Short Double- and N-Times-Authentication-Preventing Signatures from ECDSA and More

David Derler[1], Sebastian Ramacher[1], and Daniel Slamanig[2]

[1] IAIK, Graz University of Technology, Austria
[2] AIT Austrian Institute of Technology GmbH, Vienna, Austria
firstname.lastname@tugraz.at, firstname.lastname@ait.ac.at

Abstract. Double-authentication-preventing signatures (DAPS) are signatures designed with the aim that signing two messages with an identical first part (called address) but different second parts (called payload) allows to publicly extract the secret signing key from two such signatures. A prime application for DAPS is disincentivizing and/or penalizing the creation of two signatures on different payloads within the same address, such as penalizing double spending of transactions in Bitcoin by the loss of the double spender's money.

So far DAPS have been constructed from very specific signature schemes not used in practice and using existing techniques it has proved elusive to construct DAPS schemes from signatures widely used in practice. This, unfortunately, has prevented practical adoption of this interesting tool so far. In this paper we ask whether one can construct DAPS from signature schemes used in practice. We affirmatively answer this question by presenting novel techniques to generically construct provably secure DAPS from a large class of discrete logarithm based signatures. This class includes schemes like Schnorr, DSA, EdDSA, and, most interestingly for practical applications, the widely used ECDSA signature scheme. The resulting DAPS are highly efficient and the shortest among all existing DAPS schemes. They are nearly half of the size of the most efficient factoring based schemes (IACR PKC'17) and improve by a factor of 100 over the most efficient discrete logarithm based ones (ACM CCS'15). Although this efficiency comes at the cost of a reduced address space, i.e., size of keys linear in the number of addresses, we will show that this is not a limitation in practice. Moreover, we generalize DAPS to any $N > 2$, which we denote as N-times-authentication-preventing signatures (NAPS). Finally, we also provide an integration of our ECDSA-based DAPS into the OpenSSL library and perform an extensive comparison with existing approaches.

1 Introduction

Digital signatures are the prevalent cryptographic primitive to provide strong integrity and authenticity guarantees for messages exchanged in the digital realm. They are used in major cryptographic protocols such as TLS, for issuing digital

certificates (i.e., certifying public keys) within public-key infrastructures (PKIs), to authenticate executable code or digital documents such as PDF documents (in a legally binding way) or to sign transactions within the distributed cryptocurrency Bitcoin, to name some popular applications. Arguably, as they enable the secure distribution and transmission of public keys, in a very real sense, they serve as the foundation of all public key cryptography in practice.

Most widely used signature schemes today are (1) RSA-FDH, either used with PKCS#1 v1.5 padding or as probabilistic signature scheme (RSA-PSS), and (2) the discrete logarithm based (elliptic curve) digital signature algorithm (EC)DSA. While RSA is predominant in legacy applications, more recent applications that make heavy use of digital signatures (such as Bitcoin) build upon ECDSA. Actually, when analyzing the trend of the use of ECDSA for certificate signing, we can observe that its use is becoming increasingly popular over the last few years[3] (see Table 1). A similar trend can be observed in DNSSEC

Year	% of ECDSA signatures
2014	0.01 %
2015	0.02 %
2016	2.54 %
2017	36.07 %

Table 1: Usage of ECDSA signatures in certificates of the top million websites via `censys.io` [DAM+15].

in that an ever increasing number of DNSSEC resolvers support ECDSA[4] and some large companies like CloudFlare are heavily pushing ECDSA [vRJS16]. Papadopoulos et al. [PWH+17] argue that due to improved performance and security it is very likely that new features for DNSSEC such as NSEC5 will only target the elliptic curve setting instead of RSA. Actually, given that the use of RSA signatures within DNSSEC in practice suffers from deficient key generation methods [SW17], switching to elliptic curves seems to be a viable way to go.

Now let us recall digital signatures more technically. We have a signer who holds a secret signing key sk and publishes its corresponding public verification key pk. To sign a message m, the signer uses sk to produce a signature σ and anyone who is given (m, σ) together with an authentic copy of pk can verify that the message originated from the signer (authenticity) and has not been modified in any way (integrity). Formal security guarantees for a signature scheme require that anyone not holding sk, even if allowed to adaptively obtain signatures for messages of one's choice, will not be able to come up with a valid signature for a non-queried message, i.e., produce a forgery. This notion is coined existential unforgeability under chosen message attacks ($\mathsf{EUF\text{-}CMA}$), formally discussed in

[3] `https://blog.cloudflare.com/aes-cbc-going-the-way-of-the-dodo/`
[4] `https://blog.apnic.net/2016/10/06/dnssec-and-ecdsa/`

Section 4.1, and is the widely accepted security notion required by schemes used in practice today.

In this paper we consider a variant of signature schemes dubbed double-authentication-preventing signatures (DAPS) [PS14,PS17]. Here, messages to be signed are of the form $m = (a, p)$ and in particular they consist of an address a and a payload p. The basic idea behind DAPS is that they behave exactly like conventional signatures, i.e., provide unforgeability in the EUF-CMA sense, as long as no distinct payloads $p' \neq p$ are signed with respect to the same address a. If a signer produces two signatures for distinct payloads $p' \neq p$ *but* with respect to the same address a (called colliding messages), then *anyone* can compute the signer's secret key sk from these signatures (the so called double-signature extraction property).

This concept may sound awkward at first sight, but it is indeed interesting as it disincentivizes the signer from "double-signing". It suggests the use of DAPS instead of conventional signatures whenever double-signing should be disincentivized, where the address a (or its associated space respectively) can be given some application-dependent semantics. Thereby, we can consider any form of a digital processes where one wants to prevent fraud by discouraging users from submitting (signing) duplicates. Think for instance of requests for reimbursements for the same expense multiple times, which can be disincentivized when using some unique ID, identifying the invoice/payment as address. In Section 2 we discuss some representative and more concrete applications of DAPS.

We observe that this is conceptually related to some other approaches discussed subsequently, but DAPS are stronger in the sense that they reveal the secret key of the signer to the public. Within offline double spending mechanisms [CFN88] of centralized e-cash systems, as long as a user is honest, the user can anonymously conduct transactions. But if a user misbehaves and spends an e-coin multiple times, his identity is revealed. In contrast to just revealing the identity in case of misbehaviour, however, DAPS reveal the secret key of the signer. Revealing the secret key as discouragement to behave fraudulent is also related to what is done within the so called PKI-assured non-transferability approach in anonymous credential systems [CL01]. Here the secret of the credential is associated to a valuable secret outside the system, e.g., a secret key that allows to issue signatures that are equivalent to handwritten signatures, which disincentivizes the sharing of a credential. However, in contrast to DAPS the secret key is not made public per se, but known to everyone with whom the credential is shared.

A problem with existing DAPS constructions [PS14,RKS15,PS17,BPS17] is that they are not based on widely used signature schemes and thus have not seen adoption in practice. While the constructions in [PS14,PS17,BPS17] are factoring based ones (aka in the RSA setting), the one from Ruffing et al. in [RKS15] is compatible with discrete-logarithm based signature public keys (and ECDSA public keys in particular). Unfortunately, their integration of signature public

keys in so called accountable assertions[5], which Ruffing et al. instantiate with a Merkle-tree construction using chameleon hash functions [KR00], does not yield an efficient construction. Our aim in this paper is to provide a generic construction that augments existing signature schemes widely used in practice (such as ECDSA) to yield DAPS being provably secure, where the security proof makes only black-box use of the signature scheme.

1.1 Contribution

Our key contributions in this paper can be summarized as follows:

- We are the first to present DAPS that are based on widely deployed and used signature schemes and in particular ECDSA. Additionally, our approach also works identically for Schnorr signatures, DSA or EdDSA (and many other discrete-logarithm based schemes). Consequently, we provide the first construction that can be directly used in real world and deployed systems.
- Our DAPS are the *shortest* DAPS so far in any setting. For instance, for the 128 bit security level, signatures of our DAPS with ECDSA on 256 bit elliptic curve groups are 1280 bits long, whereas most efficient factoring-based DAPS with a modulus size of 2048 bit require 2049 bits. This compactness, however, comes at the cost of a reduced address space and public key size linearly depending on the address space. However, as we will show, practical use-cases only require small address spaces and thus keep the key sizes reasonably low.
- Our construction paradigm is a generic and novel approach to combine verifiable Shamir secret sharing with (linear) ElGamal encryption in a semi-black box way. In a nutshell, the idea is to homomorphically evaluate the verification relation of the verifiable secret sharing scheme in the encrypted domain and to prove that the respective encrypted evaluation actually contains the expected value. This, in turn, gives us the required flexibility to perform a black-box reduction to the EUF-CMA security of ECDSA, or, more generally, to the EUF-CMA security of any discrete logarithm based signature scheme where the public key is the image of the secret key under a group homomorphism. From a practical point of view, this allows an easy extension of existing (EC)DSA, EdDSA and Schnorr signing keys to DAPS keys.
- We generalize DAPS and show how our approach to construct DAPS can easily be extended to N-times-authentication-preventing signatures (dubbed NAPS) for any $N > 2$. This is achieved by setting the degree of the polynomial in Shamir's secret sharing to $N - 1$ (where we simply have a degree 1 polynomial in case of DAPS).
- We introduce notions of double-signing extraction security for DAPS schemes that extend keys of a conventional signature scheme. Our notions ensure that extractability of the signing key of the signature scheme, e.g., the ECDSA key, is required, even if it is not possible to extract the full DAPS secret

[5] Ruffing et al. show that certain accountable assertions (and in particular their construction) yield DAPS.

key. In applications where the signing key is also used in a different context, inadvertently leaking the signing key already disincentivizes double-authentication. We show that our construction satisfies this notion under adversarially chosen, i.e., malicious, keys.
– We provide an implementation of our DAPS and integration into the popular OpenSSL library, which requires no changes to OpenSSL's ECDSA interface and implementation. This allows faster adoption of our DAPS in existing applications such as Bitcoin.

2 Applications of DAPS

Below we discuss three appealing applications of DAPS. The first two are applications already given in [RKS15], which can be implemented with our construction much more efficiently. The last field of application is more generic and includes disincentivizing double-signing of certificates and executables.

Moreover, we stress that as our DAPS constructions are the first that are ready to be used based on a widely deployed signature scheme that is used in many real world applications and whose popularity is ever increasing. Thus, we are convinced that DAPS will find many more interesting applications.

2.1 Accountable Assertions and Non-equivocation Contracts

Accountable assertions introduced in [RKS15] are a cryptographic mechanism that allows binding of statements to contexts in an accountable way: if the attacker asserts two contradicting statements in the same context, then any observer can extract the attacker's secret key. DAPS can be viewed as a stronger variant of accountable assertions, as they are additionally required to be unforgeable. Hence efficient DAPS constructions also provide more efficient instantiations of accountable assertions.

Combining accountable assertions respectively DAPS with Bitcoin deposits as discussed in [RKS15] enables the construction of non-equivocation contracts. Latter make it possible to penalize equivocation in distributed protocols monetarily. If a party A should be penalized if it equivocates, A creates a new Bitcoin key pair and extends it to a DAPS key pair.[6] It creates a deposit under the newly created Bitcoin key pair. Whenever A is supposed to send a statement in some context, it additionally sends a signature under the corresponding DAPS key. If A equivocates, anyone can extract the secret key from the two assertions with respect to the same context and can hence transfer the funds stored in the deposit to an address under their control. In case that A does not equivocate, it keeps full control over the deposit.

[6] Ruffing et al. use the signature public key as a public key of a accountable assertion instead of using a DAPS directly.

2.2 Disincentivizing Bitcoin Double-Spending

A central issue in the Bitcoin protocol is that it takes some time (in the order of tens of minutes) until a transaction gets confirmed in the blockchain and thus becomes valid. This makes it hard to prevent double-spending for "fast" transactions, i.e., transactions which involve transferring goods immediately after completing a transaction. To this end various non-cryptographic means to detect double-spending in fast Bitcoin transactions were proposed [KAC12,KAR+15].

With DAPS we can come up with a cryptographic solution towards solving this problem that strongly disincentivizes double-spending of the aforementioned type. In particular, we can ensure that double-spending will reveal the signing key and thus the associated Bitcoin(s) of the misbehaving party. To achieve this we can follow a similar strategy as [RKS15], but building upon our DAPS yields a much more efficient solution which is suited to be directly added to the Bitcoin core with a few lines of code, i.e., by extending the existing use of ECDSA for signing to our DAPS based on ECDSA. To disincentivize double-spending for a limited number of offline transactions, a user A of a service B first transfers an amount of spendable coins and a penalty to a deposit. After the deposit was confirmed by the blockchain, A can buy services from B offline by signing transactions with the DAPS scheme and giving the signatures to B. Now, if A is honest throughout all transactions, A can clear the deposit after some threshold. However, when A double-spends the DAPS signatures leak the secret (ECDSA) key to B. Thus A looses the coins deposited as penalty, since B is now able to transfer the coins to a wallet under its control.

