Double-authentication-preventing signatures*

Bertram Poettering[1] and Douglas Stebila[2]

[1] Royal Holloway, University of London, United Kingdom
bertram.poettering@rhul.ac.uk
[2] Queensland University of Technology, Brisbane, Australia
stebila@qut.edu.au

Abstract. Digital signatures are often used by trusted authorities to make unique bindings between a subject and a digital object; for example, certificate authorities certify a public key belongs to a domain name, and time-stamping authorities certify that a certain piece of information existed at a certain time. Traditional digital signature schemes however impose no uniqueness conditions, so a trusted authority could make multiple certifications for the same subject but different objects, be it intentionally, by accident, or following a (legal or illegal) coercion. We propose the notion of a *double-authentication-preventing signature*, in which a value to be signed is split into two parts: a *subject* and a *message*. If a signer ever signs two different messages for the same subject, enough information is revealed to allow anyone to compute valid signatures on behalf of the signer. This double-signature forgeability property discourages signers from misbehaving—a form of *self-enforcement*—and would give binding authorities like CAs some cryptographic arguments to resist legal coercion. We give a generic construction using a new type of trapdoor functions with extractability properties, which we show can be instantiated using the group of sign-agnostic quadratic residues modulo a Blum integer.

Keywords: digital signatures, double signatures, dishonest signer, coercion, compelled certificate creation attack, self-enforcement, two-to-one trapdoor functions

1 Introduction

Digital signatures are used in several contexts by authorities who are trusted to behave appropriately. For instance, certificate authorities (CAs) in public key infrastructures, who assert that a certain public key belongs to a party with a certain identifier, are trusted to not issue fraudulent certificates for a domain name; time-stamping services, who assert that certain information existed at a certain point in time, are trusted to not retroactively certify information (they

* Parts of this work were funded by EPSRC Leadership Fellowship EP/H005455/1 and by European Commission ICT Programme Contract ICT-2007-216676 ECRYPT II (for BP), and by the Australian Technology Network and German Academic Exchange Service (ATN-DAAD) Joint Research Co-operation Scheme and Australian Research Council (ARC) Discovery Project DP130104304 (for DS).

should not "change the past"). In both of these cases, the authority is trusted to make a *unique* binding between a subject—a domain name or time—and a digital object—a public key or piece of information. However, traditional digital signatures provide no assurance of the uniqueness of this binding. As a result, an authority could make multiple bindings per subject.

Multiple bindings per subject can happen due to several reasons: poor management practices, a security breach, or coercion by external parties. Although there have been a few highly publicized certificate authority failures due to either poor management practices or security breaches, the vast majority of certificate authorities seem to successfully apply technological measures—including audited key generation ceremonies, secret sharing of signing keys, and use of hardware security modules—to securely and correctly carry out their role.

However, CAs have few tools to resist coercion, especially in the form of legal demands from governments. This was identified by Soghoian and Stamm [1] as the *compelled certificate creation attack*. For example, a certificate authority may receive a national security letter compelling it to assist in an investigation by issuing a second certificate for a specified domain name but containing the public key of the government agency, allowing the agency to impersonate Internet services to the target of the investigation. Regardless of one's opinions on the merits of these legal actions, they are a violation of the trust promised by certificate authorities: to never issue a certificate to anyone but the correct party. The extent to which legal coercion of CAs occurs is unknown, however there are indications that the technique is of interest to governments. A networking device company named Packet Forensics sells a device for eavesdropping on encrypted web traffic in which, reportedly, "users have the ability to import a copy of any legitimate key they obtain (potentially by court order)".[3] Various documents released by NSA contractor Edward Snowden in June–September 2013 indicate government interest in executing man-in-the-middle attacks on SSL users.[4]

Two certificates for the same domain signed by a single CA indeed constitute a cryptographic proof of fraud. However, in practice, it is currently up to the "market" to decide how to respond: the nature of the response depends on the scope and nature of the infraction and the CA's handling of the issue. The consequences that have been observed from real-world CA incidents range from minimal, such as the CA revoking the extra certificates amid a period of bad publicity (as in the 2011 Comodo incident[5]), up to the ultimate punishment for a CA on the web: removal of its root certificate from web browsers' lists of trusted CAs (as in the 2011 DigiNotar incident [2], which was found to have issued fraudulent certificates that were used against Iranian Internet users [3], and which lead to the bankruptcy of DigiNotar).

For a CA making business decisions on management and security practices, such consequences may be enough to convince it to invest in better systems. For a CA trying to resist a lawful order compelling it to issue a fraudulent certificate,

[3] http://www.wired.com/threatlevel/2010/03/packet-forensics/
[4] https://www.schneier.com/blog/archives/2013/09/new_nsa_leak_sh.html
[5] https://www.comodo.com/Comodo-Fraud-Incident-2011-03-23.html

such consequences may not be enough to convince a judge that it should not be compelled to violate the fundamental duty with which it was entrusted.

1.1 Contributions

We propose a new type of digital signature scheme for which the consequences of certain signer behaviours are unambiguous: any double signing, for any reason, leads to an immediate, irreversible, incontrovertible loss of confidence in the signature system. This "fragility" provides no room for mistakes, thereby encouraging "self-enforcement" of correct behaviour and allows a signer to make a more compelling argument resisting lawful coercion. If a CA fulfills a request to issue a double signature even to a lawful agency, the agency, by using the certificate, enables the attacked party to issue arbitrary certificates as well.

In a *double-authentication-preventing signature (DAPS)*, the data that is to be signed is split into two parts: a *subject* and a *message*. If a signer ever signs two messages for the same subject, then enough information is revealed for anyone to be able to forge signatures on arbitrary messages, rendering the signer immediately and irrevocably untrustworthy. More precisely, in addition to unforgeability we require a new security property for DAPS, *double-signature extractability*: from any two signatures on the same subject the signing key can be fully recovered. Depending on the nature of the subjects, an honest signer may need to track the list of subjects signed to avoid signing the same subject twice.

We give a generic construction for DAPS based on a new primitive called *extractable two-to-one trapdoor function* which allows anyone, given two preimages of the same value, to recover the trapdoor required for inverting the function. We show how to construct these functions using the group of sign-agnostic quadratic residues modulo a Blum integer (RSA modulus), an algebraic reformulation of a mathematical construction that has been used in several cryptographic primitives. The resulting DAPS scheme is efficient; with 1024-bit signing and verification keys, the signature size is about 20 KiB, and the runtime of our implementation using libgcrypt is about 0.3 s for signing and 0.1 s for verifying.

1.2 Related work

Certificate auditing and other techniques. Mechanisms such as Certificate Transparency[6] and others aim to identify malicious or incorrect CA behaviour by collecting and auditing public certificates. Incorrect behaviour, such as a CA issuing two certificates for the same domain name, can be identified and then presented as evidence possibly leading to a loss of trust. DAPS differs in that it provides an immediate and irrevocable loss of confidence and, importantly, provides a completely non-interactive solution.

Self-enforcement and traitor tracing. Dwork *et al.* [4] introduced the notion of *self-enforcement* in cryptography, in which the cryptosystem is designed to force

[6] http://www.certificate-transparency.org/

the user to keep the functionality private, that is, to not delegate or transfer the functionality to another user. There are a variety of techniques for ensuring self-enforcement: tradeoffs in efficiency [4] or by allowing recovering of some associated secret value with any delegated version of the secret information [5–7]. Broadcast encryption schemes often aim for a related notion, traitor tracing [8], in which the broadcaster aims to detect which of several receivers have used their private key to construct and distribute a pirate device; typically the broadcaster can identify which private key was leaked. DAPS differs from this line of research in that it does not aim to deter delegation or transferring of keys, rather it aims to deter a single party from performing a certain local operation (double signing).

