Double-Authentication-Preventing Signatures from Trapdoor Identification

Mihir Bellare[1] Douglas Stebila[2]

February 2015

Abstract

This paper presents efficient designs and software implementations of signature schemes that are double authentication preventing. We give a general transform for constructing these double-authentication preventing signatures (DAPS) from a special class of identification schemes that we define and call trapdoor. We instantiate this to get specific schemes, namely GQ-DAPS (based on RSA) and CF-DAPS (using factoring-based claw-free functions). Our implementations, using OpenSSL's crypto library on an Intel Core i7, show that our DAPS schemes are not only significantly more efficient than prior DAPS schemes 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 and a gift from Microsoft corporation.

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

Contents

1 Introduction

DAPS. Double-authentication-preventing signature (DAPS) schemes were introduced by Poettering and Stebila (PS) [25]. 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 $(a_0, p_0), (a_1, p_1)$ are *colliding* if $a_0 = a_1$ but $p_0 \neq p_1$. The double-authentication prevention requirement is that there is an efficient extraction algorithm that given a public key PK and valid signatures σ_0, σ_1 on colliding messages $(a, p_0), (a, p_1)$, respectively, returns the secret signing key SK underlying PK. Additonally, the scheme must satisfy standard unforgeability under a chosen-message attack [16], 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 [25] discuss several potential applications. For completeness, let us briefly recall one. The Snowden revelations have shown that the NSA may coerce corporations into measures that compromise security. PS [25] consider, in this light, the subversion of certificate authorities (CAs) and the use of DAPS as a deterrent. Thus, suppose `example.com` has a (legitimate) certificate $\text{cert}_1 = (\texttt{example.com}, pk_1, \sigma_1)$ from a particular CA such as Comodo, where 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 induces the CA to issue another certificate $\text{cert}_0 = (\texttt{example.com}, pk_0, \sigma_0)$ in the name of `example.com` where pk_0 is a public key supplied by big brother, so that it knows the corresponding secret key sk_0, and σ_0 is the CA's signature on the pair $(\texttt{example.com}, pk_0)$, 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 the latter. But if the CAs signatures are produced with a DAPS, then σ_1, σ_2 are valid signatures on the colliding messages $(\texttt{example.com}, pk_0), (\texttt{example.com}, pk_1)$, respectively, which means that anyone can extract the CA's signing key SK. This would lead to public loss of reputation and business for the CA, increasing the CA's incentive, or giving it an argument, to not comply with big brother's request to create the rogue certificate.

Prior schemes. PS [25] give a factoring-based DAPS that we call PS-DAPS. Its signature contains $n + 1$ elements in the group $\mathbb{Z}_N^*$, where n is the length of a hash of the address and N is the modulus in the public key. Specifically, for 80-bit security (1024-bit modulus, 160-bit hash), a signature contains 161 group elements, for a length of 164,864 bits or about 20 Kbytes. This is a factor 161 times longer than a 1024 bit RSA PKCS#1 signature, more than enough to preclude use of the scheme in practice. Furthermore, signing and verifying times are significantly greater than for signature schemes currently used for certificates such as RSA PKCS#1 (cf. Fig. 11.)

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. This is what we deliver. We will actually obtain numerous schemes via a new and general paradigm that transforms a class of identification schemes, that we call *trapdoor*, into DAPS schemes. For concreteness let us first sketch two particular DAPS schemes that we obtain in this way. Then we will discuss the transform.

CF-DAPS and GQ-DAPS. In our CF-DAPS, the public key is a pair (f_0, f_1) of claw-free permutations on a domain D [16], the secret key being (f_0^{-1}, f_1^{-1}). To an l-bit string $c = c[1] \ldots c[l]$ we associate the permutation f_c: $D \to D$ defined for $x \in D$ via $f_c(x) = f_{c[1]}(f_{c[2]}(\cdots(f_{c[l]}(x))\cdots))$, and let f_c^{-1} denote its inverse. To sign message (a, p), pick a random sl-bit s —the seed length sl is a parameter of the scheme— let $Y = H_2(a) \in D$, let $c = H_1(Y \| a \| p \| s) \in \{0,1\}^l$, let $z = f_c^{-1}(Y)$, and return (z, s) as the signature, where H_1, H_2 are public, collision-resistant hash functions. We

omit describing verification here; see Fig. 10 for a full description of the scheme. Given valid signatures $(z_0, s_0), (z_1, s_1)$ on colliding messages $(a, p_0), (a, p_1)$, respectively, one can compute a claw for f_0, f_1 —meaning an input $x \in D$ such that $f_0(x) = f_1(x)$— leading to recovery of f_0^{-1}, f_1^{-1}. This yields double-authentication-prevention. Using a factoring-based instantiation of the claw-free permutations from [16] and exploiting some clever computational number-theoretic tricks from [15] we will get a scheme that is efficient on all fronts (cf. Fig. 11).

In our GQ-DAPS, the public key is (N, e, X, TK) and the secret key is (x, d) where $N = pq$ is an RSA modulus, e is an encryption exponent, d is the corresponding decryption exponent, $x, X \in \mathbb{Z}_N^*$ satisfy $X = x^e \bmod N$, and $TK = H_1(x) \oplus d$ where H_1 is a public hash function. The keys are thus as in GQ [17] except that we add d to the secret key and TK to the public key. To sign message (a, p), pick a random sl-bit s —the seed length sl is a parameter of the scheme— let $Y = H_2(a)$ —the commitment is not picked at random but determined uniquely by the address— let $y = Y^d \bmod N$, let $c = H_3(Y \| a \| p \| s) \in \{0,1\}^l$ —the challenge— let $z = yx^c \bmod N$ and return (z, s) as the signature, where H_2, H_3 are public, collision-resistant hash functions. We omit describing verification; see Fig. 9 for a full description of the scheme. Given valid signatures $(z_0, p_0), (z_1, p_1)$ on colliding messages $(a, p_0), (a, p_1)$, respectively, one has GQ [17] conversation transcripts with the same commitment and different challenges —this is why we set the commitment to a hash of the address— and can use the GQ Sigma protocol extractability property to extract x. This is not quite enough for double-authentication-prevention because we must also extract d. It was for this that TK was put in the public key: from x we can recover $d = H_1(x) \oplus TK$. Efficiency is again good on all fronts (cf. Fig. 11).

In both these schemes, the double-authentication prevention property is easy to see; indeed the schemes were designed so that this is true. The challenge is showing unforgeability. This will emanate from the general results we discuss next.

DAPS FROM TRAPDOOR IDENTIFICATION. We obtain both the above schemes, and more, via a general transform that turns a class of identification schemes that we call *trapdoor* into DAPS. By an identification scheme we mean a three-move Sigma protocol in which the prover sends a *commitment* computed using private randomness, the verifier sends a random *challenge*, the prover returns a *response* computed using the prior private randomness and its secret key, and the verifier computes a boolean decision from the conversation transcript and public key (see Fig. 2). We call such a scheme *trapdoor* if the prover can pick the commitment directly at random from the space of commitments and then compute the associated private randomness using its secret key. Not all identification schemes have this property. For example GQ [17] does —this requires adding d to the secret key as above— but Schnorr's (discrete-log based) protocol [28] doesn't. We present a way to convert any trapdoor identification scheme into a DAPS scheme. Our DAPS sets the commitment to a hash of the address, computes the private randomness via the trapdoor, and then follows a randomized version of the Fiat-Shamir transform [13, 1]. Additionally the public key is enhanced so that recovery of the secret identification key allows recovery of the full DAPS secret key. See Section 4. GQ-DAPS is obtained in this way from the GQ identification scheme of [17]. CF-DAPS is obtained in this way from a trapdoor identification scheme based on claw-free permutations that we will define and call CF-ID. By applying our transform to the Fiat-Shamir [13] or Ong-Schnorr [24] identification schemes we can obtain further DAPS, but they are less efficient than CF-DAPS and GQ-DAPS.

Setting the commitment to a hash of the address ensures that the conversation transcripts corresponding to signatures of colliding messages have the same commitment, so that double-authentication prevention of the DAPS can be shown based on the Sigma protocol extractability property of the identification scheme. We prove unforgeability of the DAPS under the assumption

that the identification scheme is secure against multiple impersonation attempts under passive attack, a notion we define and call *mimp*. This has two advantages. On the theoretical side, we can establish mimp security under standard assumptions —one-wayness of RSA for GQ and hardness of factoring for CF— using the reset lemma of [6]. On the practical side we get improved concrete security compared to using the forking lemma [26, 5, 4], which means we can use smaller modulii and thus gain efficiency. We now discuss the latter in more depth.

TIGHT REDUCTIONS. Proofs of unforgeability of Fiat-Shamir signatures —by this we mean signatures derived from identification schemes via the Fiat-Shamir transform [13]— traditionally exploited forking lemmas [26, 5, 4]. We could take the same route for DAPS. However forking lemmas, and thus reductions based on them, are notoriously non tight, resulting in large losses in concrete security, and thus in efficiency because one must use larger modulii. Meanwhile cryptanalysis indicates that these losses are not real, but artifacts of the proof.

Abdalla, An, Bellare and Namprempre [1] provide an alternative approach, proving unforgeability of Fiat-Shamir signatures by reduction to the security of the identification scheme against impersonation under passive attack (imp). The latter can then be established, separately, via the reset lemma of [6]. This results in a more modular proof. But the concrete security of the reduction is the same as with the forking lemma approach because the reset lemma too is not tight.

We take this a step further. Our metric of security of an identification scheme is security against *multiple* impersonation attempts under passive attack (mimp). We are able to give a *tight* reduction of the unforgeability of our DAPS to the mimp security of the underlying trapdoor identification scheme (cf. Theorem 2). Now, rather than estimate mimp security via the reset lemma, we estimate it cryptanalytically. Our bound corresponds to the assumption that the best mimp attack is to guess a challenge or break the underlying algebraic problem. From this and Theorem 2, we can get 80-bit security with a 1024-bit modulus just as for standard RSA PKCS#1 signatures.

IMPLEMENTATION. In theoretical cryptography, "efficient" often just means "polynomial time," which is quite divorced from efficiency in practice. Some works measure "efficiency" by counting modular exponentiations or hash operations. Even these estimates can, in our experience, be moot. Implementation is key to gauge and show efficiency. We implement GQ-DAPS, CF-DAPS and the prior PS-DAPS using OpenSSL's BIGNUM library on an Intel Core i7 machine for both 1024-bit and 2048-bit modulii. (The latter is what commercial CA's currently use.) Fig. 11 shows the signing time, verifying time, signature size and key sizes for all schemes. GQ-DAPS, CF-DAPS emerge as around 150 times faster than PS-DAPS for signing and verifying while also having signatures about 140 times shorter. In fact the Figure shows that GQ-DAPS, CF-DAPS are close to RSA PKCS #1v1.5 in all parameters and runtimes. 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 Appendix C that from *any* DAPS satisfying double-authentication-prevention and unforgeability, one can build a trapdoor identification scheme that is mimp-secure and satisfies the Sigma protocol extractability property. This shows that the assumption we make is effectively necessary for DAPS.

