Deterring Certificate Subversion: Efficient Double-Authentication-Preventing Signatures

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

October 2016

Abstract

This paper presents highly efficient designs of double authentication preventing signatures (DAPS). In a DAPS, signing two messages with the same first part and differing second parts reveals the signing key. In the context of PKIs we suggest that CAs who use DAPS to create certificates have a court-convincing argument to deny big-brother requests to create rogue certificates, thus deterring certificate subversion. We give two general methods for obtaining DAPS. Both start from trapdoor identification schemes. We instantiate our transforms to obtain numerous specific DAPS that, in addition to being efficient, are proven with tight security reductions to standard assumptions. We implement our DAPS schemes to show that they are not only several orders of magnitude more efficient than prior DAPS but competitive with in-use signature schemes that lack the double authentication preventing property.

[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, a gift from Microsoft corporation and ERC Project ERCC (FP7/615074).

[2] Department of Mathematics, Ruhr University Bochum, Universitätsstr. 150, 44801 Bochum, Germany. Email: bertram.poettering@rub.de. URL: http://www.crypto.rub.de/. Supported by ERC Project ERCC (FP7/615074).

[3] Department of Computing and Software, McMaster University, Hamilton, Ontario, Canada. Email: stebilad@mcmaster.ca. URL: https://www.cas.mcmaster.ca/~stebilad/. Supported in part by Australian Research Council (ARC) Discovery Project grant DP130104304 and Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery grant RGPIN-2016-05146.

Contents

1 Introduction

DAPS. Double authentication preventing signature (DAPS) schemes were introduced by Poettering and Stebila (PS) [21]. In such a signature scheme, the message being signed is a pair $m = (a, p)$ consisting of an "address" a and a "payload" p. Let us say that messages $(a_1, p_1), (a_2, p_2)$ are *colliding* if $a_1 = a_2$ but $p_1 \neq p_2$. The double authentication prevention requirement is that there be an efficient extraction algorithm that given a public key PK and valid signatures σ_1, σ_2 on colliding messages $(a, p_1), (a, p_2)$, respectively, returns the secret signing key SK underlying PK. Additionally, the scheme must satisfy standard unforgeability under a chosen-message attack [13], but in light of the first property we must make the restriction that the address components of all messages signed in the attack are different.

Why DAPS? PS [21] suggested that DAPS could deter *certificate subversion*. This is of particular interest now in light of the Snowden revelations. We know that the NSA obtains court orders to compel corporations into measures that compromise security. The case we consider here is that the corporation is a Certificate Authority (CA) and the court order asks it to produce a rogue certificate. Thus, the CA (eg. Comodo, Go Daddy, ...) has already issued a (legitimate) certificate $\text{cert}_1 = (\texttt{example.com}, pk_1, \sigma_1)$ for a server `example.com`. Here pk_1 is the public key of `example.com` and σ_1 is the CA's signature on the pair $(\texttt{example.com}, pk_1)$, computed under the secret key SK of the CA. Big brother (this is what we will call the subverting adversary) is targeting clients communicating with `example.com`. It obtains a court order that requires the CA to issue another certificate —this is the rogue certificate— $\text{cert}_2 = (\texttt{example.com}, pk_2, \sigma_2)$ in the name of `example.com`, where now pk_2 is a public key supplied by big brother, so that the latter knows the corresponding secret key sk_2, and σ_2 is the CA's signature on the pair (`example.com`, pk_2), again computed under the secret key SK of the CA. With this rogue certificate in hand, big brother could impersonate `example.com` in a TLS session with a client, compromising security of `example.com`'s communications.

The CA wants to deny the order (complying with it only hurts its reputation and business) but, under normal conditions, has no argument to make to the court in support of such a denial. Using DAPS to create certificates, rather than ordinary signatures, gives the CA such an argument, namely that complying with the order (issuing the rogue certificate) would compromise not just the security of big brother's target clients communicating with `example.com`, but would compromise security much more broadly. Indeed, if big brother uses the rogue certificate with a client, it puts the rogue certificate in the client's hand. The legitimate certificate can be viewed as public. So the client has σ_1, σ_2. But these are valid signatures on the colliding messages (`example.com`, pk_1), (`example.com`, pk_2), respectively, which means that the client can extract the CA's signing key SK. This would lead to widespread insecurity. The court may be willing to allow big brother to compromise communications of clients with `example.com`, but it will *not* be willing to create a situation where the security of *all* TLS hosts with certificates from this CA is compromised. Ultimately this means the court would have strong incentives to deny big brother's request for a court order to issue a rogue certificate in the first place.

Further discussion of this application of DAPS may be found in [21, 22] and also in our Appendix A. The latter includes comparisons with other approaches such as certificate transparency and public key pinning.

Prior DAPS schemes. PS [21, 22] give a factoring-based DAPS that we call PS. Its signature contains $n+1$ elements in a group $\mathbb{Z}_N^*$, where n is the length of the output of a hash function and N is a (composite) modulus in the public key. With a 2048-bit modulus and 256-bit hash, a signature contains 257 group elements, for a length of 526,336 bits or 64.25 KiB. This is a factor 257 times

longer than a 2048-bit RSA PKCS#1 signature. Signing and verifying times are also significantly greater than for RSA PKCS#1. Ruffing, Kate, and Schröder [24, Appendix A] give a chameleon hash function (CHF) based DAPS that we call RKS and recall in our Appendix B. Instantiating it with DLP-based CHFs makes signing quite efficient, but signature sizes and verification times are about the same as in PS. The large signature sizes in particular of both PS and RKS inhibits their use in practice.

Goals and contributions. If we want DAPS to be a viable practical option, we need DAPS schemes that are competitive with current non-DAPS schemes on *all* cost parameters, meaning signature size, key size, signing time and verifying time. Furthermore, to not lose efficiency via inflated security parameters, we need to establish the unforgeability with tight security reductions. Finally, given the high damage that would be created by certificate forgery, we would prefer these reductions to be to assumptions that are standard (factoring, RSA, ...) rather than new. This is what we deliver. We will give two general methods to build DAPS, and thence obtain many particular schemes that are efficient while having tight security reductions to standard algebraic assumptions. We begin with some background on our main tool, identification schemes.

Background. By an identification scheme we mean a three-move protocol ID where 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. 2 on p. 8). Practical identification schemes are typically Sigma protocols, which means they satisfy honest-verifier zero-knowledge and special soundness. The latter says that from two accepting conversation transcripts with the same commitment but different challenges, one can extract the secret key. The identification scheme is *trapdoor* [18, 2] if 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.

The classic way to get a signature scheme from an identification scheme is via the Fiat-Shamir transform [11], denoted **FS**. Here, 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. This signature scheme meets the standard unforgeability notion of [13] assuming the identification scheme is secure against impersonation under passive attack (IMP-PA) [1]. BPS [2] give several alternative transforms of (trapdoor) identification schemes to unforgeable signature schemes, the advantage over **FS** being that in some cases the reduction of unforgeability to the underlying algebraic assumption is tight. (That of **FS** is notoriously loose.) No prior transform yields DAPS. Our first transform, described next, is however an adaptation and extension of the **MdCmtCh** transform of [2].

Double-hash transform **H2**. The novel challenge in getting DAPS is to provide the double authentication prevention property. Our idea is to turn to identification schemes, and specifically to exploit their special soundness. Recall this says that from two accepting conversations with the same commitment and different challenges, one can extract the secret key. What we want now is to create identification-based signatures in such a way that signatures are accepting conversations and *signatures of messages with the same address have the same commitment, but if payloads differ then challenges differ*. This will allow us, from valid signatures of colliding messages, to obtain the secret key.

To ensure signatures of messages with the same address have the same commitment, we make the commitment a hash of the address. This, however, leaves us in general unable to complete the signing, because the prover in an identification scheme relies on having create the commitment Y in such a way that it knows some underlying private randomness y which is used crucially in the

identification. To get around this, we use identification schemes that are trapdoor (see above), so y can be derived from the commitment given a secret key. To ensure unforgeability, we incorporate a fresh random seed into each signature.

In more detail, our first method to obtain DAPS from a trapdoor identification scheme is via a transform that we call the double-hash transform and denote **H2** (cf. Section 5.1). To sign a message $m = (a, p)$, the signer specifies the commitment as a hash $Y = \mathrm{H}_1(a)$ of the address, picks a random seed s of length sl (a typical seed length would be $\mathsf{sl} = 256$), obtains a challenge $c = \mathrm{H}_2(a\|p\|s)$, uses the trapdoor property of the identification scheme and the secret key to compute a response z, and returns (z, s) as the signature. Additionally the public key is enhanced so that recovery of the secret identification key allows recovery of the full DAPS secret key. Theorem 2 (on p. 13) establishes the double-authentication prevention property via the special soundness property of the identification Sigma protocol, and is unconditional. Theorem 3 shows unforgeability of the DAPS in the ROM under two assumptions on the identification scheme: (1) CIMP-UU, a notion defined in [2], and (2) KR, security against key recovery. Specific identification schemes can be shown to meet both notions under standard assumptions [2], yielding DAPS from the same assumptions. If typical factoring or RSA based identification schemes are used, DAPS signatures have size $k + \mathsf{sl}$, where k is the bitlength of the modulus.

Double-ID transform **ID2**. The signature size $k + \mathsf{sl}$ of **H2** when instantiated with RSA is more than the length k of a signature in RSA PKCS#1. We address this via a second transform of trapdoor identification schemes into DAPS that we call the double ID transform, denoted **ID2**. When instantiated with the same identification schemes as above, corresponding DAPS signatures have length $k + 1$ bits, while maintaining (up to a small constant factor) the signing and verifying times of schemes obtained via **H2**.

The **ID2** transform has several novel features. It requires that the identification scheme supports multiple challenge lengths, specifically challenge lengths 1 and l (think of $l = 256$). To sign a message $m = (a, p)$, first we work with the single challenge-bit version of the identification scheme, computing for this a commitment $Y_1 = H_1(a)$, picking a random 1-bit challenge c_1, and letting z_1 be the response, computed using the trapdoor and secret key. Now a random bijection (a public bijection accessible, in both directions, via oracles) is applied to z_1 to get a commitment Y_2 for the l-bit challenge version of the identification scheme. A challenge for this is computed as $H_2(a, p)$, and then a response z_2 is produced. The signature is simply (c_1, z_2). Section 5.2 specifies the transform in detail and proves the DAP property and unforgeability, modeling the random bijection as ideal. Notably, the CIMP-UU assumption used for the **H2** transform needs to be replaced by the (slightly stronger) CIMP-UC notion [2].

Instantiations. We discuss three different instantiations of the above in Section 6. The RSA-based GQ identification scheme [14] is not trapdoor as usually written, but can be made so by including the decryption exponent in the secret key [2]. Applying **H2** and **ID2**, we get **H2**[GQ] and **ID2**[GQ]. The factoring-based MR identification scheme of Micali and Reyzin [18] is trapdoor, which we exploit to get **H2**[MR]. For details see Fig. 16 and Fig. 19. (Both GQ and MR support multiple challenge lengths and meet the relevant security requirements.) Fig. 1 shows the signing time, verifying time and signature size for these schemes. In a bit we will discuss implementation results that measure actual performance.

Reduction tightness. Fig. 1 says the signing time for **H2**[GQ] is $\mathcal{O}(lk^2+k^3)$, but what this means in practice depends very much on the choice of k (the length of composite N). Roughly speaking, we can expect that doubling k leads to an 8-fold increase in runtime, so signing with $k = 2048$ is 8 times slower than with $k = 1024$. So we want to use the smallest k for which we have a desired

Scheme	Signing		Verifying		|sig|	(bits)
PS [21, 22]	$\mathcal{O}(nk^3)$	516.58 ms	$\mathcal{O}(nk^3)$	161.84 ms	nk	528 384
RKS [24]	$\mathcal{O}(n^4)$	13.48 ms	$\mathcal{O}(n^4)$	5.99 ms	$2n^2$	131 072
H2[GQ]	$\mathcal{O}(lk^2 + k^3)$	0.88 ms	$\mathcal{O}(lk^2)$	0.41 ms	$k + \mathsf{sl}$	2 304
ID2[GQ]		1.80 ms		1.49 ms	$k + 1$	2 049
H2[MR]	$\mathcal{O}(k^3)$	1.27 ms	$\mathcal{O}(lk^2)$	0.37 ms	$k + \mathsf{sl}$	2 304

Figure 1: **DAPS efficiency.** We show performance indications for the DAPS obtained by our **H2** and **ID2** transforms applied to the GQ and MR trapdoor identification schemes. The first two rows show the prior scheme of PS [21, 22] and the scheme of RKS [24], with n being the length of the output of a hash function, eg. $n = 256$. By k we denote the length of a composite modulus N in the public key, eg. $k = 2048$. The challenge length of GQ and MR is l, and sl is the seed length, eg. $l = \mathsf{sl} = 256$. The 4th column is the size of a signature in bits. Absolute runtimes and signature sizes are for $k = 2048$-bit moduli and $n = l = \mathsf{sl} = 256$-bit hashes/challenges/seeds; details appear in Section 6.

---

level of security. Suppose this is approximately 128 bits. Many keylength recommendations match the difficulty of breaking a 128-bit symmetric cipher with the difficulty of factoring a 2048-bit modulus. But this does not generally mean it is safe to use **H2**[GQ] with $k = 2048$, because the reduction of unforgeability to RSA may not be tight: the Fiat-Shamir transform **FS** has a very loose reduction, so when signatures are identification based, one should be extra suspicious. Remarkably, our reductions are tight, so we can indeed get 128 bits of security with $k = 2048$. This tightness has two steps or components. First, the reduction of unforgeability to the CIMP-UU/CIMP-UC and KR security of the identification scheme, as given by Theorem 3 and Theorem 5, is tight. Second, the reductions of CIMP-UU/CIMP-UC and KR to the underlying algebraic problem (here RSA or factoring) are also tight (cf. Lemma 1 on p. 10, adapting [2]).

Implementation. The efficiency measures of Fig. 1 are asymptotic, with hidden constants. Implementation is key to gauge and compare performance in practice. We implement our two GQ based schemes, **H2**[GQ] and **ID2**[GQ], as well as **H2**[MR]. For comparison we also implement the prior PS DAPS, and also compare with the existing implementation of RKS. Fig. 21 (on p. 29) shows the signing time, verifying time, signature size and key sizes for all schemes. **H2**[GQ] emerges as around 587 times faster than PS for signing and 394 times faster for verifying while also having signatures about 229 times shorter. Compared with the previous fastest and smallest DAPS, RKS, **H2**[GQ] is 15× faster for both signing and verifying, with signatures 56× shorter. **ID2**[GQ] is about a factor two slower than **H2**[GQ] but with signatures about 15% shorter. **H2**[MR] has the smallest public keys of our new DAPS schemes, with signing runtime about halfway between **H2**[GQ] and **ID2**[GQ]. The DAPS by RKS remains the one with the smallest public keys, (640 bits), but the schemes in this paper have public keys that are still quite reasonable (between 2048 and 6144 bits). As Fig. 21 shows, **H2**[GQ], **H2**[MR], and **ID2**[GQ] are close to RSA PKCS#1 in all parameters and runtimes (but with potentially improved security, considering our reductions to RSA and factoring are tight). This means that DAPS can replace the signatures currently used for certificates with minimal loss in performance.

Necessity of our assumption. Trapdoor identification schemes may seem a very particular assumption from which to obtain DAPS. However we show in Section 7 that from *any* DAPS

satisfying double authentication prevention and unforgeability, one can build a CIMP-UU and CIMP-UC secure trapdoor identification scheme. This shows that the assumption we make is effectively necessary for DAPS.

FURTHER RELATED WORK AND OPEN QUESTIONS. Ordinary signatures are possible from any one-way function [23]. Is it possible to obtain DAPS from any one-way function? Or, can one give some evidence that this will not be true, for example by showing that DAPS implies a primitive like key exchange that is not likely to be possible based on one-way functions [15]?

The DAPS property that the secret key is recoverable from signatures of colliding messages is conceptually similar to the recoverability of the spender's identity from double-spending of an e-coin in offline e-cash [7]. Interestingly, the RKS DAPS emerges from designing double-spending resistant e-cash.

2 Preliminaries

NOTATION. By ε we denote the empty string. If X is a finite set, then $x \leftarrow\!\!\$\, X$ denotes selecting an element of X uniformly at random and $|X|$ denotes the size of X. We use $a_1\|a_2\|\cdots\|a_n$ as shorthand for $(a_1, a_2, \ldots, a_n)$, and by $a_1\|a_2\|\cdots\|a_n \leftarrow x$ we mean that x is parsed into its constituents. If A is an algorithm, $y \leftarrow A(x_1, \ldots; r)$ denotes running A on inputs $x_1, \ldots$ with random coins r and assigning the result to y, and $y \leftarrow\!\!\$\, A(x_1, \ldots)$ means we pick r at random and let $y \leftarrow A(x_1, \ldots; r)$. By $[A(x_1, \ldots)]$ we denote the set of all y that have positive probability of being returned by $A(x_1, \ldots)$.

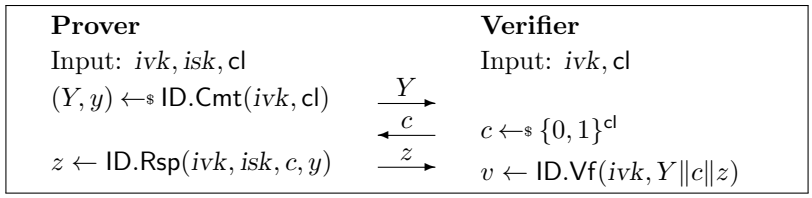
Our proofs use the code-based game playing framework of BR [5]. In these proofs, $\Pr[\mathrm{G}]$ denotes the event that game G returns true. When we speak of running time of algorithms, we mean worst case. For adversaries playing games, this includes the running time of the adversary and that of the game, i.e., the time taken by game procedures to respond to oracle queries is included. Boolean flags (like bad) in games are assumed initialized to false.

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 l. To provide a single unified notation, following [2], we have the game procedure H 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.

Our **ID2** transform also relies on a random bijection. In the spirit of a random oracle, a random bijection is an idealized unkeyed public bijection to which algorithms and adversaries have access via two oracles, one for the forward direction and one for the backward direction. Cryptographic constructions that build on such objects include the Even-Mansour cipher and the SHA3 hash function. We denote by $\Pi^{+}(\cdot, \mathrm{Dom}, \mathrm{Rng})$ a bijection from Dom to Rng, and we denote its inverse with Π^{-1}. Once Dom and Rng are fixed, our results view $\Pi^{+1}(\cdot, \mathrm{Dom}, \mathrm{Rng})$ as being randomly sampled from the set of all bijections from Dom to Rng. We discuss instantiation of a random bijection in Section 6.

For simplicity, we typically omit idealized primitives (random oracles, random bijections) from the games defining security, with the understanding that if a scheme is defined relative to an idealized primitive, then the adversary has access to this primitive. Algorithms of the scheme of course also have such access as indicated by the scheme description.