Accountable IBE. Goyal [9] aimed to reduce trust in the key generation centre (KGC) in identity-based encryption.In accountable IBE, the key generation protocol between the user and the KGC results in one of a large number of possible keys being generated, and which one is generated is unknown to the KGC. Thus if the KGC issues a second key, it will with high probability be different, and the two different keys for the same identity serve as a proof that the KGC misbehaved. This effectively allows IBE to achieve the same level of detection as normal public key infrastructures: two certificates for the same subject serve as a proof that the CA misbehaved. However, neither approach has the stronger level of deterrence of DAPS: double signing leads to an immediate loss of confidence, rather than just proof of misbehaving for consideration of prosecution.

Digital cash. Digital cash schemes [10] often aim to detect double spending: a party who uses a token once maintains anonymity, but a party who uses a token twice reveals enough information for her identity to be recovered and traced. DAPS has some conceptual similarities, in that a party who signs two messages with the same subject reveals enough information for her secret key to be recovered. In both settings, double operations leak information, but double spending in digital cash typically leaks only an identity, whereas double signing in DAPS leaks the signer's private key. It is interesting to note that the number-theoretic structures our DAPS scheme builds on are similar to those used in early digital cash to provide double spending traceability [10]: both schemes use RSA moduli that can be factored if signers/spenders misbehave. However, there does not seem to be a direct connection between the primitives.

One-time signatures. One-time signatures, first proposed by Lamport using a construction based on hash functions [11], allow at most one message to be signed. Many instances can be combined using Merkle trees [12] to allow multiple signatures with just a single verification key, but key generation time becomes a function of the total number of signatures allowed. DAPS differs in that the number of messages to be signed need not be fixed a priori, and our construction relies on number-theoretic trapdoor functions, rather than solely hash functions.

Fail-stop signatures. Fail-stop signatures [13–17] allow a signer to prove to a judge that a forgery has occurred; a signer is protected against cryptanalytic attacks by even an unbounded adversary. Verifiers too are protected against computationally bounded signers who try to claim a signature is a forgery

when it is not. When a forgery is detected, generally the security of the scheme collapses, because some secret information can be recovered, and so the security of previous signatures is left in doubt. Forgery-resilient signatures [18] aim to have similar properties to fail-stop signatures—the ability for a signer to prove a cryptanalytic forgery—but discovery of a forgery does not immediately render previous signatures insecure. Both focus on an *honest* signer proving someone else has constructed a forgery, whereas DAPS is about what happens when a *dishonest* or *coerced* signer signs two messages for the same subject.

Chameleon hash functions. Chameleon hash functions [19] are trapdoor-based and randomized. Hashing is collision-resistant as long as only the public parameters are known. However, given the trapdoor and the message-randomness pair used to create a specific hash value, a collision for that value can be efficiently found. Some constructions allow the extraction of the trapdoor from any collision [20, 21]. However, it remains open how DAPS could be constructed from Chameleon hash functions.

2 Definitions

We now present our main definitions: a *double-authentication-preventing signature* and its security requirements: the standard (though slightly adapted) notion of *existential unforgeability*, as well as the new property of *signing key extractability* given two signatures on the same subject.

Notation. If S is a finite set, let $U(S)$ denote the uniform distribution on S and $x \leftarrow_R S$ denote sampling x uniformly from S. If A and B are two probability distributions, then notation $A \approx B$ denotes that the statistical distance between A and B is negligible. If $\mathcal{A}$ is a (probabilistic) algorithm, then $x \leftarrow_R \mathcal{A}^{\mathcal{O}}(y)$ denotes running $\mathcal{A}$ with input y on uniformly random coins with oracle access to $\mathcal{O}$, and setting x to be the output. We use $\mathcal{A}(y; r)$ to explicitly identify the random coins r on which the otherwise deterministic algorithm $\mathcal{A}$ is run.

Definition 1 (Double-authentication-preventing signature). *A* double-authentication-preventing signature (DAPS) *is a tuple of efficient algorithms* $(\mathsf{KGen}, \mathsf{Sign}, \mathsf{Ver})$ *as follows:*

- $\mathsf{KGen}(1^\lambda)$*: On input security parameter* 1^λ*, this algorithm outputs a signing key* sk *and a verification key* vk*.*
- $\mathsf{Sign}(\mathsf{sk}, \mathsf{subj}, \mathsf{msg})$*: On input signing key* sk *and subject/message pair* subj, $\mathsf{msg} \in \{0,1\}^*$*, this algorithm outputs a signature* σ*.*
- $\mathsf{Ver}(\mathsf{vk}, \mathsf{subj}, \mathsf{msg}, \sigma)$*: On input verification key* vk*, subject/message pair* subj, $\mathsf{msg} \in \{0,1\}^*$*, and candidate signature* σ*, this algorithm outputs* 0 *or* 1*.*

Definition 2 (Correctness). *A DAPS scheme is* correct *if, for all* $\lambda \in \mathbb{N}$*, for all key pairs* $(\mathsf{sk}, \mathsf{vk}) \leftarrow_R \mathsf{KGen}(1^\lambda)$*, for all* $\mathsf{subj}, \mathsf{msg} \in \{0,1\}^*$*, and for all signatures* $\sigma \leftarrow_R \mathsf{Sign}(\mathsf{sk}, \mathsf{subj}, \mathsf{msg})$*, we have that* $\mathsf{Ver}(\mathsf{vk}, \mathsf{subj}, \mathsf{msg}, \sigma) = 1$*.*

Our unforgeability notion largely coincides with the standard unforgeability notion for digital signature schemes [22]; the main difference is that, for DAPS,

$\mathbf{Exp}^{\mathrm{EUF}}_{\mathsf{DAPS},\mathcal{A}}(\lambda)$:

1. SignedList $\leftarrow \emptyset$
2. $(\mathsf{sk}, \mathsf{vk}) \leftarrow_R \mathsf{KGen}(1^\lambda)$
3. $(\mathsf{subj}^*, \mathsf{msg}^*, \sigma^*) \leftarrow_R \mathcal{A}^{\mathcal{O}_{\mathsf{Sign}}}(\mathsf{vk})$
   If $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{Sign}}(\mathsf{subj}, \mathsf{msg})$:
   (a) Append $(\mathsf{subj}, \mathsf{msg})$ to SignedList
   (b) $\sigma \leftarrow_R \mathsf{Sign}(\mathsf{sk}, \mathsf{subj}, \mathsf{msg})$
   (c) Return σ to $\mathcal{A}$
4. Return 1 iff all the following hold:
   – $\mathsf{Ver}(\mathsf{vk}, \mathsf{subj}^*, \mathsf{msg}^*, \sigma^*) = 1$
   – $(\mathsf{subj}^*, \mathsf{msg}^*) \notin$ SignedList
   – $\forall\, \mathsf{subj}, \mathsf{msg}_0, \mathsf{msg}_1$:
     if $(\mathsf{subj}, \mathsf{msg}_0), (\mathsf{subj}, \mathsf{msg}_1) \in$ SignedList
     then $\mathsf{msg}_0 = \mathsf{msg}_1$

$\mathbf{Exp}^{\mathrm{DSE}}_{\mathsf{DAPS},\mathcal{A}}(\lambda)$:

1. $(\mathsf{vk}, (S_1, S_2)) \leftarrow_R \mathcal{A}(1^\lambda)$
2. $\mathsf{sk}' \leftarrow_R \mathsf{Extract}(\mathsf{vk}, (S_1, S_2))$
3. Return 1 iff all the following hold:
   – (S_1, S_2) is compromising
   – sk' is not the signing key corresponding to vk

$\mathbf{Exp}^{\mathrm{DSE}^*}_{\mathsf{DAPS},\mathcal{A}}(\lambda)$:

1. $(\mathsf{sk}, \mathsf{vk}) \leftarrow_R \mathsf{KGen}(1^\lambda)$
2. $(S_1, S_2) \leftarrow_R \mathcal{A}(\mathsf{sk}, \mathsf{vk})$
3. $\mathsf{sk}' \leftarrow_R \mathsf{Extract}(\mathsf{vk}, (S_1, S_2))$
4. Return 1 iff all the following hold:
   – (S_1, S_2) is compromising
   – $\mathsf{sk}' \neq \mathsf{sk}$

Fig. 1. Security experiments for DAPS: unforgeability and double signature extractability (without and with trusted setup).

forgeries crafted by the adversary are not considered valid if the adversary has requested forgeries on different messages for the same subject.

