The Double-Hash Transform: From Identification to (Double-Authentication-Preventing) Signatures, Tightly

Mihir Bellare[1] Douglas Stebila[2]

December 2015

Abstract

We give a new method to turn identification schemes into signature schemes with two advantages over the Fiat-Shamir transform: (1) Security is proven with *tight* reductions to *standard* assumptions, and (2) We obtain not just ordinary signatures, but ones with extra properties, specifically *double-authentication prevention*. The method consists of defining a new goal for identification called security against *constrained impersonation* that, unlike standard goals, can be shown via tight reductions from standard assumptions, and then using a double-hashing transform to turn identification schemes that meet this notion, and have an additional syntactic property we call being trapdoor, into double-authentication-preventing signature (DAPS) schemes, again with a tight reduction. 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 ones, but competitive with in-use signature schemes that lack the double authentication preventing property. DAPS are of potential interest in deterring mass surveillance.

[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

Identification-based signatures are attractive because they are efficient. This paper offers a new way to obtain them that (1) Unlike the Fiat-Shamir transform, has a *tight* reduction to standard underlying algebraic problems, so that security is justified for smaller parameter sizes, thereby increasing speed, and (2) Yields not just ordinary signatures but ones with extra properties. The key is to rely on a new, weak notion of security of identification against *constrained* impersonation that is established with a tight reduction.

BACKGROUND. By an identification scheme we mean a three-move Sigma protocol[1] in which the prover sends a *commitment* Y computed using private randomness y, the verifier sends a random *challenge* c, the prover returns a *response* z computed using y and its secret key *isk*, and the verifier computes a boolean decision from the conversation transcript $Y\|c\|z$ and public key *ivk* (see Fig. 3). The most basic goal is security against impersonation under passive attack (imp) which requires that an adversary given transcripts of honest protocol executions still fails to make the honest verifier accept in an interaction where it plays the role of the prover, itself picking Y any way it likes, receiving a random c, and then producing z. Imp-security can be established by reduction to the hardness of the algebraic problem underlying the identification scheme (eg. one-wayness of RSA for GQ [17], factoring for FS [14], discrete log for Schnorr [26], ...). However a given imp adversary with advantage $\epsilon_{\rm imp}$ must be twice successful in order to extract the secret key, so, via the reset lemma of [6], $\epsilon_{\rm imp} \approx \sqrt{\epsilon_{\rm alg}}$ where $\epsilon_{\rm alg}$ is the advantage in breaking the algebraic problem, meaning the reduction is very loose.

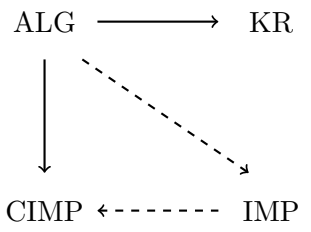
The Fiat-Shamir transform [14] derives from an identification scheme the signature scheme in which a signature of a message m is a pair (Y, z) such that the transcript $Y\|c\|z$ is accepting for $c = \mathrm{H}(Y\|m)$. AABN [1] reduce security of the signature scheme to the imp security of the identification scheme, with the hash function H modeled as a random oracle [7]. A q-query forger will have advantage $\epsilon_{\rm sig} \approx q\,\epsilon_{\rm imp}$ against the signature scheme, which, combined with the above, means $\epsilon_{\rm sig} \approx q\sqrt{\epsilon_{\rm alg}}$. There are thus two sources of loss in the reduction, namely the square-root and the factor of q. We aim to do away with *both*. Our approach relies crucially on some (new) definitions.

CONSTRAINED IMPERSONATION. We return to the identification scheme and provide an alternative definition of security. We continue, as with imp, to allow a passive attack in which the adversary against the identification scheme can obtain transcripts $Y_1\|c_1\|z_1, Y_2\|c_2\|z_2 \ldots$ of interactions between the honest prover and verifier. Recall that in imp, this is followed by letting the adversary play the role of cheating prover, where it can send a commitment Y of its choice to the verifier, and succeeds if it finds a correct response to the random challenge it receives. In constrained impersonation (cimp), there are two changes. First, the adversary may not select its own Y. It must use (is constrained to use) a commitment from one of the transcripts it received in its passive attack. Second, it is allowed multiple impersonation attempts. Each one consists of pointing at some transcript i, receiving a (fresh) random challenge c, and then, providing a response z, with success meaning that the transcript $Y_i\|c\|z$ is accepting for at least one of the impersonation attempts.[2]

Relations between cimp, imp and the algebraic problem alg are depicted in Fig. 1. (Ignore kr for now.) A dotted line is a loose reduction and a full line is a tight reduction. We see that the

[1] This means it is honest-verifier zero knowledge, and from two accepting transcripts with the same commitment but different challenges, one can extract the secret key.

[2] One might note that cimp is quite useless from the point of view of the use in practice of the identification scheme for actual identification, because in that setting we may not be able to constrain the adversary to use only prior commitments. However we are interested in identification only as an intermediate tool, not as an end goal. Cimp will work well for us in the former regard.



ALG : Algebraic assumption underlying ID scheme
IMP : Impersonation under passive attack
CIMP : Constrained impersonation under passive attack
KR : Key recovery

Transform	Signature	Where $\mathsf{ID.Vf}(ivk, Y\|\|c\|\|z)$ and	ID security	**Reduction**
FS	(Y, z)	$c = \mathrm{H}(Y\|\|m)$	imp	loose
Swap	(c, z)	$Y = \mathrm{H}(c\|\|m)$	N/A	tight
H2	(z, s)	$Y = \mathrm{H}_1(m), \quad c = \mathrm{H}_2(m\|\|s)$	cimp	tight

Transform	DAPS	Where $\mathsf{ID.Vf}(ivk, Y\|\|c\|\|z)$ and	ID security	**Reduction**
H2$_+$	(z, s)	$Y = \mathrm{H}_1(a), \quad c = \mathrm{H}_2(a\|\|p\|\|s)$	cimp, kr	tight

Figure 1: **Top:** Relations between notions of security for an identification scheme ID. Dotted lines denote loose reductions and full lines denote tight reductions. **Middle:** Transforms of identification schemes into signature schemes. A signature as shown in the 2nd column defines an accepting transcript $Y\|c\|z$ with quantities as determined in the third column where $\mathrm{H}, \mathrm{H}_1, \mathrm{H}_2$ are random oracles and where s for **H2** is a randomly-chosen seed. The 4th column shows the assumption made on the identification scheme to prove security of the signature scheme. The last column indicates tightness of the signature reduction all the way down to the underlying algebraic assumption. The **Swap** and **H2** transforms require the identification scheme to be trapdoor. **Bottom:** Our transform generalized to yield double-authentication preventing signatures (DAPS) rather than ordinary signatures; in DAPS, a message $m = a\|p$ consists of an address a and a payload p.

---

reduction from the algebraic problem to cimp, unlike the one to imp, is tight. Imp does imply cimp, but there is a factor of loss equal to the number of impersonation attempts, so this reduction is loose. Besides being tight, the reduction of cimp to the algebraic problem has the advantage of being easy and direct to establish. Theorems 4 and 5 detail instances of it, showing smaller terms we have omitted in this discussion.

Now the question we pose is, can we take advantage of this by devising a transform of identification into signatures that *relies only on the cimp security of the identification*? Furthermore, we must ensure that our transform itself incurs no losses, meaning *security of the signature reduces tightly to cimp*. If so, we would get signatures tightly from the algebraic problem. Our **H2** transform does this. First we need another definition.

Trapdoor identification. Recall that in an identification scheme, the prover uses private randomness y to generate its commitment Y. The **H2** transform will not work with an arbitrary identification scheme. It requires the latter to be *trapdoor*. This means the prover can pick the commitment Y directly at random from the space of commitments and then compute the associated private randomness y using its secret key via a prescribed algorithm. A formal definition is in Section 3.

Many existing identification schemes will meet our definition of being trapdoor modulo possibly

some changes to the key structure. Thus the GQ-ID of [17] is trapdoor if we add the decryption exponent d to the secret key. With similar changes to the keys, the Fiat-Shamir [14] and Ong-Schnorr [22] identification schemes are trapdoor. The MR-ID factoring-based identification scheme of [20] is trivially trapdoor. We also define and work with a trapdoor identification scheme CF-ID based on claw-free permutations that generalizes MR-ID. But not all identification schemes are trapdoor. A prominent one that is not is Schnorr's (discrete-log based) scheme [26].

BASIC DOUBLE-HASH TRANSFORM **H2**. The construction and its attributes are summarized in the 3rd row of the first table in Fig. 1. The signer specifies the commitment as a hash $Y = \mathrm{H}_1(m)$ of the message under a hash function H_1. Then it picks a random seed s, of some seed length sl associated to the scheme, and obtains a challenge $c = \mathrm{H}_2(m\|s)$ by hashing the message and seed under another hash function H_2. Using the trapdoor property of the identification scheme and the secret key, it computes a response z, returning (z, s) as the signature. Verification consists of checking that the transcript $Y\|c\|z$ is accepting with Y, c specified as above. Theorem 1 shows unforgeability[3] with a tight reduction to the cimp security of the identification scheme, meaning $\epsilon_{\mathrm{sig}} \approx \epsilon_{\mathrm{cimp}}$. This, by the above fact that cimp is itself obtained tightly from the algebraic problem, results in a tight reduction of unforgeability to the latter, meaning $\epsilon_{\mathrm{sig}} \approx \epsilon_{\mathrm{alg}}$.

We stress that the algebraic assumptions we use are the same standard and classical ones used historically to show imp of the identification scheme and security of the FS-derived signature scheme. For example, when the base identification scheme is GQ-ID, the assumption is one-wayness of RSA. When it is the Fiat-Shamir or MR-ID, it is factoring, and so on.

DAPS. Double-authentication-preventing signature (DAPS) schemes were introduced by Poettering and Stebila (PS) [23]. 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. Additionally, 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 [23] 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 [23] 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 $\mathrm{cert}_1 = (\texttt{example.com}, pk_1, \sigma_1)$ 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 $\mathrm{cert}_0 = (\texttt{example.com}, pk_0, \sigma_0)$ 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

[3] A technical point is that this is under the assumption that a particular message is not signed twice. To remove this restriction, we would de-randomize signing, specifying the seed via the RO applied to the secret key and message.

with big brother's request to create the rogue certificate.

