Stronger Security and Constructions of Multi-Designated Verifier Signatures *

Ivan Damgård[1], Helene Haagh[12], Rebekah Mercer[3],
Anca Nitulescu[12], Claudio Orlandi[1], and Sophia Yakoubov[1]

[1] Aarhus University {ivan, orlandi, sophia.yakoubov}@cs.au.dk,
[2] {helenehaagh, anca.nitulesc}@gmail.com,
[3] rebekah@o1labs.org, O(1) Labs, USA

Abstract. Off-the-Record (OTR) messaging is a two-party message authentication protocol that also provides plausible deniability: there is no record that can later convince a third party what messages were actually sent. The challenge in *group OTR*, is to enable the sender to sign his messages so that group members can verify who sent a message (signatures should be *unforgeable*, even by group members). Also, we want the off-the-record property: even if some verifiers are corrupt and collude, they should not be able to prove the authenticity of a message to any outsider. Finally, we need *consistency*, meaning that if any group member accepts a signature, then all of them do.

To achieve these properties it is natural to consider Multi-Designated Verifier Signatures (MDVS). However, existing literature defines and builds only limited notions of MDVS, where (a) the off-the-record property (*source hiding*) only holds when *all* verifiers could conceivably collude, and (b) the consistency property is not considered.

The contributions of this paper are two-fold: stronger definitions for MDVS, and new constructions meeting those definitions. We strengthen source-hiding to support any subset of corrupt verifiers, and give the first formal definition of consistency. We build three new MDVS: one from generic standard primitives (PRF, key agreement, NIZK), one with concrete efficiency and one from functional encryption.

1 Introduction

Encrypted and authenticated messaging has experienced widespread adoption in recent years, due to the attractive combination of properties offered by, for example, the Signal protocol [Mar13]. With so many conversations happening over the internet, there is a growing need for protocols offering security to conversation participants. Encryption can be used to guarantee privacy of message contents, but authenticating messages while maintaining the properties of an

* This research was supported by: the Concordium Blockhain Research Center, Aarhus University, Denmark; the Carlsberg Foundation under the Semper Ardens Research Project CF18-112 (BCM); the European Research Council (ERC) under the European Unions's Horizon 2020 research and innovation programme under grant agreement No 669255 (MPCPRO) and No 803096 (SPEC); the Danish Independent Research Council under Grant-ID DFF-6108-00169 (FoCC); the NSF MACS project.

in person conversation is more involved. There are two properties of in person conversations related to authenticity that we wish to emulate in the context of digital conversations:

- *Unforgeability*, meaning that the receiver should be convinced that the message actually came from the sender in question, and
- *Off-the-record* or *deniability*, meaning that the receiver cannot later prove to a third party that the message came from the sender.

Off-the-record (OTR) messaging offers a solution to this in the two-party case, enabling authentication of messages such that participants can convincingly deny having made certain statements, or even having taken part in the conversation at all [BGB04]. The protocol deals with encrypted messages accompanied by a message authentication code (MAC) constructed with a shared key. MACs work well in two-party conversations, because for parties S(ender) and R(eciever) with a shared secret key, a MAC attests 'this message comes from S or R'. MACs provide unforgeability, since a party R receiving a message authenticated with such a MAC knows that if this MAC verifies, the message came from S. MACs provide off-the-record (deniable) communication as R cannot convince a third party that a message and MAC originally came from S (since R could have produced it just as easily). More generally, tools that provide unforgeable, off-the-record two-party communication are known as *Designated Verifier Signatures* (DVSs, proposed by [JSI96] and [Cha96]).

When there are multiple recipients, for example in group messaging, the situation becomes more complicated. The need for unforgeability can be generalized as the need for all parties in the group to agree on a conversation transcript. There are two components to this: unforgeability, as before, and *consistency*, which requires that if one recipient can verify a signature, they all can. Without the consistency property, a signer could send a message that only one recipient could verify; that recipient would then be unable to convince the rest to accept that message, and would disagree with them about the transcript of the conversation.

DVSs have been extended to the multiparty setting under the name of *Multi-Designated Verifier Signatures (MDVSs)* (we give a number of references in Figure 1). One might hope that these schemes would work for off-the record group messaging; however, it turns out that existing MDVS definitions and schemes do not have the properties one would naturally ask for. In the following section, we give a motivating example illustrating which properties we should actually ask from an MDVS scheme, and we explain how existing schemes fall short of providing them.

1.1 A Motivating Example for MDVS

Imagine a government official Sophia who wants to blow the whistle on some corrupt government activity; e.g., perhaps her colleague, $\mathcal{A}$aron, accepted a bribe.[4]

[4] Ring signatures [RST01] can be similarly used in this context; Sophia could use a ring signature scheme to sign in such a way that anyone could verify that the signature

She wants to send a message describing this corruption to Robert, Rachel and Rebekah, who are all Reporters at national newspapers.

Naturally, Sophia wants the Reporters to be convinced that she is the true sender of the message. Otherwise, they would have no reason to believe — or print — the story.

Goal 1 (Unforgeability) *It is vital that each of the* R*eporters be able to authenticate that the message came from* S*ophia.*

In order to achieve unforgeability, Sophia produces a signature σ using an MDVS scheme, and attaches it to her message. (In such a scheme, each sender has a private signing key and each recipient has a private verification key.) However, blowing the whistle and reporting on $\mathcal{A}$aron's corrupt activity could put Sophia in danger. If any of Robert, Rachel or Rebekah could use σ to demonstrate to $\mathcal{A}$aron that Sophia blew the whistle on him, she could lose her position, or face other grave consequences.

Goal 2 (Source-Hiding / Off-the-Record) *It is vital that the* R*eporters be unable to prove to an outsider ($\mathcal{A}$aron) that the message came from* S*ophia.*

One way to guarantee that the Reporters cannot link Sophia to the message is to require that the Reporters can *simulate* a signature σ themselves. Then, if they try to implicate Sophia by showing σ to $\mathcal{A}$aron, he would have no reason to believe them; as far as he is concerned, the Reporters could have produced σ to try to frame Sophia.

All previous constructions only support off-the-record in the limited sense that *all* of the Reporters must collaborate in order to produce a simulated signature.[5] However, this is insufficient. Suppose, for instance, that $\mathcal{A}$aron knows Rachel was undercover — and thus unreachable — for the entire time between the bribery taking place, and Robert and Rebekah bringing σ to $\mathcal{A}$aron. Then he would conclude that Rachel could not have collaborated in simulating σ, and so it must be genuine. Even with the off-the-record definition used in prior works, it is still possible that some subset of the Reporters would be able to implicate Sophia in the eyes of $\mathcal{A}$aron. We therefore need a stronger off-the-record defintion.

---

came from someone in her organization, but not that it came from her, specifically. However, MDVS has an advantage here, since it is possible that $\mathcal{A}$aron only doubts the trustworthiness of one of his colleagues; if Sophia uses a ring signature, that signature would convince $\mathcal{A}$aron that she was the signer, but if she uses an MDVS signature, $\mathcal{A}$aron wouldn't know whether she was the true signer, or whether the signature was just a simulation (even if Sophia was his only suspect).

In the context of a group off-the-record conversation, MDVS signatures are clearly the right tool, as members of the group should learn the identity of the sender of each message.

[5] One previous work [Tia12] allows a single verifier to simulate a signature. However, in this construction a simulated signature created by a malicious verifier will look like a real signature for all other designated verifers, violating unforgeability.

Contribution 1 (Off-the-record For Any Subset) *We give a stronger definition of the off-the-record property, where* any *subset of* Reporters *must be able to simulate a signature. A simulation looks like a genuine signature to an outsider, even given the verification keys of the subset that produced it (as well as a number of other signatures that are guaranteed to be genuine).*

Under our stronger definition, no set of Reporters is able to use σ to provably tie Sophia to the message *even if Aaron has side information about communication amongst the Reporters* as well as guaranteed-to-be-genuine signatures.

Remark 1. (The Tension Between Off-The-Record and Unforgeability) Note that, if Rachel did not participate in Robert and Rebekah's signature simulation (e.g. if she was undercover at the time), she will later be able to distinguish the simulation from a real signature produced by Sophia. Otherwise, Robert and Rebekah would have succeeded in producing a forgery that fools Rachel.

This means that under a sufficiently strong model of attack, we cannot have unforgeability and off-the-record at the same time. Namely, suppose Aaron first gets a signature σ from Robert and Rebekah, while preventing them from communicating with Rachel. Then he coerces Rachel into giving him her secret verification key. By the unforgeability property, he can use this key to tell if σ is a simulation. (Note that Aaron will be able to tell whether Rachel gives him her true verification key, since he may have other signatures from Sophia that he knows are genuine that he can use to test it. So, she has no choice but to hand over her real verification key.)

Given this observation, we choose to explore the model where the secret keys of all coerced/corrupted verifiers (but not honest ones) can be used to simulate a signature, as this is the strongest model of attack in which both unforgeability and off-the-record can be achieved. As we shall see, even in this model, achieving both properties requires highly non-trivial constructions and implies a lower bound on the size of signatures.

Finally, let us fast forward to the moment when Robert, Rachel and Rebekah receive Sophia's message. They want to print this high-profile story as soon as possible, but of course they want to be sure they won't make themselves look foolish by printing the story if their colleagues — the other well-respected Reporters listed as recipients — don't believe it actually came from Sophia. The concern here is that Sophia could be dishonest and her actual goal could be to discredit the Reporters. Hence we need another property — consistency, or designated verifier transferability.

Goal 3 (Consistency / Designated Verifier Transferability) *It is desirable that, even if Sophia is malicious, if one of the Reporters can authenticate that the message came from Sophia, all of them can.*

Contribution 2 *We provide the first formal definition of consistency.*

Now that we have covered the basic storyline, let us consider a few possible plot-twists. First, what if Aaron is tapping the wires connecting the government building to the outside world? Then he will see Sophia's message — together with her signature σ — as she sends it to the Reporters. In such a situation, we would want the signature σ not to give Sophia— or the Reporters— away.

Goal 4 (Privacy of Identities) *It is desirable that σ shouldn't reveal* Sophia's *or the* Reporters' *identities* [6]*. When only the signer's — Sophia's — identity is hidden, this property is called* privacy of signer identity (PSI)*.*

Next, what if, at the time at which Sophia has the opportunity to send out her message, she cannot look up Rebekah's public key securely — perhaps because Rebekah has not yet set up an account on the secure messaging system Sophia uses? Then, it would be ideal for Sophia to need nothing other than Rebekah's identity (and some global public parameters) in order to include her as a designated verifier. Rebekah would then be able to get the appropriate key from a trusted authority such as the International Press Institute[7] (having proved that she is, in fact, Rebekah), and would be able to use that key to verify Sophia's signature.

Goal 5 (Verifier-Identity-Based (VIB) Signing) *It is desirable that* Sophia *should only need the* Reporters' *identities, not their public keys, in order to produce her designated verifier signature.*

Contribution 3 *We give the first three constructions that achieve unforgeability, off-the-record with any-subset simulation, and consistency. One of them additionally achieves privacy of identities and verifier-identity-based signing.*

The third construction achieves privacy of verifiers identities (PVI) even if the secret signing key is leaked (but not the random coins used to produce the signature). This is a stronger flavor of the PVI notion with more possible applications, such as *Post-Compromise Anonymity* guarantees.

This last construction, may, at first glance, seem strictly better; however, the price it pays is two-fold: It uses functional encryption (which requires strong computational assumptions), and it requires an involved trusted setup in which a master secret is used to derive verifier keys. Note that such a trusted setup is clearly necessary in order to achieve verifier-identity-based signing.

In contrast, our first two constructions can be instantiated either in the random oracle model, or with a common reference string — in both cases avoiding the need for a master secret key. They use only standard primitives such as pseudorandom functions, pseudorandom generators, key agreement and NIZKs. The first construction uses these primitives in a black-box way; the second construction uses specific instances of these primitives, for concrete efficiency.

[6] Note that privacy of identities is related to — but very different from — off-the-record. Neither of these definitions is strictly stronger than the other. *Privacy of identities* is weaker in that it assumes that none of the Reporters help in identifying Sophia as the sender, while *off-the-record* makes no such assumptions. However, *privacy of identities* is stronger in that it requires that σ alone reveal nothing about Sophia's identity to anyone other than the Reporters; *off-the-record* allows such leakage, as long as it is not provable.

[7] This trusted authority can also be distributed; perhaps the master secret is secret-shared across several different institutions, who must collaborate in order to produce a secret verification key.

In the following subsections, we give an overview of previous work and then discuss our results in more detail. The main challenge of building stronger MDVS schemes is combining the three core properties we strive for: unforgeability, off-the-record for any subset, and consistency. This is highly non-trivial.

1.2 Flavors of Multi-Designated Verifier Signatures

There are many ways to define MDVS and its properties. Figure 1 summarizes the approaches taken by prior work, compared to our own.

Verification There are several different flavors of verification. In some MDVS schemes, even a single designated verifier cannot link a signature to the signer; the designated verifiers need to work together in order to verify a signature. Thus, we have two notions of verification: *local* verification and *cooperative* verification (where *all* designated verifiers need to cooperate in order to verify the signature).

