Shorter double-authentication preventing signatures for small address spaces*

Bertram Poettering
bertram.poettering@rhul.ac.uk

Information Security Group
Royal Holloway, University of London

Abstract. A recent paper by Derler, Ramacher, and Slamanig (IEEE EuroS&P 2018) constructs double-authentication preventing signatures ("DAP signatures", a specific self-enforcement enabled variant of signatures where messages consist of an address and a payload) that have—if the supported address space is not too large—keys and signatures that are considerably more compact than those of prior work. We embark on their approach to restrict attention to small address spaces and construct novel DAP schemes that beat their signature size by a factor of five and reduce the signing key size from linear to constant (the verification key size remains almost the same). We construct our DAP signatures generically from identification protocols, using a transform similar to but crucially different from that of Fiat and Shamir. We use random oracles. We don't use pairings.

Keywords: Signature schemes, self-enforcement, identification protocols, provable security

1 Introduction

Digital signatures. Digital signature schemes are a ubiquitous cryptographic primitive. They are extensively used for message and entity authentication and find widespread application in real-world protocols. The basic functionality of a signature scheme is as follows: A signer first runs the *key generation* algorithm to create a key pair consisting of a (secret) signing key and a (public) verification key. The signing key can then be used with the *signing algorithm* to create signatures on messages. Such a signature is a short bitstring that serves as an authenticator for a message: Given an authentic copy of the verification key, anybody can invoke the *verification algorithm* together with a message and the signature to check for the latter's validity. The output of this algorithm is binary: either "accept", interpreted as indicating that the message is authentic in the sense that it was fed by the signer into its signing algorithm, or "reject", which means it is not. Signature schemes were first proposed about four decades ago,

* The full version is available in the IACR eprint archive as article 2018/223 [16].

and gazillions of constructions are known today. Among the standardized ones, and these are the fewest, are RSA-PSS, RSA-PKCS#1 v1.5, DSA, ECDSA, and EdDSA.

Digital signatures with self-enforcement. According to the classic understanding of a signature scheme, signers can, in principle, sign any message they want. Applications, however, might require that specific relations hold between signed messages. For instance, in a public-key infrastructure (PKI), a certificate authority (CA) is expected not to generate certificates on the same identity for different public keys (as this could lead to impersonation attacks), and in cryptographic "currencies" like Bitcoin users shall not double-spend (sign two transactions that transfer the same coins to different recipients). While a regular signature scheme offers no means to enforce specific configurations of signed messages, signature schemes with self-enforcement do. We describe e-cash and double-authentication-preventing signatures as two examples for the latter.

E-Cash. In electronic cash [6], users can transfer virtual coins to each other, and they have a guarantee of anonymity: Nobody, including any central issuing authority, can trace their payments. However, if double-spending happens (this is very explicitly forbidden in e-cash systems) then the spending user's anonymity is automatically revoked in that the user can be identified given the two transactions. This allows for penalizing misbehaving users, for instance by freezing their account.

Double-Authentication Preventing Signatures. A signature scheme is double authentication preventing [17], or "has the DAP property", if messages consist of an address and a payload field, and the scheme is such that the signer is penalized if it signs any address twice with different payloads. In this setting the penalty is more drastic than in the e-cash setting: misbehaving signers don't have their anonymity revoked, but instead a verbatim copy of their signing key is leaked to the public. A natural application is in the PKI setting: If domain names are used as addresses, and public keys as payloads, then the DAP property means that the certification authority is penalized if it issues different-key certificates for the same domain. A different application is related to cryptographic currencies: users that double-spend make necessarily also their signing key public, i.e., give everybody access to their funds. Further applications have been proposed in [17,18,19,5,3,7], for instance in the context of secure contract signing (and enforcing).

A brief history of double-authentication preventing signatures. The first DAP scheme is by Poettering and Stebila [17,18]. Their construction is in the RSA setting. More precisely, it builds on the number-theoretic fact that if one has two square roots of an element modulo a Blum integer, then one also has the factorization of this number.[1] Signatures of the DAP scheme use the factorization as signing key, and signatures consist of a vector of square roots of RSA

[1] It is further required that these square roots are not additive inverses of each other.

elements such that the decision about which specific square root is the valid one is a function of the payload. Signing different payloads means releasing different square roots means revealing the signing key. The signatures of this scheme are very large: If a 128 bit security level shall be reached then their size is at least 256 times 2048 bit.

A DAP scheme in the DLP setting was proposed by Ruffing, Kate, and Schröder [19, Appendix A], based on Merkle trees based on Chameleon hash functions [13]. The addresses are associated with the leaves of these trees, and signatures consist of vectors of 'openings' of the Chameleon hashes leading to the root. While their signatures are shorter than those of [17,18] (if implemented over elliptic curves), the signature size is linear in the bit-length of the address space, and thus still prohibitively large for many applications.

The work by Bellare, Poettering, and Stebila [3] improves on the work of [17,18] by compressing DAP signatures, still in the RSA setting and targeting the 128 bit security level, down to the size of 2048 bit. The trick is to evaluate the square root function iteratively, instead of in parallel. While this requires more algebraic properties than [17,18], the Blum integer setting fulfills these. We note that the DAP scheme of [3] is, so far, the only one with tight reductions.

The work of Boneh, Kim, and Nikolaenko [5] constructs DAP signatures based on lattice assumptions. The focus is on finding a solution for the post-quantum setting, not to beat the signature size of [3] (which they don't).

The very recent work of Derler, Ramacher, and Slamanig [7] reports the smallest DAP signature size so far: 1280 bit at the 128 bit security level, in the DLP-setting. Their scheme, however, can only be used in a restricted setting: As the sizes of signing and verification keys grow linearly in the cardinality of the address space, the scheme is not practical unless the latter is small. Note that all prior works (in particular [17,18,19,5,3]) support address spaces of exponential size. This drawback is acknowledged in [7], but the authors also report on specific applications where a small address space is just naturally occurring. Technically, the scheme of [7] builds on regular DLP-based signatures like Schnorr and DSA, and achieves the DAP property by including in each signature a payload-dependent share of the signing key, created with a secret-sharing scheme such that from any two shares associated with an address the signing key can be recovered. To show that the shares are well-formed, i.e., indeed allow for key recovery, the DAP signatures also contain a corresponding NIZK proof.

Contribution. In this article we embark on the approach of Derler et al. [7] and study DAP signatures for small address spaces. Also we work in the DLP setting, and build on (EC)DSA or Schnorr signatures. However, we improve drastically on signature and key sizes: Our signatures are five times shorter (namely only 256 bits) and our signing keys are constant size (instead of linear in the cardinality of the address space). While our and their verification keys are roughly the same size, our signing and verification times are better.

In a nutshell, our approach is to draw on the special-soundness property of the identification schemes underlying Schnorr and (EC)DSA signatures, assigning to each address one particular commitment. As these commitments are included in

the verification key, the size of the latter is linear in the cardinality of the address space. Note that this linear blow-up is also the case in [7], but in contrast to their scheme we do not build on further primitives (e.g., secret sharing), which overall leads to more robustness, an easier analysis, more compact keys and signatures, and faster algorithms.

