Double-Authentication-Preventing Signatures in the Standard Model

Dario Catalano[1], Georg Fuchsbauer[2], and Azam Soleimanian[3,4]

[1] Dipartimento di Matematica e Informatica – Università di Catania, Italy
[2] TU Wien, Austria
[3] Inria de Paris, France
[4] École normale supérieure, CNRS, PSL University, Paris, France
catalano@dmi.unict.it {georg.fuchsbauer,azam.soleimanian}@ens.fr

Abstract. A double-authentication preventing signature (DAPS) scheme is a digital signature scheme equipped with a self-enforcement mechanism. Messages consist of an address and a payload component, and a signer is penalized if she signs two messages with the same addresses but different payloads. The penalty is the disclosure of the signer's signing key. Most of the existing DAPS schemes are proved secure in the random oracle model (ROM), while the efficient ones in the standard model only support address spaces of polynomial size.
We present DAPS schemes that are efficient, secure in the standard model under standard assumptions and support large address spaces. Our main construction builds on vector commitments (VC) and double-trapdoor chameleon hash functions (DTC). We also provide a DAPS realization from Groth-Sahai (GS) proofs that builds on a generic construction by Derler et al., which they instantiate in the ROM. The GS-based construction, while less efficient than our main one, shows that a general yet efficient instantiation of DAPS in the standard model is possible.
An interesting feature of our main construction is that it can be easily modified to guarantee security even in the most challenging setting where no trusted setup is provided. It seems to be the first construction achieving this in the standard model.

Keywords: Double-spending · digital signature · cryptocurrencies · certificate subversion.

1 Introduction

Digital signatures (DS) are a cryptographic primitive that guarantees authenticity and integrity. Its security is defined via the notion of unforgeability, which protects the signer, and there is no notion of a signer behaving badly. There are however applications in which the signer should be restricted; for example, a certificate authority should not certify two different public keys for the same domain.

Double-authentication-prevention signatures (DAPS) are a natural extension of digital signatures that prevent malicious behavior of the signer by a self-enforcement strategy. A message for DAPS consists of two parts, called address and payload i.e., $m = (a, p)$. A signer behaves maliciously if it signs two messages with the same addresses but different payloads, that is, $m_1 = (a_1, p_1)$ and $m_2 = (a_2, p_2)$ with $a_1 = a_2$ and $p_1 \neq p_2$. Such a pair (m_1, m_2) is called *compromising* and the signer is penalized for signing a compromising pair by having its signing key revealed. We next discuss typical applications of DAPS.

Certificate Subversion. Consider a certificate authority (CA) issuing certificates to different severs. A certificate is of the form $(server.com, \mathsf{pk}_s, \sigma_s)$, where *server.com* is the domain, pk_s is the server's public key and σ_s is a signature on $(server.com, \mathsf{pk}_s)$ by the CA. Entities that trust the CA's public key can now securely communicate with *server.com* by using pk_s. Consider a national state court that has jurisdiction over the CA and compels it to issue a rogue certificate for *server.com* for a public key under the control of an intelligence agency. The latter can then impersonate *server.com* without its clients detecting the attack. Using DAPS for certification gives the CA a strong argument to deny the order, as otherwise its key is leaked. It leads to an all-or-nothing situation where if one certificate has been subverted then all have (as once the key is revealed, everything can be signed).

Cryptocurrencies and Non-Equivocation Contracts. In traditional e-cash systems (such as [CFN90]), double-spending is prevented by revealing the identity of the misbehaving party. This works well in systems where some central authority (e.g. a bank) can take concrete actions to penalize dishonest behaviors (such as blocking their accounts). In the setting of modern cryptocurrencies, disclosing the identity of users is much harder to implement mainly because of the decentralized nature of these systems, and indeed double-spending is typically prevented by consensus. Transactions are considered effective only after a certain amount of time. This naturally prevents double-spending but induces delays to reach agreement. Using DAPS to sign transactions could provide a strong deterrent to malicious behaviors. Double-spenders would disclose their secret signing keys, which, for the case of Bitcoin, translates to an immediate financial loss.

Translating to the DAPS setting, the address would be the coin and the payload the receiver when spending it. This is reminiscent to accountable assertions [RKS15] (who give a ROM instantiation). For cryptocurrencies it is natural to implement this mechanism via DAPS, as digital signatures are already needed to authenticate transactions.

Practically, one can make non-equivocation contracts (i.e. contracts that allow to penalize parties that make conflicting statements to others, by the loss of money) by combining a DAPS scheme and a deposit. To ensure that an extracted DAPS secret key is a worthy secret, it can be associated to a deposit. Each party is required to put aside a certain amount of currency in a deposit which can be securely retrieved by the owner at the end of a time-locked session if the owner has not made conflicting statements during the session. Otherwise, anyone obtaining the extracted secret key has access to the funds.

1.1 Challenges in Constructing DAPS

Here we discuss some general challenges in constructing DAPS regarding their security, efficiency and functionality. Our aim is to construct a scheme that achieves a good balance among them.

Exponentially large address space. The address part of a message for DAPS typically refers to a user/coin identity. Only allowing a polynomial number of predefined identities [DRS18b, Poe18] severely restricts the possible applications, while an exponential number of addresses practically amounts to no restrictions, as one can use a collision-resistant hash function to map into the address space.

Security against untrusted setup. DAPS schemes should satisfy two security notions. *Unforgeability* ensures that signatures from an honest signer (who does not sign compromising message pairs) are secure against an outside attacker. *Key extractability* requires that issuing signatures on compromising message pairs leaks the signer's signing key; the notion can be defined with respect to a trusted or untrusted setup. In the latter case, each signer generates its own key pair, while assuming a trusted setup, which generates and distributes key pairs to the signers, is arguably unrealistic.

The majority of existing DAPS constructions assumes a *trusted setup* [DRS18b, Poe18, PS14, BPS17]. Those that do not, are in the random oracle model [RKS15] or have polynomial-size signatures (w.r.t the length of the address) [BKN17] or only support small address spaces [DRS18b, Poe18].

Standard assumptions. While giving reductionist security proofs is the preferred method of gaining trust in a cryptosystem, these guarantees are only meaningful if the underlying hardness assumptions have been well-studied. Moreover, such analyses often rely on idealizations like assuming hash functions are random functions (in the ROM). Our schemes are proven secure from very well-studied assumptions (e.g. RSA, CDH) and we do not make idealizing assumptions in our proofs, i.e., they are in the standard model.

Efficient/concrete instantiations. Some prior DAPS schemes [DRS18a] that claim efficient size of the signatures or achieve others of the above properties are black-box constructions from building blocks whose instantiation is often left open. This can be a non-trivial task and leaves the concrete efficiency of such schemes unclear.

1.2 Our Contribution

In this paper we present new DAPS constructions that address all of the above challenges. In this sense, our main contributions are as follows.

Exponentially large address spaces without random oracles. Most of the existing DAPS schemes supporting an exponentially large address space need to rely on the RO heuristic (e.g. [RKS15, PS14, BKN17, BPS17]). For some of these constructions such as the one based on ID-protocols, the need for the RO assumption mainly comes from the transformation of an interactive protocol to an non-interactive version like the DAPS scheme of [Poe18]. In the other schemes it is due to the fact that the simulator cannot simulate the signatures in a security reduction. The RO assumption then lets the simulator program its responses without having access to the signing key [RKS15]. We circumvent many of the difficulties arising in previous works by combining vector commitments [CF13] and double-trapdoor chameleon hash functions [BCG07, CDFG08]. Our methodology follows the authentication tree approach also adopted in a previous work [RKS15]: a signature is simply an authenticated path from a leaf (the address part of the message) to a given root (the public key).

In previous work, the signer either had to create the whole tree in advance (thus limiting the address space to polynomial size) or use the random oracle to be able to deal with exponentially large address spaces. In our construction the signer creates the tree incrementally using the equivocation properties of the chameleon hash. Moreover, we prove our schemes secure by relying on the double-trapdoor property: the simulator will be able to issue signatures by knowing *only one* of the two trapdoors. If an adversary manages to create a forgery (or if it signs a compromising pair), our reduction uses this information to extract the *other* trapdoor with non-negligible probability. Moreover, we use vector commitments [CF13] to realize a "flat" authentication tree (i.e. a tree

Table 1: Comparison with related works. Here n is the bit length of the address, which we write as $n = h \cdot \log q$ for integers h and q; values n_0 and $q_0 = \text{poly}(n_0)$ are LWE parameters, and $\text{poly}_{\mathsf{NIZK}}$ stands for the proof size in the underlying NIZK system for statements of length n. The notation $|\mathbb{G}|$ stands for the size of group elements, λ_H is the bit length of the random oracle output, and N is an RSA modulus.

Scheme	$\lvert\mathsf{Sign}\rvert$	$\lvert\mathsf{vk}\rvert$	Assumption	ROM	address space	untrusted setup
[Poe18]	$\lvert\mathbb{G}\rvert$	$O(2^n)$	DLog	yes	poly.	no
[RKS15]	$q \cdot h \cdot \lvert\mathbb{G}\rvert$	$O(1)$	DLog	yes	exp.	yes
[PS14]	$(\lambda_H + 1) \cdot \log N$	$O(1)$	Fact	yes	exp.	no
[BPS17]	$\log N$	$O(1)$	Fact	yes	exp.	no
[BKN17]	$O(n_0^2 \log q_0)$	$O(n_0^4 \log^3 q_0)$	LWE/SIS	yes	exp.	yes
[DRS18b]	$\text{poly}_{\mathsf{NIZK}}(n)$	$O(2^n)$	DLog	yes	poly.	yes
[LGW+19]	$\log N$ or $2 \cdot \lvert\mathbb{G}\rvert$	$O(1)$	Fact or CDH	yes	exp.	yes
[DRS18a]	$\text{poly}_{\mathsf{NIZK}}(n)$	$O(1)$	PRF& OWF	yes	exp.	yes
DAPS-GS	$36n \cdot \lvert\mathbb{G}\rvert$	$O(1)$	SXDH	no	exp.	no
DAPS-VC-DTC	$3h \cdot \lvert\mathbb{G}\rvert$	q	CDH	no	exp.	no
DAPS-DTC	$q \cdot h \cdot \lvert\mathbb{G}\rvert$	$O(1)$	DLog	no	exp.	yes

with branching degree $q > 2$). Since both vector commitments [CF13] and double-trapdoor chameleon hash ([BCG07, CDFG08], see also our DTC scheme in Appendix D) can be realized under standard assumptions, the security of our schemes relies on the same assumptions (w.r.t. the trusted setup for the VC scheme).

Security in untrusted setup. Interestingly, our construction can be easily extended to the setting where no trusted setup is available. This comes at the cost of slightly longer signatures and is in sharp contrast to previous proposals that all rely on trusted setup (or random oracles). The basic intuition here is that double-trapdoor chameleon hash functions can be realized in an untrusted setup (see our DTC scheme in Appendix D), and substituting the vector commitments with a *standard* collision-resistant hash function, the construction highlighted above still works. The downside is that the produced signatures are now longer, as more values have to be stored in the authentication chain. While the DTC schemes suggested in [BCG07, CDFG08] are considered w.r.t. a trusted setup, here we present a DTC scheme without a trusted setup. The idea of our suggested DTC scheme was vaguely addressed in [CDFG08] (Section 3.1, full version), we present a concrete construction and prove the security of our suggested DTC in untrusted setup.

We also slightly generalize the definition of key-extractability such that the adversary can succeed even when it manages to produce compromising messages that do not reveal a specified sensitive information about the secret key (and not necessarily the full secret key).

A Groth-Sahai-based construction. As additional contribution of this paper we propose a DAPS construction from Groth Sahai proofs [GS08] (DAPS-GS), which builds upon a construction recently proposed by Derler et al. [DRS18a]. The scheme, which we will refer to as DAPS-DRS, supports an exponentially large address space and is based on NIZK proofs, which the authors instantiated in the random oracle model. We modify their construction so that the NIZK proof system can be instantiated with the Groth-Sahai proof system [GS08]. This system provides efficient NIZK proofs in the standard model and under standard assumptions, for a restricted class of languages.

An interesting difference between our DAPS-GS scheme and DAPS-DRS is that the latter uses a *value-key-binding* PRF, which prevents the signer from changing the PRF secret key. We assign this task to a commitment scheme and can therefore relax the requirements on the PRF to standard PRF security. The authors of DAPS-DRS instantiate their key-value binding PRF F with the block cipher LowMC. They thus need to make the (arguably non-standard) assumption that this block cipher is a value-key-binding PRF, as they do not present any proof for this. The commonality of our DAPS schemes (DAPS-VC-DTC and DAPS-GS) is that they are both in the standard model and support large address spaces. In fact, we instantiate the generic construction of [DRS18a] by Groth-Sahai NIZK proof system (as DAPS-GS) to provide a standard-model scheme and compare it against our main, more efficient (secure), DAPS-VC-DTC (DAPS-DTC).

As a final note, we remark that our solutions compare favorably to previous work not only in terms of security guarantees, but also in terms of efficiency. Our most efficient construction based on vector commitments also provides nice trade-offs between the size of its signatures and the size of its verification keys: verification keys grow with the branching degree q of the underlying authentication tree, while signatures grow with the depth h of the tree. A more precise comparison with previous works is given in Table 1.

1.3 Related work

Ruffing et al. [RKS15] presented a DAPS scheme based on Merkle trees and chameleon hash functions in the random oracle model. Their scheme can support an exponentially large address space by using a flat-tree construction. In their construction, each leaf is associated with a unique address. Some values are assigned on the fly to nodes from leaf to the root, and a signature is an authentication chain. This flat-tree construction and the idea of value assignments on the fly has also been used in other constructions [CD96, CG05].

Poettering [Poe18] gave a DAPS scheme based on a three-move identification scheme in the random oracle model. Another drawback of this scheme is that it only supports small address spaces, essentially because each address is associated with a verification key

Bellare et al. [BPS17] proposed a similar solution but managed to avoid the restriction to small address spaces by introducing a trapdoor-ID scheme. Their solution still relies on random oracles though, and it also requires a trusted setup.

Poettering and Stebila [PS14] presented a DAPS based on extractable 2-to-1 trapdoor functions (2:1-TF) in the random oracle model. A 2:1-TF is a trapdoor one-way function for which every element of the range has precisely two preimages and holding the trapdoor allows to easily invert the function. Again, the scheme requires a trusted setup and the ROM.

Boneh et al. [BKN17] proposed a lattice-based DAPS scheme based on the hardness of LWE and SIS problems. Their construction is a black-box DAPS scheme based on a variant of trapdoor dual-Regev encryption system [GPV08] resulting in non-linear-size signatures and verification keys. This construction also relies on random oracles.

Finally, Derler et al. [DRS18a] presented a general DAPS from non-interactive zero-knowledge proof systems that supports an exponentially large address space. While their generic construction does not rely on random oracles, their instantiation does. Since any NIZK in the *common reference string* (CRS) model relies on a trusted setup, so does any DAPS construction based on NIZK. If we do not accept non-falsifiable assumptions, then an alternative is NIZK in ROM, e.g., using the Fiat-Shamir transformation [FS87].

In [LGW+19], a signer is behaving maliciously if it issues at least three signatures and there is a compromising pair among them. The penalty, for the misbehaving signer, is milder as well where instead of the disclosure of the signing-key, a forgery on his previously signed messages would be possible by other participates in the system. More precisely, when the signer issues three signatures on (a, p_1), (a, p_2) and (a', p) (two first ones as the compromising pair), then a signature can be forged on (a', p^*). Meaning that, each double-spending (a, p_1), (a, p_2) allows other users to double-spend the fund a' which is already spent by the malicious signer. While this security notion still makes sense in the related applications, the scheme is proved secure in the ROM.

2 Preliminaries

Notations. We denote the security parameter by $\kappa \in \mathbb{N}$ and $x \leftarrow X$ means that element x is chosen uniformly at random from set X. If A is a probabilistic algorithm, $y \leftarrow A(\cdot)$ denotes the process of running A on some appropriate input and assigning its output to y. All algorithms run in probabilistic polynomial-time (p.p.t.) unless stated otherwise. We denote concatenation by $||$ and the set $\{1, \ldots, n\}$ by $[n]$. We say $\mathrm{negl}(\kappa)$ is a negligible function if for every positive polynomial $p(\kappa)$ there exists $\kappa_0 \in \mathbb{N}$ such that for all $\kappa > \kappa_0$: $\mathrm{negl}(\kappa) < 1/p(\kappa)$.

2.1 Digital Signatures

A digital signature (DS) scheme is defined as follows.

Definition 2.1 (Digital signature scheme). *A digital signature scheme is a tuple of* p.p.t. *algorithms* $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verif})$ *as follows:*

- $\mathsf{KeyGen}(1^\kappa)$*: On input security parameter* κ *in unary, it outputs a signing key* sk *and a verification key* vk *(which implicitly defines the message space* $\mathcal{M}$*).*
- $\mathsf{Sign}(\mathsf{sk}, m)$*: On input signing key* sk *and message* $m \in \mathcal{M}$ *this outputs a signature* σ*.*
- $\mathsf{Verif}(\mathsf{vk}, m, \sigma)$*: On input verification key* vk*, message* $m \in \mathcal{M}$ *and signature* σ*, this algorithm outputs either 0 or 1.*

Correctness. Signature scheme Σ is correct if for all $\kappa \in \mathbb{N}$ and for all $m \in \mathcal{M}$, where $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(1^\kappa)$ and $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$, we have $\Pr[\mathsf{Verif}(\mathsf{vk}, m, \sigma) = 1] = 1$, where the probability is taken over the coins of KeyGen and Sign.

