Double-authentication-preventing signatures

Bertram Poettering[1] and Douglas Stebila[2]

[1] *Information Security Group, Royal Holloway, University of London, Egham, Surrey, United Kingdom*

[2] *School of Electrical Engineering and Computer Science and School of Mathematical Sciences, Queensland University of Technology, Brisbane, Australia*

bertram.poettering@rhul.ac.uk stebila@qut.edu.au

May 30, 2013

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 malicious or coerced authority can make multiple certifications for the same subject but different objects. 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 prevents, or at least strongly *discourages*, signers misbehaving. We give a generic construction using a new type of trapdoor functions with extractability properties, which we show can be instantiated using the group of sign-agnostic quadratic residues modulo a Blum integer.

Keywords: digital signatures, double signatures, forgeability, extractability, dishonest signer, two-to-one trapdoor functions

1 Introduction

Digital signatures are used in several cryptographic 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. For example, in the public key infrastructure used in the world wide web, CAs have been known to sign multiple certificates per domain name, as a result of either (a) poor management practices, (b) a security breach, (c) malicious behavior on the part of the CA, or (d) coercion (legal or otherwise) by law enforcement agencies, governments, or other parties who "make him an offer he can't refuse"; similar concerns apply to time-stamping authorities. If it comes to light that a CA has improperly signed two or more certificates for the same domain name, various responses exist, including bad publicity, the CA revoking the extra certificates, or ultimately removal of the CA from web browsers' lists of trusted CAs. A CA acting maliciously may bet that the embarrassment from one or two lapses in judgment may not be enough to lead to a downfall.

Contributions. We propose a new type of digital signature scheme for which the response to improper signer behavior is less ambiguous. In a *double-authentication-preventing signature (DAPS)*, the data to be signed is split into two parts: a *subject* and a *message*. If a signer ever signs two messages for the same subject, then enough information is revealed for anyone to be able to forge signatures on arbitrary

messages, rendering the signer immediately and irrevocably untrustworthy. 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 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 and for a standard security level has reasonably sized signatures of 48kB.

Our quadratic residue-based construction provides double-signature extractability in what we call the *trusted setup 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 potential coercive requests by third parties, such as government agencies, to construct fraudulent certificates only after completed setup. Our construction can be translated to the *untrusted setup 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.

1.1 Outline

We define a double-authentication-preventing signature in Section 2 and its unforgeability as well as double-signature forgeability and double-signature extractability properties. We introduce in Section 3 extractable 2:1 trapdoor functions and provide a factoring-based instantiation in Section 4 using sign-agnostic quadratic residues. In Section 5 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 setup model, as well as discuss its use with untrusted setup. Section 6 examines applications of DAPS to certification and time-stamping authorities. We conclude in Section 7. The appendices contain a review of basic results from number theory (Appendix A), proofs of results from the main body (Appendix B), and a discussion on the relationship of extractable 2:1 trapdoor functions and claw-free permutations (Appendix C).

1.2 Related work

One-time signatures. One-time signatures, first proposed by Lamport using a construction based on hash functions [Lam79], allow at most one message to be signed. Many instances can be combined using Merkle trees [Mer90] to allow multiple signatures with just a single verification key, but key generation time becomes a function of the total number of signatures allowed. There is a vast literature on one-time signature schemes, with various lines of research focusing on shorter signatures, more efficient generation, smaller key sizes, and minimization of the assumptions required.

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 priori, 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 the subject serves as an index to the leaf: each subject identifies a path through the tree and hence which keys must be used to sign the message. However, this limits the size of the subject space and requires a key generation time at least linear in the size of the subject space. 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 [WP90, vP92, vPP93, BP97, PP97] 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. Recent work on fail-stop signatures focuses on shorter signatures and more efficient constructions, especially related to the use of accumulators. Forgery-resilient signatures [MO12] 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 signer* signs two messages for the same subject.

Digital cash. Although we do not see a direct functional connection between digital cash schemes (e.g., [CFN90]) and DAPS, it is interesting to note that the number-theoretic structures our scheme builds on are similar to those used in digital cash to provide double spending traceability: both schemes use RSA moduli that can be factored if signers/spenders misbehave.

2 Definitions

In this section we present the central definitions of the paper: a *double-authentication-preventing signature* and its 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.

Notation. If S is a finite set, let $U(S)$ denote the uniform distribution on S and $x \leftarrow_R S$ denote sampling x uniformly from S. If A and B are two probability distributions, then notation $A \approx B$ denotes that the statistical distance between A and B is negligible. If $\mathcal{A}$ is a (probabilistic) algorithm, then $x \leftarrow_R \mathcal{A}^{\mathcal{O}}(y)$ denotes running $\mathcal{A}$ with input y on uniformly random coins with oracle access to $\mathcal{O}$, and setting x to be the output. We use 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 (Double-authentication-preventing signature scheme)**.** A *double-authentication-preventing signature (DAPS)* is a tuple of efficient algorithms $\mathsf{DAPS} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Ver})$ 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 2 (Correctness of DAPS)**.** A double-authentication-preventing signature DAPS is *correct* if, for all $\lambda \in \mathbb{N}$, for all key pairs $(\mathsf{sk}, \mathsf{vk}) \leftarrow_R \mathsf{KGen}(1^\lambda)$, for all $\mathsf{subj}, \mathsf{msg} \in \{0,1\}^*$, and for all signatures $\sigma \leftarrow_R \mathsf{Sign}(\mathsf{sk}, \mathsf{subj}, \mathsf{msg})$, we have that $\mathsf{Ver}(\mathsf{vk}, \mathsf{subj}, \mathsf{msg}, \sigma) = 1$.

2.1 Unforgeability

Our unforgeability notion largely coincides with the standard unforgeability notion for digital signature schemes [GMR88]; 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 3 (Existential unforgeability)**.** A double-authentication-preventing signature DAPS is *existentially unforgeable under adaptive chosen message attacks* if, for all efficient adversaries $\mathcal{A}$, the success probability $\mathbf{Succ}^{\mathrm{EUF}}_{\mathsf{DAPS},\mathcal{A}}(\lambda) := \Pr\left[\mathbf{Exp}^{\mathrm{EUF}}_{\mathsf{DAPS},\mathcal{A}}(\lambda) = 1\right]$ in the EUF experiment of Figure 1 is a negligible function in λ.

2.2 Double-signature forgeability

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

Figure 1: Experiment for existential unforgeability of DAPS

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

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:
   - (S_1,S_2) is compromising
   - $\mathsf{Ver}(\mathsf{vk},\mathsf{subj}^*,\mathsf{msg}^*,\sigma^*) \neq 1$

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

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:
   - (S_1,S_2) is compromising
   - $\mathsf{Ver}(\mathsf{vk},\mathsf{subj}^*,\mathsf{msg}^*,\sigma^*) \neq 1$

Figure 2: Experiments for double-signature forgeability

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

We now define the double-signature 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 setup model*, where key generation is assumed to proceed honestly, and the *untrusted setup model*, where the adversary has full control over key generation as well.

Definition 5 (Double-signature forgeability)**.** A double-authentication-preventing signature DAPS is *double-signature forgeable* (resp. *double-signature forgeable with trusted setup*) 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 $\mathbf{Succ}^{\mathrm{DSF}^{(*)}}_{\mathsf{DAPS},\mathcal{A}}(\lambda) := \Pr\left[\mathbf{Exp}^{\mathrm{DSF}^{(*)}}_{\mathsf{DAPS},\mathcal{A}}(\lambda) = 1\right]$ of success in the DSF (resp. DSF*) experiment of Figure 2 is a negligible function in λ.

2.3 Double-signature extractability

While the notion of double-signature forgeability expresses the desired functionality of the scheme from a theoretical 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 6 (Double-signature extractability)**.** A double-authentication-preventing signature DAPS is *double-signature extractable* (resp. *double-signature extractable with trusted setup*) if an efficient algorithm

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

is known such that, for all efficient adversaries $\mathcal{A}$, the probability $\mathbf{Succ}^{\mathrm{DSE}^{(*)}}_{\mathsf{DAPS},\mathcal{A}}(\lambda) := \Pr\left[\mathbf{Exp}^{\mathrm{DSE}^{(*)}}_{\mathsf{DAPS},\mathcal{A}}(\lambda) = 1\right]$ of success in the DSE (resp. DSE*) experiment of Figure 3 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:
   - (S_1, S_2) is compromising
   - sk' is not the signing key corresponding to vk

$\mathbf{Exp}^{\mathrm{DSE}^*}_{\mathsf{DAPS},\mathcal{A}}(\lambda)$:
1. $(\mathsf{sk}, \mathsf{vk}) \leftarrow_R \mathsf{KGen}(1^\lambda)$
2. $(S_1, S_2) \leftarrow_R \mathcal{A}(\mathsf{sk}, \mathsf{vk})$
3. $\mathsf{sk}' \leftarrow_R \mathsf{Extract}(\mathsf{vk}, (S_1, S_2))$
4. Return 1 iff all the following hold:
   - (S_1, S_2) is compromising
   - $\mathsf{sk}' \neq \mathsf{sk}$

Figure 3: Experiments for double-signature extractability

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 Section 5, when instantiated with the factoring-based primitive of Section 4, it is easy to check that a signing key (p, q) corresponds to a verification key n; note that there is a canonical representation of such signing keys (take $p < q$).