SIGNATURE SCHEMES. A signature scheme DS specifies the following. The signer runs key generation algorithm $\mathsf{DS.Kg}$ to get a verification key vk and a signing key sk. A signature of message m is generated via $\sigma \leftarrow\!\!\$\, \mathsf{DS.Sig}(vk, sk, m)$. Verification is done by $v \leftarrow \mathsf{DS.Vf}(vk, m, \sigma)$, which returns



Game $\mathbf{G}^{\mathrm{ex}}_{\mathsf{ID}}(\mathcal{A})$	Game $\mathbf{G}^{\mathrm{zk}}_{\mathsf{ID},\mathsf{cl}}(\mathcal{A})$
$(ivk, isk, itk) \leftarrow\!\!\$\ \mathsf{ID.Kg}$	$(ivk, isk, itk) \leftarrow\!\!\$\ \mathsf{ID.Kg}\ ;\ b \leftarrow\!\!\$\ \{0,1\}$
$(Y, c_1, z_1, c_2, z_2) \leftarrow\!\!\$\ \mathcal{A}(ivk, isk, itk)$	$(Y_1, y_1) \leftarrow\!\!\$\ \mathsf{ID.Cmt}(ivk, \mathsf{cl})$
$T_1 \leftarrow Y\|c_1\|z_1\ ;\ T_2 \leftarrow Y\|c_2\|z_2$	$c_1 \leftarrow\!\!\$\ \{0,1\}^{\mathsf{cl}}$
$v_1 \leftarrow \mathsf{ID.Vf}(ivk, T_1)\ ;\ v_2 \leftarrow \mathsf{ID.Vf}(ivk, T_2)$	$z_1 \leftarrow \mathsf{ID.Rsp}(ivk, isk, c_1, y_1)$
If $\neg v_1 \vee \neg v_2 \vee (\lvert c_1\rvert \neq \lvert c_2\rvert) \vee (c_1 = c_2)$: Return false	$Y_0\|c_0\|z_0 \leftarrow\!\!\$\ \mathsf{ID.Sim}(ivk, \mathsf{cl})$
$isk^* \leftarrow\!\!\$\ \mathsf{ID.Ex}(ivk, T_1, T_2)$	$b' \leftarrow\!\!\$\ \mathcal{A}(ivk, \mathsf{cl}, Y_b\|c_b\|z_b)$
Return $(isk^* \neq isk)$	Return $(b = b')$

Figure 2: **Top:** Message flow of an identification scheme ID. **Bottom:** Games defining extractability and HVZK of an identification scheme ID.

---

a boolean v. DS is correct if for all $(vk, sk) \in [\mathsf{DS.Kg}]$, all messages $m \in \{0,1\}^*$ and all signatures $\sigma \in [\mathsf{DS.Sig}(vk, sk, m)]$, we have $\mathsf{DS.Vf}(vk, m, \sigma) = \mathsf{true}$.

3 Identification schemes

Identification schemes are our main tool. Here we give the necessary definitions and results.

Identification. An identification (ID) scheme ID is a three-move protocol between a prover and a verifier, as shown in Fig. 2. A novel feature of our formulation (which we exploit for the **ID2** transform) is that identification schemes support challenges of multiple lengths. Thus, associated to ID is a set $\mathsf{ID.clS} \subseteq \mathbb{N}$ of admissible challenge lengths. At setup time the prover runs key generation algorithm $\mathsf{ID.Kg}$ to generate a public *verification key* ivk, a private *identification key* isk, and a *trapdoor* itk. To execute a run of the identification scheme for a challenge length $\mathsf{cl} \in \mathsf{ID.clS}$, the prover runs $\mathsf{ID.Cmt}(ivk, \mathsf{cl})$ to generate a *commitment* Y and a private state y. The prover sends Y to the verifier, who samples a random *challenge* c of length cl and returns it to the prover. The prover computes its *response* $z \leftarrow \mathsf{ID.Rsp}(ivk, isk, c, y)$. The verifier checks the response by invoking $\mathsf{ID.Vf}(ivk, Y\|c\|z)$ which returns a boolean value. We require perfect correctness. For any ivk, cl we denote with $\mathsf{ID.CS}(ivk, \mathsf{cl})$ and $\mathsf{ID.RS}(ivk, \mathsf{cl})$ the space of commitments and responses, respectively.

In basic ID schemes, key generation only outputs ivk and isk. The inclusion of itk was given by [2] in their definition of trapdoor ID schemes. Following [2] (and extending to multiple challenge lengths) we say ID is trapdoor if it specifies an additional algorithm $\mathsf{ID.Cmt}^{-1}$ that can compute y from any Y using trapdoor itk. The property required of $\mathsf{ID.Cmt}^{-1}$ is that the following two distributions on (Y, y) are identical for any admissible challenge length cl: (1) Let $(ivk, isk, itk) \leftarrow\!\!\$\ \mathsf{ID.Kg}\ ;\ (Y, y) \leftarrow\!\!\$\ \mathsf{ID.Cmt}(ivk, \mathsf{cl})$ and return (Y, y), and (2) Let $(ivk, isk, itk) \leftarrow\!\!\$\ \mathsf{ID.Kg}\ ;\ Y \leftarrow\!\!\$\ \mathsf{ID.CS}(ivk, \mathsf{cl})\ ;\ y \leftarrow\!\!\$\ \mathsf{ID.Cmt}^{-1}(ivk, itk, Y, \mathsf{cl})$ and return (Y, y).

Further properties. We give several further identification-related definitions we will use. First we extend honest-verifier zero-knowledge (HVZK) and extractability to identification schemes with variable challenge length.

HVZK of ID asks that there exists an algorithm $\mathsf{ID.Sim}$ (called the simulator) that given the verification key and challenge length, generates transcripts which have the same distribution as

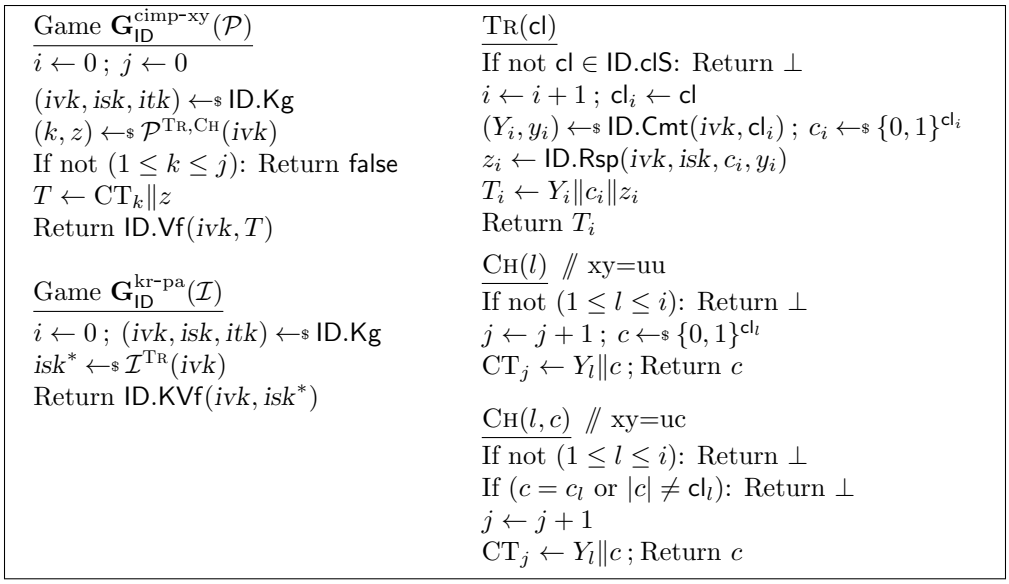


Figure 3: Games defining security of identification scheme ID against constrained impersonation (CIMP-UU and CIMP-UC) and against key recovery under passive attack.

---

honest ones. Formally, if $\mathcal{A}$ is an adversary and $\mathsf{cl} \in \mathsf{ID.clS}$ is an admissible challenge length, let $\mathbf{Adv}^{\text{zk}}_{\mathsf{ID},\mathsf{cl}}(\mathcal{A}) = 2\Pr[\mathbf{G}^{\text{zk}}_{\mathsf{ID},\mathsf{cl}}(\mathcal{A})] - 1$ where the game is shown in Fig. 2. Then ID is HVZK if $\mathbf{Adv}^{\text{zk}}_{\mathsf{ID},\mathsf{cl}}(\mathcal{A}) = 0$ for all (even computationally unbounded) adversaries $\mathcal{A}$ and all $\mathsf{cl} \in \mathsf{ID.clS}$.

Extractability of ID asks that there exists an algorithm ID.Ex (called the extractor) which from any two (valid) transcripts that have the same commitment but different same-length challenges can recover the secret key. Formally, if $\mathcal{A}$ is an adversary, let $\mathbf{Adv}^{\text{ex}}_{\mathsf{ID}}(\mathcal{A}) = \Pr[\mathbf{G}^{\text{ex}}_{\mathsf{ID}}(\mathcal{A})]$ where the game is shown in Fig. 2. Then ID is perfectly extractable if $\mathbf{Adv}^{\text{ex}}_{\mathsf{ID}}(\mathcal{A}) = 0$ for all (even computationally unbounded) adversaries $\mathcal{A}$. Perfect extractability is sometimes called special soundness. We say that an identification scheme is a Sigma protocol [8] if it is both HVZK and perfectly extractable.

We define three further notions that are not standard, but sometimes needed and true of typical schemes (cf. Section 6). For instance, at times we require that ID includes a *key-verification algorithm* $\mathsf{ID.KVf}$ for which $\mathsf{ID.KVf}(ivk, isk) = \mathsf{true}$ iff $(ivk, isk, itk) \in [\mathsf{ID.Kg}]$ for some itk. We say that ID is *commitment recovering* if $\mathsf{ID.Vf}$ verifies a transcript $Y\|c\|z$ by recovering Y from c, z and then comparing. More precisely, we require that there exist an efficient algorithm $\mathsf{ID.Rsp}^{-1}$ that takes a verification key, a challenge, and a response, and outputs a commitment, such that $\mathsf{ID.Vf}(ivk, Y\|c\|z) = \mathsf{true}$ iff $Y = \mathsf{ID.Rsp}^{-1}(ivk, c, z)$. Finally, ID is said to have *unique responses* if for any commitment Y and any challenge c there is precisely one response z such that we have $\mathsf{ID.Vf}(ivk, Y\|c\|z) = \mathsf{true}$.

Security of identification. A framework of notions of security under constrained impersonation was given in [2]. We reproduce and use their CIMP-UU and CIMP-UC notions but extend them to support multiple challenge lengths. The value of these notions as starting points is that they can be proven to be achieved by typical identification schemes with *tight* reductions to *standard* assumptions, following [2], which is not true of classical notions like IMP-PA (impersonation under passive attack [1]). The formalization relies on the games $\mathbf{G}^{\text{cimp-xy}}_{\mathsf{ID}}(\mathcal{P})$ of Fig. 3 associated to identification scheme ID and adversary $\mathcal{P}$, where $\text{xy} \in \{\text{uu}, \text{uc}\}$. The transcript oracle Tr returns a fresh identification transcript $Y_i\|c_i\|z_i$ each time it is called, for a challenge length passed in by

the adversary. This models a passive attack. In the xy = uu case, the adversary can call CH with the index l of an existing transcript $Y_l \| c_l \| z_l$ to indicate that it wants to be challenged to produce a response for a fresh challenge against the commitment Y_l. The index j records the session for future reference. In the xy = uc case, the adversary continues to call CH with the index l of an existing transcript, but this time provides its own challenge c, indicating it wants to be challenged to find a response. The game allows this only if the provided challenge is different from the one in the original transcript. The adversary can call TR and CH as many times as it wants, in any order. The adversary terminates by outputting the index k of a challenge session against which it hopes its response z will verify. Define the advantage via $\mathbf{Adv}^{\text{cimp-xy}}_{\mathsf{ID}}(\mathcal{P}) = \Pr[\mathbf{G}^{\text{cimp-xy}}_{\mathsf{ID}}(\mathcal{P})]$.

We also define a metric of security of the identification scheme against key recovery under passive attack. The formalization considers game $\mathbf{G}^{\text{kr-pa}}_{\mathsf{ID}}(\mathcal{I})$ of Fig. 3 associated to identification scheme ID and kr adversary $\mathcal{I}$. The transcript oracle TR is as before. Adversary $\mathcal{I}$ aims to find a private key isk^* that is functionally equivalent to isk in the sense that $\mathsf{ID.KVf}(ivk, isk^*) = \mathsf{true}$. (In particular, it certainly succeeds if it recovers the private key isk.) We let $\mathbf{Adv}^{\text{kr-pa}}_{\mathsf{ID}}(\mathcal{I}) = \Pr[\mathbf{G}^{\text{kr-pa}}_{\mathsf{ID}}(\mathcal{I})]$ be the probability that it succeeds. The notion of KR security from [20, 2] did not give the adversary a TR oracle (excluding even passive attacks) and required that for success it find the target key isk (rather than, as here, being allowed to get away with something functionally equivalent).

ACHIEVING THE NOTIONS. For typical identification schemes that are HVZK, security against key recovery under passive attack corresponds exactly to the standard assumption underlying the scheme, for example the one-wayness of RSA for GQ. The following says that under the assumption of security against key recovery under passive attack, we can establish both CIMP-UC and CIMP-UU for identification schemes that are extractable. In the second case, however, we require that the challenge-lengths used be large.

The identification schemes we will use to build DAPS are Sigma protocols, meaning perfectly extractable, and hence for these schemes $\mathbf{Adv}^{\text{ex}}_{\mathsf{ID}}(\mathcal{A})$ below will be 0. We give a more general result because we will use it in Section 7. We omit the proof as it uses standard arguments [2].

Lemma 1 *Let ID be an identification scheme. Then for any adversary $\mathcal{P}$ against* CIMP-UC *we construct a key recovery adversary $\mathcal{I}$ and extraction adversary $\mathcal{A}$ such that*

$$\mathbf{Adv}^{\text{cimp-uc}}_{\mathsf{ID}}(\mathcal{P}) \le \mathbf{Adv}^{\text{kr-pa}}_{\mathsf{ID}}(\mathcal{I}) + \mathbf{Adv}^{\text{ex}}_{\mathsf{ID}}(\mathcal{A}) \ . \tag{1}$$

Also for any adversary $\mathcal{P}$ against CIMP-UU *that makes q_c queries to its* CH *oracle, each with challenge length at least cl, we construct a key recovery adversary $\mathcal{I}$ such that*

$$\mathbf{Adv}^{\text{cimp-uu}}_{\mathsf{ID}}(\mathcal{P}) \le \mathbf{Adv}^{\text{kr-pa}}_{\mathsf{ID}}(\mathcal{I}) + \mathbf{Adv}^{\text{ex}}_{\mathsf{ID}}(\mathcal{A}) + q_c \cdot 2^{-\mathsf{cl}} \ . \tag{2}$$

In both cases, the running times of $\mathcal{I}$ and $\mathcal{A}$ are about that of $\mathcal{P}$ plus the time for one execution of $\mathsf{ID.Ex}$.

Above, CIMP-UU was established assuming long challenges. We note that this is necessary, meaning CIMP-UU does not hold for short challenges, such as one-bit ones. To see this, assume $\mathsf{cl} \in \mathsf{ID.clS}$ and $q \ge 1$ is a parameter. Consider the following attack (adversary) $\mathcal{P}$. It makes a single query $Y \| c \| z \leftarrow\!\!\$\ \text{TR}(\mathsf{cl})$. Then for $i = 1, \ldots, q$ it queries $c_i \leftarrow\!\!\$\ \text{CH}(1)$. If there is a k such that $c_k = c$ then it returns (k, z) and wins, else it returns $\perp$. We have

$$\mathbf{Adv}^{\text{cimp-uu}}_{\mathsf{ID}}(\mathcal{P}) = 1 - \left(1 - \frac{1}{2^{\mathsf{cl}}}\right)^q \approx \frac{q}{2^{\mathsf{cl}}} \ . \tag{3}$$

Thus, roughly, the attack succeeds in time 2^{cl}, so if the latter is small, CIMP-UU security will not hold. Our **H2** transform will use long challenges and be able to rely only on CIMP-UU, but our

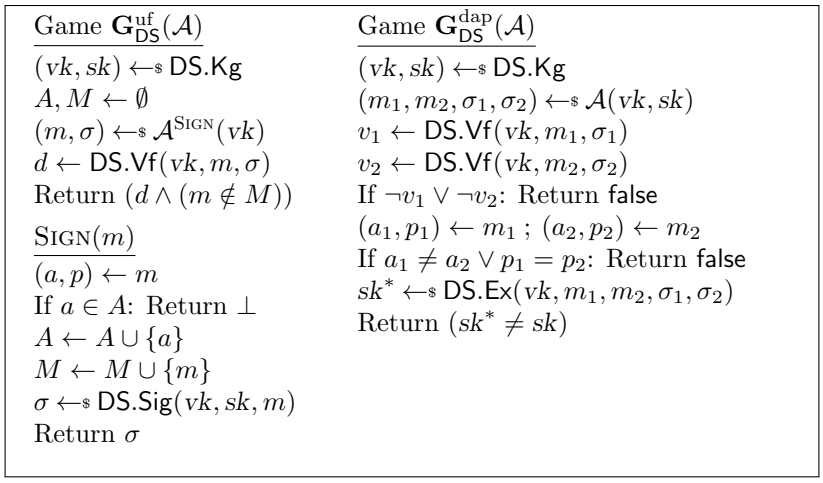


Figure 4: Games defining unforgeability and the DAP property of signature scheme DS.

ID2 transform will require security on both long and short (1-bit) challenges, and thus will rely on CIMP-UC in addition to CIMP-UU. We note that given Lemma 1, we could use CIMP-UC throughout, but for the reductions it is simpler and more convenient to work with CIMP-UU when possible.

4 DAPS definitions

Let DS be a signature scheme. When used as a DAPS [21, 22], a message $m = (a, p)$ for DS is a pair consisting of an *address* a and a *payload* p. We require (1) the double authentication prevention (DAP) property and (2) a restricted form of unforgeability, as defined below.

The DAP property. Call messages $m_1 = (a_1, p_1)$ and $m_2 = (a_2, p_2)$ *colliding* if $a_1 = a_2$ but $p_1 \neq p_2$. Double authentication prevention (DAP) [21, 22] requires that possession of signatures on colliding messages allow anyone to extract the signing key. It is captured formally by the advantage $\mathbf{Adv}^{\mathrm{dap}}_{\mathsf{DS}}(\mathcal{A}) = \Pr[\mathbf{G}^{\mathrm{dap}}_{\mathsf{DS}}(\mathcal{A})]$ associated to adversary $\mathcal{A}$, where game $\mathbf{G}^{\mathrm{dap}}_{\mathsf{DS}}(\mathcal{A})$ is in Fig. 4. The adversary produces messages m_1, m_2 and signatures σ_1, σ_2, and an extraction algorithm $\mathsf{DS.Ex}$ associated to the scheme then attempts to compute sk. The adversary wins if the key sk^* produced by $\mathsf{DS.Ex}$ is different from sk yet extraction should have succeeded, meaning the messages were colliding and their signatures were valid. The adversary has sk as input to cover the fact that the signer is the one attempting —due to coercion and subversion, but nonetheless— to produce signatures on colliding messages. (And thus it does not need access to a Sign oracle.) We note that we are not saying it is hard to produce signatures on colliding messages —it isn't, given sk— but rather that doing so will reveal sk. We also stress that extraction is not required just for honestly-generated signatures, but for *any*, even adversarially generated signatures that are valid, again because the signer is the adversary here.

Unforgeability. Let $\mathbf{Adv}^{\mathrm{uf}}_{\mathsf{DS}}(\mathcal{A}) = \Pr[\mathbf{G}^{\mathrm{uf}}_{\mathsf{DS}}(\mathcal{A})]$ be the uf-advantage associated to adversary $\mathcal{A}$, where game $\mathbf{G}^{\mathrm{uf}}_{\mathsf{DS}}(\mathcal{A})$ is in Fig. 4. This is the classical notion of [13], except that addresses of the messages the signer signs must be all different, as captured through the set A in the game. This is necessary because the double authentication prevention requirement precludes security if the signer releases signatures of two messages with the same address. In practice it means that the signer must maintain a log of all messages it has signed and make sure that it does not sign two messages

with the same address. A CA is likely to maintain such a log in any case so this is unlikely to be an extra burden.