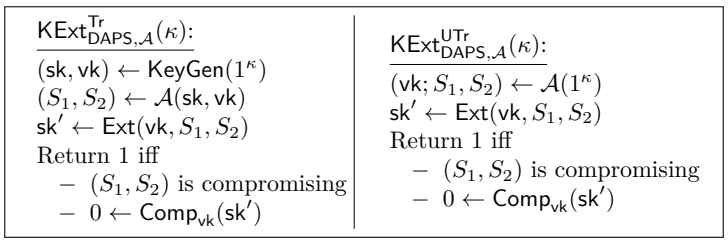

Fig. 1: Game for key-extractability of DAPS.

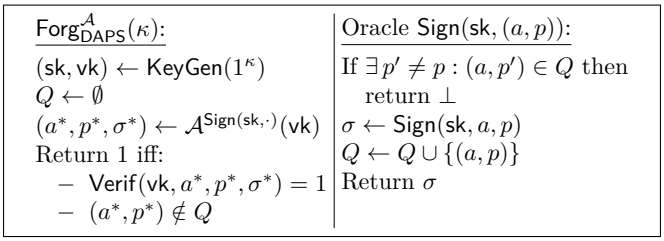

Fig. 2: Experiment for EUF-CMA security of DAPS

2.2 Double-Authentication-Preventing Signatures

Double-authentication-preventing signature (DAPS) schemes are a subclass of digital signatures where the message to be signed is split into two parts; an address and a payload[5] (i.e., in Definition 2.1, $m = (a, p) \in \mathcal{U} \times \mathcal{P}$).

Informally, compromising messages are (signed) pairs of messages with the same addresses but different payloads.

Definition 2.2 (Compromising pair of signatures [PS14]). *For a fixed verification key* vk*, a pair* (S_1, S_2) *where* $S_1 = (a_1, p_1; \sigma_1)$ *and* $S_2 = (a_2, p_2; \sigma_2)$*, is compromising if*

$$\mathsf{Verif}(\mathsf{vk}, (a_1, p_1), \sigma_1) = 1, \quad \mathsf{Verif}(\mathsf{vk}, (a_2, p_2), \sigma_2) = 1, \quad a_1 = a_2 \quad \text{and} \quad p_1 \neq p_2.$$

For a DAPS scheme, the key-extractability (KE) game is defined, where a malicious signer tries to produce a compromising pair of signatures that does not lead to the revelation of a signing key which is compatible with its verification key. To make our definition more general, we allow the adversary to succeed even when it manages to produce compromising messages that do not reveal sensitive information about the secret key (and not necessarily the full secret key).

This is captured via the Comp predicate that, informally, outputs 1 if the input is compatible with the public verification key. The exact meaning of this "compatible" depends on the specific application, but clearly for any $(\mathsf{sk},\mathsf{vk}) \leftarrow \mathsf{KeyGen}(1^\kappa)$ it should be the case that $1 \leftarrow \mathsf{Comp}_{\mathsf{vk}}(\mathsf{sk})$.

Definition 2.3 (Key extractability [PS14]). *A DAPS scheme is key extractable if there exists a* p.p.t. *algorithm* Ext*, as follows:*

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

such that for all adversaries $\mathcal{A}$*, there is a negligible function* $\mathrm{negl}(\kappa)$ *such that,* $\Pr[\mathsf{KExt}^{\mathsf{Tr/UTr}}_{\mathsf{DAPS},\mathcal{A}}(\kappa) = 1] \leq \mathrm{negl}(\kappa)$ *where the experiment* $\mathsf{KExt}^{\mathsf{Tr/UTr}}_{\mathsf{DAPS},\mathcal{A}}(\kappa)$ *is as described in Fig. 1.*

We say that a DAPS scheme is KE in trusted setup if this holds for experiment $\mathsf{KExt}^{\mathsf{Tr}}_{\mathsf{DAPS},\mathcal{A}}$*, and it is KE in untrusted setup if it holds for* $\mathsf{KExt}^{\mathsf{UTr}}_{\mathsf{DAPS},\mathcal{A}}$*.*

Since DAPS schemes are a subclass of digital signatures, the standard existential unforgeability (Definition A.1) should also be satisfied for a DAPS scheme. This requires a restriction though, as the adversary could obtain the signing key if it was allowed to query compromising pairs to its signing oracle.

Definition 2.4 (Unforgeability of DAPS). *A DAPS scheme* Σ *is existentially unforgeable under adaptive chosen-message attacks (EUF-CMA) if for all* p.p.t. *adversaries* $\mathcal{A}$ *there exists a negligible function* negl *such that* $\Pr[\mathsf{Forg}^{\mathcal{A}}_{\mathsf{DAPS}}(\kappa) = 1] \leq \mathrm{negl}(\kappa)$*, where* $\mathsf{Forg}^{\mathcal{A}}_{\mathsf{DAPS}}(\kappa)$ *is described in Fig. 2.*

[5] In [PS14] these two parts are referred as subject and message, and in [RKS15] as context and statement. Here we are following the terminologies from [Poe18])

2.3 Vector Commitments

A vector commitment (VC) is a primitive allowing to commit to an ordered sequence of q values, rather than to single messages [CF13]. One can later open the commitment at a specific position.

Definition 2.5 (Vector commitments [CF13]). *A VC scheme is a tuple of* p.p.t. *algorithms* $\mathsf{VC} = (\mathsf{Setup}, \mathsf{Cmt}, \mathsf{Open}, \mathsf{Verif})$ *as follows:*

- $\mathsf{Setup}(1^\kappa, q)$*: On input security parameter* κ *and the length* q *of the committed vector (with* $q = poly(k)$*), outputs public parameters* pp *(which are an implicit input to all algorithms and define the message space* $\mathcal{M}$*).*
- $\mathsf{Cmt}_{\mathsf{pp}}(m_1, \ldots, m_q)$*: On input a sequence of* q *messages* $m_1, \ldots, m_q \in \mathcal{M}$*, it outputs a commitment string* C*.*
- $\mathsf{Open}_{\mathsf{pp}}(m_1, \ldots, m_q; m, i)$*: It produces a proof* Λ_i *that* m *is the* i*-th committed message in the sequence* $m_1, \ldots m_q$*.*
- $\mathsf{Verif}_{\mathsf{pp}}(C, m, i, \Lambda_i)$*: Outputs 1 iff* Λ_i *is a valid proof that* C *commits to a sequence* $m_1, \ldots, m_q$ *with* $m = m_i$*.*

Definition 2.6 (Correctness of VC). *A* VC *is correct if, for all* $\kappa \in \mathbb{N}$ *and* q*, and for all vectors* $(m_1, \ldots, m_q) \in \mathcal{M}^q$*, we have*

$$\Pr\left[\begin{array}{l}\mathsf{pp} \leftarrow \mathsf{Setup}(1^\kappa, q) \\ C \leftarrow \mathsf{Cmt}_{\mathsf{pp}}(m_1, \ldots, m_q) \\ \Lambda_i \leftarrow \mathsf{Open}_{\mathsf{pp}}(m_i, i)\end{array} : \mathsf{Verif}_{\mathsf{pp}}(C, m_i, i, \Lambda_i) = 1\right] = 1.$$

The security requirement for a VC scheme is *position binding* and requires that for any p.p.t. adversary holding pp, it should be infeasible to produce a commitment C and openings to two different messages for the same position.

Definition 2.7 (Position binding). *A VC scheme* VC *satisfies position binding if for all* $i \in \{1, \ldots, q\}$ *and* p.p.t. *adversary* $\mathcal{A}$ *we have* $\Pr[\mathsf{PBind}^{\mathcal{A}}_{\mathsf{VC}}(\kappa) = 1] \leq \mathrm{negl}(\kappa)$ *where the game* $\mathsf{PBind}^{\mathcal{A}}_{\mathsf{VC}}(\kappa)$ *is defined as Fig. 3.*

$\underline{\mathsf{PBind}^{\mathcal{A}}_{\mathsf{VC}}(\kappa)}$:
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\kappa)$
$(C, m, m', i, \Lambda, \Lambda') \leftarrow \mathcal{A}(\mathsf{pp})$
Return 1 iff $m \neq m'$ and
$\mathsf{Verif}_{\mathsf{pp}}(C, m, i, \Lambda) = 1$, $\mathsf{Verif}_{\mathsf{pp}}(C, m', i, \Lambda') = 1$

Fig. 3: Game for position-binding of a VC scheme

Finally, a VC scheme is *concise* if the size of the commitment string C and the output of algorithm Open are both independent of q.

2.4 Double-Trapdoor Chameleon Hash Functions

A double-trapdoor chameleon (DTC) hash function scheme has two independent trapdoors such that knowing either of them lets one find collisions efficiently.

Definition 2.8 (DTC hash function [CDFG08]). *A DTC scheme* $\mathcal{H}$ *is a tuple of* p.p.t. *algorithms* $\mathcal{H} = (\mathsf{KeyGen}, \mathsf{TrChos}, \mathsf{CHash}, \mathsf{Coll})$ *defined as follows.*

- $\mathsf{KeyGen}(1^\kappa)$*: On input the security parameter* κ*, it outputs a triple of public/private keys* $(\mathsf{pk}, \mathsf{tk}_0, \mathsf{tk}_1)$ *(which implicitly defines the message space* $\mathcal{M}$*).*
- $\mathsf{TrChos}(1^\kappa, i)$*: On input the security parameter* κ *and a bit* i*, it outputs a pair of public/private keys* $(\mathsf{pk}, \mathsf{tk}_i)$*.*
- $\mathsf{CHash}_{\mathsf{pk}}(m, r)$*: On input the public key* pk*, a message* $m \in \mathcal{M}$ *and one (or more) random nonce* $r \in \mathcal{R}$*, it outputs a hash value.*
- $\mathsf{Coll}(\mathsf{pk}, \mathsf{tk}_i, m', m, r)$*: On input one of the two trapdoor keys* tk_i*, two messages* m, m' *and a nonce* r*, it outputs a nonce* r' *such that* $\mathsf{CHash}_{\mathsf{pk}}(m, r) = \mathsf{CHash}_{\mathsf{pk}}(m', r')$*.*

In the definition of algorithm Coll, the pair (m, r) and (m', r') is called a collision pair where $\mathsf{CHash}_{\mathsf{pk}}(m, r) = \mathsf{CHash}_{\mathsf{pk}}(m', r')$. For a DTC scheme, the following security requirements were given [CDFG08].

Definition 2.9 (Security of DTC).
Distribution of keys. *Let* $\mathsf{KeyGen}(1^\kappa, i)$ *be the algorithm that executes the algorithm* $\mathsf{KeyGen}(1^\kappa)$ *and restricts*

its output to $(\mathsf{pk}, \mathsf{tk}_i)$*. It is required that the distribution of the output of* $\mathsf{KeyGen}(1^\kappa, i)$ *is identical to the distribution of the output of* $\mathsf{TrChos}(1^\kappa, i)$*.* [6]

Collision-resistance (CR). *Let* $(\mathsf{pk}, \mathsf{tk}_0, \mathsf{tk}_1) \leftarrow \mathsf{KeyGen}(1^\kappa)$*. For every* $i = 0, 1$*, given* pk *and* tk_i *it is infeasible to find* tk_{1-i}*. Formally, in the game* $\mathsf{DTColl}^{\mathcal{A}}_{\mathsf{DTC}}$ *depicted in Fig. 4, for a* DTC *scheme and any* p.p.t. *adversary* $\mathcal{A}$*, there exists an negligible function* negl *such that,* $\Pr[\mathsf{DTColl}^{\mathcal{A}}_{\mathsf{DTC}}(\kappa) = 1] \leq \mathrm{negl}(\kappa)$*.*

Key-extractability (KE). *We say that a DTC scheme is key-extractable if there exists a* p.p.t. *algorithm* Ext*, as follows;*

- $\mathsf{Ext}(\mathsf{pk}, S_1, S_2)$*: On input the public key* pk *and a collision pair* (S_1, S_2)*, outputs a (single) secret key* tk'*.*

such that for all adversaries $\mathcal{A}$*, there is a negligible function* $\mathrm{negl}(\kappa)$ *such that,* $\Pr[\mathsf{KExt}^{\mathsf{Tr}\backslash\mathsf{UTr}}_{\mathsf{DTC},\mathcal{A}}(\kappa) = 1] \leq \mathrm{negl}(\kappa)$ *where the experiment* $\mathsf{KExt}^{\mathsf{Tr}\backslash\mathsf{UTr}}_{\mathsf{DTC},\mathcal{A}}(\kappa)$ *is as Fig. 5.*

Distribution of collisions. *For every* m, m'*, and a uniform-random* r*, the distributions of* $r' = \mathsf{Coll}(\mathsf{tk}_i, m, m', r)$ *for* $i = 0, 1$*, are identical (uniform), even when given* $\mathsf{CHash}_{\mathsf{pk}}(m, r)$*,* m *and* m'*.*

$\underline{\mathsf{DTColl}^{\mathcal{A}}_{\mathsf{DTC}}(\kappa):}$
$(\mathsf{pk}, \mathsf{tk}_0, \mathsf{tk}_1) \leftarrow \mathsf{KeyGen}(1^\kappa), \quad b \xleftarrow{R} \{0,1\}$
$\mathsf{tk}' \leftarrow \mathcal{A}(\mathsf{pk}, \mathsf{tk}_b)$
Output 1 iff $\mathsf{tk}' = \mathsf{tk}_{1-b}$.

Fig. 4: Collision-resistance game for a DTC scheme

2.5 Non-interactive Zero-Knowledge proof system

Let $L = \{x \mid \exists\, w : R(x, w) = 1\}$ be a language in NP. A non-interactive zero knowledge (NIZK) proof system for L is formally defined as follows.

Definition 2.10 (NIZK proof system). *A NIZK proof system* Π *consists of three* p.p.t. *algorithms* $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verif})$ *where*

- $\mathsf{Setup}(1^\kappa)$ *takes the security parameter* κ *as input and outputs a common reference string* crs *(which implicitly defines the language* L*).*
- $\mathsf{Prove}(\mathsf{crs}, x, w)$*, takes the* crs*, a statement* x *and a witness* w *as input and outputs a proof* π*.*
- $\mathsf{Verif}(\mathsf{crs}, x, \pi)$ *takes the* crs*, the statement* x*, and a proof* π *as input and outputs a bit 0/1.*

For a NIZK proof system $\Pi = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verif})$ some security requirements are defined and used in Appendix E.

[6] One is running $\mathsf{KeyGen}(1^\kappa, i)$ has the knowledge of both trapdoors (or there exist an algorithm $\mathsf{KeyGen}(1^\kappa)$ with such knowledge), while for TrChos is not so. Thus, the "key-distribution" property here, is simply saying that you still can run the algorithms CHash, Coll of DTC scheme only by knowledge of one of the trapdoors and pk.

$\underline{\mathsf{KExt}^{\mathsf{Tr}}_{\mathsf{CHF},\mathcal{A}}(\kappa):}$	$\underline{\mathsf{KExt}^{\mathsf{UTr}}_{\mathsf{CHF},\mathcal{A}}(\kappa):}$
$(\mathsf{pk}; \mathsf{tk}_0, \mathsf{tk}_1) \leftarrow \mathsf{KeyGen}(1^\kappa)$	$(\mathsf{pk}; S_1, S_2) \leftarrow \mathcal{A}(1^\kappa)$
$(S_1, S_2) \leftarrow \mathcal{A}(\mathsf{pk}, \mathsf{tk}_0, \mathsf{tk}_1)$	$\mathsf{tk}' \leftarrow \mathsf{Ext}(\mathsf{pk}, S_1, S_2)$
$\mathsf{tk}' \leftarrow \mathsf{Ext}(\mathsf{pk}, S_1, S_2)$	Outputs 1 iff
Outputs 1 iff	-(S_1, S_2) is a collision pair
-(S_1, S_2) is a collision pair	- $\mathsf{tk}' \neq \mathsf{tk}_0$ and $\mathsf{tk}' \neq \mathsf{tk}_1$.
- $\mathsf{tk}' \neq \mathsf{tk}_0$ and $\mathsf{tk}' \neq \mathsf{tk}_1$.	

Fig. 5: KE game for a DTC scheme. The left game is in the trusted setup and the right game is in the untrusted setup.

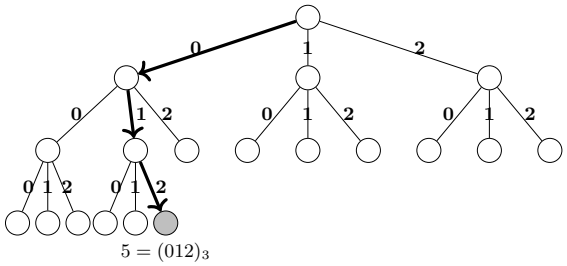

Fig. 6

3 A DAPS Scheme From VC and DTC (DAPS-VC-DTC)

3.1 Construction

In this section we present our DAPS scheme based on vector commitments and double-trapdoor chameleon hash function. Our scheme will make use of a "flat" q-ary tree (i.e. a tree with branching degree q) of height h, which we call the signing tree. The root of the tree will be a public value C_ϵ that will be part of the public key. Recall that in DAPS, messages are tuples of the form (a, p). The a component is interpreted as an integer in the range $\{0, \ldots, q^h - 1\}$. We can univocally associate each a to a path connecting the root with one leaf of the tree. In particular, a can be viewed as a number representing the labeling of the leaf in q-ary encoding (see the toy example in Fig. 6 for $q = 3$). In what follows $\mathsf{path}_{u \to w}$ denotes the ordered sequence of nodes connecting node u to node w. The root will be denoted by ϵ. Note that each node u has a unique position in the tree which we denote by pos_u. In fact, when u is the i-th node, from left-to-right, in the j-th level of the tree, we set $\mathsf{pos}_u = j||i$.

