From Identification to Signatures, Tightly: A Framework and Generic Transforms

Mihir Bellare[1] Bertram Poettering[2] Douglas Stebila[3]

February 2016

Abstract

This paper provides a framework to treat the problem of building signature schemes from identification schemes in a unified and systematic way. The outcomes are (1) Three alternatives to the Fiat-Shamir transform yielding signature schemes whose proofs give *tight* reductions to *standard* assumptions (2) An understanding and characterization of existing transforms in the literature. Reduction tightness is important because it allows the implemented scheme to use small parameters (thereby being as efficient as possible) while retaining provable security.

[1] Department of Computer Science & Engineering, University of California San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. Email: `mihir@eng.ucsd.edu`. URL: `http://cseweb.ucsd.edu/~mihir/`. Supported in part by NSF grants CNS-1228890 and CNS-1526801 and a gift from Microsoft corporation.

[2] Ruhr University Bochum, Bochum, Germany. Email: `bertram.poettering@rub.de`. URL: . Supported by ERC Project ERCC (FP7/615074).

[3] Queensland University of Technology, Brisbane, Australia. Email: `stebila@qut.edu.au`. URL: `http://www.douglas.stebila.ca/`. Supported in part by Australian Research Council (ARC) Discovery Project grant DP130104304.

Contents

1 Introduction

This paper provides a framework to treat the problem of building signature schemes from identification schemes in a unified and systematic way. We are able to explain and characterize existing transforms as well as give new ones whose security proofs give *tight* reductions to *standard* assumptions. This is important so that the implemented scheme can use small parameters, thereby being efficient while retaining provable security. Let us begin by identifying the different elements involved.

ID-TO-SIG TRANSFORMS. Recall that in a three-move identification scheme ID the prover sends a *commitment* Y computed using private randomness y, the verifier sends a random *challenge* c, the prover returns a *response* z computed using y and its secret key *isk*, and the verifier computes a boolean decision from the conversation transcript $Y\|c\|z$ and public key *ivk* (see Fig. 3). We are interested in transforms $\boxed{\textbf{Id2Sig}}$ that take ID and return a signature scheme DS. The transform must be generic, meaning DS is proven to meet some signature security goal $\boxed{\mathrm{P_{sig}}}$ assuming only that ID meets some identification security goal $\boxed{\mathrm{P_{id}}}$. This proof is underlain by a reduction $\boxed{\mathrm{P_{sig}}\to\mathrm{P_{id}}}$ that may be tight or loose. Boxing an item here highlights elements of interest and choice in the id-to-sig process.

CANONICAL EXAMPLE. In the most canonical example we have, **Id2Sig** = **FS** is the Fiat-Shamir transform [16] ; $\mathrm{P_{id}}$ = IMP-PA is security against impersonation under passive attack [14, 1] ; $\mathrm{P_{sig}}$ = UF is unforgeability under chosen-message attack [18] ; and the reduction $\mathrm{P_{sig}}\to\mathrm{P_{id}}$ is that of AABN [1], which is loose.

We are going to revisit this to give other choices of the different elements, but first let us recall some more details of the above. In the Fiat-Shamir transform **FS** [16], a signature of a message m is a pair (Y, z) such that the transcript $Y\|c\|z$ is accepting for $c = \mathrm{H}(Y\|m)$, where H is a random oracle. IMP-PA requires that an adversary given transcripts of honest protocol executions still fails to make the honest verifier accept in an interaction where it plays the role of the prover, itself picking Y any way it likes, receiving a random c, and then producing z. The loss in the $\mathrm{P_{sig}}\to\mathrm{P_{id}}$ reduction of AABN [1] is a factor of the number q of adversary queries to the random oracle H: If $\epsilon_{\mathrm{id}}, \epsilon_{\mathrm{sig}}$ denote, respectively, the advantages in breaking the IMP-PA security of ID and the UF security of DS, then $\epsilon_{\mathrm{sig}} \approx q\,\epsilon_{\mathrm{id}}$.

ALGEBRAIC ASSUMPTION TO ID. Suppose a cryptographer wants to build a signature scheme meeting the definition $\mathrm{P_{sig}}$. The cryptographer would like to base security on some algebraic assumption $\boxed{\mathrm{P_{alg}}}$. This could be factoring, RSA inversion, bilinear Diffie-Hellman, some lattice assumption, or many others. Given an id-to-sig transform as above, the task amounts to designing an identification scheme ID achieving $\mathrm{P_{id}}$ under $\mathrm{P_{alg}}$. (Then one can just apply the transform to ID.) This proof is underlain by another reduction $\boxed{\mathrm{P_{id}}\to\mathrm{P_{alg}}}$ that again may be tight or loose. The tightness of the overall reduction $\mathrm{P_{sig}}\to\mathrm{P_{alg}}$ thus depends on the tightness of both $\mathrm{P_{sig}}\to\mathrm{P_{id}}$ and $\mathrm{P_{id}}\to\mathrm{P_{alg}}$.

CANONICAL EXAMPLE. Continuing with the FS+AABN-based example from above, we would need to build an identification scheme meeting $\mathrm{P_{id}}$ = IMP-PA under $\mathrm{P_{alg}}$. The good news is that a wide swathe of such identification schemes are available, for many choices of $\mathrm{P_{alg}}$ (GQ [20] under RSA, FS [16] under Factoring, Schnorr [29] under Discrete Log, ...). However the reduction $\mathrm{P_{id}}\to\mathrm{P_{alg}}$ is (very) loose.

Again, we are going to revisit this to give other choices of the different elements, but first let us recall some more details of the above. The practical identification schemes here are typically Sigma protocols (this means they satisfy honest-verifier zero-knowledge and special soundness, the latter meaning that from two accepting conversation transcripts with the same commitment but different challenges, one can extract the secret key) and $\mathrm{P_{alg}} = \mathrm{KR}$ is the problem of computing the secret key given only the public key. To solve this problem, we have to run a given IMP-PA adversary twice and hope for two successes. The analysis exploits the Reset Lemma of [7]. If $\epsilon_{\mathrm{alg}}, \epsilon_{\mathrm{id}}$ denote, respectively, the advantages in breaking the algebraic problem and the IMP-PA security of ID, then it results in $\epsilon_{\mathrm{id}} \approx \sqrt{\epsilon_{\mathrm{alg}}}$. If ϵ_{sig} is the advantage in breaking UF security of DS, combined with the above, we have $\epsilon_{\mathrm{sig}} \approx q\sqrt{\epsilon_{\mathrm{alg}}}$.

Approach. We see from the above that a tight overall reduction $\mathrm{P_{sig}}{\rightarrow}\mathrm{P_{alg}}$ requires that the $\mathrm{P_{sig}}{\rightarrow}\mathrm{P_{id}}$ and $\mathrm{P_{id}}{\rightarrow}\mathrm{P_{alg}}$ reductions *both* be tight. What we observe is that we have a degree of freedom in achieving this, namely *the choice of the security goal* $\mathrm{P_{id}}$ *for the identification scheme.* Our hope is to pick $\mathrm{P_{id}}$ such that (1) We can give (new) transforms **Id2Sig** for which $\mathrm{P_{sig}}{\rightarrow}\mathrm{P_{id}}$ is tight, and simultaneously (2) We can give identification schemes such that $\mathrm{P_{id}}{\rightarrow}\mathrm{P_{alg}}$ is tight. We view these as two pillars of an edifice and are able to provide both via our definitions of security of identification under constrained impersonation coupled with some new id-to-sig transforms. We first pause to discuss some prior work, but a peek at Fig. 1 gives an outline of the results we will expand on later.

Prior work. The first proofs of security for **FS**-based signatures [28] reduced UF security of the **FS**-derived signature scheme directly to the hardness of the algebraic problem $\mathrm{P_{alg}}$, assuming H is a random oracle [8]. These proofs exploit forking lemmas [28, 6, 4]. Modular proofs of the form discussed above, that use identification as an intermediate step, begin with [26, 1]. The modular approach has many advantages. One is that since the id-to-sig transforms are generic, we have only to design and analyze identification schemes. Another is the better understanding and isolation of the role of random oracles: they are used by **Id2Sig** but not in the identification scheme. We have accordingly adopted this approach. Note that both the direct (forking lemma based) and the AABN-based indirect (modular) approach result in reductions of the same looseness we discussed above. Our (alternative but still modular) approaches will remove this loss.

Consideration of reduction tightness for signatures begins with BR [9], whose PSS scheme has a tight reduction to the RSA problem. GJ [17] and KW [21] give signature schemes with tight reductions to the Diffie-Hellman problem. The lack of tightness of the overall reduction for **FS**-based signatures is well recognized as an important problem and drawback. It was first addressed by Micali and Reyzin [24], who give a particular signature scheme with a tight reduction to factoring. It is obtained from a particular identification scheme via a method they call "swap". ABN [2] say that the method generalizes to other factoring-based schemes. However, "swap" has never been stated as a general transform of an identification scheme into a signature scheme; it appears rather as an ad hoc technique to go directly and tightly from the algebraic problem to the signature. This lack of abstraction is perhaps due in part to a lack of definitions, and the ones we provide allow us to fill the gap. In Section 5 we elevate the swap method to a general **Swap** transform, characterize the identification schemes to which it applies, and prove that, when it applies, it gives a tight $\mathrm{P_{sig}}{\rightarrow}\mathrm{P_{id}}$ reduction.

ABP [2] show a tight reduction of **FS**-derived GQ signatures to the Φ-hiding assumption of [12]. In contrast, our methods will yield GQ signatures with a tight reduction to the standard one-wayness of RSA. AFLT [3] use a slight variant of the Fiat-Shamir transform to turn lossy identification schemes into signature schemes with security based tightly on key indistinguishability, resulting in signature schemes with tight reductions to the decisional short discrete logarithm problem, the shortest vector problem in ideal lattices and subset sum.

Constrained impersonation. Recall our goal is to define a notion of identification security $\mathrm{P_{id}}$

$$\begin{tikzcd}
& \text{CIMP-CC} \arrow[dl] \arrow[dr] & \\
\boxed{\text{CIMP-UC}} \arrow[rr, bend right, "\shortmid" marking] \arrow[dr, shift left] & & \text{CIMP-CU} \arrow[ll, bend right, "\shortmid" marking] \arrow[dl, shift right] \\
& \boxed{\text{CIMP-UU}} \arrow[ul, shift left, "\shortmid" marking] \arrow[ur, shift right, "\shortmid" marking] &
\end{tikzcd}$$

P_{id}	**Id2Sig** Transform	$\mathrm{P}_{\mathrm{sig}}$-secure **Signature**	**Reductions** $\mathrm{P}_{\mathrm{sig}} \to \mathrm{P}_{\mathrm{id}} / \mathrm{P}_{\mathrm{id}} \to \mathrm{P}_{\mathrm{alg}}$
CIMP-CU	**FS**	$(Y, z) : \ c = \mathrm{H}(Y \| m)$	Tight/Loose
CIMP-UC	**MdCmt**	$(c, z) : \ Y = \mathrm{H}(m) \ ; \ c = \$$	Tight/Tight
CIMP-UU	**MdCmtCh**	$(s, z) : \ s = \$ \ ; \ Y = \mathrm{H}_1(m \| s) \ ; \ c = \mathrm{H}_2(m)$	Tight/Tight
CIMP-CC	**MdCh**	$(c, z) : \ Y = \$ \ ; \ c = \mathrm{H}(m) \| \$$	Tight/Unknown

Figure 1: **Top:** Relations between notions P_{id} of security for an identification scheme ID under constrained impersonation. Solid arrows denote implications, barred arrows denote separations. A solid box around a notion means a tight $\mathrm{P}_{\mathrm{id}} \to \mathrm{P}_{\mathrm{alg}}$ reduction for Sigma protocols; dotted means a loose one; no box means no known reduction. **Bottom:** Transforms of identification schemes into UUF signature schemes. The first column is the assumption P_{id} on the identification scheme.

---

such that (1) We can give transforms **Id2Sig** for which $\mathrm{P}_{\mathrm{sig}} \to \mathrm{P}_{\mathrm{id}}$ is tight, and (2) We can give identification schemes such that $\mathrm{P}_{\mathrm{id}} \to \mathrm{P}_{\mathrm{alg}}$ is tight. In fact our definitional goal is broader, namely to give a framework that allows us to understand and encompass both old and new transforms, the former including **FS** and **Swap**. We do all this with a definitional framework that we refer to as *constrained impersonation*. It yields four particular definitions denoted CIMP-XY for XY $\in \{\mathrm{CU}, \mathrm{UC}, \mathrm{UU}, \mathrm{CC}\}$. Each, in the role of P_{id}, will be the basis for an id-to-sig transform such that $\mathrm{P}_{\mathrm{sig}} \to \mathrm{P}_{\mathrm{id}}$ is tight, and two will allow $\mathrm{P}_{\mathrm{id}} \to \mathrm{P}_{\mathrm{alg}}$ to be tight.

In constrained impersonation we continue, as with IMP-PA, to allow a passive attack in which the adversary $\mathcal{A}$ against the identification scheme ID can obtain transcripts $Y_1 \| c_1 \| z_1, Y_2 \| c_2 \| z_2 \ldots$ of interactions between the honest prover and verifier. Then $\mathcal{A}$ tries to impersonate, meaning get the honest verifier to accept. If X=C then the *commitment* in this impersonation interaction is adversary-*chosen*, while if X=U (*unchosen*) it must be pegged to a commitment from one of the transcripts. If Y=C, the *challenge* is adversary-chosen, while if Y=U it is as usual picked at random by the verifier. In all cases, multiple impersonation attempts are allowed. The formal definitions are in Section 3. CIMP-CU is a multi-impersonation version of IMP-PA, but the rest are novel.

What do any of these notions have to do with identification if one understands the latter as the practical goal of proving one's identity to a verifier? Beyond CIMP-CU, very little. In practice it is unclear how one can constrain a prover to only use, in impersonation, a commitment from a prior transcript. It is even more bizarre to allow a prover to pick the challenge. Our definitions however are not trying to capture any practical usage of identification. They view the latter as an analytical tool, an intermediate land allowing a smooth transition from an algebraic problem to signatures. The constrained impersonation notions work well in this regard, as we will see, both to explain and understand existing work and to obtain new signature schemes with tight reductions.

Relations between the four notions of constrained impersonation are depicted in Fig. 1. An arrow A $\to$ B is an implication: *Every* identification scheme that is A-secure is also B-secure. A barred arrow A $\not\rightarrow$ B is a separation: *There exists* an identification scheme that is A-secure but *not* B-secure. (For now ignore the boxes around notions.) In particular we see that CIMP-UU is weaker than, and CIMP-UC incomparable to, the more standard CIMP-CU.

Auxiliary definitions and tools. Before we see how to leverage the constrained impersonation framework, we need a few auxiliary definitions and results that, although simple, are, we believe, of independent interest and utility.

We define a signature scheme to be UUF (Unique Unforgeable) if it is UF with the restriction that a message can be signed at most once. (You are not allowed to twice ask the signing oracle to sign a particular m.)

It turns out that id-to-sig transforms naturally achieve UUF, not UF. However there are simple, generic transforms of UUF signature scheme into a UF ones —succinctly, UF$\rightarrow$UUF— that do not introduce much overhead and have tight reductions. One is to remove randomness, and the other is to add it. In more detail, the first transform, **DR**, de-randomizes the signature scheme (so that it always returns the same signature when called on the same message), by the classic method of deriving the signing coins by hashing the secret key along with the message [25, 22], while the second transform, **AR**, appends a random salt to the message before signing and includes the salt in the signature. For completeness we provide the transforms, theorem statements and proofs in Appendix A. We stress that the reductions are tight in both cases, so this step does not impact overall tightness. Now we can take (the somewhat easier to achieve) UUF as our goal.

Recall that in an identification scheme, the prover uses private randomness y to generate its commitment Y. We are interested in a special class of identification schemes that we call *trapdoor*. This means the prover can pick the commitment Y directly at random from the space of commitments and then compute the associated private randomness y using its secret key via a prescribed algorithm. A formal definition is in Section 3. Many existing identification schemes will meet our definition of being trapdoor modulo possibly some changes to the key structure. Thus the GQ scheme of [20] is trapdoor if we add the decryption exponent d to the secret key. With similar changes to the keys, the Fiat-Shamir [16] and Ong-Schnorr [27] identification schemes are trapdoor. The factoring-based identification scheme of [24] is also trapdoor. But not all identification schemes are trapdoor. One that is not is Schnorr's (discrete-log based) scheme [29].

Summary of results. For each notion $\mathrm{P_{id}} \in \{\text{CIMP-CU}, \text{CIMP-UC}, \text{CIMP-UU}, \text{CIMP-CC}\}$ we give an id-to-sig transform that turns any given $\mathrm{P_{id}}$-secure identification scheme ID into a $\mathrm{P_{sig}}$ = UUF signature scheme DS. *The reduction* $\mathrm{P_{sig}}\rightarrow\mathrm{P_{id}}$ *is tight in all four cases.* (To further make the signature scheme UF secure, we can apply the above-mentioned UF$\rightarrow$UUF transforms while preserving tightness.) The table in Fig. 1 summarizes the results and the transforms. They are discussed in more detail below and then fully in Section 4.