Discussion. Regarding the dap property, asking that the key sk^* returned by the extractor $\mathsf{DS.Ex}$ be equal to sk may seem unnecessarily strong. It might suffice if sk^* was "functionally equivalent" to sk, allowing computation of signatures that could not be distinguished from real ones. Such a property is considered in PS [22]. Formalizing it would require adding another security game based on indistinguishability. As our schemes (as well as the ones from [21, 22]) achieve the simpler and stronger property we have defined, we adopt it in our definition.

The dap game chooses the keys vk, sk honestly. Allowing these to be adversarially chosen would result in a stronger requirement, also formalized in PS [21, 22]. Our view is that our (weaker) requirement is appropriate for the application we envision because the CA does not wish to create rogue certificates and has no incentive to create keys allowing it, and the court order happens after the CA and its keys are established, so that key establishment is honest. If the choice of keys is considered a potential source of vulnerability, one might generate them via secure computation between a few different parties.

5 Our ID to DAPS transforms

We specify and analyze our two generic transformations, **H2** and **ID2**, of trapdoor identification schemes to DAPS. Both deliver efficient DAPS, signature sizes being somewhat smaller in the second case.

5.1 The double-hash transform

The construction. Let ID be a trapdoor identification scheme. Our **H2** (double hash) transform associates to it, a supported challenge length $\mathsf{cl} \in \mathsf{ID.clS}$, and a seed length $\mathsf{sl} \in \mathbb{N}$, a DAPS $\mathsf{DS} = \mathbf{H2}[\mathsf{ID}, \mathsf{cl}, \mathsf{sl}]$. The algorithms of DS are defined in Fig. 5. We give some intuition on the design. In the signing algorithm, we specify the commitment Y as a hash of the address, i.e., messages with the same address result in transcripts with the same commitment. We then specify the challenge c as a hash of the message (i.e., address and payload) and a random seed. Signatures consist of the seed and the corresponding response. Concerning the extractability property, observe that the $\mathsf{ID.Ex}$ algorithm, when applied to colliding signature transcripts, reveals isk but not itk, whereas DAPS extraction needs to recover both, i.e., the full secret key $sk = (isk, itk)$. We resolve this by putting in the verification key a particular encryption, denoted ITK, of itk, under isk (we assume itk can be encoded in tl bits).

The scheme uses random oracles $\mathrm{H}(\cdot, \{0,1\}^{\mathsf{tl}})$, $\mathrm{H}(\cdot, \mathsf{ID.CS}(ivk, \mathsf{cl}))$ and $\mathrm{H}(\cdot, \{0,1\}^{\mathsf{cl}})$. For simplicity it is assumed that the three range sets involved here are distinct, which makes the random oracles independent. If the range sets are not distinct, the scheme must be modified to use domain separation [4] in calling these oracles. This can be done simply by prefixing the query to the i-th oracle with i ($i = 1, 2, 3$ for our three oracles).

DAP security of our construction. The following confirms that double authentication prevention is achieved. This is quite straightforward given the construction. We model H as a random oracle. Per our conventions, the number of (distinct) queries q to $\mathrm{H}(\cdot, \{0,1\}^{\mathsf{cl}})$, referred to below, reflects the overall number of queries made to this oracle in the execution of the game $\mathbf{G}^{\mathrm{dap}}_{\mathsf{DS}}(\mathcal{A})$, including queries made by $\mathcal{A}$ directly and queries made by game procedures; as a result it will always be the case that $q \geq 2$.

$\underline{\mathbf{H2}[\mathsf{ID},\mathsf{cl},\mathsf{sl}].\mathsf{Kg}^{\mathrm{H}}}$
$(ivk, isk, itk) \leftarrow\!\!\$\ \mathsf{ID.Kg}$
$ITK \leftarrow itk \oplus \mathrm{H}(isk, \{0,1\}^{\mathsf{tl}})$
$vk \leftarrow (ivk, ITK)$; $sk \leftarrow (isk, itk)$
Return (vk, sk)

$\underline{\mathbf{H2}[\mathsf{ID},\mathsf{cl},\mathsf{sl}].\mathsf{Ex}^{\mathrm{H}}(vk, m_1, m_2, \sigma_1, \sigma_2)}$
$(ivk, ITK) \leftarrow vk$
For $i = 1, 2$ do
  $(a_i, p_i) \leftarrow m_i$; $(z_i, s_i) \leftarrow \sigma_i$
  $Y_i \leftarrow \mathrm{H}(a_i, \mathsf{ID.CS}(ivk, \mathsf{cl}))$
  $c_i \leftarrow \mathrm{H}(a_i \| p_i \| s_i, \{0,1\}^{\mathsf{cl}})$
$isk^* \leftarrow \mathsf{ID.Ex}(ivk, Y_1 \| c_1 \| z_1, Y_2 \| c_2 \| z_2)$
$itk^* \leftarrow \mathrm{H}(isk^*, \{0,1\}^{\mathsf{tl}}) \oplus ITK$
$sk^* \leftarrow (isk^*, itk^*)$; Return sk^*

$\underline{\mathbf{H2}[\mathsf{ID},\mathsf{cl},\mathsf{sl}].\mathsf{Sig}^{\mathrm{H}}(vk, sk, m)}$
$(ivk, ITK) \leftarrow vk$; $(isk, itk) \leftarrow sk$
$(a, p) \leftarrow m$; $s \leftarrow\!\!\$\ \{0,1\}^{\mathsf{sl}}$
$Y \leftarrow \mathrm{H}(a, \mathsf{ID.CS}(ivk, \mathsf{cl}))$
$y \leftarrow\!\!\$\ \mathsf{ID.Cmt}^{-1}(ivk, itk, Y, \mathsf{cl})$
$c \leftarrow \mathrm{H}(a \| p \| s, \{0,1\}^{\mathsf{cl}})$
$z \leftarrow \mathsf{ID.Rsp}(ivk, isk, c, y)$
$\sigma \leftarrow (z, s)$; Return σ

$\underline{\mathbf{H2}[\mathsf{ID},\mathsf{cl},\mathsf{sl}].\mathsf{Vf}^{\mathrm{H}}(vk, m, \sigma)}$
$(ivk, ITK) \leftarrow vk$; $(a, p) \leftarrow m$; $(z, s) \leftarrow \sigma$
$Y \leftarrow \mathrm{H}(a, \mathsf{ID.CS}(ivk, \mathsf{cl}))$
$c \leftarrow \mathrm{H}(a \| p \| s, \{0,1\}^{\mathsf{cl}})$
Return $\mathsf{ID.Vf}(ivk, Y \| c \| z)$

Figure 5: Our construction of a DAPS $\mathbf{H2}[\mathsf{ID},\mathsf{cl},\mathsf{sl}]$ from a trapdoor identification scheme ID, a challenge length $\mathsf{cl} \in \mathsf{ID.clS}$, and a seed length $\mathsf{sl} \in \mathbb{N}$.

Theorem 2 *Let DAPS* $\mathsf{DS} = \mathbf{H2}[\mathsf{ID},\mathsf{cl},\mathsf{sl}]$ *be obtained from trapdoor identification scheme* ID*, challenge length* cl*, and seed length* sl *as above. Let* $\mathcal{A}$ *be an adversary making* $q \geq 2$ *distinct* $\mathrm{H}(\cdot, \{0,1\}^{\mathsf{cl}})$ *queries. If* ID *has perfect extractability then*

$$\mathbf{Adv}^{\mathrm{dap}}_{\mathsf{DS}}(\mathcal{A}) \leq q(q-1)/2^{\mathsf{cl}+1} \quad . \tag{4}$$

Proof of Theorem 2: In game $\mathbf{G}^{\mathrm{dap}}_{\mathsf{DS}}(\mathcal{A})$ of Fig. 4, consider the execution of the algorithm $\mathsf{DS.Ex}^{\mathrm{H}}$ of Fig. 5 on $vk, m_1, m_2, \sigma_2, \sigma_2$ where $(m_1, m_2, \sigma_1, \sigma_2) \leftarrow\!\!\$\ \mathcal{A}^{\mathrm{H}}(vk, sk)$. Let $Y_1 \| c_1 \| z_1, Y_2 \| c_2 \| z_2$ be the transcripts computed within. Assume σ_1, σ_2 are valid signatures of m_1, m_2, respectively, relative to $vk = (ivk, ITK)$. As per the verification algorithm $\mathsf{DS.Vf}^{\mathrm{H}}$ of Fig. 5 this means that the transcripts $Y_1 \| c_1 \| z_1, Y_2 \| c_2 \| z_2$ are valid under the ID scheme, meaning $\mathsf{ID.Vf}(ivk, Y_1 \| c_1 \| z_1) = \mathsf{ID.Vf}(ivk, Y_2 \| c_2 \| z_2) = \mathsf{true}$. If the messages $m_1 = (a_1, p_1)$ and $m_2 = (a_2, p_2)$ output by $\mathcal{A}$ are colliding then we also have $Y_1 = Y_2$. This is because $a_1 = a_2$ and verification ensures that $Y_1 = \mathrm{H}(a_1, \mathsf{ID.CS}(ivk, \mathsf{cl}))$ and $Y_2 = \mathrm{H}(a_2, \mathsf{ID.CS}(ivk, \mathsf{cl}))$. So if $c_1 \neq c_2$ then the extraction property of ID ensures that $isk^* = isk$. If so, we also can obtain $itk^* = itk$, so that the full secret key $sk = (isk, itk)$ is recovered. So $\mathbf{Adv}^{\mathrm{dap}}_{\mathsf{DS}}(\mathcal{A})$ is at most the probability that the challenges are equal even though the payloads are not. But the challenges are outputs of $\mathrm{H}(\cdot, \{0,1\}^{\mathsf{cl}})$, to which the game makes at most q queries. So the probability that these challenges collide is at most $q(q-1)/2^{\mathsf{cl}+1}$. ∎

We note this proof does not essentially rely on H being a random oracle, and could be adapted to rely solely on the collision resistance of H.

Unforgeability of our construction. The following shows that the restricted unforgeability of our DAPS tightly reduces to the cimp-uu plus kr security of the underlying ID scheme. As before we model H as a random oracle.

Theorem 3 *Let DAPS* $\mathsf{DS} = \mathbf{H2}[\mathsf{ID},\mathsf{cl},\mathsf{sl}]$ *be obtained from trapdoor identification scheme* ID*, challenge length* cl*, and seed length* sl *as in Fig. 5. Let* $\mathcal{A}$ *be a uf adversary against* DS *and suppose the number of queries that* $\mathcal{A}$ *makes to its* $\mathrm{H}(\cdot, \{0,1\}^{\mathsf{tl}})$*,* $\mathrm{H}(\cdot, \mathsf{ID.CS}(ivk, \mathsf{cl}))$*,* $\mathrm{H}(\cdot, \{0,1\}^{\mathsf{cl}})$*,* Sign *oracles are, respectively,* q_1, q_2, q_3, q_s*. Then from* $\mathcal{A}$ *we can construct* cimp-uu *adversary* $\mathcal{P}$ *and kr*

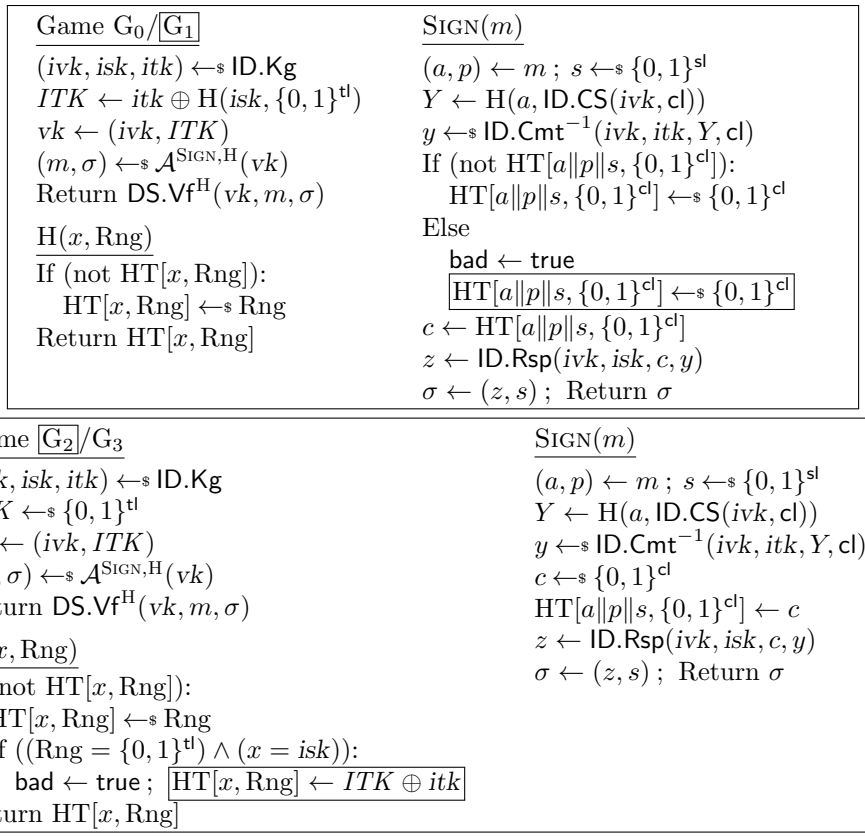


Figure 6: Games for proof of Theorem 3. Games $\mathrm{G}_1, \mathrm{G}_2$ include the boxed code and games $\mathrm{G}_0, \mathrm{G}_3$ do not.

---

adversary $\mathcal{I}$ such that

$$\mathbf{Adv}^{\mathrm{uf}}_{\mathsf{DS}}(\mathcal{A}) \leq \mathbf{Adv}^{\text{cimp-uu}}_{\mathsf{ID}}(\mathcal{P}) + \mathbf{Adv}^{\text{kr-pa}}_{\mathsf{ID}}(\mathcal{I}) + \frac{q_s(2q_3 + q_s - 1)}{2^{\mathsf{sl}+1}} \quad . \tag{5}$$

Adversaries $\mathcal{P}, \mathcal{I}$ make $q_2 + q_s + 1$ queries to Tr. *Adversary $\mathcal{P}$ makes q_3 queries to* Ch. *The running time of adversary $\mathcal{P}$ is about that of $\mathcal{A}$. The running time of adversary $\mathcal{I}$ is that of $\mathcal{A}$ plus the time for q_1 executions of* $\mathsf{ID.KVf}$.

Proof of Theorem 3: We assume that $\mathcal{A}$ avoids certain pointless behavior that would only cause it to lose. Thus, we assume that, in the messages it queries to Sign, the addresses are all different. Also we assume it did not query to Sign the message m in the forgery (m, σ) that it eventually outputs. The two together mean that the sets A, M in game $\mathbf{G}^{\mathrm{uf}}_{\mathsf{DS}}(\mathcal{A})$, and the code and checks associated with them, are redundant and can be removed. We will work with this simplified form of the game, that we call G_0.

Identical-until-bad games $\mathrm{G}_0, \mathrm{G}_1$ of Fig. 6 move us to allow picking a random seed in responding to a Sign query, regardless of whether the corresponding hash table entry was defined or not. We have

$$\begin{aligned} \mathbf{Adv}^{\mathrm{uf}}_{\mathsf{DS}}(\mathcal{A}) = \Pr[\mathrm{G}_0] &= \Pr[\mathrm{G}_1] + \Pr[\mathrm{G}_0] - \Pr[\mathrm{G}_1] \\ &\leq \Pr[\mathrm{G}_1] + \Pr[\mathrm{G}_0 \text{ sets } \mathsf{bad}] \ , \end{aligned} \tag{6}$$

Game G_4

$(ivk, isk, itk) \leftarrow\!\!\$\ \mathsf{ID.Kg}$
$ITK \leftarrow\!\!\$\ \{0,1\}^{\mathsf{tl}}$
$vk \leftarrow (ivk, ITK)$
For $i = 1, \ldots, q_2 + q_s + 1$ do
  $(Y_i, y_i) \leftarrow\!\!\$\ \mathsf{ID.Cmt}(ivk, \mathsf{cl})$
  $c_i \leftarrow\!\!\$\ \{0,1\}^{\mathsf{cl}}$
  $z_i \leftarrow \mathsf{ID.Rsp}(ivk, isk, c_i, y_i)$
$i_2 \leftarrow 0$
$(m, \sigma) \leftarrow\!\!\$\ \mathcal{A}^{\textsc{Sign},\mathrm{H}}(vk)$
$(a, p) \leftarrow m$; $(z, s) \leftarrow \sigma$
$Y \leftarrow \mathrm{H}(a, \mathsf{ID.CS}(ivk, \mathsf{cl}))$
$c \leftarrow \mathrm{H}(a \| p \| s, \{0,1\}^{\mathsf{cl}})$
Return $\mathsf{ID.Vf}(ivk, Y \| c \| z)$

Game G_5

$(ivk, isk, itk) \leftarrow\!\!\$\ \mathsf{ID.Kg}$
$ITK \leftarrow\!\!\$\ \{0,1\}^{\mathsf{tl}}$
$vk \leftarrow (ivk, ITK)$
For $i = 1, \ldots, q_2 + q_s + 1$ do
  $(Y_i, y_i) \leftarrow\!\!\$\ \mathsf{ID.Cmt}(ivk, \mathsf{cl})$
  $c_i \leftarrow\!\!\$\ \{0,1\}^{\mathsf{cl}}$
  $z_i \leftarrow \mathsf{ID.Rsp}(ivk, isk, c_i, y_i)$
$i_2 \leftarrow 0$; $T \leftarrow \emptyset$
$(m, \sigma) \leftarrow\!\!\$\ \mathcal{A}^{\textsc{Sign},\mathrm{H}}(vk)$
Return $(isk \in T)$

$\textsc{Sign}(m)$ // $\mathrm{G}_4, \mathrm{G}_5$

$(a, p) \leftarrow m$; $s \leftarrow\!\!\$\ \{0,1\}^{\mathsf{sl}}$
$Y \leftarrow \mathrm{H}(a, \mathsf{ID.CS}(ivk, \mathsf{cl}))$
$i \leftarrow \mathrm{Ind}_2(a)$
$\mathrm{HT}[a \| p \| s, \{0,1\}^{\mathsf{cl}}] \leftarrow c_i$
$\sigma \leftarrow (z_i, s)$; Return σ

$\mathrm{H}(x, \mathrm{Rng})$ // G_4

If (not $\mathrm{HT}[x, \mathrm{Rng}]$):
  $\mathrm{HT}[x, \mathrm{Rng}] \leftarrow\!\!\$\ \mathrm{Rng}$
  If ($\mathrm{Rng} = \{0,1\}^{\mathsf{cl}}$):
    $\mathrm{HT}[x, \mathrm{Rng}] \leftarrow\!\!\$\ \{0,1\}^{\mathsf{cl}}$
  If ($\mathrm{Rng} = \mathsf{ID.CS}(ivk, \mathsf{cl})$):
    $i_2 \leftarrow i_2 + 1$; $\mathrm{HT}[x, \mathrm{Rng}] \leftarrow Y_{i_2}$; $\mathrm{Ind}_2(x) \leftarrow i_2$
Return $\mathrm{HT}[x, \mathrm{Rng}]$

$\mathrm{H}(x, \mathrm{Rng})$ // G_5

If (not $\mathrm{HT}[x, \mathrm{Rng}]$):
  $\mathrm{HT}[x, \mathrm{Rng}] \leftarrow\!\!\$\ \mathrm{Rng}$
  If ($\mathrm{Rng} = \{0,1\}^{\mathsf{tl}}$):
    $T \leftarrow T \cup \{x\}$
  If ($\mathrm{Rng} = \mathsf{ID.CS}(ivk, \mathsf{cl})$):
    $i_2 \leftarrow i_2 + 1$; $\mathrm{HT}[x, \mathrm{Rng}] \leftarrow Y_{i_2}$; $\mathrm{Ind}_2(x) \leftarrow i_2$
Return $\mathrm{HT}[x, \mathrm{Rng}]$

Figure 7: More games for the proof of Theorem 3.

where the inequality is by the Fundamental Lemma of Game Playing of [5]. The random choice of s made by procedure $\textsc{Sign}$ ensures

$$\Pr[\mathrm{G}_0 \text{ sets } \mathsf{bad}] \leq \sum_{i=0}^{q_s-1} \frac{q_3 + i}{2^{\mathsf{sl}}} = \frac{q_s(2q_3 + q_s - 1)}{2^{\mathsf{sl}+1}} \ . \tag{7}$$

Now we need to bound $\Pr[\mathrm{G}_1]$. We start by considering whether the ciphertext $ITK = itk \oplus \mathrm{H}(isk, \{0,1\}^{\mathsf{tl}})$ helps $\mathcal{A}$ over and above access to $\textsc{Sign}$. Consider the games $\mathrm{G}_2, \mathrm{G}_3$ of Fig. 6. They pick ITK directly at random rather than as prescribed in the scheme. However, via the boxed code that it contains, game G_2 compensates, replying to $\mathrm{H}(\cdot, \{0,1\}^{\mathsf{tl}})$ queries in such a way that $ITK = itk \oplus \mathrm{H}(isk, \{0,1\}^{\mathsf{tl}})$. Thus G_2 is equivalent to G_1. Game G_3 omits the boxed code, but the games are identical-until-bad. So we have

$$\begin{aligned} \Pr[\mathrm{G}_1] = \Pr[\mathrm{G}_2] &= \Pr[\mathrm{G}_3] + \Pr[\mathrm{G}_2] - \Pr[\mathrm{G}_3] \\ &\leq \Pr[\mathrm{G}_3] + \Pr[\mathrm{G}_3 \text{ sets } \mathsf{bad}] \ , \end{aligned} \tag{8}$$

where again the inequality is by the Fundamental Lemma of Game Playing of [5]. Now we have two tasks, namely to bound $\Pr[\mathrm{G}_3]$ and to bound $\Pr[\mathrm{G}_3 \text{ sets } \mathsf{bad}]$. The first corresponds to showing $\mathcal{A}$ cannot forge if ciphertext ITK is random, and the second corresponds to showing that changing the ciphertext to random makes little difference. The first relies on the cimp-uu security of ID, the second on its kr security.

