On the Incoercibility of Digital Signatures

Ashley Fraser[1⋆], Lydia Garms[2⋆⋆], and Elizabeth A. Quaglia[3]

[1] Department of Computer Science, University of Surrey, UK
a.fraser@surrey.ac.uk
[2] Keyless Technologies Limited, London, UK
lydia.garms@keyless.io
[3] Information Security Group, Royal Holloway, University of London, UK
elizabeth.quaglia@rhul.ac.uk

Abstract. We introduce incoercible digital signature schemes, a variant of a standard digital signature. Incoercible signatures enable signers, when coerced to produce a signature for a message chosen by an attacker, to generate fake signatures that are indistinguishable from real signatures, even if the signer is compelled to reveal their full history (including their secret signing keys and any randomness used to produce keys/signatures) to the attacker. Additionally, we introduce an authenticator that can detect fake signatures, which ensures that coercion is identified. We present a formal security model for incoercible signature schemes that comprises an established definition of unforgeability and captures new notions of weak receipt-freeness, strong receipt-freeness and coercion-resistance. We demonstrate that an incoercible signature scheme can be viewed as a transformation of any generic signature scheme. Indeed, we present two incoercible signature scheme constructions that are built from a standard signature scheme and a sender-deniable encryption scheme. We prove that our first construction satisfies coercion-resistance, and our second satisfies strong receipt-freeness. We conclude by presenting an extension to our security model: we show that our security model can be extended to the designated verifier signature scheme setting in an intuitive way as the designated verifier can assume the role of the authenticator and detect coercion during the verification process.

1 Introduction

The fundamental security property of a digital signature scheme is considered to be the property of existential unforgeability against an adaptive chosen-message attack [17]. This captures the property that an attacker, with access only to public information, cannot output a valid message/signature pair (i.e., a forgery). In recognition of the threat posed by attackers with stronger, potentially coercive, capabilities, several security notions have been proposed that strengthen the traditional security model for signature schemes (cf. §1.1). These notions have predominantly focused on *key exposure attacks* [3, 13, 14, 18, 20–22, 30], whereby a signer is coerced to reveal (part of) their secret key to an attacker. The proposed schemes generally allow the signer to recover from the attack, most commonly by updating their secret key. The security of such schemes requires that the attacker cannot produce a valid signature on behalf of a signer whose key has been exposed. Other works have focused on specific techniques that allow a signer to evade coercion [16, 27]. These works consider an attacker that requests a signer to produce a signature for a particular message. The proposed solutions introduce a trusted authority that can detect coercion, and their security model requires that the attacker cannot distinguish a signer evading coercion from a signer who cooperates with the attacker.

In this work, we introduce *incoercible signatures*, which follow the approach of [16, 27] in that coercion can be detected by a trusted authority, the authenticator. Our contributions distinguish themselves from the existing literature in the following way. Firstly, we preserve the essence of

⋆ This author was supported by the EPSRC and the UK government as part of the Centre for Doctoral Training in Cyber Security at Royal Holloway, University of London under grant number EP/P009301/1, and the EPSRC Next Stage Digital Economy Centre in the Decentralised Digital Economy (DECaDE) at the University of Surrey under grant number EP/T022485/1.

⋆⋆ This author was supported by the Innovate UK funded project AQuaSec, whilst working at Royal Holloway, University of London, and a research grant from Nomadic Labs and the Tezos Foundation, whilst working at the IMDEA Software Institute.

a standard signature scheme, i.e., our primitive is fully non–interactive, during key generation, signing and verification. Secondly, while previous works modelled signers revealing their secret key, we allow the attacker to demand the *full* transcript of the signer throughout the protocol, thereby including, along with the secret key, any randomness chosen in key generation and signing. Thirdly, we recognise the benefits of capturing several variants of incoercibility and, therefore, our security model also captures attackers that demand a full transcript only *after* having provided instructions to the signers.

1.1 Existing Work on Incoercibility

The concept of incoercibility first emerged in the context of electronic voting (e-voting) where it was formalised as a hierarchy [11] of two security properties: receipt-freeness [4] and coercion-resistance [24]. Additionally, simulation-based definitions of incoercibility have been put forth in the generic multi-party protocol setting [2, 7, 37], again defined as a hierarchy of the receipt-freeness and coercion-resistance properties. In the digital signatures literature, coercion has been addressed in several ways. Security models and schemes have been proposed to protect against (partial) key exposure attacks, and mechanisms have been introduced to enable the signer to warn an authority of such attacks, either via direct communication or by embedding a coercion warning into the signature. We discuss these approaches in more detail next.

Key Exposure Attacks. An attacker that can obtain the secret key of a signer can easily construct valid forgeries on behalf of the signer. This is known as a key exposure attack and several solutions that allow recovery from such an attack exist. The earliest solutions required key distribution [12, 19, 32], but the most prevalent solution allows the signer to update their secret key [3, 13, 14, 20–22]. This approach became prominent following the introduction of forward secure signatures [3], which guarantee that an attacker cannot produce valid forgeries for any time period prior to key update. Subsequent works, for example [13, 14, 20–22], extend this guarantee, providing an assurance that an attacker cannot produce valid forgeries for time periods prior to, and following, the key update. However, many of these solutions provide these guarantees for partial key exposure only [13, 14, 21, 22]. By contrast, [3, 20, 22] consider full key exposure attacks. In a departure from this approach, monotonic signature schemes [30] allow the signer, in response to a partial key exposure attack, to update the verification algorithm, rather than the secret signing key. Monotonic signatures ensure that the attacker cannot produce valid forgeries after the verification algorithm is updated, but forgeries created before the algorithm update are valid. In this work, we consider an attacker that demands a full transcript from the signer, and require that the attacker cannot distinguish cooperating and non-cooperating signers. Additionally, our strongest security definition guarantees that an attacker cannot produce valid forgeries on behalf of a coerced signer.

Communicating Key Exposure. Generally, signature schemes that are secure against key exposure attacks present a limited window of time within which the attacker can produce a valid forgery, e.g., the time period when key exposure occurs in the case of [13, 14, 20–22]. Moreover, the requirement to update keys/algorithms may alert an attacker to an unsuccessful key exposure attack. For example, in the case of monotonic signatures, the attacker can check if their forgery is valid. If the forgery is invalid, it is clear that the signer updated their verification algorithm and thus evaded coercion. Similar arguments hold for other schemes secure against key exposure attacks. To address this, key evolving signature schemes [22] introduced a trusted authority with whom the signer can communicate. The signer is required to update their keys at regular intervals; if the signer does not contact the authority before key update occurs, any signatures generated since the previous key update period become invalid. In this way, forgeries created by the attacker can be invalidated. Moreover, the signer can communicate to the authority that the signatures were coerced. In doing so, the signer can evade coercion and, if the authority does not publicly invalidate the signatures, the attacker is not alerted to the unsuccessful coercion. Similarly, funkspiel schemes [18] introduce a trusted authority with whom the signer can communicate the compromise of keys. In our work, we also introduce a trusted authority that can detect coercion, though we do not require that the signer communicates with the authority. Rather, we include a 'coercion alert' in the signature that can be recovered only by the trusted authority.

Embedded Secret Signature Schemes. Embedded secret signatures [16, 27] allow signers to produce signatures that contain an embedded warning that can be extracted from the signature only by a trusted authority. In this way, the signer can evade coercion to sign a message chosen by the attacker, without detection by the attacker. Incoercible signatures achieve a similar goal. A key difference, however, is that embedded secret signature schemes require that the trusted authority interact with every signer during key generation. By contrast, incoercible signature schemes are non-interactive. Moreover, the security model for an embedded secret signature requires that an attacker cannot determine whether the signer signed the attacker's message or not, when provided with the signer's secret key. In comparison, our security model for incoercibility considers an attacker with access to full signer transcripts that include randomness used during signing and key generation, in addition to the signer's secret key.

Deniability and Incoercibility. Deniability is a property that can be useful in the context of coercion within cryptographic protocols. Indeed, if a protocol participant is coerced to perform a particular action, deniability can be used to allow the participant to perform a different action (or no action) and prove that they followed the coercer's instructions. However, deniability captures a wider range of attacks. The participant can, when coerced not to do something, perform the action and deny it afterwards. In other words, deniability captures the ability of a participant to repudiate their actions to all but the intended receiver.

Deniability has been considered widely in the cryptographic literature (e.g., [5,6,8,31,34]). In the context of digital signatures, deniability, if defined analogously to other cryptographic primitives, would provide an assurance that the signer can deny having produced a signature. Therefore, deniability captures coercive attacks where the attacker instructs the signer *not* to sign messages. However, deniability in this context seems difficult to achieve. Certainly, a message/signature pair is public, which suggests that a signer cannot deny producing the signature. Therefore, in this work, we focus on incoercibility, which captures coercive attacks where the attacker instructs the signer to sign a message of their choice. Additionally, we introduce incoercible strong designated verifier signature schemes, which are not publicly verifiable, capturing incoercibility as well as the aspects of deniability featured in strong designated verifier schemes.

1.2 Our Contributions

The main contribution of this paper is the introduction and formalisation of incoercible signature schemes. In particular, we define three notions of incoercibility: weak receipt-freeness, strong receipt-freeness and coercion-resistance. We demonstrate the feasibility of our primitive by presenting constructions of coercion-resistant and strong receipt-free incoercible signature schemes, and we prove that they satisfy our security model. We also consider a relevant related primitive: designated verifier signature schemes. Finally, we motivate the study of incoercible signatures by describing some applications. We next describe our contributions in detail.

Defining Incoercible Signatures We introduce the syntax for an incoercible signature scheme (§2). Incoercibility means that a signer can sign messages as they desire and, conversely, need not produce signatures for messages that they do not want to sign. We define incoercible signatures as a variant of a standard signature scheme, providing algorithms for key generation, signing and public verification of a signature. To realise incoercible signatures, we introduce two new requirements into our syntax. Firstly, we introduce the ability to produce 'fake' signatures that are indistinguishable from real signatures. In this way, a coerced signer can produce a fake signature for a message chosen by the attacker. Formally, we introduce a new algorithm FakeSign that outputs a fake signature. Secondly, we introduce a trusted authority, which we call the authenticator, that can distinguish real signatures produced by running standard signing algorithm Sign and signatures that are the output of algorithm FakeSign. In this paper, we adopt the convention of calling a signature valid or verifiable if it passes public verification, and, additionally, call a signature authentic if the designated authenticator deems the signature to be real, as opposed to fake. In our syntax, we introduce an algorithm Authenticate that is run by the authenticator and outputs whether the signature is authentic or not. Note that, without a trusted authority, fake signatures are identical to real signatures. However, an authenticator is only trusted to determine whether a signature is coerced

and learns nothing about the signer's secrets. Crucially, this means that the authenticator cannot produce forgeries on behalf of any signer. We also highlight that, in practice, the authenticator must be carefully chosen to ensure that the correct authority is alerted to coercion.

Our syntax assumes that signers keep a transcript of their actions, which we denote trans. Informally, trans is a record of all computations performed, and inputs generated, by the signer. We include such a transcript in our syntax to model the fact that a signer may present an attacker with 'evidence' (i.e., the transcript) to prove that they have followed the attacker's instructions. We stress that our syntax includes the transcript only for the purpose of modelling coercive attacks. It is not necessary for a signer to keep track of the transcript in order to sign. In this way, incoercible signatures can be distinguished from stateful signatures, which require that the signer's state is input to the signing algorithm to compute a signature. A transcript is uniquely defined by a given signature scheme, and we provide a concrete example of this here. We illustrate the form of trans in the case of the DSA/ECDSA signature scheme, whose description is taken from [25].

Example 1 (DSA/ECDSA). DSA (respectively, ECDSA) is defined relative to a set of public parameters $pp_{\mathsf{SIG}} = (\mathbb{G}, g, q)$ where $\mathbb{G}$ is a q-order subgroup of $\mathbb{Z}_p^*$ (resp., $E(\mathbb{Z}_p)$) for p a prime. We denote the generator of $\mathbb{G}$ as g. We also assume that functions $G : \mathbb{G} \to \mathbb{Z}_q$ and $H : \{0,1\}^* \to \mathbb{Z}_q$ are defined during setup. Then, DSA/ECDSA is defined in Figure 1. The transcript trans (after key generation and the signing of one message m) is defined as the tuple $((pk_{\mathsf{SIG}}, sk_{\mathsf{SIG}}), (k, \sigma, m))$. We note that, with the transcript, it is possible to verify that $pk_{\mathsf{SIG}} = g^{sk_{\mathsf{SIG}}}$ and recompute the signature σ generated by the signer.

$\mathsf{KGen}(pp_{\mathsf{SIG}})$	$\mathsf{Sign}(pp_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, m)$	$\mathsf{Vf}(pp_{\mathsf{SIG}}, pk_{\mathsf{SIG}}, m, \sigma)$
$sk_{\mathsf{SIG}} \leftarrow \mathbb{Z}_q$;	$k \leftarrow \mathbb{Z}_q^*$; $r := G(g^k)$;	**if** $r \neq G(g^{H(m)\cdot s^{-1}} \cdot pk_{\mathsf{SIG}}^{r\cdot s^{-1}})$
$pk_{\mathsf{SIG}} := g^{sk_{\mathsf{SIG}}}$;	$s := (k^{-1} \cdot (H(m) + sk_{\mathsf{SIG}} \cdot r)) \mod q$;	**return** 0
return $(pk_{\mathsf{SIG}}, sk_{\mathsf{SIG}})$	**return** $\sigma = (r, s)$	**return** 1

Fig. 1: The DSA/ECDSA signature scheme for which we define the transcript in Example 1.

Additionally, our syntax models the generation of a fake transcript, denoted $\mathsf{trans}_{\mathsf{fake}}$. The fake transcript is used by a signer that attempts to evade coercion by providing a transcript that will convince an attacker that the signer followed the attacker's instructions. As we shall see in Section 2.1, maintaining a real and fake transcript is necessary to ensure that a signer can evade coercion, the main security goal of incoercible signatures.

A Security Model for Incoercible Signatures We present a comprehensive security model for incoercible signatures (§2.1). Our security model captures standard notions of correctness and unforgeability for a signature scheme, adapted to our syntax. We also introduce a further property that we call completeness, which captures the notion that an honestly generated signature is authentic. We then define three further security properties for incoercible signatures: weak receipt-freeness, strong receipt-freeness and coercion-resistance. In this way, our security model captures three variants of incoercibility that are influenced and inspired by the existing literature [7, 9, 26, 37]. Here we provide an overview of our incoercibility notions, and present our formal definitions in Section 2.1.

Our incoercibility notions consider signers that fall into one of the following three categories: 1) *Honest* signers follow the protocol and do not interact with the attacker; 2) *Corrupt* signers are controlled by the attacker. The attacker can act on behalf of corrupt signers and is assumed to have the secret signing key of corrupt signers; 3) *Coerced* signers are signers that are provided with instructions by the attacker. The attacker can request that the signer provide some 'proof' that they followed the attacker's instructions. Coerced signers may appear to cooperate with the attacker when they are, in fact, deceiving the attacker. For incoercible signatures, we are most interested in this category of signers.

Our security definitions allow an attacker to *adaptively* corrupt or coerce signers, meaning that corruption or coercion can occur at any point after key generation. Furthermore, we assume that

signers can fall into only one of these categories. That is, corrupt signers cannot be coerced and coerced signers cannot be corrupt. We omit simultaneous corruption and coercion from our security model for the following reasons. Firstly, in the former scenario, an attacker gains no advantage by coercing a corrupt signer. Secondly, in the latter scenario, coercion can be trivially detected if the full transcript of the signer is revealed upon corruption. Indeed, in [37], it is established that a notion of incoercibility that allows simultaneous corruption and coercion is, arguably, too strong a notion and, for most application scenarios, is unnecessary.

Our notions of incoercibility are grounded in the understanding of incoercibility developed over the years, in particular in the e-voting literature [9, 24, 26]. We reflect this understanding by introducing the properties of *indistinguishability* and *soundness* to our security model, as we describe next.

Indistinguishability. We introduce three variants of indistinguishability that capture the following intuition: an attacker, who can request transcripts from coerced signers, cannot distinguish a signer that cooperates with an attacker and presents a real transcript from a signer that evades coercion and presents a fake transcript. Our weakest notion of indistinguishability, which we call IND1, considers an attacker that interacts with signers only after signing. Building on this, we define IND2 indistinguishability, which captures an attacker that can interact with signers throughout the protocol (i.e., before and after signing). Finally, our strongest variant, IND3 indistinguishability, captures an attacker that provides the signer with the randomness used to sign, in addition to the attacker's message.

Soundness. It is essential that, if a signature is fake, the authenticator will determine the signature to be inauthentic. To capture this, we introduce *soundness*, which guarantees that a coerced signer can communicate coercion to the authenticator. It requires that, if a signature is the output of algorithm FakeSign, the authenticator will determine the signature to be fake. We also recognise that some attackers may attempt to act on behalf of signers (for example, by attempting to produce signatures on behalf of coerced signers). Consequently, we introduce a second soundness property that we call *strong soundness*. Strong soundness is the property that an attacker cannot output an authentic signature on behalf of a coerced, but uncooperative, signer. In other words, strong soundness requires that an adversary, who can request fake signatures and fake transcripts on behalf of coerced signers, cannot produce an authentic signature for a coerced signer.

Our indistinguishability and soundness properties can be combined in various ways to capture a spectrum of incoercibility attackers. In this work, we focus on combining these properties to achieve three well-established incoercibility notions from the literature. Following convention in the e-voting literature, we call our incoercibility properties weak receipt-freeness, strong receipt-freeness and coercion-resistance, described in more detail in Section 2.

Achieving Incoercible Signatures Suprisingly, we show that even the strongest notion of incoercibility for digital signatures (cf. §2.2) can be achieved using existing well-established cryptographic primitives. Indeed, we present two conceptually simple incoercible signature scheme constructions (§3) that rely on a generic standard signature scheme and a sender-deniable encryption scheme. We prove that our first construction satisfies coercion-resistance and our second construction satisfies strong receipt-freeness. By presenting such constructions, we demonstrate that our notion of coercion-resistance is achievable, and that our notion of strong receipt-freeness is achievable with some efficiency savings (when compared to our coercion-resistant construction). Note that we do not present a construction for our weak receipt-freeness notion. Of course, our two constructions satisfy weak receipt-freeness, but we do not present a construction that satisfies *only* weak receipt-freeness. We do this as we aim to achieve the strongest security possible in our constructions (whilst balancing this with efficiency). We leave as an open problem whether more efficient constructions are possible that can satisfy our weakest notion. Looking forward, in Section 4, we explicit the relation between incoercibility and deniable encryption, and demonstrate that weak receipt-freeness implies a variant of deniable encryption. As a result, we conjecture that the efficiency of weak receipt-free schemes and deniable encryption are closely related. That being said, we choose to include weak receipt-freeness in our security model to provide a complete vision of incoercibility for the digital signature space.