2.3 Disincentivizing Double-Signing

More generally, DAPS are useful to disincentivize double-signing. Poettering and Stebila [PS14,PS17] propose the use of DAPS for certificate signing within public key infrastructures (PKIs). For this application, it seems that [PS17] is favorable to what we will present. Nevertheless, there are other similar application, where—likewise to the other applications presented in this section—our novel constructions are favorable to prior work.

Think of the application of DAPS in context of code-signing, i.e., for the signing of executables. When DAPS are used, the address represents a unique ID (such as used by Apple's App Store or Google's Play Store) and the payload is the version number. Providing a clean and a backdoored variant of the same software version will leak the signing key. This disincentivizes such a behaviour as this will then likely lead to a pandemia of malware signed with such a key.

2.4 Observation Regarding the Address Space

Interestingly, we observe that none of the applications requires an exponentially large address space. For example the application to accountable assertions inherently only requires a single address. Furthermore, in the application to disincentivizing double-spending for fast Bitcoins transaction, one may observe that

a small number of addresses suffices. Consider for example a public transport company that allows customers to charge a transport pass for multiple trips. In this case the number of taken trips can serve as address. Finally, in the application to code signing one requires a somewhat larger address space, but still having an address space of size 100 would allow to sign a new software version every week for about two years.

3 Overview

In the following we provide an overview of the path we take in this paper to construct DAPS. Previous approaches to construct DAPS follow the idea of finding and formalizing some suitable cryptographic primitive that directly allows to obtain DAPS. Examples are 2:1 trapdoor functions as in [PS14,PS17], or certain trapdoor identification schemes as in [BPS17]. While such an approach is highly challenging and interesting from a theoretical perspective, following this approach makes it very unlikely that one ends up with DAPS that are based on some already deployed signature scheme like (EC)DSA. Our approach in this paper is diametrically opposed to this approach. Namely, we look at signature schemes used in practice and ask if and how we can turn them into DAPS. Thereby, we put our focus on the elliptic-curve (discrete logarithm) setting.

The dead end. Before we present our approach we briefly discuss why a seemingly rather obvious path unfortunately does not work, as we consider this finding an interesting observation. When looking at schemes from the ElGamal family [Gam84,HPM94], like (EC)DSA or Schnorr [Sch89] signatures, it is well known that wrong usage may inadvertently leak the entire secret signing key. More precisely, due to the nature of these schemes, using the same randomness for computing signatures on different messages—as already happened in the past either due to erroneously fixing the randomness[7] or due to a bad randomness generation[8]—reveals the secret signing key. While there are countermeasures to avoid the aforementioned issues in practice at all by either making (EC)DSA deterministic [Por13] or by explicitly designing deterministic schemes such as EdDSA [BDL+12], the randomized versions, which are susceptible to the above problem, are still those most commonly used.

Now, one could try to make this aforementioned "bug" a "feature" and use this inherent property of such signature schemes in a positive way to construct DAPS. Recall, that DAPS require extraction of the signing key when given two signatures for colliding messages. Now what we could do is to adopt the idea as used by [Por13,BDL+12]. The idea would be to pseudorandomly compute the randomness used for signing from the message and the (secret) key. In contrast to making conventional signatures deterministic, in DAPS we cannot trust the signer to actually compute the randomness pseudorandomly from the address

[7] http://www.bbc.com/news/technology-12116051

[8] http://www.theregister.co.uk/2013/08/12/android_bug_batters_bitcoin_wallets/

and there must be some means for anyone to check that the signer indeed honestly computed the randomness from the address. Now, one could think that it would work to use a verifiable random function (VRFs) [MRV99] to derive the randomness pseudorandomly from the address. In short, a VRF is a public key primitive which computes some random and unique output from an input together with a publicly verifiable (implicit) proof of correct computation. If one would have a VRF where the randomness itself is not leaked, but its output is a group element and only the holder of the VRF secret key knows the discrete logarithm of this group element with respect to the base element of the group, then this could work. Indeed, the Dodis-Yampolskiy (DY) construction [DY05] satisfies this property and additionally has compact keys and proofs.[9] While using such a VRF to derive the randomness for the signature scheme from the address seems intuitively secure, there does not seem to be a viable proof strategy to prove EUF-CMA security with a (black-box) reduction to the VRF and the signature scheme. The problem is that we see no way of decoupling the output of the VRF and the randomness in the signature scheme to come up with a working simulation strategy in the security proof. Even decoupling and proving consistency using NIZKs did not work for any strategy we tried. As we, moreover, do not want to resort on highly idealized models such as the generic group model [Sho97] to directly analyse such a DAPS construction (cf. Section 4.3 for problems with such an analysis for ECDSA), we pursue an alternative path where we can avoid such models use the signature scheme in a black-box fashion.

A working path. Besides the problems which turn up when pursuing the direction sketched above, it turns out to be highly non-trivial to achieve the desired functionality in the discrete logarithm setting in general. In particular, the requirement to be able to extract a certain discrete logarithm, i.e., the secret key, as soon as more than one signature within the same context exists, makes it very hard to perform the simulation within the security reduction when trying to relate the unforgeability of the DAPS to the unforgeability of the underlying signature scheme in a black-box fashion.

Fortunately, we are nevertheless able to come up with novel techniques which are inspired by secret sharing. In particular, we use a secret sharing of the secret signing key (in $\mathbb{Z}_q$) such that producing signatures for two colliding messages, i.e., messages with identical address but different payloads, allows to reconstruct the secret, i.e., the signing key. If now every address a is associated to a degree 1 polynomial $f_a(X)$ with $f_a(0)$ being the signing key and every signature includes a share $f_a(p)$ (evaluation of the polynomial on the payload p of the message to be signed), two colliding messages reveal the signing key. The tricky part is that one additionally requires a mechanisms to convince a verifier that the signer behaves honest, i.e., really reveals a share of the key associated to the address-polynomial, while still preserving the ability to conduct the simulation in the security reduction. While latter is typically approached by adding verifiability to the secret sharing scheme using a mapping of the coefficients defining

[9] We could even avoid bilinear groups in the DY VRF by providing an efficient NIZK of validity of the verification equation instead of using a pairing to check the proof.

$f_a(X)$ to the group $\mathcal{G} = (\mathbb{G}, q, g)$, we can not do so as this immediately destroys the possibility to conduct a black-box reduction to the EUF-CMA security of the underlying signature scheme (essentially the public verifiability destroys the possibility to simulate in the security proof).

To this end, we need a trick to decouple the public verifiability of the secret sharing from the signing key to make the proof work. We approach this by encrypting the coefficients of the address-polynomials mapped to elements of $\mathbb{G}$ (except the constant term representing the public key of the signature scheme) and provide a zero-knowledge proof of knowledge (using an efficient Σ-protocol made non-interactive via Fiat-Shamir) that the value $f_a(p)$ in the signature really represents an evaluation of the encrypted address-polynomial. While conducting such a proof would already be sufficient for a working scheme, we additionally observe that we can employ linearly homomorphic encryption (e.g., ElGamal) to do some pre-computations before we actually conduct the proof. This, in turn, makes our approach highly efficient.

In addition, we observe that our approach directly allows us to derive a generalization to N-times-authentication-preventing signatures (NAPS) for arbitrary $N > 2$ by using higher degree polynomials.

Efficiency of our approach. Our constructions yield short signatures and are practically efficient (which we extensively discuss in Section 7). For instance, constructing a DAPS from ECDSA implemented using the `prime256v1` elliptic curve yield a signature of size 160 byte, being roughly 2.5 times the size of conventional ECDSA signatures. Signing is roughly 3.8 times and verification 1.6 times of conventional ECDSA. On the platform we use for benchmarking, signing and verification require 0.23 and 0.35 ms respectively.

4 Signature Schemes

In this section we firstly present a formal model for the security of signature schemes. Secondly, we present the ECDSA signature scheme which we later use to instantiate our DAPS construction.

4.1 Formal Model

Definition 1 (Signature Scheme). *A signature scheme* Σ *is a triple* $(\mathsf{KGen}_\Sigma, \mathsf{Sign}_\Sigma, \mathsf{Verify}_\Sigma)$ *of PPT algorithms, which are defined as follows:*

$\mathsf{KGen}_\Sigma(1^\kappa)$: *This algorithm takes a security parameter* κ *as input and outputs a secret (signing) key* sk_Σ *and a public (verification) key* pk_Σ *with associated message space* $\mathcal{M}$ *(we may omit to make the message space* $\mathcal{M}$ *explicit).*

$\mathsf{Sign}_\Sigma(\mathsf{sk}_\Sigma, m)$: *This algorithm takes a secret key* sk_Σ *and a message* $m \in \mathcal{M}$ *as input and outputs a signature* σ*.*

$\mathsf{Verify}_\Sigma(\mathsf{pk}_\Sigma, m, \sigma)$: *This algorithm takes a public key* pk_Σ*, a message* $m \in \mathcal{M}$ *and a signature* σ *as input and outputs a bit* $b \in \{0, 1\}$*.*

We require a signature scheme to be correct and EUF-CMA secure. For correctness we require that for all $\kappa \in \mathbb{N}$, for all $(\mathsf{sk}_\Sigma, \mathsf{pk}_\Sigma) \leftarrow \mathsf{KGen}_\Sigma(1^\kappa)$ and for all $m \in \mathcal{M}$ it holds that

$$\Pr\left[\mathsf{Verify}_\Sigma(\mathsf{pk}_\Sigma, m, \mathsf{Sign}_\Sigma(\mathsf{sk}_\Sigma, m)) = 1\right] = 1.$$

Definition 2 (EUF-CMA). *A signature scheme Σ is* EUF-CMA *secure, if for all PPT adversaries $\mathcal{A}$ there is a negligible function $\varepsilon(\cdot)$ such that*

$$\Pr\left[\mathbf{Exp}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{A},\Sigma}(\kappa) = 1\right] \leq \varepsilon(\kappa),$$

where the corresponding experiment is depicted in Figure 1.

$\mathbf{Exp}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{A},\Sigma}(\kappa)$:
  $(\mathsf{sk}_\Sigma, \mathsf{pk}_\Sigma) \leftarrow \mathsf{KGen}_\Sigma(1^\kappa)$
  $\mathcal{Q} \leftarrow \emptyset$
  $(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}'_\Sigma(\mathsf{sk}_\Sigma, \cdot)}(\mathsf{pk})$
    where oracle Sign'_Σ on input m:
      let $\sigma \leftarrow \mathsf{Sign}_\Sigma(\mathsf{sk}_\Sigma, m)$
      set $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{m\}$
      return σ
  return 1, if $\mathsf{Verify}_\Sigma(\mathsf{pk}_\Sigma, m^*, \sigma^*) = 1 \ \wedge \ m^* \notin \mathcal{Q}$
  return 0

Fig. 1: EUF-CMA **security.**

4.2 Elliptic Curve Groups

We briefly recall groups from elliptic curves. Let an elliptic curve E over the finite field $\mathbb{F}_p$ be a plane, smooth algebraic curve usually defined by a Weierstrass equation. The set $E(\mathbb{F}_p)$ of points $(x, y) \in \mathbb{F}_p^2$ satisfying this equation plus the point at infinity $\mathcal{O}$, which is the neutral element, forms an additive Abelian group, whereas the group law is determined by the chord-and-tangent method. If we write P_x we refer to the x coordinate of a point P. In general, we write $\mathcal{G} = (\mathbb{G}, q, g)$ to denote a group $\mathbb{G}$ of order q with generator g and we always use multiplicative notion throughout the paper.

4.3 ECDSA

In Scheme 1 we recall the ECDSA signature scheme. Thereby, $H' : \{0,1\}^* \to \mathbb{Z}_q$ is a hash function mapping exactly to the order of the group.

The security analysis of ECDSA was for quite some time a topic of debates. There exist proofs of security of modified variants of ECDSA [MS02].

$\underline{\mathsf{KGen}_{\mathsf{ECDSA}}(1^\kappa)}$: Let $\mathcal{G} = (\mathbb{G}, q, g)$ be an elliptic curve group. Choose $x \xleftarrow{R} \mathbb{Z}_q^*$ and set $\mathsf{sk} \leftarrow x$ and $\mathsf{pk} \leftarrow g^x$ and return $(\mathsf{sk}, \mathsf{pk})$.

