Int. J. Inf. Secur. (2017) 16:1–22
DOI 10.1007/s10207-015-0307-8

REGULAR CONTRIBUTION

Double-authentication-preventing signatures

Bertram Poettering[1] · **Douglas Stebila**[2]

Published online: 12 December 2015

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; we show an additional application of these new extractable trapdoor functions to standard digital signatures.

✉ Bertram Poettering
bertram.poettering@rub.de

Douglas Stebila
stebila@qut.edu.au

[1] Ruhr University Bochum, Bochum, Germany

[2] Queensland University of Technology, Brisbane, QLD, Australia

Keywords Digital signatures · Double signatures · Dishonest signer · Coercion · Compelled certificate creation attack · Self-enforcement · Two-to-one trapdoor functions

Mathematics Subject Classification 94A60 Cryptography

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 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 [40] 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)".[1] Moreover, various documents released by NSA contractor Edward Snowden in disclosures in June–September 2013 indicate government interest in executing man-in-the-middle attacks on SSL users.[2]

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[3]), 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 [20], which was found to have issued fraudulent certificates that were used against Iranian Internet users [23], 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 the CA to invest in better systems. For a CA trying to resist a lawful order compelling it to issue a fraudulent certificate, however, such consequences may not be enough to convince a judge that the CA 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. On the one hand, this "fragility" provides no room for mistakes but, on the other hand, encourages "self-enforcement" of correct behaviour and allows a signer to make a more compelling argument resisting lawful coercion. If a CA fulfils 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 to be signed are 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. Depending on the nature of the subjects, in some applications an honest signer may need to track the list of subjects signed to avoid signing the same subject twice.

In addition to unforgeability, we require one of two new security properties for DAPS: *double-signature forgeability*, where a signer who signs two messages for the same subject reveals enough information for anyone to sign arbitrary messages, and a stronger notion called *double-signature extractability*, where two signatures on the same subject allow full recovery of the signing key.

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 double-authentication-preventing signature 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. Note that in applications such as PKI, signing happens rarely, and verifications may be cached.

Our quadratic residue-based construction provides double-signature extractability in what we call the *trusted set-up model*, where it is assumed that the signer follows the correct procedure for key generation. This model is suitable for scenarios where signers want to be honest and create their keys with best intention—and we hope most CAs belong to this group, facing coercive requests only after they have completed set-up. Our construction can be translated to the *untrusted set-up model*, where parties do not have to trust the signer to generate keys following the scheme specification, using zero-knowledge techniques for proving well-formedness of the verification key.

[1] http://www.wired.com/threatlevel/2010/03/packet-forensics/

[2] https://www.schneier.com/blog/archives/2013/09/new_nsa_leak_sh.html

[3] http://comodo.com/Comodo-Fraud-Incident-2011-03-23.html

We also show how to use extractable two-to-one trapdoor functions to construct tightly secure standard digital signatures, demonstrating the utility of extractable two-to-one trapdoor functions beyond our immediate application of DAPS.

1.2 Outline

We recall some notation and standard definitions in Sect. 2. We define a double-authentication-preventing signature in Sect. 3 and its unforgeability as well as double-signature forgeability and double-signature extractability properties. We introduce in Sect. 4 extractable 2:1 trapdoor functions, and as a warm-up we show in Sect. 5 how to construct a tightly secure standard digital signature scheme. We provide a factoring-based instantiation of extractable 2:1 trapdoor functions in Sect. 6 using sign-agnostic quadratic residues. In Sect. 7, we generically construct a DAPS scheme from extractable 2:1 trapdoor functions and prove the scheme's security and double-signature extractability in the trusted set-up model, as well as discuss its use with untrusted set-up. Sect. 8 examines applications of DAPS to certification and time-stamping authorities. We conclude in Sect. 9. The appendices contain a review of basic results from number theory ("Appendix 1"), a construction of a random oracle that maps into the group of sign-agnostic quadratic residues ("Appendix 2"), and proofs of results from the main body ("Appendix 3").

1.3 Related work

Certificate auditing and other techniques. Mechanisms such as Certificate Transparency[4] 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. Recently, several distinct technical measures [19,26,35] have been proposed to try to wrest some trust decisions away from CAs, for example by allowing websites to make assertions to users about what certificates to accept in the future.

Self-enforcement and traitor tracing. Dwork et al. [18] introduced the notion of *self-enforcement* in cryptography, in which the cryptosystem is designed to force 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: trade-offs in efficiency [18] or by allowing recovering of some associated secret value with any delegated version of the secret information [13,28,31]. Broadcast encryption schemes often aim for a related notion, traitor tracing [12], 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 [24] aimed to reduce trust in the key generation centre (KGC) in identity-based encryption: How can a user demonstrate that the KGC created a second key for the user's identity? 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 offered by DAPS: double signing leads to an immediate and irrevocable loss of confidence, rather than just proof of misbehaving for consideration of prosecution.

Digital cash. Digital cash schemes [11] 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 [11]: 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 [32], allow at most one message to be signed. Many instances can be combined using Merkle trees [33] to allow multiple signatures with just a single verification key, but key generation time becomes a function of the total number of signatures allowed.

Double-authentication-preventing signatures are fundamentally different from one-time signatures: in DAPS, the number of messages to be signed need not be fixed a pri-

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

ori, and our construction relies on number-theoretic trapdoor functions, rather than solely hash functions. A natural first attempt at creating a DAPS scheme is to begin with a Merkle-tree construction, in which each subject identifies a path from the root to a leaf and hence which keys must be used to sign the message. However, this requires a key generation time at least linear in the size of the subject space and therefore limits the size of the latter. Moreover, in such a scheme two signatures under the same subject do not immediately lead to the ability to forge signatures on arbitrary messages. Our scheme allows for arbitrary subject spaces and has efficient key generation time, so we leave the construction of a tree-based DAPS as an open problem.

Fail-stop signatures. Fail-stop signatures considered in [7,38,43–45] 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 [34] 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 fail-stop and forgery-resilient signatures focus on the ability of an *honest* signer to prove 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 [30] 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 [1,10,41]. However, it remains open how DAPS could be constructed from Chameleon hash functions.

2 Preliminaries

In this section, we introduce some notation and recall some standard cryptographic definitions.

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. For a given x, we write $\mathcal{A}^{\mathcal{O}}(y) \Rightarrow x$ for the event that $\mathcal{A}$ outputs x. We use the notation $\mathcal{A}(y; r)$ to explicitly identify the random coins r on which the otherwise deterministic algorithm $\mathcal{A}$ is run.

Definition 1 (*Pseudorandom function*) A *pseudorandom function* (PRF) with output length c is a family $F = (F_\lambda)_{\lambda\in\mathbb{N}}$ of efficient functions $F_\lambda\colon \{0,1\}^\lambda \times \{0,1\}^* \to \{0,1\}^c$. It is *secure* if the advantage $\mathbf{Adv}^{\mathrm{prf}}_{F,\mathcal{D}}(\lambda)$ of any efficient distinguisher $\mathcal{D}$ is a negligible function in λ, where we define

$$\mathbf{Adv}^{\mathrm{prf}}_{F,\mathcal{D}}(\lambda) := \Big|\Pr\Big[\varphi \leftarrow_R \mathrm{Func}^{\{0,1\}^*\to\{0,1\}^c} : \mathcal{D}^{\varphi} \Rightarrow 1\Big] - \Pr\Big[K \leftarrow_R \{0,1\}^\lambda : \mathcal{D}^{F_\lambda(K,\cdot)} \Rightarrow 1\Big]\Big|$$

and denote with $\mathrm{Func}^{\{0,1\}^*\to\{0,1\}^c}$ the set of all functions mapping domain $\{0,1\}^*$ to range $\{0,1\}^c$.

Definition 2 (*Signature scheme*) A *signature scheme* is a tuple of efficient algorithms $\Sigma = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Ver})$ as follows:

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

Definition 3 (*Correctness*) Signature scheme Σ 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 $\mathsf{msg} \in \{0,1\}^*$, and for all signatures $\sigma \leftarrow_R \mathsf{Sign}(\mathsf{sk}, \mathsf{msg})$, we have that $\mathsf{Ver}(\mathsf{vk}, \mathsf{msg}, \sigma) = 1$.

Definition 4 (*Existential unforgeability*) A signature scheme Σ is *existentially unforgeable under adaptive chosen message attacks* if the success probability $\mathbf{Succ}^{\mathrm{EUF}}_{\Sigma,\mathcal{A}}(\lambda) := \Pr[\mathbf{Exp}^{\mathrm{EUF}}_{\Sigma,\mathcal{A}}(\lambda) = 1]$ in the EUF experiment of Fig. 1 is a negligible function in λ, for all efficient adversaries $\mathcal{A}$.

$\mathbf{Exp}^{\mathrm{EUF}}_{\Sigma,\mathcal{A}}(\lambda)$:
1 SignedList $\leftarrow \emptyset$
2 $(\mathsf{sk}, \mathsf{vk}) \leftarrow_R \mathsf{KGen}(1^\lambda)$
3 $(\mathsf{msg}^*, \sigma^*) \leftarrow_R \mathcal{A}^{\mathcal{O}_{\mathsf{Sign}}}(\mathsf{vk})$
4 If $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{Sign}}(\mathsf{msg})$:
5 Append msg to SignedList
6 $\sigma \leftarrow_R \mathsf{Sign}(\mathsf{sk}, \mathsf{msg})$
7 Return σ to $\mathcal{A}$
8 Return 1 iff all the following hold:
9 - $\mathsf{Ver}(\mathsf{vk}, \mathsf{msg}^*, \sigma^*) = 1$
10 - $\mathsf{msg}^* \notin$ SignedList

Fig. 1 Experiment for existential unforgeability of signature schemes

3 Double-authentication-preventing signatures

In this section, we present the central definitions of the paper: a *double-authentication-preventing signature* and, as security requirements, the standard (though slightly adapted) notion of *existential unforgeability*, as well as the new properties of *forgeability* and *signing key extractability* given two signatures on the same subject.

Definition 5 (*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 $\mathsf{subj}, \mathsf{msg} \in \{0,1\}^*$, this algorithm outputs a signature σ.
- $\mathsf{Ver}(\mathsf{vk}, \mathsf{subj}, \mathsf{msg}, \sigma)$: On input verification key vk, subject/message pair $\mathsf{subj}, \mathsf{msg} \in \{0,1\}^*$, and candidate signature σ, this algorithm outputs either 0 or 1.

Definition 6 (*Correctness*) A double-authentication-preventing signature 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 $\mathsf{Ver}(\mathsf{vk}, \mathsf{subj}, \mathsf{msg}, \sigma) = 1$.

3.1 Unforgeability

Our unforgeability notion largely coincides with the standard unforgeability notion for digital signature schemes [22]; the main difference is that, for DAPS, forgeries crafted by the adversary are not considered valid if the adversary has requested forgeries on different messages for the same subject.

Definition 7 (*Existential unforgeability*) A double-authentication-preventing signature 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 Fig. 2 is a negligible function in λ.

3.2 Double-signature forgeability

Although Definition 7 ensures that signatures of DAPS are generally unforgeable, we do want signatures to be forgeable in certain circumstances, namely when two different messages have been signed for the same subject. First, we define the notion of *compromising pairs of signatures*, which says when two signatures should lead to a forgery, and then define *double-signature forgeability*.

$\mathbf{Exp}^{\mathrm{EUF}}_{\mathsf{DAPS},\mathcal{A}}(\lambda)$:
1 SignedList $\leftarrow \emptyset$
2 $(\mathsf{sk}, \mathsf{vk}) \leftarrow_R \mathsf{KGen}(1^\lambda)$
3 $(\mathsf{subj}^*, \mathsf{msg}^*, \sigma^*) \leftarrow_R \mathcal{A}^{\mathcal{O}_{\mathsf{Sign}}}(\mathsf{vk})$
4 If $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{Sign}}(\mathsf{subj}, \mathsf{msg})$:
5 Append $(\mathsf{subj}, \mathsf{msg})$ to SignedList
6 $\sigma \leftarrow_R \mathsf{Sign}(\mathsf{sk}, \mathsf{subj}, \mathsf{msg})$
7 Return σ to $\mathcal{A}$
8 Return 1 iff all the following hold:
9 - $\mathsf{Ver}(\mathsf{vk}, \mathsf{subj}^*, \mathsf{msg}^*, \sigma^*) = 1$
10 - $(\mathsf{subj}^*, \mathsf{msg}^*) \notin$ SignedList
11 - $\forall\, \mathsf{subj}, \mathsf{msg}_0, \mathsf{msg}_1$:
12 if $(\mathsf{subj}, \mathsf{msg}_0), (\mathsf{subj}, \mathsf{msg}_1) \in$ SignedList
13 then $\mathsf{msg}_0 = \mathsf{msg}_1$

Fig. 2 Experiment for existential unforgeability of DAPS

Definition 8 (*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 forgeability 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 double-signature forgeries. We consider two scenarios: the *trusted set-up model*, where key generation is assumed to proceed honestly, and the *untrusted set-up model*, where the adversary has full control over key generation as well.

Definition 9 (*Double-signature forgeability*) A double-authentication-preventing signature DAPS is *double-signature forgeable* (resp. *double-signature forgeable with trusted set-up*) if an efficient algorithm

- $\mathsf{Forge}(\mathsf{vk}, (S_1, S_2), \mathsf{subj}^*, \mathsf{msg}^*)$: On input verification key vk, compromising pair (S_1, S_2), and subject/message pair $\mathsf{subj}^*, \mathsf{msg}^* \in \{0,1\}^*$, this algorithm outputs a signature σ^*.

is known such that, for all efficient adversaries $\mathcal{A}$, the probability $\mathbf{Succ}^{\mathrm{DSF}^{(*)}}_{\mathsf{DAPS},\mathcal{A}}(\lambda) := \Pr[\mathbf{Exp}^{\mathrm{DSF}^{(*)}}_{\mathsf{DAPS},\mathcal{A}}(\lambda) = 1]$ of success in the DSF (resp. DSF*) experiment of Fig. 3 is a negligible function in λ.