Definition 3 (Existential unforgeability). *A DAPS scheme is* existentially unforgeable under adaptive chosen message attacks *if, for all efficient adversaries $\mathcal{A}$, the success probability* $\mathbf{Succ}^{\mathrm{EUF}}_{\mathsf{DAPS},\mathcal{A}}(\lambda) := \Pr[\mathbf{Exp}^{\mathrm{EUF}}_{\mathsf{DAPS},\mathcal{A}}(\lambda) = 1]$ *in the* EUF *experiment of Figure 1 is a negligible function.*

Although Definition 3 ensures that signatures of DAPS are generally unforgeable, we do want signatures to be forgeable in certain circumstances. In fact we aim at an even higher goal: when two different messages have been signed for the same subject, the signing key should leak from the two signatures. The notion of *compromising pairs of signatures* makes this condition precise.

Definition 4 (Compromising pair of signatures). *For a fixed verification key* vk*, a pair* (S_1, S_2) *of subject/message/signature triples* $S_1 = (\mathsf{subj}_1, \mathsf{msg}_1, \sigma_1)$ *and* $S_2 = (\mathsf{subj}_2, \mathsf{msg}_2, \sigma_2)$ *is* compromising *if* σ_1, σ_2 *are valid signatures on different messages for the same subject; that is, if* $\mathsf{Ver}(\mathsf{vk}, \mathsf{subj}_1, \mathsf{msg}_1, \sigma_1) = 1$*,* $\mathsf{Ver}(\mathsf{vk}, \mathsf{subj}_2, \mathsf{msg}_2, \sigma_2) = 1$*,* $\mathsf{subj}_1 = \mathsf{subj}_2$*, and* $\mathsf{msg}_1 \neq \mathsf{msg}_2$*.*

We now define the double-signature extractability requirement. Here, the adversary takes the role of a malicious signer that aims to generate compromising pairs of signatures that do not lead to successful signing key extraction. We consider two scenarios: the *trusted setup model*, where key generation is assumed to proceed honestly, and the *untrusted setup model*, where the adversary has full control over key generation as well.

Definition 5 (Double-signature extractability). *A double-authentication-preventing signature* DAPS *is* double-signature extractable *(resp.* with trusted setup*) if an efficient algorithm*

– $\mathsf{Extract}(\mathsf{vk}, (S_1, S_2))$*: On input verification key* vk *and compromising pair* (S_1, S_2)*, this algorithm outputs a signing key* sk'*.*

is known such that, for all efficient adversaries $\mathcal{A}$*, the probability* $\mathbf{Succ}^{\mathrm{DSE}^{(*)}}_{\mathsf{DAPS},\mathcal{A}}(\lambda) := \Pr[\mathbf{Exp}^{\mathrm{DSE}^{(*)}}_{\mathsf{DAPS},\mathcal{A}}(\lambda) = 1]$ *of success in the* DSE *(resp.* DSE^**) experiment of Figure 1 is a negligible function in* λ*.*

The DSE experiment assumes existence of an efficient predicate that verifies that a candidate sk' is *the* signing key corresponding to a verification key. In some schemes, there may be several signing keys that correspond to a verification key or it may be inefficient to check. However, for the scheme presented in Section 5, when instantiated with the factoring-based primitive of Section 4, it is easy to check that a signing key (p, q) corresponds to a verification key n; note that there is a canonical representation of such signing keys (take $p < q$).

3 2:1 trapdoor functions and extractability

We introduce the concept of 2:1 trapdoor functions (2:1-TDF). At a high level, such functions are *trapdoor one-way functions*, meaning that they should be hard to invert except with knowledge of a trapdoor. They are *two-to-one*, meaning that the domain is exactly twice the size of the range, and every element of the range has precisely two preimages. We also describe an additional property, *extractability*, which means that given two distinct preimages of an element of the range, the trapdoor can be computed.

Consider two finite sets, A and B, such that A is twice the size of B. Let $f : A \to B$ be a surjective function such that, for any element $b \in B$, there are exactly two preimages in A; f is not injective, so the inverse function does not exist. Define instead $f^{-1} : B \times \{0, 1\} \to A$ such that for each $b \in B$ the two preimages under f are given by $f^{-1}(b, 0)$ and $f^{-1}(b, 1)$. This partitions set A into two subsets $A_0 = f^{-1}(B, 0)$ and $A_1 = f^{-1}(B, 1)$ of the same size.

Function f is a 2:1-TDF if the following additional properties hold: sets A_0, A_1, and B are efficiently samplable, function f is efficiently computable, and inverse function f^{-1} is hard to compute unless some specific trapdoor information is known. We finally require an extraction capability: there should be an efficient way to recover the trapdoor for the computation of f^{-1} from any two elements $a_0 \neq a_1$ with $f(a_0) = f(a_1)$ (we will also write $a_0 \overset{f}{\sim} a_1$ for such configurations). The setting of 2:1-TDFs is illustrated in Figure 2. We will formalize the functionality and security properties below.

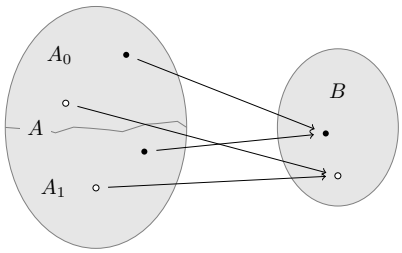

Fig. 2. Illustration of a 2:1 trapdoor function $f : A \to B$. Each element of B has exactly two preimages, one in A_0 and one in A_1.

3.1 Definition

We give a formal definition of 2:1-TDF and its correctness, and establish afterwards that it implements the intuition developed above.

Definition 6 (2:1 trapdoor function). *A* 2:1 trapdoor function (2:1-TDF) *is a tuple of efficient algorithms* $(\mathsf{TdGen}, \mathsf{Apply}, \mathsf{Reverse}, \mathsf{Decide})$ *as follows:*

- $\mathsf{TdGen}(1^\lambda)$*: On input security parameter* 1^λ*, this randomized algorithm outputs a pair* $(\mathsf{td}, \mathsf{pub})$*, where* td *is a trapdoor and* pub *is some associated public information. Each possible outcome* pub *implicitly defines finite sets* $A = A(\mathsf{pub})$ *and* $B = B(\mathsf{pub})$*.*
- $\mathsf{Apply}(\mathsf{pub}, a)$*: On input public information* pub *and element* $a \in A(\mathsf{pub})$*, this deterministic algorithm outputs an element* $b \in B(\mathsf{pub})$*.*
- $\mathsf{Reverse}(\mathsf{td}, b, d)$*: On input trapdoor* td*, element* $b \in B(\mathsf{pub})$*, and bit* $d \in \{0,1\}$*, this deterministic algorithm outputs an element* $a \in A(\mathsf{pub})$*.*
- $\mathsf{Decide}(\mathsf{pub}, a)$*: On input public information* pub *and element* $a \in A(\mathsf{pub})$*, this deterministic algorithm outputs a bit* $d \in \{0,1\}$*.*

Definition 7 (Correctness of 2:1-TDF). *A 2:1-TDF is* correct *if, for all* $(\mathsf{td}, \mathsf{pub}) \leftarrow_R \mathsf{TdGen}$*, all* $d \in \{0,1\}$*, all* $a \in A(\mathsf{pub})$*, and all* $b \in B(\mathsf{pub})$*, we have that (1)* $a \in \mathsf{Reverse}(\mathsf{td}, \mathsf{Apply}(\mathsf{pub}, a), \{0,1\})$*, (2)* $\mathsf{Apply}(\mathsf{pub}, \mathsf{Reverse}(\mathsf{td}, b, d)) = b$*, and (3)* $\mathsf{Decide}(\mathsf{pub}, \mathsf{Reverse}(\mathsf{td}, b, d)) = d$*.*

Let $(\mathsf{td}, \mathsf{pub})$ be output by TdGen. Consider partition $A(\mathsf{pub}) = A_0(\mathsf{pub}) \dot\cup A_1(\mathsf{pub})$ obtained by setting $A_d(\mathsf{pub}) = \{a \in A(\mathsf{pub}) : \mathsf{Decide}(\mathsf{pub}, a) = d\}$, for $d \in \{0,1\}$. It follows from correctness requirement (3) that function $\psi_d := \mathsf{Reverse}(\mathsf{td}, \cdot, d)$ is a mapping $B(\mathsf{pub}) \to A_d(\mathsf{pub})$. Note that ψ_d is surjective by condition (1), and injective by condition (2). Hence, we have bijections $\psi_0 : B(\mathsf{pub}) \to A_0(\mathsf{pub})$ and $\psi_1 : B(\mathsf{pub}) \to A_1(\mathsf{pub})$. Thus, $|A_0(\mathsf{pub})| = |A_1(\mathsf{pub})| = |B(\mathsf{pub})| = |A(\mathsf{pub})|/2$.