Overview of the scheme. We start with an informal description of the stateful version of our scheme where the signer needs to keep track of the produced signatures. The public key contains the public parameters for a vector commitment scheme VC and for a double-trapdoor chameleon hash function DTC. The private key is the corresponding trapdoor information. The public key also contains a value C_ϵ which is computed as follows. The signer first computes CHash on q random inputs to get the output values $m_1, \ldots, m_q$. Next, she sets C_ϵ as the value obtained when using VC to commit to the vector $m_1, \ldots m_q$. The value C_ϵ is assigned to the root of an, initially empty, tree.

Informally, the signature algorithm will "fill up" this tree on the fly. To sign the message (a, p), the signer will place p as the a-th leaf and will output an authentication chain that links p to the root C_ϵ. The verifier will follow this authentication chain and if the end of the chain matches the value in the public key, accepts the signature. Now we describe more in detail how the signer creates the authentication chain. Starting from the a-th leaf the signer produces the signature by augmenting the existing tree with the new authentication path. This is done using the following create and connect approach. First, the signer generates the possibly missing portion of the subtree containing the a-th leaf. Next, it connects this portion to signing tree. Creating the missing portion of the tree essentially amounts to creating all the missing nodes. Specifically, and letting $a = (a_0, \ldots, a_h)$ be the q-ary encoding of a, the signer computes $m_{a_h} = \mathsf{CHash}(p_h, r)$ where $p_h = p$ (for some randomness r) and, for $i \in \{0, \ldots, q\} \setminus \{a_h\}$, $m_i = \mathsf{CHash}(\$_i, r_i)$ for random $\$_i, r_i$. Next the signer computes $p_{h-1} = \mathsf{Cmt}(\boldsymbol{m})$ with $\boldsymbol{m} = (m_0, \ldots, m_q)$. The process is then repeated in a bottom-up fashion, until no more nodes need to be created. This happens when the newly created p_j needs to be inserted in a position already occupied by some other value $\$_j \neq p_j$ (i.e. for which a value $\mathsf{CHash}(\$_j, r_j)$ was previously computed). This is when the connect phase begins. The signer uses knowledge of the trapdoor key to find a "colliding" r such that $\mathsf{CHash}(\$_j, r_j) = \mathsf{CHash}(p_j, r)$.

In Fig. 7 we provide a pictorial representation of the key generation phase (for the toy case with branching degree 3). Black nodes indicate values that are obtained as outputs of the (vector) commitment and will not be altered any further in the future. The y_{ij} values are interpreted as outputs of $\mathsf{CHash}(\$_{ij}, r_{ij})$. Gray nodes indicate the frontier of the tree, i.e., nodes that are either leaves or roots of subtrees not yet explored.

Similarly, Fig. 8 pictorially describes (a toy example of) the signing procedure. To sign the message $(a_1 = 000, p_1)$, one first creates the missing part of the tree, as sketched above. This also requires the signer to store all the commitments associated to each node. Once the procedure reaches a frontier node, the signer uses knowledge of the trapdoor to find a collision for CHash. In Fig. 8 this is what happens to node 1 that, once connected with the newly created subtree, becomes black. Notice that the collision finding procedure typically alters the associated randomness $\hat{r}_{11}$. This change will be done once and for all at this stage, as once the node is blackened no further modifications are allowed. To complete the procedure, the signer also produces valid openings Λ for all the commitments encountered in the path from the leaf p_1 to the root and updates the lists of data associated to each node.

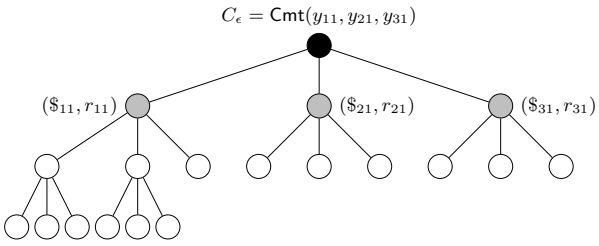

Fig. 7

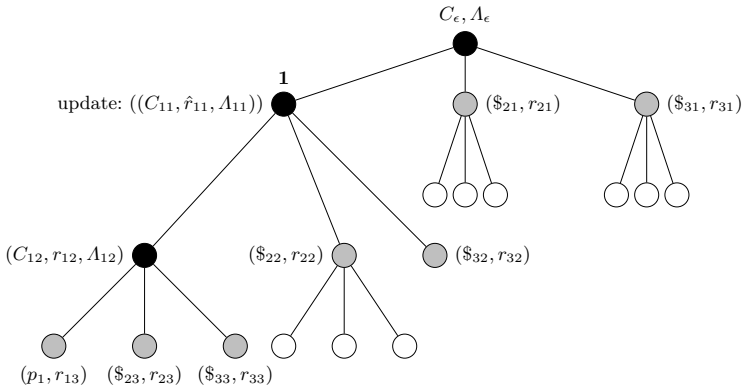

Fig. 8

Formal description of the scheme. We are now ready to present a formal description of our scheme. Let $\mathsf{VC} = (\mathsf{VC.Setup}, \mathsf{Cmt}, \mathsf{Open}, \mathsf{VC.Verif})$ be a vector commitment (VC) scheme and $\mathsf{DTC} = (\mathsf{DTC.KeyGen}, \mathsf{Trapdr}, \mathsf{CHash}, \mathsf{Coll})$ a double-trapdoor chameleon (DTC) hash function such that $\mathsf{Cmt} : \mathcal{M}^q \to \mathcal{O}_{vc}$ and $\mathsf{CHash} : (\mathcal{P}, \mathcal{R}) \to \mathcal{O}_{dtc}$. Let $H_1 : \mathcal{O}_{vc} \to \mathcal{P}$ and $H_2 : \mathcal{O}_{dtc} \to \mathcal{M}$ be two collision-resistant hash functions. The underlying data structure will be a tree T of branching degree q and height h. Messages are tuples of the form (a, p) with payload component $p \in \mathcal{P}$ and where a is an element in the address space $\mathcal{U} = \{0, \ldots, q^h - 1\}$. We use the label ϵ for the root. Also, we say that v is a child of u in position i, when v is the i-th child of u (counting from left). A pseudo-random function F is used to generate the random values $(\$_{ij}, r_{ij})$ as $\$_{ij} = F_k(j||i, 0)$ and $r_{ij} = F_k(j||i, 1)$ (to avoid keeping track of used randomnesses) where k is part of the signing key. And as we defined, $\mathsf{pos}_u = j||i$ denotes the unique position of node u in the tree. A complete description of the scheme is given in Fig. 9.

Remark 3.1. Note that intuitively for any DAPS scheme, signing is inherently stateful as the signer needs to remember the signed addresses in order not to sign a compromising pair. This is important for unforgeability rather than for correctness. Keeping track of the signed addresses can be efficiently done using a bloom filter [RKS15].

3.2 Security Analysis

We prove the security of our construction via the following two theorems, first key-extractability (Definition 2.3), then unforgeability (Definition 2.4). For the key-extractability of our DAPS scheme, we show that if a malicious signer tries to sign a compromising pair of messages, then one of the trapdoors of DTC will be revealed. Formally, we have the following theorem where we consider the KE property of our DAPS scheme, w.r.t the predicate $\mathsf{Comp}_{\mathsf{pk}}(\cdot)$ where $\mathsf{Comp}_{\mathsf{pk}}(\mathsf{tk}') = 1$ iff $\mathsf{tk}' = \mathsf{tk}_0$ or $\mathsf{tk}' = \mathsf{tk}_1$ and tk_i is the trapdoor of DTC scheme.

Theorem 3.2. *If* VC *is position-binding,* DTC *is key-extractable and H_1, H_2 are collision-resistant hash functions, then our construction (Fig. 9) is key-extractable w.r.t. the predicate* $\mathsf{Comp}_{\mathsf{pk}}(\cdot)$ *(as above, and w.r.t. the trusted setup for VC). More precisely,*

$$\Pr[\mathsf{KExt}^{\mathsf{Tr}}_{\mathsf{DAPS},\mathcal{A}}(\kappa) = 1] \leq \Pr[\mathsf{HColl}^{\mathcal{B}}_{H_1}(\kappa) = 1]$$
$$+ 3(\Pr[\mathsf{KExt}^{\mathsf{Tr/UTr}}_{\mathsf{DTC},\mathcal{C}}(\kappa) = 1] + \Pr[\mathsf{HColl}^{\mathcal{B}'}_{H_2}(\kappa) = 1] + \Pr[\mathsf{PBind}^{\mathcal{D}}_{\mathsf{VC}}(\kappa) = 1]).$$

Proof-sketch. The general intuition of the proof is as follows. By contradiction, we assume that a malicious signer can produce two valid signatures for a compromising pair without leaking any of her secret keys. We note that for any compromising pair, the authentication path to the root is the same and the end of the chain is fixed via the verification key. This means, the only way that a malicious signer can issue two valid signatures for a compromising

$\underline{\mathsf{KeyGen}(1^\kappa, q, h,)}$:

- choose $k \xleftarrow{R} \mathcal{K}_F$ and hash function families H_1 and H_2.
- run $\mathsf{pp_{VC}} \leftarrow \mathsf{VC.Setup}(1^\kappa)$ and $(\mathsf{pk_{DTC}}, \mathsf{tk}_0, \mathsf{tk}_1) \leftarrow \mathsf{DTC.KeyGen}(1^\kappa)$
- set $\$_{i1} = F_k(1||i, 0)$ and $r_{i1} = F_k(1||i, 1)$, for $i = 1, \ldots, q$
- compute the DTC-value $m_i = H_2(\mathsf{CHash}(\$_i, r_i)),\ i = 1, \ldots, q$
- compute the VC-value $C_\epsilon = \mathsf{Cmt}_{\mathsf{pp_{VC}}}(m_1, \ldots, m_q)$
- Return $\mathsf{vk} = (H_1, H_2, \mathsf{pp_{VC}}, \mathsf{pk_{DTC}}, C_\epsilon)$ and $\mathsf{sk} = (\mathsf{tk}_0, \mathsf{tk}_1, k)$.

$\underline{\mathsf{Sign}(\mathsf{vk}, \mathsf{sk}, (a, p))}$:

- set $\sigma = \emptyset$

- **Frontier-node.** let $u^* \in \mathsf{path}_{a\to\epsilon}$ be the frontier-node of the existing part of the tree (the first node on $\mathsf{path}_{a\to\epsilon}$ such that DTC-value is assigned).
- **Creation Phase.** (to create the subtree rooted in u^*) for $u \in \mathsf{path}_{a\to u^*}$ (except u^*):
  1. if u is a leaf: set $r_u = F_k(\mathsf{pos}_u, 1)$, $p_u = p$, $m_u = H_2(\mathsf{CHash}(p_u, r_u))$ and $\sigma := \sigma||r_u$.
     if u is not a leaf: set $C_u = \mathsf{Cmt}(m_1, \ldots, m_q)$ where m_i is assigned to the ith child of u, set also $p_u = H_1(C_u)$, $r_u = F_k(\mathsf{pos}_u, 1)$, $m_u = H_2(\mathsf{CHash}(p_u, r_u))$ and $\sigma = \sigma||(r_u, C_u, \Lambda_u)$ where $\Lambda_u = \mathsf{Open}(m_v, i)$ such that $v \in \mathsf{path}_{a\to\epsilon}$ is the i-th child of u.
  2. for each sibling v of u, set $\$_v = F_k(\mathsf{pos}_v, 0)$ and $r_v = F_k(\mathsf{pos}_v, 1)$, then compute $m_v = H_2(\mathsf{CHash}(\$_v, r_v))$.
- **Connection Phase.**(at node u^*)
  1. if u^* is a leaf: set $p_{u^*} = p$, run $\hat{r}_{u^*} \leftarrow \mathsf{Coll}(\mathsf{tk}_i, (\$_{u^*}, r_{u^*}), p_{u^*})$ (for $i \in \{0, 1\}$) and set $\sigma||\hat{r}_{u^*}$.
     if u^* is not a leaf: compute $C_{u^*} = \mathsf{Cmt}(m_1, \ldots, m_q)$ and set $p_{u^*} = H_1(C_{u^*})$. Then run $\hat{r}_{u^*} \leftarrow \mathsf{Coll}(\mathsf{tk}_i, (\$_{u^*}, r_{u^*}), p_{u^*})$ and set $\sigma||(\hat{r}_{u^*}, C_{u^*}, \Lambda_{u^*})$ where $\Lambda_{u^*} = \mathsf{Open}(m_v, i)$ such that $v \in \mathsf{path}_{a\to\epsilon}$ is the i-th child of u^*.
  2. let w be the parent of u^*, update Λ_w as $\Lambda_w \leftarrow \mathsf{Open}(m_{u^*}, i)$ such that u^* is i-th child of w. If $w \neq \epsilon$, set $\sigma := \sigma||\sigma_{w\to\epsilon}$ where $\sigma_{w\to\epsilon}$ is the authentication-chain from w to the root(with updated Λ_w). Else, set $\sigma := \sigma||\Lambda_w$

- Return σ

$\underline{\mathsf{Verif}(\mathsf{vk}, (a, p), \sigma)}$:

- parse vk as $(H_1, H_2, \mathsf{pp_{VC}}, \mathsf{pk_{DTC}}, C_\epsilon)$ and σ as $(r_h, (r_{h-1}, C_{h-1}, \Lambda_{h-1}), \ldots, (r_{\epsilon+1}, C_{\epsilon+1}, \Lambda_{\epsilon+1}), \Lambda_\epsilon)$. Then set $\mathsf{path}_{a\to\epsilon} = \{a_h, a_{h-1}, \ldots, \epsilon\}$.
- Set $m_h = H_2(\mathsf{CHash}(p, r_h))$ and $m_j = H_2(\mathsf{CHash}(H_1(C_j), r_j))$ for $j = h-1, \ldots, \epsilon+1$
- For $j = h-1, \ldots, \epsilon$: check that $\mathsf{VC.Verif}(C_j, m_{j+1}, i, \Lambda_j) = 1$ where node a_{j+1} is the i-th child of node a_j.
- if all the verifications pass, return 1, otherwise return 0.

Fig. 9: Our DAPS-VC-DTC scheme.

pair is to find a "collision node" on the path. That is, the values of the authentication chains for these two signatures should collide at some point and be equal after. Intuitively, due to the security of hash functions H_1, H_2 and the position-binding property of the VC scheme (which only rely on public parameters over which the signer has no control), the signer must find a collision node by creating a DTC-collision. Since this DTC-collision can be extracted through the signatures, then by key-extractability of DTC, one of the DTC-trapdoors would be revealed. We make this intuition formal in Appendix B. It remains to prove unforgeability of our DAPS scheme.

Theorem 3.3. *If* VC *is position-binding,* DTC *is a secure DTC scheme (Definition 2.9), F is PRF and H_1 and H_2 are collision-resistant hash functions, then our DAPS scheme is EUF-CMA secure. More precisely if h is the height of tree,*

$$\begin{aligned}\Pr[\mathsf{Forg}^{\mathcal{A}}_{\mathsf{DAPS}}(\kappa) = 1] \leq (2h-2) \cdot (2\Pr[\mathsf{DTColl}^{\mathcal{B}}_{\mathsf{DTC}}(\kappa) = 1] + 2\Pr[\mathsf{PRF}^{\mathcal{D}}_{F_k}(\kappa) = 1] \\ + \Pr[\mathsf{KExt}^{\mathsf{Tr}}_{\mathsf{DTC},\mathcal{A}'}(\kappa) = 1] + \Pr[\mathsf{HColl}^{\mathcal{B}_2}_{H_2}(\kappa) = 1] \\ + \Pr[\mathsf{PBind}^{\mathcal{B}'}_{\mathsf{VC}}(\kappa) = 1] + \Pr[\mathsf{HColl}^{\mathcal{B}_1}_{H_1}(\kappa) = 1]),\end{aligned}$$

Proof-Sketch. The proof goes by contradiction. We assume that the proposed scheme is not secure, this means that there exists an adversary $\mathcal{A}$ that can forge signatures with non negligible probability. Then we show that if such an $\mathcal{A}$ exists, we can use it to build a simulator S that can break one of the assumptions we started from.