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.

3 2:1 trapdoor functions and extractability

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

Consider two finite sets, A and B, such that A 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.

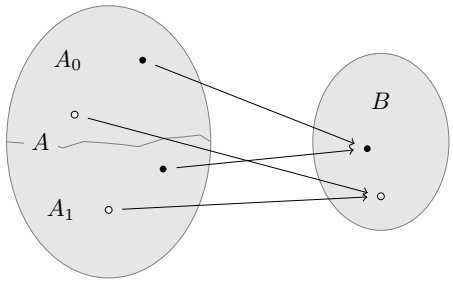

Figure 4: Illustration of a 2:1 trapdoor function $f : A \to B$. Each element of B has exactly two preimages, one in A_0 and one in A_1.

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 Figure 4. We will formalize the functionality and security properties below.

2:1-TDFs resemble to a certain extent claw-free permutations (CFPs), established by Goldwasser, Micali, and Rivest [GMR88]. We explore the connection between these primitives in Appendix C, highlighting two major differences: (a) 2:1-TDFs seem to generally rely on weaker hardness assumptions (while we show that CFPs readily imply 2:1-TDFs, we give some indication on why the converse might not hold), and (b) CFPs are generally not equipped with extraction capabilities, in contrast to extractable 2:1-TDFs.

3.1 Definition

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

Definition 7 (2:1 trapdoor function)**.** A *2:1 trapdoor function (2:1-TDF)* is a tuple of efficient algorithms $(\mathsf{TdGen}, \mathsf{Sample}_A, \mathsf{Sample}_B, \mathsf{Apply}, \mathsf{Reverse}, \mathsf{Decide})$ as follows:

- $\mathsf{TdGen}(1^\lambda)$: On input security parameter 1^λ, this randomized algorithm outputs a pair $(\mathsf{td}, \mathsf{pub})$, where td is a trapdoor and pub is some associated public information. Each possible outcome pub implicitly defines finite sets $A = A(\mathsf{pub})$ and $B = B(\mathsf{pub})$.
- $\mathsf{Sample}_A(\mathsf{pub}, d; r)$: 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, by $\mathsf{Sample}_A(\mathsf{pub}, d)$ (respectively, $\mathsf{Sample}_A(\mathsf{pub})$) we denote the result obtained by uniformly sampling $r \leftarrow_R \{0,1\}^\lambda$ (resp., $d \leftarrow_R \{0,1\}$ and $r \leftarrow_R \{0,1\}^\lambda$) and executing $\mathsf{Sample}_A(\mathsf{pub}, d; r)$.
- $\mathsf{Sample}_B(\mathsf{pub}; r)$: 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 8 (Correctness of 2:1-TDF)**.** A 2:1-TDF is *correct* if, for all $(\mathsf{td}, \mathsf{pub}) \leftarrow_R \mathsf{TdGen}$, all $d \in \{0,1\}$, all $a \in A(\mathsf{pub})$, and all $b \in B(\mathsf{pub})$, we have that (1) $a \in \mathsf{Reverse}(\mathsf{td}, \mathsf{Apply}(\mathsf{pub}, a), \{0,1\})$, (2) $\mathsf{Apply}(\mathsf{pub}, \mathsf{Reverse}(\mathsf{td}, b, d)) = b$, and (3) $\mathsf{Decide}(\mathsf{pub}, \mathsf{Reverse}(\mathsf{td}, b, d)) = d$. We further require that $\mathsf{Decide}(\mathsf{pub}, \mathsf{Sample}_A(\mathsf{pub}, d; r)) = d$ hold 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 $\overset{x}{\sim} \subseteq A(\mathsf{pub}) \times A(\mathsf{pub})$ such that

$$a \overset{x}{\sim} a' \quad \Longleftrightarrow \quad \mathsf{Apply}(\mathsf{pub}, a) = \mathsf{Apply}(\mathsf{pub}, a') \wedge \mathsf{Decide}(\mathsf{pub}, a) \neq \mathsf{Decide}(\mathsf{pub}, a') \ .$$

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

3.2 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 Appendix B.1.

3.2.1 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 9 (Sampling distance)**.** Let X be a 2:1-TDF and let S_0, S_1 be two (sampling) algorithms. We define the *sampling distance* of S_0, S_1 with respect to a distinguisher $\mathcal{D}$ as

$$\mathbf{Dist}_{X,\mathcal{D}}^{S_0,S_1}(\lambda) := \left| \begin{array}{l} \Pr\left[(\mathsf{td}, \mathsf{pub}) \leftarrow_R \mathsf{TdGen}(1^\lambda); x \leftarrow_R S_0(\mathsf{pub}) : \mathcal{D}(\mathsf{pub}, x) = 1\right] \\ \quad - \Pr\left[(\mathsf{td}, \mathsf{pub}) \leftarrow_R \mathsf{TdGen}(1^\lambda); x \leftarrow_R S_1(\mathsf{pub}) : \mathcal{D}(\mathsf{pub}, x) = 1\right] \end{array} \right| \ .$$

We consider two different strategies to obtain samples from set B: using the Sample_B algorithm directly, or using Sample_A and mapping obtained samples from set A to set B using the Apply algorithm. The latter hybrid construction is formalized in Definition 10. We show in Lemma 1 that it yields reasonable results, assuming good Sample_A and Sample_B algorithms.

$\mathbf{Exp}^{\text{INV-1}}_{X,\mathcal{A}}(\lambda)$:
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 $\mathsf{Apply}(\mathsf{pub}, a) = b$

$\mathbf{Exp}^{\text{INV-2}}_{X,\mathcal{B}}(\lambda)$:
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 $a \overset{\times}{\sim} a'$

Figure 5: Experiments for (second) preimage resistance of 2:1-TDFs

Definition 10 (Hybrid sampling)**.** For a 2:1-TDF, let $(\mathsf{td}, \mathsf{pub})$ be output by TdGen. Then sampling algorithm Sample^A_B for set $B(\mathsf{pub})$ is defined as $\mathsf{Sample}^A_B(\mathsf{pub}) := \mathsf{Apply}(\mathsf{pub}, \mathsf{Sample}_A(\mathsf{pub}))$.

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 $\mathcal{D}'_A$ and $\mathcal{D}'_B$ such that*

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

It follows from Lemma 1 that $\mathbf{Dist}^{\mathsf{Sample}_B, \mathsf{Sample}^A_B}_X$ is small if $\mathbf{Dist}^{\mathsf{Sample}_A, U(A)}_X$ and $\mathbf{Dist}^{\mathsf{Sample}_B, U(B)}_X$ are. This observation motivates the following security requirement on 2:1-TDFs.

Definition 11 (Samplability of 2:1-TDF)**.** A 2:1-TDF X is *samplable* if, for all efficient distinguishers $\mathcal{D}_A$ and $\mathcal{D}_B$, $\mathbf{Dist}^{\mathsf{Sample}_A, U(A)}_{X,\mathcal{D}_A}(\lambda)$ and $\mathbf{Dist}^{\mathsf{Sample}_B, U(B)}_{X,\mathcal{D}_B}(\lambda)$ are negligible functions in λ.

Observe that if $\mathbf{Dist}^{\mathsf{Sample}_A, U(A)}_X$ is negligible then so are $\mathbf{Dist}^{\mathsf{Sample}_A(\cdot,0), U(A_0)}_X$ and $\mathbf{Dist}^{\mathsf{Sample}_A(\cdot,1), U(A_1)}_X$.

3.2.2 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 12 (Preimage resistance of 2:1-TDF)**.** A 2:1-TDF X is *preimage resistant* and *second preimage resistant* if $\mathbf{Succ}^{\text{INV-1}}_{X,\mathcal{A}}(\lambda) := \Pr\left[\mathbf{Exp}^{\text{INV-1}}_{X,\mathcal{A}}(\lambda) = 1\right]$ and $\mathbf{Succ}^{\text{INV-2}}_{X,\mathcal{B}}(\lambda) := \Pr\left[\mathbf{Exp}^{\text{INV-2}}_{X,\mathcal{B}}(\lambda) = 1\right]$ are negligible functions in λ, for all efficient adversaries $\mathcal{A}, \mathcal{B}$, where $\mathbf{Exp}^{\text{INV-1}}_{X,\mathcal{A}}$ and $\mathbf{Exp}^{\text{INV-2}}_{X,\mathcal{B}}$ are as in Figure 5.

The following simple lemma shows that second preimage resistance implies preimage resistance. We will see in Section 3.3 that these notions are actually equivalent for an extractable variant of 2:1-TDF.

Lemma 2 (INV-2 $\Rightarrow$ INV-1 for samplable 2:1-TDF)**.** *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* $\mathbf{Succ}^{\text{INV-1}}_{X,\mathcal{A}}(\lambda) \leq 2 \cdot \mathbf{Succ}^{\text{INV-2}}_{X,\mathcal{B}}(\lambda) + \mathbf{Dist}^{\mathsf{Sample}_B, \mathsf{Sample}^A_B}_{X,\mathcal{D}_B}(\lambda)$.