Define now relation $\overset{\times}{\sim} \subseteq A(\mathsf{pub}) \times A(\mathsf{pub})$ such that

$$a \overset{\times}{\sim} a' \iff \mathsf{Apply}(\mathsf{pub}, a) = \mathsf{Apply}(\mathsf{pub}, a') \wedge \mathsf{Decide}(\mathsf{pub}, a) \neq \mathsf{Decide}(\mathsf{pub}, a').$$

Note that for each $a \in A(\mathsf{pub})$ there exists exactly one $a' \in A(\mathsf{pub})$ such that $a \overset{\times}{\sim} a'$; indeed, if $a \in A_d(\mathsf{pub})$, then $a' = \psi_{1-d}(\psi_d^{-1}(a)) \in A_{1-d}(\mathsf{pub})$. Observe how algorithms Apply and $\mathsf{Reverse}$ correspond to functions $f : A \to B$ and $f^{-1} : B \times \{0,1\} \to A$ discussed at the beginning of Section 3.

We next extend the functionality of 2:1-TDFs to include extraction of the trapdoor: knowledge of any two elements $a_0, a_1 \in A$ with $a_0 \neq a_1 \wedge f(a_0) = f(a_1)$ shall immediately reveal the system's inversion trapdoor.

Definition 8 (Extractable 2:1-TDF). *A 2:1-TDF is* extractable *if an efficient algorithm*

- $\mathsf{Extract}(\mathsf{pub}, a, a')$*: On input public information* pub *and* $a, a' \in A(\mathsf{pub})$*, this algorithm outputs a trapdoor* td^**.*

is known such that, for all $(\mathsf{td}, \mathsf{pub})$ *output by* TdGen *and all* $a, a' \in A(\mathsf{pub})$ *with* $a \overset{\times}{\sim} a'$*, we have* $\mathsf{Extract}(\mathsf{pub}, a, a') = \mathsf{td}$*.*

$\mathbf{Exp}^{\text{INV-1}}_{X,\mathcal{A}}(\lambda)$:
1. $(\mathsf{td}, \mathsf{pub}) \leftarrow_R \mathsf{TdGen}(1^\lambda)$
2. $b \leftarrow_R B(\mathsf{pub})$
3. $a \leftarrow_R \mathcal{A}(\mathsf{pub}, b)$
4. Return 1 iff $\mathsf{Apply}(\mathsf{pub}, a) = b$

$\mathbf{Exp}^{\text{INV-2}}_{X,\mathcal{B}}(\lambda)$:
1. $(\mathsf{td}, \mathsf{pub}) \leftarrow_R \mathsf{TdGen}(1^\lambda)$
2. $a \leftarrow_R A(\mathsf{pub})$
3. $a' \leftarrow_R \mathcal{B}(\mathsf{pub}, a)$
4. Return 1 iff $a \overset{\sim}{\sim} a'$

Fig. 3. Security experiments for (second) preimage resistance of 2:1-TDF X.

3.2 Security notions

We proceed with the specification of the principal security property of 2:1-TDFs: one-wayness. Intuitively, it should be infeasible to find preimages and second preimages of the Apply algorithm without knowing the corresponding trapdoor.

Definition 9 (Preimage resistance of 2:1-TDF). *A 2:1-TDF X is* preimage resistant *if* $\mathbf{Succ}^{\text{INV-1}}_{X,\mathcal{A}}(\lambda) := \Pr[\mathbf{Exp}^{\text{INV-1}}_{X,\mathcal{A}}(\lambda) = 1]$ *and* second preimage resistant *if* $\mathbf{Succ}^{\text{INV-2}}_{X,\mathcal{B}}(\lambda) := \Pr[\mathbf{Exp}^{\text{INV-2}}_{X,\mathcal{B}}(\lambda) = 1]$ *are respectively negligible functions in λ, for all efficient adversaries $\mathcal{A}$ and $\mathcal{B}$, where $\mathbf{Exp}^{\text{INV-1}}_{X,\mathcal{A}}$ and $\mathbf{Exp}^{\text{INV-2}}_{X,\mathcal{B}}$ are as in Figure 3.*

As expected, second preimage resistance implies preimage resistance. Perhaps more surprising is that notions INV-1 and INV-2 are equivalent for extractable 2:1-TDFs. The proofs of the following lemmas appear in the full version [23].

Lemma 1 (INV-2 $\Rightarrow$ INV-1)**.** *Let X be a 2:1-TDF and let $\mathcal{A}$ be an efficient algorithm for the* INV-1 *experiment. Then there exist an efficient algorithm $\mathcal{B}$ for the* INV-2 *experiment such that* $\mathbf{Succ}^{\text{INV-1}}_{X,\mathcal{A}}(\lambda) \leq 2 \cdot \mathbf{Succ}^{\text{INV-2}}_{X,\mathcal{B}}(\lambda)$.

Lemma 2 (INV-1 $\Rightarrow$ INV-2 **for extractable 2:1-TDF).** *Let X be an extractable 2:1-TDF and let $\mathcal{B}$ be an efficient algorithm for the* INV-2 *experiment. Then there exists an efficient algorithm $\mathcal{A}$ for the* INV-1 *experiment such that* $\mathbf{Succ}^{\text{INV-2}}_{X,\mathcal{B}}(\lambda) = \mathbf{Succ}^{\text{INV-1}}_{X,\mathcal{A}}(\lambda)$.

4 Constructing extractable 2:1 trapdoor functions

Having introduced 2:1-TDFs and extractable 2:1-TDFs, we now show how to construct these primitives: we propose an efficient extractable 2:1-TDF and prove it secure, assuming hardness of the integer factorization problem.

Our construction builds on a specific structure from number theory, the *group of sign-agnostic quadratic residues*. This group was introduced to cryptography by Goldwasser, Micali, and Rivest in [22], and rediscovered 20 years later by Hofheinz and Kiltz [24]. We first reproduce the results of [22,24] and then extend them towards our requirements.[7]

[7] Goldwasser *et al.* gave no name to this group; Hofheinz and Kiltz called it the group of *signed quadratic residues*, but this seems to be a misnomer as the whole point is to *ignore the sign*, taking absolute values and forcing the elements to be between 0 and $(n-1)/2$; hence our use of the term *sign-agnostic*.

In our exposition, we assume that the reader is familiar with properties of $\mathbb{Z}_n^\times$ (the mutiplicative group of integers modulo n), J_n (the subgroup of $\mathbb{Z}_n^\times$ with Jacobi symbol equal to 1), and QR_n (quadratic residues modulo n), for Blum integers n. If we additionally define $\overline{J}_n = \mathbb{Z}_n^\times \setminus J_n$ and $\overline{QR}_n = J_n \setminus QR_n$, these five sets are related to each other as visualized in Figure 4 (left). Also illustrated is the action of the squaring operation: it is 4:1 from $\mathbb{Z}_n^\times$ to QR_n, 2:1 from J_n to QR_n, and 1:1 (i.e., bijective) from QR_n to QR_n. For reference, we reproduce all number-theoretic details relevant to this paper in the full version [23].