This is one pillar of the edifice, and not useful by itself. The other pillar is the $\mathrm{P_{id}}\rightarrow\mathrm{P_{alg}}$ reduction. In the picture at the top of Fig. 1, a solid-line box around $\mathrm{P_{id}}$ means that the reduction $\mathrm{P_{id}}\rightarrow\mathrm{P_{alg}}$ is tight, a dotted-line box indicates a reduction is possible but is not tight, and no box means no known reduction. These results assume the identification scheme is a Sigma protocol, as most are. We see that two points of our framework can be *tightly* obtained from the algebraic problem, so that in these cases the overall $\mathrm{P_{sig}}\rightarrow\mathrm{P_{alg}}$ reduction is tight, which was the ultimate goal. See below and Section 6.3 for more.

More details on results. The transform from CIMP-CU is the classical **FS** one. The reduction is now tight, even though it was not from IMP-PA [1], simply because CIMP-CU allows multiple impersonation attempts. The proof follows easily from [1] and we don't give it here. In this case our framework serves to better understand and articulate something implicit in the literature rather than deliver anything new. For CIMP-UC, we give a transform called **MdCmt**, for "Message-Derived Commitment", where, to sign m, the signer computes the commitment Y as a hash of the message, picks a challenge at random, uses its trapdoor to compute the coins y corresponding to Y, uses y and the secret key to compute a response z, and returns the challenge and response as the signature. For CIMP-UU, the weakest of the four notions, our transform **MdCmtCh**, for "Message-Derived Commitment and Challenge", has the signer compute the commitment Y as a hash of the message and a random seed s, the commitment as a hash of the message, returning as signature the seed and response, the latter computed as before. In the CIMP-CC case, **MdCh** ("Message-Derived Challenge") has the signer pick a random commitment and produce a challenge half of which is the hash of the message and the other half is random. In the CIMP-UC and CIMP-UU cases, the identification scheme needs to be trapdoor. Again the salient fact is that the reductions underlying all four transforms are tight. See Section 4 for more information.

The $\mathrm{P}_{\mathrm{id}} \rightarrow \mathrm{P}_{\mathrm{alg}}$ reduction reduces P_{id} = CIMP-XY security of identification schemes that are Sigma protocols to $\mathrm{P}_{\mathrm{alg}}$, where the latter is KR, the problem of recovering the secret key given only the public key, which is the algebraic problem whose hardness is assumed. We show in Section 6.3 that this reduction is tight for $\mathrm{XY} \in \{\mathrm{UC}, \mathrm{UU}\}$, making these the most attractive starting points. For XY = CU we must use the Reset Lemma [7] so the reduction is loose. CIMP-CC is a very strong notion and we do not have a reduction of this type for it. We can show that it is achievable, however.

Swap. As indicated above, our framework allows us to generalize the swap method of [24] into an id-to-sig transform **Swap** and understand and characterize what it does. In Section 5 we present **Swap** as a generic transform of a trapdoor identification scheme ID to a signature scheme that is just like **MdCmt** (cf. row 2 of the table of Fig. 1) except that the challenge c is included in the input to the hash function. Recall that **MdCmt** turns a CIMP-UC identification scheme into a UUF signature scheme. We can thence get a UF signature scheme by applying the **AR** transform of Appendix A.2. **Swap** is a shortcut, or optimization, of this two step process: it directly turns a CIMP-UC identification scheme into a UF signature scheme by effectively re-using the randomness of **MdCmt** in **AR**. See Section 5 for details.

2 Notation and basic definitions

Notation. We let ε denote the empty string. If X is a finite set, we let $x \leftarrow\$ X$ denote picking an element of X uniformly at random and assigning it to x. Algorithms may be randomized unless otherwise indicated. Running time is worst case. If A is an algorithm, we let $y \leftarrow A(x_1, \ldots; r)$ denote running A with random coins r on inputs $x_1, \ldots$ and assigning the output to y. We let $y \leftarrow\$ A(x_1, \ldots)$ be the result of picking r at random and letting $y \leftarrow A(x_1, \ldots; r)$. We let $[A(x_1, \ldots)]$ denote the set of all possible outputs of A when invoked with inputs $x_1, \ldots$. We use the code based game playing framework of [10]. (See Fig. 2 for an example.) By Pr[G] we denote the event that the execution of game G results in the game returning true. We also adopt the convention that the running time of an adversary refers to the worst case execution time of the game with the adversary. This means that the time taken for oracles to compute replies to queries is included.

We expand on our notation and treatment of random oracles in these games since it is a bit unusual. In our constructions, we will need random oracles with different ranges. For example we may want one random oracle returning points in a group $\mathbb{Z}_N^*$ and another returning strings of some length k. To provide a single unified definition, we have the procedure H in the games take not just the input x but a description Rng of the set from which outputs are to be drawn at random. Thus $y \leftarrow\$ \mathrm{H}(x, \mathbb{Z}_N^*)$ will return a random element of $\mathbb{Z}_N^*$, and so on. If $\mathrm{Rng}_1, \mathrm{Rng}_2$ are

```
Game $\mathbf{G}^{\mathrm{uf}}_{\mathsf{DS}}(\mathcal{A})$ / $\boxed{\mathbf{G}^{\mathrm{uuf}}_{\mathsf{DS}}(\mathcal{A})}$
$M \leftarrow \emptyset$ ; $(vk, sk) \leftarrow\$ \mathsf{DS.Kg}^{\mathrm{H}}$
$(m, \sigma) \leftarrow\$ \mathcal{A}^{\textsc{Sign},\mathrm{H}}(vk)$
If $m \in M$: Return false
Return $\mathsf{DS.Vf}^{\mathrm{H}}(vk, m, \sigma)$

$\mathrm{H}(x, \mathrm{Rng})$
If not $\mathrm{HT}[x, \mathrm{Rng}]$: $\mathrm{HT}[x, \mathrm{Rng}] \leftarrow\$ \mathrm{Rng}$
Return $\mathrm{HT}[x, \mathrm{Rng}]$

$\textsc{Sign}(m)$
$\boxed{\text{If } m \in M\text{: Return } \perp}$
$\sigma \leftarrow\$ \mathsf{DS.Sig}^{\mathrm{H}}(vk, sk, m)$
$M \leftarrow M \cup \{m\}$
Return $\sigma$
```

Figure 2: Games defining unforgeability and unique-unforgeability of signature scheme DS. Game $\mathbf{G}^{\mathrm{uuf}}_{\mathsf{DS}}(\mathcal{A})$ includes the boxed code and game $\mathbf{G}^{\mathrm{uf}}_{\mathsf{DS}}(\mathcal{A})$ does not.

different descriptions (of different sets, or even of the same set) then $\mathrm{H}(\cdot, \mathrm{Rng}_1)$ and $\mathrm{H}(\cdot, \mathrm{Rng}_2)$ are independent random oracles with the indicated range sets. In instantiations, the range description would be included as an input to the hash function to realize this independence. (See again Fig. 2 for an example.) In some figures we use shorthand $\mathrm{H}(\cdot)$ with the range indicated in the caption of the figure (such as in Fig. 5).

Signatures. In a signature scheme DS, the signer generates signing key sk and verifying key vk via $(vk, sk) \leftarrow\$ \mathsf{DS.Kg}^{\mathrm{H}}$ where H is the random oracle, the latter with syntax as discussed above. Now it can compute a signature $\sigma \leftarrow\$ \mathsf{DS.Sig}^{\mathrm{H}}(vk, sk, m)$ on any message $m \in \{0,1\}^*$. A verifier can deterministically compute a boolean $v \leftarrow \mathsf{DS.Vf}^{\mathrm{H}}(vk, m, \sigma)$ indicating whether or not σ is a valid signature of m relative to vk. Correctness as usual requires that $\mathsf{DS.Vf}^{\mathrm{H}}(vk, m, \mathsf{DS.Sig}^{\mathrm{H}}(vk, sk, m)) = \mathsf{true}$ with probability one. Game $\mathbf{G}^{\mathrm{uf}}_{\mathsf{DS}}(\mathcal{A})$ associated to DS and adversary $\mathcal{A}$ as per Fig. 2 captures the classical unforgeability notion of [18] lifted to the ROM as per [8], and we let $\mathbf{Adv}^{\mathrm{uf}}_{\mathsf{DS}}(\mathcal{A}) = \Pr[\mathbf{G}^{\mathrm{uf}}_{\mathsf{DS}}(\mathcal{A})]$ be the UF-advantage of $\mathcal{A}$. The same figure also defines game $\mathbf{G}^{\mathrm{uuf}}_{\mathsf{DS}}(\mathcal{A})$ to capture *unique* unforgeability. The difference is the inclusion of the boxed code, which disallows $\mathcal{A}$ from getting more than one signature on the same message. We let $\mathbf{Adv}^{\mathrm{uuf}}_{\mathsf{DS}}(\mathcal{A}) = \Pr[\mathbf{G}^{\mathrm{uuf}}_{\mathsf{DS}}(\mathcal{A})]$ be the UUF-advantage of $\mathcal{A}$. In both games we assume (w.l.o.g.) that before requesting a signature or attempting a forgery the adversary poses all corresponding H queries that it could predict.

Of course, UF implies UUF, meaning any signature scheme that is UF secure is also UUF secure. The converse is not true, meaning there exist UUF signature schemes that are not UF secure (we will see natural examples in this paper). In Appendix A we give simple, generic and tight ways to turn any given UUF signature scheme into a UF one. We note that unique unforgeability (UUF) should not be confused with unique signatures as defined in [19, 23]. In a unique signature scheme, there is, for any message, at most one signature the verifier will accept. If a unique signature scheme is UUF then it is also UF. But there are UUF (and UF) schemes that are not unique.

3 Constrained impersonation framework

We introduce a framework of definitions of identification schemes secure against constrained impersonation.

Identification. An identification (ID) scheme ID operates as depicted in Fig. 3. First, via $(ivk, isk, itk) \leftarrow\$ \mathsf{ID.Kg}$, the prover generates a public verification key ivk, private *identification key* isk, and trapdoor itk. Via $(Y, y) \leftarrow\$ \mathsf{ID.Cmt}(ivk)$ it generates *commitment* Y and corresponding private state y. The verifier sends a random challenge of length $\mathsf{ID.cl}$. The prover's *response* z and the verifier's boolean *decision* v are deterministically computed per $z \leftarrow \mathsf{ID.Rsp}(ivk, isk, c, y)$ and

Prover		**Verifier**
Input: ivk, isk		Input: ivk
$(Y, y) \leftarrow\$ \mathsf{ID.Cmt}(ivk)$	$\xrightarrow{Y}$	
	$\xleftarrow{c}$	$c \leftarrow\$ \{0,1\}^{\mathsf{ID.cl}}$
$z \leftarrow \mathsf{ID.Rsp}(ivk, isk, c, y)$	$\xrightarrow{z}$	$v \leftarrow \mathsf{ID.Vf}(ivk, Y\|c\|z)$

Figure 3: Functioning of an identification scheme ID.

```
Game $\mathbf{G}^{\text{cimp-xy}}_{\mathsf{ID}}(\mathcal{P})$
$\text{TT} \leftarrow \emptyset$ ; $\text{CT} \leftarrow \emptyset$ ; $i \leftarrow 0$ ; $j \leftarrow 0$
$(ivk, isk, itk) \leftarrow\$ \mathsf{ID.Kg}$
$(k, z) \leftarrow\$ \mathcal{P}^{\text{Tr},\text{Ch}}(ivk)$
If not $(1 \leq k \leq j)$: Return false
$T \leftarrow \text{CT}[k]\|z$
If $T \in \text{TT}$: Return false
Return $\mathsf{ID.Vf}(ivk, T)$

Tr()
$i \leftarrow i + 1$
$(Y_i, y_i) \leftarrow\$ \mathsf{ID.Cmt}(ivk)$
$c_i \leftarrow\$ \{0,1\}^{\mathsf{ID.cl}}$
$z_i \leftarrow \mathsf{ID.Rsp}(ivk, isk, c_i, y_i)$
$T \leftarrow Y_i\|c_i\|z_i$ ; $\text{TT} \leftarrow \text{TT} \cup \{T\}$
Return $T$

Ch($l$) // xy=uu
If not $(1 \leq l \leq i)$: Return $\perp$
$j \leftarrow j + 1$ ; $c \leftarrow\$ \{0,1\}^{\mathsf{ID.cl}}$
$\text{CT}[j] \leftarrow Y_l\|c$ ; Return $(j, c)$

Ch($l, c$) // xy=uc
If not $(1 \leq l \leq i)$: Return $\perp$
$j \leftarrow j + 1$
$\text{CT}[j] \leftarrow Y_l\|c$ ; Return $(j, c)$

Ch($Y$) // xy=cu
$j \leftarrow j + 1$ ; $c \leftarrow\$ \{0,1\}^{\mathsf{ID.cl}}$
$\text{CT}[j] \leftarrow Y\|c$ ; Return $(j, c)$

Ch($Y, c$) // xy=cc
$j \leftarrow j + 1$
$\text{CT}[j] \leftarrow Y\|c$ ; Return $(j, c)$
```

Figure 4: Games defining security of identification scheme ID against constrained impersonation under passive attack.

$v \leftarrow \mathsf{ID.Vf}(ivk, Y\|c\|z)$, respectively. We require perfect correctness. An example ID scheme is the GQ-ID; see Section 6.3. For basic ID schemes, the trapdoor plays no role; its use arises in trapdoor identification.

Trapdoor identification. We now explain what it means for an ID scheme to be *trapdoor*. Namely there is an algorithm $\mathsf{ID.Cmt}^{-1}$ that produces y from Y with the aid of the trapdoor itk. Formally, the outputs of the following two processes must be identically distributed. Both processes generate $(ivk, isk, itk) \leftarrow\$ \mathsf{ID.Kg}$. The first process then lets $(Y, y) \leftarrow\$ \mathsf{ID.Cmt}(ivk)$. The second process picks $Y \leftarrow\$ \mathsf{ID.CmtSp}(ivk)$ and lets $y \leftarrow\$ \mathsf{ID.Cmt}^{-1}(ivk, itk, Y)$. (Here $\mathsf{ID.CmtSp}(ivk)$ is the space of commitments associated to ID.) Both processes return (ivk, isk, itk, Y, y). In a trapdoor scheme no one can distinguish which process was used to generate the output.

Security against impersonation. Classically, the security goal for an identification scheme ID has been impersonation [15, 1]. The framework has two stages. First, the adversary, given ivk but not isk, attacks the honest, isk-using prover. Second, using the information it gathers in the first stage, it engages in an interaction with the verifier, attempting to impersonate the real prover by successfully identifying itself. In the second stage, the adversary, in the role of malicious prover, submits a commitment Y of its choice, receives an honest verifier challenge c, submits a response z of its choice, and wins if $\mathsf{ID.Vf}(ivk, Y\|c\|z) = \mathsf{true}$. A hierarchy of possible first-phase attacks is defined in [7]. In the context of conversion to signatures, the relevant one is the weakest, namely passive attacks, where the adversary is just an eavesdropper and gets honestly-generated protocol transcripts. This is the IMP-PA notion. (Active and even concurrent attacks are relevant in other contexts [7].) We note that in the second stage, the adversary is allowed only one interaction with the honest verifier.

Security against constrained impersonation. We introduce a new framework of goals for identification that we call *constrained impersonation.* There are two dimensions, the commitment dimension X and the challenge dimension Y, for each of which there are two choices, $\mathrm{X} \in \{\mathrm{C}, \mathrm{U}\}$ and $\mathrm{Y} \in \{\mathrm{C}, \mathrm{U}\}$, where C stands for *chosen* and U for *unchosen.* This results in four notions, CIMP-UU, CIMP-UC, CIMP-CU, CIMP-CC. It works as follows. The adversary is allowed a passive attack, namely the ability to obtain transcripts of interactions between the honest prover and the verifier. The choices pertain to the impersonation, when the adversary interacts with the honest verifier in an attempt to make it accept. When $\mathrm{X} = \mathrm{C}$, the adversary can send the verifier a commitment of its choice, as in classical impersonation. But when $\mathrm{X} = \mathrm{U}$, it cannot. Rather, it is required (constrained) to use a commitment that is from one of the transcripts it obtained in the first phase and thus in particular honestly generated. Next comes the challenge. If $\mathrm{Y} = \mathrm{U}$, this is chosen freshly at random, as in the classical setting, but if $\mathrm{Y} = \mathrm{C}$, the adversary actually gets to pick its own challenge. Regardless of choices made in these four configurations, to win the adversary must finally supply a correct response. And, also regardless of these choices, the adversary can mount *multiple* attempts to convince the verifier.

For choices $\mathrm{xy} \in \{\mathrm{uu}, \mathrm{uc}, \mathrm{cu}, \mathrm{cc}\}$ of parameters, the formalization considers game $\mathbf{G}^{\text{cimp-xy}}_{\mathsf{ID}}(\mathcal{P})$ of Fig. 4 associated to identification scheme ID, adversary $\mathcal{P}$, and lets $\mathbf{Adv}^{\text{cimp-xy}}_{\mathsf{ID}}(\mathcal{P}) = \Pr[\mathbf{G}^{\text{cimp-xy}}_{\mathsf{ID}}(\mathcal{P})]$. The transcript oracle $\textsc{Tr}$ returns upon each invocation a transcript of an interaction between the honest prover and verifier, allowing $\mathcal{P}$ to mount its passive attack, and is the same for all four games. The impersonation attempts are mounted through calls to the $\textsc{Ch}$ oracle, which creates a partial transcript $\mathrm{TT}[j]$ consisting of a commitment and a challenge, where j is a session id, and it returns the session id and challenge. Multiple impersonation attempts are captured by the adversary being allowed to call $\textsc{Ch}$ as often as it wants. Eventually the adversary outputs a session id k and a response z for session k, and wins if the corresponding transcript is both accepting and new, the latter meaning not one returned by the transcript oracle. In the UU case, $\mathcal{P}$ would give $\textsc{Ch}$ only an index l of an existing transcript already returned by $\textsc{Tr}$, and $\mathrm{TT}[j]$ consists of the commitment from the l-th transcript together with a fresh random challenge. In the UC case, $\textsc{Ch}$ takes in addition a challenge c chosen by the adversary, and $\mathrm{TT}[j]$ consists of the commitment from the l-th transcript together with this challenge. In CU, the adversary can specify the commitment but the challenge is honestly chosen, while in CC, it can specify both.