PRIOR SCHEMES. PS [23] 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. 17.)

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 show that **H2** naturally and easily extends to obtain DAPS schemes that are secure and efficient.

EXTENDED DOUBLE-HASH TRANSFORM $\mathbf{H2}_+$. We obtain a DAPS from a trapdoor identification scheme via $\mathbf{H2}_+$ as summarized in the last table in Fig. 1. The signer specifies the commitment as a hash $Y = \mathrm{H}_1(a)$ of the address, picks a random seed s of length sl, 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 (cf. Fig. 10). Theorem 2 establishes the double-authentication prevention property via the extractability property of the identification Sigma protocol. For unforgeability we need to consider the additional property kr that the identification scheme is secure against recovery of the secret key under passive attack. As Fig. 1 and Theorems 4, 5 indicate, however, this is also easily and tightly established under the algebraic problem. Theorem 3 shows unforgeability with a tight reduction to the cimp+kr security of the identification scheme, meaning $\epsilon_{\mathrm{sig}} \approx \epsilon_{\mathrm{cimp}} + \epsilon_{\mathrm{kr}}$. This results in a tight reduction of unforgeability of the DAPS to the algebraic problem, meaning $\epsilon_{\mathrm{sig}} \approx 2\epsilon_{\mathrm{alg}}$.

GQ-DAPS. As an example, applying our extended $\mathbf{H2}_+$ transform to the GQ identification scheme results in the following GQ-DAPS scheme. 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 $TK = \mathrm{H}_3(x) \oplus d$. The keys are thus as in the GQ identification scheme [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 $Y = \mathrm{H}_1(a)$ —the commitment is not picked at random but determined uniquely by the address— let $y = Y^d \bmod N$, let $c = \mathrm{H}_2(a\|p\|s) \in \{0,1\}^l$ —the challenge— let $z = yx^c \bmod N$ and return (z, s) as the signature. We omit describing verification; see Fig. 15 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-ID 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 $d = \mathrm{H}_3(x) \oplus TK$. We prove unforgeability tightly under one-wayness of RSA, and efficiency is again good on all fronts (cf. Fig. 17). Many other DAPS can be similarly obtained.

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, MR-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. 17 shows the signing time, verifying time, signature size and key sizes for all schemes. GQ-DAPS, MR-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, MR-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 Section 8 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.

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 [23]. 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" [27]. 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.

Swap. Micali and Reyzin [20] give a signature scheme with a tight reduction to factoring. It is obtained from their MR-ID identification scheme via a method they call "swap." ABN [2] say that the method generalizes to other factoring-based schemes. However, "swap" has never been stated as a general transform of an identification scheme into a signature scheme; it appears rather as an ad hoc technique to go directly and tightly from the algebraic problem to the signature. This lack of abstraction is perhaps due in part to a lack of definitions, and our notion of a trapdoor identification scheme allows us to partially fill the gap. We claim —formalizing discussion in ABN [2]— that this is the syntactic condition allowing swap to operate. The 2nd row of the first table of Fig. 1 summarizes what we now call the **Swap** transform to turn a trapdoor identification scheme into a signature scheme. However, it continues to be that security is reduced directly to the algebraic problem rather than to a goal of the identification scheme, explaining the "N/A" entry in the table.

How do **Swap** and **H2** compare? They apply to the same class of identification schemes (trapdoor) and both result in tight reductions of the signature to the underlying algebraic problem. However, **H2** answers what we believe is the compelling question we posed above, namely to obtain a signature scheme whose security reduces tightly to the cimp security of an underlying identification scheme, motivated by the fact that this notion, unlike classical imp, is easily and tightly established in turn under the algebraic assumption. **Swap** does not do this: cimp security of the identification scheme demonstrably does *not* suffice to establish security of **Swap** signatures. The other difference is that, unlike either the **FS** or the **Swap** transforms, **H2** extends easily to

yield DAPS, and we obtain thereby the most efficient such signatures to date, as discussed above.

FURTHER RELATED WORK AND OPEN QUESTIONS. ABP [2] show a tight reduction of **FS**-derived GQ signatures to the Φ-hiding assumption of [10]. In contrast, both **H2**-derived and **Swap**-derived GQ signatures have a tight reduction to the standard one-wayness of RSA. AFLT [3] use a slight variant of the Fiat-Shamir transform to turn lossy identification schemes into signature schemes with security based tightly on key indistinguishability, resulting in signature schemes with tight reductions to the decisional short discrete logarithm problem, the shortest vector problem in ideal lattices and subset sum.

The first proofs of unforgeability for **FS**-derived signatures [24] were direct, meaning they reduced security of the signature scheme directly to the hardness of the algebraic problem underlying the identification scheme. This was done using forking lemmas [24, 5, 4]. Indirect proofs begin with [21, 1]. As indicated above, AABN [1] reduce security of the signature scheme to the imp security of the identification scheme. However, both the direct and the indirect approach result in reductions of the same looseness we discussed above. The advantage of working with cimp and **H2** as opposed to imp and **FS** is to remove this looseness while retaining the modular structure of the AABN approach. The benefits of the latter are that the identification schemes can be analyzed separately, and the use of random oracles is confined to the transform.

Both our DAPS and that of PS [23] 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 [25]. 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 messages is conceptually similar to the recoverability of the spender's identity from double-spending of an e-coin in offline e-cash [11]. Whether this connection can be exploited to obtain new DAPS schemes is an interesting open question.

2 Notation and some standard definitions

NOTATION. We let ε denote the empty string. If X is a finite set, we let $x \xleftarrow{\$} X$ denote picking an element of X uniformly at random and assigning it to x. Algorithms may be randomized unless otherwise indicated. Running time is worst case. If A is an algorithm, we let $y \leftarrow A(x_1, \ldots; r)$ denote running A with random coins r on inputs $x_1, \ldots$ and assigning the output to y. We let $y \xleftarrow{\$} A(x_1, \ldots)$ be the result of picking r at random and letting $y \leftarrow A(x_1, \ldots; r)$. We let $[A(x_1, \ldots)]$ denote the set of all possible outputs of A when invoked with inputs $x_1, \ldots$. We use the code based game playing framework of [8]. (See Fig. 2 for an example.) By Pr[G] we denote the event that the execution of game G results in the game returning true. We also adopt the convention that the running time of an adversary refers to the worst case execution time of the game with the adversary. This means that the time taken for oracles to compute replies to queries is included.

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

Game $\mathrm{UF}^{\mathcal{A}}_{\mathsf{DS}}$	$\textsc{Sign}(m)$
$(vk, sk) \leftarrow\!\!\$\ \mathsf{DS.Kg}^{\mathrm{H}}$; $M \leftarrow \emptyset$	$M \leftarrow M \cup \{m\}$; $\sigma \leftarrow\!\!\$\ \mathsf{DS.Sig}^{\mathrm{H}}(vk, sk, m)$
$(m, \sigma) \leftarrow\!\!\$\ \mathcal{A}^{\textsc{Sign},\mathrm{H}}(vk)$	Return σ
Return $(\mathsf{DS.Vf}^{\mathrm{H}}(vk, m, \sigma) \wedge (m \notin M))$	$\mathrm{H}(x, \mathrm{Rng})$
	If not $\mathrm{HT}[x, \mathrm{Rng}]$ then $\mathrm{HT}[x, \mathrm{Rng}] \leftarrow\!\!\$\ \mathrm{Rng}$
	Return $\mathrm{HT}[x, \mathrm{Rng}]$

Figure 2: Game defining unforgeability of signature scheme DS.

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, with syntax is as discussed above. Now it can compute a signature $\sigma \leftarrow\!\!\$\ \mathsf{DS.Sig}^{\mathrm{H}}(vk, sk, m)$ on any message $m \in \{0,1\}^*$. A verifier can deterministically compute a boolean $v \leftarrow \mathsf{DS.Vf}^{\mathrm{H}}(vk, m, \sigma)$ indicating whether or not σ is a valid signature of m relative to vk. Correctness as usual requires that $\mathsf{DS.Vf}^{\mathrm{H}}(vk, m, \mathsf{DS.Sig}^{\mathrm{H}}(vk, sk, m)) = \mathsf{true}$ with probability one. Unforgeability is measured via the advantage $\mathbf{Adv}^{\mathrm{uf}}_{\mathsf{DS}}(\mathcal{A}) = \Pr[\mathrm{UF}^{\mathcal{A}}_{\mathsf{DS}}]$ of adversary $\mathcal{A}$, where game $\mathrm{UF}^{\mathcal{A}}_{\mathsf{DS}}$ is in Fig. 2. This is the classical notion of [16], lifted to the ROM as per [7].

3 Cimp-secure Identification

We define security of identification schemes against constrained impersonation. Later we will show how any trapdoor identification scheme with this property can be transformed into a DAPS with a tight reduction.

Identification. An identification (ID) scheme ID operates as depicted in Fig. 3. 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. 15. We require the obvious correctness condition. We also require the Sigma Protocol [12] 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 ask that $\Pr[\mathrm{EX}^{\mathcal{A}}_{\mathsf{ID}}] = 0$ for all adversaries $\mathcal{A}$, where the game is in Fig. 3. Finally we require a key-verification algorithm $\mathsf{ID.KVf}$ such that $\mathsf{ID.KVf}(ivk, isk) = \mathsf{true}$ if $(isk, ivk, tk) \in [\mathsf{ID.Kg}]$ for some tk, and false otherwise.

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.

Security of ID schemes. Recall that security of an identification scheme ID under impersonation [13, 1] 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,

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

Game $\mathrm{EX}^{\mathcal{A}}_{\mathsf{ID}}$
$(ivk, isk, tk) \leftarrow\!\!\$\ \mathsf{ID.Kg}$
$(Y, c_1, z_1, c_2, z_2) \leftarrow\!\!\$\ \mathcal{A}(ivk, isk, tk)$
$v_1 \leftarrow \mathsf{ID.Vf}(ivk, Y\|c_1\|z_1)\ ;\ v_2 \leftarrow \mathsf{ID.Vf}(ivk, Y\|c_2\|z_2)$
$isk^* \leftarrow\!\!\$\ \mathsf{ID.Ex}(ivk, Y, c_1, z_1, c_2, z_2)$
Return $(isk^* \neq isk) \wedge (c_1 \neq c_2) \wedge v_1 \wedge v_2$

Figure 3: Functioning of an identification scheme ID and game defining its Sigma-Protocol extractability.