DISCUSSION, RELATED WORK AND OPEN QUESTIONS. 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 [25]. One might also ask why a CA would want, or agree, to use DAPS. Recently, we have seen Internet corporations taking steps to make subversion harder. Google's push for end-to-end encryption following the Snowden revelations is one instance. In another, Apple "reworked its encryption in a way that prevents the company ... from getting access to the ... user data stored on smartphones and tablet computers" [29]. A CA might similarly see espousing DAPS. We will not however attempt to address application issues in full here. Our motivation for this work has been theoretical interest (we find the primitive and problem technically intriguing) and the perspective that efficient, secure schemes are a necessary, even if not sufficient, condition for application. Whether DAPS as a concept has true practical utility remains to be seen, but, if it does, our schemes are better choices than prior ones.

The concept of trapdoor identification schemes is implicit in Micali and Reyzin (MR) [22], who point to this property for some specific schemes and exploit it to obtain signature schemes with better concrete security than given by the Fiat-Shamir transform [13, 26, 1]. Our definition of trapdoor identification schemes names, generalizes and formalizes their idea. MR [22] also specify an identification scheme, underlying the MSA signature scheme of [21], that is the same as the special case of our CF-ID when the claw-free permutations are given by the squaring modulo a composite construction of [16].

AFLT [3] present tight reductions for Fiat-Shamir signatures from lossy identification. This is further improved in ABP [2]. Extending this to DAPS is an interesting direction for future work. An anonymous reviewer of a prior version of this paper asked whether it is possible to instantiate our generic construction with lattice-based identification schemes from [19, 20].

Both our DAPS and that of PS [25] are proven in the random oracle model. This raises the foundational question of what are the minimal assumptions under which DAPS can be built in the standard model. Ordinary signatures are possible from any one-way function [27]. 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 secret-key exchange that is not likely to be possible based on one-way functions [18]?

The DAPS property that the secret key is recoverable from signatures of colliding properties is conceptually similar to the recoverability of the spender's identity from double-spending of an e-coin in offline e-cash [10]. Whether this connection can be exploited to obtain new DAPS schemes is an open question.

2 DAPS definitions

Signatures. In a signature scheme DS, the signer generates signing key sk and verifying key vk via $(vk, sk) \leftarrow\!\!\$\ \mathsf{DS.Kg}^{\mathrm{H}}$ where H is the random oracle [7]. Now it can compute a signature $\sigma \leftarrow\!\!\$\ \mathsf{DS.Sig}^{\mathrm{H}}(vk, sk, m)$ on any message $m \in \{0,1\}^*$. A verifier can deterministically compute a boolean $v \leftarrow \mathsf{DS.Vf}^{\mathrm{H}}(vk, m, \sigma)$ indicating whether or not σ is a valid signature of m relative to vk. Correctness as usual requires that $\mathsf{DS.Vf}^{\mathrm{H}}(vk, m, \mathsf{DS.Sig}^{\mathrm{H}}(vk, sk, m)) = \mathsf{true}$ with probability one.

The DAP property. In a DAPS [25], a message $m = (a, p)$ is a pair consisting of an *address* a and a *payload* p. Let us say that messages $m_1 = (a_1, p_1)$ and $m_2 = (a_2, p_2)$ are *colliding* if $a_1 = a_2$ but $p_1 \neq p_2$. Double authentication prevention [25] requires that signatures on colliding messages allow anyone to extract the signing key. It is captured formally by the advantage $\mathbf{Adv}^{\mathrm{dap}}_{\mathsf{DS}}(\mathcal{A}) = \Pr[\mathrm{DAP}^{\mathcal{A}}_{\mathsf{DS}}]$ associated to adversary $\mathcal{A}$, where game $\mathrm{DAP}^{\mathcal{A}}_{\mathsf{DS}}$ is in Fig. 1. The adversary produces messages m_1, m_2 and signatures σ_1, σ_2, and an extraction algorithm $\mathsf{DS.Ex}^{\mathrm{H}}$ associated to the

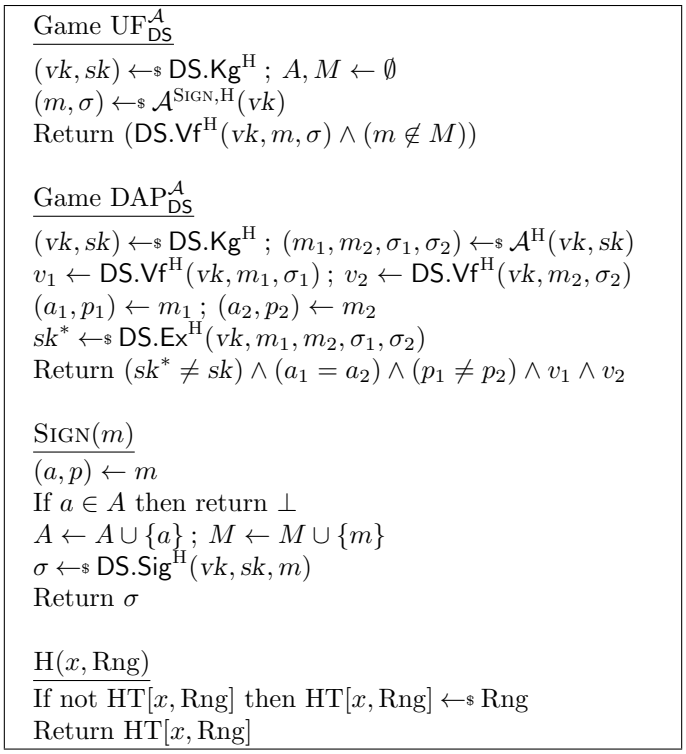


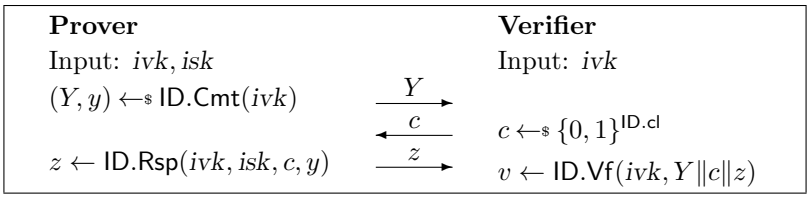
Figure 1: Games defining unforgeability and extractability conditions of DAPS DS. The $\textsc{Sign}$ procedure is invoked by game UF while H is invoked by both games.

---

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. If G is a game, we are denoting by $\Pr[\mathrm{G}]$ —here and in the rest of the paper— the probability that the game returns true. The argument Rng to the random oracle H allows the caller to specify the set from which responses are drawn in a particular scheme, for example $\mathbb{Z}_N^*$. 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 $\textsc{Sign}$ 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.

$\textsc{Unforgeability}$. We also require unforgeability, captured formally by the advantage $\mathbf{Adv}^{\mathrm{uf}}_{\mathsf{DS}}(\mathcal{A}) = \Pr[\mathrm{UF}^{\mathcal{A}}_{\mathsf{DS}}]$ associated to adversary $\mathcal{A}$, where game $\mathrm{UF}^{\mathcal{A}}_{\mathsf{DS}}$ is in Fig. 1 [25]. This is the classical notion of [16, 7] 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.

$\textsc{Discussion}$. Asking that the key sk^* returned by the extractor $\mathsf{DS.Ex}^{\mathrm{H}}$ be equal to sk may seem unnecessarily strong. It would suffice if sk^* was "functionally equivalent" to sk, meaning allowed



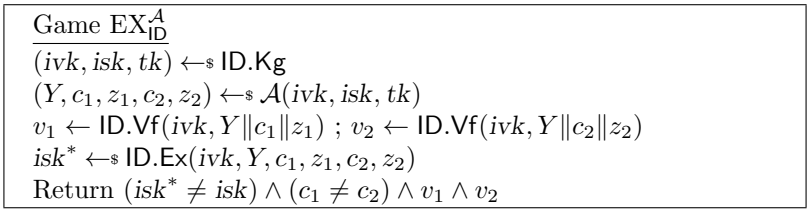


Figure 2: Functioning of an identification scheme ID and game defining its Sigma-Protocol extractability.

computation of signatures indistinguishable from real ones. Indeed, such a property is formalized in PS [25]. However our schemes achieve the stronger property we have defined, so 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 [25]. Our view is that our requirement is reasonable because the coercion happens after the CA and its keys are established. If the choice of keys is considered a potential source of vulnerability, one might generate them via secure computation between a few different parties.

3 Trapdoor identification schemes

We define a sub-class of identification schemes that we call trapdoor. Later we will show how any such scheme can be transformed into a DAPS. The trapdoor property was first recognized in [22] for specific example schemes. We name, generalize and formalize it here.

Identification. An identification (ID) scheme ID operates as depicted in Fig. 2. First, via $(isk, ivk, tk) \leftarrow\$ \mathsf{ID.Kg}$, the prover generates a private *identification key* isk, public verification key ivk and auxiliary information tk. Via $(Y, y) \leftarrow\$ \mathsf{ID.Cmt}(ivk)$ it generates *commitment* Y and corresponding private state y. The verifier sends a random challenge of length $\mathsf{ID.cl}$. The prover's *response* z and the verifier's boolean *decision* v are deterministically computed. An example is the GQ scheme of Fig. 9. We require the obvious correctness condition. We also require the Sigma Protocol [11] extractability condition, which says there is an algorithm $\mathsf{ID.Ex}$ such that if $Y_1 \| c_1 \| z_1, Y_2 \| c_2 \| z_2$ are accepting transcripts under ivk with $Y_1 = Y_2$ but $c_1 \neq c_2$ then $\mathsf{ID.Ex}$ given ivk and the transcripts returns isk. Formally we measure extractability via the advantage $\mathbf{Adv}^{\mathrm{ex}}_{\mathsf{ID}}(\mathcal{A}) = \Pr[\mathrm{EX}^{\mathcal{A}}_{\mathsf{ID}}]$ associated to an adversary $\mathcal{A}$ where the game is in Fig. 12.

The auxiliary information tk is not used in a basic ID scheme. We use it when we say what it means for the scheme to be *trapdoor*. Namely there is an algorithm $\mathsf{ID.Cmt}^{-1}$ that produces y from Y with the aid of the trapdoor tk. Formally, the outputs of the following two processes are identically distributed. Both processes generate $(isk, ivk, tk) \leftarrow\$ \mathsf{ID.Kg}$. The first process then lets $(Y, y) \leftarrow\$ \mathsf{ID.Cmt}(ivk)$. The second process picks $Y \leftarrow\$ \mathsf{ID.CmtSp}(ivk)$ and lets $y \leftarrow\$ \mathsf{ID.Cmt}^{-1}(ivk, tk, Y)$. Both processes return (isk, ivk, tk, Y, y). Here $\mathsf{ID.CmtSp}(ivk)$ is a space of commitments associated to ID. We let $\mathsf{ID.tl}$ denote the length of tk.