First notice that the interaction of $\mathcal{A}$ with the signer should be as follows. Initially $\mathcal{A}$ obtains the public key. Then she is allowed to receive signatures σ_i for a certain (polynomial) number of messages (a_i, p_i) of her own choice. Each σ_i will be an authentication chain of the form $(r_h, (r_{h-1}, C_{h-1}, \Lambda_{h-1}), \ldots, (r_{\epsilon+1}, C_{\epsilon+1}, \Lambda_{\epsilon+1}), \Lambda_\epsilon)$ where each r_j is a random element, each C_j is a (vector) commitment of q hashed values and each Λ_j a corresponding opening. The main intuition underlying the proof is that a simulator knowing only one (say tk_b, where b is a bit) of the two trapdoors associated with the double trapdoor chameleon hash function will be able

to perfectly simulate a real signer. This means that S will be able to answer all the signing queries asked by $\mathcal{A}$ correctly. At the same time, if $\mathcal{A}$ produces a valid forgery $((a^*, p^*), \sigma^*)$, S will be able to use it to either break the collision resistance of the underlying hash functions or to confute the position binding property of the vector commitments or to extract the remaining trapdoor tk_{1-b}, with non negligible probability. Slightly more in detail, the simulator will work as follows. First it creates a public key in a way that is very similar to what prescribed by the KeyGen algorithm of DAPS scheme. The most important difference is that now S uses a DTC hash function for which it knows only one of the two corresponding trapdoors. Notice that by the security properties of DTC, this will induce (public) keys whose distribution is perfectly indistinguishable with respect to correctly generated ones. Next, whenever the adversary asks a signing query on the message (a_i, p_i) the simulator will generate a corresponding signature essentially as the real signer would do. The only difference, we stress, is that collisions for the DTC are resolved using tk_b, i.e. the only trapdoor known by the simulator. Again, by the security of the DTC this will induce collisions whose distribution is identical to that of those created by the real signer. Thus, the signatures produced by the simulator are perfectly indistinguishable with respect to the signatures a real signer would generate. Notice also that, exactly as for real signatures, each issued signature contributes to the creation of an authentication tree that must be maintained by the simulator. Now assume that $\mathcal{A}$ manages to produce a forgery $\sigma^* = (r_h^*, (r_{h-1}^*, C_{h-1}^*, \Lambda_{h-1}^*), \ldots, (r_{\epsilon+1}^*, C_{\epsilon+1}^*, \Lambda_{\epsilon+1}^*), \Lambda_\epsilon^*)$ on a (up to now) unsigned message (a^*, p^*). We argue that the public key and the verification tests on valid signatures imply that the forged signature must, at some point, deviate from the authentication tree created by the simulator. This means that there exists a previously issued signature $((a_i, p_i), \sigma_i = (r_h, (r_{h-1}, C_{h-1}, \Lambda_{h-1}), \ldots, (r_{\epsilon+1}, C_{\epsilon+1}, \Lambda_{\epsilon+1}), \Lambda_\epsilon))$ and an index $0 \leq j \leq h$, such that j is the smallest index for which $(r_j \neq r_j^*) \vee \left(C_j^* \neq C_j\right) \vee \left(\Lambda_j \neq \Lambda_j^*\right)$ (for ease of notation we assume here that ϵ is interpreted as 0). It is not too hard to see that each combination of the above conditions allows the simulator to either break the collision resistance of (one of) the underlying hash functions, or to confute the position binding of the vector commitment or to break the security of the double trapdoor chameleon hash function. This latter point is the most delicate and deserves some further explanation. Recall that the simulator is *already* capable of finding collisions for the DTC, as it knows one of the two trapdoors (i.e. tk_b) associated to it. Even worse, it had to compute many such collisions in order to be able to sign correctly. The catch here is that, during the whole signing process, the bit b associated with tk_b remains information theoretically hidden to the adversary. As discussed above, this is because of the distribution of collisions property of DTCs (and the fact that the simulation is stateful). Thus, the forgery produced by $\mathcal{A}$ cannot depend on b. Concretely, this means that if σ^* provides a collision for the DTC, Key-extractability guarantees that such a collision will allow S to extract tk_{1-b} with non-negligible probability. The formal proof is given in Appendix C.

3.3 Extension to Untrusted Setup (DAPS-DTC)

We discuss a simple modification of our DAPS-VC-DTC to make it secure w.r.t. untrusted setup. Since there are not instantiations for VC scheme without a trusted setup, we remove the VC scheme and also H_1 from the construction and replace them with a collision-resistant hash function $H : \mathcal{M}^q \to \mathcal{P}$ such that in each node u, all the chameleon-values associated with its children (except the child on the current path to the root) should be stored in the authentication chain. This means we will lose a q factor of efficiency in terms of the signature-size. We call this modification DAPS-DTC and instantiate it with our suggested DTC scheme in Appendix D. Clearly, this change does not affect the security since the only property we expect from a VC scheme in our construction is position binding. And replacing VC and H_1 with a collision-resistant hash function, trivially satisfies the required security-property. We emphasize that what we mean here by untrusted setup is that in practice if one uses a standard hash function (e.g. SHA3), no trusted setup is needed to generate the PK material related to the DTC hash function. And since standard hash functions are publicly known, the untrusted setup does not concern them.

4 A DAPS Scheme Based on NIZK Proofs

We start with recalling the generic DAPS construction proposed by Derler et al. [DRS18a]. The scheme, which we will refer to as DRS (or DAPS-DRS) supports an exponentially large address space and is based on NIZK proofs, which they instantiated in RO model. Here we give an instantiation without random oracles and from standard assumptions.This answers the question that if we can have a DAPS scheme in the standard model through the existing works (just by a correct instantiation) and how they would be compared with our DAPS-VC-DTC. In DAPS-DRS scheme a signature contains a value $z = \gamma \cdot p + \mathsf{sk}_\Sigma$ where p is the payload of the message, sk_Σ is the signing key for a digital signature scheme Σ, and γ is derived from the address part of the message as $\gamma = F(\mathsf{sk}_{\mathsf{PRF}}, a)$, where F is a pseudo-random function (PRF). If the signer cannot change the keys sk_Σ and $\mathsf{sk}_{\mathsf{PRF}}$, then signing a compromising pair of messages will reveal its secret key sk_Σ. To force the signer to use the same values sk_Σ and $\mathsf{sk}_{\mathsf{PRF}}$ for all its signatures, it must commit to them during key generation. These commitments can be done by fixing a PRF F and a one-way function f, computing $c = F(\mathsf{sk}_{\mathsf{PRF}}, \beta)$ for a

fixed value β and $\mathsf{vk}_\Sigma = f(\mathsf{sk}_\Sigma)$, and publishing (β, c) and vk_Σ. We note that pseudorandomness property is not sufficient to consider c as a commitment to $\mathsf{sk}_{\mathsf{PRF}}$: if the signer can find a key $\mathsf{sk}'_{\mathsf{PRF}} \neq \mathsf{sk}_{\mathsf{PRF}}$ such that $F(\mathsf{sk}_{\mathsf{PRF}}, \beta) = F(\mathsf{sk}'_{\mathsf{PRF}}, \beta)$ then it can sign a compromising pair of messages without revealing its secret key sk_Σ. A PRF for which it is hard to find such $\mathsf{sk}'_{\mathsf{PRF}}$ is called value-key binding PRF.

While after publishing (β, c), the key $\mathsf{sk}_{\mathsf{PRF}}$ is fixed, the signer also needs to be forced to produce the correct value γ; it must thus give a proof that $(\gamma = F(\mathsf{sk}_{\mathsf{PRF}}, a) \wedge c = F(\mathsf{sk}_{\mathsf{PRF}}, \beta))$ (the same reasoning and strategy works for the key sk_Σ). The signature thus consists of z and a non-interactive zero-knowledge proof for the following language:

$$L = \left\{(\mathsf{vk}_\Sigma, \beta, c, m, z) |\ \exists\, (\mathsf{sk}_\Sigma, \mathsf{sk}_{\mathsf{PRF}}, \gamma) : ((\mathsf{vk}_\Sigma, \beta, c, m, z), (\mathsf{sk}_\Sigma, \mathsf{sk}_{\mathsf{PRF}}, \gamma)) \in R\right\}$$

where the tuple is in relation R if $\gamma = F(\mathsf{sk}_{\mathsf{PRF}}, a) \ \wedge\ c = F(\mathsf{sk}_{\mathsf{PRF}}, \beta) \ \wedge\ z = \gamma \cdot p + \mathsf{sk}_\Sigma \ \wedge\ \mathsf{vk}_\Sigma = f(\mathsf{sk}_\Sigma)$. The full description of their scheme is given in Appendix E, Fig. 15.

We slightly modify DAPS-DRS construction [DRS18a] such that NIZK proof system can be instantiated by Groth-Sahai proof [GS08] (which we call DAPS-GS, see Fig. 16). We emphasize that in [DRS18a], the authors instantiate the underlying NIZK proof system through a Fiat-Shamir-transform over a interactive-ZK proof system. And as it is well-known, the Fiat-Shamir-transform is secure in the RO model. While here we use the Groth-Sahai proof system which provides efficient NIZK proofs in the standard model and under the standard assumption, for a special, yet wide, class of languages. Another point is that DAPS-DRS (consequently our DAPS-GS instantiation) is secure only against trusted setup, since it is in the CRS model.

Another difference between our DAPS-GS scheme and DAPS-DRS is that the latter uses a value-key-binding PRF, which prevents the signer from changing the secret key of PRF. We assign this task to a commitment scheme and can therefore relax the value-key-binding requirement on the PRF to standard PRF security. In DAPS-DRS scheme, the authors instantiate their key-value binding PRF F with a block cipher LowMC. They consider this block cipher as a value-key-binding PRF just based on the intuition and without presenting any exact proof regarding key-value binding property. Thus, for our modification (Appendix E, Fig. 16) the statements belong to a language L as follows;

$$L = \big\{(\mathsf{vk}_\Sigma, \mathsf{pp}_{\mathcal{C}}, z, m, \mathcal{C}_{\mathsf{PRF}}) \mid \exists(\mathsf{sk}_\Sigma, \mathsf{sk}_{\mathsf{PRF}}, \gamma) : (\mathsf{vk}_\Sigma, \mathsf{pp}_{\mathcal{C}}, z, m, \mathcal{C}_{\mathsf{PRF}}), (\mathsf{sk}_\Sigma, \mathsf{sk}_{\mathsf{PRF}}, \gamma)) \in R^*\big\},$$

where the relation R^* is as $\gamma = F(\mathsf{sk}_{\mathsf{PRF}}, a) \ \wedge\ \mathcal{C}_{\mathsf{PRF}} = \mathsf{Commit}(\mathsf{pp}_{\mathcal{C}}, \mathsf{sk}_{\mathsf{PRF}}) \ \wedge\ z = \gamma^p \cdot \mathsf{sk}_\Sigma \ \wedge\ R(\mathsf{vk}_\Sigma, \mathsf{sk}_\Sigma) = 1$, and relation R is such that given vk_Σ it is not easy to find the witness sk_Σ.
In a second step, we then give instantiations of the underlying primitives, that is, the PRF, signatures and commitments, which must be compatible with Groth-Sahai proof system. These instantiations are given in Appendix E.2. We will see that despite our efforts to optimize the scheme, our DAPS-GS scheme is less efficient than our DAPS-VC-DTC scheme.

Acknowledgements. The second author is supported by the Vienna Science and Technology Fund (WWTF) through project VRG18-002. The third author is supported by.....I dont know by who, Michel would tell me ;).....

References

BCG07. E. Bresson, D. Catalano, and R. Gennaro. Improved on-line/off-line threshold signatures. In *PKC 2007*, *LNCS* 4450, pages 217–232. Springer, Heidelberg, April 2007. (Pages 2, 3, and 18.)

BF14. M. Bellare and G. Fuchsbauer. Policy-based signatures. In *PKC 2014*, *LNCS* 8383, pages 520–537. Springer, Heidelberg, March 2014. (Pages 21 and 23.)

BFPV11. O. Blazy, G. Fuchsbauer, D. Pointcheval, and D. Vergnaud. Signatures on randomizable ciphertexts. In *PKC 2011*, *LNCS* 6571, pages 403–422. Springer, Heidelberg, March 2011. (Pages 23 and 24.)

BKK90. J. Boyar, S. A. Kurtz, and M. W. Krentel. A discrete logarithm implementation of perfect zero-knowledge blobs. *Journal of Cryptology*, 2(2):63–76, January 1990. (Pages 14 and 20.)

BKN17. D. Boneh, S. Kim, and V. Nikolaenko. Lattice-based DAPS and generalizations: Self-enforcement in signature schemes. In *ACNS 17*, *LNCS* 10355, pages 457–477. Springer, Heidelberg, July 2017. (Pages 2, 3, and 4.)

BPS17. M. Bellare, B. Poettering, and D. Stebila. Deterring certificate subversion: Efficient double-authentication-preventing signatures. In *PKC 2017, Part II*, *LNCS* 10175, pages 121–151. Springer, Heidelberg, March 2017. (Pages 2, 3, and 4.)

CD96. R. Cramer and I. Damgård. New generation of secure and practical RSA-based signatures. In *CRYPTO'96*, *LNCS* 1109, pages 173–185. Springer, Heidelberg, August 1996. (Pages 4 and 14.)

CDFG08. D. Catalano, M. Di Raimondo, D. Fiore, and R. Gennaro. Off-line/on-line signatures: Theoretical aspects and experimental results. In *PKC 2008*, *LNCS* 4939, pages 101–120. Springer, Heidelberg, March 2008. (Pages 2, 3, 6, and 19.)

CF13. D. Catalano and D. Fiore. Vector commitments and their applications. In *PKC 2013*, *LNCS* 7778, pages 55–72. Springer, Heidelberg, February / March 2013. (Pages 2, 3, 6, and 18.)

CFN90. D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In *CRYPTO'88*, *LNCS* 403, pages 319–327. Springer, Heidelberg, August 1990. (Page 1.)

CG05. D. Catalano and R. Gennaro. Cramer-Damgård signatures revisited: Efficient flat-tree signatures based on factoring. In *PKC 2005*, *LNCS* 3386, pages 313–327. Springer, Heidelberg, January 2005. (Page 4.)

CS99. R. Cramer and V. Shoup. Signature schemes based on the strong RSA assumption. In *ACM CCS 99*, pages 46–51. ACM Press, November 1999. (Page 14.)

DRS18a. D. Derler, S. Ramacher, and D. Slamanig. Generic double-authentication preventing signatures and a post-quantum instantiation. In *ProvSec 2018*, *LNCS* 11192, pages 258–276. Springer, Heidelberg, October 2018. (Pages 2, 3, 4, 11, 12, and 21.)

DRS18b. D. Derler, S. Ramacher, and D. Slamanig. Short double- and n-times-authentication-preventing signatures from ECDSA and more. In *2018 IEEE European Symposium on Security and Privacy, EuroS&P 2018, London, United Kingdom, April 24-26, 2018*, pages 273–287, 2018. (Pages 2 and 3.)

FS87. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO'86*, *LNCS* 263, pages 186–194. Springer, Heidelberg, August 1987. (Page 4.)

GPV08. C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *40th ACM STOC*, pages 197–206. ACM Press, May 2008. (Page 4.)

Gro06. J. Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In *ASIACRYPT 2006*, *LNCS* 4284, pages 444–459. Springer, Heidelberg, December 2006. (Pages 21 and 23.)

GS08. J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In *EUROCRYPT 2008*, *LNCS* 4965, pages 415–432. Springer, Heidelberg, April 2008. (Pages 3, 12, 21, and 23.)

HK08. D. Hofheinz and E. Kiltz. Programmable hash functions and their applications. In *CRYPTO 2008*, *LNCS* 5157, pages 21–38. Springer, Heidelberg, August 2008. (Pages 23 and 24.)

KR00. H. Krawczyk and T. Rabin. Chameleon signatures. In *NDSS 2000*. The Internet Society, February 2000. (Page 14.)

LGW+19. F. Li, W. Gao, G. Wang, K. Chen, and C. Tang. Double-authentication-preventing signatures revisited: new definition and construction from chameleon hash. *Frontiers of IT & EE*, 20(2):176–186, 2019. (Pages 3 and 4.)

NR97. M. Naor and O. Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *38th FOCS*, pages 458–467. IEEE Computer Society Press, October 1997. (Page 23.)

Ped92. T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO'91*, *LNCS* 576, pages 129–140. Springer, Heidelberg, August 1992. (Pages 20 and 24.)

Poe18. B. Poettering. Shorter double-authentication preventing signatures for small address spaces. In *AFRICACRYPT 18*, *LNCS* 10831, pages 344–361. Springer, Heidelberg, May 2018. (Pages 2, 3, 4, and 5.)

PS14. B. Poettering and D. Stebila. Double-authentication-preventing signatures. In *ESORICS 2014, Part I*, *LNCS* 8712, pages 436–453. Springer, Heidelberg, September 2014. (Pages 2, 3, 4, and 5.)

RKS15. T. Ruffing, A. Kate, and D. Schröder. Liar, liar, coins on fire!: Penalizing equivocation by loss of bitcoins. In *ACM CCS 2015*, pages 219–230. ACM Press, October 2015. (Pages 2, 3, 4, 5, and 9.)

Wat05. B. R. Waters. Efficient identity-based encryption without random oracles. In *EUROCRYPT 2005*, *LNCS* 3494, pages 114–127. Springer, Heidelberg, May 2005. (Page 24.)

A Backgrounds

For a digital signature scheme the standard existential unforgeability is defined as follows.