Game $\mathrm{cIMP}^{\mathcal{P}}_{\mathsf{ID}}$
$(ivk, isk, tk) \leftarrow\!\!\$\ \mathsf{ID.Kg}\ ;\ i \leftarrow 0\ ;\ j \leftarrow 0$
$d \leftarrow\!\!\$\ \mathcal{P}^{\mathrm{Tr},\mathrm{Ch},\mathrm{Dec}}(ivk)$
Return win

Game $\mathrm{KR}^{\mathcal{P}}_{\mathsf{ID}}$
$(ivk, isk, tk) \leftarrow\!\!\$\ \mathsf{ID.Kg}\ ;\ i \leftarrow 0$
$x \leftarrow\!\!\$\ \mathcal{P}^{\mathrm{Tr}}(ivk)$
Return $(x = isk)$

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

$\mathrm{Ch}(l)$
If not $(1 \leq l \leq i)$ then return $\perp$
$j \leftarrow j + 1\ ;\ U \leftarrow U \cup \{j\}\ ;\ c \leftarrow\!\!\$\ \{0,1\}^{\mathsf{ID.cl}}$
$\mathrm{TT}[j] \leftarrow Y_l\|c\ ;$ Return (j, c)

$\mathrm{Dec}(j, z)$
If $(j \notin U)$ then return $\perp$
$U \leftarrow U \setminus \{j\}\ ;\ \mathrm{TT}[j] \leftarrow \mathrm{TT}[j]\|z$
$\mathrm{DT}[j] \leftarrow \mathsf{ID.Vf}(ivk, \mathrm{TT}[j])$
$\mathsf{win} \leftarrow \mathsf{win} \vee \mathrm{DT}[j]\ ;$ Return $\mathrm{DT}[j]$

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

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]. We are interested only in the weakest, namely passive attacks, where the adversary is just an eavesdropper and gets honestly-generated protocol transcripts. (Active and even concurrent attacks are relevant in other contexts [6].)

We suggest a new *constrained impersonation* goal that weakens the classic impersonation goal. We will be in the setting where the adversary has a passive attack, allowing it to obtain transcripts. When it makes its impersonation attempt, it is no longer allowed to choose the commitment. Rather, it is required (constrained) to use a commitment from the transcripts. The formalization considers game $\mathrm{cIMP}^{\mathcal{P}}_{\mathsf{ID}}$ of Fig. 4 associated to identification scheme ID and cimp adversary $\mathcal{P}$. The transcript oracle 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 Ch and Dec oracles. It would first have to give Ch the index l of an existing transcript. It would get back a fresh challenge c and a session id j. To win it would

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

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

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

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

have to call Dec with j and a correct response relative to the commitment from the l-th transcript and challenge c. Note that unlike the conventional definitions, we allow multiple impersonation attempts, not just one, reflected in the adversary being able to call oracles Ch, Dec as often as it likes and winning if any attempt is successful. We let $\mathbf{Adv}^{\mathrm{cimp}}_{\mathsf{ID}}(\mathcal{P}) = \Pr[\mathrm{cIMP}^{\mathcal{P}}_{\mathsf{ID}}]$.

A scheme meeting this goal is not necessarily useful for actual identification in practice because in the latter we cannot constrain the adversary to use only existing commitments. However we do not wish to use it directly in this way. We use identification only as a tool. The benefits are (1) it will suffice for our application, but (2) as indicated by Theorems 4 and 5, it can be achieved with *tight* reductions to standard assumptions, as opposed to the stronger impersonation goals.

We also define and use the security of the identification scheme against key recovery under passive attack. The formalization considers game $\mathrm{KR}^{\mathcal{P}}_{\mathsf{ID}}$ of Fig. 4 associated to identification scheme ID and kr adversary $\mathcal{P}$. The transcript oracle Tr is as before. Adversary $\mathcal{P}$ aims simply to recover isk. We let $\mathbf{Adv}^{\mathrm{kr}}_{\mathsf{ID}}(\mathcal{P}) = \Pr[\mathrm{KR}^{\mathcal{P}}_{\mathsf{ID}}]$ be the probability that it succeeds, where success means recovering the secret key. Again Theorems 4 and 5 show that this form of security is easily proven under standard assumptions with a tight reduction.

4 Basic double-hash transform

We specify the basic transform and prove it works.

The construction. Let ID be a trapdoor identification scheme. Our **H2** (double hash) transform associates to it and a seed length $\mathsf{sl} \in \mathbb{N}$ a signature scheme $\mathsf{DS} = \mathbf{H2}[\mathsf{ID}, \mathsf{sl}]$. The algorithms of DS are defined in Fig. 5. Recall that in the Fiat-Shamir transform [14], 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 alone. 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 specify the challenge as a randomized hash of the message. (Unlike in the **FS** transform, the commitment is not hashed along with the message.) The randomization is captured by the seed s whose length sl was a parameter of our transform.

Unforgeability of our construction. The following shows that the unforgeability of our signature tightly reduces to the cimp security of the underlying ID scheme. Recall that as per our conventions, 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

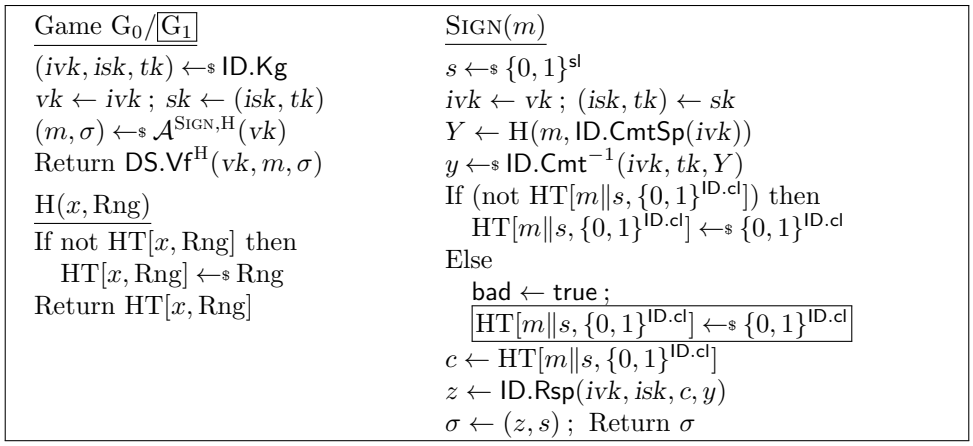


Figure 6: Games for proof of Theorem 1. Game G_1 includes the boxed code and game G_0 does not.

---

used by oracles included.

Theorem 1 *Let signature scheme* $\mathsf{DS} = \mathbf{H2}[\mathsf{ID}, \mathsf{sl}]$ *be obtained from trapdoor identification scheme* ID *and seed length* sl *as in Fig. 5. Let* $\mathcal{A}$ *be a uf-adversary against* DS *that does not repeat a query to its* Sign *oracle. Suppose the number of queries that* $\mathcal{A}$ *makes to its* $\text{H}(\cdot, \mathsf{ID.CmtSp}(ivk))$, $\text{H}(\cdot, \{0,1\}^{\mathsf{ID.cl}})$, Sign *oracles are, respectively,* q_2, q_3, q_s, *where* ivk *is as in game* $\text{UF}^{\mathcal{A}}_{\mathsf{DS}}$. *Then from* $\mathcal{A}$ *we can construct cimp adversary* $\mathcal{P}$ *such that*

$$\mathbf{Adv}^{\text{uf}}_{\mathsf{DS}}(\mathcal{A}) \;\leq\; \mathbf{Adv}^{\text{cimp}}_{\mathsf{ID}}(\mathcal{P}) + \frac{q_s(2q_3 + q_s - 1)}{2^{\mathsf{sl}+1}} \;. \tag{1}$$

Adversary $\mathcal{P}$ *make* $q_2 + q_s + 1$ *queries to* Tr. *It makes* q_3 *queries to* Ch *and one query to* Dec. *It has running time about that of* $\mathcal{A}$.

To remove the restriction to not repeat a Sign query, we can modify the scheme to make signing deterministic in a standard way. Namely, rather than pick a random seed with each signature, we can obtain the seed by applying H to the secret key and message, so that signing is deterministic.

Proof of Theorem 1: We assume that $\mathcal{A}$ avoids certain pointless behavior that would only cause it to lose. Thus, we assume it did not query to Sign the message m in the forgery (m, σ) that it eventually outputs. This means that the set M in game $\text{UF}^{\mathcal{A}}_{\mathsf{DS}}$, and the code and checks associated with it, 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, it first computes Y and picks s. We would like that, at this point, it can define the table entry $\text{HT}[m\|s, \{0,1\}^{\mathsf{ID.cl}}]$ without caring whether it was already defined. (This is to allow an eventual impersonation adversary to program this RO response with a challenge emanating from a transcript obtained from the transcript oracle.) In general, of course, this would be wrong, but intuitively the random choice of s means it is usually right. (This indeed is why we have the seed in the scheme.) To show this formally we consider the games G_0, G_1 of Fig. 6. Game G_0 excludes the boxed code, so that its Sign procedure defines $\text{HT}[m\|s, \{0,1\}^{\mathsf{ID.cl}}]$ only when this entry was not already defined, but game G_1 includes the boxed code, so that Sign defines this entry always, as we would like. But these games are identical-until-bad [8], meaning differ only in code that follows the setting of the boolean flag bad to true. So we have