3.3 Double-signature extractability

While the notion of double-signature forgeability expresses the desired functionality of the scheme from a theoretical

$\mathbf{Exp}^{\mathrm{DSF}}_{\mathsf{DAPS},\mathcal{A}}(\lambda)$:
1 $(\mathsf{vk}, (S_1, S_2), \mathsf{subj}^*, \mathsf{msg}^*) \leftarrow_R \mathcal{A}(1^\lambda)$
2 $\sigma^* \leftarrow_R \mathsf{Forge}(\mathsf{vk}, (S_1, S_2), \mathsf{subj}^*, \mathsf{msg}^*)$
3 Return 1 iff all the following hold:
4 - (S_1, S_2) is compromising
5 - $\mathsf{Ver}(\mathsf{vk}, \mathsf{subj}^*, \mathsf{msg}^*, \sigma^*) \neq 1$

$\mathbf{Exp}^{\mathrm{DSF}^*}_{\mathsf{DAPS},\mathcal{A}}(\lambda)$:
1 $(\mathsf{sk}, \mathsf{vk}) \leftarrow_R \mathsf{KGen}(1^\lambda)$
2 $((S_1, S_2), \mathsf{subj}^*, \mathsf{msg}^*) \leftarrow_R \mathcal{A}(\mathsf{sk}, \mathsf{vk})$
3 $\sigma^* \leftarrow_R \mathsf{Forge}(\mathsf{vk}, (S_1, S_2), \mathsf{subj}^*, \mathsf{msg}^*)$
4 Return 1 iff all the following hold:
5 - (S_1, S_2) is compromising
6 - $\mathsf{Ver}(\mathsf{vk}, \mathsf{subj}^*, \mathsf{msg}^*, \sigma^*) \neq 1$

Fig. 3 Experiments for double-signature forgeability (without and with trusted set-up)

point of view, from an engineering perspective it may be more natural to consider *double-signature extractability*, in which two signatures for the same subject lead to full recovery of the signing key; obviously, full recovery of the signing key gives the ability to forge.

Definition 10 (*Double-signature extractability*) A double-authentication-preventing signature DAPS is *double-signature extractable* (resp. *double-signature extractable with trusted set-up*) 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 Fig. 4 is a negligible function in λ.

$\mathbf{Exp}^{\mathrm{DSE}}_{\mathsf{DAPS},\mathcal{A}}(\lambda)$:
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:
4 - (S_1, S_2) is compromising
5 - 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:
5 - (S_1, S_2) is compromising
6 - $\mathsf{sk}' \neq \mathsf{sk}$

Fig. 4 Experiments for double-signature extractability (without and with trusted set-up)

Note that 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 Sect. 7, when instantiated with the factoring-based primitive of Sect. 6, 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 $p < q$).

Clearly, double-signature extractability implies double-signature forgeability. In fact, DSE implies that the forger can generate signatures that are perfectly indistinguishable from signatures generated by the honest signer. This is an important feature that plain double-signature forgeable schemes do not necessarily offer, and indeed one can construct degenerate examples of schemes that are double-signature forgeable but for which forged signatures are obviously different from honest signatures.

4 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 has 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 $f^{-1}(b, 1)$. Observe that this effectively 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 $a_0 \overset{x}{\sim} a_1$ for such configurations). The setting of 2:1-TDFs is illustrated in Fig. 5. We will formalize the functionality and security properties below.

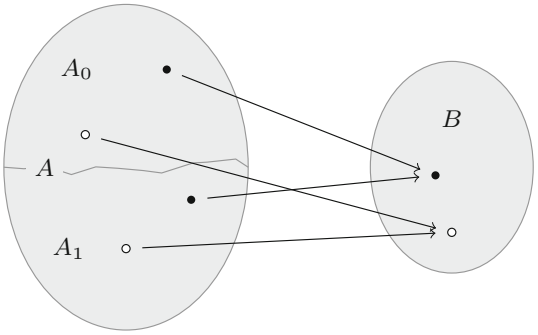

Fig. 5 Illustration of a 2:1 trapdoor function $f : A \to B$. Each element of B has exactly two preimages, one in A_0 and one in A_1

4.1 Definition

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

Definition 11 (*2:1 trapdoor function*) A *2:1 trapdoor function (2:1-TDF)* is a tuple of efficient algorithms (TdGen, Sample_A, Sample_B, Apply, $\mathsf{Reverse}$, 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 $B = B(\mathsf{pub})$.
- $\mathsf{Sample}_A(\mathsf{pub}, d; r)$: On input public information pub, bit $d \in \{0, 1\}$, and randomness $r \in \{0, 1\}^\lambda$, this algorithm outputs a value $a \in A(\mathsf{pub})$.
  As shortcuts:
  - $\mathsf{Sample}_A(\mathsf{pub}, d) := r \leftarrow_R \{0, 1\}^\lambda$; return $\mathsf{Sample}_A(\mathsf{pub}, d; r)$
  - $\mathsf{Sample}_A(\mathsf{pub}) := d \leftarrow_R \{0, 1\}$; return $\mathsf{Sample}_A(\mathsf{pub}, d)$
  - $\mathsf{Sample}_{A_0}(\mathsf{pub}) :=$ return $\mathsf{Sample}_A(\mathsf{pub}, 0)$
  - $\mathsf{Sample}_{A_1}(\mathsf{pub}) :=$ return $\mathsf{Sample}_A(\mathsf{pub}, 1)$
  - $\mathsf{Sample}_B(\mathsf{pub}; r)$: On input public information pub and randomness $r \in \{0, 1\}^\lambda$, this algorithm outputs a value $b \in B(\mathsf{pub})$.
  - $\mathsf{Apply}(\mathsf{pub}, a)$: 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 12 (*Correctness*) 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) $\mathsf{Decide}(\mathsf{pub}, \mathsf{Reverse}(\mathsf{td}, b, d)) = d$.

We further require $\mathsf{Decide}(\mathsf{pub}, \mathsf{Sample}_A(\mathsf{pub}, d; r)) = d$ for all $d \in \{0, 1\}$ and $r \in \{0, 1\}^\lambda$.

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 Sect. 4.

4.2 Security notions

We proceed with the specification of the principal security properties of 2:1-TDFs, samplability and one-wayness. The treatment of extraction follows in the next section. The proofs of Lemmas 1 and 2 appear in "Proofs from Section 4" of Appendix 3.

4.2.1 Samplability

The task of a 2:1-TDF's Sample_A and Sample_B algorithms is to provide samples from sets $A(\mathsf{pub})$ and $B(\mathsf{pub})$, respectively, that are distributed nearly uniformly. The *samplability* security property refers to the extent to which these samples are close to uniform.

Definition 13 (*Sampling distance*) Let X be a 2:1-TDF and S_0, S_1 be (sampling) algorithms. We define the *sampling distance* of S_0, S_1 with respect to a distinguisher $\mathcal{D}$ as $\mathbf{Dist}_{X,\mathcal{D}}^{S_0,S_1}(\lambda) := |P_0(\lambda) - P_1(\lambda)|$, where $P_d(\lambda) = \Pr[(\mathsf{td}, \mathsf{pub}) \leftarrow_R \mathsf{TdGen}(1^\lambda); x \leftarrow_R S_d(\mathsf{pub}) : \mathcal{D}(\mathsf{pub}, x) = 1]$.

We consider different strategies to obtain samples from set B: using the Sample_B algorithm directly, or using Sample_A (or Sample_{A_0}, or Sample_{A_1}) and mapping obtained samples from set A to set B using the Apply algorithm. The latter hybrid constructions are formalized in Definition 14. We show in Lemma 1 that they yield reasonable results, assuming good Sample_A and Sample_B algorithms.

Definition 14 (*Hybrid sampling*) For a 2:1-TDF, let $(\mathsf{td}, \mathsf{pub})$ be output by TdGen. Then sampling algorithm Sample_B^A for set $B(\mathsf{pub})$ is defined as $\mathsf{Sample}_B^A(\mathsf{pub}) := \mathsf{Apply}(\mathsf{pub}, \mathsf{Sample}_A(\mathsf{pub}))$. We define sampling algorithms $\mathsf{Sample}_B^{A_0}$ and $\mathsf{Sample}_B^{A_1}$ correspondingly.

Lemma 1 (Quality of hybrid sampling) *Let X be a 2:1-TDF and let $\mathcal{D}_B$ be an efficient distinguisher. Then there exist efficient distinguishers $\mathcal{D}'_A, \mathcal{D}'_{A_0}, \mathcal{D}'_{A_1}, \mathcal{D}'_B, \mathcal{D}''_B, \mathcal{D}'''_B$ such that*

$$\mathbf{Dist}^{\mathsf{Sample}_B, \mathsf{Sample}_B^A}_{X, \mathcal{D}_B}(\lambda) \le \mathbf{Dist}^{\mathsf{Sample}_A, U(A)}_{X, \mathcal{D}'_A}(\lambda) + \mathbf{Dist}^{\mathsf{Sample}_B, U(B)}_{X, \mathcal{D}'_B}(\lambda)$$

$$\mathbf{Dist}^{\mathsf{Sample}_B, \mathsf{Sample}_B^{A_0}}_{X, \mathcal{D}_B}(\lambda) \le \mathbf{Dist}^{\mathsf{Sample}_{A_0}, U(A_0)}_{X, \mathcal{D}'_{A_0}}(\lambda) + \mathbf{Dist}^{\mathsf{Sample}_B, U(B)}_{X, \mathcal{D}''_B}(\lambda)$$

$$\mathbf{Dist}^{\mathsf{Sample}_B, \mathsf{Sample}_B^{A_1}}_{X, \mathcal{D}_B}(\lambda) \le \mathbf{Dist}^{\mathsf{Sample}_{A_1}, U(A_1)}_{X, \mathcal{D}'_{A_1}}(\lambda) + \mathbf{Dist}^{\mathsf{Sample}_B, U(B)}_{X, \mathcal{D}'''_B}(\lambda).$$

In Lemma 1, the terms on the left-hand side are small if the terms on the right-hand side are. This observation motivates the following security requirement on 2:1-TDFs. Note that if $\mathbf{Dist}^{\mathsf{Sample}_A, U(A)}_{X, \mathcal{D}}$ is negligible, then so are $\mathbf{Dist}^{\mathsf{Sample}_{A_0}, U(A_0)}_{X, \mathcal{D}}$ and $\mathbf{Dist}^{\mathsf{Sample}_{A_1}, U(A_1)}_{X, \mathcal{D}}$.

Definition 15 (*Samplability of 2:1-TDF*) Let X denote a 2:1-TDF. We say that X is *samplable* if, for all efficient distinguishers $\mathcal{D}, \mathcal{D}'$, we have that $\mathbf{Dist}^{\mathsf{Sample}_A, U(A)}_{X, \mathcal{D}}$ and $\mathbf{Dist}^{\mathsf{Sample}_B, U(B)}_{X, \mathcal{D}'}$ are negligible functions.

4.2.2 One-wayness

We next define one-wayness for 2:1-TDFs. Intuitively, it should be infeasible to find preimages and second preimages of the Apply algorithm without knowing the corresponding trapdoor.

Definition 16 (*Preimage resistance of 2:1-TDF*) A 2:1-TDF X is *preimage resistant* and *second preimage resistant* if $\mathbf{Succ}^{\mathrm{INV}-1}_{X, \mathcal{A}}(\lambda) := \Pr[\mathbf{Exp}^{\mathrm{INV}-1}_{X, \mathcal{A}}(\lambda) = 1]$ and $\mathbf{Succ}^{\mathrm{INV}-2}_{X, \mathcal{B}}(\lambda) := \Pr[\mathbf{Exp}^{\mathrm{INV}-2}_{X, \mathcal{B}}(\lambda) = 1]$ are negligible functions in λ, for all efficient adversaries $\mathcal{A}$ and $\mathcal{B}$, where $\mathbf{Exp}^{\mathrm{INV}-1}_{X, \mathcal{A}}$ and $\mathbf{Exp}^{\mathrm{INV}-2}_{X, \mathcal{B}}$ are as in Fig. 6.

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

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

Fig. 6 Experiments for preimage and second preimage resistance of 2:1-TDFs

The following simple lemma shows that second preimage resistance implies preimage resistance. We will see in Sect. 4.3 that these notions are actually equivalent for an extractable variant of 2:1-TDF.

Lemma 2 (INV−2 ⇒ INV−1 if 2:1-TDF samplable) *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 and an efficient distinguisher $\mathcal{D}_B$ such that* $\mathbf{Succ}^{\mathrm{INV}-1}_{X, \mathcal{A}}(\lambda) \le 2 \cdot \mathbf{Succ}^{\mathrm{INV}-2}_{X, \mathcal{B}}(\lambda) + \mathbf{Dist}^{\mathsf{Sample}_B, \mathsf{Sample}_B^A}_{X, \mathcal{D}_B}(\lambda)$.

4.3 Extractable 2:1 trapdoor functions

We 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 17 (*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 we have $\mathsf{Extract}(\mathsf{pub}, a, a') = \mathsf{td}$ for all $(\mathsf{td}, \mathsf{pub})$ output by TdGen and all $a, a' \in A(\mathsf{pub})$ with $a \overset{x}{\sim} a'$.