2 Notation

We write $\mathbb{Z}$ for the set of integers, and $\mathtt{T}$ and $\mathtt{F}$ for the Boolean constants true and false, respectively.

Parts of this article involve the specification of program code. In such code we use assignment operator '$\leftarrow$' when the assigned value results from a constant expression (including from the output of a deterministic algorithm), and we write '$\leftarrow_{\$}$' when the value is either sampled uniformly at random from a finite set or is the output of a randomized algorithm. For a randomized algorithm A we write $y \leftarrow_{\$} A(x_1, x_2, \ldots)$ to denote the operation of running A with inputs $x_1, x_2, \ldots$ (and fresh coins) and assigning the output to variable y. Further, we write $[A(x_1, x_2, \ldots)]$ for the set of values that A outputs with positive probability.

Our security definitions are based on games played between a challenger and an adversary. These games are expressed using program code and terminate when the main code block executes a 'Stop with ...' command; the argument of the latter is the output of the game. We write $\Pr[G \Rightarrow \mathtt{T}]$ or just $\Pr[G]$ for the probability that game G terminates by running into a 'Stop with $\mathtt{T}$' instruction. Further, if E is some game-internal event, we similarly write $\Pr[E]$ for the probability that this event occurs. (Note the game is implicit in this notation.)

We use bracket notation to denote associative arrays (a data structure that implements a 'dictionary'). For instance, for an associative array A the instruction $A[7] \leftarrow 3$ assigns value 3 to memory position 7, and the expression $A[2] = 5$ tests whether the value at position 2 is equal to 5. Associative arrays can be indexed with elements from arbitrary sets. We use expressions like $A[\cdot] \leftarrow x$ to indicate that A is initialized with default value x. (That is, for any y, unless $A[y]$ is explicitly overwritten with a different value, $A[y]$ evaluates to x.) When assigning lists to each other, with '_' we mark "don't-care" positions. For instance, $(a, \_) \leftarrow (9, 4)$ is equivalent to $a \leftarrow 9$ (value 4 is discarded).

3 Signature schemes and key extractability

We first reproduce standard definitions associated with signature schemes, and then consider a less common property that signature schemes might have: key extractability.

3.1 Regular signature schemes

We recall the definition of digital signatures and their essential security property: unforgeability.

Digital Signature Schemes. A *digital signature* scheme (DS) for a message space $\mathcal{M}$ consists of algorithms $\mathsf{gen}, \mathsf{sgn}, \mathsf{vfy}$ together with a signing key space $\mathcal{SK}$, a verification key space $\mathcal{VK}$, and a signature space $\mathcal{S}$. The key generation algorithm gen outputs a signing key $sk \in \mathcal{SK}$ and a verification key $vk \in \mathcal{VK}$. The signing algorithm sgn takes a signing key $sk \in \mathcal{SK}$ and a message $m \in \mathcal{M}$, and outputs a signature $\sigma \in \mathcal{S}$. The verification algorithm vfy takes a verification key $vk \in \mathcal{VK}$, a message $m \in \mathcal{M}$, and a (candidate) signature $\sigma \in \mathcal{S}$, and outputs either $\mathtt{T}$ or $\mathtt{F}$ to indicate acceptance or rejection, respectively. A shortcut notation for these syntactical definitions is

$$\mathsf{gen} \to \mathcal{SK} \times \mathcal{VK} \qquad \mathcal{SK} \times \mathcal{M} \to \mathsf{sgn} \to \mathcal{S} \qquad \mathcal{VK} \times \mathcal{M} \times \mathcal{S} \to \mathsf{vfy} \to \{\mathtt{T}, \mathtt{F}\} \ .$$

For a verification key $vk \in \mathcal{VK}$ we denote with $\mathcal{V}(vk)$ the set of message-signature pairs that are valid with respect to vk:

$$\mathcal{V}(vk) := \{(m, \sigma) \in \mathcal{M} \times \mathcal{S} : \mathsf{vfy}(vk, m, \sigma) = \mathtt{T}\} \ .$$

A signature scheme is correct if for all $(sk, vk) \in [\mathsf{gen}]$ and $m \in \mathcal{M}$ and $\sigma \in [\mathsf{sgn}(sk, m)]$ we have $(m, \sigma) \in \mathcal{V}(vk)$.

Unforgeability. For reference we reproduce the definition of the standard security notion for signature schemes: (existential) unforgeability (under chosen-message attacks). For a signature scheme Σ, associate with any adversary $\mathcal{F}$ its forging advantage $\mathrm{Adv}^{\mathrm{uf}}_{\Sigma}(\mathcal{F}) := \Pr[\mathrm{UF}(\mathcal{F})]$, where the game is in Fig. 1. Intuitively, a signature scheme provides unforgeability if all practical adversaries have a negligible forging advantage.

Game $\mathrm{UF}(\mathcal{F})$	**Oracle** $\mathrm{Sign}(m)$
00 $L \leftarrow \emptyset$	06 $\sigma \leftarrow_{\$} \mathsf{sgn}(sk, m)$
01 $(sk, vk) \leftarrow_{\$} \mathsf{gen}$	07 $L \leftarrow L \cup \{m\}$
02 $(m^*, \sigma^*) \leftarrow_{\$} \mathcal{F}(vk)$	08 Return σ
03 Require $(m^*, \sigma^*) \in \mathcal{V}(vk)$	
04 Require $m^* \notin L$	
05 Stop with $\mathtt{T}$	

Fig. 1. Security experiment UF modeling the unforgeability of signatures. Adversary $\mathcal{F}$ has access to oracle Sign. We write 'Require C' for a condition C as an abbreviation for 'If not C: Stop with $\mathtt{F}$'.

3.2 Key-extractable signature schemes

We are interested in subclasses of signature schemes where the signing key can be reconstructed from (valid) signatures on specific message configurations. We formalize such extractability properties for *strictly one-time* (SOT) and *double-authentication preventing* (DAP) signatures. In a nutshell, a signature scheme

is strictly one-time if the signing key can be recovered from signatures on any two different messages, and it is double-authentication preventing if messages consist of two components, the address and the payload, and the signing key can be recovered if two messages are signed that have the same address but different payloads. While the DAP definition was first developed in [17,18] we are not aware of a prior formalization of the SOT notion.