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

Game G_2	$\underline{\textsc{Sign}(m)}$
$(ivk, isk, tk) \leftarrow\!\!\$\ \mathsf{ID.Kg}$	$s \leftarrow\!\!\$\ \{0,1\}^{\mathsf{sl}}$
$vk \leftarrow ivk$	$Y \leftarrow \mathrm{H}(m, \mathsf{ID.CmtSp}(ivk))$
For $i = 1, \ldots, q_2 + q_s + 1$ do	$i \leftarrow \mathrm{Ind}_2(m)$
$\quad (Y_i, y_i) \leftarrow\!\!\$\ \mathsf{ID.Cmt}(ivk)$	$\mathrm{HT}[m\|s, \{0,1\}^{\mathsf{ID.cl}}] \leftarrow c_i$
$\quad c_i \leftarrow\!\!\$\ \{0,1\}^{\mathsf{ID.cl}}$	$\sigma \leftarrow (z_i, s)$; Return σ
$\quad z_i \leftarrow \mathsf{ID.Rsp}(ivk, isk, c_i, y_i)$	$\underline{\mathrm{H}(x, \mathrm{Rng})}$
$i_2 \leftarrow 0$	If (not $\mathrm{HT}[x, \mathrm{Rng}]$) then
$(m, \sigma) \leftarrow\!\!\$\ \mathcal{A}^{\textsc{Sign},\mathrm{H}}(vk)$	$\quad$ If ($\mathrm{Rng} = \{0,1\}^{\mathsf{ID.cl}}$) then
$(z, s) \leftarrow \sigma$	$\quad\quad \mathrm{HT}[x, \mathrm{Rng}] \leftarrow\!\!\$\ \mathrm{Rng}$
$Y \leftarrow \mathrm{H}(m, \mathsf{ID.CmtSp}(ivk))$	$\quad$ If ($\mathrm{Rng} = \mathsf{ID.CmtSp}(ivk)$) then
$c \leftarrow \mathrm{H}(m\|s, \{0,1\}^{\mathsf{ID.cl}})$	$\quad\quad i_2 \leftarrow i_2 + 1$; $\mathrm{HT}[x, \mathrm{Rng}] \leftarrow Y_{i_2}$; $\mathrm{Ind}_2(x) \leftarrow i_2$
Return $\mathsf{ID.Vf}(ivk, Y\|c\|z)$	Return $\mathrm{HT}[x, \mathrm{Rng}]$

Figure 7: Game G_2 for the proof of Theorem 1.

Adversary $\mathcal{P}^{\textsc{Tr},\textsc{Ch},\textsc{Dec}}(ivk)$	$\underline{\textsc{Sign}(m)}$ // $\mathcal{P}$
$vk \leftarrow ivk$	$s \leftarrow\!\!\$\ \{0,1\}^{\mathsf{sl}}$
For $i = 1, \ldots, q_2 + q_s + 1$ do	$Y \leftarrow \mathrm{H}(m, \mathsf{ID.CmtSp}(ivk))$
$\quad (Y_i, c_i, z_i) \leftarrow\!\!\$\ \textsc{Tr}()$	$i \leftarrow \mathrm{Ind}_2(m)$
$i_2 \leftarrow 0$	$\mathrm{HT}[m\|s, \{0,1\}^{\mathsf{ID.cl}}] \leftarrow c_i$
$(m, \sigma) \leftarrow\!\!\$\ \mathcal{A}^{\textsc{Sign},\mathrm{H}}(vk)$	$\sigma \leftarrow (z_i, s)$; Return σ
$(z, s) \leftarrow \sigma$	$\underline{\mathrm{H}(x, \mathrm{Rng})}$ // $\mathcal{P}$
$Y \leftarrow \mathrm{H}(m, \mathsf{ID.CmtSp}(ivk))$	If (not $\mathrm{HT}[x, \mathrm{Rng}]$) then
$c \leftarrow \mathrm{H}(m\|s, \{0,1\}^{\mathsf{ID.cl}})$	$\quad$ If ($\mathrm{Rng} = \{0,1\}^{\mathsf{ID.cl}}$) then
$j \leftarrow \mathrm{Ind}_3(m\|s)$	$\quad\quad m\|s \leftarrow x$; $Y \leftarrow \mathrm{H}(m, \mathsf{ID.CmtSp}(ivk))$
$d \leftarrow \textsc{Dec}(j, z)$	$\quad\quad l \leftarrow \mathrm{Ind}_2(m)$; $(j, c) \leftarrow\!\!\$\ \textsc{Ch}(l)$
	$\quad\quad \mathrm{Ind}_3(x) \leftarrow j$; $\mathrm{HT}[x, \mathrm{Rng}] \leftarrow c$
	$\quad$ If ($\mathrm{Rng} = \mathsf{ID.CmtSp}(ivk)$) then
	$\quad\quad i_2 \leftarrow i_2 + 1$; $\mathrm{HT}[x, \mathrm{Rng}] \leftarrow Y_{i_2}$; $\mathrm{Ind}_2(x) \leftarrow i_2$
	Return $\mathrm{HT}[x, \mathrm{Rng}]$

Figure 8: Adversary for proof of Theorem 1.

---

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

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

Now we need to bound $\Pr[\mathrm{G}_1]$. Consider game G_2 of Fig. 7. Towards using cimp, this game refrains from using *isk* directly in procedure Sign. Instead, it begins by generating conversation transcripts $Y_i\|c_i\|z_i$ and has Sign use these. To make this possible, $\mathrm{H}(\cdot, \mathsf{ID.CmtSp}(ivk))$ values are set to the transcript commitments. Then Sign retrieves the corresponding commitment Y, sets $\mathrm{HT}[m\|s, \{0,1\}^{\mathsf{ID.cl}}]$ to the challenge from the same transcript, and puts the corresponding response in the signature. Since the signatures are correctly distributed we have

$$\Pr[\mathrm{G}_1] = \Pr[\mathrm{G}_2] \ . \tag{4}$$

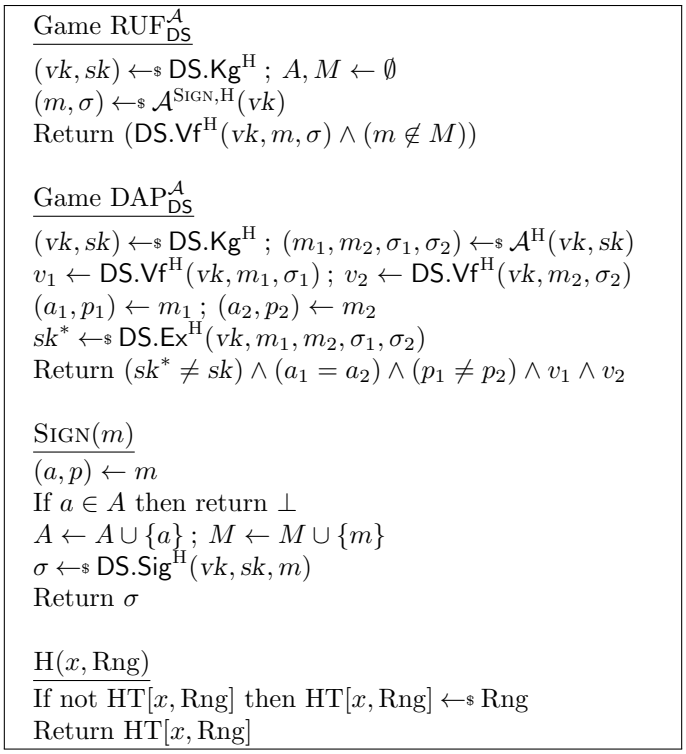


Figure 9: Games defining unforgeability and extractability conditions of DAPS DS. The SIGN procedure is invoked by game RUF while H is invoked by both games.

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

$$\Pr[\mathrm{G}_2] \leq \mathbf{Adv}^{\mathrm{cimp}}_{\mathsf{ID}}(\mathcal{P}) \ . \tag{5}$$

Game G_2 was crafted exactly to make the construction of adversary $\mathcal{P}$ quite direct. The construction is described in detail in Fig. 8. Adversary $\mathcal{P}$ has access to oracles TR, CH, DEC as per game $\mathrm{cIMP}^{\mathcal{P}}_{\mathsf{ID}}$ 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_2. Using these, SIGN can be simulated as per game G_2. Oracle $\mathrm{H}(\cdot, \mathrm{Rng})$ is simulated as in G_2 when $\mathrm{Rng} = \mathsf{ID.CmtSp}(ivk)$. When a query x is made to $\mathrm{H}(\cdot, \{0,1\}^{\mathsf{ID.cl}})$, adversary $\mathcal{P}$ parses x as $m \| s$. It then retrieves the index l corresponding to m in the list of transcripts and submits this to its challenge oracle CH to get back a session id j and a challenge, and returns this challenge as the response to the oracle query. Finally when $\mathcal{A}$ produces a forgery, the session id corresponding to the message and seed in the forgery is retrieved via 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. ▮

5 DAPS definitions

Let DS be a signature scheme. In a DAPS [23], a message $m = (a, p)$ is a pair consisting of an *address* a and a *payload* p. We require the DAP property and a restricted form of unforgeability.

THE DAP PROPERTY. 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 [23] 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. 9. 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 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 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 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. We also require *restricted unforgeability*, captured formally by the advantage $\mathbf{Adv}^{\mathrm{ruf}}_{\mathsf{DS}}(\mathcal{A}) = \Pr[\mathrm{RUF}^{\mathcal{A}}_{\mathsf{DS}}]$ associated to adversary $\mathcal{A}$, where game $\mathrm{RUF}^{\mathcal{A}}_{\mathsf{DS}}$ is in Fig. 9 [23]. This is the classical notion 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. 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 computation of signatures indistinguishable from real ones. Indeed, such a property is formalized in PS [23]. 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 [23]. 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.

6 The extended double-hash transform

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 $\mathbf{H2}_+$ (extended double hash) transform associates to it and a seed length $\mathsf{sl} \in \mathbb{N}$ a DAPS $\mathsf{DS} = \mathbf{H2}_+[\mathsf{ID}, \mathsf{sl}]$. The algorithms of DS are defined in Fig. 10. We instead specify the commitment Y as a hash of the address rather than of the full message as in **H2**. 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. We then specify the challenge c as a randomized hash of the message and seed, as in basic $\mathbf{H2}_+$. The introduction of the trapdoor tk however creates

$\underline{\mathsf{DS.Kg}^{\mathrm{H}}}$
$(ivk, isk, tk) \leftarrow\!\!\$\ \mathsf{ID.Kg}$
$TK \leftarrow tk \oplus \mathrm{H}(isk, \{0,1\}^{\mathsf{ID.tl}})$
$vk \leftarrow (ivk, TK)$; $sk \leftarrow (isk, tk)$
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
  $(a_i, p_i) \leftarrow m_i$; $(z_i, s_i) \leftarrow \sigma_i$
  $Y_i \leftarrow \mathrm{H}(a_i, \mathsf{ID.CmtSp}(ivk))$
  $c_i \leftarrow \mathrm{H}(a_i \| p_i \| s_i, \{0,1\}^{\mathsf{ID.cl}})$
$isk^* \leftarrow \mathsf{ID.Ex}(ivk, Y_1 \| c_1 \| z_1, Y_2 \| c_2 \| z_2)$
$tk^* \leftarrow \mathrm{H}(isk^*, \{0,1\}^{\mathsf{ID.tl}}) \oplus TK$
Return (isk^*, tk^*)

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

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

Figure 10: Our construction of a DAPS $\mathsf{DS} = \mathbf{H2}_{+}[\mathsf{ID}, \mathsf{sl}]$ from a trapdoor identification scheme ID and a seed length $\mathsf{sl} \in \mathbb{N}$.