4.1 Sign-agnostic quadratic residues

For an RSA modulus n, it is widely believed that efficiently distinguishing elements in QR_n from elements in $\overline{QR}_n$ is a hard problem. It also seems to be infeasible to sample elements from QR_n without knowing a square root of the samples, or to construct hash functions that map to QR_n and could be modeled as random oracles. However, such properties are a prerequisite in certain applications in cryptography [24], which renders group QR_n unsuitable for such cases. As we see next, by switching from the group of quadratic residues modulo n to the related group of *sign-agnostic quadratic residues* modulo n, sampling and hashing becomes feasible.

The use of sign-agnostic quadratic residues in cryptography is explicitly proposed in [22, 24]. However, some aspects of the algebraical structure of this group are concealed in both works by the fact that the group operation is defined to act directly on specific *representations* of elements. In the following paragraphs we use a new and more consistent notation that aims at making the algebraical structure more readily apparent.

Let $(H, \cdot)$ be an arbitrary finite abelian group that contains an element $T \in H \setminus \{1\}$ such that $T^2 = 1$. Then $\{1, T\}$ is a (normal) subgroup in H, that is, quotient group $H/_{\{1,T\}}$ is well-defined, $\psi : H \to H/_{\{1,T\}} : x \mapsto \{x, Tx\}$ is a group homomorphism, and $|\psi(H)| = |H/_{\{1,T\}}| = |H|/2$ holds. Further, for all subgroups $G \leq H$ we have that $\psi(G) \leq \psi(H) = H/_{\{1,T\}}$. In such cases, if G is such that $T \in G$, then $|\psi(G)| = |G/_{\{1,T\}}| = |G|/2$ as above; otherwise, if $T \notin G$, then $|\psi(G)| = |G|$ and thus $\psi(G) \cong G$.

Consider now the specific group $H = \mathbb{Z}_n^\times$, for a Blum integer n. Then $T = -1$ has order 2 in $\mathbb{Z}_n^\times$ and above observations apply, with mapping $\psi : x \mapsto \{x, -x\}$. For any subgroup $G \leq \mathbb{Z}_n^\times$, let $G/_{\pm 1} := \psi(G)$. For subgroup $QR_n \leq \mathbb{Z}_n^\times$, as $-1 \notin QR_n$, we have $QR_n/_{\pm 1} \cong QR_n$ and thus $|QR_n/_{\pm 1}| = \varphi(n)/4$. Moreover, as $J_n \leq \mathbb{Z}_n^\times$ and $-1 \in J_n$, we have $|J_n/_{\pm 1}| = |J_n|/2 = \varphi(n)/4$. Similarly we see $|\mathbb{Z}_n^\times/_{\pm 1}| = \varphi(n)/2$. After setting $\overline{QR}_n/_{\pm 1} := (\mathbb{Z}_n^\times/_{\pm 1}) \setminus (QR_n/_{\pm 1})$ we finally obtain $|\overline{QR}_n/_{\pm 1}| = \varphi(n)/4$.

Note that we just observed $QR_n/_{\pm 1} \leq J_n/_{\pm 1} \leq \mathbb{Z}_n^\times/_{\pm 1}$ and $|QR_n/_{\pm 1}| = \varphi(n)/4 = |J_n/_{\pm 1}|$. The overall structure is hence $QR_n/_{\pm 1} = J_n/_{\pm 1} \lneq \mathbb{Z}_n^\times/_{\pm 1}$, as illustrated in Figure 4 (right). After agreeing on notations $\{\pm x\} = \{x, -x\}$ and $\{\pm x\}^2 = \{\pm(x^2)\}$ we obtain the following (proven in the full version [23]):

Lemma 3. *Let n be a Blum integer, then $QR_n/_{\pm 1} = \left\{\{\pm x\}^2 : \{\pm x\} \in \mathbb{Z}_n^\times/_{\pm 1}\right\}$.*

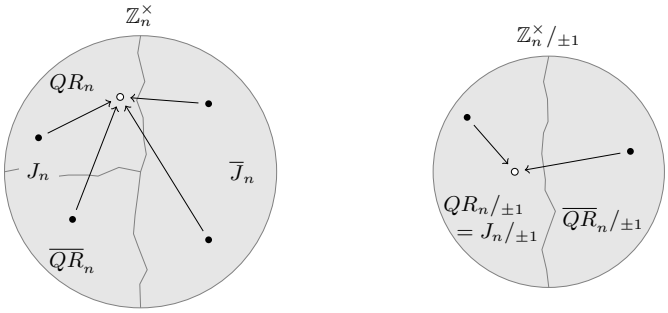

Fig. 4. Illustration of $\mathbb{Z}_n^\times$ and $\mathbb{Z}_n^\times/_{\pm 1}$ (for Blum integers n), and subgroups QR_n, J_n, and $J_n/_{\pm 1} = QR_n/_{\pm 1}$. Also visualized is the action of the squaring operation.

Moreover, by exploiting identity $QR_n/_{\pm 1} = J_n/_{\pm 1}$, we directly get the following characterizations of $QR_n/_{\pm 1}$ and $\overline{QR}_n/_{\pm 1}$. Observe that the sets are well-defined since $\left(\frac{x}{n}\right) = \left(\frac{-x}{n}\right)$ for all $x \in \mathbb{Z}_n^\times$.

$$QR_n/_{\pm 1} = \left\{\{\pm x\} \in \mathbb{Z}_n^\times/_{\pm 1} : \left(\tfrac{x}{n}\right) = +1\right\} \tag{1}$$

$$\overline{QR}_n/_{\pm 1} = \left\{\{\pm x\} \in \mathbb{Z}_n^\times/_{\pm 1} : \left(\tfrac{x}{n}\right) = -1\right\} \quad . \tag{2}$$

Many facts on the structure of $\mathbb{Z}_n^\times$ can be lifted to $\mathbb{Z}_n^\times/_{\pm 1}$. This holds in particular for the following five lemmas and corollaries, which we prove in the full version [23]. We stress that the following results do not appear in [22, 24].

Lemma 4 (Square roots in $\mathbb{Z}_n^\times/_{\pm 1}$). *Let n be a Blum integer. Every element $\{\pm y\} \in QR_n/_{\pm 1}$ has exactly two square roots in $\mathbb{Z}_n^\times/_{\pm 1}$. More precisely, there exist unique $\{\pm x_0\} \in QR_n/_{\pm 1}$ and $\{\pm x_1\} \in \overline{QR}_n/_{\pm 1}$ such that $\{\pm x_0\}^2 = \{\pm y\} = \{\pm x_1\}^2$. The factorization of n can readily be recovered from such pairs $\{\pm x_0\}, \{\pm x_1\}$: non-trivial divisors of n are given by $\gcd(n, x_0 - x_1)$ and $\gcd(n, x_0 + x_1)$. Square roots in $\mathbb{Z}_n^\times/_{\pm 1}$ can be efficiently computed if the factors of $n = pq$ are known.*

Corollary 1 (Squaring in $\mathbb{Z}_n^\times/_{\pm 1}$, $QR_n/_{\pm 1}$, $\overline{QR}_n/_{\pm 1}$). *Let n be a Blum integer. The squaring operation $\mathbb{Z}_n^\times/_{\pm 1} \to QR_n/_{\pm 1} : \{\pm x\} \mapsto \{\pm x\}^2$ is a 2:1 mapping. Moreover, squaring is a 1:1 function from $QR_n/_{\pm 1}$ to $QR_n/_{\pm 1}$ and from $\overline{QR}_n/_{\pm 1}$ to $QR_n/_{\pm 1}$. These relations are illustrated in Figure 4 (right).*

Lemma 5 (Computing square roots in $\mathbb{Z}_n^\times/_{\pm 1}$ is hard). *Let n be a Blum integer. Computing square roots in $\mathbb{Z}_n^\times/_{\pm 1}$ is as hard as factoring n.*

Lemma 6 (Samplability and decidability). *Let n be a Blum integer and $t \in \mathbb{Z}_n^\times$ be fixed with $\left(\frac{t}{n}\right) = -1$. The algorithm that samples $a \leftarrow_R \mathbb{Z}_n$ and returns $\{\pm a\}$ generates a distribution that is statistically indistinguishable from uniform on $\mathbb{Z}_n^\times/_{\pm 1}$. If the algorithm is modified such that it returns $\{\pm a\}$ if $\left(\frac{a}{n}\right) = +1$ and $\{\pm ta\}$ if $\left(\frac{a}{n}\right) = -1$, then the output is statistically indistinguishable from*

uniform on $QR_n/_{\pm 1}$. $\overline{QR}_n/_{\pm 1}$ can be sampled correspondingly. Sets $QR_n/_{\pm 1}$ and $\overline{QR}_n/_{\pm 1}$ are efficiently decidable (within $\mathbb{Z}_n^\times/_{\pm 1}$) by equations (1) and (2).