Definition A.1 (Unforgeability of digital signature). *Signature scheme Σ is existentially unforgeable under adaptive chosen message attacks if, for all* p.p.t. *adversaries $\mathcal{A}$, there exists a negligible function* negl *such that* $\Pr[\mathsf{Forg}^{\mathcal{A}}_{\mathsf{DS}}(\kappa)=1] \le \mathrm{negl}(\kappa)$ *where the experiment* $\mathsf{Forg}^{\mathcal{A}}_{\mathsf{DS}}(\kappa)$ *is described in Fig. 10.*

Definition A.2 (Chameleon Hash Function [KR00]). *A chameleon hash function (CHF), also known as trapdoor hash function, is a tuple of* p.p.t. *algorithms $\mathcal{H}=(\mathsf{KeyGen},\mathsf{CHash},\mathsf{Coll})$ defined as follows:*

- $\mathsf{KeyGen}(1^\kappa)$*: On input the security parameter κ, it outputs a pair of public/private keys $(\mathsf{pk},\mathsf{tk})$ (where pk implicitly defines the message space $\mathcal{M}$ and the randomness space $\mathcal{R}$).*
- $\mathsf{CHash}_{\mathsf{pk}}(m,r)$*: On input the public key pk, a message $m\in\mathcal{M}$ and a random nonce $r\in\mathcal{R}$, it outputs a hash value.*
- $\mathsf{Coll}(\mathsf{tk},m',m,r)$*: On input the private key tk, two messages m,m' and a nonce r, it outputs a nonce r' such that $\mathsf{CHash}_{\mathsf{pk}}(m,r)=\mathsf{CHash}_{\mathsf{pk}}(m',r')$.*

Definition A.3 (Security of CHF [KR00, BKK90]).
Collision-resistance. *For any* p.p.t. *adversary $\mathcal{A}$, there exists a negligible function* negl *such that*

$$\Pr[\mathsf{CHColl}^{\mathcal{A}}_{\mathsf{CHF}}(\kappa)=1] = \Pr\left[\begin{array}{l}(\mathsf{pk},\mathsf{tk})\leftarrow\mathsf{KeyGen}(1^\kappa)\\(m,r,m',r')\leftarrow\mathcal{A}(1^\kappa,\mathsf{pk})\end{array} : \begin{array}{l}\mathsf{CHash}_{\mathsf{pk}}(m,r)=\mathsf{CHash}_{\mathsf{pk}}(m',r')\\ \wedge\ (m,r)\neq(m',r')\end{array}\right] \le \mathrm{negl}(\kappa).$$

Key-extractability. *A chameleon hash function is key-extractable if there exists a* p.p.t. *algorithm* Ext*, as follows:*

- $\mathsf{Ext}(\mathsf{pk},S_1,S_2)$*: On input a public key pk and a collision pair $\big(S_1=(m_1,r_1),S_2=(m_2,r_2)\big)$, outputs a secret key tk'.*

such that for all adversaries $\mathcal{A}$, there is a negligible function $\mathrm{negl}(\kappa)$ *such that,* $\Pr[\mathsf{KExt}^{\mathsf{Tr/UTr}}_{\mathsf{CHF},\mathcal{A}}(\kappa)=1]\le\mathrm{negl}(\kappa)$ *where the experiment* $\mathsf{KExt}^{\mathsf{Tr/UTr}}_{\mathsf{CHF},\mathcal{A}}(\kappa)$ *is described in Fig. 11.*

Distribution of collisions. *For every m,m', and a uniformly random r, the distribution of $r'=\mathsf{Coll}(\mathsf{tk},m,m',r)$ is uniform, even when given pk, m and m'.*

Efficient CHF schemes have been instantiated based on the discrete-logarithm assumption [BKK90] and the RSA (factorization) assumption [CD96, CS99].

Definition A.4 (Collision-Resistant Hash Function). *$\mathcal{H}=\{H_i:\mathcal{X}\to\mathcal{Y}\}_{i\in\mathcal{I}}$ is a family of collision-resistant hash functions if:*

Generation. *There is a* p.p.t. *algorithm $\mathsf{Gen}(1^\kappa)$ which outputs $i\in\mathcal{I}$.*
Efficient evaluation. *Given x and i, one can compute $H_i(x)$ in time polynomial in κ.*
Collision-resistance (CR). *For every* p.p.t. *adversary $\mathcal{A}$ there is a negligible function* negl *such that for $i\leftarrow\mathsf{Gen}(1^\kappa)$*

$$\Pr\left[\mathsf{HColl}^{\mathcal{A}}_{\mathcal{H}}(\kappa)=1\right]=\Pr\left[(x,x')\leftarrow\mathcal{A}(1^\kappa,i)\ :\ \begin{array}{l}x\neq x'\ \wedge\\ H_i(x)=H_i(x')\end{array}\right]\le\mathrm{negl}(\kappa).$$

$\underline{\mathsf{Forg}^{\mathcal{A}}_{\mathsf{DS}}(\kappa)}$:	$\underline{\text{Oracle } \mathsf{Sign}(\mathsf{sk},m)}$:
$(\mathsf{sk},\mathsf{vk})\leftarrow\mathsf{KeyGen}(1^\kappa)$ $Q\leftarrow\emptyset$ $(m^*,\sigma^*)\leftarrow\mathcal{A}^{\mathsf{Sign}(\mathsf{sk},\cdot)}$ Return 1 iff: – $\mathsf{Verif}(\mathsf{vk},m^*,\sigma^*)=1$ – $m^*\notin Q$	$\sigma\leftarrow\mathsf{Sign}(\mathsf{sk},m)$ $Q\leftarrow Q\cup\{m\}$ Return σ

Fig. 10: Game for UEF-CMA security of DS.

$\underline{\mathsf{KExt}^{\mathsf{Tr}}_{\mathsf{CHF},\mathcal{A}}(\kappa)}$:	$\underline{\mathsf{KExt}^{\mathsf{UTr}}_{\mathsf{CHF},\mathcal{A}}(\kappa)}$:
$(\mathsf{pk},\mathsf{tk}) \leftarrow \mathsf{KeyGen}(1^\kappa)$	$(\mathsf{pk}, S_1, S_2) \leftarrow \mathcal{A}(1^\kappa)$
$(S_1, S_2) \leftarrow \mathcal{A}(\mathsf{pk},\mathsf{tk})$	$\mathsf{tk}' \leftarrow \mathsf{Ext}(\mathsf{pk}, S_1, S_2)$
$\mathsf{tk}' \leftarrow \mathsf{Ext}(\mathsf{pk}, S_1, S_2)$	Outputs 1 iff
Return 1 iff	– (S_1, S_2) is a collision
– (S_1, S_2) is a collision	– $\mathsf{tk}' \neq \mathsf{tk}$
– $\mathsf{tk}' \neq \mathsf{tk}$	

Fig. 11: KE game for a CHF scheme. The left game is for trusted setups and the right game for untrusted setup (maliciously generated keys).

Definition A.5 (Pseudo-Random Function (PRF)). *Let $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a family of functions $\{F_k : \mathcal{X} \to \mathcal{Y}\}_{k\in\mathcal{K}}$ (where elements in $\mathcal{K}$, $\mathcal{X}$, $\mathcal{Y}$ are of length $\mathrm{poly}(\kappa)$). We say F is a pseudorandom function (PRF) if for all probabilistic* p.p.t. *distinguishers $\mathcal{D}$, there exists a negligible function* negl *such that $\left|\Pr\left[\mathcal{D}^{F_k(\cdot)}(1^\kappa) = 1\right] - \Pr\left[\mathcal{D}^{f(\cdot)}(1^\kappa) = 1\right]\right| \leq \mathrm{negl}(\kappa)$, where $k \xleftarrow{R} \mathcal{K}$ and f is uniformly chosen at random from the set of functions mapping $\mathcal{X}$ to $\mathcal{Y}$.*

Security Requirements for NIZK Proof System. For a NIZK proof system the following security requirements are defined.

Definition A.6 (Perfect Completeness). *For all adversaries $\mathcal{A}$ whose output (x, w) satisfies $R(x, w) = 1$ we have*

$$\Pr\left[\begin{matrix}\mathsf{crs} \leftarrow \mathsf{Setup}(1^\kappa); (x, w) \leftarrow \mathcal{A}(\mathsf{crs}) \\ \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w)\end{matrix} \;:\; \mathsf{Verif}(\mathsf{crs}, x, \pi) = 1\right] = 1$$

Definition A.7 (Soundness). *For all adversaries $\mathcal{A}$ we have*[7]

$$\Pr\left[\mathsf{crs} \leftarrow \mathsf{Setup}(1^\kappa);\ (x, \pi) \leftarrow \mathcal{A}(\mathsf{crs}) \;:\; \mathsf{Verif}(\mathsf{crs}, x, \pi) = 1 \wedge x \notin L\right] \leq \mathrm{negl}(\kappa).$$

Definition A.8 (Perfect knowledge-soundness). *Π is a proof of knowledge for relation R, if there exists a knowledge extractor $\mathcal{E} = (\mathcal{E}_1, \mathcal{E}_2)$ such that for all adversaries $\mathcal{D}$ and $\mathcal{A}$ we have*

$$\Pr\left[\mathsf{crs} \leftarrow \mathsf{Setup}(1^\kappa) \;:\; \mathcal{D}(\mathsf{crs}) = 1\right] = \Pr\left[(\mathsf{crs}, \xi) \leftarrow \mathcal{E}_1(1^\kappa) \;:\; \mathcal{D}(\mathsf{crs}) = 1\right]$$
$$\text{and} \quad \Pr\left[\begin{matrix}(\mathsf{crs}, \xi) \leftarrow \mathcal{E}_1(1^\kappa) \\ (x, \pi) \leftarrow \mathcal{A}(\mathsf{crs}) \\ w \leftarrow \mathcal{E}_2(\mathsf{crs}, \xi, x, \pi)\end{matrix} \;:\; \begin{matrix}\mathsf{Verif}(\mathsf{crs}, x, \pi) = 0 \\ \vee\ R(x, w) = 1\end{matrix}\right] = 1.$$

Definition A.9 (Composable Zero-Knowledge). *Π is composable zero-knowledge if there exists a simulator $(\mathcal{S}_1, \mathcal{S}_2)$ such that for all adversaries $\mathcal{D}$ and $\mathcal{A}$ whose output (x, w) satisfies $R(x, w) = 1$:*

$$\left|\Pr\left[\mathsf{crs} \leftarrow \mathsf{Setup}(1^\kappa) \;:\; \mathcal{D}(\mathsf{crs}) = 1\right] - \Pr\left[(\mathsf{crs}, \tau) \leftarrow \mathcal{S}_1(1^\kappa) \;:\; \mathcal{D}(\mathsf{crs}) = 1\right]\right| \leq \mathrm{negl}(\kappa) \quad \text{and}$$
$$\left|\Pr\left[(\mathsf{crs}, \tau) \leftarrow \mathcal{S}_1(1^\kappa),\ (x, w) \leftarrow \mathcal{A}(\mathsf{crs}, \tau),\ \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w) \;:\; \mathcal{A}(\mathsf{crs}) = 1\right]\right.$$
$$\left. - \Pr\left[(\mathsf{crs}, \tau) \leftarrow \mathcal{S}_1(1^\kappa),\ (x, w) \leftarrow \mathcal{A}(\mathsf{crs}, \tau),\ \pi \leftarrow \mathcal{S}_2(\mathsf{crs}, \tau, x) \;:\; \mathcal{A}(\mathsf{crs}) = 1\right]\right| \leq \mathrm{negl}(\kappa).$$

Definition A.10 (Simulation-sound-extractability (SE)). *Π is SE if there exists a simulator $(\mathcal{SE}_1, \mathcal{S}_2, \mathcal{E}_2)$ such that for all adversaries $\mathcal{A}$:*

$$\Pr\left[\begin{matrix}(\mathsf{crs}, \tau, \xi) \leftarrow \mathcal{SE}_1(1^\kappa) \\ (x, \pi) \leftarrow \mathcal{A}^{\mathcal{S}_2(\mathsf{crs}, \tau, \cdot)}(\mathsf{crs}) \\ w \leftarrow \mathcal{E}_2(\mathsf{crs}, \xi, x, \pi)\end{matrix} \;:\; \begin{matrix}\mathsf{Verif}(\mathsf{crs}, x, \pi) = 1 \\ \wedge\ R(x, w) = 0 \\ \wedge\ (x, \pi) \notin Q\end{matrix}\right] \leq \mathrm{negl}(\kappa).$$

B Proof of Theorem 3.2

Let $\mathcal{A}$ be the adversary attacking the key-extractability of our DAPS scheme, that is, trying to produce a compromising pair $(\sigma, m = (a, p))$ and $(\sigma', m' = (a, p'))$, where $p' \neq p$ and σ, σ' are valid signatures. In the proof,

[7] If $\mathcal{A}$ is p.p.t., we call the system as the "Argument System", while it is called "Proof System" for an unbounded $\mathcal{A}$.

values associated with σ' are denoted by a dash $'$. Note that the validity of two signatures implies that there exists a node $u \in \mathsf{path}_{a\to\epsilon}$ where these two different authentication chains become the same in their C_u values. Clearly, such an unification point must exist as the two signatures have to be verified with respect to the same public key C_ϵ. More precisely, there exists a collision-point $u \in \mathsf{path}_{a\to\epsilon}$ such that $C_v = C'_v$ for $v \in \mathsf{path}_{u\to\epsilon}$. We distinguish two cases:

1. The collision-point is a leaf-node.
2. The collision-point is an interior-node.

In case (1), for each node v in $\mathsf{path}_{a\to\epsilon}$ we have $C_{h-1} = C'_{h-1}$. Then, the position-binding of VC scheme, implies that $m_h = H_2(\mathsf{CHash}(p, r_h))$ and $m'_h = H_2(\mathsf{CHash}(p', r'_h))$ (the values to which C_{h-1} and C'_{h-1} are opened) must be equal. By collision-resistance of H_2, we moreover have $\mathsf{CHash}(p, r_h) = \mathsf{CHash}(p', r'_h)$. Since $p' \neq p$, this constitutes a collision for the DTC scheme which then a trapdoor tk_i, for $i = 0$ or 1, can be extracted . Thus we have,

$$
\begin{aligned}
&\Pr[\mathsf{KExt}^{\mathsf{Tr}}_{\mathsf{DAPS},\mathcal{A}}(\kappa) = 1 \mid \text{case 1}] \\
&\leq \Pr[\mathsf{KExt}^{\mathsf{Tr}}_{DAPS,\mathcal{A}}(\kappa) = 1 \mid \text{case 1} \ \wedge\ \mathsf{CHash}(p', r'_h) = \mathsf{CHash}(p, r_h)] + \\
&\quad \Pr[\mathsf{KExt}^{\mathsf{Tr}}_{\mathsf{DAPS},\mathcal{A}}(\kappa) = 1 \mid \text{case 1} \ \wedge\ \mathsf{CHash}(p', r'_h) \neq \mathsf{CHash}(p, r_h) \ \wedge\ m'_h = m_h] + \\
&\quad \Pr[\mathsf{KExt}^{\mathsf{Tr}}_{\mathsf{DAPS},\mathcal{A}}(\kappa) = 1 \mid \text{case 1} \ \wedge\ \mathsf{CHash}(p', r'_h) \neq \mathsf{CHash}(p, r_h) \ \wedge\ m'_h \neq m_h] \\
&\leq \Pr[\mathsf{KExt}^{\mathsf{Tr}/\mathsf{UTr}}_{\mathsf{DTC},\mathcal{C}}(\kappa) = 1] + \Pr[\mathsf{HColl}^{\mathcal{B}'}_{H_2}(\kappa) = 1] + \Pr[\mathsf{PBind}^{\mathcal{D}}_{\mathsf{VC}}(\kappa) = 1].
\end{aligned}
$$

For case (2), lets call the child of u as node v, a similar reasoning in case (1) works for authentication chains from node v to the root. The only difference is that here in node v – corresponding to the leaf node a_h in case (1) – instead of p, p' we have $p_v = H_1(C_v)$ and $p'_v = H_1(C'_v)$. Then, since u is the first collision node, we have $C_v \neq C'_v$ which gives $p_v \neq p'_v$ with overwhelming probability, due to the collision-resistance of H_1. Thus, in node v we have $p_v \neq p'_v$. From here the same reasoning as the case (1) works. More precisely,

$$
\begin{aligned}
&\Pr[\mathsf{KExt}^{\mathsf{Tr}}_{\mathsf{DAPS},\mathcal{A}}(\kappa) = 1 \mid \text{case 2}] \\
&\leq \Pr[\mathsf{KExt}^{\mathsf{Tr}}_{\mathsf{DAPS},\mathcal{A}}(\kappa) = 1 \mid \text{case 2} \ \wedge\ C_v \neq C'_v \ \wedge\ p_v = p'_v] + \\
&\quad \Pr[\mathsf{KExt}^{\mathsf{Tr}}_{\mathsf{DAPS},\mathcal{A}}(\kappa) = 1 \mid \text{case 2} \ \wedge\ C_v \neq C'_v \ \wedge\ p_v \neq p'_v] \\
&\leq \Pr[\mathsf{HColl}^{\mathcal{B}}_{H_1}(\kappa) = 1] + \Pr[\mathsf{KExt}^{\mathsf{Tr}}_{\mathsf{DAPS},\mathcal{A}}(\kappa) = 1 \mid \ \text{case 1 starting from } v]. \\
&\leq \Pr[\mathsf{HColl}^{\mathcal{B}}_{H_1}(\kappa) = 1] + \Pr[\mathsf{KExt}^{\mathsf{Tr}/\mathsf{UTr}}_{\mathsf{DTC},\mathcal{C}}(\kappa) = 1] + \\
&\quad \Pr[\mathsf{HColl}^{\mathcal{B}'}_{H_2}(\kappa) = 1] + \Pr[\mathsf{PBind}^{\mathcal{D}}_{\mathsf{VC}}(\kappa) = 1].
\end{aligned}
$$

This completes the proof.

C Proof of Theorem 3.3

Let $\mathcal{A}$ be the attacker to the unforgeability of our DAPS scheme trying to give a forgery $((a^*, p^*), \sigma^*)$, and u be the first node in $\mathsf{path}_{a^*\to\epsilon}$ that has already been visited during the game, that is, either during the computation of the public key or during a signing query. Note that such node exists due to the construction of vk, and only DTC values are assigned to this node, while its parent also contains VC value. We denote all values associated with the forgery by an asterisk (*). There are two cases

1: node u is an internal node.
2: node u is a leaf node.

At first we consider case (1). Let $(\$_u, r_u)$ be the earliest assigned DTC-value to node u (i.e., generated by PRF F, where $\$_u$ is the message and r_u is the randomness for DTC). As we know at any such non-leaf node u, the signature algorithm at first computes p^*_u (as the hash of VC-value) and then m^*_u (as the hash of DTC-value on the newly computed VC-value p^*_u). For such node u and a valid forgery by $\mathcal{A}$, one of the following cases may happen:

(i) The DTC-values are equal. Meaning that there is a randomness r^*_u such that, $\mathsf{CHash}_{\mathsf{pk}_{\mathsf{DTC}}}(\$_u, r_u) = \mathsf{CHash}_{\mathsf{pk}_{\mathsf{DTC}}}(p^*_u, r^*_u)$.
(ii) The DTC-values are different but m_u and m^*_u (hash of TDC-values) are equal. This is, $m^*_u = H_2(\mathsf{CHash}(p^*_u, r^*_u)) = H_2(\mathsf{CHash}(\$_u, r_u)) = m_u$ while $\mathsf{CHash}(p^*_u, r^*) \neq \mathsf{CHash}(\$_u, r_u)$.