3.3 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 13 (Extractable 2:1-TDF)**.** A 2:1-TDF is *extractable* if an efficient algorithm

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

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

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) extractable 2:1-TDFs. The proof of Lemma 3 appears in Appendix B.1.

Lemma 3 (INV-1 $\Rightarrow$ INV-2 for extractable 2:1-TDF)**.** *Let X be an extractable 2:1-TDF and let $\mathcal{B}$ be an efficient algorithm for the* INV-2 *experiment. Then there exists an efficient algorithm $\mathcal{A}$ for the* INV-1 *experiment such that* $\mathbf{Succ}^{\text{INV-2}}_{X,\mathcal{B}}(\lambda) = \mathbf{Succ}^{\text{INV-1}}_{X,\mathcal{A}}(\lambda)$.

4 Constructing extractable 2:1 trapdoor functions

Having introduced 2:1-TDFs and extractable 2:1-TDFs, we now show how to construct these primitives: we propose an efficient extractable 2:1-TDF and prove it secure, assuming hardness of the integer factorization problem. A second construction, based on generic claw-free permutations, is treated in Appendix C.

Our construction builds on a specific structure from number theory, the *group of sign-agnostic quadratic residues*. This group was introduced to cryptography by Goldwasser, Micali, and Rivest in [GMR88], and rediscovered 20 years later by Hofheinz and Kiltz [HK09]. We first reproduce the results of [GMR88,HK09] and then extend them towards our requirements.[1]

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 Figure 6 (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 A.

4.1 Sign-agnostic quadratic residues

For an RSA modulus n, it is widely believed that efficiently distinguishing elements in QR_n from elements in $\overline{QR}_n$ is a hard problem. It also seems to be infeasible to sample elements from QR_n without knowing a square root of the samples, or to construct hash functions that map to QR_n and could be modeled as random oracles. However, such properties are a prerequisite in certain applications in cryptography [HK09], 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 becomes feasible.

The use of sign-agnostic quadratic residues in cryptography is explicitly proposed in [GMR88,HK09]. 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 4–6 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, $H/_{\{1,T\}}$ is a group, $\psi : H \to H/_{\{1,T\}} : x \mapsto \{x,Tx\}$ is a group homomorphism, and $|\psi(H)| = |H/_{\{1,T\}}| = |H|/2$. Further, for all subgroups $G \leq H$ we have that $\psi(G) \leq \psi(H) = H/_{\{1,T\}}$. In such cases, if G is such that $T \in G$, then $|\psi(G)| = |G/_{\{1,T\}}| = |G|/2$ as above; otherwise, if $T \notin G$, then $|\psi(G)| = |G|$ and thus $\psi(G) \cong G$.

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

Note that we just observed $QR_n/_{\pm 1} \leq J_n/_{\pm 1} \leq \mathbb{Z}_n^\times/_{\pm 1}$ and $|QR_n/_{\pm 1}| = \varphi(n)/4 = |J_n/_{\pm 1}|$. The overall structure is hence $QR_n/_{\pm 1} = J_n/_{\pm 1} \lneq \mathbb{Z}_n^\times/_{\pm 1}$, as illustrated in Figure 6 (right). After agreeing on notations $\{\pm x\} = \{x,-x\}$ and $\{\pm x\}^2 = \{\pm(x^2)\}$, the result of Lemma 7 (in Appendix B.2) is immediate:

$$QR_n/_{\pm 1} = \left\{ \{\pm x\}^2 : \{\pm x\} \in \mathbb{Z}_n^\times/_{\pm 1} \right\} \quad .$$

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

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

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 4 and 5, which directly correspond with Facts 4 and 5 from Appendix A. Similarly, Corollaries 1 and 2 correspond. We stress that the following results do not appear in [GMR88, HK09]; the corresponding proofs appear in Appendix B.2.

[1] 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*.

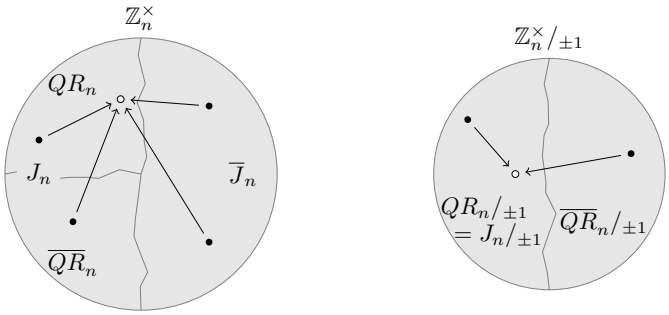

Figure 6: Illustration of $\mathbb{Z}_n^\times$ and $\mathbb{Z}_n^\times/_{\pm 1}$ (for Blum integers n), and subgroups QR_n, J_n and $J_n/_{\pm 1} = QR_n/_{\pm 1}$. Also visualized is the action of the squaring operation (see Corollaries 1 and 2).

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

Corollary 1 (Squaring in $\mathbb{Z}_n^\times/_{\pm 1}$, $QR_n/_{\pm 1}$, and $\overline{QR}_n/_{\pm 1}$)**.** *Let n be a Blum integer. The squaring operation $\mathbb{Z}_n^\times/_{\pm 1} \to QR_n/_{\pm 1} : \{\pm x\} \mapsto \{\pm x\}^2$ is a 2:1 mapping. Moreover, squaring is a 1:1 function from $QR_n/_{\pm 1}$ to $QR_n/_{\pm 1}$ and from $\overline{QR}_n/_{\pm 1}$ to $QR_n/_{\pm 1}$. These relations are illustrated in Figure 6 (right).*

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

Lemma 6 (Samplability and decidability of $\mathbb{Z}_n^\times/_{\pm 1}$, $QR_n/_{\pm 1}$, and $\overline{QR}_n/_{\pm 1}$)**.** *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 equation (1).*

Observe that, assuming the notation from Fact 6 in Appendix A, the described sampling method for $QR_n/_{\pm 1}$ can be seen as the composition $G \circ F$, where G is defined as

$$G : J_n \to QR_n/_{\pm 1} : y \mapsto \{\pm y\} \ .$$

Going a step further, a sampler $S : \{0,1\}^\ell \to QR_n/_{\pm 1}$ that takes (uniform) bitstrings as input and outputs (statistically close to uniform) elements in $QR_n/_{\pm 1}$ is given by $S = G \circ F \circ E$.

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 [GMR88]. The corresponding decoding procedure is $\overline{x} \mapsto \{\overline{x}, -\overline{x}\}$.

4.2 Indifferentiable hashing onto $QR_n/_{\pm 1}$

Specific applications of the group of sign-agnostic quadratic residues modulo a Blum integer n might rely on the existence of a hash function $H : \{0,1\}^* \to QR_n/_{\pm 1}$. Moreover, the corresponding security arguments might require modeling H as a random oracle. We show in the following how to construct such hash functions onto $QR_n/_{\pm 1}$.

Let $\ell \gg \log n$ be an integer and assume $h : \{0,1\}^* \to \{0,1\}^\ell$ is a hash function that may be modeled 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

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

are the functions specified by Lemma 6 and Fact 6 (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 [BF01] and Brier *et al.* [BCI+10, BCI+09]. Specifically, Brier *et al.* show that if function $G \circ F \circ E$ is an *admissible encoding* and h is a random oracle, then $H = G \circ F \circ E \circ h$ is indifferentiable from a random oracle [BCI+09, §3]. To program H, we can take a preimage of $G \circ F \circ E$ and program h accordingly. We reproduce the definition and main theorem from [BCI+09] as follows.

Definition 14 (Admissible encoding [BCI+09])**.** A function $F : S \to R$ between finite sets is an *admissible encoding* if it satisfies the following properties:

(1) Computable: F is computable in deterministic polynomial time.
(2) Regular: for s uniformly distributed in S, the distribution of $F(s)$ 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 $F^{-1}(r)$.

Theorem 1 (Construction of random oracle [BCI+09])**.** *Let $F : S \to R$ be an admissible encoding. If $h : \{0,1\}^* \to S$ is a random oracle, then the construction $H(m) = F(h(m))$ 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

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

(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 $G \circ F \circ E$. Hence $H = G \circ F \circ E \circ h : \{0,1\}^* \to QR_n/_{\pm 1}$ behaves like a random oracle by Theorem 1.