Extending Incoercibility to the Designated Verifier Setting We demonstrate that our security notions can be extended in an intuitive way to the setting of designated verifier signature schemes (§5). Indeed, designated verifier signature schemes are well-suited to the notion of incoercibility because the designated verifier can assume the role of the authenticator and detect coercion during verification. We provide a security model for incoercible strong designated verifier signature schemes that captures notions of correctness, unforgeability, source hiding and privacy of signer's identity as defined in [28]. Additionally, we define incoercibility, which corresponds to the definitions of weak receipt-freeness, strong receipt-freeness and coercion-resistance for publicly verifiable signature schemes. We also present a construction that satisfies our security model for coercion-resistance. Our construction is similar to our coercion-resistant construction, except that the standard publicly verifiable signature scheme is replaced with a strong designated verifier signature scheme. We present our results in Section 5.

1.3 Utility of Incoercible Signatures

In a world where online intimidation is increasingly prevalent, it seems clear that extending the notion of incoercibility to the setting of digital signatures is a useful endeavour. Indeed, in the context of signatures with a designated verifier, it is straightforward to see that our notion of incoercibility can directly offer protection against forced influence or coercion: the coerced signer can signal the coercion attempt to the verifier, who is uniquely positioned to verify the signature and discard it.

With respect to standard digital signatures, however, coercion can only be detected by the authenticator and not when signatures are verified, making a successful evasion more challenging. Nevertheless, incoercible signatures can be of use in this context. Consider, for instance, a scenario in which members of an organisation can produce endorsements on behalf of other members. Such endorsements can be used, for example, to recommend a particular member for an advertised role within the organisation. It may be desirable that members *publicly* endorse other members; endorsements accompanied with publicly verifiable signatures can facilitate this. However, it is also possible that a member may be coerced by another member (the coercer) to produce an endorsement. To address this, the organisation can implement a coercion-resistant incoercible signature scheme, ensuring that the coerced member can indicate coercion to an authenticator (i.e., the entity who ultimately determines which members are appointed to advertised roles). The authenticator can subsequently refuse to appoint the coercer to the role, without revealing the fact that the endorsing member evaded the coercion attempt. As such, the use of coercion-resistant incoercible signatures achieves a balance of public verifiability and protection from coercive attacks.

Similarly, in reputation systems reviews are provided for a product and can be accompanied by a publicly verifiable signature to provide trust in their origins, without relying on a trusted third party. Moreover, an authenticator can provide a tally that averages the uncoerced reviews, without revealing which reviews are included in this tally. In this scenario, a verifier has access to two sources of information: a set of scores that may be coerced but do not rely on a trusted third party and a tally that does not contain coerced reviews (assuming the authenticator behaves honestly). The verifier can choose which source to trust.

In these scenarios, the authenticator need not publicly reveal the coercion attempt. Indeed, in the first scenario, the authenticator could fabricate a reason for not appointing the coercer to the role. Along with the designated verifier setting, this is crucial to the utility of incoercible signatures. As with other incoercible primitives, e.g., deniable encryption, the user of an incoercible signature must consider the non-cryptographic consequences of evading coercion. In some scenarios, the actions of the verifier may indicate an unsuccessful coercion attempt to the coercer. For example, if a coercer requests the transfer of funds, the sender can indicate coercion to the entity that will perform the transfer by producing an incoercible signature for the transfer request. The entity will be alerted to the coercion attack and will not transfer the funds. However, if the coercer does not receive the funds, they will be alerted to the fact that the sender evaded the coercion attempt. In this scenario it is more difficult for the verifier to provide a convincing reason not to send the funds, and so incoercible signatures are not suitable.

Incoercible signatures can also be used within larger protocols for which receipt-freeness and coercion-resistance are desirable properties. One example of such a protocol is electronic voting

(e-voting). Briefly, registered voters could sign an encrypted vote with an incoercible signature. Voter ballots (i.e., their encrypted vote and incoercible signature) can be submitted to a trusted election official for tallying. In the role of authenticator, the election official can then check whether the signature is coerced and can discard any ballots that signal coercion. In this way, the election official can exclude ballots from the result without revealing any coercion attempts. Furthermore, ballots, including coerced ballots, can be posted to a public bulletin board. Using an incoercible signature means that anyone can verify signatures attached to ballots, thus checking that all ballots are submitted by registered voters.

2 Incoercible Signatures

In this section, we introduce the syntax and security model for an incoercible signature scheme. Recall from the introduction that incoercible signatures are a variant of a standard signature scheme, providing standard algorithms for key generation, signing and public verification of a signature. Additionally, incoercible signatures are equipped with an algorithm $\mathsf{FakeSign}$ that allows signers to generate fake signatures and algorithm $\mathsf{Authenticate}$ that is run by the authenticator and outputs whether the signature is authentic (i.e. a real signature that is the output of algorithm Sign) or not.

Our syntax assumes that signers maintain a transcript of their actions, which we denote trans. We define the transcript to contain all secrets and randomness generated by the signer during key generation and signing (cf. Example 1). We also introduce a fake transcript $\mathsf{trans}_{\mathsf{fake}}$ into our syntax. Formally, we define the syntax for an incoercible signature scheme in Definition 1.

Definition 1 (Incoercible Signature Scheme). *Let* trans *denote the transcript of a signer that contains all secrets and randomness generated by the signer during key generation and signing. Then, an* incoercible signature scheme $\mathsf{INC\text{-}SIG}$ *is a tuple of probabilistic polynomial time (PPT) algorithms* $(\mathsf{Setup}, \mathsf{AKGen}, \mathsf{SKGen}, \mathsf{FakeTrans}, \mathsf{Sign}, \mathsf{FakeSign}, \mathsf{Verify}, \mathsf{Authenticate})$ *such that:*

$\mathsf{Setup}(1^\lambda)$ On input of security parameter 1^λ, Setup outputs public parameters pp.

$\mathsf{AKGen}(pp)$ On input of public parameters pp, AKGen outputs an authenticator key pair (pk_A, sk_A) where pk_A is the authenticator's public key and sk_A is the authenticator's private key.

$\mathsf{SKGen}(pp, pk_A)$ On input of public parameters pp and authenticator public key pk_A, SKGen outputs a signer key pair (pk_S, sk_S) where pk_S is the signer's public key and sk_S is the signer's private key, and an initial transcript trans.

$\mathsf{FakeTrans}(pp, pk_A, \mathsf{trans}, \mathsf{trans}_{\mathsf{fake}})$ On input of public parameters pp, authenticator public key pk_A, a transcript trans and a fake transcript $\mathsf{trans}_{\mathsf{fake}}$, $\mathsf{FakeTrans}$ outputs an updated fake transcript $\mathsf{trans}_{\mathsf{fake}}$. We write that $\mathsf{trans}_{\mathsf{fake}} \leftarrow \mathsf{FakeTrans}(pp, pk_A, \mathsf{trans}, \perp)$ generates an initial fake transcript.

$\mathsf{Sign}(pp, sk_S, pk_A, m, \mathsf{trans})$ On input of public parameters pp, signer secret key sk_S, authenticator public key pk_A, message m and transcript trans, Sign outputs a signature σ and an updated transcript trans.

$\mathsf{FakeSign}(pp, sk_S, pk_A, m, \mathsf{trans}, \mathsf{trans}_{\mathsf{fake}})$ On input of public parameters pp, signer secret key sk_S, authenticator public key pk_A, message m, a transcript trans and a fake transcript $\mathsf{trans}_{\mathsf{fake}}$, $\mathsf{FakeSign}$ outputs a signature σ and an updated fake transcript $\mathsf{trans}_{\mathsf{fake}}$.

$\mathsf{Verify}(pp, pk_S, m, \sigma)$ On input of public parameters pp, signer public key pk_S, message m and signature σ, Verify outputs 1 if σ verifies, and 0 otherwise.

$\mathsf{Authenticate}(pp, pk_S, sk_A, m, \sigma)$ On input of public parameters pp, signer public key pk_S, authenticator secret key sk_A, message m and signature σ, $\mathsf{Authenticate}$ outputs 1 if σ is authentic, and 0 otherwise.

We require that an incoercible signature scheme satisfies *correctness*, which, as for standard signature schemes, requires that honestly generated signatures verify. Additionally, an incoercible signature scheme must satisfy *completeness*, the property that honestly generated signatures are authentic.

Definition 2 (Correctness). *An incoercible signature scheme* $\mathsf{INC\text{-}SIG}$ *satisfies* correctness *if, for any message* $m \in \{0,1\}^*$*, there exists a negligible function* negl *such that*

$$\Pr\left[\begin{array}{l} pp \leftarrow \mathsf{Setup}(1^\lambda); \\ (pk_A, sk_A) \leftarrow \mathsf{AKGen}(pp); \\ (pk_S, sk_S, \mathsf{trans}) \leftarrow \mathsf{SKGen}(pp, pk_A); \\ (\sigma, \mathsf{trans}) \leftarrow \mathsf{Sign}(pp, sk_S, pk_A, m, \mathsf{trans}) \end{array} : \mathsf{Verify}(pp, pk_S, m, \sigma) = 1\right] \geq 1 - \mathsf{negl}(\lambda).$$

Definition 3 (Completeness). *An incoercible signature scheme* INC-SIG *satisfies* completeness *if, for any message* $m \in \{0,1\}^*$*, there exists a negligible function* negl *such that*

$$\Pr\left[\begin{array}{l} pp \leftarrow \mathsf{Setup}(1^\lambda); \\ (pk_A, sk_A) \leftarrow \mathsf{AKGen}(pp); \\ (pk_S, sk_S, \mathsf{trans}) \leftarrow \mathsf{SKGen}(pp, pk_A); \\ (\sigma, \mathsf{trans}) \leftarrow \mathsf{Sign}(pp, sk_S, pk_A, m, \mathsf{trans}) \end{array} : \mathsf{Authenticate}(pp, pk_S, sk_A, m, \sigma) = 1\right] \geq 1 - \mathsf{negl}(\lambda).$$

2.1 Security Model

We present a security model for our syntax capturing a standard definition of existential unforgeability against a chosen message attack (EUF-CMA) [17], adapted to the setting of incoercible signature schemes. In our EUF-CMA experiment, we require that an adversary cannot output a valid signature on behalf of a signer, where the signature is not the output of the signing oracle, even if the adversary has access to the authenticator's key pair.

Definition 4 (Unforgeability). *An incoercible signature scheme* INC-SIG *satisfies* existential unforgeability against a chosen message attack (EUF-CMA) *if there exists a negligible function* negl *such that*

$$\Pr\left[\begin{array}{l} \mathsf{Query} \leftarrow \emptyset; \\ pp \leftarrow \mathsf{Setup}(1^\lambda); \\ (pk_A, sk_A) \leftarrow \mathsf{AKGen}(pp); \\ (pk_S, sk_S, \mathsf{trans}) \leftarrow \mathsf{SKGen}(pp, pk_A); \\ (m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{SIGN}(\cdot)}(pp, pk_S, pk_A, sk_A) \end{array} : \begin{array}{l} \mathsf{Verify}(pp, pk_S, m^*, \sigma^*) = 1 \\ \wedge\ m^* \notin \mathsf{Query} \end{array} \leq \mathsf{negl}(\lambda)\right] \leq \mathsf{negl}(\lambda)$$

where oracle $\mathsf{SIGN}(m)$*, on input message* m*, updates set* Query *to include message* m*, computes* $(\sigma, \mathsf{trans}) \leftarrow \mathsf{Sign}(pp, sk_S, pk_A, m, \mathsf{trans})$ *and outputs* σ*.*

We now present definitions of security properties that are intended to capture the fact that a signer, when coerced to sign a message by an attacker, can evade coercion, even if the signer is compelled to reveal their entire transcript to the attacker. In particular, we define two variants of receipt-freeness: weak and strong receipt-freeness. We complete our security model with a definition of coercion-resistance for incoercible signature schemes. Our definitions capture adaptive corruption and coercion strategies. We define the oracles for our experiments in Figure 2 and our experiments in Figure 3.

Oracles. Our formal security definitions rely on a number of oracles that we define formally in Figure 2. We write $\mathsf{X}_{(y_1,\ldots,y_n)}(z_1,\ldots,z_n)$ to denote oracle X that has access to parameters and lists $y_1,\ldots,y_n$ and takes as input $z_1,\ldots,z_n$. Oracle ADDU models key generation and the creation of an initial fake transcript for signers, outputting the signer's public key to the adversary. Oracle SIGN outputs an honestly generated signature and updates the fake transcript of the signer. We define oracle SIGN to take as input *any* signer (recall from the introduction that signers can be honest, coerced or corrupt) that has been previously input to oracle ADDU. Accordingly, our security model captures the fact that coerced signers, in addition to following instructions provided by the attacker, may produce honestly generated signatures for messages of their choosing. The adversary may call oracle CRPT to adaptively corrupt a signer, on the condition that the signer is not coerced, and obtain the secret key and transcript of the signer. Oracles CRCSIG, COERCE, FAKECOERCE and FAKESIGN adaptively coerce signers that are not previously corrupt and perform further functions on behalf of coerced signers. More specifically, oracle CRCSIG, depending on a bit b chosen in the security experiment, returns a real or fake signature for a message provided by the adversary. Similarly, oracle STCRCSIG returns a real or fake signature, and, allows the adversary to provide the randomness used to sign the message as input to the oracle. Oracle COERCE returns a real or fake transcript, depending on the bit b. Oracle FAKECOERCE reveals the fake transcript of a coerced signer. Finally, coerced signers can be input to FAKESIGN to obtain fake signatures. In our oracles and the corresponding experiments, we write $\mathbf{pk_S}$ to be the vector of signer public keys and $\mathbf{pk_S}[id]$ as the public key pk_S of signer id. We define the secret keys, transcripts and fake transcripts of signers analogously.

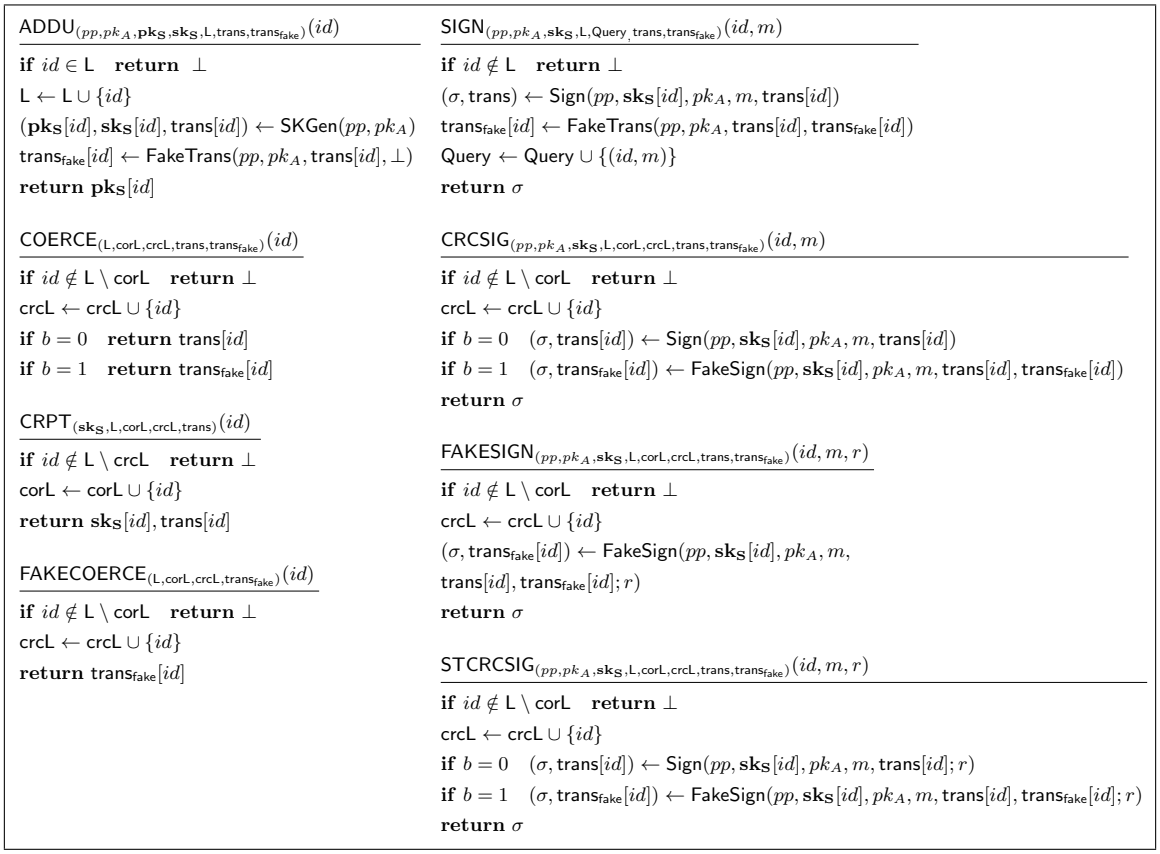

Fig. 2: Oracles used in the security experiments defined in Figure 3.

Weak Receipt-Freeness Broadly, weak receipt-freeness captures an attacker that coerces a signer to sign messages and, afterwards, demands evidence of the signer's cooperation. In our syntax, evidence refers to the signer's transcript. We do not consider that a weak receipt-freeness attacker may attempt to generate signatures on behalf of a coerced signer. Therefore, we require only a basic soundness requirement. Our definition of weak receipt-freeness captures two properties: IND1 indistinguishability and soundness.

IND1 indistinguishability defines an adversary that can request signatures from a coerced signer via oracle CRCSIG. Depending on a bit b chosen in the experiment, CRCSIG models a coerced signer that runs algorithm Sign or FakeSign to produce a signature. At the end of the experiment, the adversary can request transcripts of coerced signers via oracle COERCE, which returns a real or fake transcript, depending on the bit b. Additionally, the adversary can corrupt signers and request honestly generated signatures on behalf of any signer via oracles CRPT and SIGN respectively. We say that an incoercible signature scheme satisfies the indistinguishability requirement if the adversary can guess b with probability only negligibly more than $1/2$.

Soundness is a basic requirement requiring that, if a signature is the output of algorithm FakeSign, then algorithm Authenticate will output 1 with negligible probability.