Remark 2. In the schemes with cooperative verification, we need not additionally require consistency, since we have it implicitly: verifiers will agree on the verification decision they reach together. The notion of consistency is non-interactive, and more challenging to achieve in schemes with local verification.

Simulation Recall that the off-the-record property states that an outsider cannot determine whether a given signature was created by the signer or simulated by the designated verifiers. We have three flavors of such simulateability: *one* designated verifier (out of n) can by himself simulate a signature (as done by [Tia12])[8], *all* designated verifiers need to collude in order to simulate a signature (all other works on MDVS), or *any subset* of the designated verifiers can simulate a signature (this paper). Of course, the simulated signature should remain indistinguishable from a real one even in the presence of the secrets held by the simulating parties.

Unforgeability There is also the standard security property of signature schemes, which is *unforgeability*; no one (except the signer) should be able to construct a signature that any verifier will accept as a valid signature from that signer. There are two flavors of unforgeability. The first is *weak* unforgeability, where designated verifiers can forge, but others cannot. The second is *strong* unforgeability, where a designated verifier can distinguish between real signatures and signatures simulated by other verifiers; that is, even other designated verifiers cannot fool a verifier into accepting a simulated signature.[9] (In the weak unforgeability game, the adversary does not have access to any designated verifier

[8] If *only* one designated verifier can simulate a signature, it must be distinguishable from a real signature by other verifiers (by the strong unforgeability property). Two colluding verifiers would be able to prove to an outsider that a given signature is not a simulation by showing that it verifiers for both of them. So, any-subset simulation gives strictly stronger off-the-record guarantees than one-verifier simulation.

[9] Note that when all designated verifiers are needed for the simulation, then a designated verifier will be able to distinguish a simulation from a real signature based on whether he participated in the simulation of the signature. However, if this is the only way he can distinguish, then the signature scheme has *weak* unforgeability, since the simulated signature is still a valid forgery.

keys; in the strong unforgeability game, the adversary is allowed access to some such keys.) Since strong unforgeability is the notion of unforgeability we require, in the rest of this paper, *unforgeability* refers to *strong* unforgeability (unless otherwise specified).

Schemes	PSI	Verification	Simulation	Unforgeability	Signature Size	Consistency
[JSI96,Ver06]	No	All	All	Weak	$O(1)$	N/A
[Cho08]	No	Local	All	Weak	$O(1)$	Yes
[LSMP07]	No	Local	All	Weak	$O(1)$	No
[ZAYS12]	No	Local	All	Strong	$O(1)$	Yes
Our work, from standard primitives	No	Local	Any subset $\mathcal{C}$	Strong	$O(\|\mathcal{D}\|)$	Yes
[NSM05,Cho06]	Yes	All	All	Weak	$O(\|\mathcal{D}\|)$	N/A
[MW08,SHCL08,Cha11]	Yes	All	All	Weak	$O(1)$	N/A
[LV04]	Yes	Local	All	Weak	$O(1)$	Yes
[SKS06]	Yes	Local	All	Weak	$O(\|\mathcal{D}\|)$	No
[Ver06,LV07]	Yes	Local	All	Weak	$O(\|\mathcal{D}\|)$	Yes
[ZAYS12]	Yes	Local	All	Strong	$O(\|\mathcal{D}\|)$	Yes
[Tia12]	Yes	Local	One	Weak	$O(1)$	Yes
Our work, from FE	Yes	Local	Any subset $\mathcal{C}$ of size up to t	Strong	$O(t)$	Yes

Fig. 1. MDVS constructions and their properties. Let $\mathcal{D}$ be the set of designated verifiers, and $t \leq |\mathcal{D}|$ be an upper bound on the set of colluding designated verifiers $\mathcal{C} \subseteq \mathcal{D}$.

1.3 Our Contributions

We propose formal definitions of all the relevant security properties of MDVS in the strongest flavor, including the definition of off-the-record with any-subset simulation. We also give the first formal (game based) definition of consistency, where a corrupt signer can collude with some of the designated verifiers to create an inconsistent signature.

We then give several different constructions of MDVS that achieve these properties, including local verification, off-the-record with any-subset simulation, and strong unforgeability. Our constructions, and the tools they require, are mapped out in Figure 2. In particular, these are the first constructions that combine any-subset simulation and with strong unforgeability, as described in Figure 1. We get these results at the expense of signature sizes that are larger than in some of the earlier constructions. However, this is unavoidable, as shown in Theorem 1 below.

Theorem 1. *Any MDVS with any-subset simulation and strong unforgeability must have signature size* $\Omega(|\mathcal{D}|)$.

Remark 3. It may seem from the table that our functional encryption based scheme contradicts the theorem, but this is not the case. It can be instantiated such that signatures can be simulated by collusions up to a certain maximal size t, and then signatures will be of size $\Omega(|\mathcal{C}|)$. However, if we want *any* subset to be able to simulate, the signature size is $\Omega(|\mathcal{D}|)$, in accordance with the theorem.

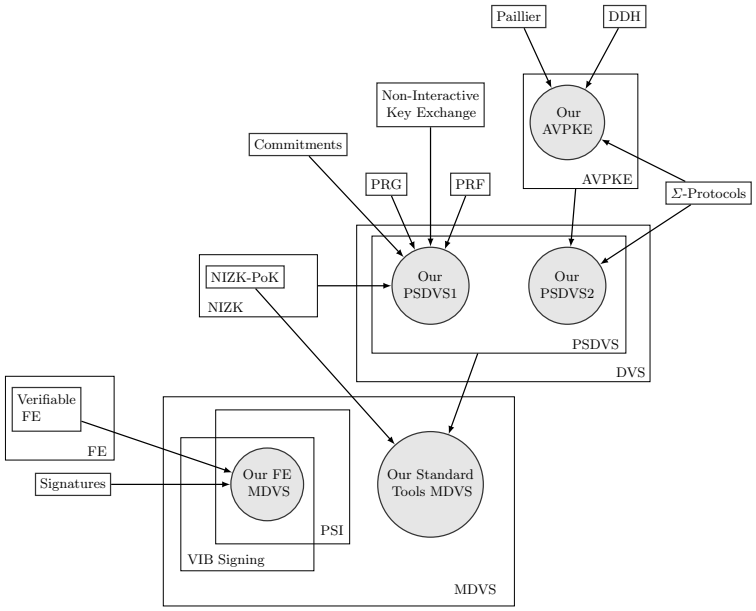

Fig. 2. Our MDVS Constructions and Building Blocks

Proof. Imagine that we give all the verifiers' keys to a sender and a receiver; the sender can now encode an arbitrary subset $\mathcal{C} \subseteq \mathcal{D}$ by letting $\mathcal{C}$ construct a simulated signature σ on some default message, and sending it to the receiver. The receiver can infer $\mathcal{C}$ from σ: by strong unforgeability, all verifiers' keys outside $\mathcal{C}$ will reject σ, whereas keys in $\mathcal{C}$ will accept, since we require the simulation to look convincing even given the secret keys in $\mathcal{C}$. It follows that σ must consist of enough bits to determine $\mathcal{C}$, which is $\log_2(2^{|\mathcal{D}|}) = |\mathcal{D}|$. □

Why First Ideas Fail

Using MACs Black-box usage of a standard MAC scheme cannot help us combine unforgeability with consistency.[10] There are two straightforward ways to use a standard MAC scheme in this context: sharing a MAC key among the entire group, and sharing MAC keys pairwise. Sharing a single key does not provide the desired notion of unforgeability, since any member of the group can forge messages from any other member. Sharing keys pairwise does not provide the desired notion of consistency. If recipients R_1 and R_2 are the chosen recipients of a message, and R_1 receives a message he accepts as coming from S, he cannot be sure that R_2 would also accept that message: If S is corrupt, he could include a valid MAC for R_1 and an invalid MAC for R_2.

[10] Note that our construction from standard primitives does make use of MAC schemes; however, it does so in a complex, non-black-box way.

Using Proofs of Knowledge A standard technique for making designated verifier signatures for a single verifier is to start from an interactive protocol that proves knowledge of either the signer's or the verifier's secret key, and turn this into a signature scheme using the Fiat-Shamir paradigm. It may seem natural to try to build an MDVS from this. However, it turns out to be challenging to achieve strong unforgeability using this technique; a signature cannot consist of a proof of knowledge of the signer's or one of the verifiers' secret keys, since any verifier will be able to convince other verifiers to accept a signature that did not come from the signer. For the same reason, a signature cannot consist of a proof of knowledge of the signer's secret key or some subset of the verifiers' secret keys.

MDVS from Standard Primitives Our first class of MDVS constructions is based only on standard primitives. With one exception specified below, all of these constructions can be instantiated in the random oracle model with no trusted setup. (Without random oracles, we would need to set up a common reference string.)

The idea is that the signer creates a DVS signature for each verifier individually, and then proves the consistency of those signatures.[11]. To support such proofs, we define a new primitive called Publicly Simulatable Designated Verifier Signatures (PSDVS) in Section 3.1, which is a single-verifier DVS equipped with extra properties. We then show, in Section 3.2, that a PSDVS together with a non-interactive zero knowledge proof of knowledge (NIZK-PoK) imply an MDVS for any number of signers and verifiers. Finally, we give some constructions of PSDVS. Our first PSDVS construction (in Section 3.3) uses only generic tools, namely psedudorandom functions, non-interactive key exchange (such as Diffie-Hellman), and non-interactive zero-knowledge proofs of knowledge. Our second PSDVS construction (in Section 3.4) aims at better concrete efficiency. It is based on DDH, strong RSA and Paillier encryption, is secure in the random oracle model, and requires a constant number of exponentiations for all operations. This scheme requires the trusted generation of an RSA modulus so that the factorization remains unknown. We also sketch a variant that requires no trusted setup, is secure in the random oracle model, and only requires (a variant of) the DDH assumption. However, this version requires double discrete log proofs, and therefore requires a non-constant number of exponentiations.

In order to support one of our constructions in which the signer sends an encrypted MAC key, we introduce a new tool we call Authenticated and Verifiable Encryption (AVPKE), which may be of independent interest. This is a variant of Paillier encryption with built-in authentication, and as such it is related to the known primitive "signcryption" [Zhe97]. However, our AVPKE scheme has the additional property that we can give efficient zero-knowledge proofs involving the encrypted message, using the algebraic properties of Paillier encryption.

To sign in our PSDVS schemes, the signer and verifiers first must establish a shared symmetric key k. In some cases they can do this non-interactively,

[11] Simply proving that all of the signatures verify would violate the off-the-record property; instead, the signer proves that either all of the signatures are real, or they are all simulated, as described in Section 3

using their secret and public keys, while in other cases the signer must send an encrypted key alongside the signature. After this, the signer sends a MAC on the message under key k; this MAC is based on a pseudorandom function.

MDVS from Functional Encryption. Our last construction is based on Verifiable Functional Encryption (VFE). It has the advantages of additionally meeting the privacy of identities and verifier-identity-based signing properties. Additionally, it can be set up to have smaller signatures if we are willing to make a stronger assumption on the number of colluding verifiers. Namely, the signature size is $O(t)$, where t is the size of the largest number of colluding verifiers we want to tolerate. The downsides are that, with current state of the art, VFE requires non-standard computational assumptions and a trusted setup.

Remark 4. If we are going to put a bound on the size of a collusion, it may seem we can use bounded collusion FE, which can be realized from standard assumptions [GVW12,AV19], and then there is no need for our other constructions from standard primitives. However, this is not true. Bounded collusion FE requires us to fix the bound on collusion size at key generation time; a bound that may later turn out to be too small. Additionally, ciphertext sizes in bounded collusion FE depend on the bound; thus, choosing a large bound to make sure we can handle the application implies a cost in efficiency. The MDVS signature sizes would depend on some upper bound on number of corrupt parties in the *system*, as opposed to on the number of recipients for the signature in question, which may be orders of magnitude smaller.

In a nutshell, the idea behind the functional encryption based construction is to do the proof of knowledge of one of the relevant secret keys "inside the ciphertext". In a little more detail, the idea is to encrypt a list of t *standard* signatures, where t is the maximal size of collusion we want to protect against (that is, $t \geq |\mathcal{C}|$), and the MDVS signature will simply be this ciphertext. To sign, the signer will generate their own standard signature σ_{S} on the message, and then encrypt a list a signatures consisting of σ_{S} followed by $t-1$ dummy values. To verify a signature, a verifier R gets a functional decryption key that will look at the list of signatures inside the ciphertext and output accept or reject. It will accept if the list contains a valid signature from S or a valid signature from R. Now, if a corrupt set of verifiers $\mathcal{C}$ wants to simulate a signature, they will all sign the message and encrypt the list of these signatures. By security of the encryption scheme, this looks like a real signature, and will indeed verify under all verification keys belonging to verifiers in $\mathcal{C}$. However, no honest verifier will accept it as a signature from S, so we have strong unforgeability.

2 Multi-Designated Verifier Signatures

MDVS Algorithms A *multi-designated verifier signature* (MDVS) scheme is defined by the following probabilistic polynomial-time algorithms:

$\mathsf{Setup}(1^\kappa) \to (pp, msk)$**:** On input the security parameter $\kappa \in \mathbb{N}$, outputs public parameters pp and the master secret key msk.

$\mathsf{SignKeyGen}(pp, msk) \to (spk, ssk)$**:** On input the public parameter pp and the master secret key msk, outputs the public key spk and secret key ssk for a signer.