Surprisingly, extractability of 2:1-TDFs has an essential effect on the relationship between INV−1 and INV−2 security notions. In combination with Lemma 2, we see that notions INV−1 and INV−2 are equivalent for (samplable)

$\mathsf{KGen}(1^\lambda)$:
1 $(\mathsf{td}, \mathsf{pub}) \leftarrow_R \mathsf{TdGen}(1^{\lambda_2})$
2 $K \leftarrow_R \{0,1\}^{\lambda_f}$
3 Return $(\mathsf{sk}, \mathsf{vk}) = ((\mathsf{td}, K), \mathsf{pub})$

$\mathsf{Sign}(\mathsf{sk}, \mathsf{msg})$:
1 $b \leftarrow H_{\mathsf{pub}}(\mathsf{msg})$
2 $d \leftarrow F(K, \mathsf{msg})$
3 $\sigma \leftarrow \mathsf{Reverse}(\mathsf{td}, b, d)$
4 Return σ

$\mathsf{Ver}(\mathsf{vk}, \mathsf{msg}, \sigma)$:
1 $b \leftarrow H_{\mathsf{pub}}(\mathsf{msg})$
2 If $\mathsf{Apply}(\mathsf{pub}, \sigma) = b$
3 then return 1
4 else return 0

Fig. 7 Signature scheme $2:1\text{−}\mathsf{Sig}$

extractable 2:1-TDFs. The proof of Lemma 3 appears in "Proofs from Section 4" of Appendix 3.

Lemma 3 (INV−1 ⇒ INV−2 if 2:1-TDF extractable) *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* $\mathbf{Succ}^{\mathrm{INV}-2}_{X,\mathcal{B}}(\lambda) = \mathbf{Succ}^{\mathrm{INV}-1}_{X,\mathcal{A}}(\lambda)$.

5 Tightly secure signatures from 2:1 trapdoor functions

As a first application of our new primitive, we present a signature scheme. The construction essentially extends the well-known full-domain hash paradigm (FDH, see [8]) from bijective trapdoor functions, e.g. TDPs, to the new 2:1-TDF notion. While security reductions of regular FDH signatures are necessarily highly untight (the security loss in comparison with TDP security is q_H, the number of adversary's hash function evaluations; but see [15] for tighter bounds for homomorphic TDPs), it turns out that our new signature scheme is *tightly* secure if the deployed 2:1-TDF is extractable. We note that our construction is similar in spirit with the tightly secure signature constructions found in [4,21]; however, our abstraction model is quite different.

Construction 1 (Signatures from 2:1-TDF) *Let λ be a security parameter, and let λ_2 and λ_f be parameters polynomially dependent on λ. Let* $X = (\mathsf{TdGen}, \mathsf{Sample}_A, \mathsf{Sample}_B, \mathsf{Apply}, \mathsf{Reverse}, \mathsf{Decide})$ *be an extractable 2:1 trapdoor function and let* $F : \{0,1\}^{\lambda_f} \times \{0,1\}^* \to \{0,1\}$ *be a PRF. For each* pub *output by* TdGen*, let* $H_{\mathsf{pub}} : \{0,1\}^* \to B(\mathsf{pub})$ *be a hash function. Signature scheme* $2:1\text{−}\mathsf{Sig}$ *consists of the algorithms specified in Fig. 7.*

We assess the security of $2:1\text{−}\mathsf{Sig}$ in Theorem 1; the corresponding proof appears in "Proofs from Section 5" of Appendix. A factor of q_H appears in the bounds claimed in Theorem 1. That factor scales a purely statistical quantity that can easily be forced to be smaller than (say) 2^{-256}, and this matches the notion of "tightness" in works like [4] (in which the sampling issues that we factor in are actually left implicit).

Theorem 1 ($2:1\text{−}\mathsf{Sig}$ is EUF) *In the setting of* Construction 1*, if X is extractable, samplable, and second preimage resistant, F is a secure PRF, and H_{pub} is a random oracle, then signature scheme* $2:1\text{−}\mathsf{Sig}$ *is existentially unforgeable under adaptive chosen message attacks. More precisely, for any efficient* EUF *algorithm $\mathcal{A}$ making at most q_H queries to the $H_{\mathsf{pub}}(\cdot)$ oracle, there exist efficient distinguishers $\mathcal{D}_A$, $\mathcal{D}_B$, and $\mathcal{C}$, and an efficient algorithm $\mathcal{B}$ such that*

$$\begin{aligned}\mathbf{Succ}^{\mathrm{EUF}}_{2:1-\mathsf{Sig},\mathcal{A}}(\lambda) \le\ & q_H \cdot \mathbf{Dist}^{\mathsf{Sample}_A, U(A)}_{X,\mathcal{D}_A}(\lambda_2) \\ & + q_H \cdot \mathbf{Dist}^{\mathsf{Sample}_B, U(B)}_{X,\mathcal{D}_B}(\lambda_2) \\ & + 2 \cdot \mathbf{Succ}^{\mathrm{INV}-2}_{X,\mathcal{B}}(\lambda_2) + \mathbf{Adv}^{\mathrm{prf}}_{F,\mathcal{C}}(\lambda_f).\end{aligned}$$

6 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 the *group of sign-agnostic quadratic residues*, a specific structure from number theory. This group was introduced to cryptography by Goldwasser, Micali, and Rivest in [22], and rediscovered 20 years later by Hofheinz and Kiltz [25]. We first reproduce the results of [22,25] and then extend them towards our requirements. [5]

In our exposition, we assume that the reader is familiar with definition and structure of groups $\mathbb{Z}_n^\times$, J_n, and QR_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 Fig. 8 (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 Facts 1–6 and Corollary 2, in "Appendix 1".

6.1 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

[5] 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*.

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 modelled as random oracles. However, such properties are a prerequisite in certain applications in cryptography [25], what 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 become feasible.

The use of sign-agnostic quadratic residues in cryptography is explicitly proposed in [22,25]. 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. The introduction to sign-agnostic quadratic residues that we give in the following paragraphs uses a new and more consistent notation that aims at making the algebraical structure more readily apparent. Using this new notation, it will not be difficult to establish Lemmas 5–8 below.

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 Fig. 8 (right). After agreeing on notations $\{\pm x\} = \{x, -x\}$ and $\{\pm x\}^2 = \{\pm(x^2)\}$, we additionally obtain the following result (proven in "Proofs from Section 6" of Appendix 3):

Lemma 4 *Let n be a Blum integer. Then $QR_n/_{\pm 1} = \big\{\{\pm x\}^2 : \{\pm x\} \in \mathbb{Z}_n^\times/_{\pm 1}\big\}$.*

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 $x \in \mathbb{Z}_n^\times$.

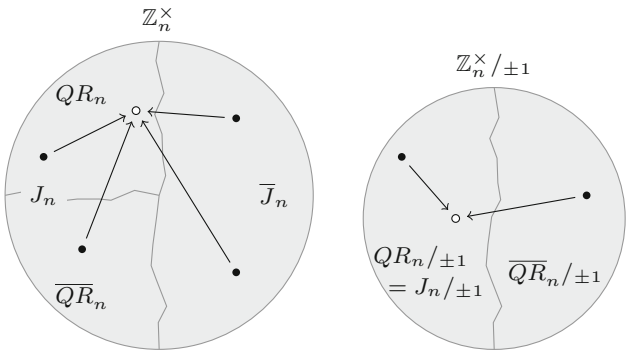

Fig. 8 Illustration of $\mathbb{Z}_n^\times$ and $\mathbb{Z}_n^\times/_{\pm 1}$ (for Blum integers n), and subgroups QR_n, J_n, and $J_n/_{\pm 1} = QR_n/_{\pm 1}$. Also visualized is the action of the squaring operation (see Corollaries 1 and 2)

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

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

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 Lemmas 5 and 6, which directly correspond with Facts 4 and 5 from "Appendix 1". Similarly, Corollaries 1 and 2 correspond. We stress that the following results do not appear in [22,25]; the corresponding proofs appear in "Proofs from Section 6" of Appendix 3.

Lemma 5 (Square roots in $\mathbb{Z}_n^\times/_{\pm 1}$) *If n is 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 $\mathbb{Z}_n^\times/_{\pm 1}$, $QR_n/_{\pm 1}$, $\overline{QR}_n/_{\pm 1}$) *If n is 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* Fig. 8 *(right).*

Lemma 6 (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 7 (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 $QR_n/_{\pm 1}$. Elements in $\overline{QR}_n/_{\pm 1}$ 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* Eqs. (1) and (2).

We anticipate the following result from "Appendix 2".

Lemma 8 (Indifferentiable hashing into $QR_n/_{\pm 1}$) *There exist efficient functions ℓ and $\varphi\colon \{0,1\}^{\ell(n)} \to QR_n/_{\pm 1}$ such that if a hash function $h\colon \{0,1\}^* \to \{0,1\}^{\ell(n)}$ behaves like a (programmable) random oracle then so does $H = \varphi \circ h\colon \{0,1\}^* \to QR_n/_{\pm 1}$.*

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 corresponding decoding procedure is $\overline{x} \mapsto \{\overline{x}, -\overline{x}\}$.

6.2 Construction of Blum-2:1-TDF from sign-agnostic quadratic residues

We use the tools from Sect. 6.1 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 the squaring operation, extractability will be possible given distinct square roots of an element.

Construction 2 (Blum-2:1-TDF) *Define algorithms* $\mathsf{Blum-2:1-TDF} = (\mathsf{TdGen}, \mathsf{Sample}_A, \mathsf{Sample}_B, \mathsf{Apply}, \mathsf{Reverse}, \mathsf{Decide}, \mathsf{Extract})$ *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 $B(\mathsf{pub}) := QR_n/_{\pm 1}$.*
- $\mathsf{Sample}_A(\mathsf{pub}, d)$*: Implement* $\mathsf{Sample}_A(\mathsf{pub}, 0)$, $\mathsf{Sample}_A(\mathsf{pub}, 1)$, *and* $\mathsf{Sample}_A(\mathsf{pub})$ *using the samplers for sets $QR_n/_{\pm 1}$, $\overline{QR}_n/_{\pm 1}$, and $\mathbb{Z}_n^\times/_{\pm 1}$ from* Lemma 7.
- $\mathsf{Sample}_B(\mathsf{pub})$*: Implement* $\mathsf{Sample}_B(\mathsf{pub})$ *using the sampler for set $QR_n/_{\pm 1}$ from* Lemma 7.
- $\mathsf{Apply}(\mathsf{pub}, \{\pm a\})$*: Return* $\{\pm b\} \leftarrow \{\pm a\}^2$.
- $\mathsf{Reverse}(\mathsf{td}, \{\pm b\}, d)$*: By* Lemma 5*, element $\{\pm b\} \in QR_n/_{\pm 1}$ has precisely 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 various security properties follow straightforwardly from the number-theoretic facts established in Sect. 6.1. The proof appears in "Proofs from Section 6" of Appendix 3.

Theorem 2 (Security and extractability of Blum-2:1-TDF) Blum-2:1-TDF *is samplable* (Def. 15)*, (second) preimage resistant* (Def. 16) *under the assumption that factoring is hard, and extractable* (Def. 17)*.*

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

7 Double-authentication-preventing signatures from extractable 2:1-TDFs

We now come to the central result of this paper, a double-authentication-preventing signature generically constructed from any extractable 2:1 trapdoor function; of course factoring-based Blum-2:1-TDF from the previous section is a suitable candidate for instantiating the scheme.

Construction 3 (DAPS from 2:1-TDF) *Let λ be a security parameter, and let λ_2 and λ_h be parameters polynomially dependent on λ. Let $X = (\mathsf{TdGen}, \mathsf{Sample}_A, \mathsf{Sample}_B, \mathsf{Apply}, \mathsf{Reverse}, \mathsf{Decide})$ 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* $\mathsf{2:1-DAPS}$ *consists of the algorithms specified in* Fig. 9.

The basic idea of the signing algorithm is as follows. From any given subject, the signer derives message-independent signing elements $b_1, \ldots, b_{\lambda_h} \in B$. 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 in 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. Moreover, 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. But, assuming collision resistance of $H^\#$, at least one different d_i is used in the two signatures, so two distinct preimages of b_i are involved, which allows anyone to recover the trapdoor.

We give further explanation on the subject-dependent value s that we embed into every signature. Consider the standard security reduction for proving FDH-TDP signatures unforgeable [9], and in particular how adversary's queries to random oracle H are answered. Usually, random oracle H is programmed such that $H(m) = g(x)$, where m is

$\mathsf{KGen}(1^\lambda)$: 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 $(d_1, \ldots, d_{\lambda_h})\ H^{\#}(\mathsf{subj}, s, \mathsf{msg}) \leftarrow$
3 For $1 \le i \le \lambda_h$:
4 $\quad b_i \leftarrow H_{\mathsf{pub}}(\mathsf{subj}, s, i)$
5 $\quad a_i \leftarrow \mathsf{Reverse}(\mathsf{td}, b_i, d_i)$
6 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 $1 \le i \le \lambda_h$:
6 $\quad$ If $\mathsf{Apply}(\mathsf{pub}, a_i) \neq H_{\mathsf{pub}}(\mathsf{subj}, s, i)$, return 0
7 $\quad$ If $\mathsf{Decide}(\mathsf{pub}, a_i) \neq d_i$, return 0
8 Return 1

Fig. 9 Double-authentication-preventing signature scheme $2:1\text{−}\mathsf{DAPS}$

the queried message, g is the TDP, and x is sampled uniformly from the domain of g. This construction exploits that g (as opposed to g^{-1}) can be efficiently computed without knowledge of any trapdoor, and it ensures that the simulation 'knows' the preimage of hash values $H(m)$, for all messages m. When switching to 2:1-TDFs, however, we observe that this method of reduction does not work satisfyingly: While for any H query a corresponding preimage $a \in A$ of the 2:1-TDF could be uniformly sampled, it might be related value $a' \in A$, $a \overset{x}{\sim} a'$, that needs to be revealed in later queries to the signing oracle. But computing a' from a, or even jointly sampling them, is infeasible without knowledge of 2:1-TDF's trapdoor. In our DAPS construction, value s ensures that the simulation is not required to program H_{pub} oracle until the point where it learns subj and msg, i.e. learns which preimage it will have to reveal. For further details, we refer to the proof of Theorem 3.

7.1 Unforgeability of $2:1\text{−}\mathsf{DAPS}$

We next establish existential unforgeability of $2:1\text{−}\mathsf{DAPS}$ (cf. Definition 7). 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; these changes cannot be detected if 2:1-TDF X is samplable. From any forgery crafted by adversary $\mathcal{A}$, either a preimage or second preimage of X, or a collision of $H^{\#}$ can be extracted. Observe that, by Lemma 2, it suffices to require second preimage resistance of X in Theorem 3. The detailed proof appears in "Proof of unforgeability (Theorem 3)" of Appendix 3.