Game $\mathrm{mIMP}^{\mathcal{P}}_{\mathsf{ID}}$	$\underline{\textsc{Ch}(Y)}$
$(ivk, isk, tk) \leftarrow\!\!\$\ \mathsf{ID.Kg}$; $i \leftarrow 0$	$i \leftarrow i+1$; $U \leftarrow U \cup \{i\}$; $c \leftarrow\!\!\$\ \{0,1\}^{\mathsf{ID.cl}}$
$d \leftarrow\!\!\$\ \mathcal{P}^{\textsc{Tr},\textsc{Ch},\textsc{Dec}}(ivk)$	$\mathrm{TT}[i] \leftarrow Y \| c$; Return (i, c)
Return win	$\underline{\textsc{Dec}(j, z)}$
$\underline{\textsc{Tr}()}$	If $(j \notin U)$ then return $\perp$
$(Y, y) \leftarrow\!\!\$\ \mathsf{ID.Cmt}(ivk)$; $c \leftarrow\!\!\$\ \{0,1\}^{\mathsf{ID.cl}}$	$U \leftarrow U \setminus \{j\}$; $\mathrm{TT}[j] \leftarrow \mathrm{TT}[j] \| z$
$z \leftarrow \mathsf{ID.Rsp}(ivk, isk, c, y)$	$\mathrm{DT}[j] \leftarrow \mathsf{ID.Vf}(ivk, \mathrm{TT}[j])$
Return $Y \| c \| z$	$\mathsf{win} \leftarrow \mathsf{win} \vee \mathrm{DT}[j]$; Return $\mathrm{DT}[j]$

Figure 3: Game defining security of identification scheme ID against multi-impersonation under passive attack.

MIMP SECURITY OF ID SCHEMES. The first proofs of unforgeability of Fiat-Shamir signatures —by this we mean signatures derived from ID schemes via the Fiat-Shamir transform [13]— are due to Pointcheval and Stern [26] and used their forking lemma. More general versions of the forking lemma followed [5, 4]. OO [23] and AABN [1] provide an alternative and more modular approach, the latter proving unforgeability assuming security of the identification scheme against impersonation under passive attack (imp). The latter can then be established, separately, via the reset lemma of [6]. We will extend this approach. We will define and use a new property of the ID scheme that we call security against *multiple* impersonations under passive attack (mimp). The gains —compared to using the forking lemma— are a simpler and more modular proof and better concrete security.

Recall that security of an identification scheme ID under impersonation [12, 6] considers an adversary who, given *ivk* but not *isk*, first attacks the honest, *isk*-using prover and then, using the information it gathers, attempts to impersonate the real prover by successfully identifying itself to the verifier. In this impersonation attempt, the adversary, in the role of malicious prover, submits a commitment Y of its choice, receives an honest verifier challenge c, submits a response z of its choice, and wins if $\mathsf{ID.Vf}(ivk, Y\|cz) = \mathsf{true}$. A hierarchy of possible first-phase attacks is defined in [6], but we will require security only against the weakest, namely passive attacks, where the adversary is just an eavesdropper and gets honestly-generated protocol transcripts. (Stronger active and even concurrent attacks are relevant in other contexts [6].)

However, this classic notion of security [13, 12, 6] allows only one impersonation attempt. Our mimp notion allows multiple attempts. The formalization considers game $\mathrm{mIMP}^{\mathcal{P}}_{\mathsf{ID}}$ of Fig. 3 associated to identification scheme ID and mimp adversary $\mathcal{P}$. The transcript oracle $\textsc{Tr}$ returns upon each invocation a transcript of an interaction between the honest prover and verifier, allowing $\mathcal{P}$ to mount its passive attack. Adversary $\mathcal{P}$ can mount an impersonation attempt through its $\textsc{Ch}$ and $\textsc{Dec}$ oracles, winning if any attempt is successful. The integer i denotes a session id, unique for each impersonation attempt. We let $\mathbf{Adv}^{\mathrm{mimp}}_{\mathsf{ID}}(\mathcal{P}) = \Pr[\mathrm{mIMP}^{\mathcal{P}}_{\mathsf{ID}}]$.

We show in Theorem 3 that mimp security is implied by standard single-impersonation security under passive attack (imp) with a loss in the advantage of a factor equal to the number of impersonation attempts. Together with Theorem 4 this implies mimp security can be established under standard assumptions, for example one-wayness of RSA for $\mathsf{GQ\text{-}ID}$. This means that in assuming mimp we are not incurring extra assumptions. But these reductions, like that of the forking lemma, are not tight. And cryptanalysis indicates that these factors are not real, but artifacts of the proofs. But our reduction of Theorem 2 to mimp security is *tight*. To avoid artificial inflation of security parameters, and corresponding loss in efficiency, when we instantiate and implement our schemes in Section 5, we pick parameters based on direct, cryptanalytic estimates of mimp security rather

$\underline{\mathsf{DS.Kg}^{\mathrm{H}}}$
$(ivk, isk, tk) \leftarrow\!\!\$\ \mathsf{ID.Kg}$
$TK \leftarrow tk \oplus \mathrm{H}(isk, \{0,1\}^{\mathsf{ID.tl}})$
$vk \leftarrow (ivk, TK)$; $sk \leftarrow (isk, tk)$
Return (vk, sk)

$\underline{\mathsf{DS.Ex}^{\mathrm{H}}(vk, m_1, m_2, \sigma_1, \sigma_2)}$
$(ivk, TK) \leftarrow vk$
For $i = 1, 2$ do
  $(a_i, p_i) \leftarrow m_i$; $(z_i, s_i) \leftarrow \sigma_i$
  $Y_i \leftarrow \mathrm{H}(a_i, \mathsf{ID.CmtSp}(ivk))$
  $c_i \leftarrow \mathrm{H}(Y_i \| a_i \| p_i \| s_i, \{0,1\}^{\mathsf{ID.cl}})$
$isk^* \leftarrow \mathsf{ID.Ex}(ivk, Y_1 \| c_1 \| z_1, Y_2 \| c_2 \| z_2)$
$tk^* \leftarrow \mathrm{H}(isk^*, \{0,1\}^{\mathsf{ID.tl}}) \oplus TK$
Return (isk^*, tk^*)

$\underline{\mathsf{DS.Sig}^{\mathrm{H}}(vk, sk, m)}$
$(a, p) \leftarrow m$; $s \leftarrow\!\!\$\ \{0,1\}^{\mathsf{sl}}$
$(ivk, TK) \leftarrow vk$; $(isk, tk) \leftarrow sk$
$Y \leftarrow \mathrm{H}(a, \mathsf{ID.CmtSp}(ivk))$
$y \leftarrow\!\!\$\ \mathsf{ID.Cmt}^{-1}(ivk, tk, Y)$
$c \leftarrow \mathrm{H}(Y \| a \| p \| s, \{0,1\}^{\mathsf{ID.cl}})$
$z \leftarrow \mathsf{ID.Rsp}(ivk, isk, c, y)$
$\sigma \leftarrow (z, s)$; Return σ

$\underline{\mathsf{DS.Vf}^{\mathrm{H}}(vk, m, \sigma)}$
$(ivk, TK) \leftarrow vk$; $(a, p) \leftarrow m$; $(z, s) \leftarrow \sigma$
$Y \leftarrow \mathrm{H}(a, \mathsf{ID.CmtSp}(ivk))$
$c \leftarrow \mathrm{H}(Y \| a \| p \| s, \{0,1\}^{\mathsf{ID.cl}})$
Return $\mathsf{ID.Vf}(ivk, Y \| c \| z)$

Figure 4: Our construction of a DAPS $\mathsf{DS} = \mathbf{Tid2Daps}[\mathsf{ID}, \mathsf{sl}]$ from a trapdoor identification scheme ID and a seed length $\mathsf{sl} \in \mathbb{N}$.

than on the bounds from Appendix B. This shows the benefit of mimp as a starting point.

4 Our general DAPS construction

We show how any trapdoor identification scheme can be transformed into a DAPS. We prove both that our DAPS is double authentication preventing and unforgeable. In the next section we will instantiate this general construction to get specific, efficient DAPS.

The construction. Let ID be a trapdoor identification scheme. Our **Tid2Daps** —trapdoor identification to DAPS— transform associates to it and a seed length $\mathsf{sl} \in \mathbb{N}$ a DAPS $\mathsf{DS} = \mathbf{Tid2Daps}[\mathsf{ID}, \mathsf{sl}]$. The algorithms of DS are defined in Fig. 4. Recall that in the Fiat-Shamir transform [13], the signer picks $(Y, y) \leftarrow\!\!\$\ \mathsf{ID.Cmt}(ivk)$ as per the ID scheme and commits to these values by hashing Y with the message to create a challenge. We instead specify the commitment Y as a hash of the message address. This is done so that messages with the same address result in transcripts with the same commitment, putting us in a position to use the extractability of ID to achieve double authentication prevention. However, doing this means that it is not clear how in general to obtain y. This is where the trapdoor property comes in, allowing our signer to obtain it as $y \leftarrow \mathsf{ID.Cmt}^{-1}(ivk, tk, Y)$. We then proceed as in Fiat-Shamir, except that we need a *randomized* version of the transform as specified in [1]. The randomization is captured by the seed s whose length sl was a parameter of our transform. The introduction of the trapdoor tk however creates a new difficulty, namely that extraction under the ID scheme will only recover isk and to achieve double authentication prevention we must recover the entire secret key $sk = (isk, tk)$. We resolve this by putting in the verification key a particular encryption, denoted TK, of tk, under isk.

DAP-security of our construction. The following confirms that double authentication prevention is achieved. This is relatively straightforward given the construction; the bigger challenge will be showing unforgeability. The number of (distinct) queries q of the adversary to $\mathrm{H}(\cdot, \{0,1\}^{\mathsf{ID.cl}})$, referred to below, is, formally, the number of queries made to this oracle in the execution of the game $\mathrm{DAP}^{\mathcal{A}}_{\mathsf{DS}}$, so that queries made not directly by $\mathcal{A}$ but by game procedures are also counted. As a result it will always be the case that $q \geq 2$.

Theorem 1 *Let DAPS* $\mathsf{DS} = \mathbf{Tid2Daps}[\mathsf{ID}, \mathsf{sl}]$ *be obtained from trapdoor identification scheme* ID *and seed length* sl *as above. Let* $\mathcal{A}$ *be an adversary making* $q \geq 2$ *distinct* $\mathrm{H}(\cdot, \{0,1\}^{\mathsf{ID.cl}})$ *queries. Then we can construct an adversary* $\mathcal{A}'$ *such that* $\mathbf{Adv}^{\mathrm{dap}}_{\mathsf{DS}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{ex}}_{\mathsf{ID}}(\mathcal{A}') + q(q-1)/2^{\mathsf{ID.cl}+1}$*. The running time of* $\mathcal{A}'$ *is about the same as that of* $\mathcal{A}$*.*

Proof of of Theorem 1: Consider the $\mathrm{DAP}^{\mathcal{A}}_{\mathsf{DS}}$ game of Fig. 1. Within this, consider the execution of the algorithm $\mathsf{DS.Ex}^{\mathrm{H}}$ of Fig. 4 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 $vk = (ivk, TK)$. As per the verification algorithm $\mathsf{DS.Vf}^{\mathrm{H}}$ of Fig. 4 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 verification ensures that $Y_1 = \mathrm{H}(a_1, \mathsf{ID.CmtSp}(ivk))$ and $Y_2 = \mathrm{H}(a_2, \mathsf{ID.CmtSp}(ivk))$. So if $c_1 \neq c_2$ then the extraction property of ID ensures that $isk^* = isk$. (We assume for simplicity the Sigma-protocol extraction always succeeds since this is true for the ones we use, else a $\mathbf{Adv}^{\mathrm{ex}}_{\mathsf{ID}}(\mathcal{A}_{\mathrm{id}})$ term as defined iabove must be added to the bound for an adversary $\mathcal{A}_{\mathrm{id}}$ that we would construct here.) If so, we also have $tk^* = tk$, so that the full secret key $sk = (isk, tk)$ 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 $\mathrm{H}(\cdot, \{0,1\}^{\mathsf{ID.cl}})$, to which the game makes at most q queries. So the chance that these challenges collide is at most $q(q-1)/2^{\mathsf{ID.cl}+1}$. ▌

Unforgeability of our construction. The following shows that the unforgeability of our DAPS tightly reduces to the mimp security of the underlying ID scheme. As before, the number of queries by $\mathcal{A}$ to some oracle includes the number made in the game, and similarly the running time of an adversary is the total execution time of the game, the time used by oracles included.