To bound $\Pr[\mathrm{G}_3]$, consider game G_4 of Fig. 7. It moves us towards using cimp-uu by generating

Adversary $\mathcal{P}^{\text{Tr},\text{Ch}}(ivk)$

$ITK \leftarrow\!\!\$\ \{0,1\}^{\mathsf{tl}}$
$vk \leftarrow (ivk, ITK)$
For $i = 1, \ldots, q_2 + q_s + 1$ do
  $(Y_i, c_i, z_i) \leftarrow\!\!\$\ \text{Tr}(\mathsf{cl})$
$i_2 \leftarrow 0$; $j \leftarrow 0$
$(m, \sigma) \leftarrow\!\!\$\ \mathcal{A}^{\text{Sign},\text{H}}(vk)$
$(a, p) \leftarrow m$; $(z, s) \leftarrow \sigma$
$Y \leftarrow \text{H}(a, \mathsf{ID.CS}(ivk, \mathsf{cl}))$
$c \leftarrow \text{H}(a \| p \| s, \{0,1\}^{\mathsf{cl}})$
$k \leftarrow \text{Ind}_3(a \| p \| s)$
Return (k, z)

Adversary $\mathcal{I}^{\text{Tr}}(ivk)$

$ITK \leftarrow\!\!\$\ \{0,1\}^{\mathsf{tl}}$
$vk \leftarrow (ivk, ITK)$
For $i = 1, \ldots, q_2 + q_s + 1$ do
  $(Y_i, c_i, z_i) \leftarrow\!\!\$\ \text{Tr}(\mathsf{cl})$
$i_2 \leftarrow 0$; $T \leftarrow \emptyset$; $j \leftarrow 0$
$(m, \sigma) \leftarrow\!\!\$\ \mathcal{A}^{\text{Sign},\text{H}}(vk)$
For all $x \in T$ do
  If $\mathsf{ID.KVf}(ivk, x)$:
    Return x
Return $\perp$

Sign(m) // $\mathcal{P}, \mathcal{I}$

$(a, p) \leftarrow m$; $s \leftarrow\!\!\$\ \{0,1\}^{\mathsf{sl}}$
$Y \leftarrow \text{H}(a, \mathsf{ID.CS}(ivk, \mathsf{cl}))$
$i \leftarrow \text{Ind}_2(a)$
$\text{HT}[a \| p \| s, \{0,1\}^{\mathsf{cl}}] \leftarrow c_i$
$\sigma \leftarrow (z_i, s)$; Return σ

H(x, Rng) // $\mathcal{P}$

If (not $\text{HT}[x, \text{Rng}]$):
  $\text{HT}[x, \text{Rng}] \leftarrow\!\!\$\ \text{Rng}$
  If $(\text{Rng} = \{0,1\}^{\mathsf{cl}})$:
    $a \| p \| s \leftarrow x$; $Y \leftarrow \text{H}(a, \mathsf{ID.CS}(ivk, \mathsf{cl}))$
    $l \leftarrow \text{Ind}_2(a)$; $j \leftarrow j + 1$; $c \leftarrow\!\!\$\ \text{Ch}(l)$
    $\text{Ind}_3(x) \leftarrow j$; $\text{HT}[x, \text{Rng}] \leftarrow c$
  If $(\text{Rng} = \mathsf{ID.CS}(ivk, \mathsf{cl}))$:
    $i_2 \leftarrow i_2 + 1$; $\text{HT}[x, \text{Rng}] \leftarrow Y_{i_2}$; $\text{Ind}_2(x) \leftarrow i_2$
Return $\text{HT}[x, \text{Rng}]$

H(x, Rng) // $\mathcal{I}$

If (not $\text{HT}[x, \text{Rng}]$):
  If $(\text{Rng} = \{0,1\}^{\mathsf{tl}})$: $T \leftarrow T \cup \{x\}$
  If $(\text{Rng} = \mathsf{ID.CS}(ivk, \mathsf{cl}))$:
    $i_2 \leftarrow i_2 + 1$; $\text{HT}[x, \text{Rng}] \leftarrow Y_{i_2}$; $\text{Ind}_2(x) \leftarrow i_2$
Return $\text{HT}[x, \text{Rng}]$

Figure 8: Adversaries for proof of Theorem 3.

conversation transcripts $Y_i \| c_i \| z_i$ and having Sign use these. We have

$$\Pr[\text{G}_3] = \Pr[\text{G}_4] \ . \tag{9}$$

We build cimp-uu adversary $\mathcal{P}$ so that

$$\Pr[\text{G}_4] \leq \mathbf{Adv}^{\text{cimp-uu}}_{\mathsf{ID}}(\mathcal{P}) \ . \tag{10}$$

The construction of $\mathcal{P}$ is described in detail in Fig. 8. The idea is as follows. Adversary $\mathcal{P}$ uses its transcript oracle Tr to generate the transcripts that G_4 generates directly. It can then simulate $\mathcal{A}$'s Sign oracle as per game G_4. Simulation of $\text{H}(\cdot, \text{Rng})$ is done directly as in the game for $\text{Rng} = \{0,1\}^{\mathsf{tl}}$ and $\text{Rng} = \mathsf{ID.CS}(ivk, \mathsf{cl})$. When a query x is made to $\text{H}(\cdot, \{0,1\}^{\mathsf{cl}})$, adversary $\mathcal{P}$ parses x as $a \| p \| s$, sends the index of the corresponding Tr query to its challenge oracle Ch to get back a challenge, and returns this challenge as the response to the oracle query. Finally when $\mathcal{A}$ produces a forgery, the response in the corresponding signature is output as an impersonation that is successful as long as the forgery was valid.

To bound $\Pr[\text{G}_3 \text{ sets } \mathsf{bad}]$, consider game G_5 of Fig. 7. It answers Sign queries just like G_4, and the only modification in answering H queries is to keep track of queries to $\text{H}(\cdot, \{0,1\}^{\mathsf{tl}})$ in the set T. The game ignores the forgery, returning true if *isk* was queried to $\text{H}(\cdot, \{0,1\}^{\mathsf{tl}})$. We have

$$\Pr[\text{G}_3 \text{ sets } \mathsf{bad}] = \Pr[\text{G}_5] \ . \tag{11}$$

We build $\mathcal{I}$ so that

$$\Pr[\text{G}_5] \leq \mathbf{Adv}^{\text{kr-pa}}_{\mathsf{ID}}(\mathcal{I}) \ . \tag{12}$$

The idea is simple, namely that if the adversary queries *isk* to $\text{H}(\cdot, \{0,1\}^{\mathsf{tl}})$ then we can obtain

$\underline{\mathbf{ID2}[\mathsf{ID},\mathsf{cl}].\mathsf{Kg}^{\mathrm{H},\Pi^{\pm 1}}}$
$(ivk, isk, itk) \leftarrow\!\!\$\ \mathsf{ID.Kg}$
$ITK \leftarrow itk \oplus \mathrm{H}(isk, \{0,1\}^{\mathsf{tl}})$
$vk \leftarrow (ivk, ITK)$; $sk \leftarrow (isk, itk)$
Return (vk, sk)

$\underline{\mathbf{ID2}[\mathsf{ID},\mathsf{cl}].\mathsf{Ex}^{\mathrm{H},\Pi^{\pm 1}}(vk, m_1, m_2, \sigma_1, \sigma_2)}$
$(ivk, ITK) \leftarrow vk$
For $i = 1, 2$ do
  $(a_i, p_i) \leftarrow m_i$ // $a_1 = a_2 \wedge p_1 \neq p_2$
  $(c_{1,i}, z_{2,i}) \leftarrow \sigma_i$; $c_{2,i} \leftarrow \mathrm{H}_2(a_i, p_i)$
  $Y_{2,i} \leftarrow \mathsf{ID.Rsp}^{-1}(ivk, c_{2,i}, z_{2,i})$
  $T_{2,i} \leftarrow Y_{2,i} \| c_{2,i} \| z_{2,i}$
  $z_{1,i} \leftarrow \Pi^{-1}(Y_{2,i})$
  $Y_{1,i} \leftarrow \mathsf{ID.Rsp}^{-1}(ivk, c_{1,i}, z_{1,i})$
  $T_{1,i} \leftarrow Y_{1,i} \| c_{1,i} \| z_{1,i}$
If $Y_{2,1} = Y_{2,2}$:
  If $c_{2,1} = c_{2,2}$: Return $\perp$
  $isk^* \leftarrow\!\!\$\ \mathsf{ID.Ex}(ivk, T_{2,1}, T_{2,2})$
Else: // $Y_{1,1} = Y_{1,2} \wedge c_{1,1} \neq c_{1,2}$
  $isk^* \leftarrow\!\!\$\ \mathsf{ID.Ex}(ivk, T_{1,1}, T_{1,2})$
$itk^* \leftarrow \mathrm{H}(isk^*, \{0,1\}^{\mathsf{tl}}) \oplus ITK$
$sk^* \leftarrow (isk^*, itk^*)$; Return sk^*

$\underline{\mathbf{ID2}[\mathsf{ID},\mathsf{cl}].\mathsf{Sig}^{\mathrm{H},\Pi^{\pm 1}}(vk, sk, m)}$
$(ivk, ITK) \leftarrow vk$; $(isk, itk) \leftarrow sk$
$(a, p) \leftarrow m$
$Y_1 \leftarrow \mathrm{H}_1(a)$; $c_1 \leftarrow\!\!\$\ \{0,1\}$
$y_1 \leftarrow\!\!\$\ \mathsf{ID.Cmt}^{-1}(ivk, itk, Y_1, 1)$
$z_1 \leftarrow \mathsf{ID.Rsp}(ivk, isk, c_1, y_1)$
$Y_2 \leftarrow \Pi^{+1}(z_1)$; $c_2 \leftarrow \mathrm{H}_2(a, p)$
$y_2 \leftarrow\!\!\$\ \mathsf{ID.Cmt}^{-1}(ivk, itk, Y_2, \mathsf{cl})$
$z_2 \leftarrow \mathsf{ID.Rsp}(ivk, isk, c_2, y_2)$
$\sigma \leftarrow (c_1, z_2)$; Return σ

$\underline{\mathbf{ID2}[\mathsf{ID},\mathsf{cl}].\mathsf{Vf}^{\mathrm{H},\Pi^{\pm 1}}(vk, m, \sigma)}$
$(ivk, ITK) \leftarrow vk$; $(a, p) \leftarrow m$
$(c_1, z_2) \leftarrow \sigma$; $c_2 \leftarrow \mathrm{H}_2(a, p)$
$Y_2 \leftarrow \mathsf{ID.Rsp}^{-1}(ivk, c_2, z_2)$
$z_1 \leftarrow \Pi^{-1}(Y_2)$
$Y_1 \leftarrow \mathsf{ID.Rsp}^{-1}(ivk, c_1, z_1)$
Return $(Y_1 = \mathrm{H}_1(a))$

Figure 9: Our construction of a DAPS $\mathbf{ID2}[\mathsf{ID},\mathsf{cl}]$ from a trapdoor identification scheme ID, where $\{1, \mathsf{cl}\} \subseteq \mathsf{ID.clS}$.

---

isk by watching the oracle queries of $\mathcal{A}$. The difficulty is that, to run $\mathcal{A}$, one first has to simulate answers to SIGN queries using transcripts, and it is to enable this that we moved to G_5. Again the game was crafted to make the construction of adversary $\mathcal{I}$ quite direct. The construction is described in detail Fig. 8. The simulation of the SIGN oracle is as before. The simulation of H is more direct, following game G_5 rather than invoking the CH oracle. When $\mathcal{A}$ returns its forgery, the set T contains candidates for the identification secret key *isk*. Adversary $\mathcal{I}$ now verifies each candidate using the key-verification algorithm of the identification scheme, returning a successful candidate if one exists in its list. ▮

5.2 The double-id transform

Our **ID2** transform roughly maintains signing and verifying time compared to **H2** but signatures are shorter, consisting of an ID response plus one bit. Since the verifier can try both possibilities for this bit, if one is willing to double the verification time, even this bit is expendable.

THE CONSTRUCTION. Our construction assumes two main ingredients: The first is a trapdoor identification scheme ID that is commitment recovering, has unique responses, and simultaneously supports challenge lengths 1 and $\mathsf{cl} \gg 1$. For the choice of cl we further assume $|\mathsf{ID.RS}(ivk, 1)| = |\mathsf{ID.CS}(ivk, \mathsf{cl})|$ for all ivk, i.e., the response space for 1-bit challenges has the same cardinality as the commitment space for cl-bit challenges. The second component is a random bijection Π (cf. Section 2) between sets $\mathsf{ID.RS}(ivk, 1)$ and $\mathsf{ID.CS}(ivk, \mathsf{cl})$, i.e., oracle Π^{+1} implements a random mapping from $\mathsf{ID.RS}(ivk, 1)$ to $\mathsf{ID.CS}(ivk, \mathsf{cl})$ and oracle Π^{-1} implements its inverse. In Section 6 we discuss trapdoor ID schemes that fulfill these requirements and show how random bijections

with the required domain and range can be obtained.

The details of the **ID2** transform are specified in Fig. 9. We write $\mathrm{H}_1(\cdot)$ shorthand for $\mathrm{H}(\cdot, \mathsf{ID.CS}(ivk, 1))$, and $\mathrm{H}_2(\cdot,\cdot)$ shorthand for $\mathrm{H}((\cdot,\cdot), \{0,1\}^{\mathsf{cl}})$. As in Section 5.1 we assume these random oracles are independent. Key generation is as in **H2**. Signing works as follows: First a commitment $Y_1 \leftarrow \mathrm{H}_1(a)$ is derived from the address using a random oracle that maps to the commitment space $\mathsf{ID.CS}(ivk, 1)$, then a random 1-bit challenge c_1 is picked and the corresponding response z_1 of the ID scheme computed. Using bijection Π^{+1}, response z_1 is mapped to a commitment $Y_2 \in \mathsf{ID.CS}(ivk, \mathsf{cl})$. A corresponding cl-bit challenge is derived from the address and the payload per $c_2 \leftarrow \mathrm{H}_2(a, p)$. The DAPS signature consists of the response z_2 corresponding to Y_2 and c_2, together with the one-bit challenge c_1. Signatures are verified using the commitment recovery algorithm $\mathsf{ID.Rsp}^{-1}$ to recover Y_2 from z_2, computing $z_1 \leftarrow \Pi^{-1}(Y_2)$, recovering Y_1 from c_1 and z_1 (again using the commitment recovery algorithm), and comparing with $\mathrm{H}_1(a)$. Extraction algorithm $\mathsf{DS.Ex}$ works in the obvious way.

DAP SECURITY. The **ID2** construction achieves double authentication prevention, as the following result confirms. The proof relies on the extractability property of the ID scheme twice: once for each challenge length. We model H as a random oracle as usual. Nothing is assumed of Π other than it being a bijection.

Theorem 4 *Let DAPS* $\mathsf{DS} = \mathbf{ID2}[\mathsf{ID}, \mathsf{cl}]$ *be obtained from trapdoor identification scheme* ID *and challenge length* cl *as above. Let* $\mathcal{A}$ *be an adversary making at most* q *queries to the* $\mathrm{H}_2(\cdot) = \mathrm{H}(\cdot, \{0,1\}^{\mathsf{cl}})$ *oracle. If* ID *has unique responses and perfect extractability, then* $\mathbf{Adv}^{\mathrm{dap}}_{\mathsf{DS}}(\mathcal{A}) \leq q(q-1)/2^{\mathsf{cl}+1}$.

Proof of Theorem 4: Assume, in experiment $\mathbf{G}^{\mathrm{dap}}_{\mathsf{DS}}(\mathcal{A})$, that the adversary outputs message-signature pairs (m_1, σ_1) and (m_2, σ_2) such that for $i \in \{1, 2\}$ we have $\mathsf{DS.Vf}(vk, m_i, \sigma_i) = \mathsf{true}$. The latter implies for $m_i = (a_i, p_i)$ and $\sigma_i = (c_{1,i}, z_{2,i})$ that for recoverable values $z_{1,i}, Y_{2,i}$ and the corresponding transcripts $T_{1,i} = \mathrm{H}_1(a_i) \| c_{1,i} \| z_{1,i}$ and $T_{2,i} = Y_{2,i} \| \mathrm{H}_2(a_i, p_i) \| z_{2,i}$ we have $\mathsf{ID.Vf}(ivk, T_{1,i}) = \mathsf{ID.Vf}(ivk, T_{2,i}) = \mathsf{true}$ and $Y_{2,i} = \Pi^{+1}(z_{1,i})$. Assume $a_1 = a_2$ and $p_1 \neq p_2$. We have either $c_{1,1} \neq c_{1,2}$ or $c_{1,1} = c_{1,2}$. In the former case, the two transcripts $T_{1,1}, T_{1,2}$ have the same commitment but different challenges. This allows us to extract the secret key via the extractability property of ID; further, by decrypting ITK we can recover itk, as required. Consider thus the case $c_{1,1} = c_{1,2}$ which implies $z_{1,1} = z_{1,2}$ and $Y_{2,1} = Y_{2,2}$ by the unique response property of ID. If $\mathrm{H}_2(a_1, p_1) \neq \mathrm{H}_2(a_2, p_2)$ we can extract isk, itk from the two transcripts $T_{2,1}, T_{2,2}$ as above. As $p_1 \neq p_2$ and H is a random oracle, the probability for $\mathrm{H}_2(a_1, p_1) = \mathrm{H}_2(a_2, p_2)$ is $q(q-1)/2^{\mathsf{cl}+1}$. ∎

UNFORGEABILITY. The following establishes that if the ID scheme offers cimp-uc and kr security, then **ID2** transforms it into an unforgeable DAPS. Here we model H as a random oracle and Π as a public random bijection.