$\underline{\mathsf{Sign}_{\mathsf{ECDSA}}(\mathsf{sk}, m)}$: Parse sk as x

1. choose $k \xleftarrow{R} \mathbb{Z}_q^*$
2. compute $R \leftarrow g^k$
3. let $r \leftarrow R_x \pmod q$ and if $r = 0$ goto step 1
4. let $s \leftarrow k^{-1}(H'(m) + rx) \pmod q$ and if $s = 0$ goto step 1
5. return $\sigma \leftarrow (r, s)$

$\underline{\mathsf{Verify}_{\mathsf{ECDSA}}(\mathsf{pk}, m, \sigma)}$: Parse σ as (r, s)

1. If $r = 0 \ \vee \ s = 0$ return 0
2. let $z \leftarrow H'(m)$ and $w \leftarrow s^{-1} \pmod q$
3. let $u_1 \leftarrow zw \pmod q$ and $u_2 \leftarrow rw \pmod q$
4. let $R \leftarrow g^{u_1} \cdot \mathsf{pk}^{u_2}$
5. if $R_x = r \pmod q$ return 1 and return 0 otherwise

Scheme 1: ECDSA signature scheme.

Brown [Bro02,Bro05] provides an analysis of standard ECDSA in the generic group model [Sho97], which quite leaves some open questions (cf. [FKP16] for a discussion why such a proof is problematic for ECDSA). The most recent work on the security of ECDSA from Fersch et al. [FKP16] avoids the generic group model and proves EUF-CMA security of ECDSA in the bijective random oracle model (ROM). We want to emphasize that we do not require details of any technique to prove security of ECDSA in this paper, as we will make a black-box reduction to EUF-CMA security of ECDSA.

5 Double-Authentication-Preventing Signatures

5.1 Formal Model

For double-authentication-preventing signatures (DAPS), we have a signature scheme on a message space $\mathcal{M} = \mathsf{A} \times \mathsf{P}$ of messages $m = (a, p)$ consisting of an address a and a payload p. The signature scheme is extended with a fourth algorithm Ex that extracts the secret key from signatures on two colliding messages. Before we can present the formal definition of DAPS we need to define the term colliding messages.

Definition 3 (Colliding Messages). *We call two messages $m_1 = (a_1, p_1)$ and $m_2 = (a_2, p_2)$ colliding if $a_1 = a_2$, but $p_1 \neq p_2$.*

Below, we now formally introduce DAPS following [PS14,PS17].

Definition 4 (DAPS). *A double-authentication-preventing signature scheme* DAPS *is a tuple* $(\mathsf{KGen}_\mathsf{D}, \mathsf{Sign}_\mathsf{D}, \mathsf{Verify}_\mathsf{D}, \mathsf{Ex}_\mathsf{D})$ *of PPT algorithms, which are defined as follows:*

$\mathsf{KGen}_\mathsf{D}(\kappa)$: *This algorithm takes a security parameter κ as input and outputs a secret (signing) key sk_D and a public (verification) key pk_D with associated message space $\mathcal{M}$ (we may omit to make the message space $\mathcal{M}$ explicit).*

$\mathsf{Sign}_\mathsf{D}(\mathsf{sk}_\mathsf{D}, m)$: *This algorithm takes a secret key* sk_D *and a message* $m \in \mathcal{M}$ *as input and outputs a signature* σ.

$\mathsf{Verify}_\mathsf{D}(\mathsf{pk}_\mathsf{D}, m, \sigma)$: *This algorithm takes a public key* pk_D*, a message* $m \in \mathcal{M}$ *and a signature* σ *as input and outputs a bit* $b \in \{0, 1\}$.

$\mathsf{Ex}_\mathsf{D}(\mathsf{pk}_\mathsf{D}, m_1, m_2, \sigma_1, \sigma_2)$: *This algorithm takes a public key* pk_D*, two colliding messages* m_1 *and* m_2 *and signatures* σ_1 *for* m_1 *and* σ_2 *for* m_2 *as inputs and outputs a secret key* sk_D.

Note that the algorithms KGen_D, Sign_D, and $\mathsf{Verify}_\mathsf{D}$ match the definition of the algorithms of a conventional signature scheme. For DAPS one requires a restricted but otherwise standard notion of unforgeability [PS14,PS17], where adversaries can adaptively query signatures for messages but only on distinct addresses. Figure 2 details the unforgeability security experiment.

Definition 5 (EUF-CMA [PS14])**.** *A DAPS scheme is* EUF-CMA *secure, if for all PPT adversaries* $\mathcal{A}$ *there is a negligible function* $\varepsilon(\cdot)$ *such that*

$$\Pr\left[\mathbf{Exp}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{A},\mathsf{DAPS}}(\kappa) = 1\right] \leq \varepsilon(\kappa),$$

where the corresponding experiment is depicted in Figure 2.

$\mathbf{Exp}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{A},\mathsf{DAPS}}(\kappa)$:
  $(\mathsf{sk}_\mathsf{D}, \mathsf{pk}_\mathsf{D}) \leftarrow \mathsf{KGen}_\mathsf{D}(1^\kappa)$
  $\mathcal{Q} \leftarrow \emptyset$, $\mathcal{R} \leftarrow \emptyset$
  $(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}'_\mathsf{D}(\mathsf{sk}_\mathsf{D}, \cdot)}(\mathsf{pk}_\Sigma)$
    where oracle $\mathsf{Sign}'_\mathsf{D}$ on input m:
      $(a, p) \leftarrow m$
      if $a \in \mathcal{R}$, return $\perp$
      $\sigma \leftarrow \mathsf{Sign}_\mathsf{D}(\mathsf{sk}_\mathsf{D}, m)$
      $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{m\}$, $\mathcal{R} \leftarrow \mathcal{R} \cup \{a\}$
      return σ
  return 1, if $\mathsf{Verify}_\mathsf{D}(\mathsf{pk}_\mathsf{D}, m^*, \sigma^*) = 1 \ \wedge \ m^* \notin \mathcal{Q}$
  return 0

Fig. 2: EUF-CMA **security for** DAPS**.**

The interesting property of a DAPS scheme is the notion of double-signature extractability (DSE). It requires that whenever one obtains signatures on two colliding messages, one should be able to extract the signing key using the extraction algorithm Ex_D. We give the security game in Figure 3, where we consider the conventional notion, denoted as DSE, which requires extraction to work if the key pair has been generated honestly. In this game, the adversary is given a key pair and outputs two colliding messages and corresponding signatures. The adversary wins the game if the key produced by Ex_D is different from the signing key although extraction should have succeeded, i.e, the messages were colliding and their signatures were valid.

Definition 6 (DSE [PS14]). *A DAPS scheme provides double-signature extraction (*DSE*), if for all PPT adversaries $\mathcal{A}$ there is a negligible function $\varepsilon(\cdot)$ such that*

$$\Pr\left[\mathbf{Exp}^{\mathsf{DSE}}_{\mathcal{A},\mathsf{DAPS}}(\kappa) = 1\right] \leq \varepsilon(\kappa),$$

where the corresponding experiment is depicted in Figure 3.

$\mathbf{Exp}^{\mathsf{DSE}}_{\mathcal{A},\mathsf{DAPS}}(\kappa)$:
$(\mathsf{sk}_\mathsf{D}, \mathsf{pk}_\mathsf{D}) \leftarrow \mathsf{KGen}_\mathsf{D}(1^\kappa)$
$(m_1, m_2, \sigma_1, \sigma_2) \leftarrow \mathcal{A}(\mathsf{sk}_\mathsf{D}, \mathsf{pk}_\mathsf{D})$
return 0, if m_1 and m_2 are not colliding
$v_i \leftarrow \mathsf{Verify}_\mathsf{D}(\mathsf{pk}_\mathsf{D}, m_i, \sigma_i)$ for $i \in [2]$
return 0, if $v_1 = 0$ or $v_2 = 0$
$\mathsf{sk}'_\mathsf{D} \leftarrow \mathsf{Ex}_\mathsf{D}(\mathsf{pk}_\mathsf{D}, m_1, m_2, \sigma_1, \sigma_2)$
return 1, if $\mathsf{sk}'_\mathsf{D} \neq \mathsf{sk}_\mathsf{D}$
return 0

Fig. 3: DSE **security for** DAPS**.**

In Appendix C we recall the strong variant of extractability under malicious keys (denoted as DSE*), where the adversary is allowed to generate the key arbitrarily. The DSE* notion is very interesting from a theoretical perspective, but no efficient DAPS construction, including ours, can achieve this notion so far. However, as we will show in Section 6.6 our, constructions besides DSE* satisfy a weaker notion under malicious keys introduced in this paper.

5.2 Existing DAPS Constructions

Poettering and Stebila [PS14,PS17] present the first ever DAPS construction in a factoring-based setting, where a signature contains $n+1$ elements in a group $\mathbb{Z}_N^*$ with n being the length of the output of a cryptographic hash function and N is an RSA modulus. At a security level of 128 bit (a 2048-bit RSA modulus and 256-bit hash), a signature contains > 250 group elements yielding a signature size of > 64 KB and signing as well as verification times much higher than standard signatures. Ruffing, Kate and Schroeder in [RKS15] introduced the notion of accountable assertions (AS), a weaker primitive than DAPS, and present one AS that also is a DAPS (termed RKS). The RKS construction is based on Merkle tress and chameleon hash functions in the discrete logarithm setting. Signing and verification are much more efficient than within PS, but signature sizes are still in the order of PS. Very recently, Bellare, Poettering and Stebila [BPS17] proposed new factoring-based DAPS from trapdoor identification-schemes using an adaption and extension of a transform from [BPS16]. Their two transforms applied to the Guillou-Quisquater (GQ) [GQ88] and Micali-Reyzin (MR) [MR02] identification scheme yield signing and verification times as well as signature

sizes comparable (or slightly above) standard RSA signatures. In a concurrent and independent work Boneh et al. [BKN17] propose constructions of DAPS from lattices. They consider DAPS as a special case of what they call predicate-authentication-preventing signatures (PAPS). In PAPS one considers a k-ary predicate on the message space and given any k valid signatures that satisfy the predicate reveal the signing key. Consequently, DAPS are PAPS for a specific 2-ary predicate and what we call N-times-authentication-preventing signatures (NAPS) is denoted as k-way DAPS in their work.

Unfortunately, as it is clear from the discussion, none of these DAPS schemes relies on widely used signature schemes such as RSA or (EC)DSA signatures. It is also important to mention that all these constructions only provide the extractability notion under honestly generated keys (DSE)[10]. We now present our DAPS in the next section and defer a detailed comparison of existing DAPS and ours to Section 6.10.

6 Short DAPS in the DL Setting

In this section we present our generic DAPS constructions from any discrete logarithm-based EUF-CMA secure signature scheme and in particular provide an instantiation with ECDSA signatures. As already mentioned, we thereby will be as non-invasive as possible in constructing DAPS "around" existing signatures without modifying the setting, e.g., groups, that are used by the respective schemes.

6.1 Intuition of Our Approach

As already mentioned in Section 3, our generic approach to construct DAPS is based on the idea of combining a signature scheme with a verifiable secret sharing scheme and in every signature include a share (specific to the address) of the secret signing key. Consequently, signing two different payloads with respect to the same address within the DAPS allows to extract the signing key of the underlying signature scheme.

Before presenting our construction paradigm and instantiations of DAPS, we introduce verifiable secret sharing in Section 6.2, ElGamal encryption in Section 6.3 and non-interactive zero-knowledge proofs from Σ-protocols (and a standard proof for the language of DDH tuples) in Section 6.4.

6.2 Verifiable Secret Sharing

Shamir's (k, ℓ)-threshold secret sharing [Sha79] allows to information-theoretically share a secret s among ℓ parties such that whenever k evaluations of the polyno-

[10] To be precise, in the initial work [PS14,PS17] the authors could tweak their construction to provide DSE* at the cost of adding quite expensive non-interactive zero-knowledge proofs to show that the public key is a well-formed Blum integer. But this would make their already rather impractical constructions with signature sizes $>$ 64 KB only more impractical.

mial (shares) are given, reconstruction of s is possible, but as long as only $k-1$ shares are available the secret s is information-theoretically hidden. Let s be the constant term of an otherwise randomly chosen $k-1$ degree polynomial

$$f(X) = \rho_{k-1}X^{k-1} + \cdots + \rho_1 X + s$$

over a prime field $\mathbb{Z}_q$. A share is computed as $f(i)$ for party i, $1 \leq i \leq \ell$. Let $\mathcal{S}$ be any set of cardinality at least k of these ℓ shares and let us denote the set of indices corresponding to shares in $\mathcal{S}$ by $I_{\mathcal{S}}$. Using Lagrange interpolation one can compute $s = f(0)$ as

$$s = \sum_{j \in I_{\mathcal{S}}} \lambda_j f(j) \text{ whereas } \lambda_j = \prod_{i \in I_{\mathcal{S}} \setminus \{j\}} \frac{j}{j-i} \ .$$

Now, we discuss a well known technique due to Feldman [Fel87] to make Shamir's secret sharing verifiable, by relaxing the otherwise information-theoretic secrecy to be only computational. The basic idea is to allow the use of a one-way homomorphism and in particular let us use a group $\mathcal{G} = (\mathbb{G}, q, g)$. To enable verifiability one publishes the sequence $(g^{\rho_{k-1}}, \ldots, g^{\rho_1}, g^{\rho_0})$ with $g^{\rho_0} = g^s$ and when given a share $f(i)$, everyone can non-interactively verify whether the share is correct by checking

$$g^{f(i)} = \prod_{j=0}^{k-1} (g^{\rho_j})^{i^j} .$$

Clearly, secrecy of s is only guaranteed if it has high min-entropy, as guesses can efficiently be verified.