Theorem 2 *Let DAPS* $\mathsf{DS} = \mathbf{Tid2Daps}[\mathsf{ID}, \mathsf{sl}]$ *be obtained from trapdoor identification scheme* ID *and seed length* sl *as above. Let* $\mathcal{A}$ *be a uf-adversary against* DS*. Suppose the number of queries that* $\mathcal{A}$ *makes to its* $\mathrm{H}(\cdot, \{0,1\}^{\mathsf{ID.tl}})$, $\mathrm{H}(\cdot, \mathsf{ID.CmtSp}(ivk))$, $\mathrm{H}(\cdot, \{0,1\}^{\mathsf{ID.cl}})$, Sign *oracles are, respectively,* q_1, q_2, q_3, q_s*, where* ivk *is as in game* $\mathrm{UF}^{\mathcal{A}}_{\mathsf{DS}}$*. Then from* $\mathcal{A}$ *we can construct mimp adversaries* $\mathcal{P}_1, \mathcal{P}_2$ *such that*

$$\begin{aligned} &\mathbf{Adv}^{\mathrm{uf}}_{\mathsf{DS}}(\mathcal{A}) \\ &\quad \leq \quad \mathbf{Adv}^{\mathrm{mimp}}_{\mathsf{ID}}(\mathcal{P}_1) + \mathbf{Adv}^{\mathrm{mimp}}_{\mathsf{ID}}(\mathcal{P}_2) + \frac{q_s(2q_3 + q_s - 1)}{2^{\mathsf{sl}+1}} \,. \end{aligned} \tag{1}$$

Adversaries $\mathcal{P}_1, \mathcal{P}_2$ *make* $q_2 + q_s + 1$ *queries to* Tr*. Adversary* $\mathcal{P}_1$ *makes* q_3 *queries to* Ch *and one query to* Dec*. Adversary* $\mathcal{P}_2$ *makes* q_1 *queries to* Ch *and* Dec*. The running time of adversaries* $\mathcal{P}_1, \mathcal{P}_2$ *is that of* $\mathcal{A}$ *plus some small overhead. In the case of* $\mathcal{P}_2$ *the overhead amounts to* q_1 *executions of the* ID *protocol.*

Proof of of Theorem 2: 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 $\mathrm{UF}^{\mathcal{A}}_{\mathsf{DS}}$, and the code and checks associated with them, are redundant and can be removed. We will work with this simplified form of the game.

When procedure Sign is replying to signing query $m = (a, p)$, it first computes Y and picks s. We would like that, at this point, it can define the table entry $\mathrm{HT}[Y \| a \| p \| s, \{0,1\}^{\mathsf{ID.cl}}]$ without caring

Game $\mathrm{G}_0/\boxed{\mathrm{G}_1}$	$\textsc{Sign}(m)$
$(ivk, isk, tk) \leftarrow\!\!\$\ \mathsf{ID.Kg}$	$(a, p) \leftarrow m\ ;\ s \leftarrow\!\!\$\ \{0,1\}^{\mathsf{sl}}$
$TK \leftarrow tk \oplus \mathrm{H}(isk, \{0,1\}^{\mathsf{ID.tl}})$	$(ivk,\ TK) \leftarrow vk\ ;\ (isk, tk) \leftarrow sk$
$vk \leftarrow (ivk,\ TK)\ ;\ sk \leftarrow (isk, tk)$	$Y \leftarrow \mathrm{H}(a, \mathsf{ID.CmtSp}(ivk))$
$(m, \sigma) \leftarrow\!\!\$\ \mathcal{A}^{\textsc{Sign},\mathrm{H}}(vk)$	$y \leftarrow\!\!\$\ \mathsf{ID.Cmt}^{-1}(ivk, tk, Y)$
Return $\mathsf{DS.Vf}^{\mathrm{H}}(vk, m, \sigma)$	If (not $\mathrm{HT}[Y\|a\|p\|s, \{0,1\}^{\mathsf{ID.cl}}]$) then
$\underline{\mathrm{H}(x, \mathrm{Rng})}$	$\quad \mathrm{HT}[Y\|a\|p\|s, \{0,1\}^{\mathsf{ID.cl}}] \leftarrow\!\!\$\ \{0,1\}^{\mathsf{ID.cl}}$
If not $\mathrm{HT}[x, \mathrm{Rng}]$ then	Else
$\quad \mathrm{HT}[x, \mathrm{Rng}] \leftarrow\!\!\$\ \mathrm{Rng}$	$\quad \mathsf{bad} \leftarrow \mathsf{true}\ ;$
Return $\mathrm{HT}[x, \mathrm{Rng}]$	$\quad \boxed{\mathrm{HT}[Y\|a\|p\|s, \{0,1\}^{\mathsf{ID.cl}}] \leftarrow\!\!\$\ \{0,1\}^{\mathsf{ID.cl}}}$
	$c \leftarrow \mathrm{HT}[Y\|a\|p\|s, \{0,1\}^{\mathsf{ID.cl}}]$
	$z \leftarrow \mathsf{ID.Rsp}(ivk, isk, c, y)$
	$\sigma \leftarrow (z, s)\ ;$ Return σ

Game $\boxed{\mathrm{G}_2}/\mathrm{G}_3$	$\textsc{Sign}(m)$
$(ivk, isk, tk) \leftarrow\!\!\$\ \mathsf{ID.Kg}$	$(a, p) \leftarrow m\ ;\ s \leftarrow\!\!\$\ \{0,1\}^{\mathsf{sl}}$
$TK \leftarrow\!\!\$\ \{0,1\}^{\mathsf{ID.tl}}$	$(ivk,\ TK) \leftarrow vk\ ;\ (isk, tk) \leftarrow sk$
$vk \leftarrow (ivk,\ TK)\ ;\ sk \leftarrow (isk, tk)$	$Y \leftarrow \mathrm{H}(a, \mathsf{ID.CmtSp}(ivk))$
$(m, \sigma) \leftarrow\!\!\$\ \mathcal{A}^{\textsc{Sign},\mathrm{H}}(vk)$	$y \leftarrow\!\!\$\ \mathsf{ID.Cmt}^{-1}(ivk, tk, Y)$
Return $\mathsf{DS.Vf}^{\mathrm{H}}(vk, m, \sigma)$	$c \leftarrow\!\!\$\ \{0,1\}^{\mathsf{ID.cl}}$
$\underline{\mathrm{H}(x, \mathrm{Rng})}$	$\mathrm{HT}[Y\|a\|p\|s, \{0,1\}^{\mathsf{ID.cl}}] \leftarrow c$
If not $\mathrm{HT}[x, \mathrm{Rng}]$ then	$z \leftarrow \mathsf{ID.Rsp}(ivk, isk, c, y)$
$\quad \mathrm{HT}[x, \mathrm{Rng}] \leftarrow\!\!\$\ \mathrm{Rng}$	$\sigma \leftarrow (z, s)\ ;$ Return σ
$\quad$ If $((\mathrm{Rng} = \{0,1\}^{\mathsf{ID.tl}}) \wedge (x = isk))$ then	
$\quad\quad \mathsf{bad} \leftarrow \mathsf{true}\ ;\ \boxed{\mathrm{HT}[x, \mathrm{Rng}] \leftarrow TK \oplus tk}$	
Return $\mathrm{HT}[x, \mathrm{Rng}]$	

Figure 5: Games for proof of Theorem 2. Games $\mathrm{G}_1, \mathrm{G}_2$ include the boxed code and games $\mathrm{G}_0, \mathrm{G}_3$ do not.

---

whether it was already defined. (This is to allow an eventual impersonation adversary to program this RO response with a challenge emanating from a transcript obtained from the transcript oracle.) In general, of course, this would be wrong, but intuitively the random choice of s means it is usually right. (This indeed is why we have the seed in the scheme.) To show this formally we consider the games $\mathrm{G}_0, \mathrm{G}_1$ of Fig. 5. Game G_0 excludes the boxed code, so that its $\textsc{Sign}$ procedure defines $\mathrm{HT}[Y\|a\|p\|s, \{0,1\}^{\mathsf{ID.cl}}]$ only when this entry was not already defined, but game G_1 includes the boxed code, so that $\textsc{Sign}$ defines this entry always, as we would like. But these games are identical-until-bad [8], meaning differ only in code that follows the setting of the boolean flag bad to true. So we have

$$\begin{aligned} \mathbf{Adv}^{\mathrm{uf}}_{\mathsf{DS}}(\mathcal{A}) = \Pr[\mathrm{G}_0] &= \Pr[\mathrm{G}_1] + \Pr[\mathrm{G}_0] - \Pr[\mathrm{G}_1] \\ &\le \Pr[\mathrm{G}_1] + \Pr[\mathrm{G}_0 \text{ sets } \mathsf{bad}]\ , \end{aligned} \tag{2}$$

where the inequality is by the Fundamental Lemma of Game Playing of [8]. The random choice of s made by procedure $\textsc{Sign}$ ensures

$$\Pr[\mathrm{G}_0 \text{ sets } \mathsf{bad}] \le \sum_{i=0}^{q_s-1} \frac{q_3 + i}{2^{\mathsf{sl}}} = \frac{q_s(2q_3 + q_s - 1)}{2^{\mathsf{sl}+1}}\ . \tag{3}$$

Now we need to bound $\Pr[\mathrm{G}_1]$. We start by considering whether the ciphertext $TK \leftarrow tk \oplus$

Game G_4

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

Game G_5

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

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

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

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

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

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

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

Figure 6: More games for the proof of Theorem 2.

$\mathrm{H}(isk, \{0,1\}^{\mathsf{ID.tl}})$ helps $\mathcal{A}$ over and above access to $\textsc{Sign}$. Consider the games $\mathrm{G}_2, \mathrm{G}_3$ of Fig. 5. They pick TK directly at random rather than as prescribed in the scheme. However, via the boxed code that it contains, game G_2 compensates, replying to $\mathrm{H}(\cdot, \{0,1\}^{\mathsf{ID.tl}})$ queries in such a way that $TK = tk \oplus \mathrm{H}(isk, \{0,1\}^{\mathsf{ID.tl}})$. 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

$$\begin{aligned}\Pr[\mathrm{G}_1] = \Pr[\mathrm{G}_2] &= \Pr[\mathrm{G}_3] + \Pr[\mathrm{G}_2] - \Pr[\mathrm{G}_3] \\ &\le \Pr[\mathrm{G}_3] + \Pr[\mathrm{G}_3 \text{ sets } \mathsf{bad}] \ , \end{aligned} \tag{4}$$

where again the inequality is by the Fundamental Lemma of Game Playing of [8]. 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 that $\mathcal{A}$ cannot forge if the ciphertext TK is random, and the second corresponds to showing that changing the ciphertext to random makes little difference. Both bounds will reply on the assumed mimp security of ID.