CIMP-CU is a multi-impersonation extension of the classical IMP-PA notion. The other notions are new, and all will be the basis of transforms of identification to signatures admitting *tight* security reductions. CIMP-CU captures a practical identification security goal. As discussed in Section 1, the other notions have no such practical interpretation. However we are not aiming to capture some practical form of identification. We wish to use identification only as an analytical tool in the design of signature schemes. For this purpose, as we will see, our framework and notions are indeed useful, allowing us to characterize past transforms and build new ones.

Relations. Fig. 1 shows the relations between the four CIMP-XY notions. The implications are straightforward; we now establish the separations.

CIMP-CU $\not\Rightarrow$ CIMP-UC as follows. Start with any CIMP-CU scheme; we will modify it so that it remains CIMP-CU-secure but is not CIMP-UC-secure. Distinguish a single challenge $c^* \in \{0,1\}^{\mathsf{ID.cl}}$, e.g., $c^* = 0^{\mathsf{ID.cl}}$. Revise the verifier's algorithm so that it will accept any transcript with challenge c^*. This is still CIMP-CU-secure (albeit with probability decreased by $1/2^{\mathsf{ID.cl}}$) since, in the CIMP-CU game, challenges are picked uniformly at random for the adversary, so existence of a magic challenge is unlikely to be useful. This is manifestly not CIMP-UC-secure since there the adversary can use any challenge of its choice. CIMP-UU $\not\Rightarrow$ CIMP-UC for the same reason.

CIMP-UC $\not\Rightarrow$ CIMP-CU as follows. Start with any CIMP-UC scheme; again we will modify it so that it remains CIMP-UC-secure but is not CIMP-CU-secure. This time, distinguish a single commitment Y^*: one way of doing this is for $\mathsf{ID.Kg}$ to sample $Y^* \leftarrow\$ \mathsf{ID.CmtSp}(ivk)$ and include Y^* in the public key ivk; another is to agree for example that $(Y^*, y^*) \leftarrow \mathsf{ID.Cmt}(ivk; 0^l)$ where l is the number of random bits required by $\mathsf{ID.Cmt}$. Revise the verifier's algorithm so that it will accept any transcript with commitment Y^*. This is still CIMP-UC-secure (albeit with probability decreased by $1/|\mathsf{ID.CmtSp}(ivk)|$) since, in the CIMP-UC game, commitments are generated randomly for the adversary, so existence of a magic commitment is unlikely to be useful. This is manifestly not CIMP-CU-secure since there the adversary can use any commitment of its choice. CIMP-UU $\not\Rightarrow$ CIMP-CU for the same reason.

Finally, CIMP-UC $\not\Rightarrow$ CIMP-CC and CIMP-CU $\not\Rightarrow$ CIMP-CC since otherwise, by transitivity in Fig. 1, we would contradict the separation between CIMP-UC and CIMP-CU.

4 Signatures from identification

We specify our three new transforms of identification schemes to signature schemes, namely the ones of rows 2,3,4 of the table of Fig. 1. For each, we give a security proof based on the assumption P_{id} listed in the 1st column of the corresponding row of the table, so that we give transforms from CIMP-UC, CIMP-UU and CIMP-CC. It turns out that these transforms naturally achieve UUF rather than UF, and this is what we prove, with tight reductions of course. The transformation UF$\rightarrow$UUF can be done at the level of signatures, not referring to identification, in generic and simple ways, and also with tight reductions, as detailed in Appendix A. We thus get UF-secure signatures with tight reductions to each of CIMP-UC, CIMP-UU and CIMP-CC.

4.1 From CIMP-UC identification to UUF signatures: MdCmt

MdCmt transforms a CIMP-UC ID scheme to a UUF signature scheme using message-dependent commitments.

THE CONSTRUCTION. Let ID be a trapdoor identification scheme and $\mathsf{ID.cl}$ its challenge length. Our **MdCmt** (message-dependent commitment) transform associates to ID the signature scheme $\mathsf{DS} = \mathbf{MdCmt}[\mathsf{ID}]$. The algorithms of DS are defined in Fig. 5. Signatures are effectively identification transcripts, but the commitments are chosen in a particular way. Recall that with trapdoor ID schemes it is the same whether one executes $(Y, y) \leftarrow\$ \mathsf{ID.Cmt}$ directly, or samples $Y \leftarrow\$ \mathsf{ID.CmtSp}(ivk)$ followed by computing $y \leftarrow\$ \mathsf{ID.Cmt}^{-1}(Y)$. Our construction exploits this: To each message m it assigns an individual commitment $Y \leftarrow \mathrm{H}(m)$, where H is a random oracle with range $\mathsf{ID.CmtSp}(ivk)$. The signing algorithm, using the trapdoor, completes this commitment to a transcript (Y, c, z) and outputs the pair c, z as the signature. Verification then consists of recomputing Y from m and invoking the verification algorithm of the ID scheme.

UNFORGEABILITY. The following theorem establishes that the (unique) unforgeability of a signature scheme constructed with **MdCmt** tightly reduces to the CIMP-UC security of the underlying ID scheme, in the random oracle model. The intuition of the proof is as follows: For answering a random oracle query $\mathrm{H}(m)$ on an unseen message, the reduction requests a fresh identification transcript (Y_i, c_i, z_i) from its TR oracle and programs H such that it maps m_i to Y_i. A signature

```
$\underline{\mathsf{DS.Kg}^{\mathrm{H}}}$
$(ivk, isk, itk) \leftarrow_\$ \mathsf{ID.Kg}$
$vk \leftarrow ivk$ ; $sk \leftarrow (isk, itk)$
Return $(vk, sk)$

$\underline{\mathsf{DS.Vf}^{\mathrm{H}}(vk, m, \sigma)}$
$ivk \leftarrow vk$ ; $(c, z) \leftarrow \sigma$
$Y \leftarrow \mathrm{H}(m)$
Return $\mathsf{ID.Vf}(ivk, Y \| c \| z)$

$\underline{\mathsf{DS.Sig}^{\mathrm{H}}(vk, sk, m)}$
$ivk \leftarrow vk$ ; $(isk, itk) \leftarrow sk$
$Y \leftarrow \mathrm{H}(m)$
$y \leftarrow_\$ \mathsf{ID.Cmt}^{-1}(ivk, itk, Y)$
$c \leftarrow_\$ \{0,1\}^{\mathsf{ID.cl}}$
$z \leftarrow \mathsf{ID.Rsp}(ivk, isk, c, y)$
$\sigma \leftarrow (c, z)$
Return $\sigma$
```

Figure 5: The construction of signature scheme $\mathsf{DS} = \mathbf{MdCmt}[\mathsf{ID}]$ from trapdoor identification scheme ID. By $\mathrm{H}(\cdot)$ we denote random oracle $\mathrm{H}(\cdot, \mathsf{ID.CmtSp}(ivk))$ with range $\mathsf{ID.CmtSp}(ivk)$.

query on m_i can subsequently be answered by releasing the pair (c_i, z_i). (This works only once per message, yielding precisely UUF unforgeability!) Finally, when the adversary outputs a forgery (c, z) on a new message m_i, with overwhelming probability we have $c \neq c_i$, as no information on c_i was exposed before.

Theorem 1 *Let signature scheme* $\mathsf{DS} = \mathbf{MdCmt}[\mathsf{ID}]$ *be obtained from trapdoor identification scheme* ID *as in Fig. 5. Let* $\mathcal{A}$ *be a* UUF*-adversary against* DS*. Suppose the number of queries that* $\mathcal{A}$ *makes to its* H *oracle is* q_h *and let* $N = \min |\mathsf{ID.CmtSp}(ivk)|$ *where the minimum is over all* $(ivk, isk, itk) \in [\mathsf{ID.Kg}]$*. Then from* $\mathcal{A}$ *we construct a* CIMP-UC *adversary* $\mathcal{P}$ *such that*

$$\mathbf{Adv}^{\mathrm{uuf}}_{\mathsf{DS}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{cimp\text{-}uc}}_{\mathsf{ID}}(\mathcal{P}) + \frac{q_h^2}{2N} + \frac{1}{2^{\mathsf{ID.cl}}} \ . \tag{1}$$

Adversary $\mathcal{P}$ *makes* q_h *queries to* Tr *and one query to* Ch *and has running time about that of* $\mathcal{A}$*.*

Proof of Theorem 1: Consider games $\mathrm{G}_0, \ldots, \mathrm{G}_4$ from Fig. 6. Game G_0 is precisely the UUF experiment (cf. Fig. 2) with the algorithms of $\mathbf{MdCmt}[\mathsf{ID}]$ plugged in. (Recall we assume the adversary, before requesting a signature or attempting a forgery, to pose all corresponding H queries.) We have $\Pr[\mathrm{G}_0] = \mathbf{Adv}^{\mathrm{uuf}}_{\mathsf{DS}}(\mathcal{A})$.

In G_0, the code implementing the H oracle generates hash values by sampling uniformly at random from $\mathsf{ID.CmtSp}(ivk)$. Game G_1 is like G_0 but it aborts if any two of these values collide (set YT is used to keep track of that). As G_0 and G_1 are identical unless such a collision happens, we have $\Pr[\mathrm{G}_0] \leq \Pr[\mathrm{G}_1] + (0 + \ldots + (q_h - 1))/|\mathsf{ID.CmtSp}(ivk)| \leq \Pr[\mathrm{G}_1] + (q_h^2/2)/N$.

In G_1, the crucial code of Sign is executed at most once per message. As by assumption each Sign query is preceded by a H query on the same message, the signatures output by Sign can be precomputed by the H oracle. This is done in Game G_2, in which table ST helps identifying for each signed message the corresponding H query. We have $\Pr[\mathrm{G}_1] = \Pr[\mathrm{G}_2]$.

In the H implementation of G_2, first $Y_i \leftarrow_\$ \mathsf{ID.CmtSp}(ivk)$ is sampled and then $y_i \leftarrow_\$ \mathsf{ID.Cmt}^{-1}(ivk, itk, Y_i)$. The definition of a trapdoor ID scheme allows us to replace this by the single instruction $(Y_i, y_i) \leftarrow_\$ \mathsf{ID.Cmt}(ivk)$. We obtain Game G_3 and $\Pr[\mathrm{G}_2] = \Pr[\mathrm{G}_3]$.

In Game G_4 an added abort condition catches the case where the adversary forges a signature on a message using as the challenge c the value c_l picked inside the H oracle for that message. As c_l is uniformly distributed in $\{0,1\}^{\mathsf{ID.cl}}$ and by $m \notin M$ no information about it is ever leaked to the adversary, the probability that $c = c_l$ is precisely $1/2^{\mathsf{ID.cl}}$. As G_3 and G_4 are identical unless this event occurs, we have $\Pr[\mathrm{G}_3] \leq \Pr[\mathrm{G}_4] + 1/2^{\mathsf{ID.cl}}$.

```
Game $\mathrm{G}_0$, $\mathrm{G}_1$, $\mathrm{G}_2$, $\mathrm{G}_3$, $\mathrm{G}_4$
$M \leftarrow \emptyset$ ; $\mathrm{ST} \leftarrow \emptyset$ ; $\mathrm{YT} \leftarrow \emptyset$ ; $i \leftarrow 0$
$(ivk, isk, itk) \leftarrow\!\$\ \mathsf{ID.Kg}$
$(m, \sigma) \leftarrow\!\$\ \mathcal{A}^{\textsc{Sign},\mathrm{H}}(ivk)$
If $m \in M$: Abort
$(c, z) \leftarrow \sigma$
$Y \leftarrow \mathrm{HT}[m]$                                  // $\mathrm{G}_0$–$\mathrm{G}_1$
$l \leftarrow \mathrm{ST}[m]$ ; $Y \leftarrow Y_l$                      // $\mathrm{G}_2$–$\mathrm{G}_4$
If $c = c_l$: Abort                                      // $\mathrm{G}_4$
Return $\mathsf{ID.Vf}(ivk, Y \| c \| z)$

$\mathrm{H}(m)$
If $\mathrm{HT}[m]$: Return $\mathrm{HT}[m]$
$i \leftarrow i + 1$ ; $\mathrm{ST}[m] \leftarrow i$
$Y_i \leftarrow\!\$\ \mathsf{ID.CmtSp}(ivk)$                          // $\mathrm{G}_0$–$\mathrm{G}_2$
$y_i \leftarrow\!\$\ \mathsf{ID.Cmt}^{-1}(ivk, itk, Y_i)$                  // $\mathrm{G}_2$
$(Y_i, y_i) \leftarrow\!\$\ \mathsf{ID.Cmt}(ivk)$                        // $\mathrm{G}_3$–$\mathrm{G}_4$
$c_i \leftarrow\!\$\ \{0,1\}^{\mathsf{ID.cl}}$                              // $\mathrm{G}_2$–$\mathrm{G}_4$
$z_i \leftarrow \mathsf{ID.Rsp}(ivk, isk, c_i, y_i)$                    // $\mathrm{G}_2$–$\mathrm{G}_4$
If $Y_i \in \mathrm{YT}$: Abort                                // $\mathrm{G}_1$–$\mathrm{G}_4$
$\mathrm{YT} \leftarrow \mathrm{YT} \cup \{Y_i\}$                           // $\mathrm{G}_1$–$\mathrm{G}_4$
$\mathrm{HT}[m] \leftarrow Y_i$
Return $\mathrm{HT}[m]$
```

```
$\textsc{Sign}(m)$
If $m \in M$: Return $\perp$
$Y \leftarrow \mathrm{HT}[m]$                                  // $\mathrm{G}_0$–$\mathrm{G}_1$
$y \leftarrow\!\$\ \mathsf{ID.Cmt}^{-1}(ivk, itk, Y)$                    // $\mathrm{G}_0$–$\mathrm{G}_1$
$c \leftarrow\!\$\ \{0,1\}^{\mathsf{ID.cl}}$                               // $\mathrm{G}_0$–$\mathrm{G}_1$
$z \leftarrow \mathsf{ID.Rsp}(ivk, isk, c, y)$                        // $\mathrm{G}_0$–$\mathrm{G}_1$
$l \leftarrow \mathrm{ST}[m]$ ; $c \leftarrow c_l$ ; $z \leftarrow z_l$                 // $\mathrm{G}_2$–$\mathrm{G}_4$
$\sigma \leftarrow (c, z)$
$M \leftarrow M \cup \{m\}$ ; Return $\sigma$
```

Figure 6: Games used in the proof of Theorem 1. Game G_0 is the UUF game with the algorithms of **MdCmt**[ID] plugged in. The Abort instruction terminates the execution of $\mathcal{A}$ and stops the game with output false.

To assess $\Pr[\mathrm{G}_4]$ we construct from $\mathcal{A}$ an adversary $\mathcal{P}$ against CIMP-UC and analyze its success probability. The code of $\mathcal{P}$ is given in Fig. 7. It internally executes $\mathcal{A}$ and mimics the environment otherwise provided by G_4. The general idea is to outsource the transcript generation that happens in H to the TR oracle, to process SIGN queries as in G_4, and to convert signature forgeries into transcript forgeries. Precisely, when $\mathcal{A}$ comes up with a forgery $m, (c, z)$, reduction $\mathcal{P}$ identifies the corresponding identification transcript $Y_l \| c_l \| z_l$ that it obtained from its own challenger as an answer to the TR query corresponding to m, forwards the forgery challenge c to its CH oracle, and finally outputs z. That is, it tries to be successful in the CIMP-UC game with transcript $Y_l \| c \| z$. Observe that this transcript is valid if and only if $\mathcal{A}$'s forgery is. Further, as Y_l is uniquely bound to m (since G_1) and $c \neq c_l$ (since G_4), the pair (Y_l, c) is different from all commitment-challenge pairs obtained from the TR oracle, i.e., the transcript is fresh. We thus have $\Pr[\mathrm{G}_4] = \mathbf{Adv}^{\text{cimp-uc}}_{\mathsf{ID}}(\mathcal{P})$. Overall, we obtain the bound from Equation (1). ▌

4.2 From CIMP-UU identification to UUF signatures: MdCmtCh

MdCmtCh transforms a CIMP-UU ID scheme to a UUF signature scheme using message-dependent commitments and challenges.