6.3 ElGamal Encryption

Before presenting ElGamal encryption [EG85], let us define an encryption scheme first.

Definition 7 (Public Key Encryption Scheme). *A public key encryption scheme Ω is a triple* $(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *of PPT algorithms such that:*

$\mathsf{KGen}(1^\kappa)$: *This algorithm on input security parameter κ outputs the secret and public key* $(\mathsf{sk}, \mathsf{pk})$ *(the public key* pk *implicitly defines the message space $\mathcal{M}$).*

$\mathsf{Enc}(\mathsf{pk}, m)$: *This algorithm input the public key* pk*, and the message $m \in \mathcal{M}$ and outputs a ciphertext C.*

$\mathsf{Dec}(\mathsf{sk}, C)$: *This algorithm on input a secret key* sk *and a ciphertext C outputs a message $m \in \mathcal{M} \cup \{\perp\}$.*

We say that an encryption scheme Ω is perfectly correct if for all $\kappa \in \mathbb{N}$, for all $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(1^\kappa)$ and for all $m \in \mathcal{M}$ it holds that

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m)) = m\right] = 1.$$

IND-CPA security requires that an adversary $\mathcal{A}$ cannot decide which message is actually contained in a ciphertext C even when allowed to choose two challenge messages m_0 and m_1. We formally define IND-CPA security in Appendix D.

The ElGamal encryption scheme is multiplicatively homomorphic and IND-CPA secure under the k-LIN assumption in $\mathcal{G}$. We briefly present the popular ElGamal encryption scheme [EG85] in a group $\mathcal{G} = (\mathbb{G}, q, g)$ where the 1-LIN (DDH) assumption holds. The key generation algorithm KGen on input κ generates a group $\mathcal{G} = (\mathbb{G}, q, g)$ of prime order q of size κ bits and sets $sk := x \xleftarrow{R} \mathbb{Z}_q$ and $\mathsf{pk} := g^x$. To encrypt a message $m \in \mathbb{G}$, Enc samples $r \xleftarrow{R} \mathbb{Z}_q$ and computes the ciphertext $(C_1, C_2) := (g^r, m \cdot \mathsf{pk}^r)$. Finally, the decryption algorithm Dec given sk and ciphertext (C_1, C_2) outputs $C_2 \cdot C_1^{-\mathsf{sk}}$.

When setting $k = 2$ instead of $k = 1$ one obtains ElGamal under the 2-LIN (DLIN) assumption [BBS04] (termed linear ElGamal). It has the benefit that it can be instantiated in groups where the DDH assumption does not hold, e.g., in certain pairing-friendly elliptic curve or Schnorr groups. We recall both assumptions in Appendix A for the convenience of the reader. In the remainder of this paper we use the DDH instantiation of ElGamal, but we stress that all our protocols can be based on linear ElGamal as well.

6.4 Σ-Protocols

Let $L \subseteq \mathsf{X}$ be an **NP**-language with associated witness relation R so that $L = \{x \mid \exists w : R(x, w) = 1\}$. A Σ-protocol for language L is an interactive three move protocol between a prover and a verifier, where the prover proves knowledge of a witness w to the statement $x \in L$. We recall the formal definition of Σ-protocols in Appendix E.

Σ-protocol for DDH-tuples. Σ-protocols for proving that elements (g_1, g_2, u_1, u_2) in a prime order group $\mathcal{G}$ form a DDH tuple are well known and established [CP92]. We define the corresponding language via relation R

$$((g_1, g_2, u_1, u_2), w) \in R \Leftrightarrow g_1^w = u_1 \wedge g_2^w = u_2 \tag{1}$$

as witness relation. In Scheme 2 we briefly recall a classical Σ-protocol for R.

Lemma 1. *The protocol in Scheme 2 represents a Σ-protocol for the relation R in* (1).

We omit the proof of Lemma 1 as it is a well known result and straightforward.

Non-Interactive ZK Proof Systems (NIZK). We recall a standard definition of non-interactive zero-knowledge proof systems. Let L be an **NP**-language with witness relation R as above.

Definition 8 (Non-Interactive Zero-Knowledge Proof System). *A non-interactive proof system Π is a tuple of algorithms $(\mathsf{Setup}_\Pi, \mathsf{Proof}_\Pi, \mathsf{Verify}_\Pi)$, which are defined as follows:*

$\mathsf{Setup}_\Pi(1^\kappa)$: *This algorithm takes a security parameter κ as input, and outputs a common reference string* crs.

Let $\mathcal{G} = (\mathbb{G}, q, g)$ and let $g_1, g_2, u_1, u_2 \in \mathbb{G}$.

Prover		**Verifier**
$(u_1, u_2, k = \log_{g_i} u_i)$		(u_1, u_2)
$r \xleftarrow{R} \mathbb{Z}_q^*,\ r_i \leftarrow g_i^r$	$\xrightarrow{r_1, r_2}$	
	$\xleftarrow{c}$	$c \xleftarrow{R} \mathbb{Z}_q$
$s \leftarrow r + kc$	$\xrightarrow{s}$	accept iff $\forall i : g_i^s = r_i u_i^c$

Scheme 2: Σ-protocol for proving that (g_1, g_2, u_1, u_2) forms a DDH-tuple.

$\mathsf{Proof}_\Pi(\mathsf{crs}, x, w)$: *This algorithm takes a common reference string* crs*, a statement* x*, and a witness* w *as input, and outputs a proof* π*.*

$\mathsf{Verify}_\Pi(\mathsf{crs}, x, \pi)$: *This algorithm takes a common reference string* crs*, a statement* x*, and a proof* π *as input, and outputs a bit* $b \in \{0, 1\}$*.*

From a non-interactive zero-knowledge proof system we require *completeness*, *soundness* and *adaptive zero-knowledge*. In Appendix F we recall formal definitions of those properties.

NIZK from Σ-protocols. One can obtain a non-interactive proof system with the above properties from any Σ-protocol by applying the Fiat-Shamir transform [FS86] where the min-entropy μ of the commitment a sent in the first message of the Σ-protocol is so that $2^{-\mu}$ is negligible in the security parameter κ and its challenge space C is exponentially large in the security parameter. Essentially, the transform removes the interaction between the prover and the verifier by using a hash function H (modelled as a random oracle) to obtain the challenge. That is, the algorithm $\mathsf{Challenge}$ obtains the challenge as $H(\mathsf{a}, x)$. More formally, $\mathsf{Setup}_\Pi(1^\kappa)$ fixes a hash function $H : \mathsf{A} \times \mathsf{X} \to \mathsf{C}$, sets $\mathsf{crs} \leftarrow (\kappa, H)$ and returns crs. The algorithms Proof_Π and Verify_Π are defined as follows:

$\mathsf{Proof}_\Pi(\mathsf{crs}, x, w)$: Start P on $(1^\kappa, x, w)$, obtain the first message a, answer with $\mathsf{c} \leftarrow H(\mathsf{a}, x)$. Finally obtain s and return $\pi \leftarrow (\mathsf{a}, \mathsf{s})$.

$\mathsf{Verify}_\Pi(\mathsf{crs}, x, \pi)$: Parse π as (a, s). Start V on $(1^\kappa, x)$ and send a as first message to the verifier. When V outputs c, reply with s and output 1 if V accepts and 0 otherwise.

Combining [FKMV12, Thm. 1, Thm. 2, Thm. 3, Prop. 1] (among others) shows that a so-obtained proof system is complete, sound, adaptively zero-knowledge, if the underlying Σ-protocol is special sound and the commitments sent in the first move are unconditionally binding. When referring to the NIZK proof system obtained from Scheme 2, we denote the algorithms as $(\mathsf{Setup}_{\mathsf{DDH}}, \mathsf{Proof}_{\mathsf{DDH}}, \mathsf{Verify}_{\mathsf{DDH}})$.

A note on the CRS. We stress that for the sake of generality the output of $\mathsf{Setup}_{\mathsf{DDH}}$ is denoted as crs. However, as we exclusively use NIZK from Σ-

protocols in our DAPS, we do not require a trusted setup and crs is just a description of the hash function which is globally fixed, e.g., to SHA-256 or SHA-3.

6.5 Generic DAPS in the Discrete Logarithm Setting

In the following, let Σ be a signature scheme in the discrete logarithm setting, which is from the class C of signature schemes where the public key is the image of the secret key under a group homomorphism. In the discrete logarithm setting this means that the secret key x is an element from $\mathbb{Z}_q$ and the public key is its image g^x in the group. We stress that the class C essentially covers any scheme in the discrete logarithm setting we can think of, and, in particular schemes like Schnorr, (EC)DSA, or EdDSA. We subsequently present our protocols based on ElGamal in the DDH setting and recall that when the DDH is not hard in the respective group, we can easily instantiate all our protocols on linear ElGamal under the DLIN assumption (cf. Section 6.3)

Our approach is as follows. First we generate an ElGamal encryption key-pair (x_E, pk_E). Then, for each possible address i we choose $\rho_i \in \mathbb{Z}_q$ uniformly at random and additionally include an encryption $(C_{i,1}, C_{i,2})$ of g^{ρ_i} as well as pk_E in the DAPS public key. The secret key additionally includes the values ρ_i and the randomness $r_i \in \mathbb{Z}_q$ used upon encrypting ρ_i. When signing a message $m = (i, p) \in [n] \times \mathbb{Z}_q^*$, we obtain a signature from Σ, and extend it with a secret share of sk_Σ: we let $f_i(X) = \rho_i X + \mathsf{sk}_\Sigma$ and include $z = f_i(p)$ in the signature. When signing two colliding messages, we obtain two shares for the same degree 1 polynomial f_i and hence can re-construct sk_Σ. To ensure the correct computation of z, each signature is extended by a proof for the following relation R, which is essentially a proof for a verifiable secret sharing using ElGamal encryption for the coefficient of the non-constant term:

$$((g, \mathsf{pk}_E, C_{i,1}, C'_{i,2}), r) \in R \Leftrightarrow C_{i,1} = g^r \wedge C'_{i,2} = \mathsf{pk}_E^r$$

where $C'_{i,2} = C_{i,2} \cdot (\mathsf{pk}_\Sigma \cdot g^{-z})^{1/p}$.

Observe that the extraction algorithm, when applied to colliding signatures, reveals the secret signing key sk_Σ, but none of the r_i and ρ_i. However, DAPS extraction needs to recover the full secret key. To achieve this, we can apply the same trick as used in [BPS17] and put a symmetric encryption $(r_i, \rho_i)_{i\in[n]} \oplus H(\mathsf{sk}_\Sigma)$ of $(r_i, \rho_i)_{i\in[n]}$ under sk_Σ in the public key, where $H : \{0,1\}^* \to \{0,1\}^\ell$ is a hash function treated as random oracle and ℓ is the length of the bit encoding of all $(r_i, \rho_i)_{i\in[n]}$.

We note that in our construction KGen_D takes the number of addresses as explicit argument. The scheme is also presented using $\mathbb{Z}_q^*$ as payload space, but it can be extended to an arbitrary payload space using the standard hash-then-sign technique.

Theorem 1. *If Σ is from class C instantiated in group $\mathcal{G}$ and $\mathsf{EUF\text{-}CMA}$-secure, DDH is hard relative to $\mathcal{G}$ and the NIZK proof system is adaptive zero-knowledge, then Σ-DAPS is an $\mathsf{EUF\text{-}CMA}$-secure DAPS in the random oracle model.*

$\underline{\mathsf{KGen}_\mathsf{D}(1^\kappa, n)}$: Let $(\mathsf{sk}_\Sigma, \mathsf{pk}_\Sigma) \leftarrow \mathsf{KGen}_\Sigma(1^\kappa)$ with $\mathcal{G} = (\mathbb{G}, q, g)$. Let $x_E \xleftarrow{R} \mathbb{Z}_q^*$ and $\mathsf{pk}_E \leftarrow g^{x_E}$. Let $(\rho_i)_{i\in[n]} \xleftarrow{R} (\mathbb{Z}_q^*)^n$ and $(r_i)_{i\in[n]} \xleftarrow{R} (\mathbb{Z}_q^*)^n$. Set $(C_i)_{i\in[n]} \leftarrow (g^{r_i}, \mathsf{pk}_E^{r_i} g^{\rho_i})_{i\in[n]}$. Let $\mathsf{crs} \leftarrow \mathsf{Setup}_{\mathsf{DDH}}(1^\kappa)$. Let $H : \{0,1\}^* \to \{0,1\}^\ell$ be a hash function where ℓ is the length of the bit encoding of $(r_i, \rho_i)_{i\in[n]}$ and set $Y \leftarrow (r_i, \rho_i)_{i\in[n]} \oplus H(\mathsf{sk}_\Sigma)$. Let $\mathsf{sk} \leftarrow (\mathsf{sk}_\Sigma, (r_i, \rho_i)_{i\in[n]})$ and $\mathsf{pk} \leftarrow (\mathsf{pk}_\Sigma, \mathsf{pk}_E, (C_i)_{i\in[n]}, \mathsf{crs}, Y)$ and return $(\mathsf{sk}, \mathsf{pk})$.