---

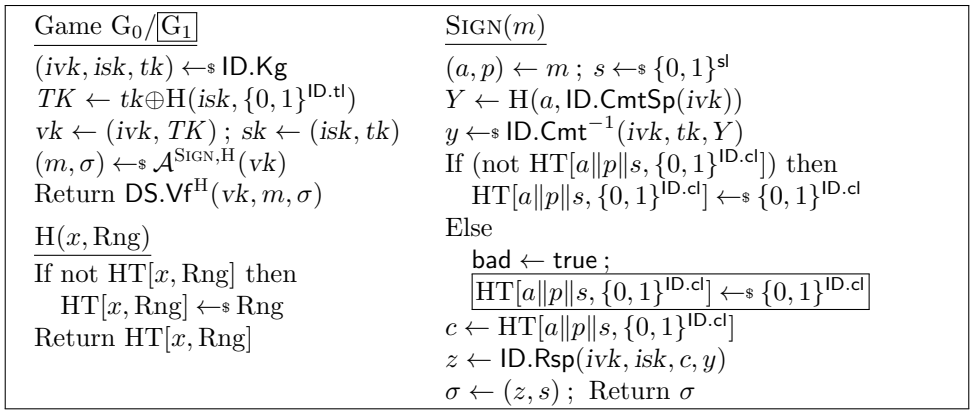
a new difficulty, namely that extraction under the ID scheme will only recover *isk* and to achieve double authentication prevention we must recover the entire secret key $sk = (isk, tk)$. We resolve this by putting in the verification key a particular encryption, denoted TK, of tk, under *isk*.

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. As per our conventions, 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 2 *Let DAPS* $\mathsf{DS} = \mathbf{H2}_{+}[\mathsf{ID}, \mathsf{sl}]$ *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* $\mathbf{Adv}^{\mathrm{dap}}_{\mathsf{DS}}(\mathcal{A}) \leq q(q-1)/2^{\mathsf{ID.cl}+1}$.

Proof of Theorem 2: Consider the $\mathrm{DAP}^{\mathcal{A}}_{\mathsf{DS}}$ game of Fig. 9. Within this, consider the execution of the algorithm $\mathsf{DS.Ex}^{\mathrm{H}}$ of Fig. 10 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. 10 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 $isk^* = isk$. If so, we also can obtain $tk^* = tk$, 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 probability that these challenges collide is at most $q(q-1)/2^{\mathsf{ID.cl}+1}$. ∎

RESTRICTED UNFORGEABILITY OF OUR CONSTRUCTION. The following shows that the restricted unforgeability of our DAPS tightly reduces to the cimp plus kr security of the underlying ID scheme.



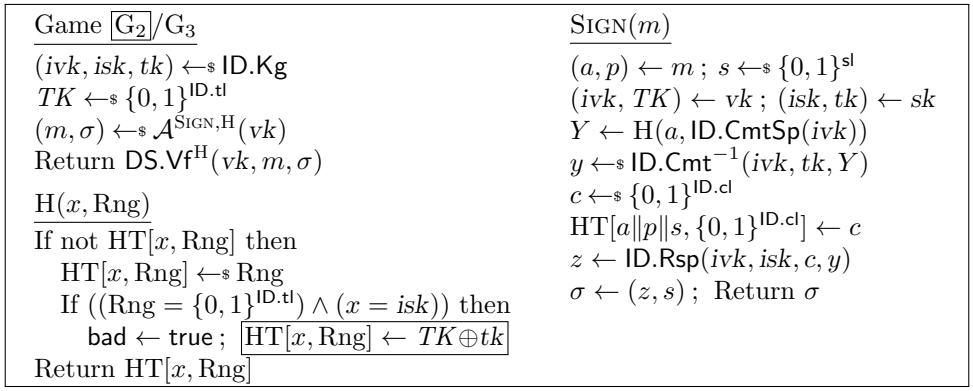


Figure 11: Games for proof of Theorem 3. Games $\mathrm{G}_1, \mathrm{G}_2$ include the boxed code and games $\mathrm{G}_0, \mathrm{G}_3$ do not.

Theorem 3 *Let DAPS* $\mathsf{DS} = \mathbf{H2}_+[\mathsf{ID}, \mathsf{sl}]$ *be obtained from trapdoor identification scheme* ID *and seed length* sl *as in Fig. 10. Let* $\mathcal{A}$ *be a ruf-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* $\mathrm{RUF}^{\mathcal{A}}_{\mathsf{DS}}$*. Then from* $\mathcal{A}$ *we can construct cimp adversary* $\mathcal{P}_1$ *and kr adversary* $\mathcal{P}_2$ *such that*

$$\begin{aligned} &\mathbf{Adv}^{\mathrm{ruf}}_{\mathsf{DS}}(\mathcal{A}) \\ &\leq \quad \mathbf{Adv}^{\mathrm{cimp}}_{\mathsf{ID}}(\mathcal{P}_1) + \mathbf{Adv}^{\mathrm{kr}}_{\mathsf{ID}}(\mathcal{P}_2) + \frac{q_s(2q_3 + q_s - 1)}{2^{\mathsf{sl}+1}} \ . \end{aligned} \tag{6}$$

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*. The running time of adversaries* $\mathcal{P}_1$ *is about that of* $\mathcal{A}$*. The running time of adversary* $\mathcal{P}_2$ *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 $\mathrm{RUF}^{\mathcal{A}}_{\mathsf{DS}}$, and the code and checks associated with them, are redundant and can be removed. We will work with this simplified form of the game.

Identical-until-bad games $\mathrm{G}_0, \mathrm{G}_1$ of Fig. 11, as in the proof of Theorem 1, 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

$$\begin{aligned}\mathbf{Adv}^{\mathrm{ruf}}_{\mathsf{DS}}(\mathcal{A}) &= \Pr[\mathrm{G}_0] = \Pr[\mathrm{G}_1] + \Pr[\mathrm{G}_0] - \Pr[\mathrm{G}_1] \\ &\le \Pr[\mathrm{G}_1] + \Pr[\mathrm{G}_0 \text{ sets } \mathsf{bad}] \ , \end{aligned} \tag{7}$$

where the inequality is by the Fundamental Lemma of Game Playing of [8]. The random choice of s made by procedure SIGN ensures

$$\Pr[\mathrm{G}_0 \text{ sets } \mathsf{bad}] \le \sum_{i=0}^{q_s-1} \frac{q_3 + i}{2^{\mathsf{sl}}} = \frac{q_s(2q_3 + q_s - 1)}{2^{\mathsf{sl}+1}} \ . \tag{8}$$

Now we need to bound $\Pr[\mathrm{G}_1]$. We start by considering whether the ciphertext $TK \leftarrow tk \oplus \mathrm{H}(isk, \{0,1\}^{\mathsf{ID.tl}})$ helps $\mathcal{A}$ over and above access to SIGN. Consider the games $\mathrm{G}_2, \mathrm{G}_3$ of Fig. 11. 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

$$\begin{aligned}\Pr[\mathrm{G}_1] = \Pr[\mathrm{G}_2] &= \Pr[\mathrm{G}_3] + \Pr[\mathrm{G}_2] - \Pr[\mathrm{G}_3] \\ &\le \Pr[\mathrm{G}_3] + \Pr[\mathrm{G}_3 \text{ sets } \mathsf{bad}] \ , \end{aligned} \tag{9}$$

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. The first relies on the assumed cimp security of ID, the second on its assumed kr security.

To bound $\Pr[\mathrm{G}_3]$, consider game G_4 of Fig. 12. It mimics game G_2 in the proof of Theorem 1, moving us towards using cimp by generating conversation transcripts $Y_i \| c_i \| z_i$ and having SIGN use these. We have

$$\Pr[\mathrm{G}_3] = \Pr[\mathrm{G}_4] \ . \tag{10}$$

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

$$\Pr[\mathrm{G}_4] \le \mathbf{Adv}^{\mathrm{cimp}}_{\mathsf{ID}}(\mathcal{P}_1) \ . \tag{11}$$

The construction of $\mathcal{P}_1$ mimics the construction of $\mathcal{P}$ in the proof of Theorem 1, and is described in detail in Fig. 13.

To bound $\Pr[\mathrm{G}_3 \text{ sets } \mathsf{bad}]$, consider game G_5 of Fig. 12. It answers SIGN queries just like G_4, and the only modification in answering H queries is to keep track of queries to $\mathrm{H}(\cdot, \{0,1\}^{\mathsf{ID.tl}})$ in the set T. The game ignores the forgery, returning true if isk was queried to $\mathrm{H}(\cdot, \{0,1\}^{\mathsf{ID.tl}})$. We have

$$\Pr[\mathrm{G}_3 \text{ sets } \mathsf{bad}] = \Pr[\mathrm{G}_5] \ . \tag{12}$$

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

$$\Pr[\mathrm{G}_5] \le \mathbf{Adv}^{\mathrm{kr}}_{\mathsf{ID}}(\mathcal{P}_2) \ . \tag{13}$$

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 $\mathcal{A}$. 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. 13, quite