To bound $\Pr[\mathrm{G}_3]$, consider game G_4 of Fig. 6. Towards using mimp, this game refrains from using isk directly in procedure $\textsc{Sign}$. Instead, it begins by generating conversation transcripts $Y_i\|c_i\|z_i$ and has $\textsc{Sign}$ use these. To make this possible, $\mathrm{H}(\cdot, \mathsf{ID.CmtSp}(ivk))$ values are set to the transcript commitments. Then $\textsc{Sign}$ retrieves the corresponding commitment Y, sets $\mathrm{HT}[Y\|a\|p\|s, \{0,1\}^{\mathsf{ID.cl}}]$ to the challenge from the same transcript, and puts the corresponding response in the signature. Since the signatures are correctly distributed we have

$$\Pr[\mathrm{G}_3] = \Pr[\mathrm{G}_4] \ . \tag{5}$$

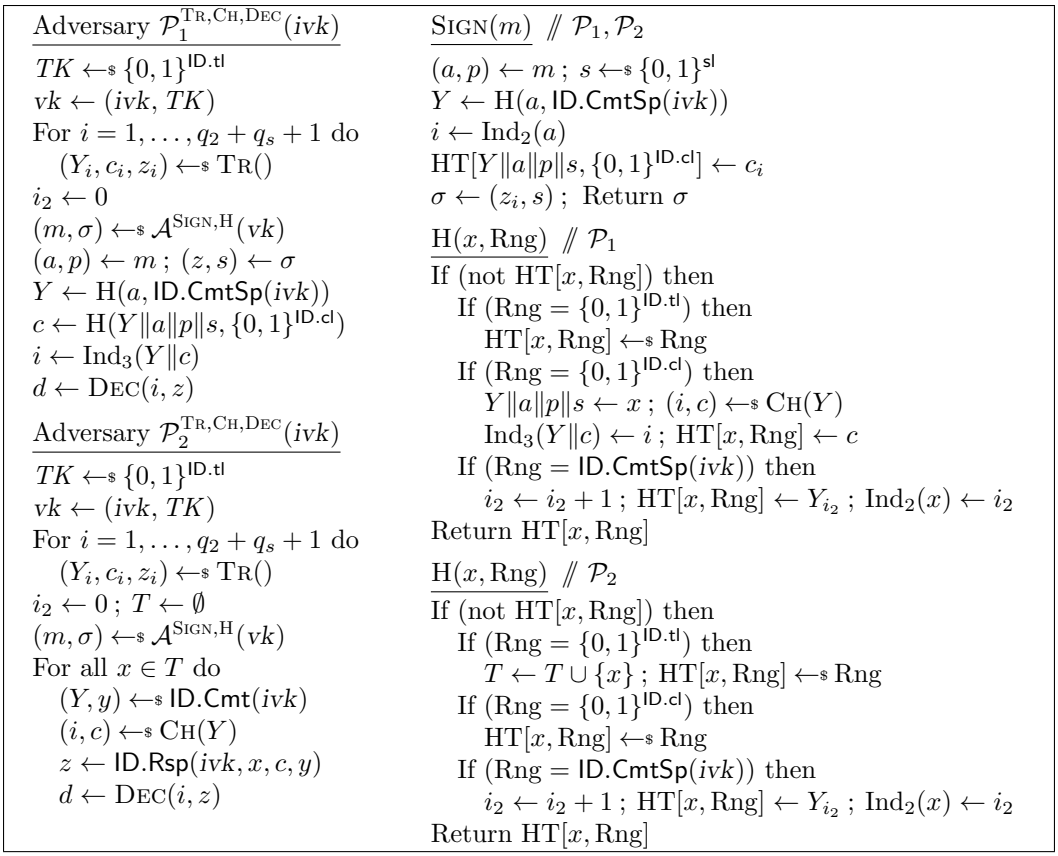


Figure 7: Adversaries for proof of Theorem 2.

We build mimp adversary $\mathcal{P}_1$ so that

$$\Pr[\text{G}_4] \leq \mathbf{Adv}^{\text{mimp}}_{\mathsf{ID}}(\mathcal{P}_1) \; . \tag{6}$$

Game G_4 was crafted exactly to make the construction of adversary $\mathcal{P}_1$ quite direct. The construction is described in detail in Fig. 7. Adversary $\mathcal{P}_1$ has access to oracles $\text{Tr}, \text{Ch}, \text{Dec}$ as per game $\text{mIMP}^{\mathcal{P}_1}_{\mathsf{ID}}$ in which it is executing. It runs $\mathcal{A}$, simulating answers to $\mathcal{A}$'s queries to Sign and H as shown. It obtains conversation transcripts using its Tr oracle to play the role of the ones generated in G_4. Using these, Sign can be simulated as per game G_4. Oracle $\text{H}(\cdot, \text{Rng})$ is simulated as in G_4 when $\text{Rng} = \{0,1\}^{\mathsf{ID.tl}}$ or $\text{Rng} = \mathsf{ID.CmtSp}(ivk)$. When a query x is made to $\text{H}(\cdot, \{0,1\}^{\mathsf{ID.cl}})$, adversary $\mathcal{P}_1$ parses x as $Y\|a\|p\|s$, sends Y 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 session id corresponding to the commitment and challenge in the forgery is retrieved via Ind_3. Now this session is completed by querying the response in the forged signature to the decision oracle Dec. We need to show that the impersonation is successful as long as the forgery was valid. A somewhat delicate point is that we use the fact that the message m in the forgery was not a Sign query. This is what ensures that a session corresponding to the forgery conversation exists.

To bound $\Pr[\text{G}_3 \text{ sets } \mathsf{bad}]$, consider game G_5 of Fig. 6. It answers Sign queries just like G_4, and the only modification in answering H queries is to keep track of queries to $\text{H}(\cdot, \{0,1\}^{\mathsf{ID.tl}})$ in the set T. The game ignores the forgery, returning true if isk was queried to $\text{H}(\cdot, \{0,1\}^{\mathsf{ID.tl}})$. We have

$$\Pr[\text{G}_3 \text{ sets } \mathsf{bad}] = \Pr[\text{G}_5] \; . \tag{7}$$

Game $\mathrm{OW}^{\mathcal{A}}_{\mathsf{RSA}}$	Game $\mathrm{CF}^{\mathcal{A}}_{\mathsf{CFTDF}}$	Game $\mathrm{FAC}^{\mathcal{A}}_{\mathsf{MOD}}$
$(N,p,q,e,d) \leftarrow\!\!\$\ \mathsf{RSA}$	$(f_0, f_1, f_0^{-1}, f_1^{-1}, D) \leftarrow\!\!\$\ \mathsf{CFTDF}$	$(N,p,q) \leftarrow\!\!\$\ \mathsf{MOD}$
$x \leftarrow\!\!\$\ \mathbb{Z}_N^*$; $X \leftarrow x^e \bmod N$	$(x_0, x_1) \leftarrow\!\!\$\ \mathcal{A}(f_0, f_1, D)$	$r \leftarrow\!\!\$\ \mathcal{A}(N)$
$x' \leftarrow\!\!\$\ \mathcal{A}(N, e, X)$	$y_0 \leftarrow f_0(x_0)$; $y_1 \leftarrow f_1(x_1)$	Return $(r \in \{p, q\})$
Return $(x' = x)$	Return $(y_0 = y_1)$	

Figure 8: Games defining one-wayness of RSA generator RSA, claw-freeness of claw-free TDF generator CFTDF and factoring security of modulus generator MOD.

We build $\mathcal{P}_2$ so that

$$\Pr[\mathrm{G}_5] \leq \mathbf{Adv}^{\mathrm{mimp}}_{\mathsf{ID}}(\mathcal{P}_2)\ . \tag{8}$$

The idea is simple, namely that if the adversary queries *isk* to $\mathrm{H}(\cdot, \{0,1\}^{\mathsf{ID.tl}})$ then we can obtain *isk* by watching the oracle queries of $\mathcal{A}$, and this will allow us to break the mimp security of ID. 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 $\mathcal{P}_2$, described in detail Fig. 7, quite direct. 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 $\mathcal{P}_2$ now makes an impersonation attempt for each $x \in T$ in which it runs the prover using x as the identification key. In the case $x = isk$, the impersonation succeeds. ▮

Necessity of trapdoor ID schemes for DAPS. Trapdoor identification may seem a very particular assumption as a starting point for DAPS. However in Appendix C we show that from any DAPS satisfying double-authentication-prevention and unforgeability we can build a simple trapdoor identification scheme satisfying mimp-security and Sigma-protocol extractability. These being exactly the assumptions for our transform, it shows that these sufficient assumptions are in fact also necessary. The link between trapdoor identification and DAPS is thus quite strong.

5 Instantiation and mplementation

We instantiate our general transform of Section 4 to obtain GQ-DAPS and CF-DAPS. We then make parameter choices and discuss our implementation and performance results.

GQ-DAPS. An RSA generator with 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. 8.

Fig. 9 shows the GQ-ID associated to RSA and a challenge length $l < k$. The commitment space is $\mathbb{Z}^*_N$. We claim that this scheme is trapdoor. The $\mathsf{GQ\text{-}ID.Cmt}^{-1}$ algorithm, on input $((N,e,X),d,Y)$, returns $y \leftarrow Y^d \bmod N$. This means we can apply our transform. The resulting GQ-DAPS is shown at the bottom of Fig. 9. It is parameterized by RSA (and thus k), the challenge length $l < k$ and a seed length sl. By egcd we denote the extended gcd algorithm that given relatively-prime inputs e,c returns a,b such that $ae + bc = 1$.