Lemma 7 (Indifferentiable hashing into $QR_n/_{\pm 1}$). *Let $H' : \{0,1\}^* \to J_n$ denote a hash function that is indifferentiable from a random oracle (see the full version [23] on how to construct one). Consider auxiliary function $G : J_n \to QR_n/_{\pm 1} : y \mapsto \{\pm y\}$ and let $H = G \circ H'$. Then $H : \{0,1\}^* \to QR_n/_{\pm 1}$ is indifferentiable as well.*

Remark 1 (Representation of elements). An efficient and compact way to represent elements $\{\pm x\} \in \mathbb{Z}_n^\times/_{\pm 1}$ is by the binary encoding of $\overline{x} = \min\{x, n-x\} \in [1, (n-1)/2]$, as proposed by [22]. The decoding procedure is $\overline{x} \mapsto \{\overline{x}, -\overline{x}\}$.

4.2 Constructing a 2:1-TDF from sign-agnostic quadratic residues

We use the tools from Section 4.1 to construct a factoring-based extractable 2:1-TDF, which will map $\mathbb{Z}_n^\times/_{\pm 1} \to QR_n/_{\pm 1}$. While the Apply algorithm corresponds to squaring, extractability is possible given distinct square roots of an element.

Construction 1 (Blum-2:1-TDF) *Define algorithms* Blum-2:1-TDF = (TdGen, Apply, Reverse, Decide, Extract) *as follows:*

- $\mathsf{TdGen}(1^\lambda)$*: Pick random Blum integer $n = pq$ of length λ such that $p < q$. Pick $t \in \mathbb{Z}_n^\times$ with $\left(\frac{t}{n}\right) = -1$. Return* $\mathsf{pub} \leftarrow (n, t)$ *and* $\mathsf{td} \leftarrow (p, q)$*. We will use sets $A_0(\mathsf{pub}) := QR_n/_{\pm 1}$, $A_1(\mathsf{pub}) := \overline{QR}_n/_{\pm 1}$, $A(\mathsf{pub}) := \mathbb{Z}_n^\times/_{\pm 1}$, and $B(\mathsf{pub}) := QR_n/_{\pm 1}$.*
- $\mathsf{Apply}(\mathsf{pub}, \{\pm a\})$*: Return $\{\pm b\} \leftarrow \{\pm a\}^2$.*
- $\mathsf{Reverse}(\mathsf{td}, \{\pm b\}, d)$*: By Lemma 4, element $\{\pm b\} \in QR_n/_{\pm 1}$ has exactly two square roots: $\{\pm a_0\} \in QR_n/_{\pm 1}$ and $\{\pm a_1\} \in \overline{QR}_n/_{\pm 1}$. Return $\{\pm a_d\}$.*
- $\mathsf{Decide}(\mathsf{pub}, \{\pm a\})$*: Return 0 if $\{\pm a\} \in QR_n/_{\pm 1}$; otherwise return 1.*
- $\mathsf{Extract}(\mathsf{pub}, \{\pm a_0\}, \{\pm a_1\})$*: Both $\gcd(n, a_0 - a_1)$ and $\gcd(n, a_0 + a_1)$ are non-trivial factors of $n = pq$. Return $\mathsf{td}^* \leftarrow (p, q)$ such that $p < q$.*

These algorithms are all efficient. Correctness of Blum-2:1-TDF and the security properties follow straightforwardly from the number-theoretic facts established in Sections 4.1; a formal proof appears in the full version [23]. Observe that the samplability of sets A, A_0, A_1, B is warranted by Lemma 6.

Theorem 1 (Security and extractability of Blum-2:1-TDF). Blum-2:1-TDF *is (second) preimage resistant (Def. 9) under the assumption that factoring is hard, and extractable (Def. 8).*

Remark 2 (Choice of element t). In Construction 1, public element t can be any quadratic non-residue; small values likely exist and might be favorable for storage efficiency. Observe that, if $p \equiv 3 \bmod 8$ and $q \equiv 7 \bmod 8$, for $t = 2$ we always have $\left(\frac{t}{n}\right) = -1$, so there is not need to store t at all.

$\mathsf{KGen}(1^\lambda)$: Return $(\mathsf{sk}, \mathsf{vk}) = (\mathsf{td}, \mathsf{pub})$ where $(\mathsf{td}, \mathsf{pub}) \leftarrow_R \mathsf{TdGen}(1^{\lambda_2})$

$\mathsf{Sign}(\mathsf{sk}, \mathsf{subj}, \mathsf{msg})$:

1. $s \leftarrow \mathsf{Reverse}(\mathsf{td}, H_{\mathsf{pub}}(\mathsf{subj}), 0)$
2. $(d_1, \ldots, d_{\lambda_h}) \leftarrow H^{\#}(\mathsf{subj}, s, \mathsf{msg})$
3. For $1 \leq i \leq \lambda_h$:
   (a) $b_i \leftarrow H_{\mathsf{pub}}(\mathsf{subj}, s, i)$
   (b) $a_i \leftarrow \mathsf{Reverse}(\mathsf{td}, b_i, d_i)$
4. Return $\sigma \leftarrow (s, a_1, \ldots, a_{\lambda_h})$

$\mathsf{Ver}(\mathsf{vk}, \mathsf{subj}, \mathsf{msg}, \sigma)$:

1. Parse $(s, a_1, \ldots, a_{\lambda_h}) \leftarrow \sigma$
2. If $\mathsf{Decide}(\mathsf{pub}, s) \neq 0$, return 0
3. If $\mathsf{Apply}(\mathsf{pub}, s) \neq H_{\mathsf{pub}}(\mathsf{subj})$, return 0
4. $(d_1, \ldots, d_{\lambda_h}) \leftarrow H^{\#}(\mathsf{subj}, s, \mathsf{msg})$
5. For $1 \leq i \leq \lambda_h$:
   (a) If $\mathsf{Apply}(\mathsf{pub}, a_i) \neq H_{\mathsf{pub}}(\mathsf{subj}, s, i)$, return 0
   (b) If $\mathsf{Decide}(\mathsf{pub}, a_i) \neq d_i$, return 0
6. Return 1

Fig. 5. Double-authentication-preventing signature scheme 2:1-DAPS

5 DAPS construction based on extractable 2:1-TDF

We now come to the central result of this paper, a DAPS scheme generically constructed from any extractable 2:1 trapdoor function, such as the factoring-based Blum-2:1-TDF from the previous section.

Construction 2 (DAPS from extractable 2:1-TDF) *Let λ denote a security parameter, and let λ_2 and λ_h be parameters polynomially dependent on λ. Let $X = (\mathsf{TdGen}, \mathsf{Apply}, \mathsf{Reverse}, \mathsf{Decide})$ be an extractable 2:1 trapdoor function and let $H^{\#} : \{0,1\}^* \to \{0,1\}^{\lambda_h}$ be a hash function. For each* pub *output by* TdGen*, let $H_{\mathsf{pub}} : \{0,1\}^* \to B(\mathsf{pub})$ be a hash function. Double-authentication-preventing signature scheme* 2:1-DAPS *consists of the algorithms specified in Figure 5.*

The basic idea of the signing algorithm is as follows. From any given subject, the signer derives message-independent signing elements $b_1, \ldots, b_{\lambda_h} \in B$.[8] The signer also hashes subject and message to a bit string $d_1 \ldots d_{\lambda_h}$; for each bit d_i, she finds the preimage a_i of the signing element b_i which is in the d_i partition of A; either in A_0 or A_1. The signature σ is basically the vector of these preimages. Intuitively, the scheme is unforgeable because it is hard to find preimages of signing elements b_i without knowing the trapdoor. The scheme is extractable because the signing elements b_i are only dependent on the subject, so the signatures of two different messages for the same subject use the same b_i; if $H^{\#}$ is collision resistance, at least one different d_i is used in the two signatures, so two distinct preimages of b_i are used, allowing recovery of the trapdoor.