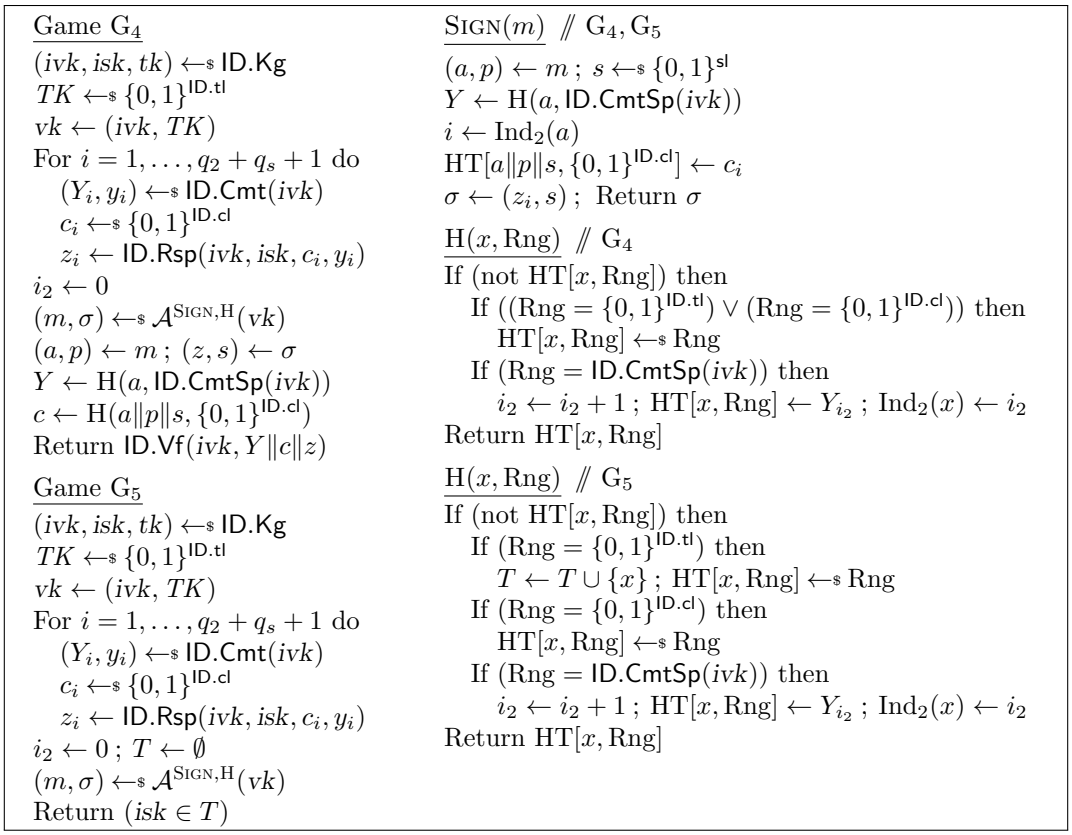


Figure 12: More games for the proof of Theorem 3.

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 verifies each candidate using the key-verification algorithm of the identification scheme, returning a successful candidate if one exists in its list. ▌

NECESSITY OF TRAPDOOR ID SCHEMES FOR DAPS. Trapdoor identification may seem a very particular assumption as a starting point for DAPS. However in Section 8 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.

7 Instantiation and implementation

We instantiate our general transform of Section 6 to obtain GQ-DAPS and CF-DAPS. We then make parameter choices and discuss our implementation and performance results.

Adversary $\mathcal{P}_1^{\text{Tr},\text{Ch},\text{Dec}}(ivk)$

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

Adversary $\mathcal{P}_2^{\text{Tr}}(ivk)$

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

Sign(m) // $\mathcal{P}_1, \mathcal{P}_2$

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

H(x, Rng) // $\mathcal{P}_1$

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

H(x, Rng) // $\mathcal{P}_2$

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

Figure 13: Adversaries for proof of Theorem 3.

Game $\text{OW}^{\mathcal{A}}_{\mathsf{RSA}}$	Game $\text{CF}^{\mathcal{A}}_{\mathsf{FG}}$	Game $\text{FAC}^{\mathcal{A}}_{\mathsf{MOD}}$
$(N, p, q, e, d) \leftarrow\!\!\$\ \mathsf{RSA}$	$(ek, ik) \leftarrow\!\!\$\ \mathsf{FG}$	$(N, p, q) \leftarrow\!\!\$\ \mathsf{MOD}$
$x \leftarrow\!\!\$\ \mathbb{Z}_N^*\ ;\ X \leftarrow x^e \bmod N$	$(x_0, x_1) \leftarrow\!\!\$\ \mathcal{A}(ek)$	$r \leftarrow\!\!\$\ \mathcal{A}(N)$
$x' \leftarrow\!\!\$\ \mathcal{A}(N, e, X)$	If $(x_0 \notin \mathsf{FG.D}_1(ek))$ or $(x_1 \notin \mathsf{FG.D}_1(ek))$	Return $(r \in \{p, q\})$
Return $(x' = x)$	then return false	
	$y_0 \leftarrow \mathsf{FG.Ev}_{ek,0}(x_0)$; $y_1 \leftarrow \mathsf{FG.Ev}_{ek,1}(x_1)$	
	Return $(y_0 = y_1)$	

Figure 14: Games defining one-wayness of RSA generator RSA, claw-freeness of claw-free TDF generator FG and factoring security of modulus generator MOD.

7.1 GQ-ID and GQ-DAPS

GQ-ID. 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 $\mathbf{Adv}^{\text{ow}}_{\mathsf{RSA}}(\mathcal{A}) = \Pr[\text{OW}^{\mathcal{A}}_{\mathsf{RSA}}]$ where the game is in Fig. 14.

Fig. 15 shows the GQ-ID identification scheme associated to RSA and a challenge length $l < k$ such that $\gcd(e, c) = 1$ for all $c \in \{0,1\}^l$ and all $(N, p, q, e, d) \in [\mathsf{RSA}]$. The commitment space is

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 $((N, e, X), x, d)$		$\xleftarrow{c}$	
	$z \leftarrow yx^c \mod N$	$\xrightarrow{z}$	$v \leftarrow (z^e \equiv YX^c \pmod N)$

GQ-ID.Ex$((N, e, X), Y, c_1, z_1, c_2, z_2)$	GQ-ID.KVf$((N, e, X), x)$
$z \leftarrow z_1 z_2^{-1} \mod N$	Return $(x^e \bmod N = X)$
$c \leftarrow (c_1 - c_2)$; $(a, b) \leftarrow \mathrm{egcd}(e, c)$	
$x \leftarrow X^a z^b \mod N$; Return x	GQ-ID.Cmt$^{-1}((N, e, X), d, Y)$
	$y \leftarrow Y^d \mod N$
	Return y

GQ-DAPS.Kg$^{\mathrm{H}}$	GQ-DAPS.Sig$^{\mathrm{H}}((N, e, X, TK), (x, d), m)$
$((N, e, X), x, d) \leftarrow_{\$} \mathsf{GQ\text{-}ID.Kg}$	$(a, p) \leftarrow m$; $s \leftarrow_{\$} \{0,1\}^{\mathsf{sl}}$
$TK \leftarrow d \oplus \mathrm{H}(x, \{0,1\}^k)$	$Y \leftarrow \mathrm{H}(a, \mathbb{Z}_N^*)$
Return $((N, e, X, TK), (x, d))$	$y \leftarrow_{\$} Y^d \bmod N$
GQ-DAPS.Ex$^{\mathrm{H}}((N, e, X, TK), m_1, m_2, \sigma_1, \sigma_2)$	$c \leftarrow \mathrm{H}(a \Vert p \Vert s, \{0,1\}^l)$
For $i = 1, 2$ do	$z \leftarrow yx^c \bmod N$
$(a_i, p_i) \leftarrow m_i$; $(z_i, s_i) \leftarrow \sigma_i$	$\sigma \leftarrow (z, s)$; Return σ
$Y_i \leftarrow \mathrm{H}(a_i, \mathbb{Z}_N^*)$	GQ-DAPS.Vf$^{\mathrm{H}}((N, e, X, TK), m, \sigma)$
$c_i \leftarrow \mathrm{H}(a_i \Vert p_i \Vert s_i, \{0,1\}^l)$	$(a, p) \leftarrow m$; $(z, s) \leftarrow \sigma$
$x \leftarrow \mathsf{GQ\text{-}ID.Ex}((N, e, X), Y_1, c_1, z_1, c_2, z_2)$	$Y \leftarrow \mathrm{H}(a, \mathbb{Z}_N^*)$
$d \leftarrow \mathrm{H}(x, \{0,1\}^k) \oplus TK$	$c \leftarrow \mathrm{H}(a \Vert p \Vert s, \{0,1\}^l)$
Return (x, d)	Return $(z^e \equiv YX^c \pmod N)$

Figure 15: Identification scheme GQ-ID associated to RSA generator RSA with modulus length k, and challenge length l, and GQ-DAPS $= \mathbf{H2}_+[\mathsf{GQ\text{-}ID}, \mathsf{sl}]$ derived via our transform from GQ-ID.

$\mathbb{Z}_N^*$. By egcd we denote the extended gcd algorithm that given relatively-prime inputs e, c returns a, b such that $ae + bc = 1$. Algorithm GQ-ID.Cmt^{-1} shows that this scheme is trapdoor.

If we want to establish cimp security, the first and natural route is to use known results. Thus let $\mathbf{Adv}^{\mathrm{imp}}_{\mathsf{ID}}(\mathcal{Q})$ denote the advantage of an adversary $\mathcal{Q}$ in violating the standard security against impersonation under passive attack as formalized in [1]. If $\mathcal{P}$ is a cimp adversary making q queries each to its Ch and Dec oracles then a standard hybrid argument shows how to build $\mathcal{Q}$ using the same number of Tr queries as $\mathcal{P}$ and with about the same running time so that $\mathbf{Adv}^{\mathrm{cimp}}_{\mathsf{ID}}(\mathcal{P}) \leq q \cdot \mathbf{Adv}^{\mathrm{imp}}_{\mathsf{ID}}(\mathcal{Q})$. The imp advantage is well studied, and using the standard bound and proof based on the reset lemma of [6] we would get a ow adversary $\mathcal{A}$ of comparable resources such that

$$\mathbf{Adv}^{\mathrm{cimp}}_{\mathsf{GQ\text{-}ID}}(\mathcal{P}_1) \leq q \cdot \left(\sqrt{\mathbf{Adv}^{\mathrm{ow}}_{\mathsf{RSA}}(\mathcal{A})} + \frac{1}{2^l} \right) . \tag{14}$$

The term in parentheses is from the reset lemma. This bound is poor, both due to the square root and due to the multiplication by q in the first term. The following says that we can do much better with a direct analysis, establishing both cimp and kr security with tight bounds.