Theorem 3 ($2:1\text{−}\mathsf{DAPS}$ is EUF) *In the setting of* Construction 3*, if X is samplable and second preimage resistant, $H^{\#}$ is collision resistant, and H_{pub} is a random oracle, then double-authentication-preventing signature* $2:1\text{−}\mathsf{DAPS}$ *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 distinguishers $\mathcal{D}_A$ and $\mathcal{D}_B$ and efficient algorithms $\mathcal{B}_1$, $\mathcal{B}_2$, and $\mathcal{C}$ such that*

$$\begin{aligned}\mathbf{Succ}^{\mathrm{EUF}}_{2:1\text{−}\mathsf{DAPS},\mathcal{A}}(\lambda) \;\le\; & (q_1 + (\lambda_h+1)q_S + 1)\,\mathbf{Dist}^{\mathsf{Sample}_{A,U(A)}}_{X,\mathcal{D}_A}(\lambda_2) + \\ & (q_1 + (\lambda_h+1)q_S)\,\mathbf{Dist}^{\mathsf{Sample}_{B,U(B)}}_{X,\mathcal{D}_B}(\lambda_2) + \\ & q_1\mathbf{Succ}^{\mathrm{INV}-1}_{X,\mathcal{B}_1}(\lambda_2) + 2q_S\lambda_h\,\mathbf{Succ}^{\mathrm{INV}-2}_{X,\mathcal{B}_2}(\lambda_2) + \\ & \mathbf{Succ}^{\mathrm{CR}}_{H^{\#},\mathcal{C}}(\lambda_h),\end{aligned}$$

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

Remark 3 ($2:1\text{−}\mathsf{DAPS}$ is deterministic and S−EUF) Note that $2:1\text{−}\mathsf{DAPS}$ is not only deterministic but also unique [15] and, in particular, strongly unforgeable.

7.2 Double-signature extractability of $2:1\text{−}\mathsf{DAPS}$

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 4; the detailed proof appears in "Proof of double-signature extractability (Theorem 4)" of Appendix 3.

Theorem 4 ($2:1\text{−}\mathsf{DAPS}$ is DSE^*) *In the setting of* Construction 3*, if X is extractable and $H^{\#}$ is collision resistant, then double-authentication-preventing signature* $2:1\text{−}\mathsf{DAPS}$ *is double-signature extractable with trusted set-up. More precisely, with the notation from* Theorem 3*, there exists an efficient algorithm* $\mathsf{Extract}$ *(described in the proof of the theorem) and an efficient algorithm $\mathcal{C}$ such that, for all algorithms $\mathcal{A}$,* $\mathbf{Succ}^{\mathrm{DSE}^*}_{\mathsf{DAPS},\mathcal{A}}(\lambda) \le \mathbf{Succ}^{\mathrm{CR}}_{H^{\#},\mathcal{C}}(\lambda_h)$.

$\mathsf{KGen}(1^\lambda)$:
1 Pick random Blum integer $n = pq$ of length λ such that $p < q$.
2 Pick $t \in \mathbb{Z}_n^\times$ with $\left(\frac{t}{n}\right) = -1$.
3 Return $\mathsf{sk} \leftarrow (p, q)$ and $\mathsf{vk} \leftarrow (n, t)$.

Let $H^\# : \{0,1\}^* \to \{0,1\}^{\lambda_h}$ be a cryptographic hash function.

Let $H_{\mathsf{vk}} : \{0,1\}^* \to QR_n/_{\pm 1}$ using the construction in Appendix B. Specifically, if $h : \{0,1\}^* \to \{0,1\}^\ell$ is a cryptographic hash function with $\ell \gg \lambda$, then:

$H_{\mathsf{vk}}(z)$:
1 $r \leftarrow h(z) \in [0\,..\,2^\ell - 1]$ (interpreting the ℓ-bit string $h(z)$ as an integer)
2 $x \leftarrow r \bmod n$
3 If $\left(\frac{x}{n}\right) = +1$, return $\{\pm x\} \in QR_n/_{\pm 1}$
4 If $\left(\frac{x}{n}\right) = -1$, return $\{\pm xt\} \in QR_n/_{\pm 1}$

$\mathsf{Sign}(\mathsf{sk}, \mathsf{subj}, \mathsf{msg})$:
1 $y \leftarrow H_{\mathsf{vk}}(\mathsf{subj}) \in QR_n/_{\pm 1}$
2 Compute $s \leftarrow x_0 \in QR_n/_{\pm 1}$ such that $x_0^2 = y$ using factors (p, q) of n and the Chinese Remainder Theorem.
3 $(d_1, \ldots, d_{\lambda_h}) \leftarrow H^\#(\mathsf{subj}, s, \mathsf{msg}) \in \{0,1\}^{\lambda_h}$
4 For $1 \le i \le \lambda_h$:
5 $\quad b_i \leftarrow H_{\mathsf{vk}}(\mathsf{subj}, s, i) \in QR_n/_{\pm 1}$
6 $\quad$ If $d_i = 0$: compute $a_i \in QR_n/_{\pm 1}$ such that $a_i^2 = b_i$ using factors (p, q) of n and the Chinese Remainder Theorem.
7 $\quad$ If $d_i = 1$: compute $a_i \in \overline{QR}_n/_{\pm 1}$ such that $a_i^2 = b_i$ using factors (p, q) of n and the Chinese Remainder Theorem.
8 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 $\left(\frac{s}{n}\right) \neq +1$, return 0
3 $s' \leftarrow H_{\mathsf{vk}}(\mathsf{subj}) \in QR_n/_{\pm 1}$
4 If $s^2 \neq s'$, return 0
5 $(d_1, \ldots, d_{\lambda_h}) \leftarrow H^\#(\mathsf{subj}, s, \mathsf{msg}) \in \{0,1\}^{\lambda_h}$
6 For $1 \le i \le \lambda_h$:
7 $\quad a' \leftarrow H_{\mathsf{vk}}(\mathsf{subj}, s, i) \in QR_n/_{\pm 1}$
8 $\quad$ If $a_i^2 \neq a'$, return 0
9 $\quad$ If $d_i = 0$ and $\left(\frac{a_i}{n}\right) \neq +1$, return 0
10 $\quad$ If $d_i = 1$ and $\left(\frac{a_i}{n}\right) \neq -1$, return 0
11 Return 1

Fig. 10 Double-authentication-preventing signature scheme $2:1\text{−}\mathsf{DAPS}$ instantiated using sign-agnostic quadratic residues (Blum-2:1-TDF). In algorithm Ver, for any element $\bar{x} = \{\pm x\} \in \mathbb{Z}_n^\times/_{\pm 1}$ we write $\left(\frac{\bar{x}}{n}\right)$ as shortcut for $\left(\frac{x}{n}\right) = \left(\frac{-x}{n}\right)$

Observe that the double-signature extractability of $2:1\text{−}\mathsf{DAPS}$ in Theorem 4 relies on the assumption that signer's verification key is well-formed. When instantiated with Blum-2:1-TDF (see Sect. 7.3), this means assuming that signer's public information n is a Blum integer, as extractability of Blum-2:1-TDF is guaranteed only in this case. Well-formedness can be shown using interactive or non-interactive zero-knowledge proofs. In particular, there is an interactive zero-knowledge protocol of van de Graaf and Peralta [42] for demonstrating that an integer n is of the form $p^r q^s$ where p and q are both primes such that $p \equiv q \equiv 3 \bmod 4$, which can be combined with the interactive protocol of Boyar et al. [6] for demonstrating that an integer n is square free, to ultimately show that a modulus n is a Blum integer. Alternatively, a non-interactive zero-knowledge proof for the well-formedness of a Blum integer was given by De Santis et al. [16] and for products of safe primes (which includes Blum integers) by Camenisch and Michels [14].

7.3 DAPS instantiation based on sign-agnostic quadratic residues

Figure 10 shows the instantiation of $2:1\text{−}\mathsf{DAPS}$ from Fig. 9 using the Blum-2:1-TDF from Construction 2. In Table 1, we analyse the corresponding sizes of verification keys, signing keys, and signatures, and the cost of signature generation and verification, with abstract results as well as results with 1024- and 2048-bit keys. We assume the element representation from Remark 1 and the verification key optimization from Remark 2.

We also report the results of our implementation of DAPS using the libgcrypt cryptographic library.[6] 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

[6] 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	$(\lambda_h + 1) \cdot$ Jac, $(\lambda_h + 1) \cdot$ 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	$(2\lambda_h + 2) \cdot$ Jac, $(\lambda_h + 1) \cdot$ sqr	0.105 s	0.276 s

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

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 our scheme is less efficient than a regular signature scheme, we believe these timings are still in the acceptable range; this holds in particular if our scheme is used to implement CA functionality where signature generation happens rarely and verification results can be cached.

8 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 or the "compelled certificate creation attack" [40]. 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 the potential for increased damage in the case of accidental errors by the signer.

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.

8.1 Certificate authorities

The potential use of DAPS for certificate authorities has been discussed in some detail in the Introduction.

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 keeping track of the previous subjects signed and refusing to sign duplicates. In commercial certificate authorities, where the 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‖2015`") 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‖20150501-20160430`" versus "`www.example.com‖20150502-20160501`").

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‖2015‖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. We could consider a multi-CA PKI in which other DAPS-based CAs agree to issue a "void certificate" for a domain name when presented with a valid certificate from another CA, thereby disqualifying them from issuing future signatures on that subject. In general, though, coordination of CAs is challenging. We believe it remains a very interesting open question to find cryptographic constructions that solve the multi-CA PKI problem.

8.2 Time-stamping

A standard approach to preventing time-stamping authorities from "changing the past" is to require that, when a digital signature is constructed that asserts 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 statefully track the list of all signed subjects, since time periods automatically increment.

8.3 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 $2:1-\mathsf{DAPS}$ 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 of the construction is indispensable.

9 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 forge arbitrary signatures or even fully recover the signer's secret key. Although this leads to less robustness in the face of accidental behaviour, it also provides a mechanism for *self-enforcement* of correct behaviour and gives trusted signers such as certificate authorities an argument to resist coercion and the compelled certificate creation attack.

Our construction is based on a new primitive called *extractable 2:1 trapdoor function*. 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. This includes the cases of 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 information.

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?

Acknowledgements Parts of this research were funded by EPSRC Leadership Fellowship EP/H005455/1 (for the first author), and by the Australian Technology Network and German Academic Exchange Service (ATN-DAAD) Joint Research Co-operation Scheme (for the second author). This work has also been supported by the European Commission through the ICT Programme under Contract ICT-2007-216676 ECRYPT II and by Australian Research Council (ARC) Discovery Project DP130104304.

Appendix 1: Basic results from number theory

We recall some definitions and results (without proof) from number theory as well as establish notation that we use in the paper. We refer the reader to classic textbooks on cryptography [36, Ch. 2–3],[29, Ch. 7, 11], or on number theory [27] for details.

Fact 1 (Quadratic residues modulo p) *If p is a prime number, the* group of quadratic residues *modulo p is denoted by $QR_p = \left\{x^2 : x \in \mathbb{Z}_p^\times\right\}$. The* Legendre symbol $\left(\frac{\cdot}{p}\right) : \mathbb{Z}_p^\times \to \{-1, 1\} : a \mapsto \left(\frac{a}{p}\right) = a^{(p-1)/2}$ *serves as an indicator function for QR_p: $a \in QR_p \Leftrightarrow \left(\frac{a}{p}\right) = 1$. We have $|QR_p| = |\mathbb{Z}_p^\times|/2 = (p-1)/2$. If $p \equiv 3 \mod 4$ then $-1 \notin QR_p$, in which case $\left(\frac{-a}{p}\right) = -\left(\frac{a}{p}\right)$ for all $a \in \mathbb{Z}_p^\times$. The Legendre symbol can be efficiently computed.*

Fact 2 (Structure of $\mathbb{Z}_n$ and $\mathbb{Z}_n^\times$) *Let n be an RSA modulus, that is, $n = pq$ is the product of distinct prime numbers p and q. When $p \equiv q \equiv 3 \mod 4$, n is called a* Blum integer. *The* Chinese Remainder Theorem *states that $\mathbb{Z}_n \cong \mathbb{Z}_p \times \mathbb{Z}_q$ (as rings), and hence $\mathbb{Z}_n^\times \cong \mathbb{Z}_p^\times \times \mathbb{Z}_q^\times$ (as groups). An isomorphism $\psi : \mathbb{Z}_n \to \mathbb{Z}_p \times \mathbb{Z}_q$ is given by $x \mapsto (x \mod p, x \mod q)$. Both ψ and ψ^{-1} can be efficiently computed if the factors of $n = pq$ are known.*

Fact 3 (Quadratic residues modulo n) *Let $n = pq$ be an RSA modulus. Then $QR_n = \left\{x^2 : x \in \mathbb{Z}_n^\times\right\}$ denotes the* group of quadratic residues *modulo n. The* Jacobi symbol $\left(\frac{\cdot}{n}\right) : \mathbb{Z}_n^\times \to \{-1, 1\} : a \mapsto \left(\frac{a}{n}\right)$ *is defined by $\left(\frac{a}{n}\right) = \left(\frac{a \mod p}{p}\right)\left(\frac{a \mod q}{q}\right)$. Although $\left(\frac{a}{n}\right) = 1$ for all $a \in QR_n$, the Jacobi symbol does not serve as an indicator for QR_n: if n is a Blum integer, then $\left(\frac{-1}{n}\right) = 1$ and thus $\left(\frac{a}{n}\right) = \left(\frac{-a}{n}\right)$ for all $a \in \mathbb{Z}_n^\times$, but fact $a \in QR_n \Rightarrow -a \notin QR_n$ implies that at most one of a, a' can be in QR_n. If n is a Blum integer such that $p \equiv 3 \mod 8$ and $q \equiv 7 \mod 8$, then $\left(\frac{2}{n}\right) = -1$. The Jacobi symbol can be efficiently computed, even if the factorization of n is not known.*