If none of these cases happen in node u, again the valid forgery implies that in the parent node of u, lets call it node w, two more cases may happen (note that due to the choice of u, a VC-value is already assigned to w):

(iii) The VC-values are equal. Meaning that $C^*_w = C_w$ where C_w is the previously assigned VC-value to this node.
(iv) The VC-value are different while p_u and p^*_u (hash of VC-Values) are equal. This is, $p^*_w = H_1(C^*_w) = H_1(C_w) = p_w$ and $C^*_w \neq C_w$.

The above reasoning would be repeated in each node from u to the root (at node u cases i and ii, at root case iii, and at other nodes cases i-iv). And since, the forgery is verified by the public-key C_ϵ, one of the mentioned cases has to happen for a node in $\mathsf{path}_{u\to\epsilon}$. Now, based on these distinctions, we have the following reductions.

Reduction of DAPS-security to collision-resistance of DTC (case (i)): For case (i), we present an attacker $\mathcal{B}$ to the collision-resistance of DTC (see Definition 2.8) that can simulate the forgery-game for the adversary $\mathcal{A}$.

- $\mathcal{B}$ receives $\mathsf{pk}_{\mathsf{DTC}}, \mathsf{tk}_b$ from its challenger. It simulate other secret and public keys similar to the real game and turns back vk to the adversary $\mathcal{A}$ while $\mathsf{sk} = (\mathsf{tk}_b, k)$.
- when $\mathcal{A}$ issues a signing query for the message (a, p), the adversary $\mathcal{B}$ responds the same as the real game. I.e., it runs $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, (a, p))$ and sends σ to $\mathcal{B}$.
- When $\mathcal{A}$ outputs a forgery, $\mathcal{B}$ extracts values $(\$_u, r_u)$ and (p^*_u, r^*_u). Then, it runs key-extraction algorithm of DTC scheme to get a trapdoor tk'. I.e., $\mathsf{tk}' \leftarrow \mathsf{Ext}(\mathsf{pk}_{\mathsf{DTC}}, (\$_u, r_u), (p^*_u, r^*_u))$. Finally it outputs tk'.

Now we discuss why this is a correct simulation and how (much) it helps $\mathcal{B}$ to win its own game.
At first, we note that the signature algorithm works even if only one of the trapdoors is available, which is due to the definition[8] and the first security-property of DTC[9]. Thus, the simulation of signing-queries is correctly done when $\mathsf{sk} = (\mathsf{tk}_b, k)$.

On the other hand, if the algorithm Ext works correctly, $\mathcal{B}$ can get a valid trapdoor (i.e., $\mathsf{tk}' = \mathsf{tk}_b$ or $\mathsf{tk}' = \mathsf{tk}_{1-b}$). Where the case $\mathsf{tk}' = \mathsf{tk}_{1-b}$ would happen with a probability $\frac{1}{2}$. In fact, due to the uniform distributions of collisions and the pseudo-random property of F, the opening values (r_u, r^*_u) (used as the input of the extractor) are independent of the trapdoor tk_b. Which essentially means the key-extractor outputs a trapdoor independent of the trapdoor tk_b used by the adversary $\mathcal{B}$.
Slightly in detail, note that during the simulation, though the trapdoor tk_b is used (polynomially) many times, it is used only and only when it comes to a node v that the values $(\$_v, r_v)$ are computed by PRF F . Then the security-property of DTC (distribution of collisions, Definition 2.9) implies that any collision generated based on uniform r_v is uniform and so does not include information about tk_b. By all we discussed above, the wining probability can be obtained as follows,

$$\Pr[\mathsf{tk}' = \mathsf{tk}_{1-b}] = \Pr[\mathsf{DTColl}^{\mathcal{B}}_{\mathsf{DTC}}(\kappa) = 1]$$

$$1/2 \cdot \Pr[\mathsf{Forg}^{\mathcal{A}}_{\mathsf{DAPS}}(\kappa) = 1 \mid \text{node } u \wedge \text{case(i)} \wedge \mathsf{Ext} \text{ works correctly} \wedge \text{uniform collisions } (r^*_u, r_u)] \leq \Pr[\mathsf{tk}' = \mathsf{tk}_{1-b}].$$

$$\Pr[\text{non-uniform collisions } (r^*_u, r_u)] \leq \Pr[\mathsf{PRF}^{\mathcal{D}}_{F_k}(\kappa) = 1].$$

$$\Pr[\mathsf{Ext} \text{ does not work correctly}] \leq \Pr[\mathsf{PRF}^{\mathcal{D}}_{F_k}(\kappa) = 1] + \Pr[\mathsf{KExt}^{\mathsf{Tr}}_{\mathsf{DTC},\mathcal{A}'}(\kappa) = 1].$$

Putting together we get,

$$\Pr[\mathsf{Forg}^{\mathcal{A}}_{\mathsf{DAPS}}(\kappa) = 1 \mid \text{node } u \wedge \text{case(i)}] \leq + 2\Pr[\mathsf{DTColl}^{\mathcal{B}}_{\mathsf{DTC}}(\kappa) = 1] + 2\Pr[\mathsf{PRF}^{\mathcal{D}}_{F_k}(\kappa) = 1] + \Pr[\mathsf{KExt}^{\mathsf{Tr}}_{\mathsf{DTC},\mathcal{A}'}(\kappa) = 1].$$

[8] Where the Coll algorithm works with the trapdoor tk_i generated by TrChos.
[9] Saying that TrChos generate the correct trapdoors as $\mathsf{KeyGen}(1^\kappa, i)$.

$\underline{\mathsf{KeyGen}(1^\kappa, q)}$: select two groups $\mathbb{G}$ and $\mathbb{G}_T$ of prime order p equipped with a bilinear map $e : \mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{G}_T$. Let $g \in \mathbb{G}$ be a random generator.
– sample $z_i, \ldots, z_q \xleftarrow{R} \mathbb{Z}_p$.
– set $h_i = g^{z_i}$ for $i = 1, \ldots, q$ and $h_{ij} = g^{z_i z_j}$ for $i, j = 1, \ldots, q$ and $i \neq j$.
Return $\mathsf{pp} = (g, \{h_i\}_i, \{h_{ij}\}_{i,j})$ and define $\mathcal{M} = \mathbb{Z}_p$

$\underline{\mathsf{Cmt}_{\mathsf{pp}}(m_1, \ldots, m_q)}$: compute $C = h_1^{m_1} h_2^{m_2} \ldots h_q^{m_q}$ (where $m_i \in \mathbb{Z}_p$).
Return C.

$\underline{\mathsf{Open}_{\mathsf{pp}}(m_i, i, m)}$: compute $\Lambda_i = \prod_{j=1, j\neq i}^{q} h_{i,j}^{m_j} = \left(\prod_{j=1, j\neq i}^{q} h_j^{m_j}\right)^{z_i}$ return Λ_i.

$\underline{\mathsf{Verif}(C, m_i, i, \Lambda_i)}$: Output 1 iff $e(C/h_i^{m_i}, h_i) = e(\Lambda_i, g)$

Fig. 12: Catalano-Fiore VC scheme [CF13]

Reductions for cases (ii,iii,iv) For these cases we construct adversaries $\mathcal{B}_2$, $\mathcal{B}'$ and $\mathcal{B}_1$ attacking respectively to the CR of H_2, position-binding of VC and CR of H_1.

For each cases, the adversary simulates the forgery-game for $\mathcal{A}$ such that the signing-queries are answered by the real algorithms and the given parameters/public-keys by the challenger. When $\mathcal{A}$ returns a forgery, then in:

Case ii. $\mathcal{B}_2$ extracts (p_u^*, r_u^*) and $(\$_u, r_u)$ from the forgery/queries and outputs $(\mathsf{CHash}(p_u^*, r_u^*), \mathsf{CHash}(\$_u, r_u))$.
Case iii. $\mathcal{B}'$ extracts m_u^*, m_u from the forgery/queries and outputs $(m_{v_1}, \ldots, m_u^*, \ldots, m_{v_q})$ and $(m_{v_1}, \ldots, m_u, \ldots, m_{v_q})$ where v_i is the i child of w.
Case iv. $\mathcal{B}_1$ outputs C_w^* and C_w which are extracted from the forgery/queries.

This complete the reduction for cases ii,iii and iv.

Similarly, the same reasoning (i),...,(iv) will apply to all other nodes in the current path to the root. Since C_ϵ is part of the verification key this recursive reasoning will be stopped at most in the root. This results in

$$\Pr[\mathsf{Forg}^{\mathcal{A}}_{\mathsf{DAPS}}(\kappa) = 1] \leq (2L - 2) \cdot \Pr[\mathsf{Forg}^{\mathcal{A}}_{\mathsf{DAPS}}(\kappa) = 1 \mid \text{node } v] \leq \mathrm{negl}(\kappa),$$

where $v \in \mathsf{path}_{u \to \epsilon}$ and L is the number of nodes on the path $\mathsf{path}_{u \to \epsilon}$.[10]

The proof for the case (2) that u is a leaf node can be done in a similar way. Here, case (i) applies but needs more attention. Because node u might already be used during the signing queries for a message $m = (a^*, p)$ where $p \neq p^*$. This cannot make any problem, since due to the condition $p^* \neq p$ (or equivalently $(a^*, p^*) \notin Q$ which comes from the security definition), in case (i) $\$_u = p, p_u^* = p^*$ which guarantees that $\$_u \neq p_u^*$ and consequently the same reduction based on the collision-resistance of DTC applies.

D Instantiations of building-blocks for DAPS-VC-DTC

We instantiate the building blocks for our DAPS construction in Fig. 9. We use the VC scheme proposed by Catalano and Fiore [CF13], presented in Fig. 12 and we propose an instantiation for DTC.

Theorem D.1 (Catalano-Fiore's VC scheme [CF13]). *If the CDH assumption holds in group $\mathbb{G}$, then the scheme of Catalano-Fiore in Fig. 12 is a position-binding VC scheme.*

A we discussed, our DAPS-VC-DTC construction can be modified to be secure against untrusted setup, if the underlying DTC scheme is secure in the untrusted setup setting. This fact gives a strong argument why we need to realize a DTC scheme in an untrusted setup.
A possible candidate to instantiate the DTC scheme is the chameleon hash given in [BCG07] where $\mathsf{CHash}(m; r, s) = g^m \mathsf{pk}_1^r \mathsf{pk}_2^s$, $\mathsf{pk}_1 = g^x$, $\mathsf{pk}_2 = g^y$. This scheme, however, does not satisfy the security requirements of DTC in the untrusted setup setting. The trouble is that in order to get the second trapdoor the extractor needs to know the first one. While this fits perfectly the application considered in [BCG07], it does not cope well with our scenario where the signer might have both trapdoors and the (KE) challenger none (in such a case a collision results in an equation with two unknowns). A way around this would be to have one of the trapdoors chosen by a trusted setup and the other one by the signer. To avoid the need of trusted setup, we build a DTC scheme based on two layers CHF such that given a collision ensures that there is a collision either in the first layer or in the second layer. This means, a collision gives enough equations to solve the system and extract unknowns.

[10] Each node, except the node u and the root, gets two roles once as a child once as a parent node leading to four cases i-iv. This is the reason for appearance of the term $(2L - 2)$.

The idea was implicitly given in [CDFG08] (Sect. 3.1, full version). However, since a more efficient DTC with security under a trusted setup was enough for them, they do not detail a construction or the security property. We note that not any arbitrary chaining may work here, and it is important to put the message in the first layer instead of the second layer (see Fig. 13).

Let $\mathcal{H}_i = (\mathsf{KeyGen}_i, \mathsf{CHash}_i, \mathsf{Coll}_i)$ for $i = 0, 1$, be two CHFs, and $H : \mathcal{O}_0 \longrightarrow \mathcal{M}_1$ be a hash function where $\mathcal{O}_0$ and $\mathcal{M}_1$ are respectively the image of $\mathcal{H}_0$ and the message space $\mathcal{H}_1$. The randomness space of $\mathcal{H}_i$ is denoted by $\mathcal{R}_i$. Our suggested DTC scheme is given in Fig. 13.

$\underline{\mathsf{KeyGen}(1^\kappa)}$:
– run $(\mathsf{pk}_i, \mathsf{tk}_i) \leftarrow \mathcal{H}_i.\mathsf{KeyGen}(1^\kappa)$ for $i = 0, 1$
– return $\mathsf{pk} = (\mathsf{pk}_0, \mathsf{pk}_1)$ and $\mathsf{tk} = (\mathsf{tk}_0||0, \mathsf{tk}_1||1)$.

$\underline{\mathsf{TrChos}(1^\kappa, i)}$:
– run $(\mathsf{pk}_i, \mathsf{tk}_i) \leftarrow \mathcal{H}_i.\mathsf{KeyGen}(1^\kappa)$ and sample pk_{1-i} from the public-key space $\mathcal{H}_{1-i}$.
– set $\mathsf{pk} = (\mathsf{pk}_0, \mathsf{pk}_1)$ and return $(\mathsf{pk}, \mathsf{tk}_i||i)$.

$\underline{\mathsf{CHash}_{\mathsf{pk}}(m, r, s)}$: for $m \in \mathcal{M}_0$, compute $\begin{cases} w = \mathcal{H}_0.\mathsf{CHash}(m, r) & \text{where } r \xleftarrow{R} \mathcal{R}_0 \\ C = \mathcal{H}_1.\mathsf{CHash}(H(w), s) & \text{where } s \xleftarrow{R} \mathcal{R}_1 \end{cases}$
– return C.

$\underline{\mathsf{Coll}(\mathsf{tk}, m', m, r, s)}$:
pars tk as $\mathsf{tk}_i||i$
– if $i = 0$, run $r' \leftarrow \mathcal{H}_0.\mathsf{Coll}(\mathsf{tk}_0, m', m, r)$ and set $s' = s$.
– if $i = 1$, choose $r' \xleftarrow{R} \mathcal{R}_0$, compute $w' = \mathcal{H}_0.\mathsf{CHash}_{\mathsf{pk}_0}(m', r')$ and run $s' \leftarrow \mathcal{H}_1.\mathsf{Coll}(\mathsf{tk}_1, H(w'), H(w), s)$ where $w = \mathcal{H}_0.\mathsf{CHash}_{\mathsf{pk}_0}(m, r)$
– return (r', s').

$\underline{\mathsf{Ext}(\mathsf{pk}, (m, r, s),), (m', r', s')}$:
– if $s \neq s'$ or $H(w) \neq H(w')$, run $\mathsf{tk}' \leftarrow \mathsf{Ext}_1(\mathsf{pk}_1; H(w), s; H(w'), s')$ where $w = \mathcal{H}_0.\mathsf{CHash}(m, r)$, $w' = \mathcal{H}_0.\mathsf{CHash}(m', r')$.
– if $w = w'$, run $\mathsf{tk}' \leftarrow \mathsf{Ext}_0(\mathsf{pk}_0; m, r; m', r')$
– return tk'.

Fig. 13: Our DTC scheme

Theorem D.2. *If H is a collision-resistant hash function, $\mathcal{H}_0, \mathcal{H}_1$ are secure chameleon hash functions, then our proposed construction in Fig. 13 is a secure DTC scheme.*

Proof. We check the required security properties for DTC.
• **Distribution of the keys.** Clearly, two distributions $\{\mathsf{KeyGen}(1^\kappa, i)\}$ and $\{\mathsf{TrChos}(1^\kappa, i)\}$ are indistinguishable.
• **Collision-resistance (CR).** Let $\mathcal{A}$ be an adversary against collision-resistance of DTC. We construct adversaries $\mathcal{B}_i$ against collision-resistance of $\mathcal{H}_i$ for $i = 0, 1$. Simply, $\mathcal{B}_i$ runs $\mathcal{H}_{1-i}.\mathsf{KeyGen}(1^\kappa)$ and returns $\mathsf{pk} = (\mathsf{pk}_0, \mathsf{pk}_1)$ and $\mathsf{tk}_{1-i}||1-i$ to $\mathcal{A}$. When $\mathcal{A}$ outputs tk_i, $\mathcal{B}_i$ uses it to find a collision.
• **Key-extractability (KE).** Here we discuss the untrusted setup, i.e., the game $\mathsf{KExt}^{\mathsf{UTr}}_{\mathsf{DTC},\mathcal{A}}(\kappa)$ in Fig. 11 where the extractor algorithm Ext is given in Fig. 13. Let $\mathcal{A}$ be the attacker to the KE of DTC, we construct the adversaries $\mathcal{B}_0$ and $\mathcal{B}_1$, $\mathcal{B}$ attacking, respectively, the KE of $\mathcal{H}_0$ and $\mathcal{H}_1$ and the CR of H.

Let $(\mathsf{pk}; (m, r, s), (m', r', s'))$ be the output of $\mathcal{A}$ in the KE game, meaning that $(m, r, s), (m', r', s')$ is a collision for the DTC scheme while $\mathsf{tk}' \neq \mathsf{tk}_0$ and $\mathsf{tk}' \neq \mathsf{tk}_1$ for a extracted tk' through Ext algorithm.

Three following cases are possible where w, w' are defined as $w = \mathcal{H}_0.\mathsf{CHash}(m, r)$ and $w' = \mathcal{H}_0.\mathsf{CHash}(m', r')$:
case 1. $(m, r) \neq (m', r')$ and $w = w'$
case 2. $(m, r) \neq (m', r')$ and $w \neq w'$ and $H(w) = H(w')$
case 3. rest (either $(m, r) = (m', r')$ (and thus $s \neq s'$) or $H(w) \neq H(w')$ must be the case)
Which in:
Case 1. $\mathcal{B}_0$ outputs $(\mathsf{pk}_0; (m, r), (m', r'))$.
Case 2. $\mathcal{B}$ outputs (w, w').
Case 3. $\mathcal{B}_1$ outputs $(\mathsf{pk}_1; (H(w), s), (H(w'), s')$.