Theorem 4 *Let* GQ-ID *be the identification scheme associated to RSA generator* RSA *with modulus*

length k and challenge length l as above. Let $\mathcal{P}_1$ be a cimp adversary making q queries each to its CH *and* DEC *oracles. Then from $\mathcal{P}_1$ we can construct ow adversary $\mathcal{A}$ such that*

$$\mathbf{Adv}^{\mathrm{cimp}}_{\mathsf{GQ\text{-}ID}}(\mathcal{P}_1) \le \mathbf{Adv}^{\mathrm{ow}}_{\mathsf{RSA}}(\mathcal{A}) + \frac{q}{2^l} \ . \tag{15}$$

The running time of $\mathcal{A}$ is that of $\mathcal{P}_1$ plus the overhead of one execution of the GQ-ID.Ex *algorithm. Also let $\mathcal{P}_2$ be a kr adversary. Then from $\mathcal{P}_2$ we can construct ow adversary $\mathcal{A}$ such that*

$$\mathbf{Adv}^{\mathrm{kr}}_{\mathsf{GQ\text{-}ID}}(\mathcal{P}_2) \le \mathbf{Adv}^{\mathrm{ow}}_{\mathsf{RSA}}(\mathcal{A}) \ . \tag{16}$$

The running time of $\mathcal{A}$ is that of $\mathcal{P}_2$.

The proof of Equation (15) is simple. In the execution of $\mathcal{P}_1$, if the impersonation is successful with a challenge different from the one in the transcript, then we can use GQ-ID.Ex to extract the secret key. The transcript queries can be simulated using the honest-verifier zero-knowledge property of GQ-ID. This doesn't add to the time overhead because by our convention running time refers to the execution of the adversary with the game, and in the real case the transcript oracle uses comparable time. The proof of Equation (16) is even more trivial since $\mathcal{P}_2$ returns the secret key directly. Thus we see that for these weaker goals we can not only establish tighter security but the proofs are much simpler.

GQ-DAPS. Fig. 15 shows the algorithms of our GQ-DAPS DAPS scheme derived by applying our transform to the trapdoor GQ-ID identification scheme of Fig. 15, where the latter is based on an RSA generator RSA with modulus length k and a challenge length $l < k$. It is thus parameterized by RSA, l and seed length sl. To estimate security for a given modulus length k we use Theorem 3 and Theorem 4. The reductions are tight in both cases and so we need to estimate the advantage of a time t adversary against the one-wayness of RSA. We do this under the assumption that the NFS is the best factoring method. Then taking into account Theorem 3 and Theorem 4, 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. 17 for implementation and performance information.

7.2 CF-ID and CF-DAPS

CF-ID. Our definition of a claw-free function generator extends the one of [16]. To be able to work with generators where membership in the domain cannot be efficiently tested, we have a superset of this domain in which such testing is possible. We also include a key-verification algorithm. Proceeding to the details, the generator FG specifies the following. Key-generation algorithm FG.Kg returns a tuple (ek, ik) consisting of an evaluation key ek and an inversion key ik. Associated to ek are finite sets $\mathsf{FG.D}_1(ek) \subseteq \mathsf{FG.D}_2(ek)$. Also specified are deterministic evaluation and inversion algorithms FG.Ev and $\mathsf{FG.Ev}^{-1}$. For $d \in \{0,1\}$, these in turn specify functions $\mathsf{FG.Ev}_{ek,d}$: $\mathsf{FG.D}_2(ek) \to \mathsf{FG.D}_1(ek)$ such that the restriction $\mathsf{FG.Ev}_{ek,d}$: $\mathsf{FG.D}_1(ek) \to \mathsf{FG.D}_1(ek)$ is a permutation with inverse $\mathsf{FG.Ev}^{-1}_{ik,d}$: $\mathsf{FG.D}_1(ek) \to \mathsf{FG.D}_1(ek)$. Membership in $\mathsf{FG.D}_2(ek)$ must be efficiently testable given ek. Membership in $\mathsf{FG.D}_1(ek)$ may not be efficiently testable given ek, but it should be possible to efficiently pick random elements from the set given 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. 14. The game tests membership in $\mathsf{FG.D}_1(ek)$ and hence may not be efficient but that's ok. 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 $(ek, x) \in [\mathsf{FG.Kg}]$.

$\underline{\mathsf{CF\text{-}ID.Kg}}$	**Prover**		**Verifier**
$(ek, ik) \leftarrow\!\!\$\ \mathsf{FG.Kg}$	Input: (ek, ik)		Input: ek
Return (ek, ik, ε)	$Y \leftarrow\!\!\$\ \mathsf{FG.D}_1(ek)$	$\xrightarrow{Y}$	$c \leftarrow\!\!\$\ \{0,1\}^l$
		$\xleftarrow{c}$	
	$z \leftarrow \mathsf{FG.Ev}^{-1}_{ek,0c}(Y)$	$\xrightarrow{z}$	If $z \notin \mathsf{FG.D}_2(ek)$ then $v \leftarrow \mathsf{false}$
			Else $v \leftarrow (\mathsf{FG.Ev}_{ek,0c}(z) = Y)$

$\underline{\mathsf{CF\text{-}ID.Ex}(ek, Y, c_1, z_1, c_2, z_2)}$	$\underline{\mathsf{CF\text{-}ID.KVf}(ek, x)}$
$z_1 \leftarrow \mathsf{FG.Ev}_{ek,0}(z_1)\ ;\ z_2 \leftarrow \mathsf{FG.Ev}_{ek,0}(z_2)$	Return $\mathsf{FG.KVf}(ek, x)$
For $i = 1, \ldots, l$ do	
$\quad z'_1 \leftarrow \mathsf{FG.Ev}_{ek,c_1[i]}(z_1)\ ;\ z'_2 \leftarrow \mathsf{FG.Ev}_{ek,c_2[i]}(z_2)$	$\underline{\mathsf{CF\text{-}ID.Cmt}^{-1}(ek, \varepsilon, Y)}$
$\quad$ If $(c_1[i] \neq c_2[i]) \wedge (z'_1 = z'_2)$ then	Return ε
$\quad\quad$ If $(c_1[i], c_2[i]) = (0,1)$ then Return $\mathsf{FG.Ex}(ek, z_1, z_2)$	
$\quad\quad$ If $(c_1[i], c_2[i]) = (1,0)$ then Return $\mathsf{FG.Ex}(ek, z_2, z_1)$	
$\quad z_1 \leftarrow z'_1\ ;\ z_2 \leftarrow z'_2$	
Return $\perp$	

$\underline{\mathsf{CF\text{-}DAPS.Kg}^{\mathrm{H}}}$	$\underline{\mathsf{CF\text{-}DAPS.Sig}^{\mathrm{H}}(ek, ik, m)}$
$(ek, ik, \varepsilon) \leftarrow\!\!\$\ \mathsf{CF\text{-}ID.Kg}$	$(a, p) \leftarrow m\ ;\ s \leftarrow\!\!\$\ \{0,1\}^{\mathsf{sl}}$
Return (ek, ik)	$Y \leftarrow \mathrm{H}(a, \mathsf{FG.D}_1(ek))$
$\underline{\mathsf{CF\text{-}DAPS.Ex}^{\mathrm{H}}(ek, m_1, m_2, \sigma_1, \sigma_2)}$	$c \leftarrow \mathrm{H}(a \| p \| s, \{0,1\}^l)$
For $i = 1, 2$ do	$z \leftarrow \mathsf{FG.Ev}^{-1}_{ek,0c}(Y)$
$\quad (a_i, p_i) \leftarrow m_i\ ;\ (z_i, s_i) \leftarrow \sigma_i$	$\sigma \leftarrow (z, s)\ ;$ Return σ
$\quad Y_i \leftarrow \mathrm{H}(a_i, \mathsf{FG.D}_1(ek))$	$\underline{\mathsf{CF\text{-}DAPS.Vf}^{\mathrm{H}}(ek, m, \sigma)}$
$\quad c_i \leftarrow \mathrm{H}(a_i \| p_i \| s_i, \{0,1\}^l)$	$(a, p) \leftarrow m\ ;\ (z, s) \leftarrow \sigma$
Return $\mathsf{CF\text{-}ID.Ex}(ek, Y_1, c_1, z_1, c_2, z_2)$	$c \leftarrow \mathrm{H}(a \| p \| s, \{0,1\}^l)$
	If $(z \notin \mathsf{FG.D}_2(ek))$ then return false
	Return $(\mathsf{FG.Ev}_{ek,0c}(z) = \mathrm{H}(a, \mathsf{FG.D}_1(ek)))$

Figure 16: Identification scheme $\mathsf{CF\text{-}ID}$ associated to claw-free function generator FG and challenge length l, and $\mathsf{CF\text{-}DAPS} = \mathbf{H2}_+[\mathsf{CF\text{-}ID}, \mathsf{sl}]$ derived via our transform from $\mathsf{CF\text{-}ID}$.

For a string $w = w[1] \ldots w[n] \in \{0,1\}^n$, let $\mathsf{FG.Ev}_{ek,w}$: $\mathsf{FG.D}_2(ek) \to \mathsf{FG.D}_1(ek)$ and $\mathsf{FG.Ev}^{-1}_{ik,w}$: $\mathsf{FG.D}_1(ek) \to \mathsf{FG.D}_1(ek)$ be defined for $x \in \mathsf{FG.D}_2(ek)$ and $y \in \mathsf{FG.D}_1(ek)$ 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

Fig. 16 shows the $\mathsf{CF\text{-}ID}$ identification scheme associated to FG and a challenge length l. The commitment space is $\mathsf{FG.D}_1(ek)$. By $0c$ we denote the challenge c prefixed with a 0 bit. This is done following [20] because the verifier may not be able to test membership of z in $\mathsf{FG.D}_1(ek)$, but now it can test membership in $\mathsf{FG.D}_2(ek)$ and use the fact that $\mathsf{FG.Ev}_{ek,0}(z) \in \mathsf{FG.D}_1(ek)$. The scheme is trivially trapdoor, with $\mathsf{CF\text{-}ID.Cmt}^{-1}$ returning ε. The key verification algorithm is the same as that of FG. The following summarizes the security properties of the scheme.