THE CONSTRUCTION. Let ID be a trapdoor identification scheme. Our **MdCmtCh** (message-dependent commitment and challenge) transform associates to ID and a seed length $\mathsf{sl} \in \mathbb{N}$ a signature scheme $\mathsf{DS} = \mathbf{MdCmtCh}[\mathsf{ID}, \mathsf{sl}]$. The algorithms of DS are defined in Fig. 8. Here we specify the commitment Y as a hash of the message alone, then use the trapdoor property to allow our signer to obtain $y \leftarrow\!\$\ \mathsf{ID.Cmt}^{-1}(ivk, itk, Y)$. We then specify the challenge as a randomized

```
Adversary $\mathcal{P}^{\text{Tr,Ch}}(ivk)$
$M \leftarrow \emptyset$ ; ST $\leftarrow \emptyset$ ; YT $\leftarrow \emptyset$ ; $i \leftarrow 0$
$(m, \sigma) \leftarrow_{\$} \mathcal{A}^{\text{Sign,H}}(ivk)$
If $m \in M$: Abort
$(c, z) \leftarrow \sigma$ ; $l \leftarrow \text{ST}[m]$
If $c = c_l$: Abort
$(j, c') \leftarrow \text{Ch}(l, c)$   // $j = 1 \wedge c' = c$
Return $(j, z)$

Sign$(m)$  // $\mathcal{P}$
If $m \in M$: Return $\perp$
$l \leftarrow \text{ST}[m]$ ; $\sigma \leftarrow (c_l, z_l)$
$M \leftarrow M \cup \{m\}$ ; Return $\sigma$

H$(m)$  // $\mathcal{P}$
If HT$[m]$: Return HT$[m]$
$i \leftarrow i + 1$ ; ST$[m] \leftarrow i$
$Y_i \| c_i \| z_i \leftarrow_{\$}$ Tr()
If $Y_i \in$ YT: Abort
YT $\leftarrow$ YT $\cup \{Y_i\}$
HT$[m] \leftarrow Y_i$
Return HT$[m]$
```

Figure 7: CIMP-UC adversary for proof of Theorem 1. The Abort instruction lets $\mathcal{P}$ terminate the execution of $\mathcal{A}$ and stop with $(\perp, \perp)$.

```
DS.Kg$^{\text{H}}$
$(ivk, isk, itk) \leftarrow_{\$}$ ID.Kg
$vk \leftarrow ivk$ ; $sk \leftarrow (isk, itk)$
Return $(vk, sk)$

DS.Vf$^{\text{H}}(vk, m, \sigma)$
$ivk \leftarrow vk$ ; $(z, s) \leftarrow \sigma$
$Y \leftarrow \text{H}_1(m)$
$c \leftarrow \text{H}_2(m \| s)$
Return ID.Vf$(ivk, Y \| c \| z)$

DS.Sig$^{\text{H}}(vk, sk, m)$
$s \leftarrow_{\$} \{0,1\}^{\mathsf{sl}}$
$ivk \leftarrow vk$ ; $(isk, itk) \leftarrow sk$
$Y \leftarrow \text{H}_1(m)$
$y \leftarrow_{\$}$ ID.Cmt$^{-1}(ivk, itk, Y)$
$c \leftarrow \text{H}_2(m \| s)$
$z \leftarrow$ ID.Rsp$(ivk, isk, c, y)$
$\sigma \leftarrow (z, s)$ ; Return $\sigma$
```

Figure 8: Our construction of signature scheme $\mathsf{DS} = \mathbf{MdCmtCh}[\mathsf{ID}, \mathsf{sl}]$ from a trapdoor identification scheme ID and a seed length $\mathsf{sl} \in \mathbb{N}$. By $\text{H}_1(\cdot)$ we denote random oracle $\text{H}(\cdot, \mathsf{ID.CmtSp}(ivk))$ with range $\mathsf{ID.CmtSp}(ivk)$ and by $\text{H}_2(\cdot)$ we denote random oracle $\text{H}(\cdot, \{0,1\}^{\mathsf{ID.cl}})$ with range $\{0,1\}^{\mathsf{ID.cl}}$.

hash of the message. (Unlike in the **FS** transform, the commitment is not hashed along with the message.) The randomization is captured by the seed s whose length sl was a parameter of our transform.

Unforgeability of our construction. The following shows that unique unforgeability of our signature tightly reduces to the CIMP-UU security of the underlying ID scheme. Standard unforgeability follows immediately (and tightly) by applying one of the UUF-to-UF transforms in Appendix A.

Theorem 2 *Let signature scheme* $\mathsf{DS} = \mathbf{MdCmtCh}[\mathsf{ID}, \mathsf{sl}]$ *be obtained from trapdoor identification scheme* ID *and seed length* sl *as in Fig. 8. Let* $\mathcal{A}$ *be a* UUF*-adversary against* DS*. Suppose the number of queries that* $\mathcal{A}$ *makes to its* H_1, H_2, Sign *oracles are, respectively,* q_1, q_2, q_s*. Then from* $\mathcal{A}$ *we can construct* CIMP-UU *adversary* $\mathcal{P}$ *such that*

$$\mathbf{Adv}^{\text{uuf}}_{\mathsf{DS}}(\mathcal{A}) \leq \mathbf{Adv}^{\text{cimp-uu}}_{\mathsf{ID}}(\mathcal{P}) + \frac{q_s(2q_2 + q_s - 1)}{2^{\mathsf{sl}+1}} . \quad (2)$$

Adversary $\mathcal{P}$ *makes* $q_1 + q_s + 1$ *queries to* Tr *and* q_2 *queries to* Ch*. It has running time about that of* $\mathcal{A}$*.*

```
Game $G_0$/$\boxed{G_1}$
$(ivk, isk, itk) \leftarrow\$ \mathsf{ID.Kg}$
$vk \leftarrow ivk$ ; $sk \leftarrow (isk, itk)$
$(m, \sigma) \leftarrow\$ \mathcal{A}^{\text{SIGN},\text{H}}(vk)$
Return $\mathsf{DS.Vf}^{\text{H}}(vk, m, \sigma)$

$\text{H}_1(x)$
If $\text{HT}_1[m]$: Return $\text{HT}_1[m]$
$\text{HT}_1[m] \leftarrow\$ \mathsf{ID.CmtSp}(ivk)$
Return $\text{HT}_1[m]$

$\text{H}_2(x)$
If $\text{HT}_2[m]$: Return $\text{HT}_2[m]$
$\text{HT}_2[m] \leftarrow\$ \{0,1\}^{\mathsf{ID.cl}}$
Return $\text{HT}_2[m]$

SIGN$(m)$
$s \leftarrow\$ \{0,1\}^{\mathsf{sl}}$
$ivk \leftarrow vk$ ; $(isk, itk) \leftarrow sk$
$Y \leftarrow \text{H}_1(m)$
$y \leftarrow\$ \mathsf{ID.Cmt}^{-1}(ivk, itk, Y)$
If (not $\text{HT}_2[m\|s]$):
   $\text{HT}_2[m\|s] \leftarrow\$ \{0,1\}^{\mathsf{ID.cl}}$
Else
   $\mathsf{bad} \leftarrow \mathsf{true}$
   $\boxed{\text{HT}_2[m\|s] \leftarrow\$ \{0,1\}^{\mathsf{ID.cl}}}$
$c \leftarrow \text{HT}_2[m\|s]$
$z \leftarrow \mathsf{ID.Rsp}(ivk, isk, c, y)$
$\sigma \leftarrow (z, s)$ ; Return $\sigma$
```

Figure 9: Games for proof of Theorem 2. Game G_1 includes the boxed code and game G_0 does not.

Proof of Theorem 2: We assume that $\mathcal{A}$ avoids certain pointless behavior that would only cause it to lose. Thus, we assume it did not query to SIGN the message m in the forgery (m, σ) that it eventually outputs. This means that the set M in game $\mathbf{G}^{\text{uuf}}_{\mathsf{DS}}(\mathcal{A})$, and the code and checks associated with it, are redundant and can be removed. We will work with this simplified form of the game. Throughout the proof, $\text{H}_1(\cdot)$ denotes $\text{H}(\cdot, \mathsf{ID.CmtSp}(ivk))$ and $\text{H}_2(\cdot)$ denotes $\text{H}(\cdot, \{0,1\}^{\mathsf{ID.cl}})$, and HT_1 and HT_2 denote the corresponding tables maintained by the code implementing random oracle H.

When procedure SIGN is replying to signing query m, it first computes Y and picks s. We would like that, at this point, it can define the table entry $\text{HT}_2[m\|s]$ without caring whether it was already defined. (This is to allow an eventual impersonation adversary to program this RO response with a challenge emanating from a transcript obtained from the transcript oracle.) In general, of course, this would be wrong, but intuitively the random choice of s means it is usually right. (This indeed is why we have the seed in the scheme.) To show this formally we consider the games G_0, G_1 of Fig. 9. Game G_0 excludes the boxed code, so that its SIGN procedure defines $\text{HT}_1[m\|s]$ only when this entry was not already defined, but game G_1 includes the boxed code, so that SIGN defines this entry always, as we would like. But these games are identical-until-bad [10], meaning differ only in code that follows the setting of the boolean flag bad to true. So we have $\mathbf{Adv}^{\text{uuf}}_{\mathsf{DS}}(\mathcal{A}) = \Pr[G_0]$. By the Fundamental Lemma of Game Playing of [10], we have $\Pr[G_0] = \Pr[G_1] + \Pr[G_0] - \Pr[G_1] \le \Pr[G_1] + \Pr[G_0 \text{ sets } \mathsf{bad}]$.

The random choice of s made by procedure SIGN ensures

$$\Pr[G_0 \text{ sets } \mathsf{bad}] \le \sum_{i=0}^{q_s-1} \frac{q_2 + i}{2^{\mathsf{sl}}} = \frac{q_s(2q_2 + q_s - 1)}{2^{\mathsf{sl}+1}} . \tag{3}$$

Now we need to bound $\Pr[G_1]$. Consider game G_2 of Fig. 10. Towards using CIMP-UU, this game refrains from using isk directly in procedure SIGN. Instead, it begins by generating conversation transcripts $Y_i\|c_i\|z_i$ and has SIGN use these. To make this possible, $\text{H}_1(\cdot)$ values are set to the transcript commitments. Then SIGN retrieves the corresponding commitment Y, sets $\text{HT}_2[m\|s]$ to the challenge from the same transcript, and puts the corresponding response in the signature. (Ind_1 is used to remember the mapping between hashes and transcripts.) Since the signatures are correctly distributed we have $\Pr[G_1] = \Pr[G_2]$.

```
Game G_2
$(ivk, isk, itk) \leftarrow_\$ \mathsf{ID.Kg}$
$vk \leftarrow ivk$
For $i = 1, \ldots, q_1 + q_s + 1$ do
    $(Y_i, y_i) \leftarrow_\$ \mathsf{ID.Cmt}(ivk)$
    $c_i \leftarrow_\$ \{0,1\}^{\mathsf{ID.cl}}$
    $z_i \leftarrow \mathsf{ID.Rsp}(ivk, isk, c_i, y_i)$
$i_1 \leftarrow 0$
$(m, \sigma) \leftarrow_\$ \mathcal{A}^{\text{SIGN},\text{H}}(vk)$
$(z, s) \leftarrow \sigma$
$Y \leftarrow \text{H}_1(m)$
$c \leftarrow \text{H}_2(m \| s)$
Return $\mathsf{ID.Vf}(ivk, Y \| c \| z)$

SIGN(m)
$s \leftarrow_\$ \{0,1\}^{\mathsf{sl}}$ ; $Y \leftarrow \text{H}_1(m)$
$i \leftarrow \text{Ind}_1(m)$ ; $\text{HT}_2[m \| s] \leftarrow c_i$
$\sigma \leftarrow (z_i, s)$ ; Return $\sigma$

H_1(x)
If $\text{HT}_1[m]$: Return $\text{HT}_1[m]$
$i_1 \leftarrow i_1 + 1$ ; $\text{HT}_1[x] \leftarrow Y_{i_1}$ ; $\text{Ind}_1(x) \leftarrow i_1$
Return $\text{HT}_1[x]$

H_2(x)
If $\text{HT}_2[m]$: Return $\text{HT}_2[m]$
$\text{HT}_2[m] \leftarrow_\$ \{0,1\}^{\mathsf{ID.cl}}$
Return $\text{HT}_2[m]$
```

Figure 10: Game G_2 for the proof of Theorem 2.

```
Adversary $\mathcal{P}^{\text{TR},\text{CH}}(ivk)$
$vk \leftarrow ivk$
For $i = 1, \ldots, q_1 + q_s + 1$:
    $(Y_i, c_i, z_i) \leftarrow_\$ \text{TR}()$
$i_1 \leftarrow 0$
$(m, \sigma) \leftarrow_\$ \mathcal{A}^{\text{SIGN},\text{H}}(vk)$
$(z, s) \leftarrow \sigma$
$Y \leftarrow \text{H}_1(m)$
$c \leftarrow \text{H}_2(m \| s)$
$j \leftarrow \text{Ind}_2(m \| s)$
Return $(j, z)$

SIGN(m) // $\mathcal{P}$
$s \leftarrow_\$ \{0,1\}^{\mathsf{sl}}$ ; $Y \leftarrow \text{H}_1(m)$
$i \leftarrow \text{Ind}_1(m)$ ; $\text{HT}_2[m \| s] \leftarrow c_i$
$\sigma \leftarrow (z_i, s)$ ; Return $\sigma$

H_1(x) // $\mathcal{P}$
If $\text{HT}_1[m]$: Return $\text{HT}_1[m]$
$i_1 \leftarrow i_1 + 1$ ; $\text{HT}_1[x] \leftarrow Y_{i_1}$ ; $\text{Ind}_1(x) \leftarrow i_1$
Return $\text{HT}_1[x]$

H_2(x) // $\mathcal{P}$
If $\text{HT}_2[m]$: Return $\text{HT}_2[m]$
$m \| s \leftarrow x$ ; $Y \leftarrow \text{H}_1(m)$ ; $l \leftarrow \text{Ind}_1(m)$
$(j, c) \leftarrow_\$ \text{CH}(l)$ ; $\text{Ind}_2(x) \leftarrow j$ ; $\text{HT}_2[x] \leftarrow c$
Return $\text{HT}_2[x]$
```

Figure 11: Adversary for proof of Theorem 2.

We build CIMP-UU adversary $\mathcal{P}$ so that $\Pr[\text{G}_2] \leq \mathbf{Adv}^{\text{cimp-uu}}_{\mathsf{ID}}(\mathcal{P})$. Game G_2 was crafted exactly to make the construction of adversary $\mathcal{P}$ quite direct. The construction is described in detail in Fig. 11. Adversary $\mathcal{P}$ has access to oracles TR, CH as per game $\mathbf{G}^{\text{cimp-uu}}_{\mathsf{ID}}(\mathcal{P})$ in which it is executing. It runs $\mathcal{A}$, simulating answers to $\mathcal{A}$'s queries to SIGN, H_1, and H_2 as shown. It obtains conversation transcripts using its TR oracle to play the role of the ones generated in G_2. Using these, SIGN can be simulated as per game G_2. Oracle $\text{H}_1(\cdot)$ is simulated as in G_2. When a query x is made to $\text{H}_1(\cdot)$, adversary $\mathcal{P}$ parses x as $m \| s$. Using mapping Ind_1, it retrieves the index l corresponding to m in the list of transcripts and submits this to its challenge oracle CH to get back a session id j and a challenge, and returns this challenge as the response to the oracle query. Finally when $\mathcal{A}$ produces a forgery, the session id corresponding to the message and seed in the forgery is retrieved via mapping Ind_2. Adversary $\mathcal{P}$ terminates by outputting the response in the forged signature as the impersonation for this session. We need to show that the impersonation is successful as long as the forgery was valid. A somewhat delicate point is that we use the fact that the message m in the forgery was not a SIGN query. This is what ensures that a CIMP-UU challenge session corresponding to the forgery conversation exists. ▌

```
$\underline{\mathsf{DS.Kg}^{\mathrm{H}}}$
$(ivk, isk, itk) \leftarrow\!\!\$\ \mathsf{ID.Kg}$
$vk \leftarrow ivk$ ; $sk \leftarrow isk$
Return $(vk, sk)$

$\underline{\mathsf{DS.Vf}^{\mathrm{H}}(vk, m, \sigma)}$
$ivk \leftarrow vk$ ; $(Y, c_2, z) \leftarrow \sigma$
$c_1 \leftarrow \mathrm{H}(m)$ ; $c \leftarrow c_1 \| c_2$
Return $\mathsf{ID.Vf}(ivk, Y \| c \| z)$

$\underline{\mathsf{DS.Sig}^{\mathrm{H}}(vk, sk, m)}$
$ivk \leftarrow vk$ ; $isk \leftarrow sk$
$(Y, y) \leftarrow\!\!\$\ \mathsf{ID.Cmt}(ivk)$
$c_1 \leftarrow \mathrm{H}(m)$
$c_2 \leftarrow\!\!\$\ \{0,1\}^{\mathsf{ID.cl}/2}$
$c \leftarrow c_1 \| c_2$
$z \leftarrow \mathsf{ID.Rsp}(ivk, isk, c, y)$
$\sigma \leftarrow (Y, c_2, z)$
Return $\sigma$
```

Figure 12: The construction of signature scheme $\mathsf{DS} = \mathbf{MdCh}[\mathsf{ID}]$ from identification scheme ID with even challenge length $\mathsf{ID.cl}$. By $\mathrm{H}(\cdot)$ we denote random oracle $\mathrm{H}(\cdot, \{0,1\}^{\mathsf{ID.cl}/2})$ with range $\{0,1\}^{\mathsf{ID.cl}/2}$.

4.3 From CIMP-CC identification to UUF signatures: MdCh

The **MdCmt** and **MdCmtCh** transformations described above crucially build on the trapdoor property of the underlying ID scheme. This is not the case for the **MdCh** construction described next. However, amongst the security notions for ID schemes that we defined, **MdCh** assumes the strongest one: CIMP-CC.

The construction. Let ID be an identification scheme and let its challenge length $\mathsf{ID.cl}$ be an even number. Our **MdCh** (message-dependent challenge) transform is defined in Fig. 12 and associates to ID the signature scheme $\mathsf{DS} = \mathbf{MdCh}[\mathsf{ID}]$. In a nutshell, signatures are identification transcripts where one half of the challenge is picked at random and the other half is derived from the message via a random oracle. Only the random part is included in the signature; the other part is recomputed by the verifier.