The set $J_n = \left\{a \in \mathbb{Z}_n^\times : \left(\frac{a}{n}\right) = 1\right\}$ is a subgroup of $\mathbb{Z}_n^\times$, and QR_n is a subgroup of J_n. Define $\overline{J}_n = \mathbb{Z}_n^\times \setminus J_n$ and $\overline{QR}_n = J_n \setminus QR_n$. If we set $\varphi(n) = (p-1)(q-1)$ then $|\mathbb{Z}_n^\times| = \varphi(n)$, $|J_n| = |\overline{J}_n| = \varphi(n)/2$, and $|QR_n| = |\overline{QR}_n| = \varphi(n)/4$. These relations are illustrated in Fig. 8 *(left).*

Fact 4 (Square roots in $\mathbb{Z}_n^\times$) *Let n be an RSA modulus. Every element $y \in QR_n$ has exactly four square roots in $\mathbb{Z}_n^\times$, namely $\{\pm x_0, \pm x_1\}$, where $x_0, x_1 \in \mathbb{Z}_n^\times$. If n is a Blum integer, then $\left(\frac{x_0}{n}\right) \neq \left(\frac{x_1}{n}\right)$ and exactly one of $\{\pm x_0, \pm x_1\}$ is in QR_n. Since $(x_0 - x_1)(x_0 + x_1) \equiv x_0^2 - x_1^2 \equiv y - y \equiv 0 \mod n$, non-trivial divisors of n are given by $\gcd(n, x_0 - x_1)$ and $\gcd(n, x_0 + x_1)$. Square roots modulo n can be efficiently computed if the factors of $n = pq$ are known.*

Corollary 2 (Squaring in $\mathbb{Z}_n^\times$, J_n, and QR_n) *Let n be an RSA modulus. The squaring operation $\mathbb{Z}_n^\times \to QR_n : x \mapsto x^2$ is a 4:1 mapping. If n is a Blum integer, then squaring is a 2:1 function from J_n to QR_n, while squaring is a 1:1 function both from QR_n to QR_n and from $\overline{QR}_n$ to QR_n. These relations are illustrated in* Fig. 8 *(left).*

Fact 5 (Computing square roots in $\mathbb{Z}_n^\times$ is hard) *Let n be an RSA modulus. Computing square roots modulo n is as hard as factoring n. In particular, given an algorithm $\mathcal{A}$ that computes square roots of elements in QR_n, factors of n can be found by randomly picking $x \leftarrow_R \mathbb{Z}_n^\times$ and running $x' \leftarrow_R \mathcal{A}(n, x^2)$ to obtain a second, potentially different, square root of x^2. With probability $1/2$, $x' \not\equiv \pm x$; by* Fact 4, *a non-trivial factor of n is given by $\gcd(n, x - x')$.*

Fact 6 (Samplability and decidability of $\mathbb{Z}_n$, $\mathbb{Z}_n^\times$, J_n, $\overline{J}_n$) *Let $n = pq$ be an RSA modulus, $t \in \mathbb{Z}_n^\times$ a fixed element with $\left(\frac{t}{n}\right) = -1$, and $\ell \gg \log n$. Identify set $\{0,1\}^\ell$ with $[0, 2^\ell - 1]$ using a canonical bijection and consider functions*

$$E : \{0,1\}^\ell \to \mathbb{Z}_n : r \mapsto r \mod n$$

$$F : \mathbb{Z}_n^\times \to J_n : x \mapsto \begin{cases} x & \text{if } \left(\frac{x}{n}\right) = +1 \\ xt & \text{if } \left(\frac{x}{n}\right) = -1 \end{cases}.$$

A common method (see [17,39] *and* [37, §B.5.1.3]*) for sampling random elements x from $\mathbb{Z}_n$ is to pick a seed $r \leftarrow_R \{0,1\}^\ell$ and to output $x = E(r)$. The resulting distribution is statistically close to uniform* [39]*. If p and q grow exponentially in a security parameter, then $|\mathbb{Z}_n^\times|/|\mathbb{Z}_n| = 1 - (p + q - 1)/pq$ becomes negligibly close to 1, so function E can be used without modification for sampling from $\mathbb{Z}_n^\times$ with a distribution statistically close to uniform. Note that membership in $\mathbb{Z}_n^\times$ can be efficiently decided since $\mathbb{Z}_n^\times = \{x \in \mathbb{Z}_n : \gcd(x, n) = 1\}$.*

Elements in J_n and $\overline{J}_n$ can be efficiently recognized by evaluating the Jacobi symbol. Moreover, it is not difficult to see that elements y can be uniformly sampled from J_n by picking a random $x \leftarrow_R \mathbb{Z}_n^\times$ and outputting $y = F(x)$. Elements from $\overline{J}_n$ can be sampled in a similar fashion.

It is widely believed that, unless the factorization of n is known, 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 these samples.

Appendix 2: Indifferentiable hashing onto $QR_n/\pm 1$

Specific applications of the group of sign-agnostic quadratic residues modulo a Blum integer n, including our constructions in Sects. 5 and 7, might rely on the existence of a

hash function $H : \{0,1\}^* \to QR_n/_{\pm 1}$. Moreover, the corresponding security arguments might require modelling H as a random oracle, as in Theorem 3. We show in the following how to construct such hash functions onto $QR_n/_{\pm 1}$ from a hash function onto bitstrings.

Let $\ell \gg \log n$ be an integer and assume $h : \{0,1\}^* \to \{0,1\}^\ell$ is a hash function that may be modelled as a random oracle. From h, we construct $H : \{0,1\}^* \to QR_n/_{\pm 1}$ as $H = G \circ F \circ E \circ h$, where

$$E : \{0,1\}^\ell \to \mathbb{Z}_n \quad , \quad F : \mathbb{Z}_n^\times \to J_n \quad , \quad G : J_n \to QR_n/_{\pm 1}$$

are the functions specified by Fact 6 and (implicitly) by Lemma 7 (observe that E maps onto $\mathbb{Z}_n$, not onto $\mathbb{Z}_n^\times$ as syntactically required for composing E with F; however, as described in Fact 6, operations involving elements from $\mathbb{Z}_n$ are statistically indistinguishable from operations involving elements from $\mathbb{Z}_n^\times$).

This method of constructing hash functions follows Boneh and Franklin [5] and Brier et al. [2,3]. Specifically, Brier et al. show that if function $\varphi = G \circ F \circ E$ is an *admissible encoding* and h is a random oracle, then $H = \varphi \circ h$ is indifferentiable from a random oracle [2, §3]. To program H, we can take a preimage of φ and program h accordingly. We reproduce the definition and main theorem from [2] as follows.

Definition 18 (*Admissible encoding* [2]) A function $\varphi : S \to R$ between finite sets is an *admissible encoding* if it satisfies the following properties:

(1) *Computable:* φ is computable in deterministic polynomial time.
(2) *Regular:* for s uniformly distributed in S, the distribution of $\varphi(s)$ is statistically indistinguishable from the uniform distribution in R.
(3) *Samplable:* there is an efficient randomized algorithm $\mathcal{I}$ such that, for any $r \in R$, $\mathcal{I}(r)$ induces a distribution that is statistically indistinguishable from the uniform distribution in $\varphi^{-1}(r)$.

Theorem 5 (Construction of random oracle [2]) *Let $\varphi : S \to R$ be an admissible encoding. If $h : \{0,1\}^* \to S$ is a random oracle, then the construction $H(m) = \varphi(h(m))$ is statistically indifferentiable from a random oracle.* □

As admissibility is a transitive property, it suffices to show that E, F, G are admissible encodings. Define corresponding inversion algorithms $\mathcal{I}_E, \mathcal{I}_F, \mathcal{I}_G$ as

$$\mathcal{I}_E : \mathbb{Z}_n \to [0, 2^\ell - 1] : x \mapsto x + kn \quad (k \leftarrow_R [0, \lfloor 2^\ell / n \rfloor - 1])$$
$$\mathcal{I}_F : J_n \to \mathbb{Z}_n^\times : x \mapsto x/t^b \quad (\text{where } b \leftarrow_R \{0,1\})$$
$$\mathcal{I}_G : QR_n/_{\pm 1} \to J_n : \{\pm x\} \mapsto (-1)^b \cdot x \quad (b \leftarrow_R \{0,1\})$$

(and observe that $\mathcal{I}_G$ is actually well-defined). Functions E, F, G and inversion algorithms $\mathcal{I}_E, \mathcal{I}_F, \mathcal{I}_G$ are clearly efficient. While the regularity of F and G is obvious, function E is regular by Fact 6. It is also easy to see that E, F, G are samplable. Thus E, F, G are admissible encodings, and so is $\varphi = G \circ F \circ E$. Hence $H = \varphi \circ h : \{0,1\}^* \to QR_n/_{\pm 1}$ behaves like a random oracle by Theorem 5.

Appendix 3: Proofs

Proofs from Section 4

Proof of Lemma 1

We only prove the first inequality; the remaining two follow analogously. Define the required distinguishers as $\mathcal{D}'_A(a) = \mathcal{D}_B(\mathsf{Apply}(a))$ and $\mathcal{D}'_B(b) = \mathcal{D}_B(b)$, where we assume implicit parameter 'pub'. After observing that Apply is 2:1 and hence we have that $\mathbf{Dist}_{X,\mathcal{D}}^{U(B),\mathsf{Apply}(U(A))}(\lambda) = 0$ for any distinguisher $\mathcal{D}$, the triangle inequality shows

$$\begin{aligned}\mathbf{Dist}_{X,\mathcal{D}_B}^{\mathsf{Sample}_B,\mathsf{Sample}_B^A}(\lambda) &\le \mathbf{Dist}_{X,\mathcal{D}_B}^{\mathsf{Sample}_B,U(B)}(\lambda) \\ &\quad + \mathbf{Dist}_{X,\mathcal{D}_B}^{U(B),\mathsf{Apply}(U(A))}(\lambda) \\ &\quad + \mathbf{Dist}_{X,\mathcal{D}_B}^{\mathsf{Apply}(U(A)),\mathsf{Apply}(\mathsf{Sample}_A)}(\lambda) \\ &= \mathbf{Dist}_{X,\mathcal{D}'_B}^{\mathsf{Sample}_B,U(B)}(\lambda) \\ &\quad + \mathbf{Dist}_{X,\mathcal{D}'_A}^{U(A),\mathsf{Sample}_A}(\lambda).\end{aligned}$$

□

Proof of Lemma 2

Construct INV−2 algorithm $\mathcal{B}$ and distinguisher $\mathcal{D}_B$ as follows: Upon receiving (pub, a), $\mathcal{B}$ computes $b \leftarrow \mathsf{Apply}(\mathsf{pub}, a)$ and outputs $a' \leftarrow_R \mathcal{A}(\mathsf{pub}, b)$. For any element b to be decided, $\mathcal{D}_B$ outputs 1 iff $\mathsf{Apply}(\mathsf{pub}, \mathcal{A}(\mathsf{pub}, b)) = b$. Inspection shows

$$\begin{aligned}&\mathbf{Dist}_{X,\mathcal{D}_B}^{\mathsf{Sample}_B,\mathsf{Sample}_B^A}(\lambda) \\ &= \left| \begin{array}{l} \Pr[(\mathsf{td},\mathsf{pub}) \leftarrow_R \mathsf{TdGen}(1^\lambda); b \leftarrow_R \mathsf{Sample}_B(\mathsf{pub}) : \\ \quad \mathcal{D}_B(\mathsf{pub}, b) = 1] - \Pr[(\mathsf{td},\mathsf{pub}) \leftarrow_R \mathsf{TdGen}(1^\lambda); \\ \quad b \leftarrow_R \mathsf{Sample}_B^A(\mathsf{pub}) : \mathcal{D}_B(\mathsf{pub}, b) = 1] \end{array} \right| \\ &= \left| \mathbf{Succ}_{X,\mathcal{A}}^{\mathrm{INV}-1}(\lambda) - \Pr\left[\mathbf{Exp}_{X,\mathcal{B}}^{\mathrm{INV}-2^*}(\lambda) = 1\right] \right|,\end{aligned}$$

where $\mathbf{Exp}_{X,\mathcal{B}}^{\mathrm{INV}-2^*}$ is identical to $\mathbf{Exp}_{X,\mathcal{B}}^{\mathrm{INV}-2}$ (cf. Fig. 6) except that it returns 1 iff $(a \overset{\times}{\sim} a' \vee a = a')$. As Apply is 2:1, we have $\Pr\left[\mathbf{Exp}_{X,\mathcal{B}}^{\mathrm{INV}-2^*}(\lambda) = 1\right] = 2 \cdot \Pr\left[\mathbf{Exp}_{X,\mathcal{B}}^{\mathrm{INV}-2}(\lambda) = 1\right] = 2 \cdot \mathbf{Succ}_{X,\mathcal{B}}^{\mathrm{INV}-2}(\lambda)$. We combine these results to obtain

$$\mathbf{Dist}^{\mathsf{Sample}_B,\mathsf{Sample}_B^{\mathcal{A}}}_{X,\mathcal{D}_B}(\lambda) = \left|\mathbf{Succ}^{\mathrm{INV}-1}_{X,\mathcal{A}}(\lambda) - 2\,\mathbf{Succ}^{\mathrm{INV}-2}_{X,\mathcal{B}}(\lambda)\right|.$$

The statement of Lemma 2 follows immediately. □

Proof of Lemma 3

Construct algorithm $\mathcal{A}$ as follows: Upon receiving (pub, b), $\mathcal{A}$ runs $a' \leftarrow_R \mathsf{Sample}_A(\mathsf{pub})$ and lets $\mathcal{B}$ compute $a'' \leftarrow_R \mathcal{B}(\mathsf{pub}, a')$ such that $a' \overset{\times}{\sim} a''$. Then $\mathcal{A}$ computes $\mathsf{td}' \leftarrow \mathsf{Extract}(\mathsf{pub}, a', a'')$ and inverts challenge b via $\mathsf{Reverse}(\mathsf{td}', b, 0)$. Algorithm $\mathcal{A}$ is successful in finding a preimage for b whenever $\mathcal{B}$ is successful in finding a second preimage for a', that is, $\mathbf{Succ}^{\mathrm{INV}-1}_{X,\mathcal{A}}(\lambda) = \mathbf{Succ}^{\mathrm{INV}-2}_{X,\mathcal{B}}(\lambda)$. □

Proofs from Section 5

Proof of Theorem 1

Fix an efficient adversary $\mathcal{A}$. Without loss of generality, we assume that each of $\mathcal{A}$'s queries to the signature oracle is preceeded by a query on the same message to random oracle H_{pub}. We also assume that $\mathcal{A}$ doesn't make redundant queries (i.e. doesn't query multiple times the same message to H_{pub} or to $\mathcal{O}_{\mathsf{Sign}}$ oracle; note that signature generation in $\mathsf{2:1{-}Sig}$ is deterministic) and that $\mathcal{A}$ queries $H_{\mathsf{pub}}(\mathsf{msg}^*)$ before outputting forgery candidate $(\mathsf{msg}^*, \sigma^*)$. We finally assume that the output distribution of random oracle H_{pub} is the one induced by Sample_B algorithm; see "Appendix 2" for the construction of such a random oracle.