$\mathsf{VerKeyGen}(pp, msk) \to (vpk, vsk)$**:** On input the public parameter pp and the master secret key msk, outputs the public key vpk and secret key vsk for a verifier.

$\mathsf{Sign}(pp, ssk_i, \{vpk_j\}_{j\in\mathcal{D}}, m) \to \sigma$**:** On input the public parameters pp, a secret signing key ssk_i, the public keys of the designated verifiers $\{vpk_j\}_{j\in\mathcal{D}}$, and a message m, outputs a signature σ.

$\mathsf{Verify}(pp, spk_i, vsk_j, \{vpk_{j'}\}_{j'\in\mathcal{D}}, m, \sigma) \to d$**:** On input the public parameters pp, a public verification key spk_i, a secret key vsk_j of a verifier such that $j \in \mathcal{D}$, the public keys of the designated verifiers $\{vpk_j\}_{j\in\mathcal{D}}$, a message m, and a signature σ, outputs a boolean decision d: $d = 1$ (accept) or $d = 0$ (reject).

$\mathsf{Sim}(pp, spk_i, \{vpk_j\}_{j\in\mathcal{D}}, \{vsk_j\}_{j\in\mathcal{C}}, m) \to \sigma'$**:** On input public parameters pp, a public verification key spk_i, the public keys of the designated verifiers $\{vpk_j\}_{j\in\mathcal{D}}$, the secret keys of the corrupt designated verifiers $\{vsk_j\}_{j\in\mathcal{C}}$, and a message m, outputs a simulated signature σ'.

The different algorithms take many different inputs, which are not all needed for all of our constructions. Thus, to simplify the notation we exclude these inputs in later sections whenever they are not needed.

MDVS Properties Let σ be a signature from signer i on message m and designated for verifiers $\mathcal{D}$. We ask for the following (informal) properties:

Correctness: All verifiers $j \in \mathcal{D}$ are able to verify an honestly generated signature σ.

Consistency: If there exists one verifier $j \in \mathcal{D}$ that accepts the signature σ, then all other designated verifiers (i.e. all $j' \in \mathcal{D} \setminus \{j\}$) also accept σ.

Unforgeability: An adversary without knowledge of the secret key ssk_i for signer i cannot create a signature σ' that is accepted by any designated verifier as a signature from signer i.

Off-The-Record: Given a signature σ, any malicious subset of the designated verifiers $\mathcal{C} \subseteq \mathcal{D}$ cannot convince any outsider that σ is a signature from signer i (i.e. the malicious set could have simulated the signature themselves).

(Optionally) Privacy of Identities: Any outsider (without colluding with any designated verifiers) cannot determine the identity of the signer and/or the identities of the designated verifiers.

(Optionally) Verifier-Identity-Based Signing: The signer should be able to produce a signature for a set of designated verifiers without requiring any information about them apart from their identities. In other words, we should have $vpk_j = j$ for a verifier with identity j.

Throughout our formal definitions we use the following six oracles:

Signer Key Generation Oracle: $\mathcal{O}_{SK}(i)$

1. If a signer key generation query has previously been performed for i, look up and return the previously generated key.
2. Otherwise, output and store $(spk_i, ssk_i) \leftarrow \mathsf{SignKeyGen}(pp, msk)$.

Verifier Key Generation Oracle: $\mathcal{O}_{VK}(j)$

1. If a verifier key generation query has previously been performed for j, look up and return the previously generated key.
2. Otherwise, output and store $(vpk_j, vsk_j) \leftarrow \mathsf{VerKeyGen}(pp, msk)$.

Public Signer Key Generation Oracle: $\mathcal{O}_{SPK}(i)$

1. $(spk_i, ssk_i) \leftarrow \mathcal{O}_{SK}(i)$.
2. Output spk_i.

Public Verifier Key Generation Oracle: $\mathcal{O}_{VPK}(j)$

1. $(vpk_j, vsk_j) \leftarrow \mathcal{O}_{VK}(j)$.
2. Output vpk_j.

Signing Oracle: $\mathcal{O}_S(i, \mathcal{D}, m)$

1. $(spk_i, ssk_i) \leftarrow \mathcal{O}_{SK}(i)$.
2. For all $j \in \mathcal{D}$: $vpk_j \leftarrow \mathcal{O}_{VPK}(j)$.
3. Output $\sigma \leftarrow \mathsf{Sign}(pp, ssk_i, \{vpk_j\}_{j\in\mathcal{D}}, m)$.

Verification Oracle: $\mathcal{O}_V(i, j, \mathcal{D}, m, \sigma)$

1. $spk_i \leftarrow \mathcal{O}_{SPK}(i)$.
2. $(vpk_j, vsk_j) \leftarrow \mathcal{O}_{VK}(j)$.
3. Output $d \leftarrow \mathsf{Verify}(pp, spk_i, vsk_j, \{vpk_{j'}\}_{j'\in\mathcal{D}}, m, \sigma)$.

Definition 1 (Correctness). *Let $\kappa \in \mathbb{N}$ be the security parameter, and let $\mathsf{MDVS} = (\mathsf{Setup}, \mathsf{SignKeyGen}, \mathsf{VerKeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Sim})$ be an MDVS scheme.* MDVS *is* correct *if for all signer identities i, messages m, verifier identity sets $\mathcal{D}$ and $j \in \mathcal{D}$, it holds that*

$$\Pr\left[\mathsf{Verify}(pp, spk_i, vsk_j, \{vpk_{j'}\}_{j'\in\mathcal{D}}, m, \sigma) \neq 1\right] = 0,$$

where the inputs to Verify *are generated as follows:*

- $(pp, msk) \leftarrow \mathsf{Setup}(1^\kappa)$;
- $(spk_i, ssk_i) \leftarrow \mathsf{SignKeyGen}(pp, msk, i)$;
- $(vpk_j, vsk_j) \leftarrow \mathsf{VerKeyGen}(pp, msk, j)$ *for* $j \in \mathcal{D}$;
- $\sigma \leftarrow \mathsf{Sign}(pp, ssk_i, \{vpk_j\}_{j\in\mathcal{D}}, m)$.

In Definition 1, we require that all the designated verifiers can verify the signature, without considering what happens for parties that are not designated verifiers (i.e. parties who should not be able to verify the signature). Parties that are not designated verifiers are accounted for by the off-the-record property.

Definition 2 (Consistency). *Let $\kappa \in \mathbb{N}$ be the security parameter, and let $\mathsf{MDVS} = (\mathsf{Setup}, \mathsf{SignKeyGen}, \mathsf{VerKeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Sim})$ be an MDVS scheme. Consider the following game between a challenger and an adversary $\mathcal{A}$:*

$\mathsf{Game}^{con}_{\mathsf{MDVS},\mathcal{A}}(\kappa)$

1. $(pp, msk) \leftarrow \mathsf{Setup}(1^\kappa)$
2. $(m^*, i^*, \mathcal{D}^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}_{SK}, \mathcal{O}_{VK}, \mathcal{O}_{SPK}, \mathcal{O}_{VPK}, \mathcal{O}_V}(pp)$

We say that $\mathcal{A}$ wins the game if there exist verifiers $j_0, j_1 \in \mathcal{D}^*$ such that:

$$\mathsf{Verify}(pp, spk_{i^*}, vsk_{j_0}, \{vpk_{j'}\}_{j'\in\mathcal{D}^*}, m^*, \sigma^*) = 0,$$
$$\mathsf{Verify}(pp, spk_{i^*}, vsk_{j_1}, \{vpk_{j'}\}_{j'\in\mathcal{D}^*}, m^*, \sigma^*) = 1,$$

where all keys are the honestly generated outputs of the key generation oracles, and $\mathcal{O}_{VK}$ is never queried on j_0 or j_1.
MDVS *is* consistent *if, for all PPT adversaries $\mathcal{A}$,*

$$\mathsf{adv}^{con}_{\mathsf{MDVS},\mathcal{A}}(\kappa) = \Pr\left[\mathcal{A} \textit{ wins } \mathsf{Game}^{con}_{\mathsf{MDVS},\mathcal{A}}(\kappa)\right] \leq \mathsf{negl}(\kappa).$$

Definition 2 states that even a valid signer (i.e. someone who knows a secret signing key) cannot create an inconsistent signature that will be accepted by some designated verifiers and rejected by others. By the correctness property, an honestly generated signature is accepted by all designated verifiers. By design, corrupt designated verifiers can construct an inconsistent signature, since some verifiers will accept it (i.e. those verifiers that created it), while the remaining honest designated verifiers will reject the simulated signature. Thus, we need to ask for $j \neq j_0, j_1$ for all queries j to the oracle $\mathcal{O}_{VK}$.

Definition 3 (Existential Unforgeability). *Let $\kappa \in \mathbb{N}$ be the security parameter, and let* $\mathsf{MDVS} = (\mathsf{Setup}, \mathsf{SignKeyGen}, \mathsf{VerKeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Sim})$ *be an MDVS scheme. Consider the following game between a challenger and an adversary $\mathcal{A}$:*

$\mathsf{Game}^{euf}_{\mathsf{MDVS},\mathcal{A}}(\kappa)$

1. $(pp, msk) \leftarrow \mathsf{Setup}(1^\kappa)$
2. $(m^*, i^*, \mathcal{D}^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}_{SK}, \mathcal{O}_{VK}, \mathcal{O}_{SPK}, \mathcal{O}_{VPK}, \mathcal{O}_S}(pp)$

We say that $\mathcal{A}$ wins the game if we have all of the following:

- *for all queries i to oracle $\mathcal{O}_{SK}$, it holds that $i^* \neq i$;*
- *for all queries $(i, \mathcal{D}, m)$ to oracle $\mathcal{O}_S$ that result in signature σ, it holds that $(i^*, \mathcal{D}^*, m^*) \neq (i, \mathcal{D}, m)$;*
- *there exists a verifier $j' \in \mathcal{D}^*$ such that for all queries j to oracle $\mathcal{O}_{VK}$, it holds that $j' \neq j$ and*
$$\mathsf{Verify}(pp, spk_{i^*}, vsk_{j'}, \{vpk_{j''}\}_{j'' \in \mathcal{D}^*}, m^*, \sigma^*) = 1,$$

where all keys are honestly generated outputs of the key generation oracles.
MDVS *is* existentially unforgeable *if, for all PPT adversaries $\mathcal{A}$,*

$$\mathsf{adv}^{euf}_{\mathsf{MDVS},\mathcal{A}}(\kappa) = \Pr\left[\mathcal{A} \textit{ wins } \mathsf{Game}^{euf}_{\mathsf{MDVS},\mathcal{A}}(\kappa)\right] \leq \mathsf{negl}(\kappa).$$

Definition 3 states that an adversary cannot create a signature that any honest verifier will accept as coming from a signer whose secret signing key the adversary does not know. The adversary will always get the public keys of the involved parties, i.e. signer with identity i^* and the designated verifiers $\mathcal{D}$, through the key generation oracles. He is also allowed to obtain the secret keys of every party except the signer i^* and at least one designated verifier. The reason why we need at least one honest verifier is that corrupt verifiers can create a simulated signature that will look like a real signature with respect to their own verifier secret keys. However, this simulation will be rejected by any honest designated verifier, i.e. the simulation will be a valid forgery for the corrupt verifiers, but not for the honest verifiers.

Definition 4 (Off-The-Record). *Let $\kappa \in \mathbb{N}$ be the security parameter, let* $\mathsf{MDVS} = (\mathsf{Setup}, \mathsf{SignKeyGen}, \mathsf{VerKeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Sim})$ *be an MDVS scheme, and let t be an upper bound on the number of verifiers an adversary $\mathcal{A}$ can corrupt. Consider the following game between a challenger and a stateful adversary $\mathcal{A}$, where all keys are honestly generated outputs of the key generation oracles:*

$\mathsf{Game}^{otr}_{\mathsf{MDVS},\mathsf{Sim},\mathcal{A}}(\kappa)$

1. $(pp, msk) \leftarrow \mathsf{Setup}(1^\kappa)$
2. $(i^*, \mathcal{D}^*, m^*, \mathcal{C}^*) \leftarrow \mathcal{A}^{\mathcal{O}_{SK}, \mathcal{O}_{VK}, \mathcal{O}_{SPK}, \mathcal{O}_{VPK}, \mathcal{O}_S, \mathcal{O}_V}(pp)$
3. $b \leftarrow \{0, 1\}$
4. $\sigma_0 \leftarrow \mathsf{Sign}(pp, ssk_{i^*}, \{vpk_j\}_{j \in \mathcal{D}^*}, m^*)$
5. $\sigma_1 \leftarrow \mathsf{Sim}(pp, spk_{i^*}, \{vpk_j\}_{j \in \mathcal{D}^*}, \{vsk_j\}_{j \in \mathcal{C}^*}, m^*)$
6. $b' \leftarrow \mathcal{A}^{\mathcal{O}_{SK}, \mathcal{O}_{VK}, \mathcal{O}_{SPK}, \mathcal{O}_{VPK}, \mathcal{O}_S, \mathcal{O}_V}(\sigma_b)$

We say that $\mathcal{A}$ wins the game if $b' = b$, and all of the following hold:

- $|\mathcal{C}^*| \leq t$ *and* $\mathcal{C}^* \subseteq \mathcal{D}^*$*;*
- *for all queries i to oracle $\mathcal{O}_{SK}$ it holds that $i^* \neq i$;*
- *for all queries j to oracle $\mathcal{O}_{VK}$ it holds that $j \notin \mathcal{D}^* \backslash \mathcal{C}^*$;*
- *for all queries $(i, j, \mathcal{D}, m, \sigma)$ to $\mathcal{O}_V$ it holds that $\sigma_b \neq \sigma$.*

We say that an MDVS scheme is t-off-the-record *if, for all PPT adversaries $\mathcal{A}$,*

$$\mathsf{adv}^{otr}_{\mathsf{MDVS},\mathsf{Sim},\mathcal{A}}(\kappa) = \Pr\left[\mathcal{A} \text{ wins } \mathsf{Game}^{otr}_{\mathsf{MDVS},\mathsf{Sim},\mathcal{A}}(\kappa)\right] - \frac{1}{2} \leq \mathsf{negl}(\kappa).$$

If a scheme supports $t = |\mathcal{D}|$, we say that it is off-the-record.