KEY EXTRACTABILITY. Let $\mathsf{gen}, \mathsf{sgn}, \mathsf{vfy}, \mathcal{SK}, \mathcal{VK}, \mathcal{S}$ be the algorithms and spaces of a digital signature scheme for a message space $\mathcal{M}$. We say that the scheme is *key-extractable* if there exists an auxiliary extraction algorithm ext that takes a verification key $vk \in \mathcal{VK}$ and two message-signature pairs (m_1, σ_1), $(m_2, \sigma_2) \in \mathcal{M} \times \mathcal{S}$, and outputs either a signing key $sk \in \mathcal{SK}$ or the failure symbol $\perp \notin \mathcal{SK}$. A shortcut notation for this syntactical definition is

$$\mathcal{VK} \times (\mathcal{M} \times \mathcal{S}) \times (\mathcal{M} \times \mathcal{S}) \to \mathsf{ext} \to \mathcal{SK} \,/\, \perp \ .$$

We formulate two notions of correctness (of extraction):

SOT. The signature scheme is *strictly one-time* if algorithm ext is such that for all $(sk, vk) \in [\mathsf{gen}]$ and $(m_1, \sigma_1), (m_2, \sigma_2) \in \mathcal{M} \times \mathcal{S}$ we have

$$(m_1, \sigma_1), (m_2, \sigma_2) \in \mathcal{V}(vk) \ \wedge \ m_1 \neq m_2 \implies \mathsf{ext}(vk, m_1, \sigma_1, m_2, \sigma_2) = sk \ .$$

DAP. The signature scheme is *double-authentication preventing* if there exist an address space $\mathcal{A}$ and a payload space $\mathcal{P}$ such that $\mathcal{M} = \mathcal{A} \times \mathcal{P}$, and algorithm ext is such that for all $(sk, vk) \in [\mathsf{gen}]$ and $(m_1, \sigma_1), (m_2, \sigma_2) \in \mathcal{M} \times \mathcal{S}$, if we write $m_i = (a_i, p_i)$ and have $(m_1, \sigma_1), (m_2, \sigma_2) \in \mathcal{V}(vk)$ then

$$a_1 = a_2 \ \wedge \ p_1 \neq p_2 \implies \mathsf{ext}(vk, m_1, \sigma_1, m_2, \sigma_2) = sk \ .$$

UNFORGEABILITY OF STRICTLY ONE-TIME SIGNATURES. In SOT signatures, everybody getting hold of two valid message-signature pairs can first recover the signing key and then create signatures on arbitrary messages. For such signature schemes the standard notion of unforgeability, where two message-signature pairs are easily obtained through the signing oracle, is thus not meaningful. We hence formalize a dedicated (weakened) unforgeability notion: For a signature scheme Σ, associate with any adversary $\mathcal{F}$ its (strictly one-time) forging advantage $\mathrm{Adv}^{\mathrm{sot}}_{\Sigma}(\mathcal{F}) := \Pr[\mathrm{SOT}(\mathcal{F})]$, where the game is in Fig. 2.[2] Note that the only difference between the UF and SOT games is the added instruction in line 06 of game SOT which precisely prevents the adversary from posing a second query to the Sign oracle. Intuitively, a SOT signature scheme provides unforgeability if all practical adversaries have a negligible forging advantage.

[2] Note that the 'strictness property' of SOT signatures involves only their functionality and is not reflected in the game which formalizes precisely the unforgeability of (regular) one-time signatures. We use the names SOT for the game and sot for the notion merely to allow for a clear association between functionality and targeted security notion.

Game SOT($\mathcal{F}$)	**Oracle** Sign(m)
00 $L \leftarrow \emptyset$	06 Require $L = \emptyset$
01 $(sk, vk) \leftarrow_{\$} \mathsf{gen}$	07 $\sigma \leftarrow_{\$} \mathsf{sgn}(sk, m)$
02 $(m^*, \sigma^*) \leftarrow_{\$} \mathcal{F}(vk)$	08 $L \leftarrow L \cup \{m\}$
03 Require $(m^*, \sigma^*) \in \mathcal{V}(vk)$	09 Return σ
04 Require $m^* \notin L$	
05 Stop with T	

Fig. 2. Security experiment SOT modeling unforgeability for strictly one-time signature schemes. Adversary $\mathcal{F}$ has access to oracle Sign. We write 'Require C' for a condition C as an abbreviation for 'If not C: Stop with F'.

Unforgeability of Double-Authentication Preventing Signatures. In DAP signatures, everybody getting hold of two valid message-signature pairs can create signatures on arbitrary messages if the two messages have the same address but different payloads. As we did for SOT signatures, also for DAP signatures we use a dedicated unforgeability notion [17,18]: For a signature scheme Σ, associate with any adversary $\mathcal{F}$ its forging advantage $\mathrm{Adv}^{\mathrm{dap}}_{\Sigma}(\mathcal{F}) := \Pr[\mathrm{DAP}(\mathcal{F})]$, where the game is in Fig. 3. In the game, note that we replaced the set L (of games UF, SOT) that keeps track of signed messages by an associative array $L[\cdot]$ that manages one such set per address, keeping track of signed payloads. Intuitively, a DAP signature scheme provides unforgeability if all practical adversaries have a negligible forging advantage.

Game DAP($\mathcal{F}$)	**Oracle** Sign(m)
00 $L[\cdot] \leftarrow \emptyset$	07 $(a, p) \leftarrow m$
01 $(sk, vk) \leftarrow_{\$} \mathsf{gen}$	08 Require $L[a] = \emptyset$
02 $(m^*, \sigma^*) \leftarrow_{\$} \mathcal{F}(vk)$	09 $\sigma \leftarrow_{\$} \mathsf{sgn}(sk, m)$
03 $(a^*, p^*) \leftarrow m^*$	10 $L[a] \leftarrow L[a] \cup \{p\}$
04 Require $(m^*, \sigma^*) \in \mathcal{V}(vk)$	11 Return σ
05 Require $p^* \notin L[a^*]$	
06 Stop with T	

Fig. 3. Security experiment DAP modeling unforgeability for double-authentication preventing signature schemes. Adversary $\mathcal{F}$ has access to oracle Sign. We write 'Require C' for a condition C as an abbreviation for 'If not C: Stop with F'.

4 Constructing DAP signatures from SOT signatures

We present a construction of a DAP signature scheme from a SOT signature scheme that offers exceptional performance but requires that the cardinality $n = |\mathcal{A}|$ of the address space is not too large. The basic idea of our scheme is to let the DAP key generation algorithm perform n-many SOT key generations independently of each other, one for each address, and to present the resulting set of

SOT signing (resp. verification) keys as a single DAP signing (resp. verification) key. To DAP-sign a message $m = (a, p)$, the SOT signing key $sk[a]$ corresponding to address a is retrieved and payload p authenticated with it. The DAP verification algorithm works analogously. Observe that, without further modification, this design is unforgeable (in the DAP sense) but not key-extractable. Indeed, double-signing, i.e., violating the DAP property, reveals only one of the required n-many SOT signing keys. We apply two tricks to achieve full key extractability: (1) Instead of generating the n-many SOT signing keys independently of each other and with individual random coins, we generate them deterministically as a function of the address they are associated with, using the output of a PRF as the coins required for key generation. More precisely, for each address a we first derive 'random' coins as per $r \leftarrow F(k, a)$, where F is the PRF and k its key (that is stored as part of the DAP signing key), and then compute the SOT key pair corresponding to a as per $(sk[a], vk[a]) \leftarrow \mathsf{gen}\langle r \rangle$, where the $\langle \cdot \rangle$ notation indicates running an (otherwise randomized) algorithm with explicitly given coins. Note that this reduces the size of the DAP signing key from linear (in n) to constant. (2) We include (one-time pad) encryptions of PRF key k in the DAP verification key such that knowledge of any SOT signing key $sk[a]$ suffices to recover it. More concretely, we embed into the DAP verification key the set of values $k + h(a, sk[a])$, for all $a \in \mathcal{A}$, where $+$ denotes (in most cases) the bit-wise XOR operation and h is a random oracle. (This technique is borrowed from [3].) Overall, as we prove, this is sufficient to achieve key-extractability in the DAP sense. We give the formal details of our construction in the following.