Definition 5 (Weak Receipt-Freeness). *An incoercible signature scheme* INC-SIG *satisfies* weak receipt-freeness *if the following conditions hold.*

- ***Indistinguishability*** (IND1)*: for any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function* negl *such that*

$$\left| \Pr\left[\mathsf{Exp}^{\mathsf{IND1},0}_{\mathcal{A},\mathsf{INC\text{-}SIG}}(\lambda) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{IND1},1}_{\mathcal{A},\mathsf{INC\text{-}SIG}}(\lambda) = 1\right] \right| \leq \mathsf{negl}(\lambda).$$

- ***Soundness****: for any message $m \in \{0,1\}^*$, there exists a negligible function* negl *such that*

$$\Pr\left[\mathsf{Exp}^{\mathsf{sound}}_{\mathsf{INC\text{-}SIG}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda).$$

where $\mathsf{Exp}^{\mathsf{IND1},b}_{\mathcal{A},\mathsf{INC\text{-}SIG}}(\lambda)$ *for* $b \in \{0,1\}$ *and* $\mathsf{Exp}^{\mathsf{sound}}_{\mathsf{INC\text{-}SIG}}(\lambda)$ *are the experiments defined in Figure 3.*

$\mathsf{Exp}^{\mathsf{IND1},b}_{\mathcal{A},\mathsf{INC\text{-}SIG}}(\lambda)$

$\mathbf{pk_S}, \mathbf{sk_S}, \mathsf{trans}, \mathsf{trans}_{\mathsf{fake}} \leftarrow ()$
$\mathsf{L}, \mathsf{crcL}, \mathsf{corL}, \mathsf{Query} \leftarrow \emptyset$
$pp \leftarrow \mathsf{Setup}(1^\lambda)$
$(pk_A, sk_A) \leftarrow \mathsf{AKGen}(pp)$
$st \leftarrow \mathcal{A}_1^{\mathsf{ADDU},\mathsf{CRPT},\mathsf{SIGN},\mathsf{CRCSIG}}(pp, pk_A)$
$b' \leftarrow \mathcal{A}_2^{\mathsf{COERCE}}(st)$
return b'

$\mathsf{Exp}^{\mathsf{IND2},b}_{\mathcal{A},\mathsf{INC\text{-}SIG}}(\lambda)$

$\mathbf{pk_S}, \mathbf{sk_S}, \mathsf{trans}, \mathsf{trans}_{\mathsf{fake}} \leftarrow ()$
$\mathsf{L}, \mathsf{crcL}, \mathsf{corL}, \mathsf{Query} \leftarrow \emptyset$
$pp \leftarrow \mathsf{Setup}(1^\lambda)$
$(pk_A, sk_A) \leftarrow \mathsf{AKGen}(pp)$
$b' \leftarrow \mathcal{A}^{\mathsf{ADDU},\mathsf{CRPT},\mathsf{COERCE},\mathsf{SIGN},\mathsf{CRCSIG}}(pp, pk_A)$
return b'

$\mathsf{Exp}^{\mathsf{IND3},b}_{\mathcal{A},\mathsf{INC\text{-}SIG}}(\lambda)$

$\mathbf{pk_S}, \mathbf{sk_S}, \mathsf{trans}, \mathsf{trans}_{\mathsf{fake}} \leftarrow ()$
$\mathsf{L}, \mathsf{crcL}, \mathsf{corL}, \mathsf{Query} \leftarrow \emptyset$
$pp \leftarrow \mathsf{Setup}(1^\lambda)$
$(pk_A, sk_A) \leftarrow \mathsf{AKGen}(pp)$
$b' \leftarrow \mathcal{A}^{\mathsf{ADDU},\mathsf{CRPT},\mathsf{COERCE},\mathsf{SIGN},\mathsf{STCRCSIG}}(pp, pk_A)$
return b'

$\mathsf{Exp}^{\mathsf{sound}}_{\mathsf{INC\text{-}SIG}}(\lambda)$

$pp \leftarrow \mathsf{Setup}(1^\lambda)$
$(pk_A, sk_A) \leftarrow \mathsf{AKGen}(pp)$
$(pk_S, sk_S, \mathsf{trans}) \leftarrow \mathsf{SKGen}(pp, pk_A)$
$\mathsf{trans}_{\mathsf{fake}} \leftarrow \mathsf{FakeTrans}(pp, pk_A, \mathsf{trans}, \perp)$
$(\sigma, \mathsf{trans}_{\mathsf{fake}}) \leftarrow \mathsf{FakeSign}(pp, sk_S, pk_A, m, \mathsf{trans}, \mathsf{trans}_{\mathsf{fake}})$
$b \leftarrow \mathsf{Authenticate}(pp, pk_S, sk_A, m, \sigma)$
return b

$\mathsf{Exp}^{\mathsf{st\text{-}sound}}_{\mathcal{A},\mathsf{INC\text{-}SIG}}(\lambda)$

$\mathbf{pk_S}, \mathbf{sk_S}, \mathsf{trans}, \mathsf{trans}_{\mathsf{fake}} \leftarrow ()$
$\mathsf{L}, \mathsf{crcL}, \mathsf{corL}, \mathsf{Query} \leftarrow \emptyset$
$pp \leftarrow \mathsf{Setup}(1^\lambda)$
$(pk_A, sk_A) \leftarrow \mathsf{AKGen}(pp)$
$(id^*, m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{ADDU},\mathsf{CRPT},\mathsf{FAKECOERCE},\mathsf{SIGN},\mathsf{FAKESIGN}}(pp, pk_A)$
if $id^* \in \mathsf{crcL} \wedge (id^*, m^*) \notin \mathsf{Query} \wedge \ \mathsf{Authenticate}(pp, \mathbf{pk_S}[id^*], sk_A, m^*, \sigma^*) = 1$
  return 1
else **return** 0

Fig. 3: Experiments for weak receipt-freeness, strong receipt-freeness and coercion-resistance where the adversary has access to oracles defined in Figure 2.

Strong Receipt-Freeness A strong receipt-freeness attacker can interact with signers throughout the protocol. Crucially, this means that a strong receipt-freeness attacker can demand the transcripts of coerced signers at any point. As such, our definition of strong receipt-freeness must capture IND2 indistinguishability. Additionally, strong receipt-freeness requires soundness, as defined for weak receipt-freeness, as we assume that attacker does not attempt to generate signatures on behalf of a coerced signer.

Our IND2 experiment is similar to IND1 with one key difference: in our IND2 indistinguishability experiment, the adversary can query oracle COERCE throughout the experiment. This models the fact that the attacker, rather than requesting transcripts of a coerced signer at the end of a protocol run, may demand the transcripts at any point. With respect to all other oracles queries, the IND2 indistinguishability adversary is identical to the IND1 indistinguishability adversary. We require that the adversary cannot determine whether the coerced signers are cooperating or evading coercion. Formally, as in our IND1 indistinguishability experiment, this means that the adversary can guess the bit b with probability only negligibly more than $1/2$.

Definition 6 (Strong Receipt-Freeness). *An incoercible signature scheme* INC-SIG *satisfies* strong receipt-freeness *if the following conditions hold.*

- ***Indistinguishability*** (IND2)*: for any PPT adversary* $\mathcal{A}$*, there exists a negligible function* negl *such that*
$$\left| \Pr\left[\mathsf{Exp}^{\mathsf{IND2},0}_{\mathcal{A},\mathsf{INC\text{-}SIG}}(\lambda) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{IND2},1}_{\mathcal{A},\mathsf{INC\text{-}SIG}}(\lambda) = 1\right] \right| \leq \mathsf{negl}(\lambda).$$
- ***Soundness****: for any message* $m \in \{0,1\}^*$*, there exists a negligible function* negl *such that*
$$\Pr\left[\mathsf{Exp}^{\mathsf{sound}}_{\mathsf{INC\text{-}SIG}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda).$$

where $\mathsf{Exp}^{\mathsf{IND2},b}_{\mathcal{A},\mathsf{INC\text{-}SIG}}(\lambda)$ *for* $b \in \{0,1\}$ *and* $\mathsf{Exp}^{\mathsf{sound}}_{\mathsf{INC\text{-}SIG}}(\lambda)$ *are the experiments defined in Figure 3.*

Coercion-Resistance A coercion-resistance attacker controls coerced signers and can interact with signers throughout the protocol. Consequently, a coercion-resistant attacker can demand the transcripts of coerced signers at any point, and can attempt to produce signatures on their behalf. Our coercion-resistance definition captures IND3 indistinguishability and strong soundness.

Our IND3 experiment is identical to IND2 with the following exception. Rather than providing the adversary with access to oracle CRCSIG, the adversary can query oracle STCRCSIG, which allows the adversary to provide the randomness used to sign the message in addition to the message itself.

Strong soundness is defined with respect to an adversary that can request fake signatures and fake transcripts on behalf of coerced signers via oracles FAKESIGN and FAKECOERCE respectively. The adversary can also query oracle CRPT to corrupt signers, obtaining their transcripts and secret keys. Moreover, the adversary can request honestly generated signatures for any signer by calling oracle SIGN. For strong soundness, we require that the adversary cannot output an authentic signature on behalf of a coerced signer, where the signature is not the output of the signing oracle.

Definition 7 (Coercion-Resistance). *An incoercible signature scheme* INC-SIG *satisfies* coercion-resistance *if the following conditions hold.*

- ***Indistinguishability*** (IND3)*: for any PPT adversary* $\mathcal{A}$*, there exists a negligible function* negl *such that*
$$\left|\Pr\left[\mathsf{Exp}^{\mathsf{IND3},0}_{\mathcal{A},\mathsf{INC\text{-}SIG}}(\lambda)=1\right]-\Pr\left[\mathsf{Exp}^{\mathsf{IND3},1}_{\mathcal{A},\mathsf{INC\text{-}SIG}}(\lambda)=1\right]\right| \leq \mathsf{negl}(\lambda).$$
- ***Strong soundness****: for any PPT adversary* $\mathcal{A}$*, there exists a negligible function* negl *such that*
$$\Pr\left[\mathsf{Exp}^{\mathsf{st\text{-}sound}}_{\mathcal{A},\mathsf{INC\text{-}SIG}}(\lambda)=1\right] \leq \mathsf{negl}(\lambda).$$

where $\mathsf{Exp}^{\mathsf{IND3},b}_{\mathcal{A},\mathsf{INC\text{-}SIG}}(\lambda)$ *for* $b \in \{0,1\}$ *and* $\mathsf{Exp}^{\mathsf{st\text{-}sound}}_{\mathcal{A},\mathsf{INC\text{-}SIG}}(\lambda)$ *are the experiments defined in Figure 3.*

2.2 On Coercion of Signers During Key Generation

Our security model for incoercible signatures incorporates a hierarchy of incoercibility properties that are influenced and inspired by the existing literature [7, 9, 26, 37]. In particular, it captures the weakest variant of incoercibility that is described in the literature. We now show that our notion of coercion-resistance is the strongest possible form of attainable incoercibility for digital signatures.

Recall that coercion-resistance captures an attacker that can view the transcript of a coerced signer at any point *after* key generation and act on their behalf thereafter. We assume that key generation is always performed honestly, regardless of whether the signer is corrupt, coerced, or honest. We now demonstrate the following result: a definition of incoercibility in which the adversary can coerce and interact with signers during key generation is not satisfiable in the public-key setting.

We consider a natural extension to coercion-resistance, that we call *strong* coercion-resistance, in which the attacker can interact with the signer during key generation. In such a model, we require a new algorithm FakeSKGen, defined as follows.

FakeSKGen(pp, pk_A, r) On input of public parameters pp, authenticator public key pk_A and randomness r provided by the attacker, FakeSKGen outputs a signer key pair (pk_S, sk_S), where pk_S is the signer's public key and sk_S is the signer's private key, and the fake transcript $\mathsf{trans}_{\mathsf{fake}}$.

We also require modifications to the ADDU oracle for both the IND3 indistinguishability and strong soundness experiments, which we detail in Figure 4. More specifically, algorithm FakeSKGen allows a signer to generate their own key pair, potentially by deviating from the honest key generation algorithm. In the indistinguishability experiment, oracle ADDU returns the output of real key generation algorithm SKGen or the output of algorithm FakeSKGen, depending on a bit b chosen in the experiment. In the strong soundness experiment, oracle ADDU returns the output of algorithm FakeSKGen for coerced signers. With the above modifications, our strong coercion-resistance definition captures an attacker that interacts with signers during key generation. We obtain the result in Lemma 1.

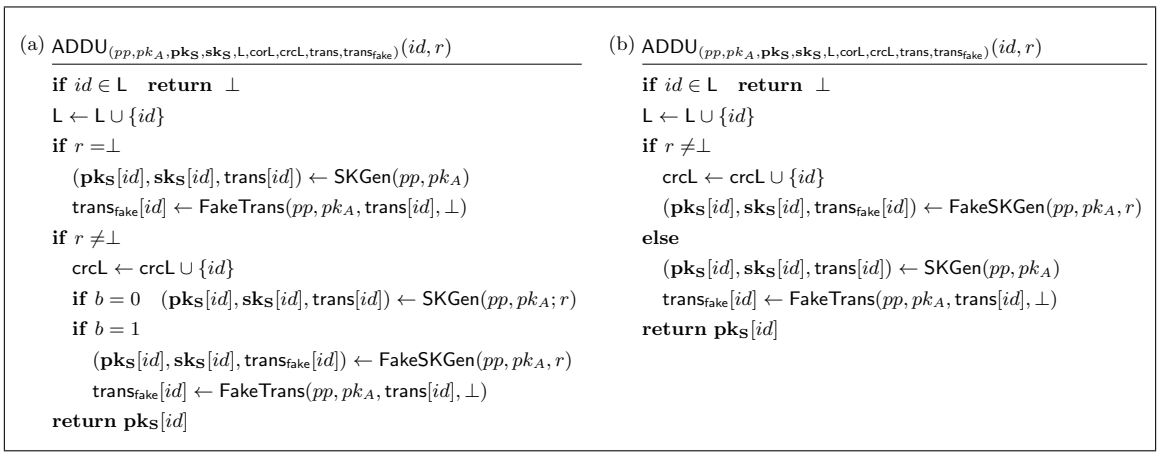

Fig. 4: The modified ADDU oracles for the (a) indistinguishability and (b) strong soundness experiments.

Lemma 1. *No construction for an incoercible signature scheme can satisfy* strong coercion-resistance.

Informally, this result holds because an adversary in the strong coercion-resistance experiments can always succeed. In particular, if the adversary can choose the randomness used in key generation, then they can use this randomness to generate their own public and secret key from honest key generation. The public key of the signer must match the public key held by the adversary, otherwise the adversary can break indistinguishability. Moreover, due to completeness, the attacker can use this key pair to construct authentic signatures in the strong soundness experiment. We now present a proof of Lemma 1.

Proof. First, we show that, if the public key output by $\mathsf{SKGen}(pp, pk_A; r)$ and $\mathsf{FakeSKGen}(pp, pk_A, r)$ are different with non–negligible probability ϵ, we can build an adversary $\mathcal{A}$ that wins in the indistinguishability game with non–negligible probability ϵ. $\mathcal{A}$ runs $\mathsf{SKGen}(pp, pk_A; r)$ and then simply guesses $b = 0$ if they receive the same public key and $b = 1$ if they do not receive the same public key.

Using the above, we now show that we can build an adversary $\mathcal{A}'$ that wins in the strong soundness game. Firstly, $\mathcal{A}$ selects some randomness r identically to in SKGen. They compute $(pk^*, sk^*, \mathsf{trans}) \leftarrow \mathsf{SKGen}(pp, pk_A; r)$. For any id^*, they input (id^*, r) to the ADDU oracle. They abort if the ADDU oracle outputs a different public key to pk^*, which occurs with negligible probability. Otherwise, they compute $(\sigma^*, \mathsf{trans}) \leftarrow \mathsf{Sign}(pp, sk^*, pk_A, m^*, \mathsf{trans})$ for any message m^*. Finally, they output (id^*, m^*, σ^*). Clearly, $id^* \in \mathsf{crcL}$ and the signing oracle has not been used. As the key generation and signing were performed honestly, $\mathsf{Authenticate}(pp, pk^*, sk_A, m^*, \sigma^*) = 0$ with negligible probability due to completeness. As pk^* is identical to $\mathbf{pk_S}[id^*]$, the adversary $\mathcal{A}'$ wins with non–negligible probability.

We note that, if the signer obtains a secret input from the authenticator during key generation, the attacker is unable to choose all of the randomness for key generation. Consequently, the result in Lemma 1 can be overcome. This was indeed the approach taken by [16]. In this paper, we choose to focus on a non-interactive setting and therefore avoid the requirement of some secret input from the authenticator, which might be difficult to implement in practice.

3 Constructions

In this section, we provide two constructions of incoercible signature schemes. Our constructions employ a standard signature scheme and a sender-deniable encryption scheme, which, throughout this work, we refer to as a deniable encryption scheme for brevity. We recall the syntax and security models for these building blocks next.

3.1 Building Blocks

Signature Schemes We require a standard signature scheme SIG, which satisfies standard notions of correctness and unforgeability ($\mathsf{EUF\text{-}CMA}$).

Definition 8 (Signature Scheme). *A* signature scheme SIG *is a tuple of PPT algorithms* $(\mathsf{SIG.Setup}, \mathsf{SIG.KGen}, \mathsf{SIG.Sign}, \mathsf{SIG.Vf})$ *such that:*

$\mathsf{SIG.Setup}(1^\lambda)$ On input of security parameter 1^λ, $\mathsf{SIG.Setup}$ outputs public parameters pp_{SIG}.
$\mathsf{SIG.KGen}(pp_{\mathsf{SIG}})$ On input of public parameters pp_{SIG}, $\mathsf{SIG.KGen}$ outputs a key pair $(pk_{\mathsf{SIG}}, sk_{\mathsf{SIG}})$ where pk_{SIG} is the public verification key and sk_{SIG} is the private signing key.
$\mathsf{SIG.Sign}(pp_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, m)$ On input of public parameters pp_{SIG}, signing key sk_{SIG} and message m, $\mathsf{SIG.Sign}$ outputs a signature σ.
$\mathsf{SIG.Vf}(pp_{\mathsf{SIG}}, pk_{\mathsf{SIG}}, m, \sigma)$ On input of public parameters pp_{SIG}, verification key pk_{SIG}, message m and signature σ, $\mathsf{SIG.Vf}$ outputs 1 if σ verifies, and 0 otherwise.

