} \mathbb{Z}_N$  uniformly random and then defines  $y_i := x_i^e \bmod N$  for  $i \in \{1, \dots, n\}$ . Then it runs  $\mathcal{A}$  on input  $(N, e, y_1, \dots, y_n)$  and returns whatever  $\mathcal{A}$  returns.

Note that if  $(N, e)$  is a “lossy” key, so that  $e \mid \phi(N)$ , then the  $y_i$  are random  $e$ -th residues. However, if  $\gcd(e, \phi(N)) = 1$ , then all  $y_i$  are random  $e$ -th non-residues, since the map  $x \mapsto x^e \bmod N$  is a permutation.

Using a similar idea, we can prove that

*Claim.*  $|\Pr[\mathcal{A}(N', e, y'_1, \dots, y'_n) = 1] - \Pr[\mathcal{A}(N, e, y'_1, \dots, y'_n) = 1]| \leq \text{Adv}^{\phi H}(\mathcal{B})$ .

Putting the two claims together, we have that  $\text{Adv}^{n\text{-HR}}(\mathcal{A}) \leq 2\text{Adv}^{\phi H}(\mathcal{B})$  and Lemma 27 follows.  $\square$