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

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

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

- $\mathsf{TdGen}(1^\lambda)$: Pick random Blum integer $n = pq$ of length λ such that $p < q$. Pick $t \in \mathbb{Z}_n^\times$ with $\left(\frac{t}{n}\right) = -1$. Return $\mathsf{pub} \leftarrow (n, t)$ and $\mathsf{td} \leftarrow (p, q)$.
  We will use sets $A_0(\mathsf{pub}) := QR_n/_{\pm 1}$, $A_1(\mathsf{pub}) := \overline{QR}_n/_{\pm 1}$, $A(\mathsf{pub}) := \mathbb{Z}_n^\times/_{\pm 1}$, and $B(\mathsf{pub}) := QR_n/_{\pm 1}$.
- $\mathsf{Sample}_A(\mathsf{pub}, d)$: 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 6.
- $\mathsf{Sample}_B(\mathsf{pub})$: Implement $\mathsf{Sample}_B(\mathsf{pub})$ using the sampler for set $QR_n/_{\pm 1}$ from Lemma 6.
- $\mathsf{Apply}(\mathsf{pub}, \{\pm a\})$: Return $\{\pm b\} \leftarrow \{\pm a\}^2$.
- $\mathsf{Reverse}(\mathsf{td}, \{\pm b\}, d)$: By Lemma 4, element $\{\pm b\} \in QR_n/_{\pm 1}$ has exactly two square roots: $\{\pm a_0\} \in QR_n/_{\pm 1}$ and $\{\pm a_1\} \in \overline{QR}_n/_{\pm 1}$. Return $\{\pm a_d\}$.
- $\mathsf{Decide}(\mathsf{pub}, \{\pm a\})$: Return 0 if $\{\pm a\} \in QR_n/_{\pm 1}$; otherwise return 1.
- $\mathsf{Extract}(\mathsf{pub}, \{\pm a_0\}, \{\pm a_1\})$: Both $\gcd(n, a_0 - a_1)$ and $\gcd(n, a_0 + a_1)$ are non-trivial factors of $n = pq$. Return $\mathsf{td}^* \leftarrow (p, q)$ such that $p < q$.

These algorithms are all efficient. Correctness of Blum-2:1-TDF and the various security properties follow straightforwardly from the number-theoretic facts established in Sections 4.1. The proof appears in Appendix B.2.

Theorem 2 (Security and extractability of Blum-2:1-TDF)**.** Blum-2:1-TDF *is samplable (Def. 11), (second) preimage resistant (Def. 12) under the assumption that factoring is hard, and extractable (Def. 13).*

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

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

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

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

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

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

Figure 7: Double-authentication-preventing signature scheme 2:1-DAPS

5 DAPS construction based on extractable 2:1-TDF

We now come to the central result of this paper, a 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 2 (DAPS from extractable 2:1-TDF)**.** Let λ be a security parameter, and let λ_2 and λ_h be security parameters derived from 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 2:1-DAPS consists of the algorithms specified in Figure 7.

The basic idea of the signing algorithm is as follows. From any given subject, the signer derives message-independent signing elements $b_1, \ldots, b_{\lambda_h}$. The signer also hashes subject and message to a bit string $d_1 \ldots d_{\lambda_h}$; for each bit d_i, she finds the preimage a_i of the signing element b_i which is in the d_i partition of A; either in A_0 or A_1. The signature σ is basically the vector of these preimages. Intuitively, the scheme is unforgeable because it is hard to find preimages of signing elements b_i without knowing the trapdoor. 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, by 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.

Remark 3 (Rationale on subj-dependent value s). 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 [BR96], 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 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 $a' \in A$, $a \not\sim a'$, 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.

5.1 Unforgeability of 2:1-DAPS

We next establish existential unforgeability of 2:1-DAPS. 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 Appendix B.3.

Theorem 3 (2:1-DAPS is EUF)**.** *In the setting of Construction 2, 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-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 $H_{\mathsf{pub}}(\cdot)$, q_2 queries to $H_{\mathsf{pub}}(\cdot,\cdot,\cdot)$, 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*

$$\begin{aligned}\mathbf{Succ}^{\text{EUF}}_{\text{2:1-DAPS},\mathcal{A}}(\lambda) \le{}& (q_1+q_2+(\lambda_h+1)q_S+1)\,\mathbf{Dist}^{\mathsf{Sample}_A,U(A)}_{X,\mathcal{D}_A}(\lambda_2)\\ &+(q_1+q_2+(\lambda_h+1)q_S)\,\mathbf{Dist}^{\mathsf{Sample}_B,U(B)}_{X,\mathcal{D}_B}(\lambda_2)\\ &+q_1\mathbf{Succ}^{\text{INV-1}}_{X,\mathcal{B}_1}(\lambda_2)+2q_S\lambda_h\,\mathbf{Succ}^{\text{INV-2}}_{X,\mathcal{B}_2}(\lambda_2)+\mathbf{Succ}^{\text{CR}}_{H^{\#},\mathcal{C}}(\lambda_h)\ ,\end{aligned}$$

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

Remark 4 (2:1-DAPS is deterministic and S-EUF). Note that 2:1-DAPS is not only deterministic, but also strongly unforgeable: it is hard for the adversary to return a new signature on a subject/message pair for which it already has a signature. Further on, the scheme can be made unique [Cor02] by adding requirement $\mathsf{Decide}(\mathsf{pub}, s) = 0$ to the verification algorithm.

5.2 Double-signature extractability of 2:1-DAPS

Assuming collision resistance of $H^{\#}$, two signatures for different messages but the same subject result in some index j where the hashes $H^{\#}(\mathsf{subj}, s, \mathsf{msg}_1)$ and $H^{\#}(\mathsf{subj}, s, \mathsf{msg}_2)$ differ. The corresponding jth values a_j in the two signatures can be used to extract the signing key. This is the intuition behind Theorem 4; the detailed proof appears in Appendix B.4.

Theorem 4 (2:1-DAPS is DSE*)**.** *In the setting of Construction 2, if X is extractable and $H^{\#}$ is collision-resistant, then double-authentication-preventing signature* 2:1-DAPS *is double-signature extractable with trusted setup.*

Remark 5 (Untrusted setup). The double-signature extractability of 2:1-DAPS in Theorem 4 relies on the assumption that signer's verification key is well-formed. When instantiated with Blum-2:1-TDF, 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 [vP88] 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.* [BFL91] 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.* [DDP94], and for products of safe primes (which includes Blum integers) by Camenisch and Michels [CM99].

5.3 Efficiency of construction based on sign-agnostic quadratic residues

Table 1 shows the size of verification keys, signing keys, and signatures, and the cost of signature generation and verification for the 2:1-DAPS based on Blum-2:1-TDF. We assume the element representation from Remark 1, the verification key optimization from Remark 2, and an implementation of random oracle H_{pub} as described in Section 4.2. We give concrete values to bound the forging probability by 2^{-60} for an adversary who makes $q_1 + q_2 \le 2^{50}$ random oracle queries and $q_S \le 2^{40}$ signature queries. Substituting into the expression in Theorem 3, noting that the **Dist** terms are (statistically) negligible, and taking $\lambda_h = 160$ so that the $\mathbf{Succ}^{\text{CR}}_{H^{\#},\mathcal{C}}(\lambda_h)$ term is insignificant, thus we need $\mathbf{Succ}^{\text{INV-2}}_{X}(\lambda) \le 2^{-112}$. Inverting in the factoring-based 2:1-TDF of Construction 1 is as hard as factoring. Applying, for example, ECRYPT recommendations [BCC+08] relating success probability and key length, we need an RSA modulus of $\lambda_2 = 2432$ bits. This results in signatures of about 48 kilobytes, which is somewhat large but not prohibitively so.

6 Applications

DAPS allows applications that rely on digital signatures to provide unique bindings to shift the risk/reward ratio for misbehaving signers. It is true that, if an accidental error occurs in a DAPS-based system, the

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

	General analysis	**Concrete parameters** ($\mathbf{Succ}^{\mathrm{EUF}}_{\text{2:1-DAPS},\mathcal{A}}(\lambda) \leq 2^{-60}$)
λ_h	—	160
λ_2 (size of n in bits)	—	2 432
Verification key size (bits)	$\log_2 n$	2 432
Signing key size (bits)	$\log_2 n$	2 432
Signature generation cost	$(\lambda_h + 1) \cdot \mathrm{Jac}$, $(\lambda_h + 1) \cdot \mathrm{sqrt}$	—
Signature size (bits)	$(\lambda_h + 1) \log_2 n$	$391\,552 = 48\,\mathrm{KiB}$
Signature verification cost	$(2\lambda_h + 1) \cdot \mathrm{Jac}$, $(\lambda_h + 1) \cdot \mathrm{sqr}$	—

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

damage is higher than with normal signatures. But this correspondingly gives signers a high incentive to not fraudulently sign multiple signatures for the same subject. When those who rely on signatures place a high value on the uniqueness of the binding between two values, it may be worthwhile to employ DAPS instead of traditional digital signatures, despite the potential for increased damage on account of accidental errors. In this section, we examine a few cryptographic applications involving unique bindings and discuss the potential applicability of DAPS.

Discouraging time-stamping authority fraud. 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 $y \neq x$ 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 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.

Discouraging CA coercion. The importance of reducing trust in CAs has been demonstrated by several recent incidents; for example, in 2011 Dutch CA DigiNotar was found to have issued fraudulent certificates that were used against Iranian Internet users [Goo11], resulting in the eventual distrusting of DigiNotar by all browser vendors and the company's subsequent bankruptcy. Recently, several distinct technical measures [EP12, MP12, HS12] have been proposed to try to wrest some trust decisions away from CAs.

DAPS could be used to ensure that certification authorities in the web PKI proceed with due diligence when signing certificates. 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`$\|$`2013`" 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.