Theorem 5 *Let DAPS* $\mathsf{DS} = \mathbf{ID2}[\mathsf{ID}, \mathsf{cl}]$ *be obtained from trapdoor identification scheme* ID *as in Fig. 9. Let* $N = \min |\mathsf{ID.CS}(ivk, \mathsf{cl})|$ *where the minimum is over all* $(ivk, isk, itk) \in [\mathsf{ID.Kg}]$. *Let* $\mathcal{A}$ *be a uf adversary against* DS *and suppose the number of queries that* $\mathcal{A}$ *makes to its* $\mathrm{H}(\cdot, \{0,1\}^{\mathsf{tl}})$, $\mathrm{H}(\cdot, \mathsf{ID.CS}(ivk, 1))$, $\mathrm{H}(\cdot, \{0,1\}^{\mathsf{cl}})$, $\Pi^{\pm 1}$, SIGN *oracles are, respectively,* q_1, q_2, q_3, q_4, q_s. *Then from* $\mathcal{A}$ *we can construct dap adversary* $\mathcal{A}'$, *kr adversary* $\mathcal{I}$ *and* cimp-uc *adversaries* $\mathcal{P}_1, \mathcal{P}_2$ *such that*

$$\mathbf{Adv}^{\mathrm{uf}}_{\mathsf{DS}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{dap}}_{\mathsf{DS}}(\mathcal{A}') + \mathbf{Adv}^{\text{kr-pa}}_{\mathsf{ID}}(\mathcal{I}) + 2\mathbf{Adv}^{\text{cimp-uc}}_{\mathsf{ID}}(\mathcal{P}_1) + 2\mathbf{Adv}^{\text{cimp-uc}}_{\mathsf{ID}}(\mathcal{P}_2) + \frac{(q_4 + q_s)^2}{2N} .$$

Game $\boxed{\mathrm{G}_0}$ / G_1

$(ivk, isk, itk) \leftarrow\!\!\$\ \mathsf{ID.Kg}$
$ITK \leftarrow itk \oplus \mathrm{H}(isk, \{0,1\}^{\mathsf{tl}})$
$vk \leftarrow (ivk, ITK)$
$(m, \sigma) \leftarrow\!\!\$\ \mathcal{A}^{\textsc{Sign},\mathrm{H},\Pi^{\pm 1}}(vk)$
Return $\mathsf{DS.Vf}^{\mathrm{H},\Pi^{\pm 1}}(vk, m, \sigma)$

$\mathrm{H}(x, \mathrm{Rng})$

If $\mathrm{HT}[x, \mathrm{Rng}]$: Return $\mathrm{HT}[x, \mathrm{Rng}]$
$\mathrm{HT}[x, \mathrm{Rng}] \leftarrow\!\!\$\ \mathrm{Rng}$
Return $\mathrm{HT}[x, \mathrm{Rng}]$

$\Pi^{+1}(z_1)$

If $z_1 \in \mathrm{dom}(\mathrm{PT})$: Return $\mathrm{PT}^{+1}(z_1)$
$Y_2 \leftarrow\!\!\$\ \mathsf{ID.CS}(ivk, \mathsf{cl})$
If $Y_2 \in \mathrm{rng}(\mathrm{PT})$: $\mathsf{bad} \leftarrow 1$
$\boxed{Y_2 \leftarrow\!\!\$\ \mathsf{ID.CS}(ivk, \mathsf{cl}) \setminus \mathrm{rng}(\mathrm{PT})}$
$\mathrm{PT} \leftarrow \mathrm{PT} \cup \{(z_1, Y_2)\}$
Return $\mathrm{PT}^{+1}(z_1)$

$\textsc{Sign}(m)$

$(a, p) \leftarrow m$
$Y_1 \leftarrow \mathrm{H}_1(a)\ ;\ c_1 \leftarrow\!\!\$\ \{0,1\}$
$y_1 \leftarrow\!\!\$\ \mathsf{ID.Cmt}^{-1}(ivk, itk, Y_1, 1)$
$z_1 \leftarrow \mathsf{ID.Rsp}(ivk, isk, c_1, y_1)$
$Y_2 \leftarrow \Pi^{+1}(z_1)\ ;\ c_2 \leftarrow \mathrm{H}_2(a, p)$
$y_2 \leftarrow\!\!\$\ \mathsf{ID.Cmt}^{-1}(ivk, itk, Y_2, \mathsf{cl})$
$z_2 \leftarrow \mathsf{ID.Rsp}(ivk, isk, c_2, y_2)$
$\sigma \leftarrow (c_1, z_2)$; Return σ

$\Pi^{-1}(Y_2)$

If $Y_2 \in \mathrm{rng}(\mathrm{PT})$: Return $\mathrm{PT}^{-1}(Y_2)$
$z_1 \leftarrow\!\!\$\ \mathsf{ID.RS}(ivk, 1)$
If $z_1 \in \mathrm{dom}(\mathrm{PT})$: $\mathsf{bad} \leftarrow 1$
$\boxed{z_1 \leftarrow\!\!\$\ \mathsf{ID.RS}(ivk, 1) \setminus \mathrm{dom}(\mathrm{PT})}$
$\mathrm{PT} \leftarrow \mathrm{PT} \cup \{(z_1, Y_2)\}$
Return $\mathrm{PT}^{-1}(Y_2)$

Figure 10: Games $\mathrm{G}_0, \mathrm{G}_1$ for proof of Theorem 5. Game G_0 includes the boxed code and game G_1 does not.

Adversaries $\mathcal{I}, \mathcal{P}_1, \mathcal{P}_2$ make $q_2 + q_3 + q_4 + q_s$ queries to Tr*, and adversaries $\mathcal{P}_1, \mathcal{P}_2$ make one query to* Ch*. Beyond that, the running time of $\mathcal{A}', \mathcal{P}_1, \mathcal{P}_2$ is about that of $\mathcal{A}$, and the running time of $\mathcal{I}$ is that of $\mathcal{A}$ plus the time for q_1 executions of* ID.KVf*.*

Proof of Theorem 5: In the proof, we handle queries to the random bijection Π (with oracles Π^{+1} and Π^{-1}) via lazy sampling and track input-output pairs using a table PT. Notation-wise we consider $\mathrm{PT} \subseteq \mathsf{ID.RS}(ivk, 1) \times \mathsf{ID.CS}(ivk, \mathsf{cl})$ a binary relation to which a mapping of the form $\Pi^{+1}(\alpha) = \beta$ or, equivalently, $\Pi^{-1}(\beta) = \alpha$ can be added by assigning $\mathrm{PT} \leftarrow \mathrm{PT} \cup \{(\alpha, \beta)\}$. We use functional expressions for table look-up, e.g., whenever $(\alpha, \beta) \in \mathrm{PT}$ we write $\mathrm{PT}^{+1}(\alpha) = \beta$ and $\mathrm{PT}^{-1}(\beta) = \alpha$. We annotate the domain of PT with $\mathrm{dom}(\mathrm{PT}) = \{\alpha : (\alpha, \beta) \in \mathrm{PT} \text{ for some } \beta\}$, and its range with $\mathrm{rng}(\mathrm{PT}) = \{\beta : (\alpha, \beta) \in \mathrm{PT} \text{ for some } \alpha\}$.

Without loss of generality we assume from $\mathcal{A}$ the following behavior: (a) if $\mathcal{A}$ outputs a forgery attempt (m, σ) then σ was not returned by Sign on input m; (b) $\mathcal{A}$ does not query Sign twice on the same address; (c) for all messages $m = (a, p)$, $\mathcal{A}$ always queries $\mathrm{H}_1(a)$ before $\mathrm{H}_2(a, p)$; further, $\mathcal{A}$ always queries $\mathrm{H}_2(a, p)$ before querying $\textsc{Sign}(m)$; (d) before outputting a forgery attempt, $\mathcal{A}$ makes all random oracle and random bijection queries required by the verification algorithm to verify the signature. We further may assume that $\mathcal{A}$ does not forge on an address a for which it queried a signature before: Otherwise, by DAP security, the adversary could extract the secret key and forge also on a fresh address; this is accounted for by the $\mathbf{Adv}^{\mathrm{dap}}_{\mathsf{DS}}(\mathcal{A}')$ term in the theorem statement. The correspondingly simplified form of the $\mathbf{G}^{\mathrm{uf}}_{\mathsf{DS}}(\mathcal{A})$ game is given as G_0 in Fig. 10. (Note that queries to Π^{+1} and Π^{-1} are expected to be answered with elements drawn uniformly at random from $\mathsf{ID.CS}(ivk, \mathsf{cl}) \setminus \mathrm{rng}(\mathrm{PT})$ and $\mathsf{ID.RS}(ivk, 1) \setminus \mathrm{dom}(\mathrm{PT})$, respectively, and that our implementation does precisely this, though in an initially surprising form).

Observe that in G_0 the flag bad is set when resampling is required in the processing of Π^{+1} and Π^{-1}. The probability that this happens is at most $(0 + 1 + \ldots + (q_4 + q_s - 1))/N$, where N is the minimum cardinality of the commitment space for challenge length cl, as defined in the theorem

Game G_2

$(ivk, isk, itk) \leftarrow\!\!\$\ \mathsf{ID.Kg}$
$ITK \leftarrow\!\!\$\ \{0,1\}^{\mathsf{tl}}$; $vk \leftarrow (ivk, ITK)$
$(m, \sigma) \leftarrow\!\!\$\ \mathcal{A}^{\textsc{Sign},\mathrm{H},\Pi^{\pm 1}}(vk)$
Return $\mathsf{DS.Vf}^{\mathrm{H},\Pi^{\pm 1}}(vk, m, \sigma)$

$\mathrm{H}(x, \mathrm{Rng})$

If $\mathrm{HT}[x, \mathrm{Rng}]$: Return $\mathrm{HT}[x, \mathrm{Rng}]$
$\mathrm{HT}[x, \mathrm{Rng}] \leftarrow\!\!\$\ \mathrm{Rng}$
If $\mathrm{Rng} = \{0,1\}^{\mathsf{tl}}$:
  If $\mathsf{ID.KVf}(ivk, x)$: $\mathsf{bad}_1 \leftarrow 1$
  If $x = isk$: $\mathrm{HT}[x, \mathrm{Rng}] \leftarrow ITK \oplus itk$
If $\mathrm{Rng} = \mathsf{ID.CS}(ivk, 1)$:
  $Y_1[x] \| c_1[x] \| z_1[x] \leftarrow\!\!\$\ \textsc{Transc}(1)$
  $\mathrm{HT}[x, \mathrm{Rng}] \leftarrow Y_1[x]$
If $\mathrm{Rng} = \{0,1\}^{\mathsf{cl}}$:
  $Y_2[x] \| c_2[x] \| z_2[x] \leftarrow\!\!\$\ \textsc{Transc}(\mathsf{cl})$
  $\mathrm{HT}[x, \mathrm{Rng}] \leftarrow c_2[x]$
Return $\mathrm{HT}[x, \mathrm{Rng}]$

Algorithm $\textsc{Transc}(cl)$

  $(Y, y) \leftarrow\!\!\$\ \mathsf{ID.Cmt}(ivk, cl)$
  $c \leftarrow\!\!\$\ \{0,1\}^{cl}$
  $z \leftarrow \mathsf{ID.Rsp}(ivk, isk, c, y)$
  Return $Y \| c \| z$

$\textsc{Sign}(m)$

$(a, p) \leftarrow m$
If $\exists z \in \mathrm{dom}(\mathrm{PT})$ s.t.
  $\mathsf{ID.Vf}(ivk, Y_1[a] \| 0 \| z)$ or
  $\mathsf{ID.Vf}(ivk, Y_1[a] \| 1 \| z)$: $\mathsf{bad}_2 \leftarrow 1$
If $z_1[a] \in \mathrm{dom}(\mathrm{PT})$:
  $Y_2 \leftarrow \mathrm{PT}^{+1}(z_1[a])$; $c_2 \leftarrow \mathrm{H}_2(a, p)$
  $y_2 \leftarrow\!\!\$\ \mathsf{ID.Cmt}^{-1}(ivk, itk, Y_2, \mathsf{cl})$
  $z_2 \leftarrow \mathsf{ID.Rsp}(ivk, isk, c_2, y_2)$
Else:
  $Y_2 \leftarrow Y_2[a, p]$; $z_2 \leftarrow z_2[a, p]$
  $\mathrm{PT} \leftarrow \mathrm{PT} \cup \{(z_1[a], Y_2)\}$
$\sigma \leftarrow (c_1[a], z_2)$; Return σ

$\Pi^{+1}(z_1)$

If $z_1 \in \mathrm{dom}(\mathrm{PT})$: Return $\mathrm{PT}^{+1}(z_1)$
$Y_2[z_1] \| c_2[z_1] \| z_2[z_1] \leftarrow\!\!\$\ \textsc{Transc}(\mathsf{cl})$
$\mathrm{PT} \leftarrow \mathrm{PT} \cup \{(z_1, Y_2[z_1])\}$
Return $\mathrm{PT}^{+1}(z_1)$

$\Pi^{-1}(Y_2)$

If $Y_2 \in \mathrm{rng}(\mathrm{PT})$: Return $\mathrm{PT}^{-1}(Y_2)$
$z_1 \leftarrow\!\!\$\ \mathsf{ID.RS}(ivk, 1)$
$\mathrm{PT} \leftarrow \mathrm{PT} \cup \{(z_1, Y_2)\}$
Return $\mathrm{PT}^{-1}(Y_2)$

Figure 11: Game G_2 for proof of Theorem 5.

statement. We define game G_1 like G_0 but with the resampling steps in the Π^{+1} and Π^{-1} oracles removed. We obtain

$$\Pr[\mathrm{G}_0] = \Pr[\mathrm{G}_1] + \frac{(q_4 + q_s)^2 - (q_4 + q_s)}{2N} \quad . \tag{13}$$

Consider next game G_2 from Fig. 11. It is obtained from G_1 by applying the following rewriting steps. First, instead of computing ITK by evaluating $itk \oplus \mathrm{H}(isk, \{0,1\}^{\mathsf{tl}})$ it picks ITK at random and programs random oracle H such that relation $ITK = itk \oplus \mathrm{H}(isk, \{0,1\}^{\mathsf{tl}})$ is maintained. Second, the way random oracle queries of the form $\mathrm{H}(x, \mathsf{ID.CS}(ivk, 1))$ and $\mathrm{H}(x, \{0,1\}^{\mathsf{cl}})$ are processed is changed: Now, the internal $\textsc{Transc}$ algorithm is invoked to produce full identification transcripts for the corresponding challenge length; the H oracle outputs one component of these transcripts and keeps the other components for itself. Also the implementation of Π^{+1} is modified to use the $\textsc{Transc}$ algorithm.

Concerning the $\textsc{Sign}$ oracle, observe that G_1 samples challenge c_1 and derives corresponding y_1 and z_1 values by itself. In G_2, as we assume that $\mathrm{H}_1(a)$ is always queried before $\textsc{Sign}(a, p)$, and as the $\mathrm{H}_1(a)$ implementation now internally prepares a full transcript, the c_1, y_1, z_1 values from this transcript generation can be used within the $\textsc{Sign}$ oracle. That is, we replace the first invocations of $\mathsf{ID.Cmt}^{-1}$ and $\mathsf{ID.Rsp}$ in $\textsc{Sign}$ of G_1 by the assignments $Y_1 \leftarrow Y_1[a]$, $y_1 \leftarrow y_1[a]$, $c_1 \leftarrow c_1[a]$, and $z_1 \leftarrow z_1[a]$ in G_2. (Note that this works only because we also assume that $\textsc{Sign}$ is not queried more than once on the same address.) Consider next the assignment $Y_2 \leftarrow \Pi^{+1}(z_1)$ of $\textsc{Sign}$ in G_1 (which now would be annotated $Y_2 \leftarrow \Pi^{+1}(z_1[a])$) and the fact that Y_2 is completed by $\textsc{Sign}$ to

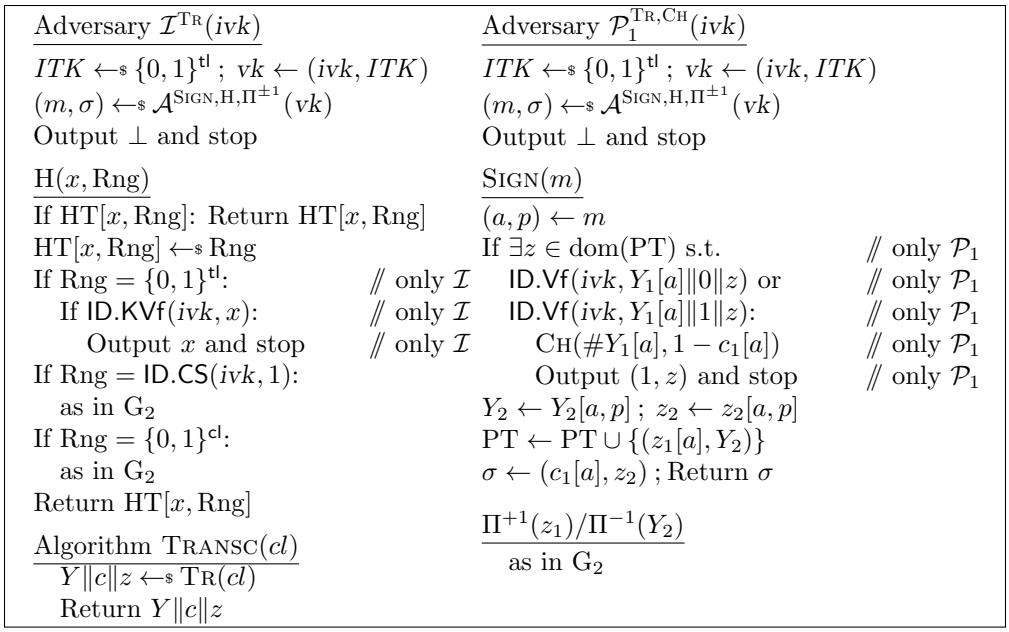


Figure 12: Adversaries for proof of Theorem 5. The oracles and the Transc implementation are shared by both adversaries. In Sign, we write $\#Y_1[a]$ for the number of the Tr query in which the value of $Y_1[a]$ was established.

---

a transcript with challenge $c_2[a,p]$. In the evaluation of $\Pi^{+1}(z_1)$, two cases can be distinguished: either the query is 'old', i.e., $z_1 \in \text{dom(PT)}$, in which case Sign proceeds its computations using the stored commitment $Y_2 = \text{PT}^{+1}(z_1)$, or the query is 'fresh', i.e., $z_1 \notin \text{dom(PT)}$, in which case a new value Y_2 is sampled from $\mathsf{ID.CS}(ivk, \mathsf{cl})$. In both cases Sign completes Y_2 to a full transcript with challenge $\text{H}_2(a,p) = c_2[a,p]$. As we assume that each Sign(a,p) query is preceded by a $\text{H}_2(a,p)$ query, and the latter internally generates a full transcript with challenge $c_2[a,p]$, similarly to what we did for the values Y_1, y_1, c_1, z_1 above, in the case of a 'fresh' $\Pi^{+1}(z_1)$ query game G_2 sets $Y_2 \leftarrow Y_2[a,p]$, $y_2 \leftarrow y_2[a,p]$, $c_2 \leftarrow c_2[a,p]$, and $z_2 \leftarrow z_2[a,p]$. The two described cases correspond with the two branches of the second If-statement in Sign of Fig. 11.