The proof proceeds with a sequence of games.

Game G_0 is the regular unforgeability game from Fig. 1.

Game G_1 is like G_0 except that in the specification of the $\mathcal{O}_{\mathsf{Sign}}$ oracle we replace PRF F by a random function $\varphi\colon \{0,1\}^* \to \{0,1\}$; by a standard argument we obtain

$$\left|\mathbf{Succ}^{G_0}_{\Sigma,\mathcal{A}}(\lambda) - \mathbf{Succ}^{G_1}_{\Sigma,\mathcal{A}}(\lambda)\right| \le \mathbf{Adv}^{\mathrm{prf}}_{F,\mathcal{C}}(\lambda_f),$$

for an efficient PRF distinguisher $\mathcal{C}$.

Game G_2 is like G_1 except that we change the way random oracle H_{pub} is implemented. Concretely, instead of assigning values $H_{\mathsf{pub}}(\mathsf{msg})$ using the Sample_B algorithm, we compute $a_{\mathsf{msg}} \leftarrow_R \mathsf{Sample}_A(\mathsf{pub}, d)$ for $d = \varphi(\mathsf{msg})$ and return $b = \mathsf{Apply}(\mathsf{pub}, a_{\mathsf{msg}})$. We obtain

$$\left|\mathbf{Succ}^{G_1}_{\Sigma,\mathcal{A}}(\lambda) - \mathbf{Succ}^{G_2}_{\Sigma,\mathcal{A}}(\lambda)\right| \le q_H \cdot \mathbf{Dist}^{\mathsf{Sample}_B,\mathsf{Sample}_B^{\mathcal{A}}}_{X,\mathcal{D}_B}(\lambda_2),$$

for some efficient distinguisher $\mathcal{D}_B$.

In Game G_2, this new way of processing H_{pub} queries allows us to accurately answer signature queries without requiring knowledge of td. Let $(\mathsf{msg}^*, \sigma^*)$ denote a valid forgery in game G_2. We then have $\mathsf{Apply}(\mathsf{pub}, a_{\mathsf{msg}^*}) = \mathsf{Apply}(\mathsf{pub}, \sigma^*)$ (by the validity of the forgery) and, with probability (close to) $1/2$, $\mathsf{Decide}(\mathsf{pub}, a_{\mathsf{msg}^*}) \neq \mathsf{Decide}(\mathsf{pub}, \sigma^*)$ (as bit $\varphi(\mathsf{msg}^*)$ remains hidden from $\mathcal{A}$). If this condition is met it allows the extraction of td from a_{msg^*} and σ^*. Once td is known, winning the INV−2 game is trivial: $\mathbf{Succ}^{G_2}_{\Sigma,\mathcal{A}}(\lambda) \le 2 \cdot \mathbf{Succ}^{\mathrm{INV}-2}_{X,\mathcal{B}}(\lambda_2)$ for an efficient INV−2 solver $\mathcal{B}$. □

Proofs from Section 6

Proof of Lemma 4

"⊆": Let $\{\pm y\} \in QR_n/_{\pm 1}$ be arbitrary. Without loss of generality assume $y \in QR_n$, i.e. there exists $x \in \mathbb{Z}_n^\times$ with $x^2 = y$. But then $\{\pm x\} \in \mathbb{Z}_n^\times/_{\pm 1}$ and $\{\pm x\}^2 = \{\pm(x^2)\} = \{\pm y\}$. "⊇": Fix an element $\{\pm x\} \in \mathbb{Z}_n^\times/_{\pm 1}$ and let $y \in \mathbb{Z}_n^\times$ be the (unique) value such that $y = x^2$. Then $y \in QR_n$ and $\{\pm x\}^2 = \{\pm y\} \in QR_n/_{\pm 1}$. □

Proof of Lemma 5

Let $\{\pm y\} \in QR_n/_{\pm 1}$ be arbitrary. Without loss of generality assume $y \in QR_n$. By Fact 4 there exist exactly four square roots $\{\pm x_0, \pm x_1\}$ of y in $\mathbb{Z}_n^\times$. These correspond to the two elements $\{\pm x_0\}, \{\pm x_1\} \in \mathbb{Z}_n^\times/_{\pm 1}$. Fact 4 further states that $\left(\frac{x_0}{n}\right) \neq \left(\frac{x_1}{n}\right)$, that is, one of $\{\pm x_0\}, \{\pm x_1\}$ is in $QR_n/_{\pm 1}$ and the other in $\overline{QR}_n/_{\pm 1}$, by Eq. (1). Factorization and computation of square roots immediately follow from Fact 4. □

Proof of Lemma 6

Assume towards contradiction the existence of an efficient algorithm $\mathcal{A}$ that computes square roots of elements in $QR_n/_{\pm 1}$. By picking $\{\pm x\} \in \mathbb{Z}_n^\times/_{\pm 1}$ at random and running $\{\pm x'\} \leftarrow_R \mathcal{A}(n, \{\pm x\}^2)$ we obtain a second, potentially different, square root of $\{\pm x\}^2$. By Corollary 1, with probability $1/2$ we have $\{\pm x'\} \neq \{\pm x\}$ and thus obtain the factorization of n by Lemma 5. □

Proof of Lemma 7

By Fact 6, the distribution of a is statistically close to uniform on $\mathbb{Z}_n^\times$. Mapping $a \mapsto \{\pm a\}$ is 2:1, so it preserves uniformity, i.e. the sampler for $\mathbb{Z}_n^\times/_{\pm 1}$ has the required property. For the $QR_n/_{\pm 1}$ sampler, we notice that if $\left(\frac{a}{n}\right) = +1$, then $\{\pm a\}$ is already close to uniform in $J_n/_{\pm 1} = QR_n/_{\pm 1}$. If $\left(\frac{a}{n}\right) = -1$, then $\left(\frac{ta}{n}\right) = +1$; since multiplication by t is a permutation of $\mathbb{Z}_n$, ta is close to uniformly distributed in J_n, so $\{\pm ta\}$

is close to uniformly distributed in $J_n/_{\pm 1} = QR_n/_{\pm 1}$. A similar argument holds for the $\overline{QR}_n/_{\pm 1}$ sampler. □

Proof of Theorem 2

Samplability. That $\mathbf{Dist}_X^{\mathsf{Sample}_A, U(A)}$ and $\mathbf{Dist}_X^{\mathsf{Sample}_B, U(B)}$ are negligible for all efficient distinguishers is precisely the statement of Lemma 7.

(Second) preimage resistance. By Lemma 2, it suffices to show second preimage resistance. Given an arbitrary element $\{\pm x_0\} \in \mathbb{Z}_n^\times/_{\pm 1}$, assume an efficient adversary could compute $\{\pm x_1\} \in \mathbb{Z}_n^\times/_{\pm 1}$ such that $\{\pm x_0\} \overset{\times}{\sim} \{\pm x_1\}$, i.e. such that $\{\pm x_0\} \neq \{\pm x_1\}$ and $\{\pm x_0\}^2 = \{\pm x_1\}^2$. By Lemma 5, this suffices for factoring n.

Extractability. Given are $\{\pm x_0\}, \{\pm x_1\} \in \mathbb{Z}_n^\times/_{\pm 1}$ such that $\{\pm x_0\} \overset{\times}{\sim} \{\pm x_1\}$, i.e. such that $\{\pm x_0\} \neq \{\pm x_1\}$ and $\{\pm x_0\}^2 = \{\pm x_1\}^2$. By Lemma 5, this suffices for factoring n and recovering trapdoor $\mathsf{td} = (p, q)$. □

Proof of unforgeability (Theorem 3)

We use a sequence of games; wavy lines and underlines are used to highlight changes and additions between games, respectively. Let $\mathcal{A}$ be an adversary for experiment $\mathbf{Exp}_{\mathsf{2:1-DAPS}}^{\mathrm{EUF}}$. Without loss of generality, we assume that $\mathcal{A}$ queries its $\mathcal{O}_{\mathsf{Sign}}$ oracle at most once per subject. We further assume that the distribution of random oracle H_{pub} is the one induced by Sample_B algorithm; see "Appendix 2" for the construction of such a random oracle. Let S_i be the event that game i outputs 1 when running $\mathcal{A}$.

Game 0. This is the original EUF experiment for $\mathsf{2:1-DAPS}$. For clarity, we reproduce it in full detail:

(td, pub) $\leftarrow_R$ TdGen(1^{λ_2})
(subj*, msg*, σ^*) $\leftarrow_R$ $\mathcal{A}^{\mathcal{O}_{\mathsf{Sign}}, H_{\mathsf{pub}}}$(pub)
  If $\mathcal{A}$ queries H_{pub}(subj):
    If (subj, b) $\in$ HList$_1$, return b to $\mathcal{A}$
    $b \leftarrow_R \mathsf{Sample}_B$(pub)
    Append (subj, b) to HList$_1$
    Return b to $\mathcal{A}$
  If $\mathcal{A}$ queries H_{pub}(subj, s, i):
    If (subj, s, i, b_i) $\in$ HList$_3$, return b_i to $\mathcal{A}$
    $b_i \leftarrow_R \mathsf{Sample}_B$(pub)
    Append (subj, s, i, b_i) to HList$_3$
    Return b_i to $\mathcal{A}$
  If $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{Sign}}$(subj, msg):
    Append (subj, msg) to SignedList
    $s \leftarrow$ Reverse(td, H_{pub}(subj), 0)
    $(d_1, \ldots, d_{\lambda_h}) \leftarrow H^{\#}$(subj, s, msg)
    $b_i \leftarrow H_{\mathsf{pub}}$(subj, s, i) for all $1 \le i \le \lambda_h$
    $a_i \leftarrow$ Reverse(td, b_i, d_i) for all $1 \le i \le \lambda_h$
    $\sigma \leftarrow (s, a_1, \ldots, a_{\lambda_h})$
    Return σ to $\mathcal{A}$
Return 1 iff all the following hold:
  - Ver(pub, subj*, msg*, σ^*) = 1
  - (subj*, msg*) $\notin$ SignedList
  - $\forall$ subj, msg$_0$, msg$_1$:
    (subj, msg$_0$), (subj, msg$_1$) $\in$ SignedList $\Rightarrow$ msg$_0$ = msg$_1$

By definition,

$$\Pr[S_0] = \mathbf{Succ}_{\mathsf{2:1-DAPS},\mathcal{A}}^{\mathrm{EUF}}(\lambda). \tag{3}$$

Game 1. In this game, we change the simulator so that it performs all operations without using the signing key. We also change the random oracles that currently sample from set B with Sample_B(pub) algorithm to instead use, in some occasions, the hybrid construction from Definition 14. These changes will not be detected unless one can either invert the 2:1-TDF or can distinguish the two sampling methods.

(·, pub) $\leftarrow_R$ TdGen(1^{λ_2})
(subj*, msg*, σ^*) $\leftarrow_R$ $\mathcal{A}^{\mathcal{O}_{\mathsf{Sign}}, H_{\mathsf{pub}}}$(pub)
  If $\mathcal{A}$ queries H_{pub}(subj):
    If (subj, $\underline{a}$, b) $\in$ HList$_1$, return b to $\mathcal{A}$
    $\underline{a} \leftarrow_R \mathsf{Sample}_A$(pub, 0)
    $\underline{b} \leftarrow$ Apply(pub, $\underline{a}$)
    Append (subj, $\underline{a}$, b) to HList$_1$
    Return b to $\mathcal{A}$
  If $\mathcal{A}$ queries H_{pub}(subj, s, i):
    If (subj, s, i, ·, b_i) $\in$ HList$_3$, return b_i to $\mathcal{A}$
    $b_i \leftarrow_R \mathsf{Sample}_B$(pub)
    Append (subj, s, i, ·, b_i) to HList$_3$
    Return b_i to $\mathcal{A}$
  If $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{Sign}}$(subj, msg):
    Append (subj, msg) to SignedList
    $\underline{t \leftarrow H_{\mathsf{pub}}(\mathsf{subj})}$
    Event F1: Abort if there is (subj, s', ·, ·, ·) $\in$ HList$_3$
      such that Apply(pub, s') = t.
    Retrieve (subj, s, t) from HList$_1$
    $(d_1, \ldots, d_{\lambda_h}) \leftarrow H^{\#}$(subj, s, msg)
    $a_i \leftarrow \mathsf{Sample}_A$(pub, d_i) for all $1 \le i \le \lambda_h$
    $b_i \leftarrow$ Apply(pub, a_i) for all $1 \le i \le \lambda_h$
    Append (subj, s, i, a_i, b_i) to HList$_3$ $\forall 1 \le i \le \lambda_h$
    $\sigma \leftarrow (s, a_1, \ldots, a_{\lambda_h})$
    Return σ to $\mathcal{A}$
Return 1 iff all the following hold:
  - Ver(pub, subj*, msg*, σ^*) = 1
  - (subj*, msg*) $\notin$ SignedList
  - $\forall$ subj, msg$_0$, msg$_1$:
    (subj, msg$_0$), (subj, msg$_1$) $\in$ SignedList $\Rightarrow$ msg$_0$ = msg$_1$

Analysis of distribution of values given to $\mathcal{A}$ in game 1. First, we show that the distribution of values returned to $\mathcal{A}$ in game 1 is indistinguishable from in game 0. Let us consider each of the values given to $\mathcal{A}$ in turn. Suppose abort event F1 does not occur.