$\underline{\mathsf{Sign}_\mathsf{D}(\mathsf{sk}, m)}$: Parse sk as $(\mathsf{sk}_\Sigma, (r_i, \rho_i)_{i\in[n]})$. Parse m as (i, p) with $i \le n$.

1. Let $\sigma \leftarrow \mathsf{Sign}_\Sigma(\mathsf{sk}_\Sigma, m)$
2. let $z \leftarrow \rho_i p + \mathsf{sk}_\Sigma$
3. let $C_2' \leftarrow C_{i,2} \cdot (\mathsf{pk}_\Sigma \cdot g^{-z})^{\frac{1}{p}}$
4. $\pi \leftarrow \mathsf{Proof}_{\mathsf{DDH}}(\mathsf{crs}, (g, \mathsf{pk}_E, C_{i,1}, C_2'), r_i)$
5. return (σ, z, π)

$\underline{\mathsf{Verify}_\mathsf{D}(\mathsf{pk}, m, \sigma)}$: Parse pk as $(\mathsf{pk}_\Sigma, \mathsf{pk}_E, (C_i)_{i\in[n]}, \mathsf{crs}, \cdot)$, m as (i, p) with $i \le n$, and σ as (σ', z, π).

1. If $\mathsf{Verify}_\Sigma(\mathsf{pk}_\Sigma, m, \sigma') = 0$, return 0
2. let $C_2' \leftarrow C_{i,2} \cdot (\mathsf{pk}_\Sigma \cdot g^{-z})^{\frac{1}{p}}$
3. return $\mathsf{Verify}_{\mathsf{DDH}}(\mathsf{crs}, (g, \mathsf{pk}_E, C_{i,1}, C_2'), \pi)$

$\underline{\mathsf{Ex}_\mathsf{D}(\mathsf{pk}, m_1, m_2, \sigma_1, \sigma_2)}$: Parse σ_i as $(\cdot, z_i, \cdot)$, m_i as (a_i, p_i) and pk as $(\cdot, \cdot, \cdot, \cdot, Y)$.

1. If m_1 and m_2 are not colliding, return $\perp$
2. if $\mathsf{Verify}_\mathsf{D}(\mathsf{pk}, m_i, \sigma_i) = 0$ for any i, return $\perp$
3. let $\mathsf{sk}_\Sigma \leftarrow z_1 \frac{p_2}{p_2 - p_1} + z_2 \frac{p_1}{p_1 - p_2}$
4. let $(r_i, \rho_i)_{i\in[n]} \leftarrow Y \oplus H(\mathsf{sk}_\Sigma)$
5. return $(\mathsf{sk}_\Sigma, (r_i, \rho_i)_{i\in[n]})$

Scheme 3: Σ-DAPS: Generic DAPS from any signature scheme Σ from class C.

Proof. We prove this theorem using a sequence of games. We denote the winning event of game G_i as S_i. We use gray textboxes to indicate changes within algorithms.

Game 0: The original EUF-CMA game.

Game 1: As before, but we modify KGen_D and Sign_D: Let $\mathcal{S}_{1,\mathsf{DDH}}$ and $\mathcal{S}_{2,\mathsf{DDH}}$ be the setup and simulation algorithms of the simulator for the NIZK proof system.

$\underline{\mathsf{KGen}_\mathsf{D}(1^\kappa, n)}$: As before, but let

$$(\mathsf{crs}, \tau) \leftarrow \mathcal{S}_{1,\mathsf{DDH}}(1^\kappa) \text{ and store } \tau.$$

$\underline{\mathsf{Sign}_\mathsf{D}(\mathsf{sk}, m)}$: As before, but let

$$\pi \leftarrow \mathcal{S}_{2,\mathsf{DDH}}(\mathsf{crs}, \tau, (g, \mathsf{pk}_E, C_1', C_2')).$$

Transition $0 \to 1$: Game 0 and Game 1 are indistinguishable under adaptive zero-knowledge of the proof system, i.e. $|\Pr[S_0] - \Pr[S_1]| \le \varepsilon_z(\kappa)$.

Game 2: As Game 1, but we run the following modified KGen_D algorithm $\mathsf{KGen}_\mathsf{D}'$:

$\underline{\mathsf{KGen}_{\mathsf{D}}(1^\kappa, n)'}$: Let $(\mathsf{sk}_{\mathbf{\Sigma}}, \mathsf{pk}_{\mathbf{\Sigma}}) \leftarrow \mathsf{KGen}_{\mathbf{\Sigma}}(1^\kappa)$ with $\mathcal{G} = (\mathbb{G}, q, g)$. Let $x_E \xleftarrow{R} \mathbb{Z}_q^*$ and $\mathsf{pk}_E \leftarrow g^{x_E}$. Let $(\rho_i)_{i\in[n]} \xleftarrow{R} (\mathbb{Z}_q^*)^n$ and $(r_i)_{i\in[n]} \xleftarrow{R} (\mathbb{Z}_q^*)^n$. Set ciphertexts as $\boxed{(C_i)_{i\in[n]} \xleftarrow{R} (\mathbb{G}^2)^n}$. Let $\mathsf{crs} \leftarrow \mathsf{Setup}_{\mathsf{DDH}}(1^\kappa)$. Also set $Y \leftarrow (r_i, \rho_i)_{i\in[n]} \oplus H(\mathsf{sk}_{\mathbf{\Sigma}})$. Let $\mathsf{sk} \leftarrow (\mathsf{sk}_{\mathbf{\Sigma}}, (r_i, \rho_i)_{i\in[n]})$ and $\mathsf{pk} \leftarrow (\mathsf{pk}_{\mathbf{\Sigma}}, \mathsf{pk}_E, (C_i)_{i\in[n]}, \mathsf{crs}, Y)$ and return $(\mathsf{sk}_E, \mathsf{pk}_E)$.

Transition $1 \to 2$: We claim that the probability to distinguish between Game 1 and Game 2 is bounded by $|\Pr[S_1] - \Pr[S_2]| \leq n \cdot \varepsilon_{\mathsf{DDH}}(\kappa)$. To see this assume n additional hybrids, where in each hybrid H_j with $1 \leq j \leq n$ we replace ciphertext C_j by a random value. Then the distinguishing probability of two consecutive hybrids is bounded by $\varepsilon_{\mathsf{DDH}}(\kappa)$. In particular, assume we obtain a DDH instance $(g^{u_1}, g^{u_2}, g^{u_3})$ relative to $\mathbb{G}$ and set $\mathsf{pk}_E \leftarrow g^{u_2}$. Then in hybrid H_j we choose all C_i where $i < j$ random (as they were also already random in the previous hybrid). For C_j, we compute $C_j \leftarrow (g^{u_1}, g^{u_3} \cdot g_i^{\rho})$. Furthermore, for C_i where $i > j$, we choose $r_i \xleftarrow{R} \mathbb{Z}_q$ and set $C_i \leftarrow (g^{r_i}, (g^{u_2})^{r_i} \cdot g^{\rho_i})$. Then the validity of the DDH instance determines whether we sample from the distribution in Game 1 or Game 2, which proves that the distinguishing probability between two intermediate hybrids is bounded by $\varepsilon_{\mathsf{DDH}}(\kappa)$. Taking all n transitions together, this yields $n \cdot \varepsilon_{\mathsf{DDH}}(\kappa)$ which proves our initial claim.

Game 3: As Game 2, but in $\mathsf{KGen}_{\mathsf{D}}$ we choose $\boxed{Y \xleftarrow{R} \{0,1\}^{|\mathbb{Z}_q|\cdot 2\cdot n}}$ and in $\mathsf{Sign}_{\mathsf{D}}$ we choose $\boxed{z \xleftarrow{R} \mathbb{Z}_q}$, and abort whenever the adversary comes up with a valid forgery or queries $H(\cdot)$ on a value sk_{Σ} which corresponds to pk_{Σ}.

Transition $2 \to 3$: We denote the event that we abort by E. Both, Game 2 and Game 3 proceed identically unless E happens, i.e., $|\Pr[S_2] - \Pr[S_3]| \leq \Pr[E]$. First observe that if E does not happen both games in fact proceed identically: the adversary has never queried $H(\mathsf{sk}_{\Sigma})$ so that $\mathsf{sk}_{\mathbf{\Sigma}}$ is consistent with $\mathsf{pk}_{\mathbf{\Sigma}}$, which implies that the adversary does not notice any distribution change with respect to Y and z. On the other hand, whenever E happens in Game 3, we can build an EUF-CMA forger for $\mathbf{\Sigma}$. To do so, we engage with an EUF-CMA challenger for $\mathbf{\Sigma}$ and obtain σ from the oracle provided by the challenger (we no longer require sk_{Σ} anywhere else). If the adversary queries $H(\mathsf{sk}_{\Sigma})$ so that $\mathsf{sk}_{\mathbf{\Sigma}}$ is consistent with $\mathsf{pk}_{\mathbf{\Sigma}}$ we choose a random message m from the message space of $\mathbf{\Sigma}$ which we never queried to the challenger, compute a signature and return the message signature pair as a forgery. Likewise, if the adversary outputs a forgery, we can output $(\sigma', (i, m))$ as a valid EUF-CMA forgery.

Taking all together we obtain that E happens with exactly the same probability as an EUF-CMA forgery and we have $|\Pr[S_2] - \Pr[S_3]| \leq \varepsilon_{\mathsf{EUF\text{-}CMA}(\kappa)}$.

In the final game, the adversary can no longer win, i.e., $\Pr[S_3] = 0$. Taking all together, we have that $\Pr[S_0] \leq \varepsilon_z(\kappa) + n \cdot \varepsilon_{\mathsf{DDH}}(\kappa) + \varepsilon_{\mathsf{EUF-CMA}}(\kappa)$, which concludes the proof.

Theorem 2. *If the NIZK proof system is sound, then* $\mathbf{\Sigma}$-DAPS *provides* DSE *security.*

Proof. We prove this theorem using a sequence of games. We denote the winning event of game G_i as S_i. Let $m_1, m_2, \sigma_1, \sigma_2$ be the output of $\mathcal{A}$. For simplicity we write $m_i = (a, p_i)$, $\sigma_i = (\cdot, z_i, \pi_i)$, and $\mathsf{pk}_\mathsf{D} = (\mathsf{pk}_\mathbf{\Sigma}, \mathsf{pk}_E, (C_i)_{i\in[n]}, \mathsf{crs}, Y)$. We also let $C'_{i,2} \leftarrow C_{a,2} \cdot (\mathsf{pk}_\mathbf{\Sigma} \cdot g^{-z_i})^{\frac{1}{p_i}}$.

Game 0: The original DSE game.
Game 1: As before, but we abort if $C'_{1,2} \neq C'_{2,2}$.
Transition $0 \to 1$: Let E be the event that $C'_{1,2} \neq C'_{2,2}$. In this case we engage with a soundness challenger $\mathcal{C}$ of proof system and modify KGen_D as follows:

$\underline{\mathsf{KGen}_\mathsf{D}(1^\kappa, n)}$: Obtain $\boxed{\mathsf{crs}}$ from $\mathcal{C}$ and compute everything else honestly.

Once $\mathcal{A}$ outputs the two colliding messages and signatures, we have proofs attesting that both $(g, \mathsf{pk}_E, C_{a,1}, C'_{i,2})$ for $i \in [2]$ are DDH tuples, but, by the perfect correctness of ElGamal, at most one of them can be a DDH tuple, i.e., one of the words is not in the language. Hence we guess $b \xleftarrow{R} \{0,1\}$, and forward $(g, \mathsf{pk}_E, C_{a,1}, C'_{b,2}), \pi_b$ to $\mathcal{C}$. We guess the word breaking soundness of DDH with probability $1/2$. Hence $\Pr[E] \leq \frac{\varepsilon_s(\kappa)}{2}$ where ε_s is the soundness error of DDH.

Now (p_1, z_1) and (p_2, z_2) are secret shares of the same polynomial $f = \rho X + \mathsf{sk}_\mathbf{\Sigma}$. Hence x is uniquely determined via

$$\mathsf{sk}_\mathbf{\Sigma} = f(0) = z_1 \frac{p_2}{p_2 - p_1} + z_2 \frac{p_1}{p_1 - p_2}.$$

Since the key was set up honestly, Y and $\mathsf{sk}_\mathbf{\Sigma}$ now uniquely determine the remaining parts of the secret key. Thus $\Pr[S_1] = 0$ and in total $\Pr[S_0] \leq \varepsilon_s(\kappa)$, which concludes the proof.