Unforgeability. As we prove below (with tight reduction), the **MdCh** construction yields a UUF-secure signature scheme if the underlying ID scheme offers CIMP-CC security. In the proof, a query to the H oracle on an unseen message m_i is answered by taking a fresh identification transcript (Y_i, c_i, z_i), splitting the challenge into two parts $c_{i,1}, c_{i,2}$, and using $c_{i,1}$ as the value assigned by the hash function. A signature query on m_i can subsequently be answered by releasing the remaining part of the transcript. Importantly, unless a signature query on m_i is posed, the second part of the challenge, $c_{i,2}$, is never revealed. Thus, with overwhelming probability, any challenge c induced by a forgery is not amongst the challenges c_i touched while processing the H queries. That is, a valid and fresh identification transcript can be put together by the reduction.

Theorem 3 *Let signature scheme $\mathsf{DS} = \mathbf{MdCh}[\mathsf{ID}]$ be obtained from identification scheme ID as in Fig. 12. Let $\mathcal{A}$ be a UUF-adversary against DS. Suppose the number of queries that $\mathcal{A}$ makes to its H oracle is q_h. Then from $\mathcal{A}$ we construct a CIMP-CC adversary $\mathcal{P}$ such that*

$$\mathbf{Adv}^{\mathrm{uuf}}_{\mathsf{DS}}(\mathcal{A}) \le \mathbf{Adv}^{\mathrm{cimp\text{-}cc}}_{\mathsf{ID}}(\mathcal{P}) + \frac{q_h^2 + 1}{2^{\mathsf{ID.cl}/2}} \ . \qquad (4)$$

Adversary $\mathcal{P}$ makes q_h queries to Tr *and one query to* Ch *and has running time about that of $\mathcal{A}$.*

Proof of Theorem 3: Consider games $\mathrm{G}_0, \ldots, \mathrm{G}_4$ from Fig. 13. Game G_0 is precisely the UUF experiment (cf. Fig. 2) with the algorithms of **MdCh**[ID] plugged in. (Recall we assume the adversary, before requesting a signature or attempting a forgery, to pose all corresponding H queries.) We have $\Pr[\mathrm{G}_0] = \mathbf{Adv}^{\mathrm{uuf}}_{\mathsf{DS}}(\mathcal{A})$.

```
Game $G_0$, $G_1$, $G_2$, $G_3$, $G_4$
$M \leftarrow \emptyset$ ; $ST \leftarrow \emptyset$ ; $CT \leftarrow \emptyset$ ; $i \leftarrow 0$
$(ivk, isk, itk) \leftarrow_\$ \mathsf{ID.Kg}$
$(m, \sigma) \leftarrow_\$ \mathcal{A}^{\text{SIGN},\text{H}}(ivk)$
If $m \in M$: Abort
$(Y, c_2, z) \leftarrow \sigma$
$c_1 \leftarrow \text{HT}[m]$                                  // $G_0$–$G_1$
$l \leftarrow \text{ST}[m]$ ; $c_1 \leftarrow c_{l,1}$                  // $G_2$–$G_4$
If $c_2 = c_{l,2}$: Abort                              // $G_3$–$G_4$
Return $\mathsf{ID.Vf}(ivk, Y \| c_1 \| c_2 \| z)$

H$(m)$
If HT$[m]$: Return HT$[m]$
$i \leftarrow i + 1$ ; ST$[m] \leftarrow i$
$(Y_i, y_i) \leftarrow_\$ \mathsf{ID.Cmt}(ivk)$                    // $G_2$–$G_4$
$c_{i,1} \leftarrow_\$ \{0,1\}^{\mathsf{ID.cl}/2}$                     // $G_0$–$G_3$
$c_{i,2} \leftarrow_\$ \{0,1\}^{\mathsf{ID.cl}/2}$                     // $G_2$–$G_3$
$c_i \leftarrow c_{i,1} \| c_{i,2}$                                // $G_2$–$G_3$
$c_i \leftarrow_\$ \{0,1\}^{\mathsf{ID.cl}}$                           // $G_4$
$c_{i,1} \| c_{i,2} \leftarrow c_i$                                // $G_4$
$z_i \leftarrow \mathsf{ID.Rsp}(ivk, isk, c_i, y_i)$                // $G_2$–$G_4$
If $c_{i,1} \in \text{CT}$: Abort                         // $G_1$–$G_4$
$\text{CT} \leftarrow \text{CT} \cup \{c_{i,1}\}$                          // $G_1$–$G_4$
HT$[m] \leftarrow c_{i,1}$
Return HT$[m]$
```

```
SIGN$(m)$
If $m \in M$: Return $\perp$
$(Y, y) \leftarrow_\$ \mathsf{ID.Cmt}(ivk)$                     // $G_0$–$G_1$
$c_1 \leftarrow \text{HT}[m]$                                 // $G_0$–$G_1$
$c_2 \leftarrow_\$ \{0,1\}^{\mathsf{ID.cl}/2}$                       // $G_0$–$G_1$
$c \leftarrow c_1 \| c_2$                                     // $G_0$–$G_1$
$z \leftarrow \mathsf{ID.Rsp}(ivk, isk, c, y)$                  // $G_0$–$G_1$
$l \leftarrow \text{ST}[m]$ ; $Y \leftarrow Y_l$                       // $G_2$–$G_4$
$c_2 \leftarrow c_{l,2}$ ; $z \leftarrow z_l$                         // $G_2$–$G_4$
$\sigma \leftarrow (Y, c_2, z)$
$M \leftarrow M \cup \{m\}$ ; Return $\sigma$
```

Figure 13: Games used in the proof of Theorem 3. Game G_0 is the UUF game with the algorithms of **MdCh**[ID] plugged in. The Abort instruction terminates the execution of $\mathcal{A}$ and stops the game with output value false.

In G_0, the code implementing the H oracle generates hash values by sampling uniformly at random from $\{0,1\}^{\mathsf{ID.cl}/2}$. Game G_1 is like G_0 but it aborts if any two of these values collide (set CT is used to keep track of that). As G_0 and G_1 are identical unless such a collision happens, we have $\Pr[G_0] \leq \Pr[G_1] + (0 + \ldots + (q_h - 1))/2^{\mathsf{ID.cl}/2} \leq \Pr[G_1] + q_h^2/2^{\mathsf{ID.cl}/2+1}$.

In G_1, the crucial code of SIGN is executed at most once per message. As by assumption each SIGN query is preceded by a H query on the same message, the signatures output by SIGN can be precomputed by the H oracle. This is done in Game G_2, in which table ST helps identifying for each signed message the corresponding H query. We have $\Pr[G_1] = \Pr[G_2]$.

In Game G_3 an added abort condition catches the case where the adversary forges a signature on a message using as the challenge component c_2 the value $c_{l,2}$ picked inside the H oracle for that message. As $c_{l,2}$ is uniformly distributed in $\{0,1\}^{\mathsf{ID.cl}/2}$ and by $m \notin M$ no information about it is ever leaked to the adversary, the probability that $c_2 = c_{l,2}$ is precisely $1/2^{\mathsf{ID.cl}/2}$. As G_2 and G_3 are identical unless this event occurs, we have $\Pr[G_2] \leq \Pr[G_3] + 1/2^{\mathsf{ID.cl}/2}$.

The transition to Game G_4 is a simple rewriting step: In the H oracle, the operation of randomly picking $c_{i,1}, c_{i,2}$ and concatenating them via $c_i \leftarrow c_{i,1} \| c_{i,2}$ is replaced by the equivalent operation of randomly picking c_i and decomposing it into $c_{i,1}$ and $c_{i,2}$. We have $\Pr[G_3] = \Pr[G_4]$.

To assess $\Pr[G_4]$ we construct from $\mathcal{A}$ an adversary $\mathcal{P}$ against CIMP-CC and analyze its success probability. The code of $\mathcal{P}$ is given in Fig. 14. It internally executes $\mathcal{A}$ and mimics the environment otherwise provided by G_4. The general idea is to outsource the transcript generation that happens

```
Adversary $\mathcal{P}^{\text{Tr,Ch}}(ivk)$
$M \leftarrow \emptyset$ ; $\text{ST} \leftarrow \emptyset$ ; $\text{CT} \leftarrow \emptyset$ ; $i \leftarrow 0$
$(m, \sigma) \leftarrow_\$ \mathcal{A}^{\text{Sign,H}}(ivk)$
If $m \in M$: Abort
$(Y, c_2, z) \leftarrow \sigma$
$l \leftarrow \text{ST}[m]$ ; $c_1 \leftarrow c_{l,1}$ ; $c \leftarrow c_1 \| c_2$
If $c_2 = c_{l,2}$: Abort
$(j, c') \leftarrow \text{Ch}(Y, c)$   // $j = 1 \wedge c' = c$
Return $(j, z)$

Sign$(m)$ // $\mathcal{P}$
If $m \in M$: Return $\bot$
$l \leftarrow \text{ST}[m]$ ; $\sigma \leftarrow (Y_l, c_{l,2}, z_l)$
$M \leftarrow M \cup \{m\}$ ; Return $\sigma$

H$(m)$ // $\mathcal{P}$
If HT$[m]$: Return HT$[m]$
$i \leftarrow i + 1$ ; ST$[m] \leftarrow i$
$Y_i \| c_i \| z_i \leftarrow_\$ \text{Tr}()$
$c_{i,1} \| c_{i,2} \leftarrow c_i$
If $c_{i,1} \in \text{CT}$: Abort
$\text{CT} \leftarrow \text{CT} \cup \{c_{i,1}\}$
HT$[m] \leftarrow c_{i,1}$
Return HT$[m]$
```

Figure 14: Adversary for proof of Theorem 3. The Abort instruction lets $\mathcal{P}$ terminate the execution of $\mathcal{A}$ and stop with $(\bot, \bot)$.

in H to the Tr oracle, to process Sign queries as in G_4, and to convert signature forgeries into transcript forgeries. Precisely, when $\mathcal{A}$ comes up with a forgery $m, (Y, c_2, z)$, reduction $\mathcal{P}$ identifies the corresponding identification transcript $Y_l \| c_{l,1} \| c_{l,2} \| z_l$ that it obtained from its own challenger as an answer to the Tr query corresponding to m, forwards the forgery commitment Y and the composed challenge $c = c_{l,1} \| c_2$ to its Ch oracle, and finally outputs z. That is, it tries to be successful in the CIMP-CC game with transcript $Y \| c_{l,1} \| c_2 \| z$. Observe that this transcript is valid if and only if $\mathcal{A}$'s forgery is. Further, as $c_{l,1}$ is uniquely bound to m (since G_1) and $c_2 \neq c_{l,2}$ (since G_3), the submitted challenge c is different from all challenges obtained from the Tr oracle, i.e., the transcript is fresh. We thus have $\Pr[G_4] = \mathbf{Adv}^{\text{cimp-cc}}_{\mathsf{ID}}(\mathcal{P})$. Overall, we obtain the bound from Equation (4). ▌

5 Swap: Optimized signatures from identification

Amongst others, in Section 4 we propose the **MdCmt** transform that constructs UUF-secure signatures from CIMP-UC-secure identification. Further, in Appendix A we propose two generic techniques that convert UUF signatures to signatures with full UF security. One of the latter, **AR**, achieves its goal by adding randomness to signed messages as follows: for signing m, it picks a fresh random seed s and signs $m \| s$ instead. The seed is included in the signature. Overall, the combination of **MdCmt** with **AR** yields tightly secure signatures of the form

$$\sigma \quad = \quad c \,\|\, \mathsf{ID.Rsp}(c, \mathsf{ID.Cmt}^{-1}(\mathrm{H}(m, s))) \,\|\, s$$

(for simplicity we do not annotate trapdoors and keys). In the following, giving up on modularity and with the goal of achieving more compact UF secure signatures, we show that it is safe to choose c and s to be identical. We call this direct transformation **Swap**; the details are specified in Fig. 15.

The name of our transform is inspired by the work of [24] where the authors use the term "swap" for a very specific construction of a signature scheme with tight reduction to the hardness of factoring. Folklore, and hints in the literature [2], indicate that researchers understand the method was more general. But exactly how general was not understood or determined before, perhaps for lack of definitions. Our definition of trapdoor identification and the CIMP-UC notion

```
DS.Kg^H                                  DS.Sig^H(vk, sk, m)
(ivk, isk, itk) ←$ ID.Kg                  ivk ← vk ; (isk, itk) ← sk
vk ← ivk ; sk ← (isk, itk)                c ←$ {0,1}^{ID.cl}
Return (vk, sk)                           Y ← H(m, c)
                                          y ←$ ID.Cmt^{-1}(ivk, itk, Y)
DS.Vf^H(vk, m, σ)                         z ← ID.Rsp(ivk, isk, c, y)
ivk ← vk ; (c, z) ← σ                     σ ← (c, z) ; Return σ
Y ← H(m, c)
Return ID.Vf(ivk, Y‖c‖z)
```

Figure 15: The construction of signature scheme $\mathsf{DS} = \mathbf{Swap}[\mathsf{ID}]$ from a trapdoor identification scheme ID. By $\mathrm{H}(\cdot,\cdot)$ we denote random oracle $\mathrm{H}((\cdot,\cdot), \mathsf{ID.CmtSp}(ivk))$ with range $\mathsf{ID.CmtSp}(ivk)$.

allows us to fill this gap and give a characterization of the swap method via the general **Swap** transform.

Theorem 4 *Let signature scheme* $\mathsf{DS} = \mathbf{Swap}[\mathsf{ID}]$ *be obtained from trapdoor identification scheme* ID *as in Fig. 15. Let* $\mathcal{A}$ *be a* UF*-adversary against* DS*. Suppose the number of queries that* $\mathcal{A}$ *makes to its* H *oracle is* q_h*, the number of queries to* Sign *is* q_s*, and that* $q = q_s + q_h$*. Let* $N = \min |\mathsf{ID.CmtSp}(ivk)|$ *where the minimum is over all* $(ivk, isk, itk) \in [\mathsf{ID.Kg}]$*. Then from* $\mathcal{A}$ *we construct a* CIMP-UC *adversary* $\mathcal{P}$ *such that*

$$\mathbf{Adv}^{\mathrm{uf}}_{\mathsf{DS}}(\mathcal{A}) \le \mathbf{Adv}^{\mathrm{cimp\text{-}uc}}_{\mathsf{ID}}(\mathcal{P}) + \frac{q^2}{2N} + \frac{q^2+1}{2^{\mathsf{ID.cl}}} \ . \qquad (5)$$

Adversary $\mathcal{P}$ *makes* q *queries to* Tr *and one query to* Ch*. Its running time is about that of* $\mathcal{A}$*.*

Proof of Theorem 4: Consider games $\mathrm{G}_0, \ldots, \mathrm{G}_6$ from Fig. 16 and Fig. 17. Game G_0 is precisely the UUF experiment (cf. Fig. 2) with the algorithms of $\mathbf{Swap}[\mathsf{ID}]$ plugged in. (Recall we assume the adversary, before requesting a signature or attempting a forgery, to pose all corresponding H queries.) We have $\Pr[\mathrm{G}_0] = \mathbf{Adv}^{\mathrm{uf}}_{\mathsf{DS}}(\mathcal{A})$.

In G_0, the code implementing the H oracle generates hash values by sampling uniformly at random from $\mathsf{ID.CmtSp}(ivk)$. Game G_1 is like G_0 but it aborts if any two of these values collide (set YT is used to keep track of that). As G_0 and G_1 are identical unless such a collision happens, we have $\Pr[\mathrm{G}_0] \le \Pr[\mathrm{G}_1] + (0 + \ldots + (q_h + q_s - 1))/|\mathsf{ID.CmtSp}(ivk)| \le \Pr[\mathrm{G}_1] + (q^2/2)/N$.

By the definition of a trapdoor ID scheme, picking commitments Y uniformly at random from $\mathsf{ID.CmtSp}(ivk)$ or by executing $(Y, \_) \leftarrow\$ \mathsf{ID.Cmt}(ivk)$ (the second value of the output is ignored) yields the same distribution. In Game G_2 we correspondingly change the implementation of H. We have $\Pr[\mathrm{G}_1] = \Pr[\mathrm{G}_2]$.

In the Sign oracle of G_2, challenges c are sampled uniformly at random from $\{0,1\}^{\mathsf{ID.cl}}$. Such a sample may collide with a value previously queried to the H oracle, or with a challenge previously sampled when processing a Sign query. Game G_3 is like G_2 but it aborts if one of these events happens, i.e., with probability at most $(0 + \ldots + (q_h + q_s - 1))/2^{\mathsf{ID.cl}} \le (q^2/2)/2^{\mathsf{ID.cl}}$. As G_2 and G_3 are identical otherwise, we have $\Pr[\mathrm{G}_2] \le \Pr[\mathrm{G}_3] + q^2/2^{\mathsf{ID.cl}+1}$.