5.1 Security of our construction

We next establish existential unforgeability of 2:1-DAPS (cf. Definition 3). The proof proceeds by changing the EUF simulation so that it performs all operations without using the signing key and without (noticeably) changing the distribution of verification key and answers to $\mathcal{A}$'s oracle queries. From any forgery crafted by adversary $\mathcal{A}$, either a preimage or second preimage of X, or a collision of $H^{\#}$ can

[8] For rationale on why the subj-dependent value s is required see the full version [23].

be extracted. Observe that, by Lemma 1, it suffices to require second preimage resistance of X in Theorem 2. The proof appears in the full version [23].

Theorem 2 (2:1-DAPS **is** EUF)**.** *In the setting of Construction 2, if X is second preimage resistant, $H^{\#}$ is collision-resistant, and H_{pub} is a random oracle, then double-authentication-preventing signature* 2:1-DAPS *is existentially unforgeable under adaptive chosen message attacks. More precisely, for any efficient* EUF *algorithm $\mathcal{A}$ making at most q_1 queries to $H_{\mathsf{pub}}(\cdot)$ and q_S queries to $\mathcal{O}_{\mathsf{Sign}}$ oracle, there exist efficient algorithms $\mathcal{B}_1$, $\mathcal{B}_2$, and $\mathcal{C}$ such that*

$$\mathbf{Succ}^{\mathrm{EUF}}_{\text{2:1-DAPS},\mathcal{A}}(\lambda) \leq q_1 \mathbf{Succ}^{\text{INV-1}}_{X,\mathcal{B}_1}(\lambda_2) + 2q_S\lambda_h \mathbf{Succ}^{\text{INV-2}}_{X,\mathcal{B}_2}(\lambda_2) + \mathbf{Succ}^{\mathrm{CR}}_{H^{\#},\mathcal{C}}(\lambda_h),$$

where $\mathbf{Succ}^{\mathrm{CR}}_{H^{\#},\mathcal{C}}(\lambda_h)$ is the success probability of algorithm $\mathcal{C}$ in finding collisions of hash function $H^{\#}$.

Assuming collision resistance of $H^{\#}$, two signatures for different messages but the same subject result in some index i where the hashes $H^{\#}(\mathsf{subj}, s, \mathsf{msg}_1)$ and $H^{\#}(\mathsf{subj}, s, \mathsf{msg}_2)$ differ. The corresponding ith values a_i in the two signatures can be used to extract the signing key. This is the intuition behind Theorem 3; the proof appears in the full version [23].

Theorem 3 (2:1-DAPS **is** DSE*)**.** *In the setting of Construction 2, if X is extractable and $H^{\#}$ is collision-resistant, then double-authentication-preventing signature* 2:1-DAPS *is double-signature extractable with trusted setup.*[9]

5.2 Efficiency of our construction

Table 1 shows the size of verification keys, signing keys, and signatures, and the cost of signature generation and verification for the 2:1-DAPS based on Blum-2:1-TDF, with abstract results as well as for 1024- and 2048-bit keys. We assume the element representation from Remark 1, the verification key optimization from Remark 2, and an implementation of H_{pub} as in Lemma 7.

We also report the results of our implementation of DAPS using the libgcrypt cryptographic library.[10] As libgcrypt does not have routines for square roots or Jacobi symbols, we implemented our own, and we expect that there may be space for improvement with optimized implementations of these operations. Timings reported are an average of 50 iterations, performed on a 2.6 GHz Intel Core i7 (3720QM) CPU, using libgcrypt 1.5.2, compiled in `x86_64` mode using LLVM 3.3 and compiler flag `-O3`. Source code for our implementation is available online at `http://eprints.qut.edu.au/73005/`.

With 1024-bit signing and verification keys, a signature is about 20 KiB in size, and takes about 0.341 s to generate and 0.105 s to verify. While less efficient than a regular signature scheme, we believe these timings are still tolerable; this holds in particular if our scheme is used to implement CA functionality where signature generation happens rarely and verification results can be cached.

[9] See the full version [23] for how to achieve double-signature extractability without trusted setup using zero-knowledge proofs.

[10] `http://www.gnu.org/software/libgcrypt/`

Table 1. Efficiency of 2:1-DAPS based on sign-agnostic quadratic residues.

	General analysis	**libgcrypt implementation**	
λ_h	—	160	160
λ_2 (size of n in bits)	—	1024	2048
Key generation time	—	0.097 s	0.759 s
Signing key size (bits)	$\log_2 n$	1024	2048
Verification key size (bits)	$\log_2 n$	1024	2048
Signature generation cost	$(\lambda_h + 1) \cdot \text{Jac}$, $(\lambda_h + 1) \cdot \text{sqrt}$	0.341 s	1.457 s
Signature size (bits)	$(\lambda_h + 1) \log_2 n$	164 864 = 20 KiB	329 728 = 40 KiB
Signature verification cost	$(2\lambda_h + 1) \cdot \text{Jac}$, $(\lambda_h + 1) \cdot \text{sqr}$	0.105 s	0.276 s

Legend: Jac: computation of Jacobi symbol modulo n; sqrt: square root modulo n; sqr: squaring modulo n.

6 Applications

DAPS allows applications that employ digital signatures for establishing unique bindings between digital objects to provide self-enforcement for correct signer behaviour, and resistance by signers to coercion. Whenever the verifier places high value on the uniqueness of the binding, it may be worthwhile to employ DAPS instead of traditional digital signatures, despite potential increased damage when signers make mistakes.

It should be noted that use of DAPS may impose an additional burden on honest signers: they need to maintain a list of previously signed subjects to avoid double signing. Some signers may already do so, but the importance of the correctness of this list is increased with DAPS. As noted below, signers may wish to use additional protections to maintain their list of signed subjects, for example by cryptographically authenticating it using a message authentication code with a key in the same hardware security module as the main signing key.

In this section, we examine a few cryptographic applications involving unique bindings and discuss the potential applicability of DAPS.

Certificate authorities. DAPS could be used to ensure that certification authorities in the web PKI behave as expected. For example, by having the subject consist of the domain name and the year, and the message consist of the public key and other certificate details, a CA who signs one certificate for "`www.example.com`" using DAPS cannot sign another for the same domain and time period without invalidating its own key. A CA using DAPS must then be stateful, carefully tracking the previous subjects signed and refusing to sign duplicates. In commercial CAs, where signing is done on a hardware security module (HSM), the list of subjects signed should be kept under authenticated control of the HSM.

A DAPS-based PKI would need to adopt an appropriate convention on validity periods to accommodate expiry of certificates without permitting double-signing. For example, a DAPS PKI may use a subject with a low-granularity non-overlapping validity period ("`www.example.com`$\|$`2014`") since high-granularity

overlapping validity periods in the subject give a malicious CA a vector for issuing two certificates without signing the exact same subject twice ("`www.example.com`$\|$`20140501-20150430`" versus "`www.example.com`$\|$`20140502-20150501`").

Furthermore, a DAPS-based PKI could support revocation using standard mechanisms such as certificate revocation lists. Reissuing could be achieved by including a counter in the DAPS subject (e.g., "`www.example.com`$\|$`2014`$\|$`0`") and using DAPS-based revocation to provide an unambiguous and unalterable auditable chain from the initial certificate to the current one.

One of the major problems with multi-CA PKIs such as the web PKI is that clients trust many CAs, any one of which can issue a certificate for a particular subject. A DAPS-based PKI would prevent one CA from signing multiple certificates for a subject, but not other CAs from also signing certificates for that subject. It remains a very interesting open question to find cryptographic constructions that solve the multi-CA PKI problem.