6.6 Extraction of the Signing Key of $\mathbf{\Sigma}$

When considering constructions that extend conventional signature schemes to a DAPS, there is a gap between DSE and DSE^* notions and ensuring extraction of the $\mathbf{\Sigma}$ signing key. Recall, that these notions require to extract the complete DAPS secret key and no existing efficient DAPS scheme provides DSE^*. When the DAPS key consists of a $\mathbf{\Sigma}$ signing key, extraction of the signing key alone, however, already disincentivizes double-authentication for many applications, where this key is also used outside the context of DAPS. Hence we define two weaker double-signature extraction notions that cover extraction of the signing key of the underlying signature scheme for honestly and maliciously generated DAPS keys. The security games for weak double-signature extraction (wDSE) and weak double-signature extraction under malicious keys (wDSE^*) are depicted in Figure 4 and Figure 5.

Definition 9 ($T \in \{\mathsf{wDSE}, \mathsf{wDSE}^*\}$). *A DAPS scheme provides weak double-signature extraction ($T = \mathsf{wDSE}$) respectively weak double-signature extraction*

under malicious keys ($T = \mathsf{wDSE}^*$), if for all PPT adversaries $\mathcal{A}$ there is a negligible function $\varepsilon(\cdot)$ such that

$$\Pr\left[\mathbf{Exp}^{T}_{\mathcal{A},\mathsf{DAPS}}(\kappa) = 1\right] \leq \varepsilon(\kappa),$$

where the corresponding experiments are depicted in Figure 4 and Figure 5 respectively.

$\mathbf{Exp}^{\mathsf{wDSE}}_{\mathcal{A},\mathsf{DAPS}}(\kappa)$:
$(\mathsf{sk}_\mathsf{D}, \mathsf{pk}_\mathsf{D}) \leftarrow \mathsf{KGen}_\mathsf{D}(1^\kappa)$ with $\mathsf{sk}_\mathsf{D} = (\mathsf{sk}_\Sigma, \dots)$
$(m_1, m_2, \sigma_1, \sigma_2) \leftarrow \mathcal{A}(\mathsf{sk}_\mathsf{D}, \mathsf{pk}_\mathsf{D})$
return 0, if m_1 and m_2 are not colliding
$v_i \leftarrow \mathsf{Verify}_\mathsf{D}(\mathsf{pk}_\mathsf{D}, m_i, \sigma_i)$ for $i \in [2]$
return 0, if $v_1 = 0$ or $v_2 = 0$
$\mathsf{sk}'_\mathsf{D} \leftarrow \mathsf{Ex}_\mathsf{D}(\mathsf{pk}_\mathsf{D}, m_1, m_2, \sigma_1, \sigma_2)$ where $\mathsf{sk}'_\mathsf{D} = (\mathsf{sk}'_\Sigma, \dots)$
return 1, if $\mathsf{sk}'_\Sigma \neq \mathsf{sk}_\Sigma$
return 0

Fig. 4: wDSE security for DAPS.

$\mathbf{Exp}^{\mathsf{wDSE}^*}_{\mathcal{A},\mathsf{DAPS}}(\kappa)$:
$(\mathsf{pk}_\mathsf{D}, m_1, m_2, \sigma_1, \sigma_2) \leftarrow \mathcal{A}(1^\kappa)$ where $\mathsf{pk}_\mathsf{D} = (\mathsf{pk}_\Sigma, \dots)$
return 0, if m_1 and m_2 are not colliding
$v_i \leftarrow \mathsf{Verify}_\mathsf{D}(\mathsf{pk}_\mathsf{D}, m_i, \sigma_i)$ for $i \in [2]$
return 0, if $v_1 = 0$ or $v_2 = 0$
$\mathsf{sk}'_\mathsf{D} \leftarrow \mathsf{Ex}_\mathsf{D}(\mathsf{pk}_\mathsf{D}, m_1, m_2, \sigma_1, \sigma_2)$ where $\mathsf{sk}'_\mathsf{D} = (\mathsf{sk}'_\Sigma, \dots)$
return 1, if sk'_Σ is not the secret key corresponding to pk_Σ
return 0

Fig. 5: wDSE* security for DAPS.

Clearly, DSE and DSE* imply their weaker counterparts and wDSE* implies wDSE. We now show that our Σ-DAPS also provide wDSE* security and thus for the first time we have some reasonable extraction guarantees under adversarially generated keys for practical DAPS.

Recall that the crs of NIZK proof systems instantiated by applying the Fiat-Shamir transform to a Σ-protocol consists of a globally fixed hash function, e.g. SHA-256 or SHA-3. Consequently, this hash function can simply be part of the DAPS description, removed from the key generation and globally fixed. Now one can observe that the properties of the proof system do not require a trusted setup. So even when considering keys generated by the adversary, this observation and the perfect correctness of the encryption scheme ensure that

our DAPS construction guarantees the successful extraction of the signing key of the underlying signature scheme. We now give a sketch of the proof.

Theorem 3. *If the NIZK proof system is sound and instantiated by applying the Fiat-Shamir transform to the Σ-protocol in Scheme 2, then* Σ-DAPS *provides* wDSE* *security.*

Proof (Sketch). We observe that the only parameter which needs to be controlled by the simulator in the proof of Theorem 2 is the crs. Now, since there is no crs in Fiat-Shamir transformed Σ-protocols, wDSE* follows from this property, Transition $0 \to 1$ of Theorem 2, and the observation that sk_Σ is then uniquely determined by the two shares included in the signatures.

6.7 DAPS from ECDSA

As an example we give a concrete instantiation of our DAPS construction based on ECDSA, dubbed ECDSA-DAPS. The full scheme is presented in Scheme 4. Furthermore, we state the following corollaries.

Corollary 1. *If ECDSA is* EUF-CMA*-secure, and the NIZK proof system is adaptive zero-knowledge, then* ECDSA-DAPS *is an* EUF-CMA*-secure DAPS in the random oracle model.*

Corollary 2. *If the NIZK proof system is sound, then* ECDSA-DAPS *provides* DSE *security.*

Corollary 3. *If the NIZK proof system is sound and instantiated by applying the Fiat-Shamir transform to the Σ-protocol in Scheme 2, then* ECDSA-DAPS *provides* wDSE* *security.*

All three corollaries follow directly from the observation that ECDSA is included in the class C and Theorem 1, Theorem 2, and Theorem 3.

6.8 Further DAPS

Our technique to construct DAPS can also be applied to the Schnorr signature scheme (cf. Appendix B) and the finite-field variant DSA. In particular, the latter is straightforward given the construction of ECDSA-DAPS in Scheme 4 and for brevity we omit the scheme. Besides DSA and Schnorr, EdDSA [BDL+12] also belongs to the class C of signatures schemes and can be extended to a DAPS in the same way. Consequently, our DAPS construction can easily be instantiated with EdDSA and curves `ed25518` [Ber06] or `ed448` [Ham15]. Even more generally, our approach towards DAPS can generically be applied to any signature schemes in the discrete logarithm setting from class C. Straightforwardly, if the public key is a single group element and otherwise for any scheme having public keys $k > 1$ group elements one simply has to combine the signature scheme with k copies of our technique. Our approach might also be applied beyond discrete logarithm based schemes if the respective setting provides a suitable encryption scheme, verifiable secret sharing scheme for secret keys and a non-interactive proof system.

$\underline{\mathsf{KGen}_\mathsf{D}(1^\kappa, n)}$: Let $\mathcal{G} = (\mathbb{G}, q, g)$ and $H' : \{0,1\}^* \to \mathbb{Z}_q$ be a hash function mapping exactly to the order of the group. Let $\mathsf{sk}_\Sigma \xleftarrow{R} \mathbb{Z}_q^*$ and $x_E \xleftarrow{R} \mathbb{Z}_q^*$, and set $\mathsf{pk}_\Sigma \leftarrow g^{\mathsf{sk}_\Sigma}$ and $\mathsf{pk}_E \leftarrow g^{x_E}$. Let $(\rho_i)_{i\in[n]} \xleftarrow{R} (\mathbb{Z}_q^*)^n$ and $(r_i)_{i\in[n]} \xleftarrow{R} (\mathbb{Z}_q^*)^n$. Set $(C_i)_{i\in[n]} \leftarrow (g^{r_i}, \mathsf{pk}_E^{r_i} g^{\rho_i})_{i\in[n]}$. Let $\mathsf{crs} \leftarrow \mathsf{Setup}_{\mathsf{DDH}}(1^\kappa)$. Let $H : \{0,1\}^* \to \{0,1\}^\ell$ be a hash function where ℓ is the length of the bit encoding of $(r_i, \rho_i)_{i\in[n]}$ and set $Y \leftarrow (r_i, \rho_i)_{i\in[n]} \oplus H(\mathsf{sk}_\Sigma)$. Let $\mathsf{sk} \leftarrow (\mathsf{sk}_\Sigma, (r_i, \rho_i)_{i\in[n]})$ and $\mathsf{pk} \leftarrow (\mathsf{pk}_\Sigma, \mathsf{pk}_E, (C_i)_{i\in[n]}, \mathsf{crs}, Y)$ and return $(\mathsf{sk}, \mathsf{pk})$.

$\underline{\mathsf{Sign}_\mathsf{D}(\mathsf{sk}, m)}$: Parse sk as $(\mathsf{sk}_\Sigma, (r_i, \rho_i)_{i\in[n]})$. Parse m as (i, p) with $i \le n$.

1. Choose $k \xleftarrow{R} \mathbb{Z}_q^*$
2. compute $R \leftarrow g^k$
3. let $r \leftarrow R_x \pmod q$ and if $r = 0$ goto step 1
4. let $s \leftarrow k^{-1}(H'(m) + r\mathsf{sk}_\Sigma) \pmod q$ and if $s = 0$ goto step 1
5. let $z \leftarrow \rho_i p + \mathsf{sk}_\Sigma$
6. let $C_2' \leftarrow C_{i,2} \cdot (\mathsf{pk}_\Sigma \cdot g^{-z})^{\frac{1}{p}}$
7. $\pi \leftarrow \mathsf{Proof}_{\mathsf{DDH}}(\mathsf{crs}, (g, \mathsf{pk}_E, C_{i,1}, C_2'), r_i)$
8. return (r, s, z, π)

$\underline{\mathsf{Verify}_\mathsf{D}(\mathsf{pk}, m, \sigma)}$: Parse pk as $(\mathsf{pk}_\Sigma, \mathsf{pk}_E, (C_i)_{i\in[n]}, \mathsf{crs}, \cdot)$, m as (i, p) with $i \le n$, and σ as (r, s, z, π).

1. If $r = 0 \ \vee \ s = 0$ return 0
2. let $z \leftarrow H'(m)$ and $w \leftarrow s^{-1} \pmod q$
3. let $u_1 \leftarrow zw \pmod q$ and $u_2 \leftarrow rw \pmod q$
4. let $R \leftarrow g^{u_1} \cdot \mathsf{pk}_\Sigma^{u_2}$
5. if $R_x = r \pmod q$ return 1 and return 0 otherwise
6. let $C_2' \leftarrow C_{i,2} \cdot (\mathsf{pk}_\Sigma \cdot g^{-z})^{\frac{1}{p}}$
7. return $\mathsf{Verify}_{\mathsf{DDH}}(\mathsf{crs}, (g, \mathsf{pk}_E, C_{i,1}, C_2'), \pi)$

$\underline{\mathsf{Ex}_\mathsf{D}(\mathsf{pk}, m_1, m_2, \sigma_1, \sigma_2)}$: Parse σ_i as $(\cdot, z_i, \cdot)$, m_i as (a_i, p_i) and pk as $(\cdot, \cdot, \cdot, \cdot, Y)$.

1. If m_1 and m_2 are not colliding, return $\perp$
2. if $\mathsf{Verify}_\mathsf{D}(\mathsf{pk}, m_i, \sigma_i) = 0$ for any i, return $\perp$
3. let $\mathsf{sk}_\Sigma \leftarrow z_1 \frac{p_2}{p_2 - p_1} + z_2 \frac{p_1}{p_1 - p_2}$
4. let $(r_i, \rho_i)_{i\in[n]} \leftarrow Y \oplus H(\mathsf{sk}_\Sigma)$
5. return $(\mathsf{sk}_\Sigma, (r_i, \rho_i)_{i\in[n]})$

Scheme 4: ECDSA-DAPS**: DAPS from ECDSA.**

6.9 *N*-Times-Authentication-Preventing Signatures

Finally, we observe that our techniques can easily be generalized to what we call N-times-authentication-preventing signatures (NAPS). That is, signature schemes where creating N signatures with respect to the same address leaks the secret key while they are unforgeable as long as there are $< N$ signatures for every address. While an extension of the formal model is straightforward and therefore omitted, we subsequently sketch the construction.