Clearly in case 2, the adversary $\mathcal{B}$ breaks the CR of H. On the other hand, $\mathsf{tk}' \neq \mathsf{tk}_0$ and $\mathsf{tk}' \neq \mathsf{tk}_1$ which means $\mathcal{B}_0$ and $\mathcal{B}_1$ respectively break the KE of $\mathcal{H}_0$ and $\mathcal{H}_1$.

• **Distribution of collisions.** One needs to show that for random $r \xleftarrow{R} \mathcal{R}_0$ and $s \xleftarrow{R} \mathcal{R}_1$,

$$\{\mathsf{CHash}_{\mathsf{pk}}(m,r,s), m, m', \mathsf{Coll}(\mathsf{pk}, \mathsf{tk}_0, m', m, r, s)\}$$
$$\cong \{\mathsf{CHash}_{\mathsf{pk}}(m,r,s), m, m', \mathsf{Coll}(\mathsf{pk}, \mathsf{tk}_1, m', m, r, s)\}.$$

where $\cong$ stands for the indistinguishability of distributions. Clearly, if the distributions of collisions for $\mathcal{H}_0$ and $\mathcal{H}_1$ are uniform, the mentioned indistinguishability relation is satisfied. □

The CHF $\mathcal{H}$ required for the previous theorem can be realized by Pedersen-commitment [Ped92] where $\mathsf{CHash}(m,r) = g^m \mathsf{pk}^r$, $\mathsf{pk} = g^x$ and $\mathsf{tk} = x$. It is easily seen that this is a secure CHF in the untrusted setup. The exact construction is depicted in Fig. 14 where $\mathcal{G} = (\mathbb{G}, q, g)$ describe the group $\mathbb{G}$ of order q generated by g and bit length q is poly(κ). The group description $\mathcal{G}$ is the implicit input of all algorithms.

$\underline{\mathsf{KeyGen}(1^\kappa, \mathcal{G})}$: – sample $x \xleftarrow{R} \mathbb{Z}_q$ return $\mathsf{tk} = x$ and $\mathsf{pk} = g^x$.	$\underline{\mathsf{Coll}(\mathsf{pk}, \mathsf{tk}, m', m, r)}$: return $r' = \frac{m-m'}{\mathsf{tk}} + r$
$\underline{\mathsf{CHash}_{\mathsf{pk}}(m, r)}$: – sample $r \xleftarrow{R} \mathbb{Z}_q$ – compute $c = g^m \cdot \mathsf{pk}^r$ return c	$\underline{\mathsf{Ext}(\mathsf{pk}, m', m, r, r')}$: return $\mathsf{tk}' = \frac{m-m'}{r'-r}$

Fig. 14: Pedersen commitment as a CHF scheme [BKK90].

E DAPS-DRS Instantiation Based on Groth-Sahai Proof

In Fig. 15 we recap the DAPS-DRS construction. $F : \mathcal{K} \times \mathcal{U} \to \mathcal{S}$ is a value-key-binding PRF where $\mathcal{U}$ is the DAPS address space and $\mathcal{S}$ is the space of signing keys for sk_Σ, that is, the image of F is a subset of the signing-key space $\mathcal{S}$. The signature scheme $\Sigma = (\mathsf{KeyGen}_\Sigma, \mathsf{Sign}_\Sigma, \mathsf{Verif}_\Sigma)$ is expected to have a very weak level of security i.e., a public key is a OWF image of the corresponding signing key: $\mathsf{vk}_\Sigma = f(\mathsf{sk}_\Sigma)$. And $\Pi = (\mathsf{Setup}_\Pi, \mathsf{Prove}_\Pi, \mathsf{Verif}_\Pi)$ is a NIZK proof system.

Security analysis of DAPS-DRS. The following theorems analyze the security of DAPS-DRS scheme.

$\mathsf{KeyGen}(1^\kappa)$: – fix a signature scheme $\Sigma = (\mathsf{KeyGen}_\Sigma, \mathsf{Sign}_\Sigma, \mathsf{Verif}_\Sigma)$, run $(\mathsf{vk}_\Sigma, \mathsf{sk}_\Sigma) \leftarrow \mathsf{KeyGen}(1^\kappa)$ – fix a value-key-binding PRF $F : \mathcal{K} \times \mathcal{U} \to \mathcal{S}$, sample $\mathsf{sk}_{\mathsf{PRF}} \xleftarrow{R} \mathcal{K}$ compute $c = F(\mathsf{sk}_{\mathsf{PRF}}, \beta)$ for $\beta \in \mathcal{U}$ – fix a NIZK scheme $\Pi = (\mathsf{Setup}_\Pi, \mathsf{Prove}_\Pi, \mathsf{Verif}_\Pi)$ run $\mathsf{crs} \leftarrow \mathsf{Setup}_\Pi(1^\kappa)$. Return $\mathsf{sk} = (\mathsf{sk}_\Sigma, \mathsf{sk}_{\mathsf{PRF}})$ and $\mathsf{pk} = (\mathsf{vk}_\Sigma, \mathsf{crs}, (\beta, c))$.	$\mathsf{Sign}(\mathsf{sk}, m)$: Parse sk as $(\mathsf{sk}_\Sigma, \mathsf{sk}_{\mathsf{PRF}})$ and m as (a, p). – set $\gamma = F(\mathsf{sk}_{\mathsf{PRF}}, a)$, $z = p \cdot \gamma + \mathsf{sk}_\Sigma$ – $\pi \leftarrow \mathsf{Prove}_\Pi\big(\mathsf{crs}, (\mathsf{vk}_\Sigma, (\beta, c), z, m), (\mathsf{sk}_\Sigma, \mathsf{sk}_{\mathsf{PRF}}, \gamma)\big)$ Return (z, π) $\mathsf{Verif}(\mathsf{pk}, m, \sigma)$: Parse pk as $(\mathsf{vk}_\Sigma, \mathsf{crs}, (\beta, c))$, m as (a, p) and σ as (z, π). Return $\mathsf{Verif}_\Pi(\mathsf{crs}, (\mathsf{vk}_\Sigma, (\beta, c), z, m), \pi)$. $\mathsf{Ext}(\mathsf{pk}, m_1, m_2, \sigma_1, \sigma_2)$: Parse σ_i as $(z_i, .)$, m_i as (a_i, p_i). – If (m_1, m_2) is not compromising then return $\perp$. – Let $\mathsf{sk}_\Sigma \leftarrow \frac{z_1 p_2 - z_2 p_1}{p_2 - p_1}$ Return sk_Σ.

Fig. 15: DAPS scheme of DRS (DAPS-DRS)

$\mathsf{KeyGen}(1^\kappa)$: fix a signature scheme $\Sigma = (\mathsf{KeyGen}_\Sigma, \mathsf{Sign}_\Sigma, \mathsf{Verif}_\Sigma)$, with secret keys in $\mathcal{S} = \mathbb{G}_i$. run $(\mathsf{vk}_\Sigma, \mathsf{sk}_\Sigma) \leftarrow \mathsf{KeyGen}(1^\kappa)$ – fix PRF $F : \mathcal{K} \times \mathcal{U} \to \mathcal{S}$ and sample $\mathsf{sk}_{\mathsf{PRF}} \xleftarrow{R} \mathcal{K}$. – fix a commitment scheme $\mathcal{C} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$ such that $\mathsf{Commit} : \mathcal{K} \longrightarrow G_i$ run $\mathsf{pp}_\mathcal{C} \leftarrow \mathsf{Setup}_\mathcal{C}(1^\kappa)$ and $\mathcal{C}_{\mathsf{PRF}} \leftarrow \mathsf{Commit}(\mathsf{pp}_\mathcal{C}, \mathsf{sk}_{\mathsf{PRF}})$. – fix Groth-Sahai proof system $\Pi = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verif})$ run $\mathsf{crs} \leftarrow \mathsf{Setup}_\Pi^{GS}(1^\kappa)$ Return $\mathsf{sk} = (\mathsf{sk}_{\mathsf{PRF}}, \mathsf{sk}_\Sigma)$ and $\mathsf{pk} = (\mathsf{vk}_\Sigma, \mathsf{crs}, \mathsf{pp}_\mathcal{C}, \mathcal{C}_{\mathsf{PRF}})$.	$\mathsf{Sign}(\mathsf{pk}, \mathsf{sk}, m)$: Parse sk as $(\mathsf{sk}_{\mathsf{PRF}}, \mathsf{sk}_\Sigma)$, m as $(a, p) \in \mathcal{U} \times \mathbb{Z}_q$. – set $\gamma = F(\mathsf{sk}_{\mathsf{PRF}}, a)$, $z = \gamma^p \cdot \mathsf{sk}_\Sigma$ – $\pi \leftarrow \mathsf{Prove}_\Pi^{GS}(\mathsf{crs}, (\mathsf{vk}_\Sigma, \mathsf{pp}_\mathcal{C}, z, m, \mathcal{C}_{\mathsf{PRF}}), (\mathsf{sk}_\Sigma, \mathsf{sk}_{\mathsf{PRF}}, \gamma))$ Return $\sigma = (z, \pi)$ $\mathsf{Verif}(\mathsf{pk}, m, \sigma)$: Parse m as (a, p), σ as (z, π) and pk as $\mathsf{pk} = (\mathsf{vk}_\Sigma, \mathsf{crs}, \mathsf{pp}_\mathcal{C}, \mathcal{C}_{\mathsf{PRF}})$. Return $\mathsf{Verif}_\Pi^{GS}(\mathsf{crs}, (\mathsf{vk}_\Sigma, \mathsf{pp}_\mathcal{C}, z, m, \mathcal{C}_{\mathsf{PRF}}), \pi)$. $\mathsf{Ext}(\mathsf{pk}, m_1, m_2, \sigma_1, \sigma_2)$: Parse σ_i as $(z_i, .)$, m_i as (a_i, p_i). – if (m_1, m_2) is not compromising then return $\perp$. – let $\mathsf{sk}_\Sigma = (z_2^{p_1} \cdot z_1^{-p_2})^{(p_1 - p_2)^{-1}}$ Return sk_Σ.

Fig. 16: a DAPS scheme based on Groth-Sahai proofs (DAPS-GS).

Theorem E.1 (Unforgeability [DRS18a]). *If the NIZK proof system Π is simulation-sound extractable, F is a PRF, and f is a OWF, then the scheme DAPS-DRS is EUF-CMA secure.*

Theorem E.2 (Key-extractability [DRS18a]). *If the NIZK proof system Π is simulation-sound extractable and F is value-key-binding PRF, then the scheme DAPS-DRS is key-extractable in the trusted setup.*

Fig. 16 describes our DAPS-GS. In this construction $\mathbb{G}_i \in \{\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T\}$ where $\mathbb{G}_i$ is a cyclic group of order q and e is a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$. The signature scheme $\Sigma = (\mathsf{KeyGen}_\Sigma, \mathsf{Sign}_\Sigma, \mathsf{Verif}_\Sigma)$ is expected to have a very weak level of security i.e., a public key is a OWF image of the corresponding signing key: $\mathsf{vk}_\Sigma = f(\mathsf{sk}_\Sigma)$. And $\Pi = (\mathsf{Setup}_\Pi, \mathsf{Prove}_\Pi, \mathsf{Verif}_\Pi)$ is a NIZK proof system. The statements belong to a language L as follows;

$$L = \{(\mathsf{vk}_\Sigma, \mathsf{pp}_\mathcal{C}, z, m, \mathcal{C}_{\mathsf{PRF}}) \mid \exists(\mathsf{sk}_\Sigma, \mathsf{sk}_{\mathsf{PRF}}, \gamma) : ((\mathsf{vk}_\Sigma, \mathsf{pp}_\mathcal{C}, z, m, \mathcal{C}_{\mathsf{PRF}}), (\mathsf{sk}_\Sigma, \mathsf{sk}_{\mathsf{PRF}}, \gamma)) \in R^*\},$$

where the relation R^* is as

$$\gamma = F(\mathsf{sk}_{\mathsf{PRF}}, a) \ \wedge \ \mathcal{C}_{\mathsf{PRF}} = \mathsf{Commit}(\mathsf{pp}_\mathcal{C}, \mathsf{sk}_{\mathsf{PRF}}) \ \wedge \ z = \gamma^p \cdot \mathsf{sk}_\Sigma \ \wedge \ R(\mathsf{vk}_\Sigma, \mathsf{sk}_\Sigma) = 1,$$

and relation R is such that given vk_Σ it is not easy to find the witness sk_Σ.

E.1 An Overview of the Groth-Sahai Proof System

In this section, we consider three groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ of order p equipped with an admissible bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ and respectively with generators g_1,g_2 and $e(g_1, g_2)$. Groth and Sahai [GS08] presented an efficient NIZK proof system for languages defined over such bilinear groups; In Fig. 17 we overview the types of supported equations we require. The use of the instantiation (of GS proof) based on the SXDH assumption results in the proofs consisting of 6 group elements for each equation in $\mathbb{G}_1$ (similarly for $\mathbb{G}_2$), and poofs of size 8 for each pairing product equation. For each variable in $\mathbb{G}_1$, $\mathbb{G}_2$ or $\mathbb{Z}_p$ we have to add 2 group elements to the proof (the size of a commitment to the variable). The general size of the proof can thus be derived from the number (and type) of equations and variables.

Making Groth-Sahai simulation-extractable. In our DAPS-GS we need a NIZK proof system with simulation-sound-extractability. There are standard techniques for making a knowledge-sound/sound NIZK proof system (unbounded) simulation-sound [Gro06, BF14]: Let L be an NP language defined via $x \in L$ iff $\exists w \ s.t. \ R(x, w) = 1$ and Π be a Groth-Sahai proof. Then Π can be made (weakly) simulation-sound-extractable by proving that

$$R(x, w) = 1 \quad \vee \quad \mathsf{Verif}(\mathsf{vk}, x, \sigma) = 1 \tag{1}$$

where σ, belonging to group $\mathbb{G}_1$ or $\mathbb{G}_2$, is a EUF-CMA signature on the statements x. The verification key vk for the signature scheme is added to the CRS. Informally, this disjunction relation helps to simulate the queries

Variables:
$\mathcal{X}_1, \ldots, \mathcal{X}_m \in \mathbb{G}_1$, $y_1, \ldots, y_{n'} \in \mathbb{Z}_p$.
Multi-scalar multiplication equations in $\mathbb{G}_1$:

$$\sum_{i=1}^{n'} y_i \mathcal{A}_i + \sum_{i=1}^{m} b_i \mathcal{X}_i + \sum_{i=1}^{m}\sum_{j=1}^{n'} \gamma_{ij} y_j \mathcal{X}_i = \mathcal{T}_1$$

for constants $\mathcal{A}_i, \mathcal{T}_1 \in \mathbb{G}_1$ and $b_i, \gamma_{ij} \in \mathbb{Z}_p$
Multi-scalar multiplication equations in $\mathbb{G}_2$:

$$\sum_{i=1}^{n} a_i \mathcal{Y}_i + \sum_{i=1}^{m'} x_i \mathcal{B}_i + \sum_{i=1}^{m'}\sum_{j=1}^{n} \gamma_{ij} x_i \mathcal{Y}_j = \mathcal{T}_2$$

for constants $\mathcal{B}_i, \mathcal{T}_2 \in \mathbb{G}_2$ and $a_i, \gamma_{ij} \in \mathbb{Z}_p$
Pairing-product equations:

$$\prod_{i=1}^{n} e(\mathcal{A}_i, \mathcal{Y}_i) \cdot \prod_{i=1}^{m} e(\mathcal{X}_i, \mathcal{B}_i) \cdot \prod_{i=1}^{m}\prod_{j=1}^{n} e(\mathcal{X}_i, \mathcal{Y}_j)^{\gamma_{ij}} = t_T$$

for constants $\mathcal{A}_i \in \mathbb{G}_1$, $\mathcal{B}_i \in \mathbb{G}_2$, $t_T \in \mathbb{G}_T$, $\gamma_{ij} \in \mathbb{Z}_p$.

Fig. 17: Groth-Sahai's equations (for our language).

issued by the adversary (attacking to simulation-extractability), only based on signatures as the witnesses. When the adversary outputs a new statement x^* for which the related extractor fails to extract a witness, then there are two types of failures; Or no witness for disjunction relation could have been extracted, which breaks the knowledge-soundness of Groth-Sahai. Or, there is a witness for signature part, which gives a forgery for the signature scheme.[11]

The Fig. 18 is formally describing this conversion.

$\underline{\mathsf{Setup}_{\Pi'}(1^\kappa)}$: run $\mathsf{crs} \leftarrow \mathsf{Setup}_{\Pi}(1^\kappa)$ and $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\kappa)$
Return $\mathsf{crs}' = (\mathsf{crs}, \mathsf{vk})$

$\underline{\mathsf{Prove}_{\Pi'}(\mathsf{crs}', x, w)}$: run $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, x)$, run $\pi \leftarrow \mathsf{Prove}_{\Pi}(\mathsf{crs}, (x, \mathsf{vk}), (w, \sigma))$
Return $\pi' = \pi$.

$\underline{\mathsf{Verif}_{\Pi'}(\mathsf{crs}', x, \pi')}$: Return $\mathsf{Verif}_{\Pi}(\mathsf{crs}, (x, \mathsf{vk}), \pi)$

Fig. 18: Conversion from knowledge-sound to SE.