Observe that in G_3 every H query within Sign is a fresh one, i.e., a new commitment is sampled and assigned (as opposed to: an old one replayed). In Game G_4 we let Sign perform the corresponding H operations itself. This is a rewriting step, so we have $\Pr[\mathrm{G}_3] = \Pr[\mathrm{G}_4]$.

```
Game G_0, G_1, G_2, G_3, G_4
  $M \leftarrow \emptyset$ ; $\mathrm{YT} \leftarrow \emptyset$
  $(ivk, isk, itk) \leftarrow_\$ \mathsf{ID.Kg}$
  $(m, \sigma) \leftarrow_\$ \mathcal{A}^{\text{Sign},\mathrm{H}}(ivk)$
  If $m \in M$: Abort
  $(c, z) \leftarrow \sigma$
  $Y \leftarrow \mathrm{HT}[m, c]$
  Return $\mathsf{ID.Vf}(ivk, Y\|c\|z)$

H$(m, c)$
  If $\mathrm{HT}[m, c]$: Return $\mathrm{HT}[m, c]$
  $Y \leftarrow_\$ \mathsf{ID.CmtSp}(ivk)$                // G_0–G_1
  $(Y, \_) \leftarrow_\$ \mathsf{ID.Cmt}(ivk)$           // G_2–G_4
  If $Y \in \mathrm{YT}$: Abort                          // G_1–G_4
  $\mathrm{YT} \leftarrow \mathrm{YT} \cup \{Y\}$        // G_1–G_4
  $\mathrm{HT}[m, c] \leftarrow Y$
  Return $\mathrm{HT}[m, c]$

Sign$(m)$
  $c \leftarrow_\$ \{0,1\}^{\mathsf{ID.cl}}$
  If $\mathrm{HT}[m, c]$: Abort                          // G_3–G_4
  $Y \leftarrow \mathrm{H}(m, c)$                        // G_0–G_3
  $(Y, \_) \leftarrow_\$ \mathsf{ID.Cmt}(ivk)$           // G_4
  If $Y \in \mathrm{YT}$: Abort                          // G_4
  $\mathrm{YT} \leftarrow \mathrm{YT} \cup \{Y\}$        // G_4
  $\mathrm{HT}[m, c] \leftarrow Y$                       // G_4
  $y \leftarrow_\$ \mathsf{ID.Cmt}^{-1}(ivk, itk, Y)$
  $z \leftarrow \mathsf{ID.Rsp}(ivk, isk, c, y)$
  $\sigma \leftarrow (c, z)$
  $M \leftarrow M \cup \{m\}$ ; Return $\sigma$
```

Figure 16: Games G_0–G_4 used in the proof of Theorem 4. Game G_0 is the UF game with the algorithms of $\mathsf{DS} = \mathbf{Swap}[\mathsf{ID}]$ plugged in. The Abort instruction terminates the execution of $\mathcal{A}$ and stops the game with output value false.

```
Game G_5, G_6
  $M \leftarrow \emptyset$ ; $\mathrm{YT} \leftarrow \emptyset$ ; $i \leftarrow 0$ ; $\mathrm{ST} \leftarrow \emptyset$
  $(ivk, isk, itk) \leftarrow_\$ \mathsf{ID.Kg}$
  $(m, \sigma) \leftarrow_\$ \mathcal{A}^{\text{Sign},\mathrm{H}}(ivk)$
  If $m \in M$: Abort
  $(c, z) \leftarrow \sigma$ ; $l \leftarrow \mathrm{ST}[m, c]$
  If $c = c_l$: Abort                                    // G_6
  Return $\mathsf{ID.Vf}(ivk, Y_l\|c\|z)$

H$(m, c)$
  If $\mathrm{HT}[m, c]$: Return $\mathrm{HT}[m, c]$
  $i \leftarrow i + 1$ ; $\mathrm{ST}[m, c] \leftarrow i$
  $(Y_i, \_) \leftarrow_\$ \mathsf{ID.Cmt}(ivk)$
  $c_i \leftarrow_\$ \{0,1\}^{\mathsf{ID.cl}}$
  If $Y_i \in \mathrm{YT}$: Abort
  $\mathrm{YT} \leftarrow \mathrm{YT} \cup \{Y_i\}$
  $\mathrm{HT}[m, c] \leftarrow Y_i$
  Return $\mathrm{HT}[m, c]$

Sign$(m)$
  $i \leftarrow i + 1$
  $(Y_i, y_i) \leftarrow_\$ \mathsf{ID.Cmt}(ivk)$
  $c_i \leftarrow_\$ \{0,1\}^{\mathsf{ID.cl}}$
  $z_i \leftarrow \mathsf{ID.Rsp}(ivk, isk, c_i, y_i)$
  If $\mathrm{HT}[m, c_i]$: Abort
  $\mathrm{ST}[m, c_i] \leftarrow i$
  If $Y_i \in \mathrm{YT}$: Abort
  $\mathrm{YT} \leftarrow \mathrm{YT} \cup \{Y_i\}$
  $\mathrm{HT}[m, c_i] \leftarrow Y_i$
  $\sigma \leftarrow (c_i, z_i)$
  $M \leftarrow M \cup \{m\}$ ; Return $\sigma$
```

Figure 17: Games G_5 and G_6 used in the proof of Theorem 4. The Abort instruction terminates the execution of $\mathcal{A}$ and stops the game with output value false.

The step from G_4 to G_5 is larger, but also pure rewriting: It consists of numerating the invocations of the $\mathsf{ID.Cmt}$ algorithm, permuting the lines of the Sign oracle to a more convenient order (note that this does not change the probability of aborting), adding to the H code the sampling of a redundant challenge, changing the way private state y is computed in the Sign oracle to an equivalent method, and introducing the look-up table ST that links message-challenge pairs to the corresponding $\mathsf{ID.Cmt}$ invocation. We have $\Pr[G_4] = \Pr[G_5]$.

In Game G_6 an added abort condition catches the case where the adversary forges a signature on

```
Adversary $\mathcal{P}^{\textsc{Tr},\textsc{Ch}}(ivk)$
$M \leftarrow \emptyset$ ; $\mathrm{YT} \leftarrow \emptyset$ ; $i \leftarrow 0$ ; $\mathrm{ST} \leftarrow \emptyset$
$(m, \sigma) \leftarrow\$ \mathcal{A}^{\textsc{Sign},\mathrm{H}}(ivk)$
If $m \in M$: Abort
$(c, z) \leftarrow \sigma$ ; $l \leftarrow \mathrm{ST}[m, c]$
If $c = c_l$: Abort
$(j, c') \leftarrow \textsc{Ch}(l, c)$   // $j = 1 \wedge c' = c$
Return $(j, z)$

$\mathrm{H}(m, c)$ // $\mathcal{P}$
If $\mathrm{HT}[m, c]$: Return $\mathrm{HT}[m, c]$
$i \leftarrow i + 1$ ; $\mathrm{ST}[m, c] \leftarrow i$
$Y_i \| c_i \| \_ \leftarrow\$ \textsc{Tr}()$
If $Y_i \in \mathrm{YT}$: Abort
$\mathrm{YT} \leftarrow \mathrm{YT} \cup \{Y_i\}$
$\mathrm{HT}[m, c] \leftarrow Y_i$
Return $\mathrm{HT}[m, c]$

$\textsc{Sign}(m)$ // $\mathcal{P}$
$i \leftarrow i + 1$
$Y_i \| c_i \| z_i \leftarrow\$ \textsc{Tr}()$
If $\mathrm{HT}[m, c_i]$: Abort
$\mathrm{ST}[m, c_i] \leftarrow i$
If $Y_i \in \mathrm{YT}$: Abort
$\mathrm{YT} \leftarrow \mathrm{YT} \cup \{Y_i\}$
$\mathrm{HT}[m, c_i] \leftarrow Y_i$
$\sigma \leftarrow (c_i, z_i)$
$M \leftarrow M \cup \{m\}$ ; Return $\sigma$
```

Figure 18: Adversary for proof of Theorem 4. The Abort instruction lets $\mathcal{P}$ terminate the execution of $\mathcal{A}$ and stop with output value $(\perp, \perp)$.

a message using as the challenge c the value c_l picked inside the H oracle for that message. As c_l is uniformly distributed in $\{0,1\}^{\mathsf{ID.cl}}$ and no information about it is ever leaked to the adversary, the probability that $c = c_l$ is precisely $1/2^{\mathsf{ID.cl}}$. As G_5 and G_6 are identical unless this event occurs, we have $\Pr[\mathrm{G}_5] \leq \Pr[\mathrm{G}_6] + 1/2^{\mathsf{ID.cl}}$.

To assess $\Pr[\mathrm{G}_6]$ we construct from $\mathcal{A}$ an adversary $\mathcal{P}$ against CIMP-UC and analyze its success probability. The code of $\mathcal{P}$ is given in Fig. 18. It internally executes $\mathcal{A}$ and mimics the environment otherwise provided by G_6. The general idea is to outsource the transcript generation that happens in H and Sign to the Tr oracle, and to convert signature forgeries into transcript forgeries. Precisely, when $\mathcal{A}$ comes up with a forgery $m, (c, z)$, reduction $\mathcal{P}$ identifies the corresponding (partial) identification transcript $Y_l \| c_l \| \_$ that it obtained from its own challenger as an answer to the Tr query corresponding to m and c, forwards the forgery challenge c to its Ch oracle, and finally outputs z. That is, it tries to be successful in the CIMP-UC game with transcript $Y_l \| c \| z$. Observe that this transcript is valid if and only if $\mathcal{A}$'s forgery is. Further, as Y_l is uniquely bound to m (since G_1) and $c \neq c_l$ (since G_6), the pair (Y_l, c) is different from all commitment-challenge pairs obtained from the Tr oracle, i.e., the transcript is fresh. We thus have $\Pr[\mathrm{G}_6] = \mathbf{Adv}^{\text{cimp-uc}}_{\mathsf{ID}}(\mathcal{P})$. Overall, we obtain the bound from Equation (5). ▌

6 Instantiation with Sigma protocols

We have shown several ways to obtain signature schemes from identification schemes that meet our new $\mathrm{P_{id}}$ notions, the advantage over the standard **FS** method being that the $\mathrm{P_{sig}} \rightarrow \mathrm{P_{id}}$ reduction is tight. To exploit this, we need another step, namely to obtain identification schemes for which $\mathrm{P_{id}}$ can be established with a tight reduction to the underlying algebraic problem or assumption $\mathrm{P_{alg}}$. Here we show that this is possible, for two choices of $\mathrm{P_{id}}$, namely CIMP-UC and CIMP-UU, for a broad class of identification schemes, namely those that are Sigma protocols. The problem $\mathrm{P_{alg}}$ will be recovery of the secret key of the identification scheme given nothing but the public key, which is usually the one under which we aim to prove security of both the identification and signature schemes. First we need to recall a few standard definitions.

6.1 Definitions

We say that an identification scheme ID is *honest verifier zero-knowledge* (HVZK) if there exists a public algorithm $\textsc{TrS}$ that generates identification transcripts with the same distribution as an honest execution of ID's algorithms. More precisely, we require the outputs of the following two processes to be identically distributed. Both processes first generate $(ivk, isk, itk) \leftarrow_{\$} \mathsf{ID.Kg}$. The first process then lets $(Y, y) \leftarrow_{\$} \mathsf{ID.Cmt}(ivk)$, followed by $c \leftarrow_{\$} \{0,1\}^{\mathsf{ID.cl}}$, followed by $z \leftarrow_{\$} \mathsf{ID.Rsp}(ivk, isk, c, y)$, and outputs $(ivk, Y\|c\|z)$. The second process simply outputs ivk and a transcript obtained via $Y\|c\|z \leftarrow_{\$} \textsc{TrS}(ivk)$.

We say that an identification scheme ID is *extractable* if from any two transcripts that have the same commitment but different challenges one can recover the secret key. Formally we require the existence of an algorithm $\mathsf{ID.Ex}$ such that for all $(ivk, isk, itk) \in [\mathsf{ID.Kg}]$, if $Y_1\|c_1\|z_1$ and $Y_2\|c_2\|z_2$ are accepting transcripts under ivk with $Y_1 = Y_2$ but $c_1 \neq c_2$, then $\mathsf{ID.Ex}$ given ivk and the transcripts returns isk.

We say that an identification scheme is a Sigma protocol [13] if it is both HVZK and extractable.

An identification scheme ID is resilient to *key recovery* if it is hard to recover the secret key from just the verification key (but no transcripts). Formally, we require the key-recovery advantage $\mathbf{Adv}^{\mathrm{kr}}_{\mathsf{ID}}(\mathcal{I}) = \Pr[\mathbf{G}^{\mathrm{kr}}_{\mathsf{ID}}(\mathcal{I})]$ to be small for any adversary $\mathcal{I}$, where the game first initialises $(ivk, isk, itk) \leftarrow_{\$} \mathsf{ID.Kg}$, then executes the adversary via $isk^* \leftarrow_{\$} \mathcal{I}(ivk)$, and finally outputs true if $isk^* = isk$.

Finally, an identification scheme ID has *unique responses* if for any $(ivk, isk, itk) \in [\mathsf{ID.Kg}]$ and any commitment-challenge pair (Y, c) there is at most one response z that is valid, i.e., satisfies $\mathsf{ID.Vf}(ivk, Y\|c\|z) = \mathsf{true}$.

6.2 Security of identification schemes

As we show next, identification schemes that fulfill suitable combinations of these properties also provide security in the sense of CIMP-UU, CIMP-UC, or CIMP-CU. The crucial point is that in the first two cases, the reduction is *tight*. This is shown by the following.

Theorem 5 *Let* ID *be an identification scheme that is honest verifier zero-knowledge and extractable. Then for any adversary* $\mathcal{P}$ *against* CIMP-UU *that poses q-many* $\textsc{Ch}$ *queries we construct a key recovery adversary* $\mathcal{I}$ *such that*

$$\mathbf{Adv}^{\mathrm{cimp\text{-}uu}}_{\mathsf{ID}}(\mathcal{P}) \leq \mathbf{Adv}^{\mathrm{kr}}_{\mathsf{ID}}(\mathcal{I}) + q/2^{\mathsf{ID.cl}} \ .$$

If ID *also has unique responses, then for any adversary* $\mathcal{P}$ *against* CIMP-UC *we construct a key recovery adversary* $\mathcal{I}$ *such that*

$$\mathbf{Adv}^{\mathrm{cimp\text{-}uc}}_{\mathsf{ID}}(\mathcal{P}) \leq \mathbf{Adv}^{\mathrm{kr}}_{\mathsf{ID}}(\mathcal{I}) \ .$$

In both cases, the running time of $\mathcal{I}$ *is about that of the corresponding* $\mathcal{P}$*.*

Proof of Theorem 5: Consider the CIMP-UU case first. The $\textsc{Tr}$ oracle of game $\mathbf{G}^{\mathrm{cimp\text{-}uu}}_{\mathsf{ID}}(\mathcal{P})$ from Fig. 4 generates transcripts $Y_i\|c_i\|z_i$ by first invoking $\mathsf{ID.Cmt}$, then samling from $\{0,1\}^{\mathsf{ID.cl}}$, and then invoking $\mathsf{ID.Rsp}$. By the zero-knowledge property if ID this can perfectly be emulated using the $\textsc{TrS}$ algorithm. Consider next the $\textsc{Ch}$ oracle: each query is associated to a specific

TR-generated transcript $Y_l \| c_l \| z_l$ and involves picking a fresh challenge c for Y_l. The probability that throughout the experiment the condition $c = c_l$ is ever met is precisely $q/2^{\mathsf{ID.cl}}$. Consider now the modification of $\mathbf{G}^{\text{cimp-uu}}_{\mathsf{ID}}(\mathcal{P})$ where (a) the TR oracle is simulated using TRS, and (b) the game aborts if $c = c_l$ holds. In this game from any winning adversary $\mathcal{P}$ we obtain a transcript $Y_l \| c \| z$ such that together with $Y_l \| c_l \| z_l$ the precondition of extraction is fulfilled. Thus, from $\mathcal{P}$ and algorithm $\mathsf{ID.Ex}$ we can construct an adversary $\mathcal{I}$ that recovers the secret key. This establishes the claimed bound.

We move on to the CIMP-UC case. Consider the modification of $\mathbf{G}^{\text{cimp-uc}}_{\mathsf{ID}}(\mathcal{P})$ where the TR oracle is simulated as above, i.e., using TRS. Note that this time the CH oracle lets the adversary specify its own challenges. From any winning adversary we obtain a transcript $Y_l \| c \| z$. Let $Y_l \| c_l \| z_l$ be the corresponding transcript simulated in the TR oracle. As we assume unique responses, if $c = c_l$ then also $z = z_l$, i.e., the adversary did not forge but instead replay. Thus for any winning adversary we have $c \neq c_l$ and can construct a key recovering algorithm $\mathcal{I}$ is above. ▌

In the CIMP-CU case, a reduction is possible but it is not tight, as shown by the following:

Theorem 6 *Let* ID *be an identification scheme that is honest verifier zero-knowledge and extractable. For any adversary* $\mathcal{P}$ *against* CIMP-CU *making* q *queries to its* CH *oracle, we construct a key recovery adversary* $\mathcal{I}$ *such that*

$$\mathbf{Adv}^{\text{cimp-cu}}_{\mathsf{ID}}(\mathcal{P}) \leq q \left(\sqrt{\mathbf{Adv}^{\text{kr}}_{\mathsf{ID}}(\mathcal{I})} + \frac{1}{2^{\mathsf{ID.cl}}} \right) .$$

The running time of $\mathcal{I}$ *is about twice that of* $\mathcal{P}$.

To establish CIMP-CU security, our route will be via standard techniques exploiting known results, and the proof can be found for completeness in Appendix B. Lemma 10 relates CIMP-CU security of ID to IMP-PA, the standard security against impersonation under passive attack as formalized in [1]. (Since CIMP-CU security in which the adversary makes exactly one query to its CH oracle is equivalent to IMP-PA security, we define $\mathbf{Adv}^{\text{imp-pa}}_{\mathsf{ID}}(\mathcal{P}) = \mathbf{Adv}^{\text{cimp-cu}}_{\mathsf{ID}}(\mathcal{P})$ such that, in $\mathbf{G}^{\text{cimp-cu}}_{\mathsf{ID}}(\mathcal{P})$, $\mathcal{P}$ makes exactly one query to its CH oracle.) Lemma 11 relates IMP-PA security to key recovery using the reset lemma of [7] and the extractability property of ID. Together these yield CIMP-CU security as in Theorem 5.