Definition 4 states that any adversary that corrupts a subset (of size t) of the designated verifiers $\mathcal{C}^*$ cannot determine whether the received signature was created by real signer i^* or simulated by the corrupt verifiers $\mathcal{C}^*$. The adversary is not allowed to see the secret keys for the designated verifiers that are in $\mathcal{D}^* \backslash \mathcal{C}^*$. If the adversary was allowed to get secret keys of additional parties in $\mathcal{D}^*$ (which are not in $\mathcal{C}^*$), then he would be able to distinguish trivially, since any honest designated verifiers (i.e. any $j \in \mathcal{D}^* \backslash \mathcal{C}^*$) can distinguish simulated signatures from real signatures (from the unforgeability property).

Definition 5 (Privacy of Identities). *Let $\kappa \in \mathbb{N}$ be the security parameter, and let* $\mathsf{MDVS} = (\mathsf{Setup}, \mathsf{SignKeyGen}, \mathsf{VerKeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Sim})$ *be an MDVS scheme. Consider the following game between a challenger and a stateful adversary $\mathcal{A}$, where all keys are the honestly generated outputs of the key generation oracles:*

$\mathsf{Game}^{pri}_{\mathsf{MDVS},\mathcal{A}}(\kappa)$

1. $(pp, msk) \leftarrow \mathsf{Setup}(1^\kappa)$
2. $(m^*, i_0, i_1, \mathcal{D}_0, \mathcal{D}_1) \leftarrow \mathcal{A}^{\mathcal{O}_{SK}, \mathcal{O}_{VK}, \mathcal{O}_{SPK}, \mathcal{O}_{VPK}, \mathcal{O}_S, \mathcal{O}_V}(pp)$
3. $b \leftarrow \{0, 1\}$
4. $\sigma^* \leftarrow \mathsf{Sign}(pp, ssk_{i_b}, \{vpk_j\}_{j \in \mathcal{D}_b}, m^*)$
5. $b' \leftarrow \mathcal{A}^{\mathcal{O}_{SK}, \mathcal{O}_{VK}, \mathcal{O}_{SPK}, \mathcal{O}_{VPK}, \mathcal{O}_S, \mathcal{O}_V}(\sigma^*)$

We say that $\mathcal{A}$ wins the game if $b = b'$, and all of the following hold:

– $|\mathcal{D}_0| = |\mathcal{D}_1|$;
– *for all queries* i *to* $\mathcal{O}_{SK}$*, it holds that* $i \notin \{i_0, i_1\}$;
– *for all queries* j *to* $\mathcal{O}_{VK}$*, it holds that* $j \notin \mathcal{D}_0 \cup \mathcal{D}_1$;
– *for all queries* $(i, j, \mathcal{D}, m, \sigma)$ *to* $\mathcal{O}_V$*, it holds that* $\sigma^* \neq \sigma$.

MDVS *has* privacy of identities *if, for all PPT adversaries* $\mathcal{A}$,

$$\mathsf{adv}^{pri}_{\mathsf{MDVS},\mathcal{A}}(\kappa) = \Pr\left[\mathcal{A} \textit{ wins } \mathsf{Game}^{pri}_{\mathsf{MDVS},\mathcal{A}}(\kappa)\right] - \frac{1}{2} \leq \mathsf{negl}(\kappa).$$

We say that MDVS *has additional properties as follows:*
– *privacy of the signer's identity (PSI) if we make the restriction that* $\mathcal{D}_0 = \mathcal{D}_1$;
– *privacy of the designated verifiers' identities (PVI) if we make the restriction that* $i_0 = i_1$.

Definition 5 states that an adversary cannot distinguish between signatures from two different signers (PSI) if he does not know the secret key of any of the signers or designated verifiers (as designated verifiers *are* allowed to identify the signer). Furthermore, it should not help him to see other signatures that he knows are from the signers in question.

In addition, if we vary the verifier sets ($\mathcal{D}_0 \neq \mathcal{D}_1$), then the MDVS scheme has privacy of designated verifier's identities (PVI), which means that any outsider without knowledge of any secret keys cannot distinguish between signatures meant for different verifiers.

Definition 6 (Verifier-Identity-Based Signing). *We say that an MDVS scheme has* verifier-identity-based signing *if for honestly generated verifier keys* (vsk_j, vpk_j) *for verifier with identity* j*, we have* $vpk_j = j$.

Note that, in order to achieve verifier-identity-based signing, verifier key generation must require a master secret key msk. Otherwise, any outsider would be able to generate a verification key for verifier j, and use it to verify signatures meant only for that verifier.

Relation to Previous Definitions Our definition of MDVS is consistent with previous work in this area, but with some differences. Our MDVS syntax closely follows the one introduced by [LV04], but we allow for a master secret key in the case where the keys are generated by a trusted party (like in our construction based on functional encryption). Our security definitions are adapted from those in [LV04,ZAYS12] to capture the flexibility introduced by allowing any subset of designated verifiers to simulate a signature, thus providing better deniability properties. Finally, we formalize consistency as an additional and desirable requirement.

3 Standard Primitive-Based MDVS Constructions

In this section we show how to create an MDVS scheme that uses only standard primitives, such as key exchange, commitments, pseudorandom functions and generators, and non-interactive zero knowledge proofs.

On a high level, one way to build an MDVS is for the signer to use a separate DVS with each verifier; the MDVS signature would then consist of a vector

of individual DVS signatures. This gives us almost everything we need — the remaining issue is consistency. Each verifier can verify one of the DVS signatures, but is not convinced that all of the other verifiers will come to the same conclusion.

A solution to this consistency issue is to include as part of the MDVS signature a zero knowledge proof that all of the DVS signatures verify. However, this introduces a new issue with off-the-record. Now, a colluding set of verifiers will not be able to simulate a signature unless *all* of the verifiers collude. In order to produce such a convincing zero knowledge proof as part of the signature, they would need to forge signatures for the other verifiers in the underlying DVS scheme, which they should not be able to do.

So, instead of using a zero knowledge proof of knowledge that all of the DVS signatures verify, we use a proof that *either* all of the DVS signatures verify, *or they are all simulated*. Then, a corrupt set of verifiers can simulate all of the underlying DVS signatures — with the caveat that the signatures they simulate for themselves should be convincing simulations even in the presence of their secret keys — and, instead of proving that all of the signatures verify, they prove that all of the signatures are simulations.

3.1 New Primitive: Provably Simulatable Designated-Verifier Signatures (PSDVS)

Designated Verifier Signatures (DVS) have a simulation algorithm Sim which is used to satisfy the off-the-record property. Given the signer's public key, the verifier's secret key and a message m, Sim should return a signature which is indistinguishable from a real signature. A *Provably Simulatable DVS (PSDVS)* has some additional properties:

Definition 7. *A PSDVS must satisfy the standard notions of correctness and existential unforgeability. Additionally, it should satisfy* PubSigSim indstinguishability *(Definition 8),* PubSigSim correctness *(Definition 9),* PubSigSim soundness *(Definition 10),* VerSigSim indstinguishability *(Definition 11),* VerSigSim correctness *(Definition 12),* VerSigSim soundness *(Definition 13),* provable signing correctness *(Definition 14), and* provable signing soundness *(Definition 15).*

Provable Public Simulation As in *PSI* (Definition 5), anyone should be able to produce a signature that is indistinguishable from a real signature. Additionally, the party simulating the signature should be able to produce a proof that this is *not* a real signature. This proof will be incorporated into the MDVS proof of consistency; the colluding verifiers, when producing a simulation, need to prove that all underlying PSDVS signatures are real, or that they are all fake.

In other words, we require two additional algorithms, as follows:

1. $\mathsf{PubSigSim}(pp, spk, vpk, m) \to (\sigma, \pi)$
2. $\mathsf{PubSigVal}(pp, spk, vpk, m, \sigma, \pi) \to d \in \{0, 1\}$

The colluding verifiers will produce a public simulation in the underlying PSDVS for verifiers outside their coalition, and use $\mathsf{PubSigSim}$ to prove that this simulation is not a real signature. π will not be explicitly included in the proof

of "the underlying PSDVS signatures are all real or all fake," of course, as it would give away the fact that all underlying signatures are fake, as opposed to all being real; rather, it will be wrapped in a larger zero knowledge proof.

Definition 8 (PubSigSim Indistinguishability). *We say that the PSDVS has* PubSigSim Indistinguishability *if* PubSigSim *produces a signature σ that is indistinguishable from real. More formally, an adversary should not be able to win the following game with probability non-negligibly more than half:*

$\mathsf{Game}^{PubSigSim\text{-}Ind}_{\mathsf{PVDVS},\mathcal{A}}(\kappa)$

1. $pp \leftarrow \mathsf{Setup}(1^\kappa)$
2. $(spk, ssk) \leftarrow \mathsf{SignKeyGen}(pp)$
3. $(vpk, vsk) \leftarrow \mathsf{VerKeyGen}(pp)$
4. $m^* \leftarrow \mathcal{A}^{\mathcal{O}_S, \mathcal{O}_V}(spk, vpk)$
5. $b \leftarrow \{0, 1\}$
6. $\sigma_0 \leftarrow \mathsf{Sign}(pp, ssk, vpk, m^*)$
7. $(\sigma_1, \pi) \leftarrow \mathsf{PubSigSim}(pp, spk, vpk, m^*)$
8. $b' \leftarrow \mathcal{A}^{\mathcal{O}_S, \mathcal{O}_V}(pp, spk, vpk, m^*, \sigma_b)$

We say that $\mathcal{A}$ wins the PubSigSim-Ind *game if $b = b'$ and for all queries (m, σ) to $\mathcal{O}_V$, it holds that $(m, \sigma) \neq (m^*, \sigma_b)$.*

Definition 9 (PubSigSim Correctness). *We say that the PSDVS has* PubSigSim Correctness *if for all $pp \leftarrow \mathsf{Setup}(1^\kappa)$; $(spk, ssk) \leftarrow \mathsf{SignKeyGen}(pp)$; $(vpk, vsk) \leftarrow \mathsf{VerKeyGen}(pp)$; $m \in \{0,1\}^*$; $(\sigma, \pi) \leftarrow \mathsf{PubSigSim}(pp, spk, vpk, m)$;*

$$\Pr[\mathsf{PubSigVal}(pp, spk, vpk, m, \sigma, \pi) = 1] = 1.$$

Definition 10 (PubSigSim Soundness). *We say that the PSDVS has* PubSigSim Soundness *if it is hard to construct a signature σ which is accepted by the verifier algorithm and at the same time can be proven to be a simulated signature. More formally, an adversary should not be able to win the following game with non-negligible probability:*

$\mathsf{Game}^{PubSigSim\text{-}Sound}_{\mathsf{PVDVS},\mathcal{A}}(\kappa)$

1. $pp \leftarrow \mathsf{Setup}(1^\kappa)$
2. $(spk, ssk) \leftarrow \mathsf{SignKeyGen}(pp)$
3. $(vpk, vsk) \leftarrow \mathsf{VerKeyGen}(pp)$
4. $(m^*, \sigma^*, \pi^*) \leftarrow \mathcal{A}(pp, ssk, spk, vpk)$

We say that $\mathcal{A}$ wins the PubSigSim-Sound *game if* $\mathsf{Verify}(pp, vsk, m^*, \sigma^*) = 1$ *and* $\mathsf{PubSigVal}(pp, spk, vpk, m^*, \sigma^*, \pi^*) = 1$.

Provable Verifier Simulation As in *off-the-record* (Definition 4), a verifier should be able to produce a signature that is indistinguishable from a real signature, *even given its secret key.* Additionally, the verifier should be able to produce a proof that the signature is *not* a real signature (that is, that the verifier, and not the signer, produced it). This proof will be incorporated into the MDVS proof of consistency.

In other words, we require two additional algorithms, as follows:

1. $\mathsf{VerSigSim}(pp, spk, vpk, vsk, m) \to (\sigma, \pi)$
2. $\mathsf{VerSigVal}(pp, spk, vpk, m, \sigma, \pi) \to d \in \{0, 1\}$

The colluding verifiers will produce a verifier simulation in the underlying PSDVS for verifiers *inside* their coalition, and use $\mathsf{VerSigSim}$ to prove that this simulation is not a real signature.