Essentially, instead of computing z by evaluating a degree 1 polynomial $f_i(X) = \rho_i X + \mathsf{sk}_\Sigma \in \mathbb{Z}_q[X]$ associated to address i at the payload p, we can generalize our approach to a degree $N-1$ polynomial $f_i(X) = \mathsf{sk}_\Sigma + \sum_{j\in[N-1]} \rho_{ij} X^j \in \mathbb{Z}_q[X]$. The evaluation in the encrypted domain works likewise (when including the values ρ_{ij} in encrypted form in the public key) and the

proof Π remains the same. Also the signature size is not influenced by this extension. Finally, the proofs easily generalize from 2 to N and hold under exactly the same argumentation. Thus we do not restate them.

6.10 Comparison with Previous Work

Now we want to compare the existing instantiations of DAPS in the factoring (F) and discrete logarithm (DL) setting with the ones presented in this paper. We stress that we are interested in cryptographic settings that are currently widely used and thus do not consider the lattice-based DAPS in [BKN17]. In Table 2, which is based on the recent work in [BPS17], we present a comparison of existing DAPS in terms of operation count and sizes of public keys and signatures. For reference, we also include the costs of ECDSA.

The costs of the factoring-based schemes are dominated by exponentiations with the respective RSA modulus. Observe that the savings in the signature size of one hash digest when applying the ID2 transform instead of the H2 transform, comes at the cost of twice the amount of operations during signing and thrice the operations during verification. While choosing MR as identification-scheme over GQ allows to reduce the operation count for verification and the size of the public key, signing costs are the same.

The performance of RKS largely depends on the concrete choice for the Merkle tree. When using a pseudorandom function (PRF) with k bit output, the arity of the tree r and the height h need to satisfy $r^h \geq 2^{2k}$. Additionally, the group $\mathbb{G}$ needs to be compatible with the PRF, i.e., $\log_2 |\mathbb{G}| = 2k$. For example, when using a binary tree ($r = 2$), then the height needs to be at least $2k$. While increasing the arity decreases the verification times, signing times and signature sizes increase.

When looking at our DAPS construction, the operation count of signing and verification takes an extra 4, respectively 6 group operations. The signature contains 3 additional $\mathbb{Z}_q$ elements. When instantiating our construction with ECDSA, signing requires 5 group operations in total, and verification takes 8 group operations. Signatures consists of 5 $\mathbb{Z}_q$ elements.

7 Implementation

We now present an implementation[11] of our ECDSA-DAPS based on the widely used OpenSSL[12] library and its ECDSA implementation. We note that OpenSSL's ECDSA implementation can be extended without any modifications. But also any other ECDSA implementation can be extended in the same way as long as an API for the necessary group operations is available. Note that any implementation of our DAPS construction is extendable to NAPS.

We observe that when implementing our scheme, we do not have to store (r_i, ρ_i) in the secret key. It is only necessary to store the secret signing key, since

[11] The implementation is available at https://github.com/IAIK/daps-dl.
[12] https://openssl.com.

Scheme	Sign	Verify	$\lvert\mathsf{pk}\rvert$	$\lvert\sigma\rvert$	**Setting**	**Model**
PS	$\ell\,\mathrm{E}_k^k$	$l\,\mathrm{E}_k^k$	k	ℓk	F	ROM
H2[GQ]	$2\,\mathrm{E}_{k/2}^{k/2}+\mathrm{E}_k^\ell$	E_k^ℓ	$3k$	$k+\ell$	F	ROM
ID2[GQ]	$4\,\mathrm{E}_{k/2}^{k/2}+2\,\mathrm{E}_k^\ell$	$3\,\mathrm{E}_k^\ell$	$3k$	$k+1$	F	ROM
H2[MR]	$2\,\mathrm{E}_{k/2}^{k/2}+\mathrm{E}_k^\ell$	$\frac{2\ell}{3}\,\mathrm{M}_k$	k	$k+\ell$	F	ROM
RKS	$(r-1)h\,\mathrm{S}_\mathbb{G}$	$2h\,\mathrm{S}_\mathbb{G}$	$2s_\mathbb{G}+k$	$((h-1)r+1)s_\mathbb{G}$	DL	ROM
Σ-DAPS	$\mathsf{Sign}_\Sigma+4\,\mathrm{S}_\mathbb{G}$	$\mathsf{Verify}_\Sigma+6\,\mathrm{S}_\mathbb{G}$	$\lvert\mathsf{pk}_\Sigma\rvert+(1+2n)s_\mathbb{G}$	$\lvert\sigma_\Sigma\rvert+3s_{\mathbb{Z}_q}$	DL	ROM
ECDSA-DAPS	$5\,\mathrm{S}_\mathbb{G}$	$8\,\mathrm{S}_\mathbb{G}$	$(2+2n)s_\mathbb{G}$	$5s_{\mathbb{Z}_q}$	DL	ROM
ECDSA	$\mathrm{S}_\mathbb{G}$	$2\,\mathrm{S}_\mathbb{G}$	$s_\mathbb{G}$	$2s_{\mathbb{Z}_q}$	DL	ROM

Table 2: Operation count, sizes of public keys (pk) and signatures (σ). Factoring-based: $\mathrm{E}_m^{m'}$ exponentiation with modulus of size m and exponent of size m', M_m multiplication with modulus of size m, k size of modulus, ℓ size of hash digest. DL-based: $\mathrm{S}_\mathbb{G}$ scalar multiplication and $s_\mathbb{G}$ size of an element in group $\mathbb{G}$, n number of addresses. RKS: r arity and h height of the tree, k size of PRF output.

we can recover (r_i, ρ_i) from their encryption under the signing key stored in the public key.

7.1 Benchmarking ECDSA-DAPS

For comparing our construction with existing DAPS implementations, we benchmarked ECDSA-DAPS using curves `secp256k1` and `prime256v1` and the DAPS schemes H2[GQ], ID2[GQ], and H2[MR] from [BPS17] with a 2048 bit modulus. The benchmarks were performed on an Intel Core i7-4790 CPU and 16 GB RAM running Ubuntu 17.04. We omit the PS and RKS DAPS in this comparison, as they are by far not competitive; neither in terms of signature size nor performance (cf. [BPS17, Figure 21] for an overview). For reference, we also include sizes and timings for ECDSA. For the sizes required to store elliptic curve points,

Scheme	Sign [ms]	Verify [ms]	$\lvert\mathsf{sk}\rvert$ [bits]	$\lvert\mathsf{pk}\rvert$ [bits]	$\lvert\sigma\rvert$ [bits]
H2[GQ]	1.12	0.65	4096	6144	2304
ID2[GQ]	2.12	2.06	4096	6144	2049
H2[MR]	1.36	0.58	4096	2048	2304
ECDSA-DAPS (`secp256k1`)	0.76	1.33	256	$514 + 1026n$	1280
ECDSA-DAPS (`prime256v1`)	0.23	0.35	256	$514 + 1026n$	1280
ECDSA (`secp256k1`)	0.09	0.35	256	257	512
ECDSA (`prime256v1`)	0.06	0.21	256	257	512

Table 3: Timings and sizes of private keys (sk), public keys (pk) and signatures (σ) with n addresses.

we assume that point compression is used.[13]

Compared to H2[GQ], ID2[GQ], and H2[MR], ECDSA-DAPS using the curve `prime256v1` is an order of magnitude faster when signing and verification is of the same order of magnitude, yet slightly faster as the faster H2 schemes. For ECDSA-DAPS using `secp256k1` the picture for verification is slightly different: verification is comparable to the slower ID2[GQ] scheme. The difference in the signing and verifications times that can be observed in conventional ECDSA and ECDSA-DAPS when switching curves, and it shows that OpenSSL includes a more optimized implementation of the arithmetic on `prime256v1`.

8 Conclusion

In this paper we asked whether one can construct DAPS from signature schemes used in practice. We affirmatively have answered this question by presenting

[13] We store the x-coordinate and a bit indicating the "sign" of the y-coordinate. So points require $b + 1$ bits instead of $2b$ bits for b-bit curves.

provably secure DAPS schemes, among others, from the widely used ECDSA signature scheme. They are the shortest among all existing DAPS schemes and improve over the most efficient factoring and discrete logarithm based schemes. Moreover, we showed how to extend our approach to N-times-authentication-preventing signatures for any $N > 2$. We provided an integration into the OpenSSL library to foster fast adoption in practical applications, of which we discuss some interesting ones in this paper.

Acknowledgements. The authors have been supported by EU H2020 Project PRISMACLOUD, grant agreement n°644962.

References

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *CRYPTO*, 2004.

[BDL+12] Daniel J. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. High-speed high-security signatures. *J. Cryptographic Engineering*, 2(2), 2012.

[Ber06] Daniel J. Bernstein. Curve25519: New diffie-hellman speed records. In *PKC*, 2006.

[BKN17] Dan Boneh, Sam Kim, and Valeria Nikolaenko. Lattice-based DAPS and generalizations: Self-enforcement in signature schemes. In *Applied Cryptography and Network Security - 15th International Conference, ACNS 2017, Kanazawa, Japan, July 10-12, 2017, Proceedings*, pages 457–477, 2017.

[BPS16] Mihir Bellare, Bertram Poettering, and Douglas Stebila. From identification to signatures, tightly: A framework and generic transforms. In *ASIACRYPT*, 2016.

[BPS17] Mihir Bellare, Bertram Poettering, and Douglas Stebila. Deterring certificate subversion: Efficient double-authentication-preventing signatures. In *PKC*, 2017.

[Bro02] Daniel R. L. Brown. Generic groups, collision resistance, and ECDSA. *IACR ePrint*, 2002.

[Bro05] Daniel R. L. Brown. Generic groups, collision resistance, and ECDSA. *Des. Codes Cryptography*, 35(1), 2005.

[CFN88] David Chaum, Amos Fiat, and Moni Naor. Untraceable electronic cash. In *CRYPTO*, 1988.

[CL01] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *EUROCRYPT*, 2001.

[CP92] David Chaum and Torben P. Pedersen. Wallet databases with observers. In *CRYPTO*, 1992.

[DAM+15] Zakir Durumeric, David Adrian, Ariana Mirian, Michael Bailey, and J. Alex Halderman. A search engine backed by internet-wide scanning. In *ACM CCS*, 2015.

[DY05] Yevgeniy Dodis and Aleksandr Yampolskiy. A verifiable random function with short proofs and keys. In *PKC*, 2005.

[EG85] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *CRYPTO*, 1985.

[Fel87] Paul Feldman. A practical scheme for non-interactive verifiable secret sharing. In *FOCS*, 1987.

[FKMV12] Sebastian Faust, Markulf Kohlweiss, Giorgia Azzurra Marson, and Daniele Venturi. On the non-malleability of the fiat-shamir transform. In *INDOCRYPT*, 2012.

[FKP16] Manuel Fersch, Eike Kiltz, and Bertram Poettering. On the provable security of (EC)DSA signatures. In *ACM CCS*, 2016.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, 1986.

[Gam84] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *CRYPTO*, 1984.

[GQ88] Louis C. Guillou and Jean-Jacques Quisquater. A "paradoxical" indentity-based signature scheme resulting from zero-knowledge. In *CRYPTO*, 1988.

[Ham15] Mike Hamburg. Ed448-goldilocks, a new elliptic curve. *IACR ePrint*, 2015.

[HPM94] Patrick Horster, Holger Petersen, and Markus Michels. Meta-elgamal signature schemes. In *ACM CCS*, 1994.

[KAC12] Ghassan Karame, Elli Androulaki, and Srdjan Capkun. Double-spending fast payments in bitcoin. In *ACM CCS*, 2012.

[KAR+15] Ghassan O. Karame, Elli Androulaki, Marc Roeschlin, Arthur Gervais, and Srdjan Capkun. Misbehavior in bitcoin: A study of double-spending and accountability. *ACM Trans. Inf. Syst. Secur.*, 18(1), 2015.

[KMP16] Eike Kiltz, Daniel Masny, and Jiaxin Pan. Optimal security proofs for signatures from identification schemes. In *CRYPTO*, 2016.

[KR00] Hugo Krawczyk and Tal Rabin. Chameleon signatures. In *NDSS*, 2000.

[MR02] Silvio Micali and Leonid Reyzin. Improving the exact security of digital signature schemes. *J. Cryptology*, 15(1), 2002.

[MRV99] Silvio Micali, Michael O. Rabin, and Salil P. Vadhan. Verifiable random functions. In *FOCS*, 1999.

[MS02] John Malone-Lee and Nigel P. Smart. Modifications of ECDSA. In *SAC*, 2002.

[Por13] T. Pornin. Deterministic usage of the digital signature algorithm (dsa) and elliptic curve digital signature algorithm (ecdsa). RFC 6979, August 2013. http://www.rfc-editor.org/rfc/rfc6979.txt.