Definition 9 (Correctness). *A signature scheme* SIG *satisfies* correctness *if, for any message* $m \in \{0,1\}^*$*, there exists a negligible function* negl *such that*

$$\Pr\left[\begin{array}{l} pp_{\mathsf{SIG}} \leftarrow \mathsf{SIG.Setup}(1^\lambda); \\ (pk_{\mathsf{SIG}}, sk_{\mathsf{SIG}}) \leftarrow \mathsf{SIG.KGen}(pp_{\mathsf{SIG}}); \\ \sigma \leftarrow \mathsf{SIG.Sign}(pp_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, m) \end{array} : \mathsf{SIG.Vf}(pp_{\mathsf{SIG}}, pk_{\mathsf{SIG}}, m, \sigma) = 1\right] \geq 1 - \mathsf{negl}(\lambda).$$

Definition 10 ($\mathsf{EUF\text{-}CMA}$)**.** *A signature scheme* SIG *satisfies* existential unforgeability against a chosen message attack ($\mathsf{EUF\text{-}CMA}$) *if there exists a negligible function* negl *such that*

$$\Pr\left[\begin{array}{l} \mathsf{Query} \leftarrow \emptyset; \\ pp_{\mathsf{SIG}} \leftarrow \mathsf{SIG.Setup}(1^\lambda); \\ (pk_{\mathsf{SIG}}, sk_{\mathsf{SIG}}) \leftarrow \mathsf{SIG.KGen}(pp_{\mathsf{SIG}}); \\ (m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{SIGN}(\cdot)}(pp_{\mathsf{SIG}}, pk_{\mathsf{SIG}}) \end{array} : \begin{array}{c} \mathsf{SIG.Vf}(pp_{\mathsf{SIG}}, pk_{\mathsf{SIG}}, m^*, \sigma^*) = 1 \\ \wedge \ m^* \notin \mathsf{Query} \end{array}\right] \leq \mathsf{negl}(\lambda)$$

where oracle $\mathsf{SIGN}(m)$*, on input message* m*, outputs* $\sigma \leftarrow \mathsf{SIG.Sign}(pp_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, m)$ *and updates set* Query *to include message* m*.*

Deniable Encryption Schemes A deniable encryption scheme DEN is a standard public-key encryption scheme, equipped with an additional algorithm $\mathsf{DEN.Exp}$ that, for a ciphertext c that encrypts message m and is output by encryption algorithm $\mathsf{DEN.Enc}$, generates randomness such that c appears to encrypt an alternative message m'. We require that the deniable encryption scheme satisfies correctness and indistinguishability under chosen plaintext attacks ($\mathsf{IND\text{-}CPA}$), as in a traditional encryption scheme, and indistinguishability of explanations ($\mathsf{IND\text{-}EXP}$), which ensures that the randomness output by algorithm $\mathsf{DEN.Exp}$ is indistinguishable from the real randomness used by algorithm $\mathsf{DEN.Enc}$.

Definition 11 (Deniable Encryption Scheme). *A deniable encryption scheme* (DEN) *is a tuple of PPT algorithms* $(\mathsf{DEN.Setup}, \mathsf{DEN.KGen}, \mathsf{DEN.Enc}, \mathsf{DEN.Dec}, \mathsf{DEN.Exp})$ *such that:*

$\mathsf{DEN.Setup}(1^\lambda)$ On input of security parameter 1^λ, $\mathsf{DEN.Setup}$ outputs public parameters pp_{DEN}.
$\mathsf{DEN.KGen}(pp_{\mathsf{DEN}})$ On input of public parameters pp_{DEN}, $\mathsf{DEN.KGen}$ outputs a key pair $(pk_{\mathsf{DEN}}, sk_{\mathsf{DEN}})$ where pk_{DEN} is the public encryption key and sk_{DEN} is the private decryption key.
$\mathsf{DEN.Enc}(pp_{\mathsf{DEN}}, pk_{\mathsf{DEN}}, m)$ On input of public parameters pp_{DEN}, public key pk_{DEN} and message m, $\mathsf{DEN.Enc}$ outputs a ciphertext c.
$\mathsf{DEN.Dec}(pp_{\mathsf{DEN}}, sk_{\mathsf{DEN}}, c)$ On input of public parameters pp_{DEN}, private key sk_{DEN} and ciphertext c, $\mathsf{DEN.Dec}$ outputs a message m.
$\mathsf{DEN.Exp}(pp_{\mathsf{DEN}}, pk_{\mathsf{DEN}}, c, m)$ On input of public parameters pp_{DEN}, public key pk_{DEN}, ciphertext c and message m, $\mathsf{DEN.Exp}$ outputs a string u.

Definition 12 (Correctness). *A deniable encryption scheme* DEN *satisfies* correctness *if, for any message* $m \in \{0,1\}^*$*, there exists a negligible function* negl *such that*

$$\Pr\left[\begin{array}{l} pp_{\mathsf{DEN}} \leftarrow \mathsf{DEN.Setup}(1^\lambda); \\ (pk_{\mathsf{DEN}}, sk_{\mathsf{DEN}}) \leftarrow \mathsf{DEN.KGen}(pp_{\mathsf{DEN}}); \\ c \leftarrow \mathsf{DEN.Enc}(pp_{\mathsf{DEN}}, pk_{\mathsf{DEN}}, m) \end{array} : \mathsf{DEN.Dec}(pp_{\mathsf{DEN}}, sk_{\mathsf{DEN}}, c) = m\right] \geq 1 - \mathsf{negl}(\lambda).$$

Definition 13 (IND-CPA). *A deniable encryption scheme* DEN *satisfies* indistinguishability under a chosen plaintext attack (IND-CPA) *if, for any PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, there exists a negligible function* negl *such that*

$$\left| \Pr\left[\mathsf{Exp}^{\mathsf{IND\text{-}CPA},0}_{\mathcal{A},\mathsf{DEN}}(\lambda) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{IND\text{-}CPA},1}_{\mathcal{A},\mathsf{DEN}}(\lambda) = 1\right] \right| \leq \mathsf{negl}(\lambda)$$

where $\mathsf{Exp}^{\mathsf{IND\text{-}CPA},b}_{\mathcal{A},\mathsf{DEN}}(\lambda)$ *is the experiment defined in Figure 5 for* $b \in \{0,1\}$*.*

Definition 14 (IND-EXP). *A deniable encryption scheme* DEN *satisfies* indistinguishability of explanation (IND-EXP) *if, for any PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, there exists a negligible function* negl *such that*

$$\left| \Pr\left[\mathsf{Exp}^{\mathsf{IND\text{-}EXP},0}_{\mathcal{A},\mathsf{DEN}}(\lambda) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{IND\text{-}EXP},1}_{\mathcal{A},\mathsf{DEN}}(\lambda) = 1\right] \right| \leq \mathsf{negl}(\lambda)$$

where $\mathsf{Exp}^{\mathsf{IND\text{-}EXP},b}_{\mathcal{A},\mathsf{DEN}}(\lambda)$ *is the experiment defined in Figure 5 for* $b \in \{0,1\}$*.*

$\mathsf{Exp}^{\mathsf{IND\text{-}CPA},b}_{\mathcal{A},\mathsf{DEN}}(\lambda)$	$\mathsf{Exp}^{\mathsf{IND\text{-}EXP},b}_{\mathcal{A},\mathsf{DEN}}(\lambda)$
$pp_{\mathsf{DEN}} \leftarrow \mathsf{DEN.Setup}(1^\lambda)$	$pp_{\mathsf{DEN}} \leftarrow \mathsf{DEN.Setup}(1^\lambda)$
$(pk_{\mathsf{DEN}}, sk_{\mathsf{DEN}}) \leftarrow \mathsf{DEN.KGen}(pp_{\mathsf{DEN}})$	$(pk_{\mathsf{DEN}}, sk_{\mathsf{DEN}}) \leftarrow \mathsf{DEN.KGen}(pp_{\mathsf{DEN}})$
$(m_0, m_1, st) \leftarrow \mathcal{A}_1(pp_{\mathsf{DEN}}, pk_{\mathsf{DEN}})$	$(m, st) \leftarrow \mathcal{A}_1(pp_{\mathsf{DEN}}, pk_{\mathsf{DEN}})$
$c \leftarrow \mathsf{DEN.Enc}(pp_{\mathsf{DEN}}, pk_{\mathsf{DEN}}, m_b)$	$c \leftarrow \mathsf{DEN.Enc}(pp_{\mathsf{DEN}}, pk_{\mathsf{DEN}}, m; u_0)$;
$b' \leftarrow \mathcal{A}_2(c, st)$	$u_1 \leftarrow \mathsf{DEN.Exp}(pp_{\mathsf{DEN}}, pk_{\mathsf{DEN}}, c, m)$;
return b'	$b' \leftarrow \mathcal{A}_2(c, u_b, st)$
	return b'

Fig. 5: Experiments for indistinguishability under a chosen plaintext attack and indistinguishability of explanation for a deniable encryption scheme.

3.2 A Coercion-Resistant Construction

We first introduce a construction that we call CR.SIG that satisfies our strongest form of incoercibility: coercion-resistance. We present CR.SIG in Figure 6: it relies on a standard signature scheme SIG and a deniable encryption scheme DEN. Additionally, our construction uses two hash functions $\mathcal{H}_1 : \{0,1\}^* \to \{0,1\}^*$, $\mathcal{H}_2 : \{0,1\}^* \to \mathcal{M}$, where $\mathcal{M}$ is the message space of the signature scheme SIG. The first is assumed to model a random oracle, whereas the second is required to be collision resistant.

At a high level, our construction works as follows. The authenticator generates a key pair for a deniable encryption scheme. During key generation, in addition to generating a key pair for signature scheme SIG, a signer generates a random string s and deniably encrypts this under the authenticator's public key. The signer's secret key consists of a secret key for SIG and a string s. The corresponding public key consists of the public key for SIG and the ciphertext that encrypts s. A signature consists of a deniable encryption of s, as well as a standard signature that signs both the message and the deniable encryption ciphertext. The authenticator can detect coercion by decrypting the ciphertexts contained in the public key and the signature, and comparing the two, via the Authenticate algorithm. The signer creates a fake transcript that indicates s', rather than s, is contained in the signer's secret key. In this way, by security of the deniable encryption scheme, the attacker cannot distinguish a real and a fake transcript. Moreover, the coercive attacker cannot forge an authentic signature without knowledge of s, and our construction achieves strong soundness.

Our coercion-resistance construction satisfies correctness, completeness, unforgeability and coercion-resistance, as defined in Section 2. We obtain the formal result in Theorem 1.

$\mathsf{CR.Setup}(1^\lambda)$

$pp_{\mathsf{DEN}} \leftarrow \mathsf{DEN.Setup}(1^\lambda)$,
$pp_{\mathsf{SIG}} \leftarrow \mathsf{SIG.Setup}(1^\lambda)$,
return $pp = (pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}})$

$\mathsf{CR.AKGen}(pp)$

parse pp as $(pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}})$
$(pk_A, sk_A) \leftarrow \mathsf{DEN.KGen}(pp_{\mathsf{DEN}})$
return (pk_A, sk_A)

$\mathsf{CR.SKGen}(pp, pk_A)$

parse pp as $(pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}})$
$(pk_{\mathsf{SIG}}, sk_{\mathsf{SIG}}) \leftarrow \mathsf{SIG.KGen}(pp_{\mathsf{SIG}}; r_{\mathsf{SIG}})$
// where r_{SIG} is the randomness sampled in the
// SIG key generation algorithm
$s \leftarrow \{0,1\}^\lambda$
$c \leftarrow \mathsf{DEN.Enc}(pp_{\mathsf{DEN}}, pk_A, s; r_c)$
// where r_c is the randomness sampled in the
// DEN encryption algorithm
$s \leftarrow \{0,1\}^\lambda$
$(pk_S, sk_S) \leftarrow ((pk_{\mathsf{SIG}}, c), (sk_{\mathsf{SIG}}, s))$
$\mathsf{trans} \coloneqq \{(r_{\mathsf{SIG}}, pk_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, s, r_c, c)\}$
return $((pk_S, sk_S), \mathsf{trans})$

$\mathsf{CR.FakeTrans}(pp, pk_A, \mathsf{trans}, \perp)$

parse pp as $(pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}})$
parse trans as $\{(r_{\mathsf{SIG}}, pk_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, s, r_c, c)\}$
$s' \leftarrow \{0,1\}^\lambda$
$r'_c \leftarrow \mathsf{DEN.Exp}(pp_{\mathsf{DEN}}, pk_A, c, s')$
return $\mathsf{trans}_{\mathsf{fake}} \coloneqq \{(r_{\mathsf{SIG}}, pk_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, s', r'_c, c)\}$

$\mathsf{CR.Sign}(pp, sk_S, pk_A, m, \mathsf{trans})$

parse pp as $(pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}}), sk_S$ as (sk_{SIG}, s)
$\sigma_1 \leftarrow \mathsf{DEN.Enc}(pp_{\mathsf{DEN}}, pk_A, \mathcal{H}_1(m||s); r_{\sigma_1})$
$\sigma_2 \leftarrow \mathsf{SIG.Sign}(pp_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, \mathcal{H}_2(m||\sigma_1); r_{\sigma_2})$
$\mathsf{trans} \leftarrow \mathsf{trans} \cup \{(m, r_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2)\}$
return $((\sigma_1, \sigma_2), \mathsf{trans})$

$\mathsf{CR.FakeSign}(pp, sk_S, pk_A, m, \mathsf{trans}, \mathsf{trans}_{\mathsf{fake}})$

parse pp as $(pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}}), sk_S$ as (sk_{SIG}, s)
parse $\mathsf{trans}_{\mathsf{fake}}$ as $\{(r_{\mathsf{SIG}}, pk_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, s', r'_c, c), \dots\}$
$\sigma_1 \leftarrow \mathsf{DEN.Enc}(pp_{\mathsf{DEN}}, pk_A, \mathcal{H}_1(m||s'); r_{\sigma_1})$,
$\sigma_2 \leftarrow \mathsf{SIG.Sign}(pp_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, \mathcal{H}_2(m||\sigma_1); r_{\sigma_2})$
$\mathsf{trans}_{\mathsf{fake}} \leftarrow \mathsf{trans}_{\mathsf{fake}} \cup \{(m, r_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2)\}$
return $(\sigma = (\sigma_1, \sigma_2), \mathsf{trans}_{\mathsf{fake}})$

$\mathsf{CR.Authenticate}(pp, pk_S, sk_A, m, \sigma)$

parse pp as $(pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}}), pk_S$ as $(pk_{\mathsf{SIG}}, c), \sigma$ as (σ_1, σ_2)
if $\mathsf{SIG.Vf}(pp_{\mathsf{SIG}}, pk_S, \mathcal{H}_2(m||\sigma_1), \sigma_2) = 0$ **return** 0
$H' \leftarrow \mathsf{DEN.Dec}(pp_{\mathsf{DEN}}, sk_A, \sigma_1)$
$sk' \leftarrow \mathsf{DEN.Dec}(pp_{\mathsf{DEN}}, sk_A, c)$,
if $H' = \mathcal{H}_1(m||sk')$ **return** 1 **else return** 0

$\mathsf{CR.Verify}(pp, pk_S, m, \sigma)$

parse pp as $(pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}}), pk_S$ as $(pk_{\mathsf{SIG}}, c), \sigma$ as (σ_1, σ_2)
if $\mathsf{SIG.Vf}(pp_{\mathsf{SIG}}, pk_{\mathsf{SIG}}, \mathcal{H}_2(m||\sigma_1), \sigma_2) = 0$ **return** 0
else return 1

$\mathsf{CR.FakeTrans}(pp, pk_A, \mathsf{trans}, \mathsf{trans}_{\mathsf{fake}})$

parse pp as $(pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}})$
parse trans as
$\{(r_{\mathsf{SIG}}, pk_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, s, r_c, c), \dots, (m, r_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2), \dots\}$
parse $\mathsf{trans}_{\mathsf{fake}}$ as $\{(r_{\mathsf{SIG}}, pk_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, s', r'_c, c), \dots\}$
For each new entry $(m, r_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2)$ from CR.Sign
$r'_{\sigma_1} \leftarrow \mathsf{DEN.Exp}(pp_{\mathsf{DEN}}, pk_A, \sigma_1, \mathcal{H}_1(m||s'))$
$\mathsf{trans}_{\mathsf{fake}} \leftarrow \mathsf{trans}_{\mathsf{fake}} \cup \{(m, r'_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2)\}$
return $\mathsf{trans}_{\mathsf{fake}}$

Fig. 6: Our coercion-resistant construction CR.SIG.

Theorem 1. *Let* SIG *and* DEN *be a secure signature scheme and deniable encryption scheme respectively, and the hash functions $\mathcal{H}_1$ and $\mathcal{H}_2$ be modelled as a random oracle model and collision resistant respectively. Then,* CR.SIG *is a secure construction of a coercion-resistant incoercible signature scheme. That is,* CR.SIG *safisfies correctness, completeness, unforgeability and coercion-resistance.*

We prove Theorem 1 via a series of Lemmata.

Lemma 2 (Correctness). CR.SIG *satisfies* correctness *if signature scheme* SIG *is correct.*

Proof. Consider a signature $\sigma = (\sigma_1, \sigma_2)$ output by algorithm CR.Sign with respect to message m for keys sk_S and pk_A generated by algorithms CR.SKGen and CR.AKGen respectively and parameters pp output by algorithm CR.Setup. Then, by definition of correctness, CR.SIG is correct

if $\mathsf{CR.Verify}(pp, pk_S, m, \sigma)$, outputs 1 with overwhelming probability. Assume that CR.Verify does not return 1. Then it must be the case that $\mathsf{SIG.Vf}(pp_{\mathsf{SIG}}, pk_{\mathsf{SIG}}, \mathcal{H}_2(m||\sigma_1), \sigma_2) = 0$. By assumption, signature scheme SIG is correct. Therefore, algorithm SIG.Vf returns 1 with overwhelming probability where pp_{SIG} and pk_{SIG} are generated according to the construction description and σ_2 is the output of SIG.Sign for message $\mathcal{H}_2(m||\sigma_1)$. Then, by contradiction, CR.SIG is correct.