SOT-TO-DAP TRANSFORM. Let $\mathcal{A}, \mathcal{P}$ be sets. From a key-extractable signature scheme Σ for message space $\mathcal{P}$ with algorithms $\mathsf{gen}, \mathsf{sgn}, \mathsf{vfy}, \mathsf{ext}$ and spaces $\mathcal{SK}, \mathcal{VK}, \mathcal{S}$ we construct a key-extractable signature scheme Σ' for message space $\mathcal{M} = \mathcal{A} \times \mathcal{P}$ with algorithms $\mathsf{gen}', \mathsf{sgn}', \mathsf{vfy}', \mathsf{ext}'$ and spaces $\mathcal{SK}', \mathcal{VK}', \mathcal{S}'$ such that if the former is strictly one-time then the latter is double-authentication preventing. As building blocks we employ a pseudorandom function $F\colon \mathcal{K} \times \mathcal{A} \to \mathcal{R}$ that has a commutative group $(\mathcal{K}, +)$ as its key space[3] and the randomness space $\mathcal{R}$ of algorithm gen as its range, and a hash function $h\colon \mathcal{A} \times \mathcal{SK} \to \mathcal{K}$. Both F and h will be modeled as random oracles in the security analysis. Let $\mathcal{SK}' = \mathcal{K}$, $\mathcal{VK}' = \mathcal{VK}^{|\mathcal{A}|} \times \mathcal{K}^{|\mathcal{A}|}$, $\mathcal{S}' = \mathcal{S}$, and implement algorithms $\mathsf{gen}', \mathsf{sgn}', \mathsf{vfy}'$ as specified in Fig. 4. (Algorithm ext' is specified in Fig. 5 and discussed below.)

The signature schemes obtained with our transform provide both unforgeability and key extractability.

Theorem 1 (DAP unforgeability). *Let Σ and Σ' be the SOT and DAP signature schemes involved in our construction. If Σ is unforgeable in the SOT sense then Σ' is unforgeable in the DAP sense. More precisely, for any adversary $\mathcal{F}'$ against Σ' there exist adversaries $\mathcal{E}, \mathcal{F}$ against Σ such that in the random oracle model for F and h we have*

$$\mathrm{Adv}^{\mathrm{dap}}_{\Sigma'}(\mathcal{F}') \leq n \cdot \mathrm{Adv}^{\mathrm{sot}}_{\Sigma}(\mathcal{F}) + q_F/|\mathcal{K}| + n \cdot q_h \cdot \mathrm{Adv}^{\mathrm{sot}}_{\Sigma}(\mathcal{E}) \ ,$$

[3] Consider that same-length bit-strings together with the bit-wise XOR operation form a commutative group to see that this requirement is easily fulfilled in practice.

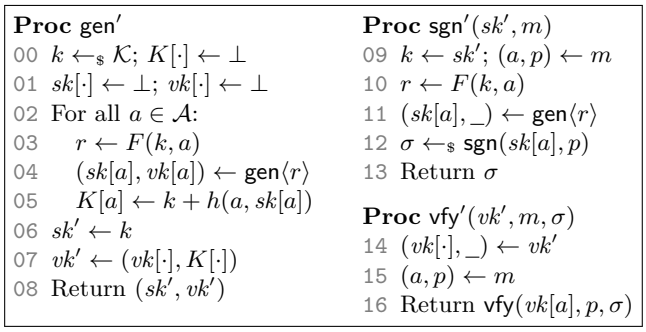

Fig. 4. Our SOT-to-DAP transform (main algorithms). The $\langle \cdot \rangle$ notation in lines 04 and 11 indicates that gen is (deterministically) invoked on random coins r.

where $n = |\mathcal{A}|$ is the cardinality of the address space, q_F and q_h are the numbers of queries to F and h, respectively, and $\mathcal{K}$ is the key space of the PRF. The running times of $\mathcal{E}$ and $\mathcal{F}$ are about that of $\mathcal{F}'$.

Proof sketch. On first sight the security argument for our construction seems to be straight-forward: Any valid DAP signature is in fact a valid SOT signature, so forging the one implies forging the other. Also the DAP signing oracle seems to be easily simulated with the SOT signing oracle. (The latter processes at most one query per key, but this is perfectly matched with the one-query-per-address requirement of DAP unforgeability). The reason why ultimately the proof is not that easy is that the DAP construction uses the SOT scheme in a non-blackbox fashion. More precisely, for a reductionist proof to go through, the random coins of the SOT key generation would need to be freshly drawn right before key generation, and be forgotten immediately afterwards. This is not necessarily the case in our construction, as the coins are generated in a specific and reproducible way, using the PRF. More precisely, as we model the PRF as a random oracle, the coins are uniform and hidden from the adversary as long as the PRF key k is not queried by the latter to its F oracle. However, as h-based encryptions of k are embedded into the verification key, bounding the probability of this event involves arguing about the probability of reconstructing SOT signing keys, and such arguments can only be given if the SOT key generation algorithm receives properly distributed coins, i.e., coins that are drawn uniformly at random. Below we give a more careful analysis that avoids this circularity by cleverly conditioning events on each other.

In the following we refer with E_F to the event that the adversary poses a query with first argument k to its F oracle. We need to bound the probability that E_F occurs to a small value. There are precisely two ways that lead the adversary to posing such a query: (1) Without any knowledge about k but by sheer luck: the adversary guesses $k \in \mathcal{K}$ and hits the right one. As the adversary can try q_F times, the probability for this is bounded by $q_F/|\mathcal{K}|$. (2) By making a more informed guess, i.e., by exploiting obtained knowledge about k. Note that the only information available about k are its random oracle based encryptions,

which are information-theoretically hiding up to the point where the adversary poses a corresponding query to the h oracle. Let E_h be the corresponding event that the adversary queries oracle h on one of the n-many SOT signing keys. Instead of deriving individual bounds for $\Pr[E_F]$ and $\Pr[E_h]$, we analyze the probabilities of two closely related events:

Let E'_F be the event that E_F occurs before E_h, including the case that E_h does not occur at all, and let E'_h be the event that E_h occurs before E_F, including the case that E_F does not occur at all. Then, as discussed above, we have $\Pr[E'_F] \leq q_F/|\mathcal{K}|$. Further, as in the E'_h case we can assume perfectly random coins of SOT key generation, we can bound $\Pr[E'_h]$ with the probability of SOT key recovery, which is in particular bounded by the SOT forging probability. In particular there exists for each $1 \leq \alpha \leq n$ an adversary $\mathcal{E}_\alpha$ such that $\Pr[E'_h \mid \text{the } h \text{ query happens for address number } \alpha] \leq q_h \mathrm{Adv}^{\mathrm{sot}}_{\Sigma}(\mathcal{E}_\alpha)$. By defining adversary $\mathcal{E}$ such that it uniformly picks a value $1 \leq \alpha \leq n$ and then behaves like $\mathcal{E}_\alpha$, we obtain $\Pr[E'_h] \leq n q_h \mathrm{Adv}^{\mathrm{sot}}_{\Sigma}(\mathcal{E})$.