[PS96] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In *EUROCRYPT '96*, 1996.

[PS14] Bertram Poettering and Douglas Stebila. Double-authentication-preventing signatures. In *ESORICS*, 2014.

[PS17] Bertram Poettering and Douglas Stebila. Double-authentication-preventing signatures. *Int. J. Inf. Sec.*, 16(1), 2017.

[PWH+17] Dimitrios Papadopoulos, Duane Wessels, Shumon Huque, Moni Naor, Jan Vcelák, Leonid Reyzin, and Sharon Goldberg. Can NSEC5 be practical for DNSSEC deployments? *IACR ePrint, to appear at NDSS'17*, 2017.

[RKS15] Tim Ruffing, Aniket Kate, and Dominique Schröder. Liar, liar, coins on fire!: Penalizing equivocation by loss of bitcoins. In *ACM CCS*, 2015.

[Sch89] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In *CRYPTO*, 1989.

[Sha79] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11), 1979.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In *EUROCRYPT '97*, 1997.

[SW17] Haya Shulman and Michael Waidner. One key to sign them all considered vulnerable: Evaluation of DNSSEC in the internet. In *USENIX*, 2017.

[vRJS16] Roland van Rijswijk-Deij, Mattijs Jonker, and Anna Sperotto. On the adoption of the elliptic curve digital signature algorithm (ECDSA) in DNSSEC. In *CNSM*, 2016.

A Cryptographic Assumptions

Subsequently, we present the decisional Diffie-Hellman (DDH or 1-LIN) and decision linear (DLIN or 2-LIN) assumptions, very common assumptions underlying the IND-CPA security of versions of the ElGamal encryption scheme.

Definition 10 (DDH). *The DDH assumptions holds relative to $\mathcal{G} = (\mathbb{G}, q, g)$, if for all PPT adversaries $\mathcal{A}$, there is a negligible function ε such that*

$$\left|\Pr\left[\begin{matrix} x, y, z \xleftarrow{R} \mathbb{Z}_q, \\ b^* \leftarrow \mathcal{A}\left(g^x, g^y, g^{b \cdot xy + (1-b)z}\right) \end{matrix} : b = b^*\right] - \frac{1}{2}\right| \leq \varepsilon(\kappa)$$

Definition 11 (DLIN). *The DLIN assumptions holds relative to $\mathcal{G} = (\mathbb{G}, q, g)$, if for all PPT adversaries $\mathcal{A}$, there is a negligible function ε such that*

$$\left|\Pr\left[\begin{matrix} u, v, h \xleftarrow{R} \mathbb{G},\ x, y, z \xleftarrow{R} \mathbb{Z}_q, \\ b^* \leftarrow \mathcal{A}\left(\begin{matrix} u, v, h, u^x, v^y, \\ h^{b \cdot (x+y) + (1-b)z} \end{matrix}\right) \end{matrix} : b = b^*\right] - \frac{1}{2}\right| \leq \varepsilon(\kappa)$$

B Schnorr Signature Scheme

The Schnorr signature scheme [Sch89] can be seen as a prime example of a signature scheme obtained from an identification scheme using the Fiat-Shamir heuristic [FS86]. We present an instantiation of Schnorr in Scheme 5. The Schnorr

$\underline{\mathsf{KGen}_{\mathsf{Schnorr}}(1^\kappa)}$: Let $\mathcal{G} = (\mathbb{G}, q, g)$. Choose $x \xleftarrow{R} \mathbb{Z}_q^*$ and set $\mathsf{sk} \leftarrow x$ and $\mathsf{pk} \leftarrow g^x$ and return $(\mathsf{sk}, \mathsf{pk})$.
$\underline{\mathsf{Sign}_{\mathsf{Schnorr}}(\mathsf{sk}, m)}$: Parse sk as x and choose $k \xleftarrow{R} \mathbb{Z}_q^*$. Compute $c \leftarrow H(g^k \| m)$, $s \leftarrow k - cx$ and return (c, s).
$\underline{\mathsf{Verify}_{\mathsf{Schnorr}}(\mathsf{pk}, m, \sigma)}$: Parse σ as (c, s) and compute $r \leftarrow g^s \mathsf{pk}^c$. Return 1 if $c = H(r \| m)$ and 0 otherwise.

Scheme 5: Schnorr signature scheme.

signature scheme can be shown to provide EUF-CMA security in the random oracle model (ROM) under the DLP in $\mathbb{G}$ by using the now popular rewinding technique [PS96] (cf. also [KMP16] for a recent treatment on tightness and optimality of such reductions).

C DSE* Security of DAPS

We recall the DSE* security notion of DAPS. The game is depicted in Figure 6, where in contrast to Figure 3 the keys are allowed to be generated by the adversary.

Definition 12 (DSE^* [PS14]). *A DAPS scheme provides double-signature extraction (DSE^*), if for all PPT adversaries $\mathcal{A}$ there is a negligible function $\varepsilon(\cdot)$ such that*

$$\Pr\left[\mathbf{Exp}^{\mathsf{DSE}^*}_{\mathcal{A},\mathsf{DAPS}^*}(\kappa) = 1\right] \leq \varepsilon(\kappa),$$

where the corresponding experiment is depicted in Figure 6.

$\mathbf{Exp}^{\mathsf{DSE}^*}_{\mathcal{A},\mathsf{DAPS}}(\kappa)$:
$(\mathsf{pk}_\mathsf{D}, m_1, m_2, \sigma_1, \sigma_2) \leftarrow \mathcal{A}(1^\kappa)$
return 0, if m_1 and m_2 are not colliding
$v_i \leftarrow \mathsf{Verify}_\mathsf{D}(\mathsf{pk}_\mathsf{D}, m_i, \sigma_i)$ for $i \in [2]$
return 0, if $v_1 = 0$ or $v_2 = 0$
$\mathsf{sk}'_\mathsf{D} \leftarrow \mathsf{Ex}_\mathsf{D}(\mathsf{pk}_\mathsf{D}, m_1, m_2, \sigma_1, \sigma_2)$
return 1, if sk' is not the secret key corresponding to pk_D
return 0

Fig. 6: DSE^* **security for** DAPS.

D IND-CPA Security

IND-CPA security of an encryption scheme Ω is depicted in Figure 7.

Definition 13 (IND-CPA). *A public key encryption scheme Ω is* IND-CPA *secure, if for all PPT adversaries $\mathcal{A}$ there is a negligible function $\varepsilon(\cdot)$ such that*

$$\Pr\left[\mathbf{Exp}^{\mathsf{IND\text{-}CPA}}_{\mathcal{A},\Omega}(\kappa) = 1\right] \leq \varepsilon(\kappa),$$

where the corresponding experiment is depicted in Figure 7.

$\mathbf{Exp}^{\mathsf{IND\text{-}CPA}}_{\mathcal{A},\Omega}(\kappa)$
$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(1^\kappa)$
$b \leftarrow \{0, 1\}$
$(m_0, m_1, \mathtt{state}_\mathcal{A}) \leftarrow \mathcal{A}(\mathsf{pk})$
if $m_0 \notin \mathcal{M} \ \vee \ m_1 \notin \mathcal{M}$, let $C \leftarrow \perp$
else, let $C^* \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)$
$b^* \leftarrow \mathcal{A}(C^*, \mathtt{state}_\mathcal{A})$
return 1, if $b^* = b$
return 0

Fig. 7: IND-CPA security.

E Σ-Protocols

Let $L \subseteq \mathsf{X}$ be an **NP**-language with associated witness relation R so that $L = \{x \mid \exists w : R(x,w) = 1\}$. A Σ-protocol for language L is defined as follows.

Definition 14. *A Σ-protocol for language L is an interactive three-move protocol between a PPT prover $\mathsf{P} = (\mathsf{Commit}, \mathsf{Prove})$ and a PPT verifier $\mathsf{V} = (\mathsf{Challenge}, \mathsf{Verify})$, where P makes the first move and transcripts are of the form $(\mathsf{a}, \mathsf{c}, \mathsf{s}) \in \mathsf{A} \times \mathsf{C} \times \mathsf{S}$. Additionally they satisfy the following properties:*

Completeness *A Σ-protocol for language L is complete, if for all security parameters κ, and for all $(x,w) \in R$, it holds that*

$$\Pr[\langle \mathsf{P}(1^\kappa, x, w), \mathsf{V}(1^\kappa, x)\rangle = 1] = 1.$$

Special Soundness *A Σ-protocol for language L is special sound, if there exists a PPT extractor $\mathcal{E}$ so that for all x, and for all sets of accepting transcripts $\{(\mathsf{a}, \mathsf{c}_i, \mathsf{s}_i)\}_{i\in[2]}$ with respect to x where $\mathsf{c}_1 \neq \mathsf{c}_2$, generated by any algorithm with polynomial runtime in κ, it holds that*

$$\Pr\left[\begin{matrix} w \leftarrow \mathcal{E}(1^\kappa, x, \\ \{(\mathsf{a}, \mathsf{c}_i, \mathsf{s}_i)\}_{i\in[2]}) \end{matrix} \;:\; (x,w) \in R\right] \geq 1 - \varepsilon(\kappa).$$

Special Honest-Verifier Zero-Knowledge *A Σ-protocol is special honest-verifier zero-knowledge, if there exists a PPT simulator $\mathcal{S}$ so that for every $x \in L$ and every challenge c from the challenge space, it holds that a transcript $(\mathsf{a}, \mathsf{c}, \mathsf{s})$, where $(\mathsf{a}, \mathsf{s}) \leftarrow \mathcal{S}(1^\kappa, x, \mathsf{c})$ is indistinguishable from a transcript resulting from an honest execution of the protocol.*

F NIZK Security Properties

Definition 15 (Completeness). *A non-interactive proof system for language L is complete, if for all $\kappa \in \mathbb{N}$, for all $\mathsf{crs} \leftarrow \mathsf{Setup}_\Pi(1^\kappa)$, for all $x \in L$, for all w such that $R(x,w) = 1$, and for all $\pi \leftarrow \mathsf{Proof}_\Pi(\mathsf{crs}, x, w)$, we have that $\mathsf{Verify}_\Pi(\mathsf{crs}, x, \pi) = 1$.*

This captures perfect completeness.

Definition 16 (Soundness). *A non-interactive proof system for language L is sound, if for every PPT adversary $\mathcal{A}$ there exists a negligible function ε such that:*

$$\Pr\left[\begin{matrix} \mathsf{crs} \leftarrow \mathsf{Setup}_\Pi(1^\kappa), \\ (x,\pi) \leftarrow \mathcal{A}(\mathsf{crs}) \end{matrix} : \begin{matrix} \mathsf{Verify}_\Pi(\mathsf{crs}, x, \pi) \\ = 1 \wedge x \notin L \end{matrix}\right] \leq \varepsilon(\kappa).$$

Definition 17 (Zero-Knowledge). *A non-interactive proof system for language L is zero-knowledge, if there exists an efficient simulator $\mathsf{S} = (\mathcal{S}_1, \mathcal{S}_2)$*

such that for any efficient adversary $\mathcal{A}$ there exist a negligible function $\varepsilon_1(\cdot)$ such that:

$$\left| \begin{array}{l} \Pr\left[\mathsf{crs} \leftarrow \mathsf{Setup}_{\Pi}(1^{\kappa}) : \mathcal{A}(\mathsf{crs}) = 1\right] \\ \Pr\left[(\mathsf{crs}, \tau) \leftarrow \mathcal{S}_1(1^{\kappa}) : \mathcal{A}(\mathsf{crs}) = 1\right] \end{array} - \right| \leq \varepsilon_1(\kappa),$$

and for any efficient adversary $\mathcal{A}$ there exists a negligible function $\varepsilon_2(\cdot)$ such that

$$\left| \Pr[\mathsf{Zero\text{-}Knowledge}^{\Pi}_{\mathcal{A},\mathsf{S}}(\kappa) = 1] - \frac{1}{2} \right| \leq \varepsilon_2(\kappa),$$

where the corresponding experiment is depicted in Figure 8.

Experiment $\mathsf{Zero\text{-}Knowledge}^{\Pi}_{\mathcal{A},\mathsf{S}}(\kappa)$
  $b \leftarrow \{0,1\}$
  $(\mathsf{crs}, \tau) \leftarrow \mathcal{S}_1(1^{\kappa})$
  $b^* \leftarrow \mathcal{A}^{\mathsf{P}_b(\cdot,\cdot)}(\mathsf{crs})$
    where oracle P_0 on input (x, w):
      return $\pi \leftarrow \mathsf{Proof}_{\Pi}(\mathsf{crs}, x, w)$, if $(x, w) \in R$
      return $\perp$
    and oracle P_1 on input (x, w):
      return $\pi \leftarrow \mathcal{S}_2(\mathsf{crs}, \tau, x)$, if $(x, w) \in R$
      return $\perp$
  return 1, if $b = b^*$
  return 0

Fig. 8: Zero-Knowledge