Lemma 3 (Completeness). CR.SIG *satisfies* completeness *if deniable encryption scheme* DEN *is correct and* CR.SIG *is correct.*

Proof. Consider a signature $\sigma = (\sigma_1, \sigma_2)$ output by algorithm CR.Sign with respect to message m for keys sk_S and pk_A generated by algorithms CR.SKGen and CR.AKGen respectively and parameters pp output by algorithm CR.Setup. By definition of completeness, CR.SIG is complete if $\mathsf{CR.Authenticate}(pp, pk_S, sk_A, m, (\sigma_1, \sigma_2))$ outputs 1 with overwhelming probability. Assume that CR.Authenticate does not return 1. Then, either $\mathsf{SIG.Vf}(pp_{\mathsf{SIG}}, pk_{\mathsf{SIG}}, \mathcal{H}_2(m||\sigma_1), \sigma_2) = 0$ or $\mathsf{DEN.Dec}(pp_{\mathsf{DEN}}, sk_A, \sigma_1) \neq \mathcal{H}_1(m||\mathsf{DEN.Dec}(pp_{\mathsf{DEN}}, sk_A, c))$. By correctness of CR.SIG, SIG.Vf returns 1 with overwhelming probability. Moreover, by correctness of the deniable encryption scheme, $\mathsf{DEN.Dec}(pp_{\mathsf{DEN}}, sk_A, c)$ returns s, and $\mathsf{DEN.Dec}(pp_{\mathsf{DEN}}, sk_A, \sigma_1)$ returns $\mathcal{H}_1(m||s)$. Then, by contradiction, CR.SIG is complete.

Lemma 4 (Unforgeability). CR.SIG *satisfies* existential unforgeability against a chosen message attack *(*EUF-CMA*) if digital signature scheme* SIG *satisfies* EUF-CMA *security and the hash function* $\mathcal{H}_2$ *is collision resistant.*

Proof. Unforgeability follows from the EUF-CMA security of SIG and the fact that hash function $\mathcal{H}_2$ is collision resistant. Indeed, our proof of unforgeability demonstrates that, if we assume that CR.SIG does not satisfy unforgeability and $\mathcal{H}_2$ is collision resistant, then it is possible to construct an adversary that succeeds in breaking the EUF-CMA security of SIG. Then, by contradiction, the result holds.

Let $\mathcal{A}$ be an adversary that succeeds in the EUF-CMA experiment (Definition 4) for the CR.SIG construction. We show that we can construct an adversary $\mathcal{A}'$ that succeeds in the EUF-CMA experiment (Definition 10) for digital signature scheme SIG. We present $\mathcal{A}'$ in Figure 7. It is clear that inputs to $\mathcal{A}$ are distributed identically to the EUF-CMA experiment of Definition 4 because keys pk_A and pk_S are generated identically. Moreover, oracle SIGN is distributed identically because $\mathcal{O}\mathsf{SIG}(\mathcal{H}_2(m||\sigma_1))$ returns $\mathsf{SIG.Sign}(pp_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, \mathcal{H}_2(m||\sigma_1))$. We now show that $\mathcal{A}'$ is successful in the EUF-CMA experiment of Definition 10. That is, we show that $(\mathcal{H}_2(m^*||\sigma_1^*), \sigma_2^*)$ is a valid message/signature pair that has not been input to $\mathcal{O}$SIG. By assumption $(m^*, (\sigma_1^*, \sigma_2^*))$ is a valid message/signature pair, i.e., $\mathsf{CR.Verify}(pp, pk_S, m^*, (\sigma_1^*, \sigma_2^*)) = 1$. Then, by definition, $\mathsf{SIG.Vf}(pp_{\mathsf{SIG}}, pk_{\mathsf{SIG}}, \mathcal{H}_2(m^*||\sigma_1^*), \sigma_2^*) = 1$. Moreover, by assumption, $m^* \notin \mathsf{Query}$ and, therefore, $\mathcal{H}_2(m^*||\sigma_1^*)$ is not input to $\mathcal{O}$SIG unless a hash collision occurs, which occurs with negligible probability. Therefore, $\mathcal{A}'$ succeeds in the EUF-CMA experiment and, by contradiction, the result holds.

$\mathsf{SIGN}(m)$	$\mathcal{A}'^{\mathcal{O}\mathsf{SIG}(\cdot)}(pp_{\mathsf{SIG}}, pk_{\mathsf{SIG}})$
$\sigma_1 \leftarrow \mathsf{DEN.Enc}(pp_{\mathsf{DEN}}, pk_A, \mathcal{H}_1(m\|\|s))$	$\mathsf{Query} \leftarrow \emptyset$
$\sigma_2 \leftarrow \mathcal{O}\mathsf{SIG}(\mathcal{H}_2(m\|\|\sigma_1))$	$pp_{\mathsf{DEN}} \leftarrow \mathsf{DEN.Setup}(1^\lambda)$
$\mathsf{Query} \leftarrow \mathsf{Query} \cup \{m\}$	$pp \leftarrow (pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}})$
return (σ_1, σ_2)	$(pk_A, sk_A) \leftarrow \mathsf{CR.AKGen}(pp)$
	$s \leftarrow \{0,1\}^\lambda, c \leftarrow \mathsf{DEN.Enc}(pp_{\mathsf{DEN}}, pk_A, s), pk_S \leftarrow (pk_{\mathsf{SIG}}, c)$
	$(m^*, (\sigma_1^*, \sigma_2^*)) \leftarrow \mathcal{A}^{\mathsf{SIGN}}(pp, pk_S, pk_A, sk_A)$
	return $(\mathcal{H}_2(m^*\|\|\sigma_1^*), \sigma_2^*)$

Fig. 7: Adversary $\mathcal{A}'$ that breaks the EUF-CMA security of signature scheme SIG.

Lemma 5 ((IND3) Indistinguishability). CR.SIG *satisfies* (IND3) *indistinguishability for a coercion-resistant incoercible signature scheme if the deniable encryption scheme satisfies* IND-CPA *security and* IND-EXP *security.*

Proof. To prove IND3 indistinguishability, we proceed through a series of game hops that we show are indistinguishable to the adversary. In the final game, the view of the adversary will be identical for $b = 0$ and $b = 1$. In Game 1, the experiment only generates a fake transcript once a signer is added to the list of coerced signers. As this change is superficial, i.e., the fake transcript is only required for coerced signers, this game hop is indistinguishable to the adversary. In Game 2, we use the real secret s in the fake transcript, rather than the fake secret s'. In Game 3, we also include the real encryption randomness in the fake transcript, instead of using DEN.Exp to produce fake randomness. The hops from Game 1 to Game 2 and Game 2 to Game 3 require a hybrid argument such that the reduction operates in k_1 steps, where k_1 is the number of queries made by the adversary to oracle ADDU. Ultimately, the hop from Game 1 to Game 2 is indistinguishable to the adversary due to the IND-CPA security of the deniable encryption scheme and the hop from Game 2 to Game 3 is indistinguishable due to the IND-EXP security of the deniable encryption scheme. Finally, in Game 3, the fake transcript is identical to the real transcript. Moreover, all signatures generated during the experiment are the output of the real signing algorithm. Therefore, the inputs to the adversary are independent of b and indistinguishability holds.

Let $\mathcal{A}$ be an adversary in the $\mathsf{Exp}^{\mathsf{IND3},b}_{\mathcal{A},\mathsf{CR.SIG}}(\lambda)$ experiment that makes at most k_1 queries to the ADDU oracle, at most k_2 queries to the SIGN oracle per signer, and at most k_3 queries to the STCRCSIG oracle per signer. We proceed through a series of game hops that we show are indistinguishable to the adversary. In the final game, the view of the adversary will be identical for $b = 0$ and $b = 1$ and the adversary will have a negligible chance of guessing b. We define Game 0 as the experiment $\mathsf{Exp}^{\mathsf{IND3},b}_{\mathcal{A},\mathsf{CR.SIG}}(\lambda)$ with b chosen randomly. Let S_i be the event that adversary $\mathcal{A}$ correctly guesses b after Game i.

Game 1 is identical to Game 0, except that the fake transcript is only generated when a signer is first added to the coerced list. We give the adjusted oracles for this game in Figure 8. As the fake transcript is only used after a signer has been coerced this is only a superficial change and does not affect the distribution of inputs to the adversary. Therefore, $Pr[S_0] = Pr[S_1]$.

$\mathsf{ADDU}_{(pp,pk_A,\mathbf{pk_S},\mathbf{sk_S},\mathsf{L},\mathsf{trans},\mathsf{trans_{fake}})}(id)$

if $id \in \mathsf{L}$ **return** $\perp$
$\mathsf{L} \leftarrow \mathsf{L} \cup \{id\}$
$(\mathbf{pk_S}[id], \mathbf{sk_S}[id], \mathsf{trans}[id]) \leftarrow \mathsf{CR.SKGen}(pp, pk_A)$
return $\mathbf{pk_S}[id]$

$\mathsf{STCRCSIG}_{(pp,pk_A,\mathbf{sk_S},\mathsf{L},\mathsf{corL},\mathsf{crcL},\mathsf{trans},\mathsf{trans_{fake}})}(id, m, r)$

if $id \notin \mathsf{L} \setminus \mathsf{corL}$ **return** $\perp$
if $id \notin \mathsf{crcL}$
  $\mathsf{crcL} \leftarrow \mathsf{crcL} \cup \{id\}$
  $\mathsf{trans_{fake}}[id] \leftarrow \mathsf{CR.FakeTrans}(pp, pk_A, \mathsf{trans}[id], \perp)$
  $\mathsf{trans_{fake}}[id] \leftarrow \mathsf{CR.FakeTrans}(pp, pk_A, \mathsf{trans}[id], \mathsf{trans_{fake}}[id])$
if $b = 0$
  $(\sigma, \mathsf{trans}[id]) \leftarrow \mathsf{CR.Sign}(pp, \mathbf{sk_S}[id], pk_A, m, \mathsf{trans}[id]; r)$
if $b = 1$
  $(\sigma, \mathsf{trans_{fake}}[id]) \leftarrow \mathsf{CR.FakeSign}(pp, \mathbf{sk_S}[id], pk_A, m, \mathsf{trans}[id], \mathsf{trans_{fake}}[id]; r)$
return σ

$\mathsf{COERCE}_{(\mathsf{L},\mathsf{corL},\mathsf{crcL},\mathsf{trans},\mathsf{trans_{fake}})}(id)$

if $id \notin \mathsf{L} \setminus \mathsf{corL}$ **return** $\perp$
if $id \notin \mathsf{crcL}$
  $\mathsf{crcL} \leftarrow \mathsf{crcL} \cup \{id\}$
  $\mathsf{trans_{fake}}[id] \leftarrow \mathsf{CR.FakeTrans}(pp, pk_A, \mathsf{trans}[id], \perp)$
  $\mathsf{trans_{fake}}[id] \leftarrow \mathsf{CR.FakeTrans}(pp, pk_A, \mathsf{trans}[id], \mathsf{trans_{fake}}[id])$
if $b = 0$ **return** $\mathsf{trans}[id]$
if $b = 1$ **return** $\mathsf{trans_{fake}}[id]$

$\mathsf{SIGN}_{(pp,pk_A,\mathbf{sk_S},\mathsf{L},\mathsf{crcL},\mathsf{Query},\mathsf{trans},\mathsf{trans_{fake}})}(id, m)$

if $id \notin \mathsf{L}$ **return** $\perp$
$(\sigma, \mathsf{trans}[id]) \leftarrow \mathsf{CR.Sign}(pp, \mathbf{sk_S}[id], pk_A, m, \mathsf{trans}[id])$
if $id \in \mathsf{crcL}$
  $\mathsf{trans_{fake}}[id] \leftarrow \mathsf{CR.FakeTrans}(pp, pk_A, \mathsf{trans}[id], \mathsf{trans_{fake}}[id])$
$\mathsf{Query} \leftarrow \mathsf{Query} \cup \{(id, m)\}$
return σ

Fig. 8: Oracles used in Game 1.

Game 2 is identical to Game 1, except for a change to the COERCE and STCRCSIG oracle. Instead of using s' during the generation of the fake transcript, we will instead use the real key s. We give the COERCE and STCRCSIG oracles used in Game 2 in Figure 9.

$\mathsf{COERCE}_{(pp,pk_A,\mathsf{L},\mathsf{corL},\mathsf{crcL},\mathsf{trans},\mathsf{trans}_{\mathsf{fake}})}(id)$

parse pp as $(pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}})$ $\wedge$ **parse** $\mathsf{trans}[id]$ as $\{(r_{\mathsf{SIG}}, pk_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, s, r_c, c), \ldots, (m, r_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2), \ldots\}$
if $id \notin \mathsf{L} \setminus \mathsf{corL}$ **return** $\perp$
if $id \notin \mathsf{crcL}$ $\mathsf{crcL} \leftarrow \mathsf{crcL} \cup \{id\}$
  $r'_c \leftarrow \mathsf{DEN.Exp}(pp_{\mathsf{DEN}}, pk_A, c, s)$
  $\mathsf{trans}_{\mathsf{fake}}[id] \coloneqq \{(r_{\mathsf{SIG}}, pk_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, s, r'_c, c)\}$
  For each entry $(m, r_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2)$ from $\mathsf{CR.Sign}$
    $r'_{\sigma_1} \leftarrow \mathsf{DEN.Exp}(pp_{\mathsf{DEN}}, pk_A, \sigma_1, \mathcal{H}_1(m||s))$
    $\mathsf{trans}_{\mathsf{fake}}[id] \leftarrow \mathsf{trans}_{\mathsf{fake}}[id] \cup \{(m, r'_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2)\}$
if $b = 0$ **return** $\mathsf{trans}[id]$
if $b = 1$ **return** $\mathsf{trans}_{\mathsf{fake}}[id]$

$\mathsf{STCRCSIG}_{(pp,pk_A,\mathbf{sk_S},\mathsf{L},\mathsf{corL},\mathsf{crcL},\mathsf{trans},\mathsf{trans}_{\mathsf{fake}})}(id, m, r)$

parse pp as $(pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}})$ $\wedge$ **parse** $\mathsf{trans}[id]$ as $\{(r_{\mathsf{SIG}}, pk_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, s, r_c, c), \ldots, (m, r_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2), \ldots\}$
if $id \notin \mathsf{L} \setminus \mathsf{corL}$ **return** $\perp$
if $id \notin \mathsf{crcL}$ $\mathsf{crcL} \leftarrow \mathsf{crcL} \cup \{id\}$
  $r'_c \leftarrow \mathsf{DEN.Exp}(pp_{\mathsf{DEN}}, pk_A, c, s)$
  $\mathsf{trans}_{\mathsf{fake}}[id] \coloneqq \{(r_{\mathsf{SIG}}, pk_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, s, r'_c, c)\}$
  For each entry $(m, r_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2)$ from $\mathsf{CR.Sign}$
    $r'_{\sigma_1} \leftarrow \mathsf{DEN.Exp}(pp_{\mathsf{DEN}}, pk_A, \sigma_1, \mathcal{H}_1(m||s))$
    $\mathsf{trans}_{\mathsf{fake}}[id] \leftarrow \mathsf{trans}_{\mathsf{fake}}[id] \cup \{(m, r'_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2)\}$
if $b = 0$ $(\sigma, \mathsf{trans}[id]) \leftarrow \mathsf{CR.Sign}(pp, \mathbf{sk_S}[id], pk_A, m, \mathsf{trans}[id]; r)$
if $b = 1$ $(\sigma, \mathsf{trans}_{\mathsf{fake}}[id]) \leftarrow \mathsf{CR.FakeSign}(pp, \mathbf{sk_S}[id], pk_A, m, \mathsf{trans}[id], \mathsf{trans}_{\mathsf{fake}}[id]; r)$
return σ

Fig. 9: COERCE, STCRCSIG oracles in Game 2.

This game hop will make use of a hybrid argument. We split the reduction into k_1 steps. We define Game 1.i to behave as in Game 2 only for the first i signers submitted to the ADDU oracle and otherwise behave as in Game 1. Clearly, Game 1.k_1 will be identical to Game 2 and Game 1.0 will be identical to Game 1.

We show that Game 1.i and Game 1.$(i+1)$ are indistinguishable assuming the encryption scheme DEN is IND-CPA secure. We give a distinguishing algorithm $\mathcal{D}_1$ in Figure 10. $\mathcal{D}_1$ aims to guess a bit b' in the IND-CPA game for multiple encryptions. In this game, instead of receiving one ciphertext, the adversary can make several queries of the form (m_0, m_1) and will receive $\mathsf{DEN.Enc}(pp_{\mathsf{DEN}}, pk_{\mathsf{DEN}}, m_b)$ for each query to an oracle $\mathsf{LR}_{\mathsf{CPA}}$.

We show that, when $b' = 0$, inputs to $\mathcal{A}$ are identical to Game 1.i and, when $b' = 1$, inputs to $\mathcal{A}$ are identical to Game 1.$(i + 1)$.

The distinguisher must correctly simulate the inputs to the adversary, whether the user $\hat{id}$ (the $i + 1$th user generated by the ADDU oracle) is corrupted or coerced by the adversary. However, to correctly simulate the outputs we need to know whether the user is corrupt or coerced at key generation, at which point the adversary has not necessarily determined this yet. The bit d, chosen by the distinguisher, is a guess as to whether the user is corrupted ($d = 0$) or coerced ($d = 1$). If this guess is incorrect then the distinguisher aborts, adding an extra tightness gap of $1/2$.