6.3 Tight signatures from RSA

Our framework applies to a large number of identification schemes to turn them into signature schemes with tight reductions. As an illustrative and canonical example, we discuss GQ. We show how it is a trapdoor identification scheme and discuss its security.

GQ. The GQ [20] scheme is shown in Fig. 19. It makes use of an RSA generator: for a modulus length k, RSA is an algorithm that returns a tuple (N, e, d) where modulus $N = pq$ in the range $2^{k-1} < N < 2^k$ is the product of distinct, odd primes p and q, encryption and decryption exponents e, d are in $\mathbb{Z}^*_{\varphi(N)}$ and $ed \equiv 1 \pmod{\varphi(N)}$. Fig. 19 shows the GQ-ID identification scheme associated to RSA and a challenge length $l < k$ such that $\gcd(e, c) = 1$ for all $c \in \{0,1\}^l$ and all $(N, e, d) \in [\mathsf{RSA}]$. The commitment space is $\mathbb{Z}_N \setminus \{0\}$. We can see via Algorithm $\mathsf{GQ\text{-}ID.Cmt}^{-1}$ of Fig. 19 that GQ-ID is trapdoor. Note that this requires putting the decryption exponent d in the secret key, a change from the classic GQ scheme.

In order to apply Theorem 5 to tightly obtain a signature scheme from GQ-ID using our transformations, we must show that GQ-ID satisfies several conditions. Most of these are standard

```
GQ-ID.Kg
$(N,e,d) \leftarrow\!\$\ \mathsf{RSA}$
$x \leftarrow\!\$\ \mathbb{Z}_N^*$
$X \leftarrow x^e \bmod N$
Return $((N,e,X),x,d)$
```

Prover		Verifier
Input: $(N,e,X),x$		Input: (N,e,X)
$y \leftarrow\!\$\ (\mathbb{Z}_N \setminus \{0\})$		
$Y \leftarrow y^e \bmod N$	$\xrightarrow{Y}$	$c \leftarrow\!\$\ \{0,1\}^l$
	$\xleftarrow{c}$	
$z \leftarrow yx^c \bmod N$	$\xrightarrow{z}$	$v \leftarrow (z^e \equiv YX^c \pmod N)$
		$\wedge\ (Y \not\equiv 0 \pmod N))$

```
GQ-ID.Cmt$^{-1}((N,e,X),d,Y)$
$y \leftarrow Y^d \bmod N$
Return $y$
```

```
GQ-ID.Ex$((N,e,X),Y,c_1,z_1,c_2,z_2)$
If $\gcd(z_1,N) \neq 1$ or $\gcd(z_2,N) \neq 1$):
    Factor $N$ to get $\varphi(N)$
    $d \leftarrow e^{-1} \bmod \varphi(N)$
    $x \leftarrow X^d \bmod N$
    Return $x$
$z \leftarrow z_1 z_2^{-1} \bmod N$
$c \leftarrow c_1 - c_2$
$(a,b) \leftarrow \mathrm{egcd}(e,c)$
$x \leftarrow X^a z^b \bmod N$
Return $x$
```

```
Game $\mathbf{G}^{\mathrm{ow}}_{\mathsf{RSA}}(\mathcal{A})$
$(N,e,d) \leftarrow\!\$\ \mathsf{RSA}$
$x \leftarrow\!\$\ \mathbb{Z}_N^*$
$X \leftarrow x^e \bmod N$
$x' \leftarrow\!\$\ \mathcal{A}(N,e,X)$
Return $(x' = x)$
```

Figure 19: Identification scheme GQ-ID associated to RSA generator RSA with modulus length k, and challenge length l. **Bottom right:** Game defining one-wayness of RSA generator RSA.

observations. Then, since we have tight reductions $\mathrm{P_{sig}}{\rightarrow}\mathrm{P_{id}}$ and $\mathrm{P_{id}}{\rightarrow}\mathrm{P_{alg}}$, we can pick the RSA modulus based on the assumption that the NFS is the best factoring method.

Sigma protocol properties. First we recall why the scheme is honest verifier zero knowledge. Given a public key (N,e,X), transcripts can be simulated as follows. Sample $c \leftarrow\!\$\ \{0,1\}^l$ and $z \leftarrow\!\$\ \mathbb{Z}_N \setminus \{0\}$. Compute $Y \leftarrow z^e/X^c \bmod N$. The transcript is $Y\|c\|z$. This has the same distribution as transcripts generated in Fig. 19: c is clearly identically distributed in Fig. 19; if y is uniform on $\mathbb{Z}_N \setminus \{0\}$ then so is Y since exponentiation by e is a permutation on $\mathbb{Z}_N$; and thus z is also uniform on $\mathbb{Z}_N \setminus \{0\}$. Next we recall why it is extractable. Algorithm GQ-ID.Ex of Fig. 19 provides extractability of the GQ secret key x given two accepting transcripts with the same commitment but distinct challenges. By egcd we denote the extended gcd algorithm that given relatively prime inputs e,c returns a,b such that $ae+bc=1$.

Resilient to key recovery. KR security of GQ-ID holds under one-wayness of RSA, formalized by defining the OW-advantage of an adversary $\mathcal{A}$ against RSA by $\mathbf{Adv}^{\mathrm{ow}}_{\mathsf{RSA}}(\mathcal{A}) = \Pr[\mathbf{G}^{\mathrm{ow}}_{\mathsf{RSA}}(\mathcal{A})]$ where the game is in Fig. 19.

Theorem 7 *Let GQ-ID be the identification scheme associated to RSA generator RSA with modulus length k and challenge length l as above. Let $\mathcal{I}$ be a* KR *adversary. Then from $\mathcal{I}$ we can construct* OW *adversary $\mathcal{A}$ such that*

$$\mathbf{Adv}^{\mathrm{kr}}_{\mathsf{GQ\text{-}ID}}(\mathcal{I}) \leq \mathbf{Adv}^{\mathrm{ow}}_{\mathsf{RSA}}(\mathcal{A})\ . \tag{6}$$

The running time of $\mathcal{A}$ is that of $\mathcal{I}$.

The proof of Theorem 7 is simple. $\mathcal{A}$ immediately provides the (N,e,X) values from the OW-challenger for RSA to $\mathcal{I}$. For simulating KR, this is sufficient: in particular, $\mathcal{A}$ does not have to simulate transcript queries. $\mathcal{I}$ returns the secret key of the ID scheme directly, which is the solution to the RSA challenge.

HASHING ONTO COMMITMENT SPACE. Several constructions require a random oracle with range $\mathsf{ID.CmtSp}(ivk)$, which is $\mathbb{Z}_N \setminus \{0\}$, which we can easily build.

References

[1] M. Abdalla, J. H. An, M. Bellare, and C. Namprempre. From identification to signatures via the Fiat-Shamir transform: Minimizing assumptions for security and forward-security. In L. R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 418–433. Springer, Heidelberg, Apr. / May 2002.

[2] M. Abdalla, F. Ben Hamouda, and D. Pointcheval. Tighter reductions for forward-secure signature schemes. In K. Kurosawa and G. Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 292–311. Springer, Heidelberg, Feb. / Mar. 2013.

[3] M. Abdalla, P.-A. Fouque, V. Lyubashevsky, and M. Tibouchi. Tightly-secure signatures from lossy identification schemes. In D. Pointcheval and T. Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 572–590. Springer, Heidelberg, Apr. 2012.

[4] A. Bagherzandi, J. H. Cheon, and S. Jarecki. Multisignatures secure under the discrete logarithm assumption and a generalized forking lemma. In P. Ning, P. F. Syverson, and S. Jha, editors, *ACM CCS 08*, pages 449–458. ACM Press, Oct. 2008.

[5] G. Barwood. Digital signatures using elliptic curves. sci.crypt mailing list, February 2 1997. `https://groups.google.com/forum/#!msg/sci.crypt/SalLSLBBTe4/xtYNGDe6irIJ`.

[6] M. Bellare and G. Neven. Multi-signatures in the plain public-key model and a general forking lemma. In A. Juels, R. N. Wright, and S. Vimercati, editors, *ACM CCS 06*, pages 390–399. ACM Press, Oct. / Nov. 2006.

[7] M. Bellare and A. Palacio. GQ and Schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. In M. Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 162–177. Springer, Heidelberg, Aug. 2002.

[8] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93*, pages 62–73. ACM Press, Nov. 1993.

[9] M. Bellare and P. Rogaway. The exact security of digital signatures: How to sign with RSA and Rabin. In U. M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 399–416. Springer, Heidelberg, May 1996.

[10] M. Bellare and P. Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, Heidelberg, May / June 2006.

[11] D. J. Bernstein. How to design an elliptic-curve signature system, Mar. 2014. `http://blog.cr.yp.to/20140323-ecdsa.html`.

[12] C. Cachin, S. Micali, and M. Stadler. Computationally private information retrieval with polylogarithmic communication. In J. Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 402–414. Springer, Heidelberg, May 1999.

[13] R. Cramer. *Modular Design of Secure, yet Practical Protocls.* PhD thesis, University of Amsterdam, 1996.

[14] U. Feige, A. Fiat, and A. Shamir. Zero knowledge proofs of identity. In A. Aho, editor, *19th ACM STOC*, pages 210–217. ACM Press, May 1987.

[15] U. Feige, A. Fiat, and A. Shamir. Zero-knowledge proofs of identity. *Journal of Cryptology*, 1(2):77–94, 1988.

[16] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A. M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Heidelberg, Aug. 1987.

[17] E.-J. Goh and S. Jarecki. A signature scheme as secure as the Diffie-Hellman problem. In E. Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 401–415. Springer, Heidelberg, May 2003.

[18] S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, Apr. 1988.

[19] S. Goldwasser and R. Ostrovsky. Invariant signatures and non-interactive zero-knowledge proofs are equivalent (extended abstract). In E. F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 228–245. Springer, Heidelberg, Aug. 1993.

[20] L. C. Guillou and J.-J. Quisquater. A "paradoxical" indentity-based signature scheme resulting from zero-knowledge. In S. Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 216–231. Springer, Heidelberg, Aug. 1990.

[21] J. Katz and N. Wang. Efficiency improvements for signature schemes with tight security reductions. In S. Jajodia, V. Atluri, and T. Jaeger, editors, *ACM CCS 03*, pages 155–164. ACM Press, Oct. 2003.

[22] N. Koblitz and A. Menezes. The random oracle model: A twenty-year retrospective. Cryptology ePrint Archive, Report 2015/140, 2015. `http://eprint.iacr.org/2015/140`.

[23] A. Lysyanskaya. Unique signatures and verifiable random functions from the DH-DDH separation. In M. Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 597–612. Springer, Heidelberg, Aug. 2002.

[24] S. Micali and L. Reyzin. Improving the exact security of digital signature schemes. *Journal of Cryptology*, 15(1):1–18, 2002.

[25] D. M'Raïhi, D. Naccache, D. Pointcheval, and S. Vaudenay. Computational alternatives to random number generators. In S. E. Tavares and H. Meijer, editors, *SAC 1998*, volume 1556 of *LNCS*, pages 72–80. Springer, Heidelberg, Aug. 1999.

[26] K. Ohta and T. Okamoto. On concrete security treatment of signatures derived from identification. In H. Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 354–369. Springer, Heidelberg, Aug. 1998.

```
$\underline{\mathsf{DDS.Kg}^{\mathrm{H}}}$
Return $\mathsf{DS.Kg}^{\mathrm{H}}$

$\underline{\mathsf{DDS.Vf}^{\mathrm{H}}(vk, m, \sigma)}$
Return $\mathsf{DS.Vf}^{\mathrm{H}}(vk, m, \sigma)$

$\underline{\mathsf{DDS.Sig}^{\mathrm{H}}(vk, sk, m)}$
$r \leftarrow \mathrm{H}(sk, m)$
Return $\mathsf{DS.Sig}^{\mathrm{H}}(vk, sk, m; r)$
```

Figure 20: Our construction of deterministic signature scheme $\mathsf{DDS} = \mathbf{DR}[\mathsf{DS}, \mathsf{rl}]$ from a signature scheme DS and a randomization length $\mathsf{rl} \in \mathbb{N}$. By $\mathrm{H}(\cdot, \cdot)$ we denote random oracle $\mathrm{H}((\cdot, \cdot), \{0,1\}^{\mathsf{rl}})$ with range $\{0,1\}^{\mathsf{rl}}$.

[27] H. Ong and C.-P. Schnorr. Fast signature generation with a Fiat-Shamir-like scheme. In I. Damgård, editor, *EUROCRYPT'90*, volume 473 of *LNCS*, pages 432–440. Springer, Heidelberg, May 1991.

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

[29] C.-P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

A Signature transformations from UUF to UF

We propose two methods (transforms) to turn an arbitrary, given UUF signature scheme into one meeting the standard UF notion. The reductions underlying both transforms are tight. Applying these transforms to the signature schemes given by our transforms of Section 4 results in UF signature schemes with tight reductions to the underlying identification. The first transform requires one more cryptographic component, the other has longer signatures.

A.1 Going from UUF to UF by removing randomness

Every signature scheme that is secure in the unique unforgeability sense can easily be transformed into one that is secure in the standard unforgeability sense by derandomization: any randomness used in the signing procedure is generated as the hash of the long-term secret key and the message. This technique has long been discussed in the context of the standardized DSA and ECDSA signature schemes [5, 11] and a proof for that case is in [22]. We generalize this. The scheme is depicted formally in the transformation **DR** of Fig. 20. The following establishes the general security of this approach.

Theorem 8 *Let* DS *be a digital signature scheme where* $\mathsf{DS.Sig}$ *uses* rl *bits of randomness. Let signature scheme* $\mathsf{DDS} = \mathbf{DR}[\mathsf{DS}, \mathsf{rl}]$ *be obtained from* DS *and* rl *as in Fig. 20. Let* $\mathcal{A}$ *be a* UF*-adversary against* DDS*. Then from* $\mathcal{A}$ *we can construct* UUF*-adversaries* $\mathcal{A}_1$ *and* $\mathcal{A}_2$ *as in Fig. 22 such that*

$$\mathbf{Adv}^{\mathrm{uf}}_{\mathsf{DDS}}(\mathcal{A}) \quad \leq \quad \mathbf{Adv}^{\mathrm{uuf}}_{\mathsf{DS}}(\mathcal{A}_1) + \mathbf{Adv}^{\mathrm{uuf}}_{\mathsf{DS}}(\mathcal{A}_2) \; . \tag{7}$$

The number of queries adversary $\mathcal{A}_1$ *and* $\mathcal{A}_2$ *makes to* SIGN *and* H *is upper-bounded by the number of queries* $\mathcal{A}$ *makes to its corresponding oracles.* $\mathcal{A}_1$ *and* $\mathcal{A}_2$ *each have running time about that of* $\mathcal{A}$*, plus at most* q *times the running time of signature generation plus signature verification in* DS*, where* q *is the number of* H *queries of* $\mathcal{A}_1$ *or* $\mathcal{A}_2$ *respectively.*

```
Game $G_0$, $G_1$, $G_2$, $G_3$, $G_4$, $G_5$
$ST \leftarrow \emptyset$ ; $M \leftarrow \emptyset$
$(vk, sk) \leftarrow\!\!\$\ \mathsf{DS.Kg}^{H}$
$(m, \sigma) \leftarrow\!\!\$\ \mathcal{A}^{\text{Sign},H}(vk)$
If $m \in M$: Abort
Return $\mathsf{DS.Vf}^{H}(vk, m, \sigma)$

$H(sk^*, m)$
If $HT[sk^*, m]$: Return $HT[sk^*, m]$
Pick some $m^* \notin M$                              // $G_3$–$G_4$
$\sigma^* \leftarrow\!\!\$\ \mathsf{DS.Sig}^{H}(vk, sk^*, m^*)$            // $G_3$–$G_4$
If $\mathsf{DS.Vf}^{H}(vk, m^*, \sigma^*)$:                     // $G_3$–$G_4$
    $\mathsf{bad} \leftarrow \mathsf{true}$                          // $G_3$–$G_4$
    Abort                                          // $G_3$
$HT[sk^*, m] \leftarrow\!\!\$\ \{0,1\}^{\mathsf{rl}}$
Return $HT[sk^*, m]$
```

```
Sign($m$)
If $m \in M$: Return $ST[m]$                          // $G_1$–$G_5$
$r \leftarrow H(sk, m)$                                 // $G_0$–$G_1$
If not $HT[sk, m]$:                                 // $G_2$–$G_4$
    $HT[sk, m] \leftarrow\!\!\$\ \{0,1\}^{\mathsf{rl}}$                 // $G_2$–$G_4$
$r \leftarrow HT[sk, m]$                                // $G_2$–$G_4$
$r \leftarrow\!\!\$\ \{0,1\}^{\mathsf{rl}}$                                // $G_5$
$\sigma \leftarrow \mathsf{DS.Sig}^{H}(vk, sk, m; r)$
$ST[m] \leftarrow \sigma$                                  // $G_1$–$G_5$
$M \leftarrow M \cup \{m\}$ ; Return $\sigma$
```

Figure 21: Games for proof of Theorem 8. The Abort instruction terminates the execution of $\mathcal{A}$ and stops the game with output value false.