Definition 11 (VerSigSim Indistinguishability). *We say that the PSDVS has* VerSigSim Indistinguishability *if* $\mathsf{VerSigSim}$ *produces a signature* σ *that is indistinguishable from real. More formally, an adversary should not be able to win the following game with probability non-negligibly more than half:*

$\mathsf{Game}^{VerSigSim\text{-}Ind}_{\mathsf{PVDVS},\mathcal{A}}(\kappa)$

1. $pp \leftarrow \mathsf{Setup}(1^\kappa)$
2. $(spk, ssk) \leftarrow \mathsf{SignKeyGen}(pp)$
3. $(vpk, vsk) \leftarrow \mathsf{VerKeyGen}(pp)$
4. $m^* \leftarrow \mathcal{A}^{\mathcal{O}_S}(pp, spk, vpk, vsk)$
5. $b \leftarrow_\$ \{0, 1\}$
6. $\sigma_0 \leftarrow \mathsf{Sign}(pp, ssk, vpk, m^*)$
7. $(\sigma_1, \pi) \leftarrow \mathsf{VerSigSim}(pp, spk, vsk, m^*)$
8. $b' \leftarrow \mathcal{A}^{\mathcal{O}_S}(pp, spk, vpk, vsk, m^*, \sigma_b)$

We say that $\mathcal{A}$ *wins the* VerSigSim-Ind *game if* $b = b'$.

Definition 12 (VerSigSim Correctness). *We say that the PSDVS has* VerSigSim Correctness *if for all* $pp \leftarrow \mathsf{Setup}(1^\kappa)$, $(spk, ssk) \leftarrow \mathsf{SignKeyGen}(pp)$, $(vpk, vsk) \leftarrow \mathsf{VerKeyGen}(pp)$, $m \in \{0, 1\}^*$, $(\sigma, \pi) \leftarrow \mathsf{VerSigSim}(pp, spk, vpk, vsk, m)$,

$$\Pr[\mathsf{VerSigVal}(pp, spk, vpk, m, \sigma, \pi) = 1] = 1.$$

Definition 13 (VerSigSim Soundness). *We say that the PSDVS has* VerSigSim Soundness *if the signer is not able to produce* σ *and* π *that pass the validation check* $\mathsf{VerSigVal}$*, i.e.* π *is a proof that* σ *was not produced by the signer. More formally, an adversary should not be able to win the following game with non-negligible probability:*

$\mathsf{Game}^{VerSigSim\text{-}Sound}_{\mathsf{PVDVS},\mathcal{A}}(\kappa)$

1. $pp \leftarrow \mathsf{Setup}(1^\kappa)$
2. $(spk, ssk) \leftarrow \mathsf{SignKeyGen}(pp)$
3. $(vpk, vsk) \leftarrow \mathsf{VerKeyGen}(pp)$
4. $(m^*, \sigma^*, \pi^*) \leftarrow \mathcal{A}(pp, ssk, spk, vpk)$

$\mathcal{A}$ *wins the* VerSigSim-Sound *game if* $\mathsf{VerSigVal}(pp, spk, vpk, m^*, \sigma^*, \pi^*) = 1$.

Provable Signing Lastly, we require a provable variant of signing, so that the signer is able to produce a proof that a signature is real. In other words, we require the signing algorithm $\mathsf{Sign}(pp, spk, ssk, vpk, m) \to (\sigma, \pi)$ to output π as well. We also require one additional validation algorithm, as follows:

$$\mathsf{RealSigVal}(pp, spk, vpk, m, \sigma, \pi) \to d \in \{0, 1\}$$

Definition 14 (Provable Signing Correctness). *We say that the PSDVS has* Provable Signing Correctness *if* $\forall pp \leftarrow \mathsf{Setup}(1^\kappa)$, $(spk, ssk) \leftarrow \mathsf{SignKeyGen}(pp)$, $(vpk, vsk) \leftarrow \mathsf{VerKeyGen}(pp)$, $m \in \{0,1\}^*$, $(\sigma, \pi) \leftarrow \mathsf{Sign}(pp, spk, ssk, vpk, m)$,

$$\Pr[\mathsf{RealSigVal}(pp, spk, vpk, m, \sigma, \pi) = 1] = 1.$$

Definition 15 (Provable Signing Soundness). *We say that the PSDVS has* Provable Signing Soundness *if the proof of correctness* π *produced by* Sign *does not verify unless* σ *verifies. More formally, an adversary should not be able to win the following game with non-negligible probability:*

$\mathsf{Game}^{Sign\text{-}Sound}_{\mathsf{PVDVS},\mathcal{A}}(\kappa)$

1. $pp \leftarrow \mathsf{Setup}(1^\kappa)$
2. $(spk, ssk) \leftarrow \mathsf{SignKeyGen}(pp)$
3. $(vpk, vsk) \leftarrow \mathsf{VerKeyGen}(pp)$
4. $(m^*, \sigma^*, \pi^*) \leftarrow \mathcal{A}(pp, ssk, spk, vpk)$

We say that $\mathcal{A}$ *wins the* Sign-Sound *game if* $\mathsf{RealSigVal}(pp, spk, vpk, m^*, \sigma^*, \pi^*) = 1$ *and* $\mathsf{Verify}(pp, spk, vsk, m^*, \sigma^*) = 0$.

Note that none of these proofs π are parts of the signature. If included in the signature, such proofs would allow an adversary to distinguish a simulation from a real signature.

3.2 Standard Primitive-Based MDVS Construction

Given a PSDVS, as defined in Section 3.1, we can build an MDVS. The transformation is straightforward: the signer uses the PSDVS to sign a message for each verifier, and proves consistency using a non-interactive zero knowledge proof of knowledge. The proof of consistency will claim that either all of the PSDVS signatures verify, or all of them are simulated.

Construction 1 *Let* $\mathsf{PSDVS} = (\mathsf{Setup}, \mathsf{SignKeyGen}, \mathsf{VerKeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{RealSigVal}, \mathsf{PubSigSim}, \mathsf{PubSigVal}, \mathsf{VerSigSim}, \mathsf{VerSigVal})$ *be a provably simulatable designated verifier signature scheme, and* $\mathsf{NIZK\text{-}PoK} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ *be a non-interactive zero knowledge proof of knowledge system and* $\mathcal{R}_{cons}$ *a relation that we will define later in the protocol.*

$\mathsf{Setup}(1^\kappa)$**:**
1. $crs \leftarrow \mathsf{NIZK\text{-}PoK.Setup}(1^\kappa, \mathcal{R}_{cons})$.
2. $\mathsf{PSDVS}.pp \leftarrow \mathsf{PSDVS.Setup}(1^\kappa)$.

Output $(crs, \mathsf{PSDVS}.pp)$ as the public parameters pp.

$\mathsf{SignKeyGen}(pp)$**:** $(spk_i, ssk_i) \leftarrow \mathsf{PSDVS.SignKeyGen}(\mathsf{PSDVS}.pp)$.

Output (spk_i, ssk_i) as signer i's public/secret key pair.

$\mathsf{VerKeyGen}(pp)$**:** $(vpk_j, vsk_j) \leftarrow \mathsf{PSDVS.VerKeyGen}(\mathsf{PSDVS}.pp)$.

Output (vpk_j, vsk_j) as verifier j's public/secret key pair.

$\mathsf{Sign}(pp, ssk_i, \{vpk_j\}_{j \in \mathcal{D}}, m)$**:**
1. For every verifier $j \in \mathcal{D}$, compute a signature and proof of signature validity as $(\sigma_j, \pi_j) \leftarrow \mathsf{PSDVS.Sign}(\mathsf{PSDVS}.pp, ssk_i, vpk_j, m)$.

2. Create a proof π of consistency, i.e a proof of knowledge of $\{\pi_j\}_{j\in\mathcal{D}}$ such that either all signatures are real (as demonstrated by $\{\pi_j\}_{j\in\mathcal{D}}$), or all signatures are fake (as could be demonstrated by the proofs produced by PSDVS.PubSigSim or PSDVS.VerSigSim).
3. $\sigma = (\{\sigma_j\}_{j\in\mathcal{D}}, \pi)$.

Output σ as the signature.

Verify$(pp, spk_i, vsk_j, m, \sigma = (\{\sigma_j\}_{j\in\mathcal{D}}, \pi))$:

1. Let $d_\pi \leftarrow$ NIZK-PoK.Verify$(crs, u = (\text{PSDVS}.pp, spk_i, \{vpk_j\}_{j\in\mathcal{D}}, \{\sigma_j\}_{j\in\mathcal{D}}), \pi)$.
2. Let $d \leftarrow$ PSDVS.Verify$(\text{PSDVS}.pp, spk_i, vsk_j, m, \sigma_j) \wedge d_\pi$.

Output d as the verification decision.

Sim$(pp, spk_i, \{vpk_j\}_{j\in\mathcal{D}}, \{vsk_j\}_{j\in\mathcal{C}}, m)$:

1. For $j \in \mathcal{D} \cap \mathcal{C}$: $(\sigma_j, \pi_j) \leftarrow$ VerSigSim$(\text{PSDVS}.pp, spk_i, vpk_j, vsk_j, m)$.
2. For $j \in \mathcal{D}\backslash\mathcal{C}$: $(\sigma_j, \pi_j) \leftarrow$ PubSigSim$(\text{PSDVS}.pp, spk_i, vpk_j, m)$.
3. Use these signatures and proofs to produce the NIZK π for consistency.
4. $\sigma = (\{\sigma_j\}_{j\in\mathcal{D}}, \pi)$.

Output σ as the signature.

Theorem 2. *Assume* PSDVS *is a secure provably simulatable designated verifier signature scheme and* NIZK-PoK *is a secure non-interactive zero knowledge proof of knowledge system. Then Construction 1 is a correct and secure MDVS scheme (without privacy of identities (Definition 5)).*

Due to space limitations, the proof of Theorem 2 is deferred to the full version.

3.3 Standard Primitive-Based PSDVS Construction

We can build a PSDVS from a special message authentication code (MAC) which looks uniformly random without knowledge of the secret MAC key — such a MAC can be built from any pseudorandom function. A signature on a message m will be a MAC on (m, t), where t is some random tag. Proving that the signature is real simply involves proving knowledge of a MAC key that is consistent with the MAC and some global public commitment to the MAC key. A public proof that the signature is simulated and does not verify would involve proving that the MAC was pseudorandomly generated. A verifier's proof that the signature is simulated would involve proving that the tag was generated in a way that only the verifier could use (e.g. from a PRF to which only the verifier knows the key).

Of course, this is not ideal, since MACs require knowledge of a shared key; in order to use MACs, we would need to set up shared keys between every possible pair of signer and verifier. However, we can get around this using non-interactive key exchange (NIKE). Each signer and verifier publishes a public key, and any pair of them can agree on a shared secret key by simply using their own secret key and the other's public key. The construction is as follows:

Construction 2 *Let:*

- COMM = (Setup, Commit, Open) *be a commitment scheme,*
- PRF = (KeyGen, Compute) *be a length-preserving pseudorandom function,*
- PRG *be a length-doubling pseudorandom generator,*

– $\mathsf{NIZK} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ *be a non-interactive zero knowledge proof system, and*
– $\mathsf{NIKE} = (\mathsf{KeyGen}, \mathsf{KeyExtract}, \mathsf{KeyMatch})$ *be a non-interactive key exchange protocol.* $\mathsf{KeyMatch}$ *is an additional algorithm that checks if a public key and a secret key match.* $\mathsf{KeyMatch}$ *is not typically defined as a part of a* NIKE *scheme; however, such an algorithm always exists.*

$\mathsf{Setup}(1^\kappa)$**:**

1. $crs_i \leftarrow \mathsf{NIZK.Setup}(1^\kappa, \mathcal{R}_i), i = 1, 2, 3.$
2. $ck \leftarrow \mathsf{COMM.Setup}(1^\kappa)$.

Output $(\{crs_1, crs_2, crs_3\}, ck)$ as the public parameters pp.

$\mathsf{SignKeyGen}(pp)$**:**

1. $(\mathsf{NIKE}.pk_\mathsf{S}, \mathsf{NIKE}.sk_\mathsf{S}) \leftarrow \mathsf{NIKE.KeyGen}(1^\kappa)$.
2. $ssk = \mathsf{NIKE}.sk_\mathsf{S}$.
3. $spk = \mathsf{NIKE}.pk_\mathsf{S}$.

Output ssk as the signer's secret key and spk as the signer's public key.

$\mathsf{VerKeyGen}(pp)$**:**

1. $(\mathsf{NIKE}.pk_\mathsf{R}, \mathsf{NIKE}.sk_\mathsf{R}) \leftarrow \mathsf{NIKE.KeyGen}(1^\kappa)$.
2. $k_\mathsf{R} \leftarrow \mathsf{PRF.KeyGen}(1^\kappa)$. (Informally, this key will be used by the verifier to simulate signatures using $\mathsf{VerSigSim}$.)
3. Choose randomness (i.e. decommitment value) r_R at random.
4. $c_\mathsf{R} = \mathsf{COMM.Commit}(ck, k_\mathsf{R}; r_\mathsf{R})$. (Informally, this commitment will be used by the verifier to support its proofs of fake-ness.)
5. $vsk = (\mathsf{NIKE}.sk_\mathsf{R}, k_\mathsf{R}, r_\mathsf{R})$.
6. $vpk = (\mathsf{NIKE}.pk_\mathsf{R}, c_\mathsf{R})$.