The remaining changes between G_1 and G_2 concern the two added flags bad_1 and bad_2 and can be ignored for now. Thus all changes between games G_1 and G_2 are pure rewriting, so we obtain

$$\Pr[G_1] = \Pr[G_2] \ . \tag{14}$$

Consider next in more detail the flags bad_1 and bad_2 that appear in game G_2. The former is set whenever a value is queried to $\text{H}(\cdot, \{0,1\}^{\mathsf{tl}})$ that is a valid secret identification key for verification key ivk, and the latter is set when Sign is queried on some address a and the domain of PT contains an element that is a valid response for commitment $Y_1[a]$ and one of the two possible challenges $c_1 \in \{0,1\}$. Observe that any use of itk in H is preceded by setting $\mathsf{bad}_1 \leftarrow 1$, and that any execution of the first branch of the second If-statement of Sign in G_2 is preceded by setting $\mathsf{bad}_2 \leftarrow 1$.

We would like to proceed the proof by bounding the probabilities $\Pr[G_2 \text{ sets } \mathsf{bad}_1]$ and $\Pr[G_2 \text{ sets } \mathsf{bad}_2]$ (based on the hardness of key recovery and cimp-uc impersonation, respectively). However, the following technical problem arises: While in the two corresponding reductions we would be able

Game G_3

$(ivk, isk, itk) \leftarrow\!\!\$\ \mathsf{ID.Kg}$
$ITK \leftarrow\!\!\$\ \{0,1\}^{\mathsf{tl}}$; $vk \leftarrow (ivk, ITK)$
$(m, \sigma) \leftarrow\!\!\$\ \mathcal{A}^{\textsc{Sign},\mathrm{H},\Pi^{\pm 1}}(vk)$
$(a, p) \leftarrow m$; $(c_1, z_2) \leftarrow \sigma$
$Y_2 \leftarrow \mathsf{ID.Rsp}^{-1}(ivk, c_2[a,p], z_2)$
$z_1 \leftarrow \Pi^{-1}(Y_2)$
$Y_1 \leftarrow \mathsf{ID.Rsp}^{-1}(ivk, c_1, z_1)$
If $Y_1 \neq Y_1[a]$: Return false
If $c_1 \neq c_1[a]$: $\mathsf{bad} \leftarrow 1$
Return true

$\textsc{Sign}(m)$

$(a, p) \leftarrow m$
If $\exists z \in \mathrm{dom}(\mathrm{PT})$ s.t.
  $\mathsf{ID.Vf}(ivk, Y_1[a] \| 0 \| z)$ or
  $\mathsf{ID.Vf}(ivk, Y_1[a] \| 1 \| z)$: $\mathsf{bad}_2 \leftarrow 1$
$\mathrm{PT} \leftarrow \mathrm{PT} \cup \{(z_1[a], Y_2[a,p])\}$
$\sigma \leftarrow (c_1[a], z_2[a,p])$; Return σ

$\mathrm{H}(x, \mathrm{Rng})$

If $\mathrm{HT}[x, \mathrm{Rng}]$: Return $\mathrm{HT}[x, \mathrm{Rng}]$
$\mathrm{HT}[x, \mathrm{Rng}] \leftarrow\!\!\$\ \mathrm{Rng}$
If $\mathrm{Rng} = \{0,1\}^{\mathsf{tl}}$:
  If $\mathsf{ID.KVf}(ivk, x)$: $\mathsf{bad}_1 \leftarrow 1$
If $\mathrm{Rng} = \mathsf{ID.CS}(ivk, 1)$:
  $Y_1[x] \| c_1[x] \| z_1[x] \leftarrow\!\!\$\ \textsc{Transc}(1)$
  $\mathrm{HT}[x, \mathrm{Rng}] \leftarrow Y_1[x]$
If $\mathrm{Rng} = \{0,1\}^{\mathsf{cl}}$:
  $Y_2[x] \| c_2[x] \| z_2[x] \leftarrow\!\!\$\ \textsc{Transc}(\mathsf{cl})$
  $\mathrm{HT}[x, \mathrm{Rng}] \leftarrow c_2[x]$
Return $\mathrm{HT}[x, \mathrm{Rng}]$

$\Pi^{+1}(z_1)/\Pi^{-1}(Y_2)$

  as in G_2

Algorithm $\textsc{Transc}(cl)$

  as in G_2

Adversary $\mathcal{P}_2^{\textsc{Tr},\textsc{Ch}}(ivk)$

$ITK \leftarrow\!\!\$\ \{0,1\}^{\mathsf{tl}}$; $vk \leftarrow (ivk, ITK)$
$(m, \sigma) \leftarrow\!\!\$\ \mathcal{A}^{\textsc{Sign},\mathrm{H},\Pi^{\pm 1}}(vk)$
$(a, p) \leftarrow m$; $(c_1, z_2) \leftarrow \sigma$
$Y_2 \leftarrow \mathsf{ID.Rsp}^{-1}(ivk, c_2[a,p], z_2)$
$z_1 \leftarrow \Pi^{-1}(Y_2)$
$Y_1 \leftarrow \mathsf{ID.Rsp}^{-1}(ivk, c_1, z_1)$
If $Y_1 \neq Y_1[a]$: Return $\perp$
If $c_1 \neq c_1[a]$:
  $\textsc{Ch}(\#Y_1[a], c_1)$
  Output $(1, z_1)$ and stop
Return $\perp$

Algorithm $\textsc{Transc}(cl)$

  $Y \| c \| z \leftarrow\!\!\$\ \textsc{Tr}(cl)$
  Return $Y \| c \| z$

$\textsc{Sign}(m)$

$(a, p) \leftarrow m$
$\mathrm{PT} \leftarrow \mathrm{PT} \cup \{(z_1[a], Y_2[a,p])\}$
$\sigma \leftarrow (c_1[a], z_2[a,p])$; Return σ

$\mathrm{H}(x, \mathrm{Rng})$

If $\mathrm{HT}[x, \mathrm{Rng}]$: Return $\mathrm{HT}[x, \mathrm{Rng}]$
$\mathrm{HT}[x, \mathrm{Rng}] \leftarrow\!\!\$\ \mathrm{Rng}$
If $\mathrm{Rng} = \mathsf{ID.CS}(ivk, 1)$:
  as in G_3
If $\mathrm{Rng} = \{0,1\}^{\mathsf{cl}}$:
  as in G_3
Return $\mathrm{HT}[x, \mathrm{Rng}]$

$\Pi^{+1}(z_1)/\Pi^{-1}(Y_2)$

  as in G_3

Figure 13: **Top:** Game G_3 for proof of Theorem 5. **Bottom:** One more adversary for proof of Theorem 5. We write $\#Y_1[a]$ for the number of the Tr query in which the value of $Y_1[a]$ was established.

---

to simulate the Transc algorithm with the Tr oracle, when aiming at bounding the probability of $\mathsf{bad}_1 \leftarrow 1$ it would be unclear how to simulate the Sign oracle (that uses *isk* and *itk* in the first If-branch), and when aiming at bounding the probability of $\mathsf{bad}_2 \leftarrow 1$ it would be unclear how to simulate the H oracle (that uses *itk* in the $\mathrm{Rng} = \{0,1\}^{\mathsf{tl}}$ branch). We help ourselves by defining the following three complementary events: (a) neither bad_1 nor bad_2 is set, (b) bad_1 is set before bad_2 (this includes the case that bad_2 is not set at all), and (c) bad_2 is set before bad_1 (this includes the case that bad_1 is not set at all). In Fig. 12 we construct a kr adversary $\mathcal{I}$ and a cimp-uc adversary $\mathcal{P}_1$ from $\mathcal{A}$ such that

$$\Pr[\mathrm{G}_2 \text{ sets } \mathsf{bad}_1 \text{ first}] = \mathbf{Adv}^{\text{kr-pa}}_{\mathsf{ID}}(\mathcal{I}) \tag{15}$$

$\underline{\mathsf{GQ.Kg}}$	**Prover**		**Verifier**
$(N,p,q,e,d) \leftarrow_{\$} \mathsf{RSA}$	Input: $(N,e,X),x,\mathsf{cl}$		Input: $(N,e,X),\mathsf{cl}$
$x \leftarrow_{\$} \mathbb{Z}_N^*$	$y \leftarrow_{\$} \mathbb{Z}_N^*$		
$X \leftarrow x^e \mod N$	$Y \leftarrow y^e \mod N$	$\xrightarrow{Y}$	$c \leftarrow_{\$} \{0,1\}^{\mathsf{cl}}$
Return $((N,e,X),x,d)$		$\xleftarrow{c}$	
	$z \leftarrow yx^c \mod N$	$\xrightarrow{z}$	$v \leftarrow (z^e \equiv YX^c \pmod N)$

$\underline{\mathsf{GQ.Ex}((N,e,X),Y_1,c_1,z_1,Y_2,c_2,z_2)}$	$\underline{\mathsf{GQ.KVf}((N,e,X),x)}$
If $z_1^e \not\equiv Y_1X^{c_1} \vee z_2^e \not\equiv Y_2X^{c_2}$: Return $\perp$	Return $(x^e \bmod N = X)$
If $Y_1 \neq Y_2 \vee \lvert c_1\rvert \neq \lvert c_2\rvert \vee c_1 = c_2$: Return $\perp$	
$z \leftarrow z_1 z_2^{-1} \mod N$	$\underline{\mathsf{GQ.Cmt}^{-1}((N,e,X),d,Y,\mathsf{cl})}$
$c \leftarrow c_1 - c_2$ // w.l.o.g. $c > 0$	$y \leftarrow Y^d \mod N$
$(a,b) \leftarrow \mathrm{egcd}(e,c)$	Return y
$x \leftarrow X^a z^b \mod N$	
Return x	

Figure 14: Trapdoor identification scheme GQ associated to RSA generator RSA.

---

and

$$\Pr[\mathrm{G}_2 \text{ sets } \mathsf{bad}_2 \text{ first}] = 2\mathbf{Adv}^{\text{cimp-uc}}_{\mathsf{ID}}(\mathcal{P}_1) \ . \tag{16}$$

The strategy for constructing the adversaries is clear: We derive $\mathcal{I}$ from G_2 by stripping off all code that is only executed after bad_2 is set, and we construct $\mathcal{P}_1$ by removing all code only executed after bad_1 is set. The $\mathcal{P}_1$-related code in SIGN deserves further explanation. The reduction obtained commitment $Y_1[a]$ via H from the TR oracle of the cimp-uc game, together with challenge $c_1[a]$ and response $z_1[a]$. As at the time the bad_2 flag is set in G_2 no information on $c_1[a]$ was used in the game or exposed to the adversary, for the challenge c^* for which $\mathsf{ID.Vf}(ivk, Y_1[a]\|c^*\|z) = \mathsf{true}$ we have that $c^* \neq c_1[a]$ with probability $1/2$. The reduction thus tries to break cimp-uc security with challenge $1 - c_1[a]$ and response z. Whenever this challenge is admissible (i.e., with probability $1/2$), the response is correct. That is, $\mathcal{P}_1$ is successful with breaking impersonation with half the probability of $\mathcal{A}$ having flag bad_2 be set first.

In Fig. 13 we define game G_3 which behaves exactly like G_2 until either bad_1 or bad_2 is set. Thus we have

$$\Pr[\mathrm{G}_2 \text{ sets neither } \mathsf{bad}_1 \text{ nor } \mathsf{bad}_2] = \Pr[\mathrm{G}_3] \ . \tag{17}$$

In G_3 we expand the $\mathsf{DS.Vf}$ algorithm, i.e., the steps where the forgery attempt of $\mathcal{A}$ is verified. If signature $\sigma = (c_1, z_2)$ is identified as valid, the game sets flag bad to 1 if $c_1 \neq c_1[a]$, i.e., if the challenge c_1 included in the signature does not coincide with the one simulated in the H oracle for address a. Using the assumption that $\mathcal{A}$ does not forge on addresses a for which it posed a $\textsc{Sign}(a,\cdot)$ query, observe that the game did not release any information on $c_1[a]$, so by an information theoretic argument, $c_1 \neq c_1[a]$ and thus $\mathsf{bad} \leftarrow 1$ with probability $1/2$.

In Fig. 13 we construct a cimp-uc adversary $\mathcal{P}_2$ from $\mathcal{A}$ that is successful whenever bad is set in game G_3. We obtain

$$\Pr[\mathrm{G}_3] = 2\mathbf{Adv}^{\text{cimp-uc}}_{\mathsf{ID}}(\mathcal{P}_2) \ . \tag{18}$$

Taken together, the established bounds imply the theorem statement. ∎

6 Instantiation and implementation

We illustrate how to instantiate our **H2** and **ID2** transforms, using the GQ and MR identification schemes as examples, to obtain **H2**[GQ], **ID2**[GQ], and **H2**[MR]. We implement these to get performance data. Similar instantiations and implementations are possible with many other trapdoor identification schemes.

6.1 GQ-based schemes

GQ. An RSA generator for modulus length k is an algorithm RSA that returns a tuple (N, p, q, e, d) where p, q are distinct odd primes, $N = pq$ is the modulus in the range $2^{k-1} < N < 2^k$, encryption and decryption exponents e, d are in $\mathbb{Z}^*_{\varphi(N)}$, and $ed \equiv 1 \pmod{\varphi(N)}$. The assumption is one-wayness, formalized by defining the ow-advantage of an adversary $\mathcal{A}$ against RSA by $\mathbf{Adv}^{\mathrm{ow}}_{\mathsf{RSA}}(\mathcal{A}) = \Pr[\mathrm{OW}^{\mathcal{A}}_{\mathsf{RSA}}]$ where the game is in Fig. 15. Let L be a parameter and RSA be such that $\gcd(e, c) = 1$ for all $0 < c < 2^L$. (For instance, RSA may select encryption exponent e as an $L + 1$ bit prime number.) If egcd denotes the extended gcd algorithm that given relatively-prime inputs e, c returns a, b such that $ea + cb = 1$, the GQ identification scheme associated to RSA is shown in Fig. 14. Any challenge length up to L is admissible, i.e., $\mathsf{ID.clS} \subseteq \{1, \ldots, L\}$, and for all $\mathsf{cl} \in \mathsf{ID.clS}$ the commitment and response space is $\mathsf{ID.CS}(ivk, \mathsf{cl}) = \mathsf{ID.RS}(ivk, \mathsf{cl}) = \mathbb{Z}^*_N$. Extraction works because of identity $X^a z^b = x^{ea} x^{(c_1 - c_2)b} = x$. Algorithm $\mathsf{GQ.Cmt}^{-1}$ shows that the scheme is trapdoor; that it also is commitment recovering and has unique responses follows from inspection of the $z^e = YX^c$ condition of the verification algorithm. Finally, it is a standard result, and in particular follows from Lemma 1 (p. 10), that KR, CIMP-UU, CIMP-UC security of GQ tightly reduce to the one-wayness of RSA (note the CIMP-UU case requires a restriction on the deployed challenge lengths).

Game $\mathrm{OW}^{\mathcal{A}}_{\mathsf{RSA}}$
$(N, p, q, e, d) \leftarrow\!\!\$\ \mathsf{RSA}$
$x \leftarrow\!\!\$\ \mathbb{Z}^*_N\ ;\ X \leftarrow x^e \bmod N$
$x' \leftarrow\!\!\$\ \mathcal{A}(N, e, X)$
Return $(x' = x)$

Figure 15: Game defining one-wayness of RSA generator RSA.

H2[GQ]. Fig. 16 shows the algorithms of the **H2**[GQ] DAPS scheme derived by applying our **H2** transform to the GQ identification scheme of Fig. 14. To estimate security for a given modulus length k we use Theorem 2, Theorem 3, and the reductions between CIMP-UU and KR security of GQ and the one-wayness of RSA from Lemma 1. The reductions are tight and so we need to estimate the advantage of an adversary against the one-wayness of RSA. We do this under the assumption that the NFS is the best factoring method. Thus, our implementation uses a 2048-bit modulus and 256-bit hashes and seeds. See below and Fig. 21 for implementation and performance information.

ID2[GQ]. Fig. 16 also shows the algorithms of the DAPS scheme derived by applying the **ID2** transform to GQ. Reductions continue to be tight so instantiation and implementation choices are as for **H2**[GQ]. Concerning the random permutation Π on $\mathbb{Z}^*_N$ that the scheme requires, it effectively suffices to construct one that maps $\mathbb{Z}_N$ to $\mathbb{Z}_N$, and we propose one way to instantiate it in the following.

A random permutation Π on $\mathbb{Z}_N$ can be constructed from a random permutation Γ on $\{0,1\}^k$, where $2^{k-1} < N < 2^k$, by cycle walking [6, 19]: if x is the input, let $c \leftarrow \Gamma(x)$; if $c \in \mathbb{Z}_N$, return c; else recurse on c; the inverse is analogous. A Feistel network can be used to construct a random permutation Γ on $\{0,1\}^{2n}$ from a set of public random functions $F_1, \ldots, F_r$ on $\{0,1\}^n$. In other words, for input $x_0 \| x_1 \in \{0,1\}^{2n}$, return $x_r \| x_{r+1}$ where $x_{i+1} = x_{i-1} \oplus F_i(x_i)$. Dai and Steinberger [9] give an indifferentiability result for 8 rounds, under the assumption that the F_i are independent public

$\underline{\mathbf{H2}[\mathsf{GQ}].\mathsf{Kg}^{\mathrm{H}}}$

$((N,e,X),x,d) \leftarrow\!\!\$\ \mathsf{GQ.Kg}\ ;\ ITK \leftarrow d \oplus \mathrm{H}(x,\{0,1\}^k)$
Return $((N,e,X,ITK),(x,d))$

$\underline{\mathbf{H2}[\mathsf{GQ}].\mathsf{Sig}^{\mathrm{H}}((N,e,X,ITK),(x,d),m)}$

$(a,p) \leftarrow m\ ;\ s \leftarrow\!\!\$\ \{0,1\}^{\mathsf{sl}}\ ;\ Y \leftarrow \mathrm{H}(a,\mathbb{Z}_N^*)\ ;\ y \leftarrow Y^d \bmod N$
$c \leftarrow \mathrm{H}(a\|p\|s,\{0,1\}^{\mathsf{cl}})\ ;\ z \leftarrow yx^c \bmod N\ ;\ \sigma \leftarrow (z,s)\ ;$ Return σ

$\underline{\mathbf{H2}[\mathsf{GQ}].\mathsf{Vf}^{\mathrm{H}}((N,e,X,ITK),m,\sigma)}$

$(a,p) \leftarrow m\ ;\ (z,s) \leftarrow \sigma\ ;\ Y \leftarrow \mathrm{H}(a,\mathbb{Z}_N^*)\ ;\ c \leftarrow \mathrm{H}(a\|p\|s,\{0,1\}^{\mathsf{cl}})$
Return $(z^e \equiv YX^c \pmod N)$

$\underline{\mathbf{H2}[\mathsf{GQ}].\mathsf{Ex}^{\mathrm{H}}((N,e,X,ITK),m_1,m_2,\sigma_1,\sigma_2)}$

For $i = 1,2$ do
  $(a_i,p_i) \leftarrow m_i\ ;\ (z_i,s_i) \leftarrow \sigma_i$
  $Y_i \leftarrow \mathrm{H}(a_i,\mathbb{Z}_N^*)\ ;\ c_i \leftarrow \mathrm{H}(a_i\|p_i\|s_i,\{0,1\}^{\mathsf{cl}})$
$x \leftarrow \mathsf{GQ.Ex}((N,e,X),Y_1,c_1,z_1,Y_2,c_2,z_2)$
$d \leftarrow \mathrm{H}(x,\{0,1\}^k) \oplus ITK$; Return (x,d)