Of key importance in the following is Lemma 1, which gives an upper bound on the distinguishability of values returned by Sample_B(pub) from values returned by running $a \leftarrow_R \mathsf{Sample}_A$(pub) and then returning Apply(pub, a).

– pub in line 1: This value is distributed identically to game 0.
– $H_{\mathsf{pub}}(\mathsf{subj})$ queries: These values are always consistent with other queries in this game. Any algorithm that distinguishes the values used for this query in this game from the previous game allows us to construct a distinguisher $\mathcal{D}_B$ between $\mathsf{Sample}_B^{A_0}$ and Sample_B.
– $\mathcal{O}_{\mathsf{Sign}}(\mathsf{subj}, \mathsf{msg})$ queries: These values are always consistent with $H_{\mathsf{pub}}(\mathsf{subj})$ queries. Moreover, they are also consistent with $H_{\mathsf{pub}}(\mathsf{subj}, s, i)$ queries unless the $\mathcal{O}_{\mathsf{Sign}}(\mathsf{subj}, \mathsf{msg})$ query is asked after an $H_{\mathsf{pub}}(\mathsf{subj}, s', i)$ query with $\mathsf{Apply}(\mathsf{pub}, s') = H_{\mathsf{pub}}(\mathsf{subj})$. As this case is covered by the F1 event, we disregard it for now. Any algorithm that distinguishes the values used for this query in this game from the previous game allows us to construct a distinguisher $\mathcal{D}_B$ between $\mathsf{Sample}_B^{A_0}$ and Sample_B or between $\mathsf{Sample}_B^{A_1}$ and Sample_B.

Thus,

$$|\Pr[S_0] - \Pr[S_1]| \leq (q_1 + (\lambda_h + 1)q_S)\,\mathbf{Dist}_{X,\mathcal{D}_B}^{\mathsf{Sample}_B^A, \mathsf{Sample}_B}(\lambda_2) + \Pr[\mathsf{F1}]. \tag{4}$$

Analysis of abort event F1. We claim that, if $\mathcal{A}$ makes at most q_1 queries to its $H_{\mathsf{pub}}(\cdot)$ oracle, then we can construct an efficient algorithm $\mathcal{B}_1$ against preimage resistance of 2:1-TDF X such that

$$\Pr[\mathsf{F1}] \leq q_1\,\mathbf{Succ}_{X,\mathcal{B}_1}^{\mathrm{INV}-1}(\lambda_2). \tag{5}$$

Proof of claim: Let (pub, b^*) be the INV−1 challenge. Construct $\mathcal{B}_1$ as a modification of game 1 in which $\mathcal{B}_1$ guesses a value $\hat{\jmath} \leftarrow_R [1, q_1]$ and, upon $\mathcal{A}$'s $\hat{\jmath}$th (unique) query to $H_{\mathsf{pub}}(\cdot)$, $\mathcal{B}_1$ returns the challenge value b^* to $\mathcal{A}$ instead of following the algorithm in game 1. If event F1 occurs, then with probability $1/q_1$ the value subj for which it occurs is the value of subj that was queried to the $\hat{\jmath}$th $H_{\mathsf{pub}}(\cdot)$ query. But then there is some $(\mathsf{subj}, s', \cdot, \cdot, \cdot) \in \mathrm{HList}_3$ such that $\mathsf{Apply}(\mathsf{pub}, s') = H_{\mathsf{pub}}(\mathsf{subj}) = b^*$. In other words, s' in an inverse of b^*, and hence $\mathcal{B}_1$ has successfully inverted the INV−1 challenge, winning $\mathbf{Exp}_{X,\mathcal{B}_1}^{\mathrm{INV}-1}(\lambda_2)$. Thus, $\Pr[\mathsf{F1}] \leq q_1 \Pr\left[\mathbf{Succ}_{X,\mathcal{B}_1}^{\mathrm{INV}-1}(\lambda_2) = 1\right]$.

Game 2. In this game, we place an additional condition on the simulator to output 1, namely that the signature returned by the adversary must include an s value which was previously queried to H_{pub}. However, since the s value for a subject is only known to the challenger before an $\mathcal{O}_{\mathsf{Sign}}$ query, no adversary should be able to construct a valid signature without querying $\mathcal{O}_{\mathsf{Sign}}$.

1 $(\cdot, \mathsf{pub}) \leftarrow_R \mathsf{TdGen}(1^{\lambda_2})$
2 $(\mathsf{subj}^*, \mathsf{msg}^*, \sigma^*) \leftarrow_R \mathcal{A}^{\mathcal{O}_{\mathsf{Sign}}, H_{\mathsf{pub}}}(\mathsf{pub})$
3 If $\mathcal{A}$ queries $H_{\mathsf{pub}}(\mathsf{subj})$:
4 If $(\mathsf{subj}, a, b) \in \mathrm{HList}_1$, return b to $\mathcal{A}$
5 $a \leftarrow_R \mathsf{Sample}_A(\mathsf{pub}, 0)$
6 $b \leftarrow \mathsf{Apply}(\mathsf{pub}, a)$
7 Append (subj, a, b) to HList_1
8 Return b to $\mathcal{A}$
9 If $\mathcal{A}$ queries $H_{\mathsf{pub}}(\mathsf{subj}, s, i)$:
10 If $(\mathsf{subj}, s, i, a_i, b_i) \in \mathrm{HList}_3$, return b_i to $\mathcal{A}$
11 $a_i \leftarrow_R \mathsf{Sample}_A(\mathsf{pub}, 0)$
12 $b_i \leftarrow \mathsf{Apply}(\mathsf{pub}, a_i)$
13 Append $(\mathsf{subj}, s, i, a_i, b_i)$ to HList_3
14 Return b_i to $\mathcal{A}$
15 If $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{Sign}}(\mathsf{subj}, \mathsf{msg})$:
16 Append $(\mathsf{subj}, \mathsf{msg})$ to SignedList
17 $t \leftarrow H_{\mathsf{pub}}(\mathsf{subj})$
18 Event F1: Abort if there is $(\mathsf{subj}, s', \cdot, \cdot, \cdot) \in \mathrm{HList}_3$
19 such that $\mathsf{Apply}(\mathsf{pub}, s') = t$.
20 Retrieve (subj, s, t) from HList_1
21 $(d_1, \ldots, d_{\lambda_h}) \leftarrow H^{\#}(\mathsf{subj}, s, \mathsf{msg})$
22 $a_i \leftarrow \mathsf{Sample}_A(\mathsf{pub}, d_i)$ for all $1 \leq i \leq \lambda_h$
23 $b_i \leftarrow \mathsf{Apply}(\mathsf{pub}, a_i)$ for all $1 \leq i \leq \lambda_h$
24 Append $(\mathsf{subj}, s, i, a_i, b_i)$ to HList_3 $\forall 1 \leq i \leq \lambda_h$
25 $\sigma \leftarrow (s, a_1, \ldots, a_{\lambda_h})$
26 Return σ to $\mathcal{A}$
27 Return 1 iff all the following hold:
28 - $\mathsf{Ver}(\mathsf{pub}, \mathsf{subj}^*, \mathsf{msg}^*, \sigma^*) = 1$
29 - $(\mathsf{subj}^*, \mathsf{msg}^*) \notin$ SignedList
30 - $\forall\, \mathsf{subj}, \mathsf{msg}_0, \mathsf{msg}_1$:
31 $(\mathsf{subj}, \mathsf{msg}_0), (\mathsf{subj}, \mathsf{msg}_1) \in \mathrm{SignedList} \Rightarrow \mathsf{msg}_0 = \mathsf{msg}_1$
32 - Event $\neg$F2: $\forall\, i\ \exists\, (\mathsf{subj}^*, s^*, i, a_i^*, b_i) \in \mathrm{HList}_3$:
33 $\mathsf{Apply}(\mathsf{pub}, a_i^*) = b_i$

Analysis of difference in success probabilities in game 1 and game 2. The messages that $\mathcal{A}$ sees in game 2 have exactly the same distribution as in game 1. The only difference is the additional condition ¬F2 for the experiment to output 1. Clearly, then,

$$|\Pr[S_1] - \Pr[S_2]| \leq \Pr[\mathsf{F2}]. \tag{6}$$

If event F2 occurs, then there is some i such that $\mathcal{A}$ never queried $H_{\mathsf{pub}}(\mathsf{subj}^*, s^*, i)$ but, since the signature σ^* verified, $\mathsf{Apply}(\mathsf{pub}, a_i^*) = H_{\mathsf{pub}}(\mathsf{subj}^*, s^*, i)$. In other words, the value $b_i = H_{\mathsf{pub}}(\mathsf{subj}^*, s^*, i)$ was first computed when the challenger tried to verify the signature in step 27. If b_i had been picked uniformly at random, the probability of it being guessed would be $1/|B|$. If b_i had been picked using Sample_B, the probability of it being guessed would have been at most the probability of a uniform b_i being guessed plus the probability of distinguishing the uniform distribution from Sample_B's distribution, namely: $1/|B| + \mathbf{Dist}_{X,\mathcal{D}_B^1}^{U(B), \mathsf{Sample}_B}(\lambda_2)$ for an efficient distinguisher $\mathcal{D}_B^1$. Finally, if b_i had been picked using Sample_B^A (which is indeed the case), the probability of it being guessed

would have been at most the probability of a uniform b_i being guessed plus the probability of distinguishing the uniform distribution from Sample_B's distribution plus the probability of distinguishing Sample_B's distribution from Sample_B^A's distribution, namely:

$$\Pr[\mathsf{F2}] \leq \frac{1}{|B|} + \mathbf{Dist}^{U(B),\mathsf{Sample}_B}_{X,\mathcal{D}_B^1}(\lambda_2) + \mathbf{Dist}^{\mathsf{Sample}_B,\mathsf{Sample}_B^A}_{X,\mathcal{D}_B^2}(\lambda_2).$$

Analysis of success in game 2. Claim: For every probabilistic algorithm $\mathcal{A}$ making q_S queries to $\mathcal{O}_{\mathsf{Sign}}$, there exists probabilistic algorithms $\mathcal{B}_2$ and $\mathcal{C}$ with running time linear in that of $\mathcal{A}$ such that

$$\Pr[S_2] \leq 2q_S\lambda_h \mathbf{Succ}^{\mathrm{INV}-2}_{X,\mathcal{B}_2}(\lambda_2) + \mathbf{Succ}^{\mathrm{CR}}_{H^\#,\mathcal{C}}(\lambda_h). \tag{7}$$

Proof of claim: We will construct an adversary $\mathcal{B}_2$ for $\mathbf{Exp}^{\mathrm{INV}-2}_{X,\cdot}(\lambda_2)$ using algorithm $\mathcal{A}$. Let (pub, a^*) be the challenge received by $\mathcal{B}_2$ in $\mathbf{Exp}^{\mathrm{INV}-2}_{X,\mathcal{B}_2}(\lambda_2)$.

Next, $\mathcal{B}_2$ guesses a value $\hat{\jmath} \leftarrow_R [1, q_S]$ and, upon $\mathcal{A}$'s $\hat{\jmath}$th query to $\mathcal{O}_{\mathsf{Sign}}$, $\mathcal{B}_2$ further guesses a value $\hat{\imath} \leftarrow_R [1, \lambda_h]$. If $d_{\hat{\imath}} \neq \mathsf{Decide}(\mathsf{pub}, a^*)$, then $\mathcal{B}_2$ aborts. Otherwise, it sets $a_{\hat{\imath}} \leftarrow a^*$.

Suppose game 2 outputs 1. Then $\mathcal{A}$ has output $(\mathsf{subj}^*, \mathsf{msg}^*, \sigma^*)$ which is a valid signature under pub, was not signed by $\mathcal{O}_{\mathsf{Sign}}$, and there was no double signature for any subject queried to $\mathcal{O}_{\mathsf{Sign}}$. Moreover, since neither event $\mathsf{F1}$ nor $\mathsf{F2}$ occurred, $\mathcal{A}$ must have queried $\mathcal{O}_{\mathsf{Sign}}(\mathsf{subj}^*, \mathsf{msg}')$ for some $\mathsf{msg}' \neq \mathsf{subj}^*$. With probability $1/q_S$, $\mathcal{A}$ issued this query on its $\hat{\jmath}$th to $\mathcal{O}_{\mathsf{Sign}}$. If this was not the case, then $\mathcal{B}_2$ aborts.

Now, either $H^\#(\mathsf{subj}^*, s^*, \mathsf{msg}^*) = H^\#(\mathsf{subj}^*, s^*, \mathsf{msg}')$, or not. If so, then a collision has been found in $H^\#$, then this experiment serves as an efficient algorithm $\mathcal{C}$ which finds collisions in $H^\#$. Hence, suppose no such collision occurs, namely that $H^\#(\mathsf{subj}^*, s^*, \mathsf{msg}^*) \neq H^\#(\mathsf{subj}^*, s^*, \mathsf{msg}')$. In particular, there is some bit i where $H^\#(\mathsf{subj}^*, s^*, \mathsf{msg}^*)$ and $H^\#(\mathsf{subj}^*, s^*, \mathsf{msg}')$ differ. With probability $1/\lambda_h$, $i = \hat{\imath}$. When this is the case, we have that $a_i \overset{\times}{\sim} a^*$. This is a solution to the INV-2 challenge a^*, which $\mathcal{B}_2$ outputs to win $\mathbf{Exp}^{\mathrm{INV}-2}_{X,\mathcal{B}_2}(\lambda_2)$.