Note that pp, pk_A input to $\mathcal{A}$ are honestly generated and are identical in Games 1.i and 1.$(i + 1)$. Furthermore, sk_A is not known but is never used. The ADDU and SIGN oracles are

$\mathsf{ADDU}_{(pp,pk_A,\mathbf{pk_S},\mathbf{sk_S},\mathsf{L},\mathsf{trans},\mathsf{trans_{fake}})}(id)$

parse pp as $(pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}})$
if $id \in \mathsf{L}$ **return** $\perp$
$\mathsf{L} \leftarrow \mathsf{L} \cup \{id\}$
$q \leftarrow q + 1$
if $q = i + 1$ $\hat{id} \leftarrow id$
if $b = 1 \ \wedge \ d = 1 \ \wedge \ id = \hat{id}$
  $(pk_{\mathsf{SIG}}, sk_{\mathsf{SIG}}) \leftarrow \mathsf{SIG.KGen}(pp_{\mathsf{SIG}}; r_{\mathsf{SIG}})$
  $s_0^* \leftarrow\$ \{0,1\}^\lambda, s_1^* \leftarrow\$ \{0,1\}^\lambda, c \leftarrow \mathsf{LR}_{\mathsf{CPA}}(s_0^*, s_1^*)$
  $(\mathbf{pk_S}[id], \mathbf{sk_S}[id]) \leftarrow ((pk_{\mathsf{SIG}}, c), (sk_{\mathsf{SIG}}, \perp))$
  $\mathsf{trans}[id] \coloneqq \{(r_{\mathsf{SIG}}, pk_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, \perp, \perp, c)\}$
  return $\mathbf{pk_S}[id]$
else as in Game 1

$\mathsf{SIGN}_{(pp,pk_A,\mathbf{sk_S},\mathsf{L},\mathsf{Query},\mathsf{trans},\mathsf{trans_{fake}})}(id, m)$

parse pp as $(pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}})$
parse $\mathbf{sk_S}[id]$ as (sk_{SIG}, s)
if $b = 1 \ \wedge \ d = 1 \ \wedge \ id = \hat{id}$
  $\sigma_1 \leftarrow \mathsf{LR}_{\mathsf{CPA}}(\mathcal{H}_1(m||s_0^*), \mathcal{H}_1(m||s_1^*))$
  $\sigma_2 \leftarrow \mathsf{SIG.Sign}(pp_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, \mathcal{H}_2(m||\sigma_1); r_{\sigma_2})$
  $\sigma \leftarrow (\sigma_1, \sigma_2)$
  $\mathsf{trans}[id] \leftarrow \mathsf{trans}[id] \cup \{(m, \perp, \sigma_1, r_{\sigma_2}, \sigma_2)\}$
  if $id \in \mathsf{crcL}$
    $\mathsf{trans_{fake}}[id] \leftarrow \mathsf{CR.FakeTrans}(pp, pk_A, \mathsf{trans}[id], \mathsf{trans_{fake}}[id])$
  $\mathsf{Query} \leftarrow \mathsf{Query} \cup \{(id, m)\}$
  return σ
else as in Game 1

$\mathsf{CRPT}_{(\mathbf{sk_S},\mathsf{L},\mathsf{corL},\mathsf{crcL},\mathsf{trans})}(id)$

if $id \notin \mathsf{L} \setminus \mathsf{crcL}$ **return** $\perp$
$\mathsf{corL} \leftarrow \mathsf{corL} \cup \{id\}$
if $d = 1 \ \wedge \ id = \hat{id}$ $\mathcal{D}_1$ aborts
else return $(\mathbf{sk_S}[id], \mathsf{trans}[id])$

$\mathcal{D}_1^{\mathsf{LR}_{\mathsf{CPA}}}(pp_{\mathsf{DEN}}, pk_{\mathsf{DEN}})$

$b \leftarrow\$ \{0,1\}, \ q \leftarrow 0, \ d \leftarrow\$ \{0,1\}$
$\mathbf{pk_S}, \mathbf{sk_S}, \mathsf{trans}, \mathsf{trans_{fake}} \leftarrow ()$,
$\mathsf{L}, \mathsf{crcL}, \mathsf{corL}, \mathsf{Query} \leftarrow \emptyset$
$pp_{\mathsf{SIG}} \leftarrow \mathsf{SIG.Setup}(1^\lambda)$
$pp \leftarrow (pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}})$
$pk_A \leftarrow pk_{\mathsf{DEN}}$
$b' \leftarrow \mathcal{A}^{\mathsf{ADDU},\mathsf{CRPT},\mathsf{COERCE},\mathsf{SIGN},\mathsf{STCRCSIG}}(pp, pk_A)$
if $b' = b$ **return** 1

$\mathsf{COERCE}_{(pp,pk_A,\mathsf{L},\mathsf{corL},\mathsf{crcL},\mathsf{trans},\mathsf{trans_{fake}})}(id)$

if $id \notin \mathsf{L} \setminus \mathsf{corL}$ **return** $\perp$
if $id = \hat{id}$
  if $b = 0$
    $\mathsf{crcL} \leftarrow \mathsf{crcL} \cup \{id\}$
    return $\mathsf{trans}[id]$
  if $d = 0$ $\mathcal{D}_1$ aborts
  parse pp as $(pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}})$
  parse $\mathsf{trans}[id]$ as $\{(r_{\mathsf{SIG}}, pk_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, \perp, \perp, c), \ldots, (m, \perp, \sigma_1, r_{\sigma_2}, \sigma_2), \ldots\}$
  if $id \notin \mathsf{crcL}$
    $\mathsf{crcL} \leftarrow \mathsf{crcL} \cup \{id\}$
    $r'_c \leftarrow \mathsf{DEN.Exp}(pp_{\mathsf{DEN}}, pk_A, c, s_1^*)$
    $\mathsf{trans_{fake}}[id] \coloneqq \{(r_{\mathsf{SIG}}, pk_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, s_1^*, r'_c, c)$
    For each entry $(m, \perp, \sigma_1, r_{\sigma_2}, \sigma_2)$ from $\mathsf{CR.Sign}$
      $r'_{\sigma_1} \leftarrow \mathsf{DEN.Exp}(pp_{\mathsf{DEN}}, pk_A, \sigma_1, \mathcal{H}_1(m||s_1^*))$
      $\mathsf{trans_{fake}}[id] \leftarrow \mathsf{trans_{fake}}[id] \cup \{(m, r'_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2)\}$
  return $\mathsf{trans_{fake}}[id]$
else As in Game 1.i

$\mathsf{STCRCSIG}_{(pp,pk_A,\mathbf{sk_S},\mathsf{L},\mathsf{crcL},\mathsf{corL},\mathsf{trans},\mathsf{trans_{fake}})}(id, m, r)$

if $id \notin \mathsf{L} \setminus \mathsf{corL}$ **return** $\perp$
if $id = \hat{id}$
  if $b = 0$
    $\mathsf{crcL} \leftarrow \mathsf{crcL} \cup \{id\}$
    $(\sigma, \mathsf{trans}[id]) \leftarrow \mathsf{CR.Sign}(pp, \mathbf{sk_S}[id], pk_A, m, \mathsf{trans}[id]; r)$
    return σ
  if $d = 0$ $\mathcal{D}_1$ aborts
  parse pp as $(pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}})$
  parse $\mathsf{trans}[id]$ as $\{(r_{\mathsf{SIG}}, pk_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, \perp, \perp, c), \ldots, (m, \perp, \sigma_1, r_{\sigma_2}, \sigma_2), \ldots\}$
  if $id \notin \mathsf{crcL}$
    $\mathsf{crcL} \leftarrow \mathsf{crcL} \cup \{id\}$
    $r'_c \leftarrow \mathsf{DEN.Exp}(pp_{\mathsf{DEN}}, pk_A, c, s_1^*)$
    $\mathsf{trans_{fake}}[id] \coloneqq \{(r_{\mathsf{SIG}}, pk_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, s_1^*, r'_c, c)$
    For each entry $(m, \perp, \sigma_1, r_{\sigma_2}, \sigma_2)$ from $\mathsf{CR.Sign}$
      $r'_{\sigma_1} \leftarrow \mathsf{DEN.Exp}(pp_{\mathsf{DEN}}, pk_A, \sigma_1, \mathcal{H}_1(m||s_1^*))$
      $\mathsf{trans_{fake}}[id] \leftarrow \mathsf{trans_{fake}}[id] \cup \{(m, r'_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2)\}$
  $(\sigma, \mathsf{trans_{fake}}[id]) \leftarrow \mathsf{CR.FakeSign}(pp, \mathbf{sk_S}[id], pk_A, m, \mathsf{trans}[id], \mathsf{trans_{fake}}[id]; r)$
  return σ
else As in Game 1.i

Fig. 10: $\mathcal{D}_1$ that distinguishes between Game 1.i and Game 1.i+1.

distributed identically to both Games 1.i and 1.$(i+1)$, because when $b = 1, d = 1$ and $id = \hat{id}$, $s_{b'}^*$ is essentially used as the secret key. It is encrypted in both the public key and in honest signatures. The CRPT oracle is distributed identically to both Games 1.i and 1.$(i+1)$, provided it doesn't abort. This is because the real transcript is generated identically provided $id \neq \hat{id}$ or $d = 0$. Games 1.i and 1.$(i+1)$ are identical with the exception of oracles COERCE and $\mathsf{STCRCSIG}$ when $id = \hat{id}$ where $\hat{id}$ is the $i+1$th signer created. When $b = 0$, the fake transcript is never used and the real transcript is generated normally. Therefore the outputs of the COERCE and $\mathsf{STCRCSIG}$ oracles are distributed identically to both Games 1.i and 1.$(i+1)$. When $d = 0$, the oracles both abort. When $d = 1$ and $b = 1$, if $b' = 0$ the oracle is distributed identically to Game 1 because s_0^* is used as the secret key and s_1^* is used in the fake transcript. When $d = 1$ and $b = 1$, if $b' = 1$ the oracle is distributed identically to Game 2 because s_1^* is used as both the

secret key and in the fake transcript. The probability that distinguisher $\mathcal{D}_1$ aborts is at most 1/2. Therefore $|Pr[S_{1.i}] - Pr[S_{1.(i+1)}]| \leq 2\epsilon_{\mathsf{IND\text{-}CPA-mult}}$, where $\epsilon_{\mathsf{IND\text{-}CPA-mult}}$ is the advantage in breaking the IND-CPA for multiple encryptions property of the deniable encryption scheme DEN. By a standard argument, $|Pr[S_{1.i}] - Pr[S_{1.(i+1)}]| \leq 2(1+k_2)\epsilon_{\mathsf{IND\text{-}CPA}}$, where $\epsilon_{\mathsf{IND\text{-}CPA}}$ is the advantage in breaking the IND-CPA property of DEN. Therefore $|Pr[S_1] - Pr[S_2]| \leq 2k_1(1+k_2)\epsilon_{\mathsf{IND\text{-}CPA}}$

Game 3 is identical to Game 2, except for another change to the SIGN, COERCE and STCRCSIG oracles. We include the real randomness used to encrypt in the fake transcript, instead of obtaining fake randomness from algorithm DEN.Exp. Additionally, in STCRCSIG we use the honest signing oracle regardless of b and then update the fake transcript as in the real transcript. We give the SIGN, COERCE and STCRCSIG oracles used in Game 3 in Figure 11.

$\mathsf{SIGN}_{(pp,pk_A,\mathbf{sk_S},\mathsf{L},\mathsf{corL},\mathsf{crcL},\mathsf{Query},\mathsf{trans},\mathsf{trans_{fake}})}(id, m)$

if $id \notin \mathsf{L}$ **return** $\perp$
$(\sigma, \mathsf{trans}[id]) \leftarrow \mathsf{CR.Sign}(pp, \mathbf{sk_S}[id], pk_A, m, \mathsf{trans}[id])$
if $id \in \mathsf{crcL}$ $\mathsf{trans_{fake}}[id] \leftarrow \mathsf{trans}[id]$
$\mathsf{Query} \leftarrow \mathsf{Query} \cup \{(id, m)\}$
return σ

$\mathsf{COERCE}_{(\mathsf{L},\mathsf{corL},\mathsf{crcL},\mathsf{trans},\mathsf{trans_{fake}})}(id)$

if $id \notin \mathsf{L} \setminus \mathsf{corL}$ **return** $\perp$
if $id \notin \mathsf{crcL}$
  $\mathsf{crcL} \leftarrow \mathsf{crcL} \cup \{id\}$
  $\mathsf{trans_{fake}}[id] \leftarrow \mathsf{trans}[id]$
if $b = 0$ **return** $\mathsf{trans}[id]$
if $b = 1$ **return** $\mathsf{trans_{fake}}[id]$

$\mathsf{STCRCSIG}_{(pp,pk_A,\mathbf{sk_S},\mathsf{L},\mathsf{crcL},\mathsf{corL},\mathsf{trans},\mathsf{trans_{fake}})}(id, m, r)$

parse $\mathsf{trans}[id]$ as $\{(r_{\mathsf{SIG}}, pk_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, s, r_c, c), \ldots, (m, r_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2), \ldots\}$
if $id \notin \mathsf{L} \setminus \mathsf{corL}$ **return** $\perp$
$\mathsf{crcL} \leftarrow \mathsf{crcL} \cup \{id\}$
$(\sigma, \mathsf{trans}[id]) \leftarrow \mathsf{CR.Sign}(pp, \mathbf{sk_S}[id], pk_A, m, \mathsf{trans}[id]; r)$
$\mathsf{trans_{fake}}[id] \leftarrow \mathsf{trans}[id]$
return σ

Fig. 11: SIGN, COERCE and STCRCSIG oracles in Game 3.

This game hop will again make use of a hybrid argument. We split the reduction into k_1 steps. We define Game 2.i to behave as in Game 3 only for the first i signers submitted to the ADDU oracle and otherwise behave as in Game 2. Clearly, Game 2.k_1 will be identical to Game 3 and Game 2.0 will be identical to Game 2.

We show that Game 2.i and Game 2.$(i+1)$ are indistinguishable assuming the deniable encryption scheme DEN is IND-EXP secure. We give a distinguishing algorithm $\mathcal{D}_2$ in Figure 12. $\mathcal{D}_2$ aims to guess a bit b' in the IND-EXP game, which is adjusted similarly to the IND-CPA experiment for multiple encryptions. That is, the adversary can submit multiple queries to an oracle $\mathsf{LR_{EXP}}$ that, on input a message, returns a ciphertext and randomness. If $b' = 0$, oracle $\mathsf{LR_{EXP}}$ returns the randomness used to encrypt and, if $b' = 1$, it returns randomness obtained via the algorithm DEN.Exp.

We show that, when $b' = 0$, inputs to $\mathcal{A}$ are identical to Game 2.$(i+1)$ and, when $b' = 1$, inputs to $\mathcal{A}$ are identical to Game 2.i. Note that pp, pk_A input to $\mathcal{A}$ are honestly generated and are identical in Games 2.i and 2.$(i+1)$. Furthermore, sk_A is not known but is never used. The outputs of the ADDU and SIGN oracles are distributed identically to both Games 2.i and 2.$(i+1)$, because when $b = 1, d = 1$ and $id = \hat{id}$, the secret key s is always encrypted in both the public key and in signatures. The CRPT oracle is distributed identically to both Games 2.i and 2.$(i+1)$, provided it doesn't abort. This is because the real transcript is generated identically provided $id \neq \hat{id}$ or $d = 0$.

Games 2.i and 2.$(i+1)$ are identical with the exception of oracles SIGN, COERCE and STCRCSIG when $id = \hat{id}$. When $b = 0$, the fake transcript is never used and the real transcript is generated normally. Therefore the outputs of the COERCE and STCRCSIG oracles are distributed

$\mathsf{ADDU}_{(pp,pk_A,\mathbf{pk_S},\mathbf{sk_S},\mathsf{L},\mathsf{trans},\mathsf{trans}_{\mathsf{fake}})}(id)$

parse pp as $(pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}})$
if $id \in \mathsf{L}$ **return** $\perp$
$\mathsf{L} \leftarrow \mathsf{L} \cup \{id\}$
$q \leftarrow q + 1$
if $q = i + 1$ $\hat{id} \leftarrow id$
if $b = 1 \ \wedge \ d = 1 \ \wedge \ id = \hat{id}$
  $(pk_{\mathsf{SIG}}, sk_{\mathsf{SIG}}) \leftarrow \mathsf{SIG.KGen}(pp_{\mathsf{SIG}}; r_{\mathsf{SIG}})$
  $s \leftarrow\!\!\$ \ \{0,1\}^{\lambda}$
  $(c, r_c) \leftarrow \mathsf{LR}_{\mathsf{EXP}}(s)$
  $\mathsf{trans}[id] := \{(r_{\mathsf{SIG}}, pk_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, s, r_c, c)$
  $(\mathbf{pk_S}[id], \mathbf{sk_S}[id]) \leftarrow ((pk_{\mathsf{SIG}}, c), (sk_{\mathsf{SIG}}, s))$
  return $\mathbf{pk_S}[id]$
else as in Game 2

$\mathsf{SIGN}_{(pp,pk_A,\mathbf{sk_S},\mathsf{L},\mathsf{Query},\mathsf{trans},\mathsf{trans}_{\mathsf{fake}})}(id, m)$

parse pp as $(pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}})$
parse $\mathbf{sk_S}[id]$ as (sk_{SIG}, s)
if $b = 1 \ \wedge \ d = 1 \ \wedge \ id = \hat{id}$
  $(\sigma_1, r_{\sigma_1}) \leftarrow \mathsf{LR}_{\mathsf{EXP}}(\mathcal{H}_1(m||s))$
  $\sigma_2 \leftarrow \mathsf{SIG.Sign}(pp_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, \mathcal{H}_2(m||\sigma_1); r_{\sigma_2})$
  $\sigma \leftarrow (\sigma_1, \sigma_2)$
  $\mathsf{trans}[id] \leftarrow \mathsf{trans}[id] \cup \{(m, r_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2)\}$
  if $id \in \mathsf{crcL}$
    $\mathsf{trans}_{\mathsf{fake}}[id] \leftarrow \mathsf{trans}_{\mathsf{fake}}[id] \cup \{(m, r_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2)\}$
  $\mathsf{Query} \leftarrow \mathsf{Query} \cup \{(id, m)\}$
  return σ
else as in Game 2

$\mathsf{CRPT}_{(\mathsf{L},\mathsf{corL},\mathsf{crcL},\mathbf{sk_S},\mathsf{trans})}(id)$

if $id \notin \mathsf{L} \setminus \mathsf{crcL}$ **return** $\perp$
$\mathsf{corL} \leftarrow \mathsf{corL} \cup \{id\}$
if $d = 1 \ \wedge \ id = \hat{id}$ $\mathcal{D}_1$ aborts
else return $(\mathbf{sk_S}[id], \mathsf{trans}[id])$

$\mathsf{COERCE}_{(\mathsf{L},\mathsf{corL},\mathsf{crcL},\mathsf{trans},\mathsf{trans}_{\mathsf{fake}})}(id)$

if $id \notin \mathsf{L} \setminus \mathsf{corL}$ **return** $\perp$
if $id = \hat{id}$
  if $b = 0$
    $\mathsf{crcL} \leftarrow \mathsf{crcL} \cup \{id\}$
    return $\mathsf{trans}[id]$
  if $d = 0$ $\mathcal{D}_1$ aborts
  parse $\mathsf{trans}[id]$ as $\{(r_{\mathsf{SIG}}, pk_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, s, r_c, c), \ldots, (m, r_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2), \ldots\}$
  if $id \notin \mathsf{crcL}$
    $\mathsf{crcL} \leftarrow \mathsf{crcL} \cup \{id\}$
    $\mathsf{trans}_{\mathsf{fake}}[id] := \{(r_{\mathsf{SIG}}, pk_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, s, r_c, c), \ldots, (m, r_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2), \ldots\}$
  return $\mathsf{trans}_{\mathsf{fake}}[id]$
else As in Game 1.i