$\underline{\mathbf{ID2}[\mathsf{GQ}].\mathsf{Kg}^{\mathrm{H},\Pi^{\pm 1}}}$

$((N,e,X),x,d) \leftarrow\!\!\$\ \mathsf{GQ.Kg}\ ;\ ITK \leftarrow d \oplus \mathrm{H}(x,\{0,1\}^k)$
Return $((N,e,X,ITK),(x,d))$

$\underline{\mathbf{ID2}[\mathsf{GQ}].\mathsf{Sig}^{\mathrm{H},\Pi^{\pm 1}}((N,e,X,ITK),(x,d),m)}$

$(a,p) \leftarrow m\ ;\ Y_1 \leftarrow \mathrm{H}(a,\mathbb{Z}_N^*)\ ;\ c_1 \leftarrow\!\!\$\ \{0,1\}\ ;\ y_1 \leftarrow Y_1^d \bmod N$
$z_1 \leftarrow y_1 x^{c_1} \bmod N\ ;\ Y_2 \leftarrow \Pi^{+1}(z_1)\ ;\ y_2 \leftarrow\!\!\$\ Y_2^d \bmod N$
$c_2 \leftarrow \mathrm{H}(a\|p,\{0,1\}^{\mathsf{cl}})\ ;\ z_2 \leftarrow y_2 x^{c_2} \bmod N$
$\sigma \leftarrow (c_1,z_2)$; Return σ

$\underline{\mathbf{ID2}[\mathsf{GQ}].\mathsf{Vf}^{\mathrm{H},\Pi^{\pm 1}}((N,e,X,ITK),m,\sigma)}$

$(a,p) \leftarrow m\ ;\ (c_1,z_2) \leftarrow \sigma\ ;\ c_2 \leftarrow \mathrm{H}(a\|p,\{0,1\}^{\mathsf{cl}})$
$Y_2 \leftarrow (z_2)^e X^{-c_2}\ ;\ z_1 \leftarrow \Pi^{-1}(Y_2)\ ;\ Y_1 \leftarrow (z_1)^e X^{-c_1}$
Return $(Y_1 = \mathrm{H}(a,\mathbb{Z}_N^*))$

$\underline{\mathbf{ID2}[\mathsf{GQ}].\mathsf{Ex}^{\mathrm{H},\Pi^{\pm 1}}((N,e,X,ITK),m_1,m_2,\sigma_1,\sigma_2)}$

For $i = 1,2$ do
  $(a_i,p_i) \leftarrow m_i\ ;\ (c_{1,i},z_{2,i}) \leftarrow \sigma_i\ ;\ c_{2,i} \leftarrow \mathrm{H}(a_i\|p_i,\{0,1\}^{\mathsf{cl}})$
  $Y_{2,i} \leftarrow (z_{2,i})^e X^{-c_{2,i}}\ ;\ z_{1,i} \leftarrow \Pi^{-1}(Y_{2,i})$
  $Y_{1,i} \leftarrow (z_{1,i})^e X^{-c_{1,i}}$
If $Y_{2,1} = Y_{2,2}$: $x \leftarrow \mathsf{GQ.Ex}((N,e,X),Y_{2,1},c_{2,1},z_{2,1},Y_{2,2},c_{2,2},z_{2,2})$
Else: $x \leftarrow \mathsf{GQ.Ex}((N,e,X),Y_{1,1},c_{1,1},z_{1,1},Y_{1,2},c_{1,2},z_{1,2})$
$d \leftarrow \mathrm{H}(x,\{0,1\}^k) \oplus ITK$; Return (x,d)

Figure 16: DAPS schemes $\mathbf{H2}[\mathsf{GQ},\mathsf{cl},\mathsf{sl}]$ and $\mathbf{ID2}[\mathsf{GQ},\mathsf{cl}]$ derived via our transforms from ID scheme GQ.

$\underline{\mathsf{CF.Kg}}$	**Prover**		**Verifier**
$(ek, ik) \leftarrow\!\!\$\ \mathsf{FG.Kg}$	Input: ek, ik, cl		Input: ek, cl
Return (ek, ik, ε)	$Y \leftarrow\!\!\$\ \mathsf{FG.D}(ek)$	$\xrightarrow{Y}$	$c \leftarrow\!\!\$\ \{0,1\}^{\mathsf{cl}}$
		$\xleftarrow{c}$	
	$z \leftarrow \mathsf{FG.Ev}^{-1}_{ik,c}(Y)$	$\xrightarrow{z}$	$v \leftarrow (\mathsf{FG.Ev}_{ek,c}(z) = Y)$

$\underline{\mathsf{CF.Ex}(ek, Y_1, c_1, z_1, Y_2, c_2, z_2)}$	$\underline{\mathsf{CF.KVf}(ek, x)}$
If $\mathsf{FG.Ev}_{ek,c_1}(z_1) \neq Y_1 \vee \mathsf{FG.Ev}_{ek,c_2}(z_2) \neq Y_2$: Return $\perp$	Return $\mathsf{FG.KVf}(ek, x)$
If $Y_1 \neq Y_2 \vee \lvert c_1\rvert \neq \lvert c_2\rvert \vee c_1 = c_2$: Return $\perp$	
$z_1 \leftarrow \mathsf{FG.Ev}_{ek,0}(z_1)\ ;\ z_2 \leftarrow \mathsf{FG.Ev}_{ek,0}(z_2)$	$\underline{\mathsf{CF.Cmt}^{-1}(ek, \varepsilon, Y, \mathsf{cl})}$
For $i = 1, \ldots, \lvert c_1\rvert$ do	Return ε
$\quad z_1' \leftarrow \mathsf{FG.Ev}_{ek,c_1[i]}(z_1)\ ;\ z_2' \leftarrow \mathsf{FG.Ev}_{ek,c_2[i]}(z_2)$	
$\quad$ If $(c_1[i] \neq c_2[i]) \wedge (z_1' = z_2')$:	
$\quad\quad$ If $(c_1[i], c_2[i]) = (0,1)$: Return $\mathsf{FG.Ex}(ek, z_1, z_2)$	
$\quad\quad$ If $(c_1[i], c_2[i]) = (1,0)$: Return $\mathsf{FG.Ex}(ek, z_2, z_1)$	
$\quad z_1 \leftarrow z_1'\ ;\ z_2 \leftarrow z_2'$	
Return $\perp$	

Figure 17: Identification scheme CF associated to claw-free function generator FG.

random functions. We construct F_i on $\{0,1\}^n$ as $F_i(x) = H(i\|1\|x)\|\ldots\|H(i\|\ell\|x)$ using $H =$ SHA-256, where $\ell = n/256$ (assuming for simplicity n is a multiple of 256), and the inputs to SHA-256 are encoded to the same length to avoid length extension attacks that make Merkle–Damgård constructions differentiable from a random oracle. Our implementation uses $r = 20$ rounds of the Feistel network as a safety margin for good indifferentiability and to avoid the non-tightness of the result [9] for $r = 8$.

6.2 CF-based schemes

CF. Our definition of a claw-free function generator follows the one of [13]. The generator FG specifies the following. Key-generation algorithm FG.Kg returns a pair (ek, ik) consisting of an evaluation key ek and an inversion key ik. Associated to ek is finite set $\mathsf{FG.D}(ek)$. Also specified are deterministic evaluation and inversion algorithms FG.Ev and $\mathsf{FG.Ev}^{-1}$. For $d \in \{0,1\}$, these in turn specify permutations $\mathsf{FG.Ev}_{ek,d}$: $\mathsf{FG.D}(ek) \to \mathsf{FG.D}(ek)$ with inverse $\mathsf{FG.Ev}^{-1}_{ik,d}$: $\mathsf{FG.D}(ek) \to \mathsf{FG.D}(ek)$. The assumption is claw-freeness, formalized by defining the cf-advantage of an adversary $\mathcal{A}$ against FG by $\mathbf{Adv}^{\mathrm{cf}}_{\mathsf{FG}}(\mathcal{A}) = \Pr[\mathrm{CF}^{\mathcal{A}}_{\mathsf{FG}}]$ where the game is in Fig. 18. There is an extraction algorithm FG.Ex that takes ek, x_0, x_1 such that $\mathsf{FG.Ev}_{ek,0}(x_0) = \mathsf{FG.Ev}_{ek,1}(x_1)$ —x_0, x_1 is referred to as a claw— and returns ik. There is a key-verification algorithm FG.KVf that takes ek, x and returns true iff $(ek, x) \in [\mathsf{FG.Kg}]$.

Game $\mathrm{CF}^{\mathcal{A}}_{\mathsf{FG}}$
$(ek, ik) \leftarrow\!\!\$\ \mathsf{FG.Kg}$
$(x_0, x_1) \leftarrow\!\!\$\ \mathcal{A}(ek)$
$y_0 \leftarrow \mathsf{FG.Ev}_{ek,0}(x_0)$
$y_1 \leftarrow \mathsf{FG.Ev}_{ek,1}(x_1)$
Return $(y_0 = y_1)$

Figure 18: Game defining claw-freeness of claw-free function generator FG.

For a string $w = w[1]\ldots w[n] \in \{0,1\}^n$, let $\mathsf{FG.Ev}_{ek,w}$: $\mathsf{FG.D}(ek) \to \mathsf{FG.D}(ek)$ and $\mathsf{FG.Ev}^{-1}_{ik,w}$: $\mathsf{FG.D}(ek) \to \mathsf{FG.D}(ek)$ be defined by

$\underline{\text{Function } \mathsf{FG.Ev}_{ek,w}(x)}$	$\underline{\text{Function } \mathsf{FG.Ev}^{-1}_{ik,w}(y)}$
For $i = 1, \ldots, n$ do $x \leftarrow \mathsf{FG.Ev}_{ek,w[i]}(x)$	For $i = n, \ldots, 1$ do $y \leftarrow \mathsf{FG.Ev}^{-1}_{ik,w[i]}(y)$
Return x	Return y

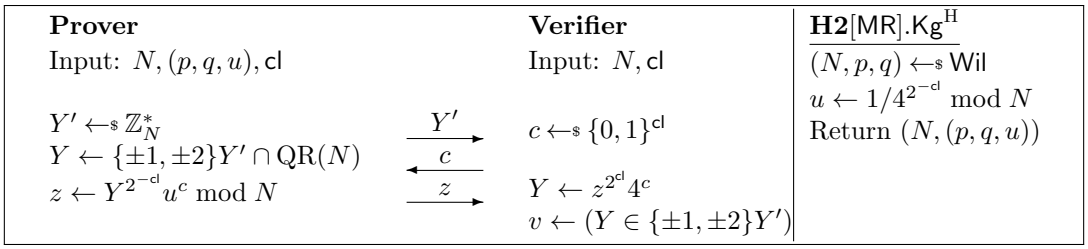


Figure 19: **Top left:** MR identification scheme; **remainder:** DAPS scheme $\mathbf{H2}[\mathsf{MR}, \mathsf{cl}, \mathsf{sl}]$ derived via our **H2** transform from ID scheme MR.

Fig. 17 shows the CF identification scheme associated to FG. It supports arbitrary challenge lengths, i.e., $\mathsf{ID.clS} = \mathbb{N}$. The commitment and response spaces are $\mathsf{ID.CS}(ivk, \mathsf{cl}) = \mathsf{ID.RS}(ivk, \mathsf{cl}) = \mathsf{FG.D}(ek)$ for all cl. The scheme is trivially trapdoor, with $\mathsf{CF.Cmt}^{-1}$ returning ε. Likewise, it is commitment recovering. The key-verification algorithm ID.KVf is the same as that of FG. Lemma 1 immediately shows that CF offers CIMP-UU and CIMP-UC security (the perfect extractability of FG implies perfect extractability of CF, and KR tightly reduces to claw-freeness).

H2[CF] AND **ID2**[CF]. Our **H2**[CF] and **ID2**[CF] DAPS schemes can be derived by applying the respective transforms to the CF identification scheme of Fig. 17, where the latter is based on a function generator FG. Due to tight reductions, security will amount to that of the function generator.

The classic claw-free function generator is the one by GMR [13] where evaluation keys ek coincide with RSA moduli N and $\mathsf{FG.D}(ek) = \mathrm{QR}(N)$. Unfortunately, publicly deciding $\mathrm{QR}(N)$ is assumed hard, and so is sampling elements from $\mathrm{QR}(N)$ without knowing a square root. As instantiating the **H2** and **ID2** transforms with CF would require a random oracle that maps into commitment space $\mathrm{QR}(N)$, **ID2** in addition requires a random permutation on response space $\mathrm{QR}(N)$, and the unique response property required by **ID2** does not necessarily hold for GMR's function generator, all in all it is not clear how to practically implement the schemes. We suggest two ways out: The first is to switch to the claw-free function generator suggested in [22] (their 2:1 trapdoor functions are similar in spirit to the function generator of GMR, but have samplable and decidable domains), and the second is to tweak **H2**[CF] a bit so that hashing into $\mathrm{QR}(N)$ is avoided. How to do this will become clear in Section 6.3.

6.3 CF-based schemes using MR

We next discuss our **H2**[MR] DAPS. While it does not entirely follow the generic approach of the previous sections, it can be seen as a close variant of **H2**[CF] instantiated with a squaring-

based construction of claw-free functions. It will become clear that by giving a direct construction we obtain a highly efficient scheme without sacrificing security. From the technical perspective, amongst others we show how to resolve the challenge of hashing into $\mathrm{QR}(N)$.

A Williams integer has the form $N = pq$ where p, q are prime numbers of about the same size and $p \equiv 3 \pmod 8$ and $q \equiv 7 \pmod 8$. We have $+1, +4 \in \mathrm{QR}(N)$ (trivially) and $-1, +2, -2 \notin \mathrm{QR}(N)$ (by the constraints put on p, q). Further, for any $x \in \mathbb{Z}_N^*$ precisely one element of $\{+x, -x, +2x, -2x\}$ is an element of $\mathrm{QR}(N)$. Williams integers are a special case of Rabin integers and thus each $x \in \mathrm{QR}(N)$ has a total of four square roots in $\mathbb{Z}_N^*$, precisely one of which finds itself in $\mathrm{QR}(N)$. Publishing N makes squaring an efficient public permutation on $\mathrm{QR}(N)$; its inverse, taking square roots, is however known to require knowledge of (secret) factors p, q.

Following [13], define functions $F_0, F_1 \colon \mathrm{QR}(N) \to \mathrm{QR}(N)$ such that $F_0(x) = x^2$ and $F_1(x) = 4x^2$. Generalize this definition to any $c \in \{0,1\}^*$ by setting $F_c(x) = F_{c_1 \ldots c_l}(x) = F_{c_l}(F_{c_1 \ldots c_{l-1}}(x)) = x^{2^l} 4^c \bmod N$ (where in the last term we write c to denote the integer represented by string c). As suggested in [12], if $u = 1/4^{2^{-|c|}}$ is precomputed, $F_c^{-1}(y) = y^{2^{-|c|}} u^c$. The following extraction property is reported in [18]: From any colliding quadruple x, c, x', c', i.e., $F_c(x) = F_{c'}(x')$ and $|c| = |c'|$ and $c \neq c'$, one can efficiently compute p and q. The hardness of finding such tuples can be (tightly) reduced to the hardness of factoring, the latter formalized by factoring advantage $\mathbf{Adv}^{\mathrm{fac}}_{\mathsf{Wil}}(\mathcal{A}) = \Pr[\mathrm{FAC}^{\mathcal{A}}_{\mathsf{Wil}}]$ of an adversary $\mathcal{A}$ against a Williams modulus generator Wil, where the game is in Fig. 20.

This setup suggests a (trapdoor) ID scheme where the commitment Y is a random value in $\mathrm{QR}(N)$, the challenge c is picked from $\{0,1\}^{\mathsf{cl}}$, and the response is obtained per $z \leftarrow F_c^{-1}(Y)$; verifying a transcripts means checking $Y = F_c(z)$. Unfortunately, for reasons discussed above, having $\mathsf{ID.CS} = \mathrm{QR}(N)$ as a commitment space is not compatible with our transformations. We correspondingly tweak the ID scheme as follows. We recall that for each $Y' \in \mathbb{Z}_N^*$ precisely one element $Y \in \{+Y', -Y', +2Y', -2Y'\}$ is in $\mathrm{QR}(N)$, and thus set $\mathsf{ID.CS} = \mathbb{Z}_N^*$, let the prover pick commitment $Y' \in \mathbb{Z}_N^*$, identify $Y \in \mathrm{QR}(N)$ corresponding to Y' (the prover can do this for knowing p, q), and complete the transcript, for challenge c, by computing response $z \leftarrow F_c^{-1}(Y)$. The specification of this ID scheme, that we call MR, is in Fig. 19, where with u we denote the (cl-dependent) precomputed value from above. The corresponding extraction algorithm, detailed in [18], we denote with $\mathsf{MR.Ex}$.

$\underline{\text{Game } \mathrm{FAC}^{\mathcal{A}}_{\mathsf{Wil}}}$
$(N, p, q) \leftarrow\!\!\$\ \mathsf{Wil}$
$r \leftarrow\!\!\$\ \mathcal{A}(N)$
Return $(r \in \{p, q\})$

Figure 20: Game defining factoring security of Williams modulus generator Wil.

H2[MR]. Taking the MR scheme and applying the **H2** transform to it yields the DAPS scheme **H2**[MR] shown in Fig. 19. Deriving Y during signing is efficient because p, q are known; further, the extractor can find the right values Y_i by evaluating the squaring chain of the verification algorithm on signature components z_i, c_i. Given the tightness of the reductions we can again pick the modulus based on the assumption that the NFS is the best factoring method.

6.4 Implementation and performance

Implementation. We implemented **H2**[GQ], **H2**[MR], and **ID2**[GQ]. For comparison purposes we also implemented the original PS and the standard RSA PKCS#1v.5 currently used by CAs for creating certificates. Our implementation is in C, using OpenSSL's BIGNUM library for number theoretic operations.[1] We use the Chinese remainder theorem to speed-up secret key operations whenever possible. For GQ, we use encryption exponent $e = \mathrm{nextprime}(2^{\mathsf{cl}})$; for RSA public key

[1] The source code can be downloaded from https://github.com/dstebila/daps.

			$k = 2048$-*bit modulus,* $n = l = 256$-*bit hash*			
Scheme	**Operation count**		**Runtime** (ms)		**Size** (bits)	
	sign	**verify**	**sign**	**verify**	**pub.**	**sig.**
PS [22]	$n \exp_k^k$	$n \exp_k^k$	516.58 ± 15.3	161.84 ± 7.96	2 048	528 384
RKS [24]	$2k$ grp exp	$2k$ grp dbl exp	13.48	5.99	640	131 072
H2[GQ] (Fig. 16)	$2 \exp_{k/2}^{k/2} + \exp_k^l$	$\exp_k^l$	0.88 ± 0.04	0.41 ± 0.02	6 144	2 304
ID2[GQ] (Fig. 16)	$4 \exp_{k/2}^{k/2} + 2 \exp_k^l$	$3 \exp_k^l$	1.80 ± 0.14	1.49 ± 0.26	6 144	2 049
H2[MR] (Fig. 19)	$2 \exp_{k/2}^{k/2} + \exp_k^l$	$1.5l$ mul$_k$	1.27 ± 0.16	0.37 ± 0.01	2 048	2 304
RSA PKCS#1v1.5	$2 \exp_{k/2}^{k/2}$	$\exp_k^{\lvert e \rvert}$	0.53 ± 0.08	0.02 ± 0.00	2 048	2 048

Figure 21: Operation count, average runtime in milliseconds, and public key/signature sizes of DAPS schemes and RSA signatures. By $\exp_m^x$ we denote the cost of a computing a modular exponentiation with modulus of bitlength m and exponent of bitlength x. See text for more information.

encryption we use OpenSSL's default public key exponent, $e = 65537$. We compared against the RKS DAPS implementation.