Output vsk as the verifier's secret key and vpk as the verifier's public key.

$\mathsf{Sign}(pp, ssk = \mathsf{NIKE}.sk_\mathsf{S}, vpk = (\mathsf{NIKE}.pk_\mathsf{R}, c_\mathsf{R}), m)$**:**

The signer computes a shared key with the designated verifier and proceeds to sign the message m:

1. $k_{shared} = \mathsf{NIKE.KeyExtract}(\mathsf{NIKE}.sk_\mathsf{S}, \mathsf{NIKE}.pk_\mathsf{R})$. (Informally, this key will be used as a MAC key.)
2. Choose t at random.
3. $\sigma = (\sigma_1, \sigma_2) \leftarrow (t, \mathsf{PRF}_{k_{shared}}((m, t)))$.
4. $\pi \leftarrow \mathsf{NIZK.Prove}(crs_1, u, w)$ where $u = ((\sigma_1, \sigma_2), \mathsf{NIKE}.pk_\mathsf{S}, \mathsf{NIKE}.pk_\mathsf{R}, m)$ and $w = (\mathsf{NIKE}.sk_\mathsf{S}, k_{shared}))$
   We define the relation $\mathcal{R}_1$ indexed by NIKE public parameters and PRF for a statement u and witness w:

$$\begin{aligned}\mathcal{R}_1 = \{(u = (\sigma_1, \sigma_2, \mathsf{NIKE}.pk_\mathsf{S}, \mathsf{NIKE}.pk_\mathsf{R}, m), w = (\mathsf{NIKE}.sk_\mathsf{S}, k_{shared})) :\\ \mathsf{KeyMatch}(\mathsf{NIKE}.pk_\mathsf{S}, \mathsf{NIKE}.sk_\mathsf{S}) = 1\\ \wedge\ k_{shared} = \mathsf{NIKE.KeyExtract}(\mathsf{NIKE}.sk_\mathsf{S}, \mathsf{NIKE}.pk_\mathsf{R})\\ \wedge\ \sigma_2 = \mathsf{PRF}_{k_{shared}}((m, \sigma_1))\}\end{aligned}$$

Output σ as the signature, and π as the proof of real-ness.

$\mathsf{Verify}(pp, spk = \mathsf{NIKE}.pk_\mathsf{S}, vsk = (\mathsf{NIKE}.sk_\mathsf{R}, k_\mathsf{R}, r_\mathsf{R}), m, \sigma = (\sigma_1, \sigma_2))$**:**

1. $k_{shared} = \mathsf{NIKE.KeyExtract}(\mathsf{NIKE}.sk_\mathsf{R}, \mathsf{NIKE}.pk_\mathsf{S})$. (Informally, this key will be used as a MAC key.)
2. If $\mathsf{PRF}_{k_{shared}}((m, \sigma_1)) = \sigma_2$, set $d = 1$. Otherwise, set $d = 0$.

Output d as the verification decision.

$\mathsf{RealSigVal}(pp, spk, vpk, m, \sigma, \pi)$**:**

Output $d \leftarrow \mathsf{NIZK.Verify}(crs_1, \sigma, \pi)$ as the validation decision.

$\mathsf{PubSigSim}(pp, m)$**:**

1. Choose a PRG seed seed.
2. Choose σ_1 and σ_2 pseudorandomly by running PRG on seed.
3. $\sigma \leftarrow (\sigma_1, \sigma_2)$.
4. Let $\pi \leftarrow \mathsf{NIZK.Prove}(crs_2, u = \sigma, w = \mathsf{seed})$.
   We define the relation $\tilde{\mathcal{R}}_2$ indexed by the PRG for a statement $u = (\sigma = (\sigma_1, \sigma_2))$ and the witnesses $w = \mathsf{seed}$:

$$\tilde{\mathcal{R}}_2 = \{(u = \sigma; w = \mathsf{seed}) : u = \mathsf{PRG}(w)\} \tag{1}$$

Output σ as the simulated signature, and π as the proof of fake-ness.

$\mathsf{PubSigVal}(pp, spk, vpk, m, \sigma = (\sigma_1, \sigma_2), \pi)$**:**

Output $d \leftarrow \mathsf{NIZK.Verify}(crs_2, \sigma, \pi)$ as the validation decision.

$\mathsf{VerSigSim}(pp, spk = \mathsf{NIKE}.pk_{\mathsf{S}}, vpk = (\mathsf{NIKE}.pk_{\mathsf{R}}, c_{\mathsf{R}}), vsk = (\mathsf{NIKE}.sk_{\mathsf{R}}, k_{\mathsf{R}}, r_{\mathsf{R}}), m)$**:**

The verifier can fake a signature using its PRF key k_{R}.

1. $k_{shared} = \mathsf{NIKE.KeyExtract}(\mathsf{NIKE}.sk_{\mathsf{R}}, \mathsf{NIKE}.pk_{\mathsf{S}})$.
2. Choose r at random.
3. $t \leftarrow \mathsf{PRF}_{k_{\mathsf{R}}}(r)$.
4. $\sigma \leftarrow (t, \mathsf{PRF}_{k_{shared}}((m, t)))$.
5. Let $\pi \leftarrow \mathsf{NIZK.Prove}(crs_3, u = (c_{\mathsf{R}}, \sigma_1), w = (k_{\mathsf{R}}, r_{\mathsf{R}}, r))$.
   We define the relation $\tilde{\mathcal{R}}_3$ indexed by the c public parameters and PRF for statements u and witnesses w:

$\tilde{\mathcal{R}}_3 = \{(u = (c_{\mathsf{R}}, \sigma_1), w = (k_{\mathsf{R}}, r_{\mathsf{R}}, r)) : k_{\mathsf{R}} = \mathsf{COMM.Open}(c_{\mathsf{R}}, r_{\mathsf{R}}) \wedge \sigma_1 = \mathsf{PRF}_{k_{\mathsf{R}}}(r)\}$

Output σ as the simulated signature and π as the proof of fake-ness.

$\mathsf{VerSigVal}(pp, spk, vpk, m, \sigma, \pi)$**:**

Output $d \leftarrow \mathsf{NIZK.Verify}(crs_3, (c_{\mathsf{R}}, \sigma_1), \pi)$ as the validation decision.

Theorem 3. *If the schemes* $\mathsf{COMM}, \mathsf{PRF}, \mathsf{PRG}, \mathsf{NIZK}, \mathsf{NIKE}$ *are secure, then Construction 2 is a correct and secure PSDVS scheme as per Definition 7.*

Due to space limitations, the proof of Theorem 3 is deferred to the full version.

3.4 DDH and Paillier-Based PSDVS Construction

The goal of this section is to construct a PSDVS scheme based on DDH and the security of Paillier encryption. The idea in the PSDVS construction is that the authenticator for a message m will be $\mathsf{H}(m, t)^k$ in a group G where t is a nonce, k is a key known to both parties and H is a hash function modeled as a random oracle. The construction requires that certain properties of the key can be proved in zero-knowledge, and we can do this efficiently using standard Σ-protocols because the key is in the exponent. However, naive use of this idea would mean that a sender needs to store a key for every verifier he talks to, and the set-up must generate correlated secret keys for the parties. To get around this, we will instead let the sender choose k on the fly and send it to the verifier, encrypted using a new variant of Paillier encryption. In the following subsection we describe and prove this new encryption scheme, and then we specify the actual PSDVS construction. Paillier-style encryption comes in handy since its algebraic properties are useful in making our zero knowledge proofs efficient.

Paillier-based Authenticated and Verifiable Encryption An authenticated and verifiable encryption scheme (AVPKE) involves a sender S and a receiver R. Such a scheme comes with the following polynomial time algorithms:

$\mathsf{Setup}(1^\kappa) \to pp$**:** A probabilistic algorithm for setup which outputs public parameters.
$\mathsf{KeyGen}_\mathsf{S}(pp) \to (sk_\mathsf{S}, pk_\mathsf{S})$**:** A probabilistic sender key generation algorithm.
$\mathsf{KeyGen}_\mathsf{R}(pp) \to (sk_\mathsf{R}, pk_\mathsf{R})$**:** A probabilistic receiver key generation algorithm.
$\mathsf{Enc}_{pp,sk_\mathsf{S},pk_\mathsf{R}}(k) \to c$**:** A probabilistic encryption algorithm for message k.
$\mathsf{Dec}_{pp,sk_\mathsf{R},pk_\mathsf{S}}(c) \to \{k, \perp\}$**:** A decryption algorithm that outputs either reject or a message.

We require, of course, that $\mathsf{Dec}_{pp,sk_\mathsf{R},pk_\mathsf{S}}(\mathsf{Enc}_{pp,sk_\mathsf{S},pk_\mathsf{R}}(k)) = k$ for all messages k.

Intuitively, the idea is that given only the receiver public key pk_R and his own secret key sk_S, the sender S can encrypt a message k in such a way that on receiving the ciphertext, R can check that k comes from S, no third party knows k and finally, the encryption is verifiable in that it allows S to efficiently prove in zero-knowledge that k satisfies certain properties.

Our AVPKE scheme adds an authentication mechanism on top of Paillier encryption:

Construction 3 *Let:*

- Ggen *be a* Group Generator*, a probabilistic polynomial time algorithm which on input* 1^κ *outputs the description of a cyclic group* G *and a generator* g*, such that the order of* G *is a random* κ*-bit RSA modulus* n*, which is the product of so-called safe primes. (That is,* $n = pq$ *where* $p = 2p' + 1, q = 2q' + 1$ *and* p', q' *are also primes.) Finally, we need the algorithm to output an element* $\hat{g} \in \mathbb{Z}_n^*$ *of order* $p'q'$*.*
- $\mathsf{NIZK} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ *be a simulation-sound non-interactive zero knowledge proof system. In this section, we will use* Σ*-protocols made non-interactive using the Fiat-Shamir heuristic, so in this case* Setup *is empty and there is no common reference string.*

Ggen *can be constructed using standard techniques. For instance, first generate* n *using standard techniques, then repeatedly choose a small random number* r *until* $P = 2rn + 1$ *is a prime. Let* g' *be a generator of* $\mathbb{Z}_P^*$*. Then let* G *be the subgroup of* $\mathbb{Z}_P^*$ *generated by* $g = g'^{2r} \bmod P$*.*[12] *Finally, to construct the element* $\hat{g}$*, let* $u \in_\mathsf{R} \mathbb{Z}_n$ *and set* $\hat{g} = u^2 \bmod n$*. Indeed, this is a random square, and since the subgroup of squares modulo* n *has only large prime factors in its order (*p' *and* q'*), a random element is a generator with overwhelming probability*[13]*.*

$\mathsf{Setup}(1^\kappa)$**:** *Run* Ggen *to generate a modulus* n *and* $\hat{g} \in \mathbb{Z}_n^*$ *as explained above. Output* $pp = (n, \hat{g})$*.*

[12] The group G will be used in the construction of the PSDVS scheme.

[13] This set-up need to keep the factorization of n secret. Hence, to avoid relying on a trusted party, the parties can use an interactive protocol to generate n securely, there are several quite efficient examples in the literature.