Proof:

When procedure Sign is replying to signing query m in the deterministic scheme DDS, it should compute the randomness by hashing together its secret key *sk* and the message m. Ideally, this would be freshly chosen the first time a new message is signed; and if each message is signed only once, then we are in the same scenario as the unique unforgeability security game for the non-deterministic scheme DS. However, there is the possibility that the adversary has previously queried this input to the random oracle. If this occurs, then the adversary has recovered the signing key, which enables a signature forgery. This intuition underlies the proof in the sequence of games shown in Fig. 21.

Game G_0 is precisely the unmodified UF game of Fig. 2 with the construction from Fig. 20 plugged in. Observe the scheme is deterministic, so the signing oracle may cache signatures. This is implemented in Game G_1, using the ST[ ] table. We have $\Pr[G_0] = \Pr[G_1]$.

In Game G_2, the signing oracle localises the evaluation of the random oracle if the input is of the form $x = sk \| m$ and the requested output range is $\{0,1\}^{\mathsf{rl}}$. This is again a rewriting step, and thus $\Pr[G_1] = \Pr[G_2]$.

In Game G_3 the oracle implementing H watches out for queries from the adversary that correspond to derandomization—meaning they start with the signing key (or something related to the signing key). When such a query occurs, we check to see if the adversary somehow computed a signing key sk^* that yields a valid signature. Note however that, rather than checking if the queried sk^* is equal to the hidden signing key *sk*, we instead check to see if the queried sk^* enables the creation of valid signatures. When this occurs, G_3 terminates with success, so the adversary has higher advantage to win in G_3 than in G_2: We have $\Pr[G_2] \leq \Pr[G_3]$.

The only difference between G_3 and G_4 is that, when G_3's procedure H finds a valid signature, it terminates the experiment, whereas G_4 continues. This is where both games set the flag bad to true. These games are identical-until-bad [10], meaning differ only in code that follows the setting

```
Adversary $\mathcal{A}_1^{\textsc{Sign},\mathrm{H},\mathcal{A}}(vk)$
$(m,\sigma) \leftarrow\$ \mathcal{A}^{\textsc{Sign}',\mathrm{H}'}(vk)$

Adversary $\mathcal{A}_2^{\textsc{Sign},\mathrm{H},\mathcal{A}}(vk)$
$(m,\sigma) \leftarrow\$ \mathcal{A}^{\textsc{Sign}',\mathrm{H}}(vk)$

$\textsc{Sign}'(m)$
If not $\mathrm{ST}[m]$:
    $\mathrm{ST}[m] \leftarrow\$ \textsc{Sign}(m)$
Return $\mathrm{ST}[m]$

$\mathrm{H}'(sk^*, m)$
Pick some $m^* \notin M$
$\sigma^* \leftarrow\$ \mathsf{DS.Sig}^{\mathrm{H}}(vk, sk^*, m^*)$
If $\mathsf{DS.Vf}^{\mathrm{H}}(vk, m^*, \sigma^*)$:
    Return $(m^*, \sigma^*)$ from $\mathcal{A}_1$
Return $\mathrm{H}(sk^*, m)$
```

Figure 22: Adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$ for proof of Theorem 8.

of the boolean flag bad to true. So we have

$$\Pr[\mathrm{G}_3] = \Pr[\mathrm{G}_4] + \Pr[\mathrm{G}_3] - \Pr[\mathrm{G}_4] \tag{8}$$

$$\leq \Pr[\mathrm{G}_4] + \Pr[\mathrm{G}_4 \text{ sets } \mathsf{bad}] \ . \tag{9}$$

We will need to bound $\Pr[\mathrm{G}_4]$ and $\Pr[\mathrm{G}_4 \text{ sets } \mathsf{bad}]$.

We can see that, when G_4 sets bad, it has found a valid signature (m^*, σ^*) but $m^* \notin M$, meaning m^* was never queried to $\textsc{Sign}$. This is a forgery. UUF adversary $\mathcal{A}_1$ in Fig. 22 simulates G_4 and uses $\mathcal{A}$ to construct a forgery for DS; thus

$$\Pr[\mathrm{G}_4 \text{ sets } \mathsf{bad}] \leq \mathbf{Adv}^{\mathrm{uuf}}_{\mathsf{DS}}(\mathcal{A}_1) \tag{10}$$

To bound $\Pr[\mathrm{G}_4]$, consider Game G_5. Since the portion of G_4's H procedure that leads to setting $\mathsf{bad} \leftarrow \mathsf{true}$ does not actually change the behaviour of the experiment, we can remove that code, resulting in G_5. Furthermore, in G_4, the logic established by G_1 and G_3 ensures that for each m, the value $\mathrm{H}[sk, m]$ is evaluated at most once throughout the whole experiment, so the corresponding line in the signature oracle can be replaced with a random assignment. The hop to G_5 implements this change. Thus $\Pr[\mathrm{G}_4] = \Pr[\mathrm{G}_5]$.

Game G_5 is not quite the UUF game from Fig. 2, but it is close. The difference is that for repeated queries to $\textsc{Sign}$, game G_5 returns the cached previous signature $\mathrm{ST}[m]$, whereas the UUF game would return $\perp$. It is trivial to create an algorithm that connects Game 5 and the uuf game. Namely, algorithm $\mathcal{A}_2$ in Fig. 22 wins the uuf game whenever $\mathcal{A}$ wins G_5, and thus $\Pr[\mathrm{G}_5] \leq \mathbf{Adv}^{\mathrm{uuf}}_{\mathsf{DS}}(\mathcal{A}_2)$. ▌

We remark that in bounding $\Pr[\mathrm{G}_4 \text{ sets } \mathsf{bad}]$, the proof works by obtaining a valid signing key from the adversary and then using that to forge messages. This suggests that it would be reduce to the key recoverability of DS; however, since we already have a $\mathbf{Adv}^{\mathrm{uuf}}_{\mathsf{DS}}(\mathcal{A}_1)$ term from the bound on $\Pr[\mathrm{G}_5]$, there is little value in reducing to key recoverability instead of uuf in G_4.

A.2 Going from UUF to UF by adding randomness

A complementary method for constructing UF signatures from UUF is by adding randomness: before being signed, the message is concatenated with a random seed s, so even for the same message, the input to the UUF signing algorithm is (with high probability) distinct.

```
DDS.Kg
    Return DS.Kg

DDS.Vf$(vk, m, \sigma)$
    $\sigma' \| s \leftarrow \sigma$
    Return DS.Vf$(vk, m \| s, \sigma')$

DDS.Sig$(vk, sk, m)$
    $s \leftarrow\$ \{0,1\}^{\mathsf{sl}}$
    $\sigma' \leftarrow\$$ DS.Sig$(vk, sk, m \| s)$
    $\sigma \leftarrow \sigma' \| s$
    Return $\sigma$
```

Figure 23: Our construction of added-randomness signature scheme $\mathsf{DDS} = \mathbf{AR}[\mathsf{DS}, \mathsf{sl}]$ from a signature scheme DS and a seed length $\mathsf{sl} \in \mathbb{N}$.

Theorem 9 *Let signature scheme* $\mathsf{DDS} = \mathbf{AR}[\mathsf{DS}, \mathsf{sl}]$ *be obtained from signature scheme* DS *and seed length* sl *as in Fig. 23. Let* $\mathcal{A}$ *be a UF-adversary against* DDS*. Then from* $\mathcal{A}$ *we construct a* UUF *adversary* $\mathcal{A}_1$ *such that*

$$\mathbf{Adv}^{\mathrm{uf}}_{\mathsf{DDS}}(\mathcal{A}) \quad \leq \quad \mathbf{Adv}^{\mathrm{uuf}}_{\mathsf{DS}}(\mathcal{A}_1) + \frac{q^2}{2^{\mathsf{sl}+1}} \ , \tag{11}$$

where q is the number of Sign *queries posed by* $\mathcal{A}$*. Adversaries* $\mathcal{A}$ *and* $\mathcal{A}_1$ *have comparable running times.*

Proof of Theorem 9: Let $\mathcal{A}$ be a UF-adversary against DDS. Consider the sequence of games from Fig. 24. Game G_0 is precisely the UF experiment (cf. Fig. 2) with the algorithms of $\mathbf{AR}[\mathsf{DS}, \mathsf{sl}]$ plugged in. We have $\Pr[\mathrm{G}_0] = \mathbf{Adv}^{\mathrm{uf}}_{\mathsf{DDS}}(\mathcal{A})$.

In the Sign oracle of G_0, random seeds are sampled uniformly at random from $\{0,1\}^{\mathsf{sl}}$. Game G_1 is like G_0 but it aborts if any two of these seeds collide. This is tracked via set S. As G_0 and G_1 are identical unless a collision occurs, and the latter happens with probability $(0 + \ldots + (q-1))/2^{\mathsf{sl}} = ((q-1)^2 + (q-1))/2^{\mathsf{sl}+1} \leq q^2/2^{\mathsf{sl}+1}$, we have $\Pr[\mathrm{G}_0] \leq \Pr[\mathrm{G}_1] + q^2/2^{\mathsf{sl}+1}$.

In Game G_2 we introduce an additional abort condition that triggers whenever the Sign oracle invokes the underlying DS.Sig algorithm twice on the same message-seed combination $m \| s$. This is tracked by set M'. As by the change introduced with G_1 it is ensured that seeds s do not repeat, this abort condition is never met. We have $\Pr[\mathrm{G}_1] = \Pr[\mathrm{G}_2]$.

The winning condition $m \notin M$ of G_2 is replaced by $m \| s \notin M'$ in Game G_3. Observe that $m \| s \in M'$ always implies $m \in M$, i.e., the winning condition is relaxed by this step and we have $\Pr[\mathrm{G}_2] \leq \Pr[\mathrm{G}_3]$.

To assess $\Pr[\mathrm{G}_3]$ we construct from $\mathcal{A}$ an adversary $\mathcal{A}_1$ against UUF and analyze its success probability. The code of $\mathcal{A}_1$ is given in Fig. 25. It internally executes $\mathcal{A}$ and mimics the environment otherwise provided by G_3. The general idea is to outsource the signature generation in $\mathcal{A}$'s Sign oracle to $\mathcal{A}_1$'s Sign$_1$ oracle, and to convert DDS forgeries to DS forgeries. Note that the $m \| s \in M'$ condition in Sign ensures that Sign$_1$ always returns a signature (as opposed to $\perp$). Observe further that whenever $\mathcal{A}$'s forgery $(m, \sigma' \| s)$ is valid and fresh, then so is the output $(m \| s, \sigma')$ of $\mathcal{A}_1$. We thus have $\Pr[\mathrm{G}_3] = \mathbf{Adv}^{\mathrm{uuf}}_{\mathsf{DS}}(\mathcal{A}_1)$. Overall, we obtain the bound from Equation (11). ▌

B Proof of Theorem 6

The following results imply Theorem 6. They rely on standard techniques and are included here for completeness.

```
Game G_0, G_1, G_2, G_3
    $M \leftarrow \emptyset$ ; $S \leftarrow \emptyset$ ; $M' \leftarrow \emptyset$
    $(vk, sk) \leftarrow\!\!\$\ \mathsf{DS.Kg}$
    $(m, \sigma) \leftarrow\!\!\$\ \mathcal{A}^{\textsc{Sign}}(vk)$
    If $m \in M$: Abort                      // G_0–G_2
    $\sigma' \| s \leftarrow \sigma$
    If $m \| s \in M'$: Abort                // G_3
    Return $\mathsf{DS.Vf}(vk, m\|s, \sigma')$

Sign(m)
    $s \leftarrow\!\!\$\ \{0,1\}^{\mathsf{sl}}$
    If $s \in S$: Abort                      // G_1–G_3
    $S \leftarrow S \cup \{s\}$              // G_1–G_3
    If $m\|s \in M'$: Abort                  // G_2–G_3
    $\sigma' \leftarrow\!\!\$\ \mathsf{DS.Sig}(vk, sk, m\|s)$
    $M' \leftarrow M' \cup \{m\|s\}$         // G_2–G_3
    $\sigma \leftarrow \sigma' \| s$
    $M \leftarrow M \cup \{m\}$ ; Return $\sigma$
```

Figure 24: Games used in the proof of Theorem 9. Game G_0 is the UF game with the algorithms of $\mathbf{AR}[\mathsf{DS}, \mathsf{sl}]$ plugged in. The Abort instruction terminates the execution of $\mathcal{A}$ and stops the game with output value false.

```
Adversary $\mathcal{A}_1^{\textsc{Sign}_1}(vk)$
    $M \leftarrow \emptyset$ ; $S \leftarrow \emptyset$ ; $M' \leftarrow \emptyset$
    $(m, \sigma) \leftarrow\!\!\$\ \mathcal{A}^{\textsc{Sign}}(vk)$
    $\sigma' \| s \leftarrow \sigma$
    If $m\|s \in M'$: Abort
    Return $(m\|s, \sigma')$

Sign(m) // $\mathcal{A}_1$
    $s \leftarrow\!\!\$\ \{0,1\}^{\mathsf{sl}}$
    If $s \in S$: Abort
    $S \leftarrow S \cup \{s\}$
    If $m\|s \in M'$: Abort
    $\sigma' \leftarrow\!\!\$\ \textsc{Sign}_1(m\|s)$
    $M' \leftarrow M' \cup \{m\|s\}$
    $\sigma \leftarrow \sigma' \| s$
    $M \leftarrow M \cup \{m\}$ ; Return $\sigma$
```

Figure 25: Adversary for proof of Theorem 9. The Abort instruction lets $\mathcal{A}_1$ terminate the execution of $\mathcal{A}$ and stop with output value $(\bot, \bot)$.

Lemma 10 *Let* ID *be an identification scheme. Let* $\mathcal{P}$ *be a* CIMP-CU-*adversary against* ID *making* q *queries to its* Ch *oracle. Then from* $\mathcal{P}$ *we can construct* IMP-PA *adversary* $\mathcal{P}_1$ *such that*

$$\mathbf{Adv}_{\mathsf{ID}}^{\text{cimp-cu}}(\mathcal{P}) \leq q \cdot \mathbf{Adv}_{\mathsf{ID}}^{\text{imp-pa}}(\mathcal{P}_1) .$$

Adversary $\mathcal{P}_1$ *makes as many queries to its* Tr *oracle as* $\mathcal{P}$ *does. The running time of* $\mathcal{P}_1$ *is that of* $\mathcal{P}$ *plus some small overhead.*

Proof of Lemma 10: Adversary $\mathcal{P}_1$ is shown in Fig. 26. It has access to oracles Tr, Ch as per game $\mathbf{G}_{\mathsf{ID}}^{\text{cimp-cu}}(\mathcal{P}_1)$ in which it is executing, but may only make one query to its Ch oracle. It guesses an instance k^* uniformly from $\{1, \ldots, q\}$ and runs $\mathcal{P}$. Adversary $\mathcal{P}_1$ passes $\mathcal{P}$'s Tr queries directly to its own Tr oracle. $\mathcal{P}_1$ simulates answers to $\mathcal{P}$'s queries to its Ch oracle via the subroutine ChS, calling its own oracles inside this. Adversary $\mathcal{P}_1$'s simulation is perfect. Since $\mathcal{P}_1$ will guess the instance k^* which $\mathcal{P}$ successfully impersonates with probability $1/q$, adversary $\mathcal{P}_1$'s success probability is at least $1/q$ times that of $\mathcal{P}$. ▌

Lemma 11 *Let* ID *be an identification scheme that is honest verifier zero-knowledge and extractable. Then for any adversary* $\mathcal{P}$ *against the* IMP-PA *security of* ID*, we can construct an adversary* $\mathcal{I}$ *such that*

$$\mathbf{Adv}_{\mathsf{ID}}^{\text{imp-pa}}(\mathcal{P}) \leq \frac{1}{2^{\mathsf{ID.cl}}} + \sqrt{\mathbf{Adv}_{\mathsf{ID}}^{\text{kr}}(\mathcal{I})} .$$

The running time of $\mathcal{I}$ *is about twice that of* $\mathcal{P}$*, plus the time for an execution of* $\mathsf{ID.Ex}$.

```
Adversary $\mathcal{P}_1^{\text{Tr,Ch}}(ivk)$
    $k^* \leftarrow\!\!\$\ \{1,\ldots,q\}\ ;\ j \leftarrow 0$
    $(k,z) \leftarrow\!\!\$\ \mathcal{P}^{\text{Tr,ChS}}(ivk)$
    If $k \neq k^*$: Return $\perp$
    Return $(1,z)$

ChS$(Y)$
    $j \leftarrow j+1$
    If $j \neq k^*$: $c \leftarrow\!\!\$\ \{0,1\}^{\mathsf{ID.cl}}$
    Else $(1,c) \leftarrow\!\!\$\ \text{Ch}(Y)$
    CT$[j] \leftarrow Y\|c$ ; Return $(j,c)$
```

Figure 26: Adversary for proof of Lemma 10.

Lemma 11 follows directly from the reset lemma of [7] and the extractability property of ID. The reset lemma gives a bound on $\mathbf{Adv}_{\mathsf{ID}}^{\text{imp-pa}}(\mathcal{P})$ based on the probability of obtaining a pair of distinct valid transcripts $Y\|c_1\|z_1, Y\|c_2\|z_2$ running the same adversary twice, the second time where the adversary is "reset" with the same state it had the moment it output its commitment Y. From these distinct valid transcripts and the verification key ivk, we can execute $\mathsf{ID.Ex}$ to obtain the secret key isk, winning the key recoverability game.