PKIs generally provide functionalities beyond unique binding of a key to an identifier, including revocation and reissuing of certificates. 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`$\|$`2013`$\|$`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.

A DAPS-based CA also has some additional risks compared with a normal PKI: while the CA is discouraged from intentionally issuing multiple certificates for the same subject, the margin for accidental error is also eliminated. A naive application of DAPS leads to catastrophic failure if a double-sign occurs, but it is also possible to design less catastrophic DAPS.

Non-catastrophic DAPS. As described so far, when a signer signs two different messages with the same subject, anyone can recover enough information to either forge signatures or fully recover the signing key. This maximum disclosure is by design and may serve as a strong incentive for signers to behave. However, it may be desirable to construct schemes that have less catastrophic penalties. We can do so, 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 DAPS verification key (RSA modulus) N, and a verifiable Rabin encryption of, say, the first half of the bits of the standard scheme's signing key. The non-catastrophic 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.

7 Conclusions

We have introduced a new type of signatures, *double-authentication-preventing signatures*, in which a subject/message pair is signed. In certain situations, DAPS can provide greater assurance to verifiers that signers behave honestly since there is a great disincentive for signers who misbehave: if a signer ever signs two different messages for the same subject, then enough information is revealed to allow anyone to forge arbitrary signatures or even fully recover the signer's secret key. 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 not-unreasonable signature sizes.

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.

On the one hand, DAPS is more fragile and less robust to failure compared to traditional schemes. But on the other hand, in DAPS, any failure is catastrophic, and thus signers have a much higher incentive to ensure absolutely nothing goes wrong. Considering the amount of trust placed in CAs in public PKIs, DAPS may encourage CAs to behave even more securely and responsibly, since any mistake puts them out of business. For cases where a more graceful degradation is desired, in Section 6 we discuss a variant of DAPS that alleviates the extreme consequences of an infringement of the rules it enforces in an adjustable manner.

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?

Acknowledgments

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.

References

[BCC+08] Steve Babbage, Dario Catalano, Carlos Cid, Orr Dunkelman, Christian Gehrmann, Louis Granboulan, Tanja Lange, Arjen Lenstra, Phong Q. Nguyen, Christof Paar, Jan Pelzl, Thomas Pornin, Bart Preneel, Christian Rechberger, Vincent Rijmen, Matt Robshaw, Andy Rupp, Nigel Smart, and Michael Ward. ECRYPT yearly report on algorithms and keysizes (2007–2008), July 2008. URL http://www.ecrypt.eu.org/documents/D.SPA.28-1.1.pdf.

[BCI+09] Eric Brier, Jean-Sebastien Coron, Thomas Icart, David Madore, Hugues Randriam, and Mehdi Tibouchi. Efficient indifferentiable hashing into ordinary elliptic curves. Cryptology ePrint Archive, Report 2009/340, 2009. http://eprint.iacr.org/.

[BCI+10] Eric Brier, Jean-Sébastien Coron, Thomas Icart, David Madore, Hugues Randriam, and Mehdi Tibouchi. Efficient indifferentiable hashing into ordinary elliptic curves. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, *Lecture Notes in Computer Science*, volume 6223, pp. 237–254. Springer, August 2010.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, *Lecture Notes in Computer Science*, volume 2139, pp. 213–229. Springer, August 2001.

[BFL91] Joan Boyar, Katalin Friedl, and Carsten Lund. Practical zero-knowledge proofs: Giving hints and using deficiencies. *Journal of Cryptology*, **4**(3):185–206, 1991.

[BP97] Niko Bari and Birgit Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT'97*, *Lecture Notes in Computer Science*, volume 1233, pp. 480–494. Springer, May 1997.

[BR96] Mihir Bellare and Phillip Rogaway. The exact security of digital signatures: How to sign with RSA and Rabin. In Ueli M. Maurer, editor, *Advances in Cryptology – EUROCRYPT'96*, *Lecture Notes in Computer Science*, volume 1070, pp. 399–416. Springer, May 1996.

[CFN90] David Chaum, Amos Fiat, and Moni Naor. Untraceable electronic cash. In Shafi Goldwasser, editor, *Advances in Cryptology – CRYPTO'88*, *Lecture Notes in Computer Science*, volume 403, pp. 319–327. Springer, August 1990.

[CM99] Jan Camenisch and Markus Michels. Proving in zero-knowledge that a number is the product of two safe primes. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT'99*, *Lecture Notes in Computer Science*, volume 1592, pp. 107–122. Springer, May 1999.

[Cor02] Jean-Sébastien Coron. Optimal security proofs for PSS and other signature schemes. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, *Lecture Notes in Computer Science*, volume 2332, pp. 272–287. Springer, April / May 2002.

[Dam88] Ivan Damgård. Collision free hash functions and public key signature schemes. In David Chaum and Wyn L. Price, editors, *Advances in Cryptology – EUROCRYPT'87*, *Lecture Notes in Computer Science*, volume 304, pp. 203–216. Springer, April 1988.

[DDP94] Alfredo De Santis, Giovanni Di Crescenzo, and Giuseppe Persiano. Secret sharing and perfect zero knowledge. In Douglas R. Stinson, editor, *Advances in Cryptology – CRYPTO'93*, *Lecture Notes in Computer Science*, volume 773, pp. 73–84. Springer, August 1994.

[Des95] Yvo Desmedt. Securing traceability of ciphertexts - towards a secure software key escrow system (extended abstract). In Louis C. Guillou and Jean-Jacques Quisquater, editors, *Advances in Cryptology – EUROCRYPT'95*, *Lecture Notes in Computer Science*, volume 921, pp. 147–157. Springer, May 1995.

[DR02] Yevgeniy Dodis and Leonid Reyzin. On the power of claw-free permutations. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *SCN 02: 3rd International Conference on Security in Communication Networks*, *Lecture Notes in Computer Science*, volume 2576, pp. 55–73. Springer, September 2002.

[DY05] Yevgeniy Dodis and Aleksandr Yampolskiy. A verifiable random function with short proofs and keys. In Serge Vaudenay, editor, *PKC 2005: 8th International Workshop on Theory and Practice in Public Key Cryptography*, *Lecture Notes in Computer Science*, volume 3386, pp. 416–431. Springer, January 2005.

[EP12] Chris Evans and Chris Palmer. Public key pinning extension for HTTP, June 2012. URL http://tools.ietf.org/html/draft-ietf-websec-key-pinning-02. Internet-Draft, http://tools.ietf.org/html/draft-ietf-websec-key-pinning-02.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, **17**(2):281–308, 1988.

[Goo11] Google Online Security Blog. An update on attempted man-in-the-middle attacks, August 2011. URL http://googleonlinesecurity.blogspot.de/2011/08/update-on-attempted-man-in-middle.html. http://googleonlinesecurity.blogspot.de/2011/08/update-on-attempted-man-in-middle.html.

[HK09] Dennis Hofheinz and Eike Kiltz. The group of signed quadratic residues and applications. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, *Lecture Notes in Computer Science*, volume 5677, pp. 637–653. Springer, August 2009.

[HS12] Paul Hoffman and Jakob Schlyter. The DNS-based Authentication of Named Entities (DANE) Transport Layer Security (TLS) protocol: TLSA, August 2012. URL http://www.ietf.org/rfc/rfc6698.txt.

[IR90] K. Ireland and M. Rosen. *A Classical Introduction to Modern Number Theory*. Graduate Texts in Mathematics. Springer, 1990.

[KL07] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. Chapman and Hall/CRC Press, 2007.

[KR00] Hugo Krawczyk and Tal Rabin. Chameleon signatures. In *ISOC Network and Distributed System Security Symposium – NDSS 2000*. The Internet Society, February 2000.

[Lam79] Leslie Lamport. Constructing digital signatures from a one way function. Technical Report CSL-98, SRI International, October 1979.

[Mer90] Ralph C. Merkle. A certified digital signature (subtitle: That antique paper from 1979). In Gilles Brassard, editor, *Advances in Cryptology – Proc. CRYPTO '89*, *LNCS*, volume 435, pp. 218–238. Springer, 1990. DOI:10.1007/0-387-34805-0_21.

[MO12] Atefeh Mashatan and Khaled Ouafi. Forgery-resilience for digital signature schemes. In *Proc. 7th ACM Symposium on Information, Computer and Communications Security (ASIACCS) 2012*, pp. 24–25. ACM, 2012. DOI:10.1145/2414456.2414469.

[MP12] Moxie Marlinspike and Trevor Perrin. Trust assertions for certificate keys, September 2012. URL http://tools.ietf.org/html/draft-perrin-tls-tack-01. Internet-Draft, http://tools.ietf.org/html/draft-perrin-tls-tack-01.

[MvOV01] Alfred Menezes, Paul van Oorschot, and Scott Vanstone. *Handbook of Applied Cryptography*. CRC Press, 2001. URL http://www.cacr.math.uwaterloo.ca/hac/.

[Nat07] National Institute of Standards and Technology. Recommendation for random number generation using deterministic random bit generators (revised), March 2007. URL http://csrc.nist.gov/publications/nistpubs/800-90/SP800-90revised_March2007.pdf. NIST Special Publication 800-90.