To estimate security for a given modulus length k we use Theorem 2 and estimate that a time

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

$\underline{\mathsf{GQ\text{-}DAPS.Kg}^{\mathrm{H}}}$
$((N, e, X), x, d) \leftarrow_{\$} \mathsf{GQ\text{-}ID.Kg}$
$TK \leftarrow d \oplus \mathrm{H}(x, \{0,1\}^k)$
Return $((N, e, X, TK), (x, d))$

$\underline{\mathsf{GQ\text{-}DAPS.Ex}^{\mathrm{H}}((N, e, X, TK), m_1, m_2, \sigma_1, \sigma_2)}$
For $i = 1, 2$ do
  $(a_i, p_i) \leftarrow m_i$; $(z_i, s_i) \leftarrow \sigma_i$
  $Y_i \leftarrow \mathrm{H}(a_i, \mathbb{Z}_N^*)$
  $c_i \leftarrow \mathrm{H}(Y_i \| a_i \| p_i \| s_i, \{0,1\}^l)$
$z \leftarrow z_1 z_2^{-1} \mod N$
$c \leftarrow c_1 - c_2$; $(a, b) \leftarrow \mathrm{egcd}(e, c)$
$x \leftarrow X^a z^b \mod N$
$d \leftarrow \mathrm{H}(x, \{0,1\}^k) \oplus TK$
Return (x, d)

$\underline{\mathsf{GQ\text{-}DAPS.Sig}^{\mathrm{H}}((N, e, X, TK), (x, d), m)}$
$(a, p) \leftarrow m$; $s \leftarrow_{\$} \{0,1\}^{\mathsf{sl}}$
$Y \leftarrow \mathrm{H}(a, \mathbb{Z}_N^*)$
$y \leftarrow_{\$} Y^d \bmod N$
$c \leftarrow \mathrm{H}(Y \| a \| p \| s, \{0,1\}^l)$
$z \leftarrow yx^c \bmod N$
$\sigma \leftarrow (z, s)$; Return σ

$\underline{\mathsf{GQ\text{-}DAPS.Vf}^{\mathrm{H}}((N, e, X, TK), m, \sigma)}$
$(a, p) \leftarrow m$; $(z, s) \leftarrow \sigma$
$Y \leftarrow \mathrm{H}(a, \mathbb{Z}_N^*)$
$c \leftarrow \mathrm{H}(Y \| a \| p \| s, \{0,1\}^l)$
Return $(z^e \equiv YX^c \pmod N)$

Figure 9: **Top:** Identification scheme GQ-ID associated to RSA generator RSA with modulus length k, and challenge length l. **Bottom:** GQ-DAPS = **Tid2Daps**[GQ-ID, sl] derived via our transform.

t mimp adversary $\mathcal{P}$ making q_c queries to Ch, Dec has advantage

$$\mathbf{Adv}^{\mathrm{mimp}}_{\mathsf{ID}}(\mathcal{P}) \leq \frac{q_c}{2^l} + \mathbf{Adv}^{\mathrm{ow}}_{\mathsf{RSA}}(\mathcal{A}) \tag{9}$$

where $\mathcal{A}$ is the best known time t adversary against the one-wayness of RSA. We can estimate $\mathbf{Adv}^{\mathrm{ow}}_{\mathsf{RSA}}(\mathcal{A})$ under the assumption that the NFS is the best factoring method. Then taking into account Equation (1), our implementation uses a 1024-bit modulus, a 160-bit hash and a seed length of 160 for the usual expected 80 bits of security. Since CA's now use 2048-bit modulii, we also implement the scheme with a 2048-bit modulus and 256-bit hashes and seeds. See below and Fig. 11 for implementation and performance information.

CF-DAPS. A claw-free TDF generator [16] is an algorithm CFTDF that returns a tuple $(f_0, f_1, f_0^{-1}, f_1^{-1}, D)$ consisting of (descriptions of) a finite set D, permutations $f_0, f_1\colon D \to D$, and their respective inverses $f_0^{-1}, f_1^{-1}\colon D \to D$. The assumption is claw-freeness, formalized by defining the cf-advantage of an adversary $\mathcal{A}$ against CFTDF by $\mathbf{Adv}^{\mathrm{cf}}_{\mathsf{CFTDF}}(\mathcal{A}) = \Pr[\mathrm{CF}^{\mathcal{A}}_{\mathsf{CFTDF}}]$ where the game is in Fig. 8.

A modulus generator with security parameter k is an algorithm MOD that returns a tuple (N, p, q) where p, q are primes satisfying $p \equiv 3 \pmod 8$ and $q \equiv 7 \pmod 8$ and $N = pq$ is the modulus, in the range $2^{k-1} < N < 2^k$. The assumption is hardness of factoring, formalized by defining the factoring-advantage of an adversary $\mathcal{A}$ against MOD by $\mathbf{Adv}^{\mathrm{fac}}_{\mathsf{MOD}}(\mathcal{A}) = \Pr[\mathrm{FAC}^{\mathcal{A}}_{\mathsf{MOD}}]$ where the game is in Fig. 8.

We associate to a modulus generator MOD the particular claw-free TDF generator CFTDF[MOD] that runs MOD to get (N, p, q) and then returns $(f_0, f_1, f_0^{-1}, f_1^{-1}, D)$ defined as follows. The domain is $D = \{z^2 \mod N : z \in \mathbb{Z}_N^*\}$. For $x \in D$ let $f_0(x) = x^2 \bmod N$ and $f_1(x) = 4x^2 \bmod N$. Note that, since $p \equiv 3 \pmod 8$ and $q \equiv 7 \pmod 8$, we have that neither ± 2 is a quadratic residue mod

N. For $y \in D$ the inverses are defined as $f_0^{-1}(y) = \sqrt{y} \mod N$ and $f_1^{-1}(y) = \sqrt{4^{-1}y} \mod N$ where $\sqrt{z}$ denotes the square root of z that is itself a quadratic residue mod N. Note that we cannot efficiently check membership in D given N.

For a binary string $c = c[1] \dots c[n] \in \{0,1\}^n$, let $f_c\, f_c^{-1}\colon\ D \to D$ be defined for $x, y \in D$ by

$$\begin{aligned} f_c(x) &= f_{c[1]}(\cdots(f_{c[n]}(x))\cdots) \\ f_c^{-1}(y) &= f_{c[n]}^{-1}(\cdots(f_{c[1]}^{-1}(y))\cdots)\ . \end{aligned}$$

In Fig. 10 we show an identification scheme we call CF-ID. It is associated to CFTDF and a challenge length $l < k$. The commitment space is D. When CFTDF = CFTDF[MOD], this is an identification scheme mentioned in [22] as underlying the MSA signature scheme [21]. Following the latter, the prefixing of the challenge with a 0 bit is to ensure that, in this case, $f_0(z) \in D$. This scheme is trivially trapdoor: CF-ID.Cmt^{-1} is just the identity function. This means we can apply our transform. The resulting CF-DAPS is shown at the bottom of Fig. 10. It is parameterized by CFTDF[MOD] (and thus k), the challenge length $l < k$ and a seed length sl. In the CF-DAPS.Ex algorithm, we reference the claw-free TDF extraction algorithm CFTDF.Ex. This algorithm takes f_0, f_1, x_0, x_1 such that $x_0, x_1 \in D$ and $f_0(x_0) = f_1(x_1)$, and computes f_0^{-1} and f_1^{-1}. For CFTDF[MOD], this is done as follows. We have $x_0^2 \equiv 4x_1^2 \mod N$ and thus $r = \gcd(x_0 - 2x_1, N)$ divides N. However $x_0, x_1 \in D$ and hence $x_0 \not\equiv \pm 2x_1 \pmod N$, so r is a non-trivial factor of N. For the parameter choices for implementations, we use the same estimates for a time t mimp adversary in breaking CF-DAPS as in GQ-DAPS above.

Implementation. We implemented our CF-DAPS and GQ-DAPS schemes. For comparison purposes we also implemented the original PS-DAPS and used an implementation of the standard RSA PKCS#1v.5 currently used for signing certificates. Our implementation is in C, using OpenSSL's BIGNUM library for number theoretic operations. [1]

For the implementation of the claw-free TDF for CF-DAPS, we need to compute $f_c^{-1}(x)$ for a c of length l. The naive approach requires computing l square roots modulo N, which takes $O(lk^3)$ time. Instead, we use the following technique suggested by Goldreich [15] which computes $f_c^{-1}(x)$ with a constant number of exponentiations (assuming a small amount of pre-computation which can be reused), thereby achieving an overall runtime of $O(k^3)$. Compute

$$f_c^{-1}(x) = \frac{R_N(2^l, x)}{(R_N(2^l, 4))^{i(c)}} \mod N$$

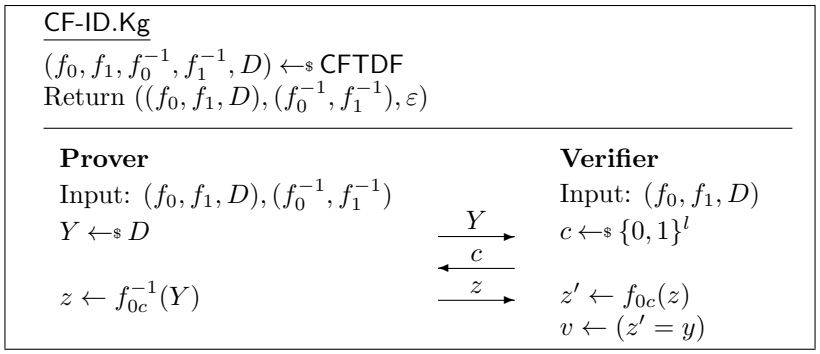
where $R_N(2^l, x)$ denotes the 2^l-th square root of x modulo N, l is the bit-length of c, and $i(c)$ denotes the integer representation of c. $R_N(2^l, x)$ can be computed quickly by computing $R_p(2^l, x)$ and $R_q(2^l, x)$ and using the Chinese remainder theorem. $R_p(2^l, x)$ can be computed by precomputing $a = (p+1)/4$ (the "inverse" of 2 modulo $\varphi(p)$) and $b = a^l \mod \varphi(p)$ (the "inverse" of 2^l modulo $\varphi(p)$), and then computing $R_p(2^l, x)$ as $x^b \mod p$.

To hash onto quadratic residues we follow the framework of Brier et al. for indifferentiable hashing [9] as described by Poettering and Stebila [25]: we first hash onto $\mathbb{Z}_N$ to obtain an element r. With high probability, randomly chosen elements of $\mathbb{Z}_N$ are also in $\mathbb{Z}_N^*$. If r has Jacobi symbol -1, we set $r \leftarrow rt \mod N$ where t is a fixed element with Jacobi symbol -1, in our case $t = 2$ always suffices. Exactly one of r and $N - r$ will be a quadratic residue mod N.

For the implementation of GQ-DAPS, we use encryption exponent $e = 65537$ as this is the default RSA public key exponent in OpenSSL, allowing for fair comparisons with RSA PKCS#1v1.5.

Performance experiments. Timings were run on an Intel Core i7 (3720QM) with 4 cores each

[1] The implementation source code can be downloaded from the anonymous URL `https://173.203.208.70:54242/npfTVfFK/src.zip`.



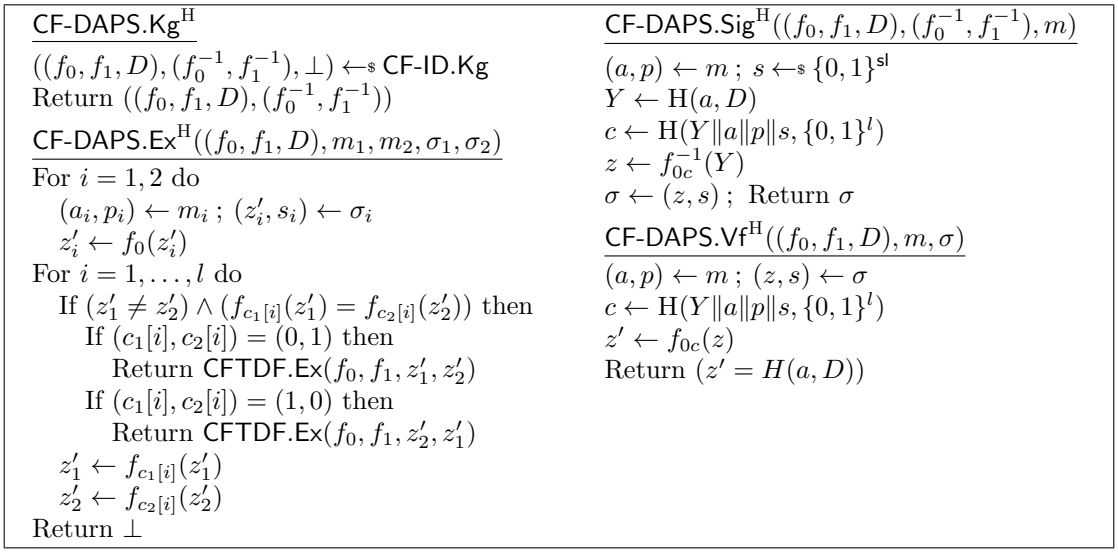


Figure 10: **Top:** Identification scheme CF-ID associated to claw-free TDF generator CFTDF and challenge length l. **Bottom:** CF-DAPS = **Tid2Daps**[CF-ID, sl] derived via our transform.

running at 2.6 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` 6.0 (`clang` 600.0.56). The OpenSSL version used was v1.0.2.