The bounds on E'_F and E'_h can be additively combined as $\Pr[E_F \vee E_h] = \Pr[E'_F] + \Pr[E'_h]$. If E_σ denotes the event of a DAP forgery then overall we have $\mathrm{Adv}^{\mathrm{dap}}_{\Sigma'}(\mathcal{F}') = \Pr[E_\sigma] \leq \Pr[E_\sigma \mid \neg E_F \wedge \neg E_h] + \Pr[E_F \vee E_h]$. We already bounded the second term. For the first term note that $\neg E_F \wedge \neg E_h$ means that the adversary does not exploit the random oracle based encryption of the signing key. In this case the initially discussed natural reduction works, showing $\Pr[E_\sigma \mid \neg E_F \wedge \neg E_h] \leq n \mathrm{Adv}^{\mathrm{sot}}_{\Sigma}(\mathcal{F})$, for a forger $\mathcal{F}$ that results from the reduction.

By combining the above bounds we obtain the one from the theorem statement. □

Theorem 2 (DAP key extractability). *Let Σ and Σ' be the SOT and DAP signature schemes involved in our construction. If Σ is strictly one-time then Σ' is double-authentication preventing. More precisely, the algorithm* ext' *specified in Fig. 5 is an extraction algorithm for scheme Σ' if its building block* ext *is an extraction algorithm for scheme Σ.*

Proc $\mathsf{ext}'(vk', m_1, \sigma_1, m_2, \sigma_2)$
00 $(vk[\cdot], K[\cdot]) \leftarrow vk'$
01 $(a_1, p_1) \leftarrow m_1$; $(a_2, p_2) \leftarrow m_2$
02 Require $a_1 = a_2 \wedge p_1 \neq p_2$
03 Require $(p_1, \sigma_1), (p_2, \sigma_2) \in \mathcal{V}(vk[a_1])$
04 $sk[a_1] \leftarrow \mathsf{ext}(vk[a_1], p_1, \sigma_1, p_2, \sigma_2)$
05 Require $sk[a_1] \neq \bot$
06 $k \leftarrow K[a_1] - h(a_1, sk[a_1])$
07 Return k

Fig. 5. SOT-to-DAP transform (extraction algorithm). We write 'Require C' for a condition C as an abbreviation for 'If not C: Return $\bot$'. Note that the condition in line 05 is always fulfilled.

Proof. The argument is immediate: Having two "colliding" DAP signatures means having two SOT signatures that are valid under the same verification key but are on different messages. The DAP extraction algorithm in Fig. 5 applies the SOT extraction algorithm to this setting to first recover the SOT signing key and then, by decrypting the corresponding h-based ciphertext, the DAP signing key. □

5 Constructing SOT signatures

We propose the *Fixed-Commitment transform* that constructs signature schemes from generic identification (ID) protocols, in a way related to that of the classic Fiat–Shamir transform [10]. While the latter turns ID schemes into standard unforgeable signature schemes, the signature schemes obtained with our new transform are strictly one-time. We first recall details of (three-move) identification protocols, then of the Fiat–Shamir transform, and then specify and study our own construction.

5.1 Three-move ID protocols

We recall the definition of an important class of identification protocols and of the security properties connected to it: special soundness, (honest-verifier) zero-knowledge, and resilience against key recovery. While we refer to [11,12,14] for general treatments of ID protocols, our notation is in particular close to that of [2].

Three-Move ID Protocols. A *three-move ID protocol* consists of algorithms $\mathrm{G}, \mathrm{P}_1, \mathrm{P}_2, \mathrm{V}$, an identification secret key space ISK, an identification public key space IPK, a commitment space CMT, a challenge space CH, a response space RSP, and a (prover) state space ST. The key generation algorithm G outputs a secret key $isk \in \mathrm{ISK}$ and a public key $ipk \in \mathrm{IPK}$. Algorithms P_1 and P_2 are for the prover: Algorithm P_1 takes a secret key $isk \in \mathrm{ISK}$ and a public key $ipk \in \mathrm{IPK}$, and outputs a state $st \in \mathrm{ST}$ and a commitment $cmt \in \mathrm{CMT}$. Algorithm P_2 takes a state $st \in \mathrm{ST}$ and a challenge $ch \in \mathrm{CH}$, and outputs a response $rsp \in \mathrm{RSP}$. Algorithm V is for the verifier: It takes a public key $ipk \in \mathrm{IPK}$, a commitment $cmt \in \mathrm{CMT}$, a challenge $ch \in \mathrm{CH}$, and a response $rsp \in \mathrm{RSP}$, and outputs $\mathtt{T}$ or $\mathtt{F}$ to indicate acceptance or rejection, respectively. A shortcut notation for these syntactical definitions is

$$\begin{array}{rcccl} & & \mathrm{G} & \to & \mathrm{ISK} \times \mathrm{IPK} \\ \mathrm{ISK} \times \mathrm{IPK} & \to & \mathrm{P}_1 & \to & \mathrm{ST} \times \mathrm{CMT} \\ \mathrm{ST} \times \mathrm{CH} & \to & \mathrm{P}_2 & \to & \mathrm{RSP} \\ \mathrm{IPK} \times \mathrm{CMT} \times \mathrm{CH} \times \mathrm{RSP} & \to & \mathrm{V} & \to & \{\mathtt{T}, \mathtt{F}\} \end{array}$$

We further write $\mathrm{TR} = \mathrm{CMT} \times \mathrm{CH} \times \mathrm{RSP}$ for the transcript space of the ID protocol. A three-move ID protocol is correct if for all $(isk, ipk) \in [\mathrm{G}]$ and $(st, cmt) \in [\mathrm{P}_1(isk, ipk)]$ and $ch \in \mathrm{CH}$ and $rsp \in [\mathrm{P}_2(st, ch)]$ we have $\mathrm{V}(ipk, cmt, ch, rsp) = \mathtt{T}$.

Special soundness. A three-move ID protocol has *special soundness* if there exists an extraction algorithm that recovers the identification secret key from all (valid) same-commitment-different-challenge transcript pairs. Formally, the notion requires the existence of an algorithm E that takes an identification public key $ipk \in \mathrm{IPK}$ and two transcripts $T_1, T_2 \in \mathrm{TR}$ and outputs either a secret key $isk \in \mathrm{ISK}$ or the failure symbol $\perp \notin \mathrm{ISK}$. For correctness (of extraction) we require that for all $(isk, ipk) \in [\mathrm{G}]$ and $T_1, T_2 \in \mathrm{TR}$, if we write $T_i = (cmt_i, ch_i, rsp_i)$ and have that $\mathrm{V}(ipk, cmt_1, ch_1, rsp_1)$ and $\mathrm{V}(ipk, cmt_2, ch_2, rsp_2)$ evaluate to $\mathtt{T}$, then $cmt_1 = cmt_2 \wedge ch_1 \neq ch_2$ implies $\mathrm{E}(ipk, T_1, T_2) = isk$.