[PP97] Torben Pryds Pedersen and Birgit Pfitzmann. Fail-stop signatures. *SIAM Journal on Computing*, **26**(2):291–330, 1997. DOI:10.1137/S009753979324557X.

[Sch90] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, *Lecture Notes in Computer Science*, volume 435, pp. 239–252. Springer, August 1990.

[Sho05] Victor Shoup. *A computational introduction to number theory and algebra*. Cambridge University Press, New York, NY, USA, 2005.

[vP88] Jeroen van de Graaf and René Peralta. A simple and secure way to show the validity of your public key. In Carl Pomerance, editor, *Advances in Cryptology – CRYPTO'87*, *Lecture Notes in Computer Science*, volume 293, pp. 128–134. Springer, August 1988.

[vP92] Eugène van Heyst and Torben P. Pedersen. How to make efficient fail-stop signatures. In Rainer A. Rueppel, editor, *Advances in Cryptology – EUROCRYPT'92*, *Lecture Notes in Computer Science*, volume 658, pp. 366–377. Springer, May 1992.

[vPP93] Eugène van Heijst, Torben P. Pedersen, and Birgit Pfitzmann. New constructions of fail-stop signatures and lower bounds (extended abstract). In Ernest F. Brickell, editor, *Advances in Cryptology – CRYPTO'92*, *Lecture Notes in Computer Science*, volume 740, pp. 15–30. Springer, August 1993.

[WP90] Michael Waidner and Birgit Pfitzmann. The dining cryptographers in the disco - underconditional sender and recipient untraceability with computationally secure serviceability (abstract) (rump session). In Jean-Jacques Quisquater and Joos Vandewalle, editors, *Advances in Cryptology – EUROCRYPT'89*, *Lecture Notes in Computer Science*, volume 434, p. 690. Springer, April 1990.

A 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 [MvOV01, Ch. 2–3], [KL07, Ch. 7, 11], or on number theory [IR90] for details.

Fact 1 (Quadratic residues modulo p)**.** *Let p be a prime number. Then $QR_p = \{x^2 : x \in \mathbb{Z}_p^\times\}$ denotes the* group of quadratic residues *modulo p. 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 $p \equiv 3 \bmod 4$ 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 $p \equiv q \equiv 3 \bmod 4$, 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 $x \mapsto (x \bmod p, x \bmod q)$. 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 $QR_n = \{x^2 : x \in \mathbb{Z}_n^\times\}$ 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 $\left(\frac{a}{n}\right) = \left(\frac{a \bmod p}{p}\right)\left(\frac{a \bmod q}{q}\right)$. 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 $p \equiv 3 \bmod 8$ and $q \equiv 7 \bmod 8$, 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 Figure 6 (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 $(x_0 - x_1)(x_0 + x_1) \equiv x_0^2 - x_1^2 \equiv y - y \equiv 0 \bmod n$, 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 Figure 6 (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, and $\overline{J}_n$)**.** *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*

$$E : \{0,1\}^\ell \to \mathbb{Z}_n : r \mapsto r \bmod n \qquad \text{and} \qquad F : \mathbb{Z}_n^\times \to J_n : x \mapsto \begin{cases} x & \text{if } \left(\frac{x}{n}\right) = +1 \\ xt & \text{if } \left(\frac{x}{n}\right) = -1 \end{cases}.$$

A common method (see [Des95, Sho05] and [Nat07, §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 [Sho05]. 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.

B Proofs

B.1 Proofs from Section 3

B.1.1 Proof of Lemma 1

Proof. 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 $\mathbf{Dist}^{U(B),\mathsf{Apply}(U(A))}_{X,\mathcal{D}}(\lambda) = 0$ for any distinguisher $\mathcal{D}$, the triangle inequality shows

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

□ □

B.1.2 Proof of Lemma 2

Proof. 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

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

where $\mathbf{Exp}^{\text{INV-2}^*}_{X,\mathcal{B}}$ is identical to $\mathbf{Exp}^{\text{INV-2}}_{X,\mathcal{B}}$ (cf. Figure 5) except that it returns 1 iff $(a \overset{\times}{\sim} a' \vee a = a')$. As Apply is 2:1, we have $\Pr\left[\mathbf{Exp}^{\text{INV-2}^*}_{X,\mathcal{B}}(\lambda) = 1\right] = 2 \cdot \Pr\left[\mathbf{Exp}^{\text{INV-2}}_{X,\mathcal{B}}(\lambda) = 1\right] = 2 \cdot \mathbf{Succ}^{\text{INV-2}}_{X,\mathcal{B}}(\lambda)$. We combine these results to obtain

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

The statement of Lemma 2 follows immediately. □ □

B.1.3 Proof of Lemma 3

Proof. 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, $\mathbf{Succ}^{\text{INV-1}}_{X,\mathcal{A}}(\lambda) = \mathbf{Succ}^{\text{INV-2}}_{X,\mathcal{B}}(\lambda)$. □ □

B.2 Lemma 7 and proofs from Section 4

Lemma 7. *Let n be a Blum integer. Then $QR_n/_{\pm 1} = \left\{\{\pm x\}^2 : \{\pm x\} \in \mathbb{Z}_n^\times/_{\pm 1}\right\}$.*

Proof. "$\subseteq$": 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 $\{\pm x\}^2 = \{\pm(x^2)\} = \{\pm y\}$. "$\supseteq$": 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 $\{\pm x\}^2 = \{\pm y\} \in QR_n/_{\pm 1}$. □ □

B.2.1 Proof of Lemma 4

Proof. 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 equation (1). Factorization and computation of square roots immediately follow from Fact 4. □ □

B.2.2 Proof of Lemma 5

Proof. 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 4. □ □

B.2.3 Proof of Lemma 6

Proof. 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. □ □

B.2.4 Proof of Theorem 2

Proof. Samplability. That $\mathbf{Dist}_{X,\mathcal{D}_A}^{\mathsf{Sample}_A, U(A)}(\lambda)$ and $\mathbf{Dist}_{X,\mathcal{D}_B}^{\mathsf{Sample}_B, U(B)}(\lambda)$ are negligible for all efficient algorithms $\mathcal{D}_A$ and $\mathcal{D}_B$ is exactly the statement of Lemma 6.

(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 $\{\pm x_0\} \not\sim \{\pm x_1\}$, i.e. such that $\{\pm x_0\} \neq \{\pm x_1\}$ and $\{\pm x_0\}^2 = \{\pm x_1\}^2$. By Lemma 4, this suffices for factoring n.

Extractability. Given are $\{\pm x_0\}, \{\pm x_1\} \in \mathbb{Z}_n^\times/_{\pm 1}$ such that $\{\pm x_0\} \not\sim \{\pm x_1\}$, i.e. such that $\{\pm x_0\} \neq \{\pm x_1\}$ and $\{\pm x_0\}^2 = \{\pm x_1\}^2$. By Lemma 4, this suffices for factoring n and recovering trapdoor $\mathsf{td} = (p, q)$. □ □

B.3 Proof of unforgeability (Theorem 3)

Proof. We use a sequence of games; underlining colors are used to highlight changes and additions between games. Let $\mathcal{A}$ be an adversary for experiment $\mathbf{Exp}_{2\text{:}1\text{-}\mathsf{DAPS}}^{\mathrm{EUF}}$. 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[2]. Let S_i be the event that game i outputs 1 when running $\mathcal{A}$.

Game 0. This is the original EUF experiment for 2:1-DAPS. For clarity, we write it in full detail:

1. $(\mathsf{td}, \mathsf{pub}) \leftarrow_R \mathsf{TdGen}(1^{\lambda_2})$
2. $(\mathsf{subj}^*, \mathsf{msg}^*, \sigma^*) \leftarrow_R \mathcal{A}^{\mathcal{O}_{\mathsf{Sign}}, H_{\mathsf{pub}}}(\mathsf{pub})$
   - If $\mathcal{A}$ queries $H_{\mathsf{pub}}(\mathsf{subj})$:
     (a) If $(\mathsf{subj}, b) \in \mathrm{HList}_1$, return b to $\mathcal{A}$
     (b) $b \leftarrow_R \mathsf{Sample}_B(\mathsf{pub})$
     (c) Append (subj, b) to HList_1
     (d) Return b to $\mathcal{A}$
   - If $\mathcal{A}$ queries $H_{\mathsf{pub}}(\mathsf{subj}, s, i)$:
     (a) If $(\mathsf{subj}, s, i, b_i) \in \mathrm{HList}_3$, return b_i to $\mathcal{A}$
     (b) $b_i \leftarrow_R \mathsf{Sample}_B(\mathsf{pub})$
     (c) Append $(\mathsf{subj}, s, i, b_i)$ to HList_3
     (d) Return b_i to $\mathcal{A}$
   - If $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{Sign}}(\mathsf{subj}, \mathsf{msg})$:
     (a) Append $(\mathsf{subj}, \mathsf{msg})$ to SignedList
     (b) $s \leftarrow \mathsf{Reverse}(\mathsf{td}, H_{\mathsf{pub}}(\mathsf{subj}), 0)$
     (c) $(d_1, \ldots, d_{\lambda_h}) \leftarrow H^{\#}(\mathsf{subj}, s, \mathsf{msg})$
     (d) $b_i \leftarrow H_{\mathsf{pub}}(\mathsf{subj}, s, i)$ for all $1 \leq i \leq \lambda_h$
     (e) $a_i \leftarrow \mathsf{Reverse}(\mathsf{td}, b_i, d_i)$ for all $1 \leq i \leq \lambda_h$
     (f) $\sigma \leftarrow (s, a_1, \ldots, a_{\lambda_h})$

[2]Observe that this assumption is quite natural as random oracles are usually constructed from such samplers. This holds in particular for Blum-2:1-TDF and the random oracle implementation we propose in Section 4.2.