$\mathsf{STCRCSIG}_{(pp,pk_A,\mathbf{sk_S},\mathsf{L},\mathsf{crcL},\mathsf{corL},\mathsf{trans},\mathsf{trans}_{\mathsf{fake}})}(id, m, r)$

if $id \notin \mathsf{L} \setminus \mathsf{corL}$ **return** $\perp$
if $id = \hat{id}$
  if $b = 0$
    $\mathsf{crcL} \leftarrow \mathsf{crcL} \cup \{id\}$
    $(\sigma, \mathsf{trans}[id]) \leftarrow \mathsf{CR.Sign}(pp, \mathbf{sk_S}[id], pk_A, m, \mathsf{trans}[id]; r)$
    return σ
  if $d = 0$ $\mathcal{D}_1$ aborts
  parse $\mathsf{trans}[id]$ as $\{(r_{\mathsf{SIG}}, pk_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, s, r_c, c), \ldots, (m, r_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2), \ldots\}$
  parse r as $(r_{\sigma_1}, r_{\sigma_2})$
  if $id \notin \mathsf{crcL}$
    $\mathsf{crcL} \leftarrow \mathsf{crcL} \cup \{id\}$
    $\mathsf{trans}_{\mathsf{fake}}[id] := \{(r_{\mathsf{SIG}}, pk_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, s, r_c, c), \ldots, (m, r_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2), \ldots\}$
  $\sigma_1 \leftarrow \mathsf{DEN.Enc}(pp_{\mathsf{DEN}}, pk_A, \mathcal{H}_1(m||s); r_{\sigma_1})$
  $\sigma_2 \leftarrow \mathsf{SIG.Sign}(pp_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, \mathcal{H}_2(m||\sigma_1); r_{\sigma_2}), \sigma \leftarrow (\sigma_1, \sigma_2)$
  $\mathsf{trans}_{\mathsf{fake}}[id] \leftarrow \mathsf{trans}_{\mathsf{fake}}[id] \cup \{(m, r_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2)\}$
  return σ
else As in Game 1.i

$\mathcal{D}_2^{\mathsf{LR}_{\mathsf{EXP}}}(pp_{\mathsf{DEN}}, pk_{\mathsf{DEN}})$

$b \leftarrow\!\!\$ \ \{0,1\}, \ q \leftarrow 0, \ d \leftarrow\!\!\$ \ \{0,1\}$
$\mathbf{pk_S}, \mathbf{sk_S}, \mathsf{trans}, \mathsf{trans}_{\mathsf{fake}} \leftarrow ()$
$\mathsf{L}, \mathsf{crcL}, \mathsf{corL}, \mathsf{Query} \leftarrow \emptyset$
$pp_{\mathsf{SIG}} \leftarrow \mathsf{SIG.Setup}(1^{\lambda})$
$pp \leftarrow (pp_{\mathsf{DEN}}$
$pp_{\mathsf{SIG}}), pk_A \leftarrow pk_{\mathsf{DEN}}$
$b' \leftarrow \mathcal{A}^{\mathsf{ADDU},\mathsf{CRPT},\mathsf{COERCE},\mathsf{SIGN},\mathsf{STCRCSIG}}(pp, pk_A)$
if $b' = b$ **return** 1

Fig. 12: $\mathcal{D}_2$ that distinguishes between Game 2.i and Game 2.$(i+1)$.

identically to both Games 2.i and 2.$(i+1)$. When $d = 0$, the COERCE and STCRCSIG oracles both abort. When $d = 1$ and $b = 1$, if $b' = 0$ the fake transcript is distributed identically to Game 3 because $\mathcal{D}_2$ is input the real randomness used in the encryption. When $d = 1$ and $b = 1$, if $b' = 1$ the fake transcript is distributed identically to Game 2 because $\mathcal{D}_2$ is input the randomness output by DEN.Exp. In STCRCSIG, as the real key is included in the fake transcript in both Games 2 and Games 3, the signature output is distributed identically to both Game 2 and Game 3. The probability that distinguisher $\mathcal{D}_2$ aborts is at most 1/2. Therefore $|Pr[S_{2.i}] - Pr[S_{2.(i+1)}]| \leq 2\epsilon_{\mathsf{IND\text{-}EXP\text{-}mult}}$, where $\epsilon_{\mathsf{IND\text{-}EXP\text{-}mult}}$ is the advantage in breaking the IND-EXP for multiple messages property of deniable encryption scheme DEN. By a standard argument, $|Pr[S_{2.i}] - Pr[S_{2.(i+1)}]| \leq 2(1 + k_2 + k_3)\epsilon_{\mathsf{IND\text{-}EXP}}$ where $\epsilon_{\mathsf{IND\text{-}EXP}}$ is the advantage in breaking the IND-EXP property of DEN. Therefore $|Pr[S_2] - Pr[S_3]| \leq 2k_1(1 + k_2 + k_3)\epsilon_{\mathsf{IND\text{-}EXP}}$.

In Game 3, the fake transcript is identical to the real transcript. Also signatures output by STCRCSIG are now always generated by the honest signing algorithm. Therefore, inputs to the adversary $\mathcal{A}$ are independent of b and so $Pr[S_3] = 1/2$.

We now have that

$$|Pr[S_0] - 1/2| \leq 2k_1(1 + k_2)(\epsilon_{\mathsf{IND\text{-}EXP}} + \epsilon_{\mathsf{IND\text{-}CPA}}) + 2k_1k_3\epsilon_{\mathsf{IND\text{-}EXP}}$$

and conclude that the advantage of the adversary in Game 0 is negligible as required.

Lemma 6 (Strong Soundness). CR.SIG *satisfies strong soundness for a coercion-resistant incoercible signature scheme if* $\mathcal{H}_1$ *is modelled as a random oracle model and the deniable encryption scheme satisfies* IND-CPA *security.*

Proof. We show that, if there exists an adversary that succeeds in the strong soundness experiment, then it is possible to construct an adversary that breaks the IND-CPA property of deniable encryption scheme DEN, if it is assumed that hash function $\mathcal{H}_1$ is a random oracle. This is because, to win, the adversary must output a signer id^* alongside an encryption of $\mathcal{H}_1(m^*||s)$, where s is the secret key of the signer id^*. Therefore, they must have input $(m^*||s)$ to the $\mathcal{H}_1$ random oracle. In our reduction we show that if they can do so we can break the IND-CPA security of the deniable encryption scheme DEN. Then, the result holds by contradiction.

First we show that if there exists an adversary $\mathcal{A}$ that makes k_1, k_2 queries to the SIGN oracle and ADDU oracle respectively, such that $\Pr\left[\mathsf{Exp}^{\mathsf{st\text{-}sound}}_{\mathcal{A},\mathsf{CR.SIG}}(\lambda) = 1\right] = \epsilon$, where ϵ is non-negligible, then we can build an adversary $\mathcal{A}'$ that breaks the IND-CPA for multiple encryptions security of the deniable encryption scheme DEN with non-negligible probability. We describe the IND-CPA multiple encryptions security game in the proof of Lemma 5. We give the detailed description of $\mathcal{A}'$ in Figure 13, and explain here how $\mathcal{A}'$ works.

First note that all inputs that $\mathcal{A}'$ provides to $\mathcal{A}$ are distributed identically to the strong soundness experiment for the CR.SIG construction. The pp, pk_A input to $\mathcal{A}$ are honestly generated and are identical to in the strong soundness experiment. Furthermore, sk_A is not known but is never used.

Simulating the ADDU *oracle.* For all signers other than $\hat{id}$, the ADDU oracle is identical to in the strong soundness experiment. For the signer $\hat{id}$, sk_{SIG} and pk_{SIG} are generated identically. c is the encryption of $s^*_{b'}$ under pk_A, where b' is the bit chosen in the IND-CPA experiment. As both s^*_1, s^*_0 are chosen identically to s, then c is distributed correctly. The second part of the secret key and real transcript is not known, but we will show that this is not used throughout. The fake transcript is generated as normal using the fake key and the deniable encryption explanation algorithm.

Simulating the SIGN *oracle.* For all signers other than $\hat{id}$, this is identical to in the strong soundness experiment. For the signer $\hat{id}$, σ_1 is the encryption of $\mathcal{H}_1(m||s^*_{b'})$ under pk_A, where b' is the bit chosen in the IND-CPA experiment. This is consistent with the ADDU oracle, because $s^*_{b'}$ is encrypted in the public key. σ_2 and the fake transcript are generated identically to in the strong soundness experiment. Again, the real transcript is not known in full, but is never used.

Simulating the other oracles. If CRPT is input $\hat{id}$, then $\mathcal{A}'$ will abort. For all other signers, the ADDU and SIGN oracles were performed as normal. FAKECOERCE and FAKESIGN do not require s or the real transcript as the fake key and transcript are used instead. The hash oracle $\mathcal{H}_1$ is distributed identically to the random oracle model.

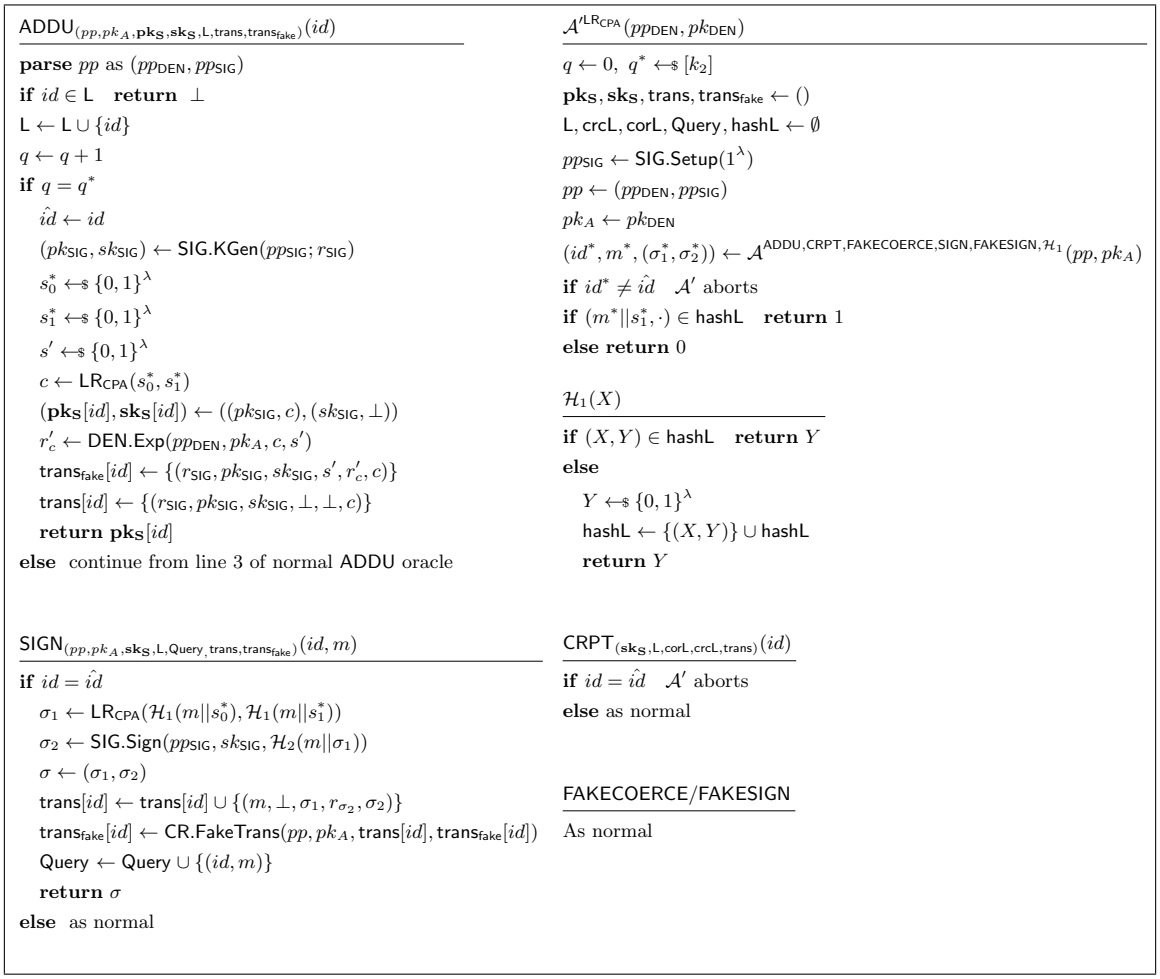

Fig. 13: $\mathcal{A}'$ that breaks the IND-CPA security of the deniable encryption scheme.

Reduction to IND-CPA *security.* Assume $\mathcal{A}$ is successful. We assume with probability $1/k_2$ that $\mathcal{A}'$ guesses correctly and $\hat{id} = id^*$. Then $\mathcal{A}'$ will not abort because, for $\mathcal{A}$ to be successful, $id^* \in \mathsf{crcL}$ and so cannot have been input to CRPT.

Parse $\mathbf{pk_S}[id^*] = (pk_{\mathsf{SIG}}, c)$. For $\mathcal{A}$ to be successful, $\mathsf{DEN.Dec}(pp_{\mathsf{DEN}}, sk_A, \sigma_1^*) = \mathcal{H}_1(m^*||\mathsf{DEN.Dec}(pp_{\mathsf{DEN}}, sk_A, c))$. Therefore, $(m^*||\mathsf{DEN.Dec}(pp_{\mathsf{DEN}}, sk_A, c), \cdot)$ must be included in hashL, otherwise the probability of this occurring is negligible. We know that $\mathsf{DEN.Dec}(pp_{\mathsf{DEN}}, sk_A, c)$ is either s_0^* or s_1^*, and so either $m^*||s_0^*$ or $m^*||s_1^*$ must be included in hashL. As $(id^*, m^*) \notin \mathsf{Query}$, we have not queried either to the signing oracle. As such, the adversary must have queried $\mathcal{H}_1$ directly. All inputs to the adversary are independent of $s_{1-b'}^*$. Therefore, except for the negligible probability of the adversary guessing $s_{1-b'}^*$, they must have queried only $m^*||s_{b'}^*$ to the hash oracle. Therefore, we successfully guess b' in the IND-CPA game.

Therefore, the probability that $\mathcal{A}'$ outputs 1 given $b' = 1$ is greater than $(1 - \mathsf{negl}(\lambda))\epsilon$. The $(1 - \mathsf{negl}(\lambda))$ factor is due to the negligible chance that $\mathcal{A}$ wins without querying $m^*||s_1^*$ to the hash oracle. The probability that $\mathcal{A}'$ outputs 1 given $b' = 0$ is $\leq \mathsf{negl}(\lambda)$, as all inputs to $\mathcal{A}$ are independent of s_1^* and so the chance that $\mathcal{A}$ has input this to the hash oracle is negligible. Therefore, the IND-CPA multiple encryptions advantage is $\geq k_2((1 - \mathsf{negl}(\lambda))\epsilon - \mathsf{negl}(\lambda))$, which is non-negligible. By contradiction, our CR.SIG construction satisfies strong soundness.

3.3 A Strong Receipt-Free Construction

We present a strong receipt-free incoercible signature scheme construction (RF.SIG) in Figure 14 that uses a standard signature scheme SIG and a deniable encryption scheme DEN as building blocks. We also use a collision resistant hash function $\mathcal{H} : \{0,1\}^* \to \mathcal{M}$, where $\mathcal{M}$ is the message space of the signature scheme SIG.

We briefly note the key differences between our constructions. During key generation, a signer only generates a key pair for signature scheme SIG, and no longer generates a random string s. To sign a message, the signer now generates an encryption of a bit that denotes whether the signer is being coerced, i.e., the signer encrypts 1 to indicate a genuine signature and 0 to indicate coercion. Signing then proceeds as in our coercion-resistant construction. The authenticator can decrypt this bit (and consequently detect coercion) via the Authenticate algorithm. By using deniable encryption, our construction ensures that, when a signer is coerced, they can produce fake randomness such that they appear to have encrypted a different bit.

Our strong receipt-freeness construction RF.SIG does not satisfy coercion-resistance. In fact, if the attacker can obtain a coerced signer's transcript and provide a message and randomness to the signer, the coerced signer cannot output a fake transcript that will convince the attacker that the signer cooperated. Hence, RF.SIG cannot satisfy the IND3 indistinguishability requirement of coercion-resistance. Moreover, the fake transcript of a coerced signer contains the real secret key of the signer, which a coercive attacker can use to create valid and authentic forgeries on behalf of the signer. As such, RF.SIG cannot satisfy the strong soundness property that is necessary for coercion-resistance.

We show that our strong receipt-free construction satisfies correctness, completeness, unforgeability and strong receipt-freeness, as defined in Section 2. In fact, we obtain Theorem 2.

$\mathsf{RF.Setup}(1^\lambda)$

$pp_{\mathsf{DEN}} \leftarrow \mathsf{DEN.Setup}(1^\lambda)$
$pp_{\mathsf{SIG}} \leftarrow \mathsf{SIG.Setup}(1^\lambda)$
return $pp = (pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}})$

$\mathsf{RF.AKGen}(pp)$

parse pp as $(pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}})$
$(pk_A, sk_A) \leftarrow \mathsf{DEN.KGen}(pp_{\mathsf{DEN}})$
return (pk_A, sk_A)

$\mathsf{RF.SKGen}(pp, pk_A)$

parse pp as $(pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}})$
$(pk_S, sk_S) \leftarrow \mathsf{SIG.KGen}(pp_{\mathsf{SIG}}; r_{\mathsf{SIG}})$
$\mathsf{trans} := \{(r_{\mathsf{SIG}}, pk_S, sk_S)\}$
return $((pk_S, sk_S), \mathsf{trans})$

$\mathsf{RF.FakeTrans}(pp, pk_A, \mathsf{trans}, \mathsf{trans}_{\mathsf{fake}})$

parse pp as $(pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}})$
parse trans as $\{(r_{\mathsf{SIG}}, pk_S, sk_S), \ldots, (m, r_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2), \ldots\}$
For each new entry $(m, r_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2)$from RF.Sign
  $\mathsf{trans}_{\mathsf{fake}} \leftarrow \mathsf{trans}_{\mathsf{fake}} \cup \{(m, r_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2)\}$
return $\mathsf{trans}_{\mathsf{fake}}$

$\mathsf{RF.FakeTrans}(pp, pk_A, \mathsf{trans}, \perp)$

return $\mathsf{trans}_{\mathsf{fake}} := \mathsf{trans} = \{(r_{\mathsf{SIG}}, pk_S, sk_S)\}$

$\mathsf{RF.Sign}(pp, sk_S, pk_A, m, \mathsf{trans})$

parse pp as $(pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}})$
$\sigma_1 \leftarrow \mathsf{DEN.Enc}(pp_{\mathsf{DEN}}, pk_A, 1; r_{\sigma_1})$
$\sigma_2 \leftarrow \mathsf{SIG.Sign}(pp_{\mathsf{SIG}}, sk_S, \mathcal{H}(m||\sigma_1); r_{\sigma_2})$
$\mathsf{trans} \leftarrow \mathsf{trans} \cup \{(m, r_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2)\}$
return $(\sigma = (\sigma_1, \sigma_2), \mathsf{trans})$

$\mathsf{RF.Verify}(pp, pk_S, m, \sigma)$

parse pp as $(pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}})$, σ as (σ_1, σ_2)
if $\mathsf{SIG.Vf}(pp_{\mathsf{SIG}}, pk_S, \mathcal{H}(m||\sigma_1), \sigma_2) = 0$ **return** 0
return 1

$\mathsf{RF.Authenticate}(pp, pk_S, sk_A, m, \sigma)$

parse pp as $(pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}})$, σ as (σ_1, σ_2)
if $\mathsf{SIG.Vf}(pp_{\mathsf{SIG}}, pk_S, \mathcal{H}(m||\sigma_1), \sigma_2) = 0$ **return** 0
$t \leftarrow \mathsf{DEN.Dec}(pp_{\mathsf{DEN}}, sk_A, \sigma_1)$
if $t = 1$ **return** 1 **else** **return** 0

$\mathsf{RF.FakeSign}(pp, sk_S, pk_A, m, \mathsf{trans}, \mathsf{trans}_{\mathsf{fake}})$

parse pp as $(pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}})$
$\sigma_1 \leftarrow \mathsf{DEN.Enc}(pp_{\mathsf{DEN}}, pk_A, 0; r_{\sigma_1})$
$\sigma_2 \leftarrow \mathsf{SIG.Sign}(pp_{\mathsf{SIG}}, sk_S, \mathcal{H}(m||\sigma_1); r_{\sigma_2})$
$r'_{\sigma_1} \leftarrow \mathsf{DEN.Exp}(pp_{\mathsf{DEN}}, pk_A, \sigma_1, 1)$
$\mathsf{trans}_{\mathsf{fake}} \leftarrow \mathsf{trans}_{\mathsf{fake}} \cup \{(m, r'_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2)\}$
return $(\sigma = (\sigma_1, \sigma_2), \mathsf{trans}_{\mathsf{fake}})$

Fig. 14: Our strong receipt-free construction RF.SIG.