Honest-Verifier Zero-Knowledge. A three-move ID protocol is (perfectly) *honest-verifier zero-knowledge* if honestly generated transcripts leak nothing about the involved secret key material. Formally, the notion requires the existence of a simulator S that takes a public key $ipk \in \mathrm{IPK}$ and outputs a transcript $(cmt, ch, rsp) \in \mathrm{TR}$ such that for all $(isk, ipk) \in [\mathrm{G}]$ the distributions

$$\big\{(st, cmt) \leftarrow_{\$} \mathrm{P}_1(isk, ipk);\, ch \leftarrow_{\$} \mathrm{CH};\, rsp \leftarrow_{\$} \mathrm{P}_2(st, ch) : (cmt, ch, rsp)\big\}$$

and

$$\big\{(cmt, ch, rsp) \leftarrow_{\$} \mathrm{S}(ipk) : (cmt, ch, rsp)\big\}$$

are identical.

Resilience against Key Recovery. A three-move ID protocol ID is resilient against (blind) key recovery attacks if no adversary can reconstruct the identification secret key from just the identification public key (no sample transcripts are provided). Formally, for all inverters $\mathcal{I}$ we define the advantage $\mathrm{Adv}^{\mathrm{kr}}_{\mathsf{ID}}(\mathcal{I}) := \Pr[(isk, ipk) \leftarrow_{\$} \mathrm{G};\, isk' \leftarrow_{\$} \mathcal{I}(ipk) : isk = isk']$. Intuitively, an ID scheme is resilient against key recovery if all practical inverters have a negligible advantage.

5.2 The Fiat–Shamir transform

A well-known generic construction of a signature scheme from a three-move ID scheme and a random oracle is by Fiat and Shamir [10]. In a nutshell, for creating a signature on a message the signer invokes the P_1 algorithm to obtain a fresh commitment, simulates an (honest) verifier by letting the random oracle, on input the commitment and the message, specify a challenge, and finally invokes the P_2 algorithm to obtain a response that completes the transcript. The signature consists of the commitment and the response. The verification algorithm recovers the challenge by querying the random oracle and checks for transcript validity using the V algorithm. For reference we reproduce the details of this construction in the following.

Fiat–Shamir Transform. Let $\mathcal{M}$ be a message space and let $\mathrm{G}, \mathrm{P}_1, \mathrm{P}_2, \mathrm{V}$ be the algorithms and $\mathrm{ISK}, \mathrm{IPK}, \mathrm{CMT}, \mathrm{CH}, \mathrm{RSP}, \mathrm{ST}$ be the spaces of a three-move ID protocol. Let $H\colon \mathrm{IPK} \times \mathrm{CMT} \times \mathcal{M} \to \mathrm{CH}$ be a hash function. Then the *Fiat–Shamir* transform (FS) converts the ID protocol into a signature scheme Σ: After letting $\mathcal{SK} = \mathrm{ISK} \times \mathrm{IPK}$, $\mathcal{VK} = \mathrm{IPK}$, $\mathcal{S} = \mathrm{CMT} \times \mathrm{RSP}$, the algorithms $\mathsf{gen}, \mathsf{sgn}, \mathsf{vfy}$ of the scheme are as specified in Fig. 6.

Proc gen	**Proc** sgn(sk, m)	**Proc** vfy(vk, m, σ)
00 $(isk, ipk) \leftarrow_{\$} \mathrm{G}$	04 $(isk, ipk) \leftarrow sk$	10 $ipk \leftarrow vk$; $(cmt, rsp) \leftarrow \sigma$
01 $sk \leftarrow (isk, ipk)$	05 $(st, cmt) \leftarrow_{\$} \mathrm{P}_1(isk, ipk)$	11 $ch \leftarrow H(ipk, cmt, m)$
02 $vk \leftarrow ipk$	06 $ch \leftarrow H(ipk, cmt, m)$	12 $d \leftarrow \mathrm{V}(ipk, cmt, ch, rsp)$
03 Return (sk, vk)	07 $rsp \leftarrow_{\$} \mathrm{P}_2(st, ch)$	13 Return d
	08 $\sigma \leftarrow (cmt, rsp)$	
	09 Return σ	

Fig. 6. Signature scheme obtained via the Fiat–Shamir transform.

5.3 The Fixed-Commitment transform

We propose the Fixed-Commitment transform (FC) as an alternative way of constructing a signature scheme from a three-move ID protocol. It differs from the Fiat–Shamir transform in that generating a commitment using the P_1 algorithm happens only once and during key generation, instead of during signing operations. The challenge (of the ID protocol) continues to be a function of commitment and message. Thus, signatures on different messages share the same commitment but use different challenges, allowing for the extraction of the identification secret key via the special soundness property. By using a similar trick as in our SOT-to-DAP transform (see Section 4), i.e., by embedding a random oracle based encryption of the remaining signing key components into the verification key, the signature scheme is rendered strictly one-time. We specify the details of the FC transform in the following.

Fixed-Commitment Transform. Let $\mathcal{M}$ be a message space and let $\mathrm{G}, \mathrm{P}_1, \mathrm{P}_2, \mathrm{V}$ be the algorithms and $\mathrm{ISK}, \mathrm{IPK}, \mathrm{CMT}, \mathrm{CH}, \mathrm{RSP}, \mathrm{ST}$ be the spaces of a three-move ID protocol. Assume $(\mathrm{ST}, +)$ is a commutative group.[3] Let $H\colon \mathrm{IPK} \times \mathrm{CMT} \times \mathcal{M} \to \mathrm{CH}$ and $h\colon \mathrm{ISK} \to \mathrm{ST}$ be hash functions, both of which will be modeled as random oracles in the security analysis. Then the *Fixed-Commitment* transform (FC) converts the ID protocol into a signature scheme Σ: After letting $\mathcal{SK} = \mathrm{IPK} \times \mathrm{ST} \times \mathrm{CMT}$, $\mathcal{VK} = \mathrm{IPK} \times \mathrm{CMT} \times \mathrm{ST}$, $\mathcal{S} = \mathrm{RSP}$, the algorithms $\mathsf{gen}, \mathsf{sgn}, \mathsf{vfy}, \mathsf{ext}$ of the scheme are as specified in Fig. 7.

The signature schemes obtained with our transform provide both unforgeability and key extractability.