Table 11 shows average runtimes and key sizes using 1024-bit modulii and 160-bit hashes and 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. For RSA sign and verify operations, standard deviation was between 3% and 44%. For all other operations, standard deviation was less than 4%.

The table omits runtimes for key generation, as this is a one-time operation. Key generation times are fairly similar across schemes, as for all schemes the main cost is the generation of an RSA modulus. For all schemes with 1024-bit keys, key generation times, from the top row to the bottom row, are 29.9ms, 24.2ms, 31.5ms, and 23.7ms;; with 2048-bit keys, generation times are 156.5ms, 135.6ms, 167.8ms, and 125.5ms. For all key generation operations, standard deviation was between 64% and 74% (this is to be expected, as key generation involves generating primes, a probabilistic process with high variance in runtime). While key generation is substantially more expensive than signing or verification, it is still less than a second, and each signer needs to do it only once.

Compared with the existing PS-DAPS, our CF-DAPS and GQ-DAPS are several orders of mag-

Scheme	*1024-bit modulus, 160-bit hash*				*2048-bit modulus, 256-bit hash*			
	Runtime (ms)		**Size** (bits)		**Runtime** (ms)		**Size** (bits)	
	sign	**verify**	**pub.**	**sig.**	**sign**	**verify**	**pub.**	**sig.**
PS-DAPS [25]	208.30	71.33	1024	164864	1009.88	271.36	2048	528384
GQ-DAPS (Fig. 9)	0.76	0.15	2048	1184	5.10	0.68	4096	2304
CF-DAPS (Fig. 10)	1.26	1.00	1024	1184	3.00	2.34	2048	2304
RSA PKCS#1v1.5	0.21	0.02	1024	1024	1.32	0.05	2048	2048

Figure 11: Average runtime in milliseconds and public key/signature sizes for double-authentication preventing signatures and standard RSA signatures. Secret key sizes are the same as the modulus size for all schemes.

nitude faster for both signing and verification. When using 2048-bit modulii, CF-DAPS signatures can be generated 336× and verified 116× faster, and GQ-DAPS signatures can be generated 198× and verified 399× faster; moreover our signatures are much smaller, both just 2304 bits, compared with 528384 bits for PS-DAPS, and nearly the same size as RSA PKCS#1v1.5 signatures. Signing times for our schemes are competitive with RSA PKCS#1v1.5 signatures. Using CF-DAPS or GQ-DAPS for signatures in digital certificates would incur little computational or size overhead relative to currently used signatures.

References

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

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

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

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

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

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

[7] 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.

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

[9] E. Brier, J.-S. Coron, T. Icart, D. Madore, H. Randriam, and M. Tibouchi. Efficient indifferentiable hashing into ordinary elliptic curves. In T. Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 237–254. Springer, Heidelberg, Aug. 2010.

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

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

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

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

[14] M. Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In V. Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 152–168. Springer, Heidelberg, Aug. 2005.

[15] 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, Heidelberg, Aug. 1987.

[16] 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.

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

[18] 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.

[19] V. Lyubashevsky. Lattice-based identification schemes secure under active attacks. In R. Cramer, editor, *PKC 2008*, volume 4939 of *LNCS*, pages 162–179. Springer, Heidelberg, Mar. 2008.

[20] V. Lyubashevsky. Lattice signatures without trapdoors. In D. Pointcheval and T. Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 738–755. Springer, Heidelberg, Apr. 2012.

[21] S. Micali. A secure and efficient digital signature algorithm. Technical Memo MIT/LCS/TM-501b, Massachusetts Institute of Technology, Laboratory for Computer Science, Apr. 1994.

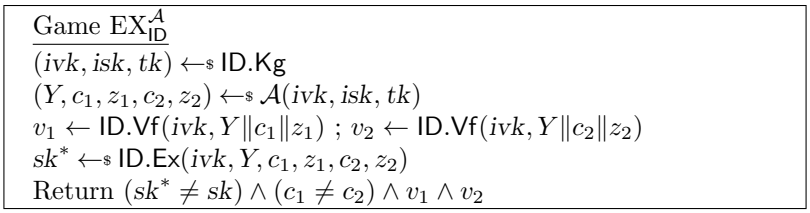


Figure 12: Game defining extractability of identification scheme ID.

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

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

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

[25] 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, Heidelberg, Sept. 2014.

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

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

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

[29] 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 Identification scheme extractability

The extractability property for identification schemes is defined as follows. If ID is an identification scheme and $\mathcal{A}$ an adversary then we let $\mathbf{Adv}^{\mathrm{ex}}_{\mathsf{ID}}(\mathcal{A}) = \Pr[\mathrm{EX}^{\mathcal{A}}_{\mathsf{ID}}]$ where the game is in Fig. 12. This measures the ability of an adversary (who knows the secret key) to create two accepting transcripts with the same commitment but different challenges which cannot be used by the extraction algorithm to recover the secret key. This is sometimes called special soundness for a Sigma protocol [14].

B Mimp from one wayness

We establish mimp security of an identification scheme based on the one-wayness of the key-generation process. All proofs are omitted.

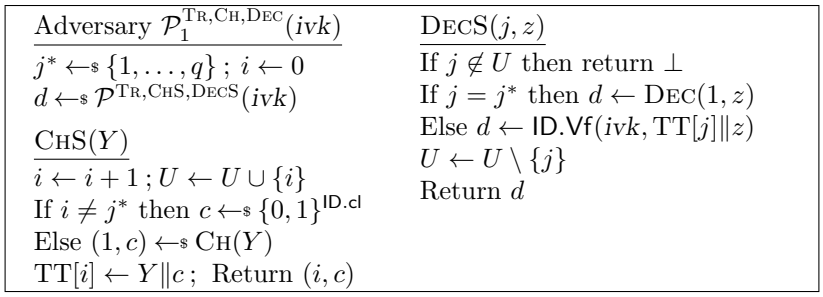


Figure 13: Adversary for proof of Theorem 3.

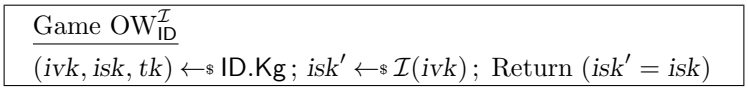


Figure 14: Game defining one-wayness of the (key-generation process of) an identification scheme ID.

MIMP SECURITY FROM IMP. In the first step we show that mimp security reduces to standard imp security with a factor in loss equal to the number of Ch, Dec queries of the adversary. We do not need to define imp security separately; it is simply mimp security for adversaries making only one query to each of their Ch, Dec oracles. The result is thus captured by the following.

Theorem 3 *Let* ID *be an identification scheme. Let* $\mathcal{P}$ *be a mimp-adversary against* ID *making* q *queries to its* Ch *oracle and* q *queries to its* Dec *oracle. Then from* $\mathcal{P}$ *we can construct mimp adversary* $\mathcal{P}_1$ *making only one query to its* Ch *oracle and only one query to its* Dec *oracle such that*

$$\mathbf{Adv}^{\text{mimp}}_{\mathsf{ID}}(\mathcal{P}) \leq q \cdot \mathbf{Adv}^{\text{mimp}}_{\mathsf{ID}}(\mathcal{P}_1) \ .$$

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

Proof: Adversary $\mathcal{P}_1$ is shown in Fig. 13. It has access to oracles Tr, Ch, Dec as per game $\text{mIMP}^{\mathcal{P}_1}_{\mathsf{ID}}$ in which it is executing, but makes only make one query to each of the second and third oracles. It guesses an instance j^* uniformly from $\{1,\ldots,q\}$ and runs $\mathcal{P}$. Adversary $\mathcal{P}_1$ passes $\mathcal{P}$'s Tr queries directly to its own Tr oracle. $\mathcal{P}_1$ simulates answers to $\mathcal{P}$'s queries to its Ch, Dec oracles via the shown subroutines ChS, DecS, calling its own oracles inside these. Adversary $\mathcal{P}_1$'s simulation is perfect. Since $\mathcal{P}_1$ will guess the instance j^* which $\mathcal{P}$ successfully impersonates with probability $1/q$, adversary $\mathcal{P}_1$'s success probability is at least $1/q$ times that of $\mathcal{P}$. ▌

IMP SECURITY FROM OW. If ID is an identification scheme and $\mathcal{I}$ an adversary then we let $\mathbf{Adv}^{\text{ow}}_{\mathsf{ID}}(\mathcal{I}) = \Pr[\text{OW}^{\mathcal{I}}_{\mathsf{ID}}]$ where the game is in Fig. 14. This simply measures the one-wayness of the key-generation algorithm, meaning how hard it is to recover the secret identification key from the public verification key. For GQ-ID this is the one-wayness of the underlying RSA generator. For CF-ID it is the hardness of factoring the modulus. Now for identification schemes satisfying the Sigma protocol extractability and honest-verifier zero-knowledge conditions, one can use the reset lemma of [6] to show that imp security follows from this one-wayness:

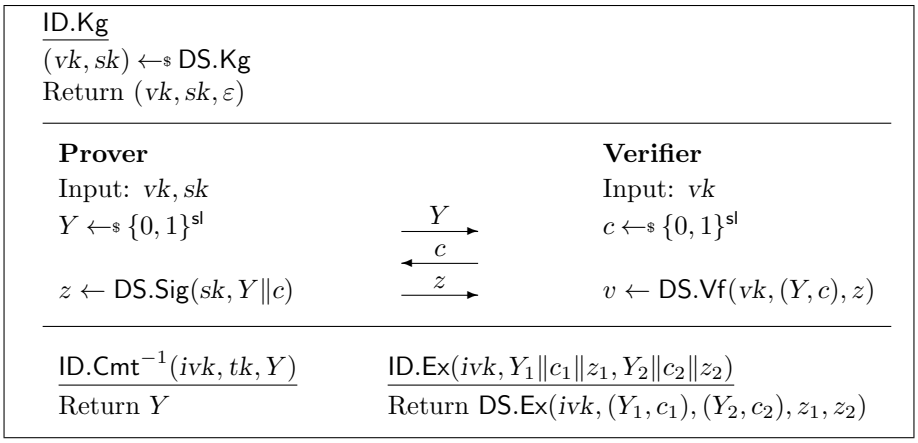


Figure 15: Our construction of a trapdoor identification scheme $\mathsf{ID} = \mathbf{Daps2Tid}[\mathsf{DS}, \mathsf{sl}]$ from a DAPS DS and a seed length $\mathsf{sl} \in \mathbb{N}$. $\mathsf{ID.CmtSp}(ivk) = \{0,1\}^{\mathsf{sl}}$ for all ivk.