In the following lemma, we have assumed that the equations in the language $L = \{x \mid \exists\, w : R(x, w) = 1\}$, the equation $\sigma = \mathsf{Sign}_{\mathsf{vk}}(x)$, and their disjunction relation as Eq. (1) have the supported form by GS-proof.

Lemma E.3. *Let Π be a Grosh-Sahai proof system and Σ be an EUF-CMA signature . Then, Π' is a simulation-sound-extractable (SE) Groth-Sahai proof for the language L.*

Proof. Assume that there is an adversary $\mathcal{A}$ attacking the SE-security of Π'. When $\mathcal{A}$ outputs a pair (x^*, π^*), if it is a valid pair, then one of the statements (in the statement"$\vee$") should be correct. Meaning that, for a potentially extractable witness W^*, one of the following cases has happed,
case (1). $\mathsf{Verif}(\mathsf{vk}, x^*, W^*) = 1$.
case (2). $R(x^*, W^*) = 1$.
Which in the first case, it breaks the security of the signature scheme and in the second case it breaks the knowledge-soundness of Π. The formal proof is as follows,

Reduction from SE of Π' to EUF-CMA of Σ. Let $\mathcal{B}$ be an adversary attacking the EUF-CMA security of Σ. We show how the adversary $\mathcal{B}$ uses $\mathcal{A}$ to win its game.

– the adversary $\mathcal{B}$ receives the verification key vk from its challenger, it runs $(\mathsf{crs}, \xi) \longleftarrow \mathcal{E}_1(1^\kappa)$ and send $\mathsf{crs}' = (\mathsf{crs}, \mathsf{vk})$ to $\mathcal{A}$.

[11] Note that in the Groth-Sahai proof only the witnesses belonging to the groups $\mathbb{G}_1$, $\mathbb{G}_2$ can be extracted. But this is enough for our aim in DAPS-GS since sk_Σ belongs in these groups.

– when the adversary $\mathcal{A}$ issues a statement x, the adversary $\mathcal{B}$ sends x to its challenger to receive $\sigma = \mathsf{Sign}(\mathsf{sk}, x)$. Then it runs $\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, (x, \mathsf{vk}), \sigma)$ and sends back π to $\mathcal{A}$. It also adds (x, π) to the set Q.
– when $\mathcal{A}$ outputs a statement x^* and a proof π^*, The adversary $\mathcal{B}$ runs $W^* \leftarrow \mathcal{E}_2(\mathsf{crs}, \xi, x^*, \pi^*)$. Then, it outputs (x^*, W^*).

Clearly, in the case (1), (x^*, W^*) is a forgery for the signature scheme.

Reduction from SE of Π' to knowledge-soundness of Π. Let $\mathcal{B}'$ be the adversary attacking to the knowledge-soundness of Π for language L. It simulates the SE game for $\mathcal{A}$ as follows:

– The adversary $\mathcal{B}$ receives a simulated crs from its challenger, it runs $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\kappa)$ and sends back $\mathsf{crs}' = (\mathsf{crs}, \mathsf{vk})$ to $\mathcal{A}$.
– whenever $\mathcal{A}$ issues a query for the statement x, the adversary $\mathcal{B}$ generate $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, x)$ and runs $\pi \leftarrow \mathsf{Prove}_{\Pi}(\mathsf{crs}, (x, \mathsf{vk}), \sigma)$. Then it sends π to $\mathcal{A}$.
– when $\mathcal{A}$ outputs (x^*, π^*), the adversary $\mathcal{B}'$ returns it as its output.

Clearly, when the challenger runs $W^* \leftarrow \mathcal{E}_2(\mathsf{crs}, \xi, x^*, \pi^*)$, with the condition in the case (2), it verifies that $R(x^*, W^*) = 1$. □

Regarding the knowledge-soundness of GS-proof we have the following lemma.

Lemma E.4 ([GS08]). *From any valid proof, witnesses in $\mathbb{G}_i$ can be completely extracted, while for witness $a \in \mathbb{Z}_p$, we can extract g_i^a.*

Note that in our DAPS scheme, we just need the extraction of elements in group $\mathbb{G}_i$. Thus, it does not hurt to assume the GS-proof is knowledge-sound.

Waters's signatures (defined below) are defined over bilinear groups so that signatures consist of group elements only. Thus, the scheme fits to instantiate our DAPS scheme and it also can properly instantiate the simulation-sound-extractable transform just discussed.

To prove a disjunction relation like equation (1) in the Groth-Sahai framework, one needs to represent it as a conjunctions of equations. Groth [Gro06] already suggested an elegant technique for this conversion by adding a variable and equations in a special way (see also [BF14, Appendix C of the full version]). Assume that we want a disjunction relation between some equations of the following form where P and Q are known constants:

$$eq(*): \qquad \prod_{i=1}^{n} e(\mathcal{A}_i, \mathcal{Y}_i) \cdot \prod_{i=1}^{m} e(\mathcal{X}_i, \mathcal{B}_i) \cdot \prod_{i=1}^{m}\prod_{j=1}^{n} e(\mathcal{X}_i, \mathcal{Y}_i)^{\gamma_{ij}} = e(P, Q)$$

Then each of the above equations would be replaced with the following equations where $\mathcal{Q}'$ and $\mathcal{T}$ are variables:

$$\prod_{i=1}^{n} e(\mathcal{A}_i, \mathcal{Y}_i) \cdot \prod_{i=1}^{m} e(\mathcal{X}_i, \mathcal{B}_i) \cdot \prod_{i=1}^{m}\prod_{j=1}^{n} e(\mathcal{X}_i, \mathcal{Y}_i)^{\gamma_{ij}} = e(P, \mathcal{Q}'), \quad e(\mathcal{T}, Q^{-1} \cdot \mathcal{Q}') = 1.$$

Note that if $\mathcal{T} \neq \mathbb{1}$, then the only solution is $\mathcal{Q}' = Q$, which in turn means that $eq(*)$ is satisfied. Now for each equation $eq(*)$ we have a separate variable $\mathcal{T}$. To express a disjunction of two equations of this kind, we add $e(\mathcal{T}_1 \cdot \mathcal{T}_2 \cdot g^{-1}, h) = 1$, which guarantees that not both $\mathcal{T}_1$ and $\mathcal{T}_2$ are $\mathbb{1}$ and thus one of the equations $eq(*)$ must be satisfied.

E.2 Instantiation of DAPS-GS

As we mentioned in the previous section, Groth-Sahai proofs work for statements over bilinear groups. Thus to use Groth-Sahai proofs when instantiating DAPS-DRS, we need to find proper instantiations of the statements over bilinear groups. Here are some theorems regarding such possible instantiations.

Theorem E.5 (Naor-Reingold's PRF [NR97]). *If the DDH assumption holds in the group $\mathbb{G}$, then the function F defined as follows is a PRF.*

$$F_k(x) = F_0^w \quad \textit{where } F_0 = g^{k_0} \in \mathbb{G}, k = (F_0, k_1, \ldots, k_n),\ x_i \in \{0,1\},\ w = \prod_{i=1}^{n} k_i^{x_i}$$

The signature scheme Σ can be instantiated with Waters' signature scheme since the signatures of this scheme are elements of bilinear groups. The following gives an improvement of Waters' scheme and its security analysis [HK08] over asymmetric bilinear maps [BFPV11]. The need for an asymmetric variant of Waters's signature comes from our instantiation for PRF which needs DDH assumption.

$\underline{\mathsf{KeyGen}(1^\kappa)}$: select cyclic groups $\mathbb{G}_i$ of prime order q with generator g_i and bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$.
– sample $h, h_0, h_1, \ldots, h_l \xleftarrow{R} \mathbb{G}_1$ and $\alpha, \xleftarrow{R} \mathbb{Z}_q$.
– fix a hash-function $\mathcal{H} : \{0,1\}^l \longrightarrow \mathbb{G}$ where $\mathcal{H}(m) = h_0 \prod_{i=1}^{l} h_i^{m_i}$ and m_i is the ith bit of m.
– set $\mathsf{pp} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, (h_i)_i)$
Return $\mathsf{pk} = (h, g_2^\alpha, g_1^\alpha)$, pp and $\mathsf{sk} = h^\alpha$.

$\underline{\mathsf{Sign}(\mathsf{pk}, \mathsf{sk}, m)}$: sample $r \xleftarrow{R} \mathbb{Z}_q$ and compute $\sigma_1 = \mathsf{sk} \cdot \mathcal{H}(m)^r, \sigma_2 = g_1^{-r}, \sigma_3 = g_2^{-r}$.
Return $\sigma = (\sigma_1, \sigma_2, \sigma_3)$.

$\underline{\mathsf{Verif}(\mathsf{pk}, \sigma, m)}$: parse pk as $(\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{pk}_3)$ Return 1 iff $e(\sigma_1, g_2) \cdot e(\mathcal{H}(m), \sigma_3) = e(\mathsf{pk}_1, \mathsf{pk}_2)$ and $e(\sigma_2, g_2) = e(g_1, \sigma_3)$.

Fig. 19: Asymmetric variant of Waters' signature scheme [BFPV11, Wat05].

Theorem E.6 (Waters' signature scheme [Wat05, HK08, BFPV11]). *If the CDH$^+$ assumption holds, then the scheme in Fig. 19 is UNF-CMA secure.*[12]

To instantiate $\mathcal{C}$, we use Pedersen commitments.

Theorem E.7 (Pedersen commitments [Ped92]). *The commitment scheme in Fig. 20 is perfectly hiding and computationally binding under the discrete logarithm assumption in group G.*

$\underline{\mathsf{Setup}(1^\kappa)}$: select a group $\mathbb{G}$ of prime order q with generator g.
Sample $y \xleftarrow{R} \mathbb{G}$ and return $\mathsf{pp} = (g, y)$.

$\underline{\mathsf{Commit}(\mathsf{pp}, m)}$: sample $r \xleftarrow{R} \mathbb{Z}_q$ and compute $c = g^m \cdot y^r$. Return c and $d = (m, r)$.

$\underline{\mathsf{Verif}(\mathsf{pp}, c, d = (m, r))}$: Return 1 if $c = g^m \cdot y^r$.

Fig. 20: Pedersen's commitment scheme [Ped92].

In Figs. 21 and 22 we show how to combine this instantiations together to get a Groth-Sahai compatible system of statements.

[12] CDH$^+$ is a variant of CDH in asymmetric bilinear groups. The adversary is given $(g_1, g_2, g_1^a, g_1^b, g_2^a)$ and its goal is to compute g_1^{ab} (see [BFPV11]).

System	Description
S_1	publicly knowns: (a,p), $\mathsf{vk}_\Sigma, \mathsf{pp}_\mathcal{C}$, $\mathcal{C}_{\mathsf{PRF}}$ and z variables: $\mathsf{sk}_{\mathsf{PRF}}$, sk_Σ. statements: $\gamma = F(\mathsf{sk}_{\mathsf{PRF}}, a),\ \mathcal{C}_{\mathsf{PRF}} = \mathsf{Commit}(\mathsf{pp}_\mathcal{C}, \mathsf{sk}_{\mathsf{PRF}})$ $z = \gamma^p \cdot \mathsf{sk}_\Sigma,\ R(\mathsf{sk}_\Sigma, \mathsf{vk}_\Sigma) = 1$
remark	instantiation with: Naor-reingold PRF F; Waters' signature scheme Σ; Pedersen's commitment scheme
S_2	publicly knowns: (a,p), $\mathsf{vk}_\Sigma, \mathsf{pp}_\mathcal{C}$, $\mathcal{C}_{\mathsf{PRF}}$ and z variables: $\mathsf{sk}_{\mathsf{PRF}}$, sk_Σ,α_i and α. statements: $\gamma = F(\mathsf{sk}_{\mathsf{PRF}}, a) = g^{k_0}\prod k_i^{a_i}$ where $\mathsf{sk}_{\mathsf{PRF}} = (k_0,\ldots,k_n)$; $c_i = g_1^{k_i} y^{\alpha_i}\quad i = 0,\ldots,n,\quad y \in \mathbb{G}_1$; $z = \gamma^p \cdot \mathsf{sk}_\Sigma \in \mathbb{G}_1,\quad \mathsf{pk}_3 = g_1^\alpha,\ \mathsf{pk}_2 = g_2^\alpha,\ \mathsf{sk}_\Sigma = \mathsf{pk}_1^\alpha \in \mathbb{G}_1$
remark	representation of the statements in quadratic forms
S_4	publicly knowns: (a,p), $\mathsf{vk}_\Sigma, \mathsf{pp}_\mathcal{C}$, $\mathcal{C}_{\mathsf{PRF}}$ and z variables: $\mathsf{sk}_{\mathsf{PRF}} = (k_0, k_1, \ldots, k_n)$, sk_Σ, α_i, F_l and α. statements: Quadratic form of PRF: $F_0 \cdot g^{k_0} = \mathbb{1}_{\mathbb{G}_1}$; $F_{l+1}^{-1} \cdot F_l^{k_{l+1}} = \mathbb{1}_{\mathbb{G}_1} \quad a_{l+1} = 1$; $F_{l+1}^{-1} \cdot F_l = \mathbb{1}_{\mathbb{G}_1} \quad a_{l+1} = 0$ $c_i = g^{k_i} y^{\alpha_i}$ $z = F_n^p \cdot \mathsf{sk}_\Sigma \in \mathbb{G}_1 \quad \mathsf{pk}_3 = g_1^\alpha,\ \mathsf{pk}_2 = g_2^\alpha,\ \mathsf{sk}_\Sigma = \mathsf{pk}_1^\alpha$
remark	extension to simulation-soundness;

Fig. 21: Transformation of statements to a Groth-Sahai-compatible form.

System	Description
remark	extension to simulation-soundness;
S_5	publicly knowns: (a,p), $\mathsf{vk}_\Sigma, \mathsf{pp}_\mathcal{C}, \mathcal{C}_{\mathsf{PRF}}$ and z variables: $\mathsf{sk}_{\mathsf{PRF}} = (k_0, k_1, \ldots, k_n)$, sk_Σ, α_i, F_l, α, $\sigma = (\sigma_1, \sigma_2, \sigma_3)$. statements 1: $\begin{cases} F_0 \cdot g^{k_0} = \mathbb{1}_{\mathbb{G}_1} \\ F_{l+1}^{-1} \cdot F_l^{k_{l+1}} = \mathbb{1}_{\mathbb{G}_1} \quad a_{l+1} = 1 \\ F_{l+1}^{-1} \cdot F_l = \mathbb{1}_{\mathbb{G}_1} \quad a_{l+1} = 0 \\ c_i = g^{k_i} y^{\alpha_i} \\ z = F_n^p \cdot \mathsf{sk}_\Sigma \in \mathbb{G}_1 \quad \mathsf{pk}_3 = g_1^\alpha,\ \mathsf{pk}_2 = g_2^\alpha,\ \mathsf{sk}_\Sigma = \mathsf{pk}_1^\alpha \end{cases}$ OR statement 2: $e(\sigma_1, g_2) \cdot e(\mathcal{H}(\mathsf{pp}_\mathcal{C}, \mathsf{vk}_\Sigma, z, a, p, \mathcal{C}_{\mathsf{PRF}}), \sigma_3) = e(\mathsf{pk}_1, \mathsf{pk}_2)$, $e(\sigma_2, g_2) = e(g_1, \sigma_3)$
remark	conversion from disjunction to conjunction
S_6	publicly knowns: (a,p), $\mathsf{vk}_\Sigma, \mathsf{pp}_\mathcal{C}, \mathcal{C}_{\mathsf{PRF}}$ and z variables: $\mathsf{sk}_{\mathsf{PRF}} = (k_0, k_1, \ldots, k_n)$, sk_Σ, α_i, F_l and $\alpha, \sigma = (\sigma_1, \sigma_2, \sigma_3)$. statements: $\begin{cases} F_0 \cdot g^{k_0} \cdot Q_0^f = \mathbb{1},\ F_{l+1}^{-1} \cdot F_l^{k_{l+1}} \cdot Q_l^f = \mathbb{1}\ a_{l+1} = 1, \\ F_{l+1}^{-1} \cdot F_l \cdot Q_l^f = \mathbb{1}\ a_{l+1} = 0 \\ c_i = Q_i^c \cdot g^{k_i} y^{\alpha_i} \\ z = Q_z \cdot F_n^p \cdot \mathsf{sk}_\Sigma \in \mathbb{G}_1 \\ g_1^\alpha \cdot Q_3^\Sigma = \mathsf{pk}_3,\ e(g_1, g_2^\alpha) = e(Q_2^\Sigma, \mathsf{pk}_2),\ \mathsf{sk}_\Sigma^{-1} \cdot \mathsf{pk}_1^\alpha \cdot Q_1^\Sigma = \mathbb{1} \\ - \\ e(\sigma_1, g_2) \cdot e(\mathcal{H}(\mathsf{pp}_\mathcal{C}, \mathsf{vk}_\Sigma, z, a, p, \mathcal{C}_{\mathsf{PRF}}), \sigma_3) = e(Q_1, \mathsf{pk}_2) \\ e(\sigma_2, g_2) \cdot e(g_1^{-1}, \sigma_3) = e(Q_2, g_2) \\ - \\ e(Q_0^f, T) = \mathbb{1},\ e(Q_l^f, T) = \mathbb{1},\ e(Q_i^c, T) = \mathbb{1}\ e(Q_z, T) = \mathbb{1}, \\ e(Q_3^\Sigma, T) = \mathbb{1}, e(Q_2^\Sigma, T) = \mathbb{1}, e(Q_1^\Sigma, T) = \mathbb{1} \\ - \\ e(\mathsf{pk}_1^{-1} \cdot Q_1, T') = 1, e(Q_2, T') = 1 \\ - \\ e(g_1^{-1}, T \cdot T' \cdot g_2) = 1 \end{cases}$

Fig. 22: Transformation of statements to a Groth-Sahai-compatible form.