Theorem 3 (SOT unforgeability). *Let* ID *be a three-move ID protocol and let* Σ *be the signature scheme obtained from it via the Fixed-Commitment transform. If* ID *has special soundness, is honest-verifier zero-knowledge, and is resilient against key recovery, then* Σ *is unforgeable in the SOT sense. More precisely, for any adversary* $\mathcal{F}$ *against* Σ *there exists an inverter* $\mathcal{I}$ *such that in the random oracle model for* H *and* h *we have*

$$\mathrm{Adv}^{\mathrm{sot}}_{\Sigma}(\mathcal{F}) \leq Q \cdot \mathrm{Adv}^{\mathrm{kr}}_{\mathsf{ID}}(\mathcal{I}) + (q_H)^2/|\mathrm{CH}| \ ,$$

where $Q = q_H + q_h$ *and* q_H, q_h *are the numbers of queries to random oracles* H *and* h*, respectively. The running time of* $\mathcal{I}$ *is about that of* $\mathcal{F}$*.*

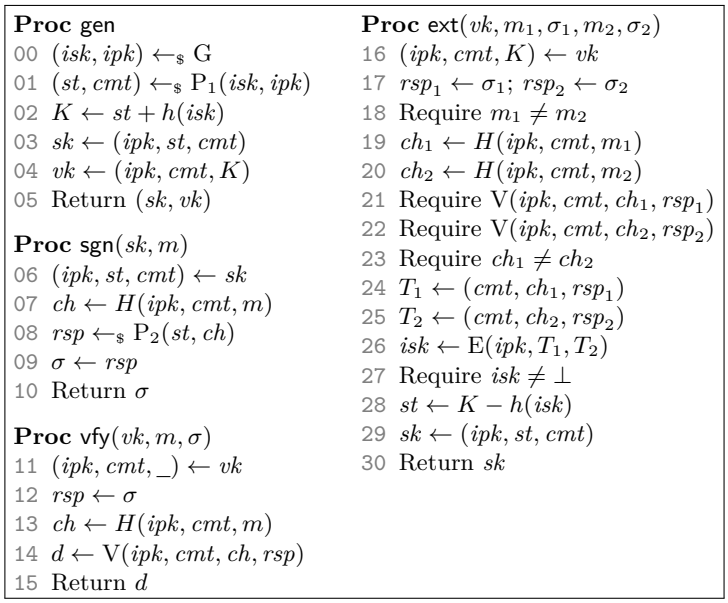

Fig. 7. SOT signature scheme obtained via the Fixed-Commitment transform. We write 'Require C' for a condition C as an abbreviation for 'If not C: Return $\perp$'. Note that the condition in line 27 is always fulfilled.

Proof sketch. Consider first the variant of the FC transform that does not embed encrypted state information in the verification key. With respect to this scheme, the simulator for game SOT has to provide the adversary with a verification key, a signature oracle that processes at most one query, and access to random oracle H. Insist w.l.o.g. that the adversary poses precisely one signing query, and that before it does so it poses the corresponding query to H. Let m be the message for which the signature is requested, and let $1 \leq i \leq q_H$ be the index of the corresponding H-query. The simulator proceeds as follows: it generates a key pair (isk, ipk) with G; it guesses an index $1 \leq j \leq q_H$ uniformly at random; it aborts, with probability $1 - 1/q_H$, if $j \neq i$; it generates a protocol transcript (cmt, ch, rsp) using the zero-knowledge simulator; it answers the jth H-query with challenge ch (all remaining H-queries are answered with uniformly picked challenges); it invokes the adversary on input the verification key composed of ipk and cmt. Note that the simulator can properly simulate a signature on message m, just by releasing rsp. Note further that with probability $1 - q_H/|\mathrm{CH}|$, for the challenge ch^* corresponding to the forgery output by the adversary we have $ch \neq ch^*$. As in this situation the special-soundness extraction algorithm is applicable to recover isk, a natural reduction shows that the forging advantage is bounded by $q_H \mathrm{Adv}^{\mathrm{kr}}_{\mathsf{ID}}(\mathcal{I}) + (q_H)^2/|\mathrm{CH}|$, for an inverter $\mathcal{I}$.

Consider next the full scheme that includes the encryption in the verification key. This additional information is completely useless to the adversary up to the

point where it poses a $h(isk)$ query. Each such query can be seen as trying to break a key recovery challenge against scheme ID. That is, to the above bound the term $q_h \mathrm{Adv}^{\mathrm{kr}}_{\mathsf{ID}}(\mathcal{I})$ needs to be added. The overall result is the bound claimed in the theorem statement. □

Theorem 4 (SOT key extractability). *Let* ID *be a three-move ID protocol and let* Σ *be the signature scheme obtained from it via the Fixed-Commitment transform. If* ID *has special soundness then* Σ *is strictly one-time. More precisely, if* H *is collision resistant then algorithm* ext *in Fig. 7 constructs an extraction algorithm for scheme* Σ *from the extraction algorithm* E *of scheme* ID.

Proc gen
00 $k \leftarrow_{\$} \{0,1\}^{\kappa}$
01 $vk[\cdot] \leftarrow \perp$; $K[\cdot] \leftarrow \perp$
02 For all $a \in \mathcal{A}$:
03 $(x, r) \leftarrow F(k, a)$
04 $X \leftarrow g^x$; $R \leftarrow g^r$
05 $vk[a] \leftarrow (X, R)$
06 $K[a] \leftarrow k + h(a, x, r)$
07 $sk \leftarrow k$
08 $vk \leftarrow (vk[\cdot], K[\cdot])$
09 Return (sk, vk)

Proc $\mathsf{sgn}(sk, m)$
10 $k \leftarrow sk$; $(a, p) \leftarrow m$
11 $(x, r) \leftarrow F(k, a)$
12 $X \leftarrow g^x$; $R \leftarrow g^r$
13 $c \leftarrow H(X, R, m)$
14 $\sigma \leftarrow r + xc$
15 Return σ

Proc $\mathsf{vfy}(vk, m, \sigma)$
16 $(vk[\cdot], \_) \leftarrow vk$
17 $(a, p) \leftarrow m$
18 $(X, R) \leftarrow vk[a]$
19 $c \leftarrow H(X, R, m)$
20 Return $[RX^c \stackrel{?}{=} g^{\sigma}]$

Fig. 8. DAP signature scheme based on Schnorr signatures, defined in respect to a cyclic group $\mathbb{G} = \langle g \rangle$ of prime-order q. We assume hash functions $F\colon \{0,1\}^{\kappa} \times \mathcal{A} \to \mathbb{Z}_q \times \mathbb{Z}_q$, $h\colon \mathcal{A} \times \mathbb{Z}_q \times \mathbb{Z}_q \to \{0,1\}^{\kappa}$, $H\colon \mathbb{G} \times \mathbb{G} \times \mathcal{M} \to \mathbb{Z}_q$.

Proof. The argument is immediate: Having two signatures on different messages means having two ID protocol transcripts with the same commitment but different challenges (this requires that hash function H be collision resistant, see line 23 in Fig. 7). Our SOT extraction algorithm applies the special-soundness extraction algorithm to this setting to first recover the identification secret key, and then, by decrypting the corresponding h-based ciphertext, the missing component of the SOT signing key. □

6 Putting things together: DLP-based DAP signatures

The overall goal of this article is to construct a practical DLP-based DAP signature scheme with short signatures. As in Section 4 we constructed DAP signatures generically from SOT signatures, and in Section 5 we constructed SOT signatures generically from ID schemes, what is missing is the specification of

appropriate DLP-based identification schemes. The classic candidates for this are the schemes by Schnorr [20] and Okamoto [15]. Both of them provide special soundness and honest-verifier zero-knowledge, and thus fit into our ID protocol framework. A less known and less general scheme is the one underlying DSA and ECDSA signatures [1,8] (which can be seen as a variant of Schnorr's scheme, obfuscated to avoid intellectual property issues).