Time-stamping. A standard approach to preventing time-stamping authorities from "changing the past" is to require that, when asserting that certain pieces of information x exist at a particular time t, the actual message being signed must also include the (hash of) messages authenticated in the previous time periods. The authority is prevented from trying to change the past and assert that $x' \neq x$ existed at time t because the signatures issued at time periods $t+1, t+2, \ldots$ chain back to the original message x.

DAPS could be used to alternatively discourage time-stamping authority fraud by having the subject consist of the time period t and the message consist of whatever information x is to be signed at that time period. A time-stamping authority who signs an assertion for a given time period using DAPS cannot sign another for the same time period without invalidating its own key. Assuming an honest authority's system is designed to only sign once per time period, the signer need not track all signed subjects, since time periods automatically increment.

Hybrid DAPS + standard signatures. DAPS could be combined with a standard signature scheme to provide more robustness in the case of an accidental error, but also provide a clear and quantifiable decrease in security due to a double signing, giving users a window of time in which to migrate away from the signer.

We can achieve this goal by augmenting a generic standard signature scheme with our factoring-based DAPS as follows. The signer publishes a public key consisting of the standard signature's verification key, the 2:1-DAPS verification key n, and a verifiable Rabin encryption under key n of, say, the first half of the bits of the standard scheme's signing key. The hybrid DAPS signature for a subject/message pair would consist of the standard scheme's signature on subject and message concatenated, and the DAPS signature on separated subject and message. If two messages are ever signed for the same subject, then the signer's DAPS secret key can be recovered, which can then be used to decrypt the Rabin ciphertext containing the first half of the standard scheme's signing key. This is not quite enough to readily forge signatures, but it substantially and quantifiably weakens trust in this signer's signatures, making it clear that migration to a new

signer must occur but still providing a window of time in which to migrate. As the sketched combination of primitives exhibits non-standard dependencies between different secret keys, a thorough cryptographic analysis would be required.

7 Conclusions

We have introduced a new type of signatures, *double-authentication-preventing signatures*, in which a subject/message pair is signed. In certain situations, DAPS can provide greater assurance to verifiers that signers behave honestly since there is a great disincentive for signers who misbehave: if a signer ever signs two different messages for the same subject, then enough information is revealed to allow anyone to fully recover the signer's secret key. Although this leads to less robustness in the face of accidental errors, it also provides a mechanism for *self-enforcement* of correct behaviour and gives trusted signers such as CAs an argument to resist coercion and the compelled certificate creation attack.

Our construction is based on a new primitive called *extractable 2:1 trapdoor functions*. We have shown how to instantiate this using an algebraic reformulation of sign-agnostic quadratic residues modulo Blum integers; the resulting DAPS is unforgeable assuming factoring is hard, with reasonable signature sizes and computation times.

We believe DAPS can be useful in scenarios where trusted authorities are meant to make *unique* bindings between identifiers and digital objects, such as certificate authorities in PKIs who are supposed to make unique bindings between domain names and public keys, and time-stamping authorities who are supposed to make unique bindings between time periods and pieces of data.

Besides the practical applications of DAPS, several interesting theoretical questions arise from our work. Are there more efficient constructions of DAPS? How else can extractable 2:1 trapdoor functions be instantiated? Given that DAPS and double-spending-resistant digital cash use similar number-theoretic primitives, can DAPS be used to generically construct untraceable digital cash? Can these techniques be applied to key generation in the identity-based setting? Can DAPS be adapted to provide assurance in a multi-CA setting?

References

1. Soghoian, C., Stamm, S.: Certified lies: Detecting and defeating government interception attacks against SSL (short paper). In Danezis, G., ed.: FC 2011. Volume 7035 of LNCS., Springer (2011) 250–259
2. Fox-It: Black tulip: Report of the investigation into the DigiNotar certificate authority breach (2012)
3. Google Online Security Blog: An update on attempted man-in-the-middle attacks (2011)
4. Dwork, C., Lotspiech, J.B., Naor, M.: Digital signets: Self-enforcing protection of digital information (preliminary version). In: 28th ACM STOC, ACM Press (1996) 489–498

5. Camenisch, J., Lysyanskaya, A.: An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In Pfitzmann, B., ed.: EUROCRYPT 2001. Volume 2045 of LNCS., Springer (2001) 93–118
6. Jakobsson, M., Juels, A., Nguyen, P.Q.: Proprietary certificates. In Preneel, B., ed.: CT-RSA 2002. Volume 2271 of LNCS., Springer (2002) 164–181
7. Kiayias, A., Tang, Q.: How to keep a secret: leakage deterring public-key cryptosystems. In Sadeghi, A.R., Gligor, V.D., Yung, M., eds.: ACM CCS 13, ACM Press (2013) 943–954
8. Chor, B., Fiat, A., Naor, M.: Tracing traitors. In Desmedt, Y., ed.: CRYPTO'94. Volume 839 of LNCS., Springer (1994) 257–270
9. Goyal, V.: Reducing trust in the PKG in identity based cryptosystems. In Menezes, A., ed.: CRYPTO 2007. Volume 4622 of LNCS., Springer (2007) 430–447
10. Chaum, D., Fiat, A., Naor, M.: Untraceable electronic cash. In Goldwasser, S., ed.: CRYPTO'88. Volume 403 of LNCS., Springer (1988) 319–327
11. Lamport, L.: Constructing digital signatures from a one way function. Technical Report CSL-98, SRI International (1979)
12. Merkle, R.C.: A certified digital signature (subtitle: That antique paper from 1979). In Brassard, G., ed.: Advances in Cryptology – Proc. CRYPTO '89. Volume 435 of LNCS., Springer (1990) 218–238
13. Waidner, M., Pfitzmann, B.: The dining cryptographers in the disco - unconditional sender and recipient untraceability with computationally secure serviceability (abstract) (rump session). In Quisquater, J.J., Vandewalle, J., eds.: EUROCRYPT'89. Volume 434 of LNCS., Springer (1989) 690
14. van Heyst, E., Pedersen, T.P.: How to make efficient fail-stop signatures. In Rueppel, R.A., ed.: EUROCRYPT'92. Volume 658 of LNCS., Springer (1992) 366–377
15. van Heijst, E., Pedersen, T.P., Pfitzmann, B.: New constructions of fail-stop signatures and lower bounds (extended abstract). In Brickell, E.F., ed.: CRYPTO'92. Volume 740 of LNCS., Springer (1992) 15–30
16. Bari, N., Pfitzmann, B.: Collision-free accumulators and fail-stop signature schemes without trees. In Fumy, W., ed.: EUROCRYPT'97. Volume 1233 of LNCS., Springer (1997) 480–494
17. Pedersen, T.P., Pfitzmann, B.: Fail-stop signatures. SIAM Journal on Computing **26** (1997) 291–330
18. Mashatan, A., Ouafi, K.: Forgery-resilience for digital signature schemes. In: Proc. 7th ACM Symposium on Information, Computer and Communications Security (ASIACCS) 2012, ACM (2012) 24–25
19. Krawczyk, H., Rabin, T.: Chameleon signatures. In: NDSS 2000, The Internet Society (2000)
20. Shamir, A., Tauman, Y.: Improved online/offline signature schemes. In Kilian, J., ed.: CRYPTO 2001. Volume 2139 of LNCS., Springer (2001) 355–367
21. Ateniese, G., de Medeiros, B.: Identity-based chameleon hash and applications. In Juels, A., ed.: FC 2004. Volume 3110 of LNCS., Springer (2004) 164–180
22. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM J. Comput. **17** (1988) 281–308
23. Poettering, B., Stebila, D.: Double-authentication-preventing signatures (full version). Cryptology ePrint Archive, Report 2013/333 (2014)
24. Hofheinz, D., Kiltz, E.: The group of signed quadratic residues and applications. In Halevi, S., ed.: CRYPTO 2009. Volume 5677 of LNCS., Springer (2009) 637–653