$\mathsf{KeyGen}_\mathsf{S}(pp)$**:** *Pick* $sk_\mathsf{S} \in_\mathsf{R} \mathbb{Z}_n$*, and set* $pk_\mathsf{S} = \hat{g}^{sk_\mathsf{S}}$*. Output* $(sk_\mathsf{S}, pk_\mathsf{S})$*.*
$\mathsf{KeyGen}_\mathsf{R}(pp)$**:** *Pick* $\alpha_1, \alpha_2 \in_\mathsf{R} \mathbb{Z}_n$*, set* $sk_\mathsf{R} = (\alpha_1, \alpha_2)$*, and set* $pk_\mathsf{R} = (\beta_1, \beta_2) = (\hat{g}^{\alpha_1}, \hat{g}^{\alpha_2})$*.*

The public key values are statistically indistinguishable from random elements in the group generated by $\hat{g}$ *since* n *is a sufficiently good "approximation" to the order* $p'q'$ *of* $\hat{g}$*.*

$\mathsf{Enc}_{pp,sk_\mathsf{S},pk_\mathsf{R}}(k; r, b_1, b_2)$**:**

1. *The randomness should have been picked as follows:* $r \in_\mathsf{R} \mathbb{Z}_n$ *and* $b_1, b_2 \in_\mathsf{R} \{0,1\}$*.*
2. *Set* $c_1 = (-1)^{b_1}\hat{g}^r \bmod n$*.*
3. *Set* $c_2 = (n+1)^k((-1)^{b_2}\beta_1^{sk_\mathsf{S}}\beta_2^r \bmod n)^n \bmod n^2$*.*
4. *Let* π_{valid} *be a non-interactive zero-knowledge proof of knowledge wherein given public data* $(n, \hat{g}, (c_1, c_2))$*, the prover shows knowledge of a witness* $w = (k, r, b_1, v)$ *such that* $c_1 = (-1)^{b_1}\hat{g}^r$ *and* $c_2 = (n+1)^k v^n \bmod n^2$*. An honest prover can use* $v = (-1)^{b_2}\beta_1^{sk_\mathsf{S}}\beta_2^r \bmod n$*. The factor* $(-1)^{b_1}$ *is only in the ciphertext for technical reasons: it allows* π_{valid} *to be efficient.*

Output $c = (c_1, c_2, \pi_{valid})$*.*

$\mathsf{Dec}_{pp,sk_\mathsf{R}=(\alpha_1,\alpha_2),pk_\mathsf{S}}(c = (c_1, c_2, \pi_{valid}))$**:**

1. *Check that* c_1, c_2 *have Jacobi symbol 1 modulo* n*, and check* π_{valid}*. Output reject if either check fails.*
2. *Let* $u = pk_\mathsf{S}^{\alpha_1}c_1^{\alpha_2} \bmod n$ *and check that* $(c_2u^{-n})^n \bmod n^2 = \pm 1$*.* $\mathbb{Z}^*_{n^2}$ *contains a unique subgroup of order* n*, generated by* $n+1$*. So here we are verifying that – up to a sign difference –* $c_2u^{-n} \bmod n^2$ *is in the subgroup generated by* $n+1$*. If the check fails, output reject.*
3. *Otherwise, compute* k *such that* $(n+1)^k = \pm c_2u^{-n} \bmod n^2$*.*[14]

An AVPKE scheme should allow anyone to make "fake" ciphertexts that look indistinguishable from real encryptions, given only the system parameters. Furthermore, the receiver R should be able to use his own secret key sk_R and the public key pk_S of the sender to make ciphertexts with exactly the same distribution as real ones. This is indeed true for our scheme:

Fake Encryption: Let $r \in_\mathsf{R} \mathbb{Z}_n, b, b' \in_\mathsf{R} \{0,1\}$ and $v \in \mathbb{Z}^*_n$ be a random square. Then, $\mathsf{Enc}_{pp,fake}(k; r, b, b', v) = ((-1)^b\hat{g}^r \bmod n, (n+1)^k((-1)^{b'}v)^n \bmod n^2), \pi_{valid}$ where π_{valid} is constructed following the NIZK prover algorithm.

R's Equivalent Encryption: $\mathsf{Enc}_{pp,sk_\mathsf{R},pk_\mathsf{S}}(k; r, b_1, b_2) =$
$((-1)^{b_1}\hat{g}^r \bmod n,\ (n+1)^k(-1)^{b_2}(pk_\mathsf{S}^{\alpha_1}\hat{g}^{r\alpha_2} \bmod n)^n \bmod n^2),\ \pi_{valid}$
where $r \in_\mathsf{R} \mathbb{Z}_n$, $b_1, b_2 \in_\mathsf{R} \{0,1\}$ and π_{valid} is constructed following the NIZK prover algorithm.
In the following, we will sometimes suppress the randomness from the notation and just write, e.g., $\mathsf{Enc}_{pp,sk_\mathsf{S},pk_\mathsf{R}}(k)$.

By simple inspection of the scheme it can be seen that:

[14] k can be computed using the standard "discrete log" algorithm from Paillier decryption.

Lemma 1. *For all k, $\mathsf{Dec}_{pp,sk_\mathsf{R},pk_\mathsf{S}}(\mathsf{Enc}_{pp,sk_\mathsf{S},pk_\mathsf{R}}(k)) = k$. Furthermore, encryption by S and by R returns the same ciphertexts: for all messages k and randomness r, b_1, b_2, we have $\mathsf{Enc}_{pp,sk_\mathsf{S},pk_\mathsf{R}}(k; r, b_1, b_2) = \mathsf{Enc}_{pp,sk_\mathsf{R},pk_\mathsf{S}}(k; r, b_1, b_2)$.*

Lemma 2. *If DDH in $\langle \hat{g} \rangle$ is hard, then $(k, \mathsf{Enc}_{pp,sk_\mathsf{S},pk_\mathsf{R}}(k; r, b_1, b_2))$ is computationally indistinguishable from $(k, \mathsf{Enc}_{pp,fake}(k; r', b, b', v))$ for any fixed message k and randomness $r, b_1, b_2, r', b, b', v$, as long as the discrete log of β_2 to the base $\hat{g}$ is unknown.*

Definition 16. *Consider the following experiment for an AVPKE scheme and a probabilistic polynomial time adversary $\mathcal{A}$: Run the set-up and key generation and run $A^{\mathcal{O}_E, \mathcal{O}_D}(pp, pk_\mathsf{R}, pk_\mathsf{S})$. Here, $\mathcal{O}_E$ takes a message k as input and returns $\mathsf{Enc}_{pp,sk_\mathsf{S},pk_\mathsf{R}}(k)$, while $\mathcal{O}_D$ takes a ciphertext and returns the result of decrypting it under $pk_\mathsf{S}, sk_\mathsf{R}$ (which will be either reject or a message). $\mathcal{A}$ wins if it makes $\mathcal{O}_D$ accept a ciphertext that was not obtained from $\mathcal{O}_E$. The scheme is* authentic *if any PPT $\mathcal{A}$ wins with negligible probability.*

Lemma 3. *If the DDH problem in $\langle \hat{g} \rangle$ is hard, the AVPKE scheme defined above is authentic.*

Due to space limitations, the proof of Lemma 3 is deferred to the full version.

We proceed to show that the AVPKE scheme hides the message encrypted even if adversary knows the secret key of the sender, and even if a decryption oracle is given. This is essentially standard CCA security.

Definition 17. *Consider the following experiment for an AVPKE scheme and a probabilistic polynomial time adversary $\mathcal{A}$: Run the set-up and key generation and run $\mathcal{A}^{\mathcal{O}_E}(pp, pk_\mathsf{R}, sk_\mathsf{S})$. Here, $\mathcal{O}_E$ takes two messages k_0, k_1 as input, selects a bit η at random and returns $c^* = \mathsf{Enc}_{pp,sk_\mathsf{S},pk_\mathsf{R}}(k_\eta)$. $\mathcal{O}_D$ takes a ciphertext and returns the result of decrypting it under $pk_\mathsf{S}, sk_\mathsf{R}$ (which will be either reject or a message). $\mathcal{A}$ may submit anything other than c^* to $\mathcal{O}_D$, and must output a bit η' at the end. It wins if $\eta' = \eta$. The scheme is* private *if any PPT $\mathcal{A}$ wins with negligible advantage over $\frac{1}{2}$.*

In the following we will use the assumption underlying the Paillier encryption scheme, sometimes known as the *composite degree residuosity assumption* (CDRA): a random element x in $\mathbb{Z}^*_{n^2}$ where $x \bmod n$ has Jacobi symbol 1 is computationally indistinguishable from $y^n \bmod n^2$ where $y \in \mathbb{Z}^*_n$ is random of Jacobi symbol 1[15].)

Lemma 4. *Assume that DDH in $\langle \hat{g} \rangle$ is hard and that CDRA holds. Then the AVPKE scheme satisfies Definition 17.*

Due to space limitations, the proof of Lemma 4 is deferred to the full version.

We say that an AVPKE scheme is *secure* if it is authentic, private, supports equivalent encryption by R and indistinguishable fake encryption.

[15] The original CDRA assumption does not have the restriction to Jacobi symbol 1, but since the Jacobi symbol is easy to compute without the factors of n, the two versions are equivalent.

Construction 4 (PSDVS Scheme) *Let:*

- Ggen *be a* Group Generator*, a probabilistic polynomial time algorithm which on input* 1^κ *outputs* $G, g, n, \hat{g}$ *exactly as in the previous AVPKE construction.*
- H *be a hash function which we model as a random oracle. We assume it maps onto the group* G*.*
- $\mathsf{NIZK} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ *be a simulation-sound non-interactive zero knowledge proof system. In this section, we will use* Σ*-protocols made non-interactive using the Fiat-Shamir heuristic, so in this case* Setup *is empty and there is no common reference string.*

$\mathsf{Setup}(1^\kappa)$**:** Let $(G, g, \hat{g}, n) \leftarrow \mathsf{Ggen}(1^\kappa)$ and let $h \in_\mathsf{R} G$. Set $pp = (G, g, \hat{g}, n, h)$. Return pp as the public parameters.

$\mathsf{SignKeyGen}(pp)$**:** Run key generation for the AVPKE scheme as defined above to get keys $ssk = sk_\mathsf{S}, spk = pk_\mathsf{S}$ for the signer S. Output ssk as the signer's secret key and spk as the signer's public key.

$\mathsf{VerKeyGen}(pp)$**:**

1. Run key generation for the AVPKE scheme as defined above to get keys $sk_\mathsf{R}, pk_\mathsf{R}$ for the verifier R. (These keys will be used to sign messages and verify signatures.)
2. Choose $k_\mathsf{R} \in_\mathsf{R} \mathbb{Z}_n$. (This key will be used by the verifier to simulate signatures using $\mathsf{VerSigSim}$.)
3. Choose $r_\mathsf{R} \in_\mathsf{R} \mathbb{Z}_n$ and let $c_\mathsf{R} = g^{k_\mathsf{R}} h^{r_\mathsf{R}}$. (This commitment will be used by the verifier to support its proofs of fake-ness.)
4. $vsk = (sk_\mathsf{R}, k_\mathsf{R}, r_\mathsf{R})$, $vpk = (pk_\mathsf{R}, c_\mathsf{R})$.

Output vsk as the verifier's secret key and vpk as the verifier's public key.

$\mathsf{Sign}(pp, ssk = sk_\mathsf{S}, pk_\mathsf{R}, m)$**:**

1. Choose $t \in_\mathsf{R} G, r \in_\mathsf{R} \mathbb{Z}_n, b_1, b_2 \in_\mathsf{R} \{0,1\}, s \in_\mathsf{R} \mathbb{Z}_n^*, k_s \in_\mathsf{R} \mathbb{Z}_n$.
2. Let $\sigma \leftarrow (t, \mathsf{H}(m,t)^{k_s}, \mathsf{Enc}_{pp, sk_\mathsf{S}, pk_\mathsf{R}}(k_s; r, b_1, b_2))$.
3. $\pi \leftarrow \mathsf{NIZK.Prove}(u = (\sigma = (\sigma_1, \sigma_2, \sigma_3), pk_V, pk_S, m), w = (sk_S, k_s, r, b_1, b_2))$ be a zero knowledge proof of knowledge of witness w such that:

$$\sigma_2 = \mathsf{H}(m, \sigma_1)^{k_s} \wedge \sigma_3 = \mathsf{Enc}_{pp, sk_\mathsf{S}, pk_\mathsf{R}}(k_s; r, b_1, b_2)$$

Output σ as the signature, and π as the proof of real-ness.

$\mathsf{Verify}(pp, spk = pk_\mathsf{S}, vsk = (sk_\mathsf{R}, k_\mathsf{R}, r_\mathsf{R}), m, \sigma = (\sigma_1, \sigma_2, \sigma_3))$**:**

1. Decrypt σ_3 as $k_s = \mathsf{Dec}_{pp, sk_\mathsf{R}, pk_\mathsf{S}}(\sigma_3)$. If this fails, set $d = 0$ and abort.
2. If $\sigma_2 = \mathsf{H}(m, \sigma_1)^{k_s}$, set $d = 1$. Otherwise, set $d = 0$.

Output d as the verification decision.

$\mathsf{PubSigSim}(pp, m)$**:**

1. Choose $k, k' \in_\mathsf{R} \mathbb{Z}_n$, such that $k \neq k'$.
2. Choose $t \in_\mathsf{R} G$, $r \in_\mathsf{R} \mathbb{Z}_n, b, b' \in_\mathsf{R} \{0,1\}$, $v \in_\mathsf{R} \mathbb{Z}_n$, such that v has Jacobi symbol 1.
3. $\sigma \leftarrow (t, \mathsf{H}(m,t)^k, \mathsf{Enc}_{pp, fake}(k'; r, b, b', v))$.
4. Let $\pi \leftarrow \mathsf{NIZK.Prove}(u = \sigma = (\sigma_1, \sigma_2, \sigma_3), w = (k, k'))$ be a zero-knowledge proof of knowledge such that:

$$\sigma_2 = \mathsf{H}(m, \sigma_1)^k \wedge \sigma_3 = \mathsf{Enc}_{pp, fake}(k'; \cdot, \cdot, \cdot, \cdot) \wedge k \neq k'.$$

Output σ as the simulated signature, and π as the proof of fake-ness. The notation $\mathsf{Enc}_{pp,fake}(k';\cdot,\cdot,\cdot,\cdot)$ means that the proof only has to establish that the plaintext inside the encryption is some value k' different from k.