Theorem 4 *Let* ID *be an identification scheme satisfying the Sigma protocol extractability and honest-verifier zero-knowledge conditions. Let* $\mathcal{P}$ *be a mimp-adversary against* ID *making one query to its* Ch *oracle and one query to its* Dec *oracle. Then from* $\mathcal{P}$ *we can construct an adversary* $\mathcal{I}$ *such that*

$$\mathbf{Adv}^{\mathrm{mimp}}_{\mathsf{ID}}(\mathcal{P}) \leq \frac{1}{2^{\mathsf{ID.cl}}} + \sqrt{\mathbf{Adv}^{\mathrm{ow}}_{\mathsf{ID}}(\mathcal{I})} \tag{10}$$

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

Combing this with Theorem 3 we have a proof of the mimp security of an identification scheme based on the one-wayness of its key-generation process. In particular, this proves mimp-security of GQ-ID based on the one-wayness of RSA and mimp-security of CF-ID based on the hardness of factoring.

This establishes that mimp security of the identification schemes, and thus unforgeability of our DAPSs, can be based on standard assumptions. However, the reduction emanating from the combination of Theorems 4 and 3 is not tight. Furthermore, cryptanalytic information says that this lack of tightness does not reflect real security losses but is rather an artifact of the proofs. Accordingly, we do not use these formulas to pick parameter sizes in Section 5. Instead we cryptanalytically and directly estimate mimp security and then use Theorem 2.

C From DAPS to trapdoor ID

Here we show that DAPS implies trapdoor identification. Given any DAPS satisfying double-authentication-prevention and unforgeability, we build a trapdoor identification scheme, via the construction **Daps2Tid** in Fig. 15, that is mimp-secure and satisfies the Sigma protocol extractability condition. This shows that the assumption we make to obtain DAPS is effectively necessary. All proofs are omitted.

The basic idea of the construction is as follows. The ID scheme's keys are just the keys of a DAPS. A commitment is a random string, as is a challenge; the response is generated as a DAPS signature with the commitment as the address and challenge as the payload. Verification in the ID scheme is just verification in the DAPS. The ID scheme is trapdoor because the commitment

Adversary $\mathcal{A}_1^{\text{Sign}}(vk)$	$\underline{\text{ChS}(Y)}$
$d \leftarrow\!\!\$\ \mathcal{A}^{\text{TrS},\text{ChS},\text{DecS}}(vk)$	$i \leftarrow i+1\ ;\ U \leftarrow U \cup \{i\}\ ;\ c \leftarrow\!\!\$\ \{0,1\}^{\mathsf{ID.cl}}$
$\underline{\text{TrS}()}$	$\text{TT}[i] \leftarrow Y \| c\ ;$
$Y \leftarrow\!\!\$\ \mathsf{ID.CmtSp}(vk)$	Return (i, c)
$c \leftarrow\!\!\$\ \{0,1\}^{\mathsf{sl}}$	$\underline{\text{DecS}(j, z)}$
$z \leftarrow\!\!\$\ \text{Sign}((Y, c))$	If $(j \notin U)$ then return $\bot$
Return $Y \| c \| z$	$U \leftarrow U \setminus \{j\}\ ;\ Y \| c \leftarrow \text{TT}[j]$
	$v \leftarrow \mathsf{DS.Vf}(vk, (Y, c), z)$
	If v then $\mathcal{A}_1$ returns $((Y, c), z)$ to its uf-challenger
	Return v (to $\mathcal{A}$)

Figure 16: Adversary for proof of Theorem 6.

"secret" is just the commitment itself, and the extractability of the Sigma protocol comes from the double-signature extractability of the DAPS.

We now make and prove three claims about the identification scheme: (1) it is trapdoor (2) It is mimp-secure, and (3) It satisfies Sigma-protocol extractability as defined in Appendix A. These are exactly the properties assumed of the identification scheme for our transform to work, so that our result here shows that the sufficient assumptions we make in Section 4 on the identification scheme to obtain DAPS are in fact also necessary.

Theorem 5 *Let* DS *be a DAPS and let* $\mathsf{sl} \in \mathbb{N}$*. Then* $\mathsf{ID} = \mathbf{Daps2Tid}[\mathsf{DS}, \mathsf{sl}]$ *is trapdoor.*

Proof: Recall that for an ID scheme to be trapdoor, the following two processes must be identically distributed:

1. $(isk, ivk, tk) \leftarrow\!\!\$\ \mathsf{ID.Kg}\ ;\ (Y, y) \leftarrow\!\!\$\ \mathsf{ID.Cmt}(ivk)\ ;$ Return (isk, ivk, tk, Y, y).
2. $(isk, ivk, tk) \leftarrow\!\!\$\ \mathsf{ID.Kg}\ ;\ Y \leftarrow\!\!\$\ \mathsf{ID.CmtSp}(ivk)\ ;\ y \leftarrow\!\!\$\ \mathsf{ID.Cmt}^{-1}(ivk, tk, Y)\ ;$ Return (isk, ivk, tk, Y, y).

For $\mathsf{ID} = \mathbf{Daps2Tid}[\mathsf{DS}, \mathsf{sl}]$, since $\mathsf{ID.Cmt}(ivk)$ simply selects $Y \leftarrow\!\!\$\ \mathsf{ID.CmtSp}(ivk)$ and $Y = y$, we have that both processes above are equivalent to:

$$(isk, ivk, tk) \leftarrow\!\!\$\ \mathsf{ID.Kg}\ ;\ Y \leftarrow\!\!\$\ \mathsf{ID.CmtSp}(ivk)\ ;\ \text{Return}\ (isk, ivk, tk, Y, Y)$$

This completes the proof. ∎

Next we show that our constructed identification scheme is mimp secure.

Theorem 6 *Let* DS *be a DAPS and let* $\mathsf{sl} \in \mathbb{N}$*. Let* $\mathcal{A}$ *be a mimp-adversary against* $\mathsf{ID} = \mathbf{Daps2Tid}[\mathsf{DS}, \mathsf{sl}]$ *making* q *queries to its* Tr *oracle. Then from* $\mathcal{A}$ *we can construct uf-adversary* $\mathcal{A}_1$ *such that* $\mathbf{Adv}^{\text{mimp}}_{\mathsf{ID}}(\mathcal{A}) \le \mathbf{Adv}^{\text{uf}}_{\mathsf{DS}}(\mathcal{A}_1)$*.* $\mathcal{A}_1$ *makes* q *queries to its* Sign *oracle and the running time of* $\mathcal{A}_1$ *is that of* $\mathcal{A}$ *plus some small overhead, including one execution of* $\mathsf{DS.Vf}$ *for each call by* $\mathcal{A}$ *to its* Dec *oracle.*

Proof: Adversary $\mathcal{A}_1$ is shown in Fig. 16. $\mathcal{A}_1$ directly simulates the mimp experiment for $\mathcal{A}$; to create transcripts, $\mathcal{A}_1$ uses its Sign oracle. If $\mathcal{A}$ submits an accepting transcript to its DecS oracle, this immediately gives $\mathcal{A}_1$ a forgery for DS. $\mathcal{A}_1$'s simulation of game $\text{UF}^{\mathcal{A}}_{\mathsf{DS}}$ is perfect. The bound follows. ∎

Finally we show that our constructed identification scheme satisfies Sigma-protocol extractability.

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 17: Adversary for proof of Theorem 7.

Game $\mathrm{DPRF}_{\mathsf{F}}^{\mathcal{A}}$	Game $\mathrm{EX}_{\mathsf{F}}^{\mathcal{A}}$
$(s, St) \leftarrow\!\!\$\ \mathcal{A}$	$(s, St) \leftarrow\!\!\$\ \mathcal{A}$
$(pk, sk) \leftarrow\!\!\$\ \mathsf{F.Kg}(s)\ ;\ A \leftarrow \emptyset$	$(pk, sk) \leftarrow\!\!\$\ \mathsf{F.Kg}(s)$
$b \leftarrow\!\!\$\ \{0,1\}$	$(a, p_1, p_2) \leftarrow\!\!\$\ \mathcal{A}(pk, sk, St)$
$b' \leftarrow\!\!\$\ \mathcal{A}^{\mathrm{SMP}}(pk, St)$	$y_1 \leftarrow \mathsf{F.Ev}(sk, a, p_1, s)$
Return $(b' = b)$	$y_2 \leftarrow \mathsf{F.Ev}(sk, a, p_2, s)$
	$(sk^*, s^*) \leftarrow\!\!\$\ \mathsf{F.Ex}(pk, a, p_1, p_2, y_1, y_2)$
$\underline{\mathrm{SMP}(a, p)}$	Return $((sk^*, s^*) \neq (sk, s)) \wedge (p_1 \neq p_2)$
If $a \in A$ then return $\perp$	
$A \leftarrow A \cup \{a\}$	
$y_1 \leftarrow\!\!\$\ \mathsf{F.Ev}(sk, a, p, s)$	
$y_0 \leftarrow\!\!\$\ \{0,1\}^{\lvert y_1 \rvert}$	
Return y_b	

Figure 18: Games defining security of DPRF F.

Theorem 7 *Let* DS *be a DAPS and let* $\mathsf{sl} \in \mathbb{N}$. *Let* $\mathcal{A}$ *be a ex-adversary against* $\mathsf{ID} = \mathbf{Daps2Tid}[\mathsf{DS}, \mathsf{sl}]$. *From* $\mathcal{A}$ *we can 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. 17. $\mathcal{A}_1$ directly calls $\mathcal{A}$ which is an ex adversary against the identification scheme ID. Note that, for $\mathsf{ID} = \mathbf{Daps2Tid}[\mathsf{DS}, \mathsf{sl}]$, the trapdoor key $tk = \varepsilon$, so this is a perfect simulation of $\mathrm{EX}^{\mathcal{A}}_{\mathsf{ID}}$. 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. ▌

D DPRFs

Here we define a new primitive we call a DPRF. It can be used to build a DAPS. We would like to construct this without random oracles but do not know how at this point.

DEFINITIONS. A DPRF F specifies the following. Key generation algorithm $\mathsf{F.Kg}$ takes input a string s called the message and returns a key pair (pk, sk), where pk is a public key and sk is a secret key. Deterministic evaluation algorithm $\mathsf{F.Ev}$ takes sk, a, p, s and returns an output y. Extraction algorithm $\mathsf{F.Ex}$ takes $pk, a, p_1, p_2, y_1, y_2$ and returns a string.

Define the advantage $\mathbf{Adv}^{\mathrm{ex}}_{\mathsf{F}}(\mathcal{A}) = \Pr[\mathrm{EX}^{\mathcal{A}}_{\mathsf{F}}]$ associated to adversary $\mathcal{A}$, where game $\mathrm{EX}^{\mathcal{A}}_{\mathsf{F}}$ is in Fig. 18. We require that this advantage be negligible for all polynomial time $\mathcal{A}$. This measures extraction.

We also require pseudo-randomness. Define the advantage $\mathbf{Adv}^{\mathrm{dprf}}_{\mathsf{F}}(\mathcal{A}) = 2\Pr[\mathrm{DPRF}^{\mathcal{A}}_{\mathsf{F}}] - 1$ where game $\mathrm{DPRF}^{\mathcal{A}}_{\mathsf{F}}$ is in Fig. 18. We require that this advantage be negligible for all polynomial time $\mathcal{A}$. This says that the outputs look random as long as addresses do not repeat.