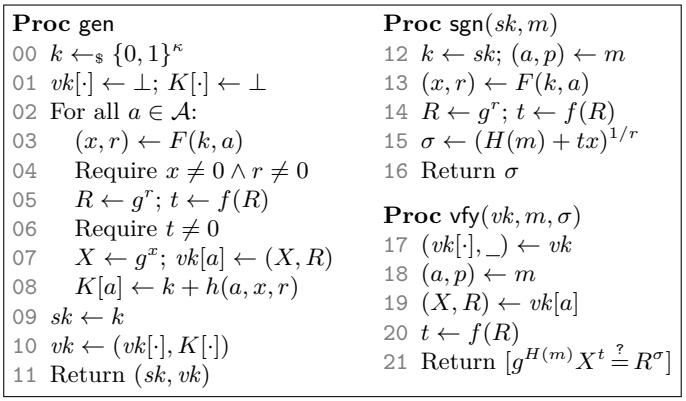

Fig. 9. DAP signature scheme based on (EC)DSA signatures (where we use the notation of the DSA algorithms from [9]). We assume a cyclic group $\mathbb{G}$ as in Fig. 8, and hash functions $F\colon \{0,1\}^{\kappa} \times \mathcal{A} \to \mathbb{Z}_q \times \mathbb{Z}_q$, $f\colon \mathbb{G} \to \mathbb{Z}_q$, $h\colon \mathcal{A} \times \mathbb{Z}_q \times \mathbb{Z}_q \to \{0,1\}^{\kappa}$, $H\colon \mathcal{M} \to \mathbb{Z}_q$. We write 'Require C' for a condition C as an abbreviation for 'If not C: Return $\bot$'.

In Figs. 8 and 9 we expose how our overall DAP scheme looks like if the three-move ID protocol is instantiated with the ones underlying Schnorr and (EC)DSA signatures, respectively. In both cases, targeting the 128 bit security level, we propose a PRF key length of $\kappa = 128$ bit and a group order of $2\kappa = 256$ bit. Further, for compactness of verification keys, we suggest using elliptic curve groups. In particular it would be somehow natural to instantiate the Schnorr-based scheme with the parameters of (Schnorr-based) EdDSA signatures [4] (i.e., on Edwards curves) and the (EC)DSA-based scheme with NIST-standardized curves.[4] In both cases the signature size would be 256 bit. No DAP signature scheme proposed in the past has that short signatures (so far, the shortest DAP schemes have 2048 bit [3] and 1280 bit [7] signatures, which we beat by a factor of 8 and 5, respectively), and likely the length is even optimal (in the DLP setting). The verification keys are considerably less compact, with a size of $640|\mathcal{A}|$ bits.

[4] That this is "natural" was communicated to us by software engineers. From an academic perspective the choice of curve should be orthogonal to the choice of ID scheme. On the other hand, there seems nothing wrong with the proposal, so we stick to it.

Note this is only slightly larger than those of [7] which are roughly $512|\mathcal{A}|$ bits in size.

References

1. Barker, E.B.: FIPS PUB 186-4—FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION Digital Signature Standard (DSS) (2009), https://dx.doi.org/10.6028/NIST.FIPS.186-4
2. Bellare, M., Poettering, B., Stebila, D.: From identification to signatures, tightly: A framework and generic transforms. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016, Part II. LNCS, vol. 10032, pp. 435–464. Springer, Heidelberg, Germany, Hanoi, Vietnam (Dec 4–8, 2016)
3. Bellare, M., Poettering, B., Stebila, D.: Deterring certificate subversion: Efficient double-authentication-preventing signatures. In: Fehr, S. (ed.) PKC 2017, Part II. LNCS, vol. 10175, pp. 121–151. Springer, Heidelberg, Germany, Amsterdam, The Netherlands (Mar 28–31, 2017)
4. Bernstein, D.J., Duif, N., Lange, T., Schwabe, P., Yang, B.Y.: High-speed high-security signatures. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 124–142. Springer, Heidelberg, Germany, Nara, Japan (Sep 28 – Oct 1, 2011)
5. Boneh, D., Kim, S., Nikolaenko, V.: Lattice-based DAPS and generalizations: Self-enforcement in signature schemes. In: Gollmann, D., Miyaji, A., Kikuchi, H. (eds.) ACNS 17. LNCS, vol. 10355, pp. 457–477. Springer, Heidelberg, Germany, Kanazawa, Japan (Jul 10–12, 2017)
6. Chaum, D., Fiat, A., Naor, M.: Untraceable electronic cash. In: Goldwasser, S. (ed.) CRYPTO'88. LNCS, vol. 403, pp. 319–327. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 21–25, 1990)
7. Derler, D., Ramacher, S., Slamanig, D.: Short double- and N-times-authentication-preventing signatures from ECDSA and more. Cryptology ePrint Archive, Report 2017/1203 (2017), https://eprint.iacr.org/2017/1203
8. Fersch, M., Kiltz, E., Poettering, B.: On the provable security of (EC)DSA signatures. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS 16. pp. 1651–1662. ACM Press, Vienna, Austria (Oct 24–28, 2016)
9. Fersch, M., Kiltz, E., Poettering, B.: On the one-per-message unforgeability of (EC)DSA and its variants. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017, Part II. LNCS, vol. 10678, pp. 519–534. Springer, Heidelberg, Germany, Baltimore, MD, USA (Nov 12–15, 2017)
10. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO'86. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 1987)
11. Goldreich, O.: Foundations of Cryptography: Basic Tools, vol. 1. Cambridge University Press, Cambridge, UK (2001)
12. Katz, J., Lindell, Y.: Introduction to Modern Cryptography. Chapman and Hall/CRC Press (2007)
13. Krawczyk, H., Rabin, T.: Chameleon signatures. In: NDSS 2000. The Internet Society, San Diego, CA, USA (Feb 2–4, 2000)
14. Menezes, A., van Oorschot, P., Vanstone, S.: Handbook of Applied Cryptography. CRC Press (2001), http://www.cacr.math.uwaterloo.ca/hac/

15. Okamoto, T.: Provably secure and practical identification schemes and corresponding signature schemes. In: Brickell, E.F. (ed.) CRYPTO'92. LNCS, vol. 740, pp. 31–53. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 16–20, 1993)
16. Poettering, B.: Shorter double-authentication preventing signatures for small address spaces. Cryptology ePrint Archive, Report 2018/223 (2018), https://eprint.iacr.org/2018/223
17. Poettering, B., Stebila, D.: Double-authentication-preventing signatures. In: Kutylowski, M., Vaidya, J. (eds.) ESORICS 2014, Part I. LNCS, vol. 8712, pp. 436–453. Springer, Heidelberg, Germany, Wroclaw, Poland (Sep 7–11, 2014)
18. Poettering, B., Stebila, D.: Double-authentication-preventing signatures. Int. J. Inf. Sec. 16(1), 1–22 (2017)
19. Ruffing, T., Kate, A., Schröder, D.: Liar, liar, coins on fire!: Penalizing equivocation by loss of bitcoins. In: Ray, I., Li, N., Kruegel:, C. (eds.) ACM CCS 15. pp. 219–230. ACM Press, Denver, CO, USA (Oct 12–16, 2015)
20. Schnorr, C.P.: Efficient identification and signatures for smart cards. In: Brassard, G. (ed.) CRYPTO'89. LNCS, vol. 435, pp. 239–252. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 20–24, 1990)