(g) Return σ to $\mathcal{A}$

3. Return 1 iff all the following hold:
   - $\mathsf{Ver}(\mathsf{pub}, \mathsf{subj}^*, \mathsf{msg}^*, \sigma^*) = 1$
   - $(\mathsf{subj}^*, \mathsf{msg}^*) \notin \text{SignedList}$
   - $\forall\, \mathsf{subj}, \mathsf{msg}_0, \mathsf{msg}_1 : (\mathsf{subj}, \mathsf{msg}_0), (\mathsf{subj}, \mathsf{msg}_1) \in \text{SignedList} \Rightarrow \mathsf{msg}_0 = \mathsf{msg}_1$

By definition,

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

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 $\mathsf{Sample}_B(\mathsf{pub})$ algorithm to use instead the hybrid construction from Definition 10. These changes will not be detected unless one can either invert the 2:1-TDF or can distinguish the two sampling methods.

1. $(\cdot, \mathsf{pub}) \leftarrow_R \mathsf{TdGen}(1^{\lambda_2})$
2. $(\mathsf{subj}^*, \mathsf{msg}^*, \sigma^*) \leftarrow_R \mathcal{A}^{\mathcal{O}_{\mathsf{Sign}}, H_{\mathsf{pub}}}(\mathsf{pub})$
   - If $\mathcal{A}$ queries $H_{\mathsf{pub}}(\mathsf{subj})$:
     (a) If $(\mathsf{subj}, a, b) \in \text{HList}_1$, return b to $\mathcal{A}$
     (b) $a \leftarrow_R \mathsf{Sample}_A(\mathsf{pub}, 0)$
     (c) $b \leftarrow \mathsf{Apply}(\mathsf{pub}, a)$
     (d) Append (subj, a, b) to HList_1
     (e) Return b to $\mathcal{A}$
   - If $\mathcal{A}$ queries $H_{\mathsf{pub}}(\mathsf{subj}, s, i)$:
     (a) If $(\mathsf{subj}, s, i, a_i, b_i) \in \text{HList}_3$, return b_i to $\mathcal{A}$
     (b) $a_i \leftarrow_R \mathsf{Sample}_A(\mathsf{pub})$
     (c) $b_i \leftarrow \mathsf{Apply}(\mathsf{pub}, a_i)$
     (d) Append $(\mathsf{subj}, s, i, a_i, b_i)$ to HList_3
     (e) Return b_i to $\mathcal{A}$
   - If $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{Sign}}(\mathsf{subj}, \mathsf{msg})$:
     (a) Append $(\mathsf{subj}, \mathsf{msg})$ to SignedList
     (b) $t \leftarrow H_{\mathsf{pub}}(\mathsf{subj})$
     (c) Event F1: Abort if there exists $(\mathsf{subj}, s, \cdot, \cdot, \cdot) \in \text{HList}_3$ such that $\mathsf{Apply}(\mathsf{pub}, s) = t$.
     (d) Retrieve (subj, s, t) from HList_1
     (e) $(d_1, \ldots, d_{\lambda_h}) \leftarrow H^{\#}(\mathsf{subj}, s, \mathsf{msg})$
     (f) $a_i \leftarrow \mathsf{Sample}_A(\mathsf{pub}, d_i)$ for all $1 \le i \le \lambda_h$
     (g) $b_i \leftarrow \mathsf{Apply}(\mathsf{pub}, a_i)$ for all $1 \le i \le \lambda_h$
     (h) Append $(\mathsf{subj}, s, i, a_i, b_i)$ to HList_3 for all $1 \le i \le \lambda_h$
     (i) $\sigma \leftarrow (s, a_1, \ldots, a_{\lambda_h})$
     (j) Return σ to $\mathcal{A}$
3. Return 1 iff all the following hold:
   - $\mathsf{Ver}(\mathsf{pub}, \mathsf{subj}^*, \mathsf{msg}^*, \sigma^*) = 1$
   - $(\mathsf{subj}^*, \mathsf{msg}^*) \notin \text{SignedList}$
   - $\forall\, \mathsf{subj}, \mathsf{msg}_0, \mathsf{msg}_1 : (\mathsf{subj}, \mathsf{msg}_0), (\mathsf{subj}, \mathsf{msg}_1) \in \text{SignedList} \Rightarrow \mathsf{msg}_0 = \mathsf{msg}_1$

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 $\mathsf{Sample}_B(\mathsf{pub})$ from values returned by running $a \leftarrow_R \mathsf{Sample}_A(\mathsf{pub})$ and then returning $\mathsf{Apply}(\mathsf{pub}, a)$.

- 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 Sample^A_B and Sample_B.
- $H_{\mathsf{pub}}(\mathsf{subj}, s, i)$ queries: These values are always consistent with $H_{\mathsf{pub}}(\mathsf{subj})$ queries. 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 Sample^A_B and Sample_B. We note in the following point that $\mathcal{O}_{\mathsf{Sign}}$ queries might become inconsistent in certain circumstances.

- $\mathcal{O}_{\mathsf{Sign}}(\mathsf{subj}, \mathsf{msg})$ 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 $\mathsf{Apply}(\mathsf{pub}, s) = H_{\mathsf{pub}}(\mathsf{subj})$ and $\mathsf{Decide}(\mathsf{pub}, s) = 0$. 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 Sample_B^A and Sample_B.

Thus,

$$|\Pr[S_0] - \Pr[S_1]| \leq (q_1 + q_2 + (\lambda_h + 1)q_S)\, \mathbf{Dist}_{X,\mathcal{D}_B}^{\mathsf{Sample}_B^A, \mathsf{Sample}_B}(\lambda_2) + \Pr[\mathsf{F1}] \ . \tag{3}$$

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

$$\Pr[\mathsf{F1}] \leq q_1\, \mathbf{Succ}_{X,\mathcal{B}_1}^{\text{INV-1}}(\lambda_2) \ . \tag{4}$$

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 $(\mathsf{subj}, s, \cdot, \cdot, \cdot) \in \text{HList}_3$ 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 $\mathbf{Exp}_{X,\mathcal{B}_1}^{\text{INV-1}}(\lambda_2)$. Thus, $\Pr[\mathsf{F1}] \leq q_1 \Pr\left[\mathbf{Succ}_{X,\mathcal{B}_1}^{\text{INV-1}}(\lambda_2) = 1\right]$.

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. $(\mathsf{subj}^*, \mathsf{msg}^*, \sigma^*) \leftarrow_R \mathcal{A}^{\mathcal{O}_{\mathsf{Sign}}, H_{\mathsf{pub}}}(\mathsf{pub})$
   - If $\mathcal{A}$ queries $H_{\mathsf{pub}}(\mathsf{subj})$:
     (a) If $(\mathsf{subj}, a, b) \in \text{HList}_1$, return b to $\mathcal{A}$
     (b) $a \leftarrow_R \mathsf{Sample}_A(\mathsf{pub}, 0)$
     (c) $b \leftarrow \mathsf{Apply}(\mathsf{pub}, a)$
     (d) Append (subj, a, b) to HList_1
     (e) Return b to $\mathcal{A}$
   - If $\mathcal{A}$ queries $H_{\mathsf{pub}}(\mathsf{subj}, s, i)$:
     (a) If $(\mathsf{subj}, s, i, a_i, b_i) \in \text{HList}_3$, return b_i to $\mathcal{A}$
     (b) $a_i \leftarrow_R \mathsf{Sample}_A(\mathsf{pub})$
     (c) $b_i \leftarrow \mathsf{Apply}(\mathsf{pub}, a_i)$
     (d) Append $(\mathsf{subj}, s, i, a_i, b_i)$ to HList_3
     (e) Return b_i to $\mathcal{A}$
   - If $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{Sign}}(\mathsf{subj}, \mathsf{msg})$:
     (a) Append $(\mathsf{subj}, \mathsf{msg})$ to SignedList
     (b) $t \leftarrow H_{\mathsf{pub}}(\mathsf{subj})$
     (c) Event F1: Abort if there exists $(\mathsf{subj}, s, \cdot, \cdot, \cdot) \in \text{HList}_3$ such that $\mathsf{Apply}(\mathsf{pub}, s) = t$.
     (d) Retrieve (subj, s, t) from HList_1
     (e) $(d_1, \ldots, d_{\lambda_h}) \leftarrow H^{\#}(\mathsf{subj}, s, \mathsf{msg})$
     (f) $a_i \leftarrow \mathsf{Sample}_A(\mathsf{pub}, d_i)$ for all $1 \leq i \leq \lambda_h$
     (g) $b_i \leftarrow \mathsf{Apply}(\mathsf{pub}, a_i)$ for all $1 \leq i \leq \lambda_h$
     (h) Append $(\mathsf{subj}, s, i, a_i, b_i)$ to HList_3 for all $1 \leq i \leq \lambda_h$
     (i) $\sigma \leftarrow (s, a_1, \ldots, a_{\lambda_h})$
     (j) Return σ to $\mathcal{A}$
3. Return 1 iff all the following hold:
   - $\mathsf{Ver}(\mathsf{pub}, \mathsf{subj}^*, \mathsf{msg}^*, \sigma^*) = 1$
   - $(\mathsf{subj}^*, \mathsf{msg}^*) \notin \text{SignedList}$
   - $\forall\, \mathsf{subj}, \mathsf{msg}_0, \mathsf{msg}_1 : (\mathsf{subj}, \mathsf{msg}_0), (\mathsf{subj}, \mathsf{msg}_1) \in \text{SignedList} \Rightarrow \mathsf{msg}_0 = \mathsf{msg}_1$
   - Event $\neg$F2: $\forall\ i\ \exists\ (\mathsf{subj}^*, s^*, i, a_i^*, b_i) \in \text{HList}_3 : \mathsf{Apply}(\mathsf{pub}, a_i^*) = b_i$