By the argument above, if $\mathcal{B}_2$ correctly guesses $\hat{\jmath}$ and $\hat{\imath}$, and if $\mathsf{Decide}(\mathsf{pub}, a^*) = d_{\hat{\imath}}$, then whenever $\mathcal{A}$ wins game 2, $\mathcal{B}$ wins $\mathbf{Exp}^{\mathrm{INV}-2}_{X,\mathcal{B}_2}(\lambda_2)$.

Final result. The final result follows from combining Eqs. (4) through (7) and applying Lemma 1. □

Proof of double-signature extractability (Theorem 4)

We propose the following DSE* extractor (cf. Definition 10):

– $\mathsf{Extract}(\mathsf{pub}, (\mathsf{subj}, \mathsf{msg}_1, \sigma_1), (\mathsf{subj}, \mathsf{msg}_2, \sigma_2))$: Parse $(s_1, a_1^1, \ldots, a_{\lambda_h}^1) \leftarrow \sigma_1$ and $(s_2, a_1^2, \ldots, a_{\lambda_h}^2) \leftarrow \sigma_2$. Let $(d_1^1, \ldots, d_{\lambda_h}^1) \leftarrow H^\#(\mathsf{subj}, s_1, \mathsf{msg}_1)$ and $(d_1^2, \ldots, d_{\lambda_h}^2) \leftarrow H^\#(\mathsf{subj}, s_2, \mathsf{msg}_2)$. Let $i \in [1, \lambda_h]$ be such that $d_i^1 \neq d_i^2$. Use 2:1-TDF's $\mathsf{Extract}$ algorithm to output $\mathsf{td} \leftarrow \mathsf{Extract}(\mathsf{pub}, a_i^1, a_i^2)$.

It is straightforward to see that this is a valid extractor. Given two valid subject–message–signature tuples $(\mathsf{subj}, \mathsf{msg}_1, \sigma_1)$ and $(\mathsf{subj}, \mathsf{msg}_2, \sigma_2)$ for which $\mathsf{msg}_1 \neq \mathsf{msg}_2$, except with negligible probability, $H^\#(\mathsf{subj}, s_1, \mathsf{msg}_1) \neq H^\#(\mathsf{subj}, s_2, \mathsf{msg}_2)$. Assume no such collision occurs and the hash values are $(d_1^1, \ldots, d_{\lambda_h}^1)$ and $(d_1^2, \ldots, d_{\lambda_h}^2)$. Then there exists some position $i \in [1, \lambda_h]$ such that $d_i^1 \neq d_i^2$.

Now, since both σ_1 and σ_2 are valid, we have that $\mathsf{Apply}(\mathsf{pub}, a_i^1) = H_{\mathsf{pub}}(\mathsf{subj}, s_1, i) = H_{\mathsf{pub}}(\mathsf{subj}, s_2, i) = \mathsf{Apply}(\mathsf{pub}, a_i^2)$, but $\mathsf{Decide}(\mathsf{pub}, a_i^1) \neq \mathsf{Decide}(\mathsf{pub}, a_i^2)$. In other words, $a_i^1 \overset{\times}{\sim} a_i^2$. Thus, 2:1-TDF's $\mathsf{Extract}(\mathsf{pub}, a_i^1, a_i^2)$ returns the trapdoor td corresponding to pub. □

References

1. Ateniese, G., de Medeiros, B.: Identity-based chameleon hash and applications. In Juels, A. (ed.) FC 2004, LNCS, vol. 3110, pp. 164–180. Key West, USA, February 9–12. Springer, Heidelberg (2004)
2. Brier, E., Coron, J.-S., Icart, T., Madore, D., Randriam, H., Tibouchi, M.: Efficient indifferentiable hashing into ordinary elliptic curves. In: Cryptology ePrint Archive, Report 2009/340. http://eprint.iacr.org/2009/340 (2009)
3. Brier, E., Coron, J.-S., Icart, T., Madore, D., Randriam, H., Tibouchi, M.: Efficient indifferentiable hashing into ordinary elliptic curves. In: Rabin, T. (ed.) CRYPTO 2010, LNCS, vol. 6223, pp. 237–254, Santa Barbara, CA, USA, August 15–19. Springer, Heidelberg (2010)
4. Bernstein, D.J.: Proving tight security for Rabin–Williams signatures. In: Smart, N.P. (ed.) EUROCRYPT 2008, LNCS, vol. 4965, pp. 70–87, Istanbul, Turkey, April 13–17. Springer, Heidelberg (2008)
5. Boneh, D., Franklin, M.K.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001, LNCS, vol. 2139, pp. 213–229, Santa Barbara, CA, USA, August 19–23. Springer, Heidelberg (2001)
6. Boyar, J., Friedl, K., Lund, C.: Practical zero-knowledge proofs: giving hints and using deficiencies. J. Cryptol. **4**(3), 185–206 (1991)
7. Bari, N., Pfitzmann, B.: Collision-free accumulators and fail-stop signature schemes without trees. In Fumy, W. (ed.) EUROCRYPT'97, LNCS, vol. 1233, pp. 480–494, Konstanz, Germany, May 11–15. Springer, Heidelberg (1997)
8. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In Ashby, V. (ed.) ACM CCS 93, pp. 62–73, Fairfax, Virginia, USA, November 3–5. ACM Press (1993)
9. Bellare, M., Rogaway, P.: The exact security of digital signatures: how to sign with RSA and Rabin. In Maurer, U.M. (ed.) EUROCRYPT'96, LNCS, vol. 1070, pp. 399–416, Saragossa, Spain, May 12–16. Springer, Heidelberg (1996)

10. Bellare, M., Ristov, T.: Hash functions from sigma protocols and improvements to VSH. In: Pieprzyk, J. (ed.) ASIACRYPT 2008, LNCS, vol. 5350, pp. 125–142, Melbourne, Australia, December 7–11. Springer, Heidelberg (2008)
11. Chaum, D., Fiat, A., Naor, M.: Untraceable electronic cash. In: Goldwasser, S. (ed.) CRYPTO'88, LNCS, vol. 403, pp. 319–327, Santa Barbara, CA, USA, August 21–25. Springer, Heidelberg (1990)
12. Chor, B., Fiat, A., Naor, M.: Tracing traitors. In: Desmedt, Y. (ed.) CRYPTO'94, LNCS, vol. 839, pp. 257–270, Santa Barbara, CA, USA, August 21–25. Springer, Heidelberg (1994)
13. Camenisch, J., Lysyanskaya, A.: An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In: Pfitzmann, B. (ed.) EUROCRYPT 2001, LNCS, vol. 2045, pp. 93–118, Innsbruck, Austria, May 6–10. Springer, Heidelberg (2001)
14. Camenisch, J., Michels, M.: Proving in zero-knowledge that a number is the product of two safe primes. In: Stern, J. (ed.) EUROCRYPT'99, LNCS, vol. 1592, pp. 107–122, Prague, Czech Republic, May 2–6. Springer, Heidelberg (1999)
15. Coron, J.-S.: Optimal security proofs for PSS and other signature schemes. In: Knudsen, L.R. (ed.) EUROCRYPT 2002, LNCS, vol. 2332, pp. 272–287, Amsterdam, The Netherlands, April 28 – May 2. Springer, Heidelberg (2002)
16. De Santis, A., Di Crescenzo, G., Persiano, G.: Secret sharing and perfect zero knowledge. In: Stinson, D.R. (ed.) CRYPTO'93, LNCS, vol. 773, pp. 73–84, Santa Barbara, CA, USA, August 22–26. Springer, Heidelberg (1994)
17. Desmedt, Y.: Securing traceability of ciphertexts—towards a secure software key escrow system (extended abstract). In: Guillou, L.C., Quisquater, J.-J. (eds.) EUROCRYPT'95, LNCS, vol. 921, pp. 147–157, Saint-Malo, France, May 21–25. Springer, Heidelberg (1995)
18. Dwork, C., Lotspiech, J.B., Naor, M.: Digital signets: self-enforcing protection of digital information (preliminary version). In: 28th ACM STOC, pp. 489–498, Philadephia, Pennsylvania, USA, May 22–24. ACM Press (1996)
19. Evans, C., Palmer, C., Sleevi, R.: RFC 7469: public key pinning extension for HTTP. https://tools.ietf.org/html/rfc7469 (2015)
20. Fox-It. Black tulip: Report of the investigation into the DigiNotar certificate authority breach. http://www.rijksoverheid.nl/bestanden/documenten-en-publicaties/rapporten/2012/08/13/black-tulip-update/black-tulip-update.pdf (2012)
21. Goh, E.J., Jarecki, S., Katz, J., Wang, N.: Efficient signature schemes with tight reductions to the Diffie–Hellman problems. J. Cryptol. **20**(4), 493–514 (2007)
22. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM J. Comput. **17**(2), 281–308 (1988)
23. Google Online Security Blog. An update on attempted man-in-the-middle attacks. http://googleonlinesecurity.blogspot.de/2011/08/update-on-attempted-man-in-middle.html (2011)
24. Goyal, V.: Reducing trust in the PKG in identity based cryptosystems. In Menezes, A. (ed.) CRYPTO 2007, LNCS, vol. 4622, pp. 430–447, Santa Barbara, CA, USA, August 19–23. Springer, Heidelberg (2007)
25. Hofheinz, D., Kiltz, E.: The group of signed quadratic residues and applications. In: Halevi, S. (ed.) CRYPTO 2009, LNCS, vol. 5677, pp. 637–653, Santa Barbara, CA, USA, August 16–20. Springer, Heidelberg (2009)
26. Hoffman, P., Schlyter, J.: RFC 6698: the DNS-based authentication of named entities (DANE) transport layer security (TLS) protocol: TLSA. https://tools.ietf.org/html/rfc6698 (2012)
27. Ireland, K.,Rosen,M.: A classical introduction to modern number theory. In: Axler, S., Gehring, F.W., Ribet, K.A. (eds.) Graduate Texts in Mathematics. Springer, New York (1990)
28. Jakobsson, M., Juels, A., Nguyen, P.Q.: Proprietary certificates. In: Preneel, B. (ed.) CT-RSA 2002, LNCS, vol. 2271, pp. 164–181, San Jose, CA, USA, February 18–22. Springer, Heidelberg (2002)
29. Katz, J., Lindell, Y.: Introduction to Modern Cryptography. Chapman and Hall/CRC Press, Boca Raton (2007)
30. Krawczyk, H., Rabin, T.: Chameleon signatures. In: NDSS 2000, San Diego, California, USA, February 2–4. The Internet Society (2000)
31. Kiayias, A., Tang, Q.: How to keep a secret: leakage deterring public-key cryptosystems. In: Sadeghi, A.R., Gligor, V.D., Yung, M. (ed.) ACM CCS 13, pp. 943–954, Berlin, Germany, November 4–8. ACM Press (2013)
32. Lamport, L.: Constructing digital signatures from a one way function. In: Technical Report CSL-98, SRI International (1979)
33. Merkle, R.C.: A certified digital signature. In: Brassard, G. (ed.) CRYPTO'89, LNCS, vol. 435, pp. 218–238, Santa Barbara, CA, USA, August 20–24. Springer, Heidelberg (1990)
34. Mashatan, A., Ouafi, K.: Forgery-resilience for digital signature schemes. In Youm, H.Y., Won, Y. (ed.) ASIACCS 12, pp. 24–25, Seoul, Korea, May 2–4. ACM Press (2012)
35. Marlinspike, M., Perrin, T.: Trust assertions for certificate keys (Internet-draft). http://tools.ietf.org/html/draft-perrin-tls-tack-02 (2013)
36. Menezes, A., van Oorschot, P., Vanstone, S.: Handbook of Applied Cryptography. CRC Press, Boca Raton (2001)
37. NIST—National Institute of Standards and Technology. Special publication 800-90. In: Recommendation for Random Number Generation Using Deterministic Random Bit Generators. http://csrc.nist.gov/publications/nistpubs/800-90A/SP800-90A.pdf (2012)
38. Pedersen, T.P., Pfitzmann, B.: Fail-stop signatures. SIAM J. Comput. **26**(2), 291–330 (1997)
39. Shoup, V.: A Computational Introduction to Number Theory and Algebra. Cambridge University Press, New York (2005)
40. Soghoian, C., Stamm, S.: Certified lies: detecting and defeating government interception attacks against SSL (short paper). In: Danezis, G. (ed.) FC 2011, LNCS, vol. 7035, pp. 250–259, Gros Islet, St. Lucia, February 28–March 4. Springer, Heidelberg (2012)
41. Shamir, A., Tauman, Y.: Improved online/offline signature schemes. In: Kilian, J. (ed.) CRYPTO 2001, LNCS, vol. 2139, pp. 355–367, Santa Barbara, CA, USA, August 19–23. Springer, Heidelberg (2001)
42. van de Graaf, J., Peralta, R.: A simple and secure way to show the validity of your public key. In: Pomerance, C. (ed.) CRYPTO'87, LNCS, vol. 293, pp. 128–134, Santa Barbara, CA, USA, August 16–20. Springer, Heidelberg (1988)
43. van Heyst, E., Pedersen, T.P.: How to make efficient fail-stop signatures. In: Rueppel, R.A. (ed.) EUROCRYPT'92, LNCS, vol. 658, pp. 366–377, Balatonfüred, Hungary, May 24–28. Springer, Heidelberg (1993)
44. 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, LNCS, vol. 740, pp. 15–30, Santa Barbara, CA, USA, August 16–20. Springer, Heidelberg (1993)
45. Waidner, M., Pfitzmann, B.: The dining cryptographers in the disco—underconditional sender and recipient untraceability with computationally secure serviceability (abstract) (rump session). In: Quisquater, J.J., Vandewalle, J. (eds.) EUROCRYPT'89, LNCS, vol. 434, pp. 690, Houthalen, Belgium, April 10–13. Springer, Heidelberg (1990)