$\mathsf{VerSigSim}(pp, spk = pk_{\mathsf{S}}, vpk = (pk_{\mathsf{R}}, c_{\mathsf{R}}), vsk = (sk_{\mathsf{R}}, k_{\mathsf{R}}, r_{\mathsf{R}}), m)$**:**

1. Choose $r_t \in_{\mathsf{R}} \mathbb{Z}_n$, $t = g^{k_{\mathsf{R}}} h^{r_t}$, $k_s \in_{\mathsf{R}} \mathbb{Z}_n$, $b_1, b_2 \in_{\mathsf{R}} \{0,1\}$ and $v \in_{\mathsf{R}} \mathbb{Z}_n^*$ of Jacobi symbol 1.
2. $\sigma \leftarrow (t, \mathsf{H}(m,t)^{k_s}, \mathsf{Enc}_{pp,sk_{\mathsf{R}},pk_{\mathsf{S}}}(k_s; r, b_1, b_2))$.
3. Let $\pi \leftarrow \mathsf{NIZK.Prove}(u = ((\sigma_1, \sigma_2, \sigma_3), c_{\mathsf{R}}, m), w = (k_{\mathsf{R}}, r_{\mathsf{R}}, r_t))$ be a zero-knowledge proof of knowledge of witness $w = (k_{\mathsf{R}}, r_{\mathsf{R}}, r_t)$ such that:
$$\sigma_1 = g^{k_{\mathsf{R}}} h^{r_t} \wedge c_{\mathsf{R}} = g^{k_{\mathsf{R}}} h^{r_{\mathsf{R}}}.$$

Output σ as the simulated signature and π as the proof of fake-ness.

Theorem 4. *If the AVPKE scheme is secure, and under the DDH assumption, Construction 4 is a secure PSDVS scheme.*

Due to space limitations, the proof of Theorem 4 is deferred to the full version. In the full version we describe a PSDVS scheme based on prime order groups. It gets around the need to generate a Paillier modulus securely, at the cost of requiring double discrete log proofs.

4 FE-based Construction

In this section, we present an MDVS scheme based on functional encryption. One disadvantage of this scheme is that it requires a trusted setup; secret verification keys must be derived from a master secret key. However, the accompanying advantage is that this scheme has verifier-identity-based signing; verifiers' public keys consist simply of their identity, allowing any signer to encrypt to any set of verifiers without needing to retrieve their keys from some PKI first.

At a high level, we are first given a digital signature scheme (DS) and a functional encryption scheme (FE). The keys of the signer with identity i are a secret DS signing key sk_i and corresponding public DS verification key vk_i. An MDVS signature c is a FE ciphertext obtained by encrypting the plaintext that consists of the message m, the signer's DS verification key vk_i, a set of designated verifier identities $\mathcal{D}$, and the signer's DS signature σ on the message using the secret DS signing key sk_i. That is, $c = \mathsf{FE.Enc}(pp, (m, vk_i, \mathcal{D}, \sigma))$. Verifier j's public key is simply their identity j (that is, $vpk_j = j$). Their secret key consists of a DS key pair (sk_j, vk_j), and an FE secret key dk_j. dk_j is the secret key for a function that checks whether j is among the specified designated verifiers, and then checks whether the DS signature σ inside the ciphertext c is either a valid signature under the signer's verification key vk_i, or under the verifier's verification key vk_j. However, this basic scheme does not give us the off-the-record property; we therefore tweak it slightly, as we describe below.

From One to Many DS Signatures In order to ensure that any subset of valid verifiers cannot convince an outsider of the origin of the MDVS signature, we need to replace the one DS signature in the ciphertext with a set of DS signatures. The reason is that, if only one signature is contained in the ciphertext, any designated verifier can prove to an outsider that "it was either me or the

signer that constructed the signature". If more than one verifier proves this about the same MDVS signature, then the signature must have come from the signer.

To prevent this kind of "intersection attack", we allow the ciphertext to contain a set Σ of DS signatures, and change the corresponding FE secret keys to check if there exists a DS signature in the set that either verifies under the signer's or the verifier's DS verification key. Now, an outsider will no longer be convinced that it was the signer who constructed the MDVS signature, since each of the colluding verifiers could have constructed a DS signature that verifies under their own verification key, and then encrypted this set together with the public verification key of the signer.

Achieving Consistency In order to achieve consistency, we need security against malicious encryption in the underlying FE scheme. We need to ensure that any (possibly maliciously generated) ciphertext is consistent with one specific message across decryption with different functions. Otherwise, a malicious MDVS signer may be able to construct a ciphertext (i.e. a signature) that will be valid for one designated verifier but not valid for another, thereby breaking the consistency property. Security against a malicious encryption is a property of verifiable functional encryption (VFE), which was introduced by Badrinarayanan et. al [BGJS16]. However, it turns out that we do not need the full power of VFE, which also includes precautions against a malicious setup. Thus, we define a weaker notion of VFE, and substitute the standard FE scheme with this new scheme allowing us to achieve the MDVS consistency property.

4.1 Ciphertext Verifiable Functional Encryption

The formal definition of Functional Encryption and Ciphertext Verifiable FE and the security notions can be found in the full version. Informally, the ciphertext verifiability property states that for all ciphertexts c, it must hold that if c passes the verification algorithm, then there exists a unique plaintext x asociated with c, meaning that for all functions $f \in \mathcal{F}$ the decryption of c will yield $f(x)$.

4.2 The MDVS Construction

Construction 5 *Let* $\mathsf{SIGN} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *be a standard digital signature scheme and let* $\mathsf{VFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Verify})$ *be a functional encryption scheme secure with ciphertext verifiability. Then we define a MDVS scheme* $\mathsf{FEMDVS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Sim})$ *as follows:*

$\mathsf{Setup}(1^\kappa)$**:** $(pp^{\mathsf{FE}}, msk^{\mathsf{FE}}) \leftarrow \mathsf{VFE.Setup}(1^\kappa)$.

Output public parameter $pp = pp^{\mathsf{FE}}$ and master secret key $msk = msk^{\mathsf{FE}}$.

$\mathsf{SignKeyGen}(i)$**:** $(sk_i, vk_i) \leftarrow \mathsf{SIGN.KeyGen}(1^\kappa)$.

Output the signer's secret key $ssk_i = sk_i$ and public key $spk_i = vk_i$.[16]

$\mathsf{VerKeyGen}(msk, j)$**:**

1. $vpk_j = j$,
2. $(sk_j, vk_j) \leftarrow \mathsf{SIGN.KeyGen}(1^\kappa)$,

[16] We assume that the mapping $i \rightarrow (ssk_i, spk_i)$ is unique in the system. This can be achieved without loss of generality by pseudorandomly generating the randomness required in the key generation process from the identity i and the master secret key.

3. $dk_j \leftarrow \mathsf{VFE.KeyGen}(msk^{\mathsf{FE}}, f_j)$, where f_j is defined as follows.

Function f_j
Input: $m, vk_i, \{vpk_{j'}\}_{j' \in \mathcal{D}}, \Sigma$;
Const: vpk_j, vk_j;
1. If $vpk_j \notin \{vpk_{j'}\}_{j' \in \mathcal{D}}$: output $\perp$;
2. If $\exists \sigma \in \Sigma : \mathsf{SIGN.Verify}(vk_i, m, \sigma) = 1$ OR $\mathsf{SIGN.Verify}(vk_j, m, \sigma) = 1$: output $(m, vk_i, \{vpk_{j'}\}_{j' \in \mathcal{D}})$;
3. Else: output $\perp$

Output the verifiers secret key $vsk_j = (sk_j, dk_j)$ and public key $vpk_j = j$.[17]

$\mathsf{Sign}(pp, ssk_i, \{vpk_j\}_{j \in \mathcal{D}}, m)$**:**

1. $\sigma \leftarrow \mathsf{SIGN.Sign}(sk_i, m)$.
2. Output $c = \mathsf{VFE.Enc}(pp^{\mathsf{FE}}, (m, vk_i, \{vpk_j\}_{j \in \mathcal{D}}, \{\sigma, \perp, \cdots, \perp\}))$.

$\mathsf{Verify}(pp, spk_i, vsk_j, \{vpk_j\}_{j \in \mathcal{D}}, m, c)$**:**

1. Check whether $\mathsf{VFE.Verify}(pp^{\mathsf{FE}}, c) = 1$. If not, output 0.
2. Compute $(m', vk_i', \{vpk_j\}_{j \in \mathcal{D}'}) \backslash \perp \leftarrow \mathsf{VFE.Dec}(dk_j, c)$. If the output is $\perp$, output 0.
3. Check $m' = m$, $vk_i' = vk_i$ (with $spk_i = vk_i$), and $\mathcal{D}' = \mathcal{D}$. If all hold, output 1. Otherwise output 0.

$\mathsf{Sim}(pp, spk_i, \{vpk_j\}_{j \in \mathcal{D}}, \{vsk_j\}_{j \in \mathcal{C}}, m)$**:**

1. For each $j \in \mathcal{C}$, $vsk_j = (sk_j, dk_j)$.
2. Compute $\sigma_j \leftarrow \mathsf{SIGN.Sign}(sk_j, m^*)$.
3. Let $\Sigma = \{\sigma_j\}_{j \in \mathcal{C}^*}$, add default values to get the required size.
4. Output $c = \mathsf{VFE.Enc}(pp^{\mathsf{FE}}, (m^*, spk_i, \{vpk_j\}_{j \in \mathcal{D}}, \Sigma))$.

Theorem 5. *Assume that* VFE *is an IND-CPA secure functional encryption scheme with ciphertext verifiability, and* SIGN *is an existential unforgeable digital signature scheme. Then Construction 5 is a correct and secure MDVS scheme with privacy of identities and verifier-identity-based signing.*

Due to space limitations, the proof of Theorem 5 is deferred to the full version.

References

AV19. Prabhanjan Ananth and Vinod Vaikuntanathan. Optimal bounded-collusion secure functional encryption. pages 174–198, 2019.

BGB04. Nikita Borisov, Ian Goldberg, and Eric Brewer. Off-the-record communication, or, why not to use PGP. In *Proceedings of the 2004 ACM workshop on Privacy in the electronic society*, pages 77–84. ACM, 2004.

BGJS16. Saikrishna Badrinarayanan, Vipul Goyal, Aayush Jain, and Amit Sahai. Verifiable functional encryption. pages 557–587, 2016.

Cha96. David Chaum. Private signature and proof systems, 1996. US Patent 5,493,614.

Cha11. Ting Yi Chang. An id-based multi-signer universal designated multi-verifier signature scheme. *Inf. Comput.*, 209(7):1007–1015, 2011.

[17] We assume that the mapping $j \to (vsk_j, vpk_j)$ is unique in the system. This can be achieved wlog by pseudorandomly generating the randomness required in the key generation process from the identity j and the master secret key.

Cho06. Sherman S. M. Chow. Identity-based strong multi-designated verifiers signatures. In *Public Key Infrastructure, Third European PKI Workshop: Theory and Practice, EuroPKI 2006, Turin, Italy, June 19-20, 2006, Proceedings*, pages 257–259, 2006.

Cho08. Sherman S. M. Chow. Multi-designated verifiers signatures revisited. *I. J. Network Security*, 7(3):348–357, 2008.

GVW12. Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. pages 162–179, 2012.

JSI96. Markus Jakobsson, Kazue Sako, and Russell Impagliazzo. Designated verifier proofs and their applications. pages 143–154, 1996.

LSMP07. Yong Li, Willy Susilo, Yi Mu, and Dingyi Pei. Designated verifier signature: Definition, framework and new constructions. In *Ubiquitous Intelligence and Computing, 4th International Conference, UIC 2007, Hong Kong, China, July 11-13, 2007, Proceedings*, pages 1191–1200, 2007.

LV04. Fabien Laguillaumie and Damien Vergnaud. Multi-designated verifiers signatures. pages 495–507, 2004.

LV07. Fabien Laguillaumie and Damien Vergnaud. Multi-designated verifiers signatures: anonymity without encryption. *Inf. Process. Lett.*, 102(2-3):127–132, 2007.

Mar13. Moxie Marlinspike. Advanced cryptographic ratcheting. 2013.

MW08. Yang Ming and Yumin Wang. Universal designated multi verifier signature scheme without random oracles. *Wuhan University Journal of Natural Sciences*, 13(6):685–691, Dec 2008.

NSM05. Ching Yu Ng, Willy Susilo, and Yi Mu. Universal designated multi verifier signature schemes. In *11th International Conference on Parallel and Distributed Systems, ICPADS 2005, Fuduoka, Japan, July 20-22, 2005*, pages 305–309, 2005.

RST01. Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. pages 552–565, 2001.

SHCL08. Seung-Hyun Seo, Jung Yeon Hwang, Kyu Young Choi, and Dong Hoon Lee. Identity-based universal designated multi-verifiers signature schemes. *Computer Standards & Interfaces*, 30(5):288–295, 2008.

SKS06. G. Shailaja, K. Phani Kumar, and Ashutosh Saxena. Universal designated multi verifier signature without random oracles. In *9th International Conference in Information Technology, ICIT 2006, Bhubaneswar, Orissa, India, 18-21 December 2006*, pages 168–171, 2006.

Tia12. Haibo Tian. A new strong multiple designated verifiers signature. *IJGUC*, 3(1):1–11, 2012.

Ver06. Damien Vergnaud. New extensions of pairing-based signatures into universal designated verifier signatures. pages 58–69, 2006.

ZAYS12. Yunmei Zhang, Man Ho Au, Guomin Yang, and Willy Susilo. (strong) multi-designated verifiers signatures secure against rogue key attack. In *Network and System Security - 6th International Conference, NSS 2012, Wuyishan, Fujian, China, November 21-23, 2012. Proceedings*, pages 334–347, 2012.

Zhe97. Yuliang Zheng. Digital signcryption or how to achieve cost(signature & encryption) $\ll$ cost(signature) + cost(encryption). pages 165–179, 1997.