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 $\neg$F2 for the experiment to output 1. Clearly, then,

$$|\Pr[S_1] - \Pr[S_2]| \leq \Pr[\mathsf{F2}] \ . \tag{5}$$

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 $H_{\mathsf{pub}}(\mathsf{subj}^*, s^*, i)$ was first computed when the challenger tried to verify the signature in step 3. Since it was computed by choosing $a_i \leftarrow_R \mathsf{Sample}_A(\mathsf{pub})$, it is unlikely that $\mathcal{A}$ could guess this a_i in advance. In particular, any algorithm that detects this serves as a distinguisher $\mathcal{D}_A$ between Sample_A and $U(A)$

$$\Pr[\mathsf{F2}] \leq \mathbf{Dist}_{X,\mathcal{D}_A}^{\mathsf{Sample}_A, U(A)}(\lambda_2) \ . \tag{6}$$

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

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

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

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 F1 nor 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 $a_i \not\sim a^*$. This is a solution to the INV-2 challenge a^*, which $\mathcal{B}_2$ outputs to win $\mathbf{Exp}_{X,\mathcal{B}_2}^{\text{INV-2}}(\lambda_2)$.

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 $\mathbf{Exp}_{X,\mathcal{B}_2}^{\text{INV-2}}(\lambda_2)$.

Final result. The final result follows from combining equations (3) through (7) and applying Lemma 1. □ □

B.4 Proof of double-signature extractability (Theorem 4)

Proof. We propose the following DSE* extractor (cf. Definition 6):

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

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, $H^\#(\mathsf{subj}, s, \mathsf{msg}_1) \neq H^\#(\mathsf{subj}, s, \mathsf{msg}_2)$. 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 $j \in [1, \lambda_h]$ such that $d_j^1 \neq d_j^2$.

Now, since both σ_1 and σ_2 are valid, we have that $\mathsf{Apply}(\mathsf{pub}, a_j^1) = H_{\mathsf{pub}}(\mathsf{subj}, s, j) = \mathsf{Apply}(\mathsf{pub}, a_j^2)$, but $\mathsf{Decide}(\mathsf{pub}, a_j^1) \neq \mathsf{Decide}(\mathsf{pub}, a_j^2)$. In other words, $a_j^1 \not\sim a_j^2$. Thus, 2:1-TDF's $\mathsf{Extract}(\mathsf{pub}, a_j^1, a_j^2)$ returns the trapdoor td corresponding to pub. □ □

C Claw-free permutations

Claw-free permutations (CFPs) were proposed by Goldwasser, Micali, and Rivest [GMR88], and have found several applications [KR00, Dam88, DR02, GMR88]. We briefly describe their functionality and security properties.

Consider a finite set D and a pair of one-way trapdoor permutations $g_0, g_1 : D \to D$. More precisely, functions g_0, g_1 shall be bijective and efficiently computable, while inverse functions g_0^{-1}, g_1^{-1} shall be computable only with knowledge of a specific trapdoor. Such a triple (D, g_0, g_1) is called *claw-free* if the problem of finding *claws* — pairs $x, y \in D$ such that $g_0(x) = g_1(y)$ — is hard. See Figure 8 (left) for an illustration.

Let us recall the factoring-based CFP of Goldwasser *et al.* [GMR88], adapted to our notation from Section 4.1. Let $n = pq$ be a Blum integer with the additional property that $p \equiv 3 \bmod 8$ and $q \equiv 7 \bmod 8$, so $\left(\frac{2}{n}\right) = -1$. Let $D = QR_n/_{\pm 1}$, and define $g_0 : D \to D : \{\pm x\} \mapsto \{\pm x\}^2$ and $g_1 : D \to D : \{\pm x\} \mapsto 4 \cdot \{\pm x\}^2$. This yields a secure (i.e., claw-free) CFP, under the assumption that factoring Blum integer n is hard [GMR88]. In the corresponding security reduction it is shown that n can be factored by knowledge of any claw, i.e., any pair $\{\pm x\}, \{\pm y\} \in QR_n/_{\pm 1}$ with $\{\pm x\}^2 = 4 \cdot \{\pm y\}^2$. We therefore see that this specific CFP has an extraction functionality: anybody that finds a claw for g_0, g_1 can also compute the correct trapdoor to invert these functions. This property is similar to the extraction property of 2:1-TDFs from Section 3.3. However, it was not formalized by Goldwasser *et al.* [GMR88].

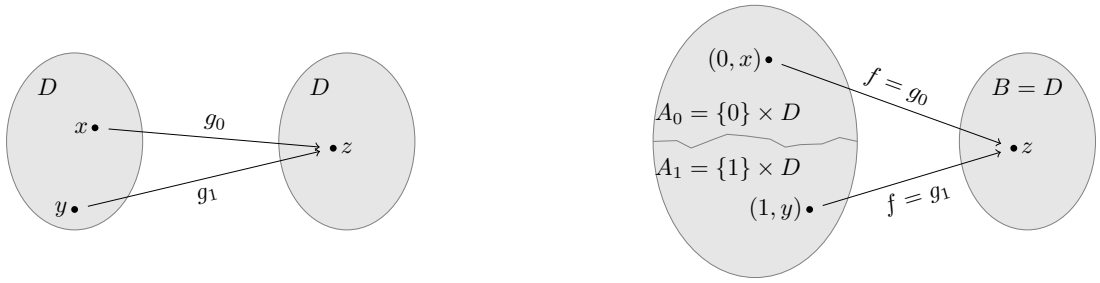

Figure 8: Left: Illustration of a CFP. Right: Illustration of the CFP-based 2:1-TDF construction.

C.1 Constructing (extractable) 2:1-TDFs from claw-free permutations

There is a simple way to construct 2:1-TDFs from CFPs (D, g_0, g_1). Using the notation from Section 3, let $A = \{0,1\} \times D$ and $B = D$. Define $f : A \to B : (d, x) \mapsto g_d(x)$ and $f^{-1} : B \times \{0,1\} \to A : (b, d) = (d, g_d^{-1}(b))$. That is, f processes elements in A_0 with permutation g_0 and elements in A_1 with permutation g_1, as illustrated in Figure 8 (right).

It is immediate to see that $|A| = 2|B|$ and that $f : A \to B$ is 2:1. Moreover, the decision whether an element $(d, x) \in A$ is in A_0 or in A_1 is determined directly from bit d: $A_0 = \{0\} \times D$ and $A_1 = \{1\} \times D$. It is also easy to verify that the 2:1-TDF correctness requirements from Definition 8 hold.

Observe that this construction inherits extractability from the underlying CFP. Indeed, claws of the CFP correspond exactly with pairs $a, a' \in A$ with $a \overset{\times}{\sim} a'$. Hence, if the CFP's trapdoor can be recovered from any claw, then the 2:1-TDF's trapdoor can be recovered from any pair a, a' with $a \overset{\times}{\sim} a'$.

Although we have seen how to construct (extractable) 2:1-TDFs from (extractable) claw-free permutations, it remains unclear whether the converse is feasible as well: given a generic 2:1-TDF, can we construct a CFP from it? We found no such construction, and the fact that CFPs are permutations $D \to D$, while 2:1-TDFs are defined as $A \to B$ for *arbitrary* (generally unrelated) sets A and B, might indicate that such constructions may not necessarily exist.

If it indeed turns out to be impossible to generically construct CFPs from 2:1-TDFs, one could say that 2:1-TDFs are formally based on weaker assumptions, and thus should be favored over CFPs, where possible, when used as a building block in a cryptographic protocol. On the other hand, it is true that the security of all currently known constructions of both 2:1-TDFs and CFPs is based on the same hard problem: factoring Blum integers.