Theorem 5 *Let* $\mathsf{CF\text{-}ID}$ *be the identification scheme associated to function generator* FG *and challenge length* l *as above. Let* $\mathcal{P}_1$ *be a cimp adversary making* q *queries each to its* CH *and* DEC

oracles. Then from $\mathcal{P}_1$ we can construct cf adversary $\mathcal{A}$ such that

$$\mathbf{Adv}^{\text{cimp}}_{\mathsf{CF\text{-}ID}}(\mathcal{P}_1) \le \mathbf{Adv}^{\text{cf}}_{\mathsf{FG}}(\mathcal{A}) + \frac{q}{2^l} \; . \tag{17}$$

The running time of $\mathcal{A}$ is that of $\mathcal{P}_1$ plus the overhead of one execution of the CF-ID.Ex *algorithm. Also let $\mathcal{P}_2$ be a kr adversary. Then from $\mathcal{P}_2$ we can construct cf adversary $\mathcal{A}$ such that*

$$\mathbf{Adv}^{\text{kr}}_{\mathsf{CF\text{-}ID}}(\mathcal{P}_2) \le \mathbf{Adv}^{\text{cf}}_{\mathsf{FG}}(\mathcal{A}) \; . \tag{18}$$

The running time of $\mathcal{A}$ is that of $\mathcal{P}_2$.

Again the value here is that the reductions for these weaker security goals are tight. For the classical imp goal, the reductions are analogous to Equation (14). The proof of Theorem 5 is again simple. The identification protocol is honest-verifier zero-knowledge and this together with extractability establishes Equation (17). Equation (18) is even more direct.

CF-DAPS. Fig. 16 shows the algorithms of our CF-DAPS DAPS scheme derived by applying our transform to the trapdoor CF-ID identification scheme of Fig. 15, where the latter is based on an function generator FG and a challenge length l. It is thus parameterized by FG, l and seed length sl. Due to the tight reductions in Theorem 3 and Theorem 5, security will amount to that of the function generator. Our implementation will use a particular choice of the latter for speed, and we now discuss the schemes based on this choice.

MR-ID AND MR-DAPS. 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 $\mathbf{Adv}^{\text{fac}}_{\mathsf{MOD}}(\mathcal{A}) = \Pr[\text{FAC}^{\mathcal{A}}_{\mathsf{MOD}}]$ where the game is in Fig. 14.

We associate to a modulus generator MOD the particular function generator FG = FG[MOD] defined as follows. FG.Kg runs MOD to get (N, p, q) and then returns $(N, (N, p, q))$. We let $\mathsf{FG.D}_2(N) = \mathbb{Z}^*_N$, in which membership is efficiently testable given N. We let $\mathsf{FG.D}_1(N) = \text{QR}(N) = \{z^2 \mod N : z \in \mathbb{Z}^*_N\}$ be the subset of quadratic residues. Membership in $\text{QR}(N)$ is not known to be efficiently testable given only N, but one can sample a random point in it by picking $z \leftarrow\!\!\$\; \mathbb{Z}^*_N$ and returning $z^2 \bmod N$. For $x \in \mathbb{Z}^*_N$ let $\mathsf{FG.Ev}_{N,0}(x) = x^2 \bmod N$ and $\mathsf{FG.Ev}_{N,1}(x) = 4x^2 \bmod N$. Note that, since $p \equiv 3 \pmod 8$ and $q \equiv 7 \pmod 8$, we have that neither 2 nor -2 is a quadratic residue mod N. For $y \in \text{QR}(N)$ the inverses are defined as

$$\mathsf{FG.Ev}^{-1}_{(N,p,q),0}(y) = \sqrt{y} \mod N \quad \text{and} \quad \mathsf{FG.Ev}^{-1}_{(N,p,q),1}(y) = \sqrt{4^{-1}y} \mod N$$

where $\sqrt{z}$ denotes the square root of z that is itself a quadratic residue mod N. This can be efficiently computed given p, q. Extraction algorithm FG.Ex takes N, x_0, x_1 such that $x_0, x_1 \in \text{QR}(N)$ and $x_0^2 \equiv 4x_1^2 \bmod N$, which means $r = \gcd(x_0 - 2x_1, N)$ divides N. However $x_0, x_1 \in \text{QR}(N)$ and hence $x_0 \not\equiv \pm 2x_1 \pmod N$, so r is a non-trivial factor of N, and the algorithm can thus return (N, p, q). Key verification algorithm $\mathsf{FG.KVf}(N, (N, p, q))$ returns true if $N = pq$. Claw-freeness tightly reduces to factoring, so the latter is the basis of security.

When FG = FG[MOD], identification scheme CF-ID is an identification scheme specified by MR [20] as underlying the signature scheme of [19]. We denote this identification scheme by MR-ID. The corresponding CF-DAPS is denoted by MR-DAPS. This is the choice for our implementation. Given the tightness of the reduction of Theorem 3 and Theorem 5 we can again pick the modulus based on the assumption that the NFS is the best factoring method.

7.3 Implementation and performance

We implemented the MR-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. [4]

For the implementation of the claw-free TDF for MR-DAPS, we need to compute $\mathsf{FG.Ev}^{-1}_{(N,p,q),c}(x)$ 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 $\mathsf{FG.Ev}^{-1}_{(N,p,q),c}(x)$ with a constant number of exponentiations (assuming a small amount of precomputation which can be reused), thereby achieving an overall runtime of $O(k^3)$. Compute

$$\mathsf{FG.Ev}^{-1}_{(N,p,q),c}(x) = \frac{R_N(2^l, x)}{(R_N(2^l, 4))^{i(c)}} \mod N$$

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 $b = a^l \bmod \varphi(p)$ (the "inverse" of 2^l modulo $\varphi(p)$), and then computing $R_p(2^l, x)$ as $x^b \bmod p$.

To hash onto quadratic residues we follow the framework of Brier et al. for indifferentiable hashing [9] as described by Poettering and Stebila [23]: 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.

Timings were run on an Intel Core i7 (3720QM) with 4 cores each 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 17 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 MR-DAPS and GQ-DAPS are several orders of magnitude faster for both signing and verification. When using 2048-bit modulii, MR-DAPS signatures can be generated 336× and verified 116× faster, and GQ-DAPS signatures can be generated 198×

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

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 [23]	208.30	71.33	1024	164864	1009.88	271.36	2048	528384
GQ-DAPS (Fig. 15)	0.76	0.15	2048	1184	5.10	0.68	4096	2304
MR-DAPS (Fig. 16)	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 17: 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.

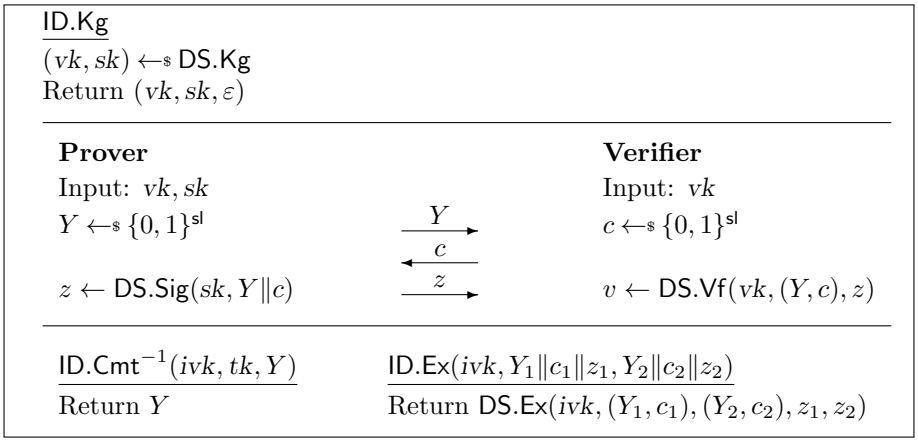


Figure 18: Our construction of a trapdoor identification scheme $\mathsf{ID} = \mathbf{H2}_{+}^{-\mathbf{1}}[\mathsf{DS}, \mathsf{sl}]$ from a DAPS DS and a seed length $\mathsf{sl} \in \mathbb{N}$. $\mathsf{ID.CmtSp}(ivk) = \{0,1\}^{\mathsf{sl}}$ for all ivk.

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 MR-DAPS or GQ-DAPS for signatures in digital certificates would incur little computational or size overhead relative to currently used signatures.

8 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 $\mathbf{H2}_{+}^{-\mathbf{1}}$ in Fig. 18, 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 $\perp$
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 19: Adversary for proof of Theorem 7.

"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 3. 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 6 on the identification scheme to obtain DAPS are in fact also necessary.

Theorem 6 *Let* DS *be a DAPS and let* $\mathsf{sl} \in \mathbb{N}$*. Then* $\mathsf{ID} = \mathbf{H2}_{+}^{-\mathbf{1}}[\mathsf{DS}, \mathsf{sl}]$ *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 $\mathsf{ID} = \mathbf{H2}_{+}^{-\mathbf{1}}[\mathsf{DS}, \mathsf{sl}]$, 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 7 *Let* DS *be a DAPS and let* $\mathsf{sl} \in \mathbb{N}$*. Let* $\mathcal{A}$ *be a mimp-adversary against* $\mathsf{ID} = \mathbf{H2}_{+}^{-\mathbf{1}}[\mathsf{DS}, \mathsf{sl}]$ *making* q *queries to its* Tr *oracle. Then from* $\mathcal{A}$ *we can construct uf-adversary* $\mathcal{A}_1$ *such that* $\mathbf{Adv}_{\mathsf{ID}}^{\text{cimp}}(\mathcal{A}) \leq \mathbf{Adv}_{\mathsf{DS}}^{\text{ruf}}(\mathcal{A}_1)$*.* $\mathcal{A}_1$ *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. 19. $\mathcal{A}_1$ 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 $\text{RUF}_{\mathsf{DS}}^{\mathcal{A}}$ 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 20: Adversary for proof of Theorem 8.

Theorem 8 *Let* DS *be a DAPS and let* $\mathsf{sl} \in \mathbb{N}$. *Let* $\mathcal{A}$ *be a ex-adversary against* $\mathsf{ID} = \mathbf{H2}_{+}^{-\mathbf{1}}[\mathsf{DS}, \mathsf{sl}]$. *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. 20. $\mathcal{A}_1$ directly calls $\mathcal{A}$ which is an ex adversary against the identification scheme ID. Note that, for $\mathsf{ID} = \mathbf{H2}_{+}^{-\mathbf{1}}[\mathsf{DS}, \mathsf{sl}]$, 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. ▌

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

[11] 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.

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

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

[14] 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.

[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] 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.

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

[21] 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.

[22] 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.

[23] 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.

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

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

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

[27] 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`.