Performance. We measured timings of our implementations on an Intel Core i7 (6700K "Skylake") with 4 cores each running at 4.0 GHz. The tests were run on a single core with TurboBoost and hyper-threading disabled. Software was compiled for the `x86_64` architecture with `-O3` optimizations using `llvm` 8.0.0 (`clang` 800.0.38). The OpenSSL version used was v1.0.2j. We use RKS' implementation of their DAPS, which relies on a different library for the `secp256k1` elliptic curve. Table 21 shows mean runtimes in milliseconds (with standard deviations) and key sizes using 2048-bit modulii and 256-bit hashes. For DAPS schemes, address is 15 bytes and payload is 33 bytes; for RSA PKCS#1v1.5, message is 48 bytes. Times reported are an average over 30 seconds. The table omits runtimes for key generation as this is a one-time operation.

Compared with the existing PS, our **H2**[GQ], **ID2**[GQ], and **H2**[MR] schemes are several orders of magnitude faster for both signing and verification. When using 2048-bit modulii, **H2**[GQ] signatures can be generated 587× and verified 394× faster, and **ID2**[GQ] signatures can be generated 287× and verified 108× faster; moreover our signatures are 229× and 257× shorter, respectively, compared with PS, and ours are nearly the same size as RSA PKCS#1v1.5 signatures. Compared with the previous fastest and smallest DAPS, RKS, **H2**[GQ] signatures can be generated and verified 15× faster; **ID2**[GQ] generated 7× and verified 4× faster; and **H2**[MR] generated 10× and verified 16× faster. **H2**[GQ] and **H2**[MR] signatures are 56× shorter compared with RKS; **H2**[GQ] and **ID2**[GQ] public keys are 9.6× larger, though still under 1 KiB total, and **H2**[MR] keys are only 3.2× larger than RKS.

Signing times for our schemes are competitive with RSA PKCS#1v1.5: using **H2**[GQ], **ID2**[GQ], or **H2**[MR] for signatures in digital certificates would incur little computational or size overhead relative to currently used signatures.

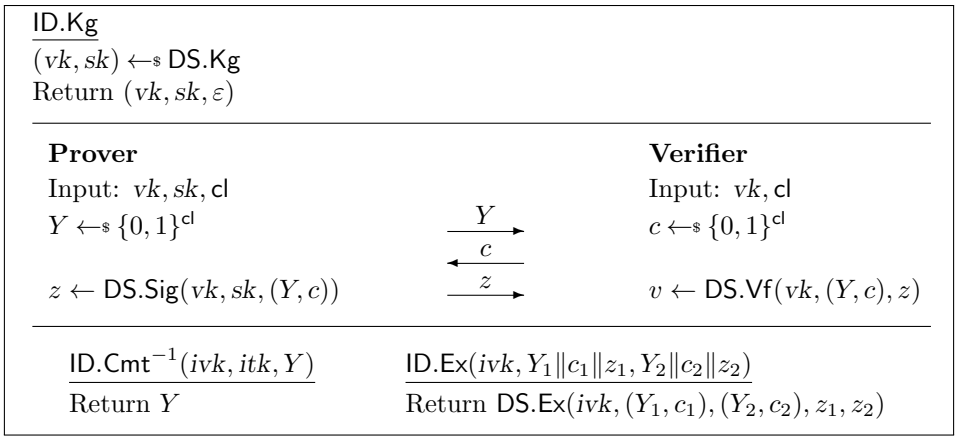


Figure 22: Our construction of a trapdoor identification scheme $\mathsf{ID} = \mathbf{DAPS2ID}[\mathsf{DS}]$ from a DAPS DS.

7 From DAPS to trapdoor ID

Here we show that DAPS implies trapdoor identification that is CIMP-UC and CIMP-UU secure, which shows that the assumptions we make to obtain DAPS are necessary.

Construction. Let DS be a DAPS, meaning a signature scheme satisfying double authentication prevention and unforgeability. We build from it the identification scheme $\mathsf{ID} = \mathbf{DAPS2ID}[\mathsf{DS}]$ depicted in Fig. 22. The set $\mathsf{ID.clS}$ of admissible challenge lengths is the set of positive integers. Key generation algorithm $\mathsf{ID.Kg}$ lets $(vk, sk) \leftarrow\$ \mathsf{DS.Kg}$ and returns $(ivk, isk, itk) = (vk, sk, \varepsilon)$. The commitment is a random cl-bit string, meaning $\mathsf{ID.Cmt}(ivk, \mathsf{cl})$ picks $Y \leftarrow\$ \{0,1\}^{\mathsf{cl}}$ and returns (Y, Y). The response is a signature of the commitment and challenge, meaning $\mathsf{ID.Rsp}(vk, sk, c, Y)$ returns $z \leftarrow\$ \mathsf{DS.Sig}(vk, sk, (Y, c))$. This identification scheme is trivially trapdoor, $\mathsf{ID.Cmt}^{-1}(vk, \varepsilon, Y, \mathsf{cl})$ returning Y.

Properties. We have already shown that $\mathsf{ID} = \mathbf{DAPS2ID}[\mathsf{DS}]$ is trapdoor. We want to show it is CIMP-UC and CIMP-UU secure. In fact we will show something stronger, namely that it satisfies extractability and security against key recovery under passive attack. CIMP-UC and CIMP-UU will then follow from Lemma 1.

Theorem 6 *Let DS be a DAPS and let $\mathsf{ID} = \mathbf{DAPS2ID}[\mathsf{DS}]$. Let $\mathcal{A}$ be an ex-adversary against ID. From $\mathcal{A}$ we construct dap-adversary $\mathcal{A}_1$ such that $\mathbf{Adv}^{\mathrm{ex}}_{\mathsf{ID}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{dap}}_{\mathsf{DS}}(\mathcal{A}_1)$. The running time of $\mathcal{A}_1$ is that of $\mathcal{A}$.*

Proof: Adversary $\mathcal{A}_1$ is shown in Fig. 23. $\mathcal{A}_1$ directly calls $\mathcal{A}$ which is an ex adversary against the identification scheme ID. Note that, for $\mathsf{ID} = \mathbf{DAPS2ID}[\mathsf{DS}]$, the trapdoor key is $itk = \varepsilon$, so this is a perfect simulation of $\mathbf{G}^{\mathrm{ex}}_{\mathsf{ID}}(\mathcal{A})$. If $\mathcal{A}$ returns two accepting transcripts $Y \| c_1 \| z_1$ and $Y \| c_2 \| z_2$ with $c_1 \neq c_2$, then (Y, c_1) and (Y, c_2) are a pair of colliding messages for DS and z_1 and z_2, respectively, are valid signatures. $\mathsf{ID.Ex}$ fails to return the correct secret key from this part of transcripts exactly when $\mathsf{DS.Ex}$ fails. The bound in the theorem statement follows. ∎

Theorem 7 *Let DS be a DAPS and let $\mathsf{ID} = \mathbf{DAPS2ID}[\mathsf{DS}]$. Let $\mathcal{I}$ be a kr-adversary against ID. From $\mathcal{I}$ we construct uf-adversary $\mathcal{A}_2$ such that $\mathbf{Adv}^{\text{kr-pa}}_{\mathsf{ID}}(\mathcal{I}) \leq \mathbf{Adv}^{\mathrm{uf}}_{\mathsf{DS}}(\mathcal{A}_2)$. The running time of $\mathcal{A}_2$ is that of $\mathcal{I}$.*

Adversary $\mathcal{A}_1(vk, sk)$
$(Y, c_1, z_1, c_2, z_2) \leftarrow\!\!\$\ \mathcal{A}(vk, sk, \varepsilon)$
Return $((Y, c_1), (Y, c_2), z_1, z_2)$

Figure 23: Adversary for proof of Theorem 6.

Adversary $\mathcal{A}_2^{\text{SIGN}}(vk)$	$\text{TR}^*(\mathsf{cl})$
$L \leftarrow \emptyset$	$Y \leftarrow\!\!\$\ \{0,1\}^{\mathsf{cl}}$; $c \leftarrow\!\!\$\ \{0,1\}^{\mathsf{cl}}$
$isk^* \leftarrow\!\!\$\ \mathcal{I}^{\text{TR}^*}(vk)$	$z \leftarrow\!\!\$\ \text{SIGN}(Y\|c)$
Pick some m with $\|m\| \notin L$	$T \leftarrow Y\|c\|z$; $L \leftarrow L \cup \{\|Y\|c\|\}$
$\sigma \leftarrow\!\!\$\ \mathsf{DS.Sig}(ivk, isk^*, m)$	Return T
Return (m, σ)	

Figure 24: Adversary for proof of Theorem 7.

Proof: Adversary $\mathcal{A}_2$ is shown in Fig. 24. To create transcripts, $\mathcal{A}_2$ uses its SIGN oracle. It stores the lengths of all messages that are signed. When $\mathcal{I}$ returns a signing key, $\mathcal{A}_2$ uses it to sign a message and create a forgery. To avoid this message m having been a SIGN query, it picks it to have a length different from the length of any message queried to SIGN. ∎

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, Apr. / May 2002.

[2] M. Bellare, B. Poettering, and D. Stebila. From identification to signatures, tightly: A framework and generic transforms. In *ASIACRYPT 2016*, LNCS. Springer, 2016.

[3] M. Bellare and T. Ristov. Hash functions from sigma protocols and improvements to VSH. In J. Pieprzyk, editor, *ASIACRYPT 2008*, volume 5350 of *LNCS*, pages 125–142. Springer, Dec. 2008.

[4] 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.

[5] 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, May / June 2006.

[6] J. Black and P. Rogaway. Ciphers with arbitrary finite domains. In B. Preneel, editor, *CT-RSA 2002*, volume 2271 of *LNCS*, pages 114–130. Springer, Feb. 2002.

[7] D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In S. Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 319–327. Springer, Aug. 1990.

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

[9] Y. Dai and J. P. Steinberger. Indifferentiability of 8-round feistel networks. In M. Robshaw and J. Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 95–120. Springer, Aug. 2016.

[10] C. Evans, C. Palmer, and R. Sleevi. Public Key Pinning Extension for HTTP. RFC 7469 (Proposed Standard), Apr. 2015.

[11] 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, Aug. 1987.

[12] O. Goldreich. Two remarks concerning the Goldwasser-Micali-Rivest signature scheme. In A. M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 104–110. Springer, Aug. 1987.

[13] 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.

[14] 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, Aug. 1990.

[15] R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In *21st ACM STOC*, pages 44–61. ACM Press, May 1989.

[16] H. Krawczyk and T. Rabin. Chameleon signatures. In *NDSS 2000*. The Internet Society, Feb. 2000.

[17] B. Laurie, A. Langley, and E. Kasper. Certificate Transparency. RFC 6962 (Experimental), June 2013.

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

[19] S. Miracle and S. Yilek. Reverse cycle walking and its applications. In *ASIACRYPT 2016*, LNCS. Springer, 2016 (to appear).

[20] 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, Aug. 1998.

[21] B. Poettering and D. Stebila. Double-authentication-preventing signatures. In M. Kutylowski and J. Vaidya, editors, *ESORICS 2014, Part I*, volume 8712 of *LNCS*, pages 436–453. Springer, Sept. 2014.

[22] B. Poettering and D. Stebila. Double-authentication-preventing signatures. *International Journal of Information Security*, December 2015.

[23] J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *22nd ACM STOC*, pages 387–394. ACM Press, May 1990.

[24] T. Ruffing, A. Kate, and D. Schröder. Liar, liar, coins on fire!: Penalizing equivocation by loss of bitcoins. In I. Ray, N. Li, and C. Kruegel:, editors, *ACM CCS 15*, pages 219–230. ACM Press, Oct. 2015.

[25] C. Timberg. Apple will no longer unlock most iPhones, iPads for police, even with search warrants, Sept. 2014. Washington Post, http://www.washingtonpost.com/business/technology/2014/09/17/2612af58-3ed2-11e4-b03f-de718edeb92f_story.html.

A Applicability of DAPS

Applicability of DAPS. As a reader may justifiably point out, various issues must be addressed for PS's application of DAPS to the deterrence of certificate subversion, that we sketched above, to be a full solution. For example, there may be legitimate reasons for a CA to issue a new certificate in the name of `example.com` (the old one may have expired or been revoked) which at first glance is precluded by DAPS. Or, big brother might approach a different CA. (Indeed, the DAPS idea is inherently restricted to a single CA environment.) There are various answers to these questions which in particular are discussed to some extent by PS [22].

One might also ask why a CA would want, or agree, to use DAPS. Recently, we have seen Internet corporations opposing court orders asking them to compromise security of their products (Apple vs. FBI) or taking steps to make subversion harder (Google's push for end-to-end encryption

or the information that Apple "reworked its encryption in a way that prevents the company ... from getting access to the ... user data stored on smartphones and tablet computers" [25]). A CA might similarly see espousing DAPS.

We will not however attempt to address application issues in full here. Whether DAPS as a concept has true practical utility remains to be seen, but our sense is that DAPS is a tool of sufficient technical interest and *potential* applicability to merit research and exposure of this research. Efficient schemes, such as the ones we provide, are a necessary (even if not sufficient) condition for application.

Comparing DAPS and other PKI technologies. Other recent developments aim to deter, detect, or mitigate the risks from rogue certificates. Certificate Transparency (CT) [17] involves a network of public logs and monitors with the goal of ensuring that all server certificates observed by clients are eventually visible in public logs; monitors, acting on behalf of domain owners, can then watch for rogue certificates. For a malicious CA issuing rogue certificates, CT implies that their malicious actions are more likely to be detected. However, the result of their malicious action remains uncertain: depending on the policy of the relying parties (such as browser vendors), a few rogue certificates, with plausible excuses for their issuance, may be tolerated. With DAPS, no such luck: a single double-issued certificate is enough to ruin the CA. In fact, CT could work hand-in-hand with DAPS to ensure greater visibility of colliding messages, and thus even greater deterrence to CAs.

Public key pinning [10] allows a web server to indicate to clients that future connections should involve certain certificates: clients can "pin" the server's cryptographic identity. This can be used by a web server to restrict which CAs should be trusted to issue certificates for this domain; since browsers trust many CAs by default, this allows a server to reduce its attack surface by drastically reducing the number of CAs clients should trust for this domain. (Pinning is "trust-on-first-use", so its promises only apply if the first connection is uncompromised.) Pinning and DAPS could work together: a server uses pinning to restrict certificate issuance to one CA (or a handful of CAs), and then DAPS prevents that CA from issuing rogue certificates for that domain.

B Details on prior DAPS construction of RKS

We reproduce some details of the DAPS suggested by RKS [24, Appendix A] and assess its efficiency. In a nutshell, the scheme builds a Merkle tree from a chameleon hash function and associates leaf nodes with DAPS addresses. The size of signatures is linear in the height of the tree.

A chameleon hash function (CHF, [16]) is a randomized hash function where collisions are hard to find unless some trapdoor information is known. Technically, a CHF consist of algorithms $\mathsf{ChGen}, \mathsf{ChHash}, \mathsf{ChCol}$ where ChGen outputs a public hashing key hk and a secret collision-finding key cfk, ChHash implements a (public) randomized hash function in the sense that any value $h = \mathsf{ChHash}_{hk}(m; r)$ is considered a hash value of message m (with randomness r), and ChCol is a (secret) collision finding algorithm that for any m, r, m' finds a randomness $r' = \mathsf{ChCol}_{cfk}(m, r, m')$ such that $\mathsf{ChHash}_{hk}(m; r) = \mathsf{ChHash}_{hk}(m'; r')$. The important security notion is collision resistance: given hk (but not the trapdoor cfk), it should be hard to find colliding $(m; r), (m'; r')$. Further, a CHF is extractable if from any two colliding $(m; r), (m'; r')$ the trapdoor cfk can be recovered. Many constructions of CHF are known [3], and a standard one in the DLP setting is based on Pedersen commitments, setting $hk = h = g^x$, $cfk = x$, and $\mathsf{ChHash}(m; r) = g^m h^r$, and implementing ChCol by solving the equation $m + xr = m' + xr'$ for r'.

We give a brief overview of the DAPS scheme of RKS; for details we refer to [24]. The scheme combines a CHF, a pseudo-random function (PRF), and a regular hash function H to build a

Merkle tree, as follows. Let $l, n \in \mathbb{N}$ be parameters defining the height of the tree and the arity of its inner nodes, respectively. A DAPS signing key sk consists of a CHF collision-finding key cfk and a PRF key K, and a DAPS verification key vk consists of the hash value of the root node, the CHF hashing key hk, and an encryption $H(cfk) \oplus K$ of the PRF key under the collision-finding key. To the root and all inner nodes we assign values of the form $z = H(y_1, \ldots, y_n)$ where inputs y_i are associated to the node's n children and computed per $y_i = \mathsf{ChHash}(x_i; r_i)$ with values x_i, r_i computed using the PRF on input a string encoding the position of the respective child node in the tree. Note this construction allows for computing the DAPS verification key in $O(n)$ time, i.e., without considering all n^l nodes of the tree individually.

The idea of RKS is to associate DAPS addresses with the tree's leaf nodes: For signing a message $m = (a, p)$, the leaf node corresponding to a is identified, a value $z = H(p)$ is computed for it, and the ChCol algorithm is invoked to find r' such that $(z; r')$ and the node's PRF-generated pair $(x; r)$ hash, under hk, to the same value y. The DAPS signature σ consists of this value r', plus, as is common for Merkle tree based signatures, for each intermediate node on the path from the leaf to the root one vector $(y_1, \ldots, y_n)$ (actually, with one element missing) and a randomness r'' such that $z'' = H(y_1, \ldots, y_n)$ and r'' hash, under hk, to the next level's y value. It is not difficult to see that the DAP property of this construction reduces to the extractability of the CHF, and the unforgeability to its collision resistance.

Efficiency We assess the efficiency of the RKS scheme with respect to key sizes, signature sizes, signing costs, and verification costs. We assume the underlying CHF is the DLP-based one described above as it promises shortest keys and signatures, and fastest signing and verification times. This choice was also made in [24]. Due to generic attacks, targeting a security level of k (e.g., $k = 128$) implies that group elements and exponents have size at least $2k$ bits. Assuming that PRF keys have length k, we obtain that DAPS verification keys have size $2k + 2k + k = 5k$, signing keys have size $2k + k = 3k$, and signatures have size $2k + (l-1)((n-1)2k + 2k) \approx 2nlk$ (all sizes in bits). Signing requires $n - 1$ exponentiations per node on the path connecting the leaf with the root (for simplicity we do not count PRF and ChCol invocations), plus one exponentiation for the leaf node, i.e., a total of about $(n-1)l$ exponentiations. Verification requires one double-exponentiation per path node, and one for the leaf, i.e., a total of l double-exponentiations.

To get concrete numbers for security level $k = 128$ we plug in some combinations of parameters (n, l). Observe that the number of leaf nodes is given by n^l and determines the cardinality of the DAPS address space; we thus require $n^l \geq 2^{2k}$. If we work with a binary tree, i.e., have $n = 2$, we obtain $l = 2k$ and thus signatures of size $2nlk = 16\,\text{KiB}$; a verification would involve 256 double-exponentiations. For trees with arity $n = 4$ (and $n = 8$) we obtain $l = k$ (and $l = \lceil 2k/3 \rceil$), i.e., signatures have size 16 KiB (22 KiB) and require 128 (resp. 85) exponentiations to be verified.