Theorem 2. *Let* SIG *and* DEN *be a secure signature scheme and deniable encryption scheme respectively, and the hash function* $\mathcal{H}$ *be collision resistant. Then,* RF.SIG *is a secure construction of a strong receipt-free incoercible signature scheme. That is,* RF.SIG *satisfies correctness, completeness, unforgeability and strong receipt-freeness.*

We prove Theorem 2 through a series of Lemmata.

Lemma 7 (Correctness). RF.SIG *satisfies* correctness *if signature scheme* SIG *is correct.*

Proof. Consider a signature $\sigma = (\sigma_1, \sigma_2)$ output by algorithm RF.Sign with respect to message m for keys sk_S and pk_A generated by algorithms RF.SKGen and RF.AKGen respectively and parameters pp output by algorithm RF.Setup. Then, by definition of correctness, RF.SIG is correct if $\mathsf{RF.Verify}(pp, pk_S, m, \sigma)$ outputs 1 with overwhelming probability. Assume that RF.Verify does not return 1. Then it must be the case that $\mathsf{SIG.Vf}(pp_{\mathsf{SIG}}, pk_S, \mathcal{H}(m||\sigma_1), \sigma_2) = 0$. By assumption, signature scheme SIG is correct. Therefore, algorithm SIG.Vf returns 1 with overwhelming probability where pp_{SIG} and pk_S are generated according to the construction description and σ_2 is the output of SIG.Sign for message $\mathcal{H}(m||\sigma_1)$. Then, by contradiction, RF.SIG is correct.

Lemma 8 (Completeness). RF.SIG *satisfies* completeness *if deniable encryption scheme* DEN *is correct and* RF.SIG *is correct.*

Proof. Consider a signature $\sigma = (\sigma_1, \sigma_2)$ output by algorithm RF.Sign with respect to message m for keys sk_S and pk_A generated by algorithms RF.SKGen and RF.AKGen respectively and parameters pp output by algorithm RF.Setup. By definition of completeness, RF.SIG is complete if $\mathsf{RF.Authenticate}(pp, pk_S, sk_A, m, (\sigma_1, \sigma_2))$ outputs 1 with overwhelming probability. Assume that RF.Authenticate does not return 1. Then, either $\mathsf{SIG.Vf}(pp_{\mathsf{SIG}}, pk_S, \mathcal{H}(m||\sigma_1), \sigma_2) = 0$ or $\mathsf{DEN.Dec}(pp_{\mathsf{DEN}}, sk_A, \sigma_1) \neq 1$. By correctness of RF.SIG, SIG.Vf returns 1 with overwhelming probability. Moreover, by correctness of the deniable encryption scheme, DEN.Dec returns 1 where $\sigma_1 \leftarrow \mathsf{DEN.Enc}(pp_{\mathsf{DEN}}, pk_A, 1)$. Then, by contradiction, RF.SIG is complete.

Lemma 9 (Unforgeability). RF.SIG *satisfies* existential unforgeability against a chosen message attack *(*EUF-CMA*) if digital signature scheme* SIG *satisfies* EUF-CMA *security and the hash function* $\mathcal{H}$ *is collision resistant.*

Proof. The proof of unforgeability is very similar to the unforgeability proof of our coercion-resistant construction. That is, it follows from the EUF-CMA security of SIG and the fact that hash function $\mathcal{H}$ is collision resistant.

Let $\mathcal{A}$ be an adversary that succeeds in the EUF-CMA experiment (Definition 4) for the RF.SIG construction. We show that we can construct an adversary $\mathcal{A}'$ that succeeds in the EUF-CMA experiment (Definition 10) for digital signature scheme SIG. We present $\mathcal{A}'$ in Figure 15. It is clear that inputs to $\mathcal{A}$ are distributed identically to the EUF-CMA experiment of Definition 4 because keys pk_A and pk_S are generated identically. Moreover, oracle SIGN is distributed identically because $\mathcal{O}\mathsf{SIG}(\mathcal{H}(m||\sigma_1))$ returns $\mathsf{SIG.Sign}(pp_{\mathsf{SIG}}, sk_S, \mathcal{H}(m||\sigma_1))$. We now show that $\mathcal{A}'$ is successful in the EUF-CMA experiment of Definition 10. That is, we show that $(\mathcal{H}(m^*||\sigma_1^*), \sigma_2^*)$ is a valid message/signature pair that has not been input to $\mathcal{O}\mathsf{SIG}$. By assumption $(m^*, (\sigma_1^*, \sigma_2^*))$ is a valid message/signature pair, i.e., $\mathsf{RF.Verify}(pp, pk_S, m^*, (\sigma_1^*, \sigma_2^*)) = 1$. Then, by definition, $\mathsf{SIG.Vf}(pp_{\mathsf{SIG}}, pk_S, \mathcal{H}(m^*||\sigma_1^*), \sigma_2^*) = 1$. Moreover, by assumption, $m^* \notin \mathsf{Query}$ and, therefore, $\mathcal{H}(m^*||\sigma_1^*)$ is not input to $\mathcal{O}\mathsf{SIG}$ unless a hash collision occurs, which occurs with negligible probability. Therefore, $\mathcal{A}'$ succeeds in the EUF-CMA experiment and, by contradiction, the result holds.

$\mathsf{SIGN}(m)$	$\mathcal{A}'^{\mathcal{O}\mathsf{SIG}(\cdot)}(pp_{\mathsf{SIG}}, pk_{\mathsf{SIG}})$
$\sigma_1 \leftarrow \mathsf{DEN.Enc}(pp_{\mathsf{DEN}}, pk_A, 1)$	$\mathsf{Query} \leftarrow \emptyset$
$\sigma_2 \leftarrow \mathcal{O}\mathsf{SIG}(\mathcal{H}(m\|\|\sigma_1))$	$pp_{\mathsf{DEN}} \leftarrow \mathsf{DEN.Setup}(1^\lambda)$
$\mathsf{Query} \leftarrow \mathsf{Query} \cup \{m\}$	$pp \leftarrow (pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}})$
return $\sigma = (\sigma_1, \sigma_2)$	$(pk_A, sk_A) \leftarrow \mathsf{RF.AKGen}(pp)$
	$pk_S \leftarrow pk_{\mathsf{SIG}}$
	$(m^*, (\sigma_1^*, \sigma_2^*)) \leftarrow \mathcal{A}^{\mathsf{SIGN}}(pp, pk_S, pk_A, sk_A)$
	return $(\mathcal{H}(m^*\|\|\sigma_1^*), \sigma_2^*)$

Fig. 15: Adversary $\mathcal{A}'$ that breaks the EUF-CMA security of signature scheme SIG.

Lemma 10 ((IND2) Indistinguishability). RF.SIG *satisfies* (IND2) *indistinguishability for a strong receipt-free incoercible signature scheme if the deniable encryption scheme satisfies* IND-CPA *security and* IND-EXP *security.*

Proof. Indistinguishability holds as a result of the IND-CPA and IND-EXP properties of the deniable encryption scheme. To prove indistinguishability, we proceed through a series of game hops, demonstrating that the hops are indistinguishable to the adversary. In our first game hop, if $b = 1$, we change oracle CRCSIG to encrypt 1 rather than 0 when generating the fake signature. In our second game hop, if $b = 1$, we attach the real randomness used to encrypt to the fake transcript, rather than the randomness generated via algorithm DEN.Exp. These hops are indistinguishable if the deniable encryption scheme satisfies IND-CPA and IND-EXP security respectively. Through these game hops we arrive at a game in which the view of the adversary is identical for $b = 0$ and $b = 1$. In particular, regardless of bit b, the adversary views a signature that contains an encryption of a bit 1 and views a transcript that contains the real encryption randomness.

Let $\mathcal{A}$ be an adversary in the $\mathsf{Exp}^{\mathsf{IND2},b}_{\mathcal{A},\mathsf{RF.SIG}}(\lambda)$ experiment that makes at most k queries to oracle CRCSIG. We proceed through a series of game hops that we show are indistinguishable to the adversary. In the final game, the view of the adversary will be identical for $b = 0$ and $b = 1$ and the adversary will have a negligible chance of guessing b. We define Game 0 as the experiment $\mathsf{Exp}^{\mathsf{IND2},b}_{\mathcal{A},\mathsf{RF.SIG}}(\lambda)$ with b chosen randomly. Let S_i be the event that adversary $\mathcal{A}$ correctly guesses b after Game i.

Game 1 is identical to Game 0 except for a change to the CRCSIG oracle. When $b = 1$ we encrypt '1' rather than '0'. Both the real and fake transcript are updated as normal. We show that Game 0 and Game 1 are indistinguishable assuming the deniable encryption scheme DEN is IND-CPA secure. We give a distinguishing algorithm $\mathcal{D}_1$ in Figure 16. $\mathcal{D}_1$ aims to guess a bit b^* in the IND-CPA game for multiple encryptions. In this game, instead of receiving one ciphertext, the adversary can make several queries of the form (m_0, m_1) and will receive $\mathsf{DEN.Enc}(pp_{\mathsf{DEN}}, pk_{\mathsf{DEN}}, m_{b^*})$ for each query to an oracle $\mathsf{LR}_{\mathsf{CPA}}$.

$\mathsf{CRCSIG}_{(pp,pk_A,\mathbf{sk_S},\mathsf{L},\mathsf{corL},\mathsf{crcL},\mathsf{trans},\mathsf{trans}_{\mathsf{fake}})}(id, m)$

if $id \notin \mathsf{L} \setminus \mathsf{corL}$ **return** $\perp$
$\mathsf{crcL} \leftarrow \mathsf{crcL} \cup \{id\}$
if $b = 1$
  parse pp as $(pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}})$
  $\sigma_1 \leftarrow \mathsf{LR}_{\mathsf{CPA}}(0, 1)$
  $\sigma_2 \leftarrow \mathsf{SIG.Sign}(pp_{\mathsf{SIG}}, \mathbf{sk_S}[id], \mathcal{H}(m||\sigma_1); r_{\sigma_2})$
  $\sigma \leftarrow (\sigma_1, \sigma_2)$
  $r'_{\sigma_1} \leftarrow \mathsf{DEN.Exp}(pp_{\mathsf{DEN}}, pk_A, \sigma_1, 1)$
  $\mathsf{trans}_{\mathsf{fake}}[id] \leftarrow \mathsf{trans}_{\mathsf{fake}}[id] \cup \{(m, r'_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2)\}$
  return σ
else as normal

ADDU/CRPT/COERCE/SIGN

As normal.

$\mathcal{D}_1^{\mathsf{LR}_{\mathsf{CPA}}}(pp_{\mathsf{DEN}}, pk_{\mathsf{DEN}})$

$b \leftarrow \{0, 1\}$
$\mathbf{pk_S}, \mathbf{sk_S}, \mathsf{trans}, \mathsf{trans}_{\mathsf{fake}} \leftarrow ()$
$\mathsf{L}, \mathsf{crcL}, \mathsf{corL}, \mathsf{Query} \leftarrow \emptyset$
$pp_{\mathsf{SIG}} \leftarrow \mathsf{SIG.Setup}(1^\lambda)$
$pp \leftarrow (pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}})$
$pk_A \leftarrow pk_{\mathsf{DEN}}$
$b' \leftarrow \mathcal{A}^{\mathsf{ADDU},\mathsf{CRPT},\mathsf{COERCE},\mathsf{SIGN},\mathsf{CRCSIG}}(pp, pk_A)$
if $b' = b$ **return** 1

Fig. 16: Distinguisher $\mathcal{D}_1$ that distinguishes between Game 0 and Game 1.

We show that, when $b^* = 0$, inputs to $\mathcal{A}$ are identical to Game 0 and, when $b^* = 1$, inputs to $\mathcal{A}$ are identical to Game 1. Note that pp and pk_A input to $\mathcal{A}$ are honestly generated and are identical in Games 0 and 1. Furthermore, sk_A is not known but is never used. Games 0 and 1 are identical with the exception of oracle CRCSIG when $b = 1$. If $b^* = 0$, $\mathcal{D}_1$ is input an encryption of '0' as in Game 0. If $b^* = 1$, $\mathcal{D}_1$ is input an encryption of '1' as in Game 1. The real transcript is not known but this is never used because the same users cannot be input to both CRCSIG and CRPT, and $b = 1$ so the fake transcript is output by COERCE. Therefore, $\Big|\Pr[S_0] - \Pr[S_1]\Big| \leq \epsilon_{\mathsf{IND\text{-}CPA\text{-}mult}}$ where $\epsilon_{\mathsf{IND\text{-}CPA\text{-}mult}}$ is the advantage in breaking the IND-CPA for multiple encryptions property of deniable encryption scheme DEN. By a standard argument $|\Pr[S_0] - \Pr[S_1]| \leq k\epsilon_{\mathsf{IND\text{-}CPA}}$ where

$\epsilon_{\mathsf{IND\text{-}CPA}}$ is the advantage in breaking the IND-CPA property of DEN.

Game 2 is identical to Game 1 except for a further change to oracle CRCSIG. When $b = 1$, we include the real randomness used to encrypt in the fake transcript, rather than obtaining fake randomness from algorithm DEN.Exp. We show that Game 1 and Game 2 are indistinguishable assuming the deniable encryption scheme DEN is IND-EXP secure. We give a distinguishing algorithm $\mathcal{D}_2$ in Figure 17. $\mathcal{D}_2$ aims to guess a bit b^* in the IND-EXP game which is adjusted similarly to the IND-CPA experiment for multiple encryptions used in Game 1. That is, the adversary can submit multiple queries to an oracle $\mathsf{LR}_{\mathsf{EXP}}$ that, on input a message, returns a ciphertext and randomness. If $b^* = 0$, oracle $\mathsf{LR}_{\mathsf{EXP}}$ returns the randomness used to encrypt and, if $b^* = 1$, it returns randomness obtained via the algorithm DEN.Exp.

$\mathsf{CRCSIG}_{(pp,pk_A,\mathbf{sk_S},\mathsf{L},\mathsf{corL},\mathsf{crcL},\mathsf{trans},\mathsf{trans}_{\mathsf{fake}})}(id, m)$

if $id \notin \mathsf{L} \setminus \mathsf{corL}$ **return** $\perp$
$\mathsf{crcL} \leftarrow \mathsf{crcL} \cup \{id\}$
if $b = 1$
  parse pp as $(pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}})$
  $(\sigma_1, r_{\sigma_1}) \leftarrow \mathsf{LR}_{\mathsf{EXP}}(1)$
  $\sigma_2 \leftarrow \mathsf{SIG.Sign}(pp_{\mathsf{SIG}}, \mathbf{sk_S}[id], \mathcal{H}(m||\sigma_1); r_{\sigma_2})$
  $\sigma \leftarrow (\sigma_1, \sigma_2)$
  $\mathsf{trans}_{\mathsf{fake}} \leftarrow \mathsf{trans}_{\mathsf{fake}} \cup \{(m, r_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2)\}$
  return σ
else as normal

ADDU/CRPT/COERCE/SIGN

As normal.

$\mathcal{D}_1^{\mathsf{LR}_{\mathsf{EXP}}}(pp_{\mathsf{DEN}}, pk_{\mathsf{DEN}})$

$b \leftarrow \{0, 1\}$
$\mathbf{pk_S}, \mathbf{sk_S}, \mathsf{trans}, \mathsf{trans}_{\mathsf{fake}} \leftarrow ()$
$\mathsf{L}, \mathsf{crcL}, \mathsf{corL}, \mathsf{Query} \leftarrow \emptyset$
$pp_{\mathsf{SIG}} \leftarrow \mathsf{SIG.Setup}(1^\lambda)$
$pp \leftarrow (pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}})$
$pk_A \leftarrow pk_{\mathsf{DEN}}$
$b' \leftarrow \mathcal{A}^{\mathsf{ADDU},\mathsf{CRPT},\mathsf{COERCE},\mathsf{SIGN},\mathsf{CRCSIG}}(pp, pk_A)$
if $b' = b$ **return** 1

Fig. 17: Distinguisher $\mathcal{D}_1$ that distinguishes between Game 1 and Game 2.

We show that, when $b^* = 0$, inputs to $\mathcal{A}$ are identical to Game 2 and, when $b^* = 1$ inputs to $\mathcal{A}$ are identical to Game 1. As before, pp and pk_A input to $\mathcal{A}$ are identical in Games 1 and 2 and sk_A is not known but is never used. Games 1 and 2 are identical with the exception of oracle CRCSIG when $b = 1$. Regardless of b^*, oracle $\mathsf{LR}_{\mathsf{EXP}}$ always returns an encryption of '1' under pk_A. If $b^* = 0$, $\mathcal{D}_2$ is input the real randomness used to encrypt as used in Game 2. If $b^* = 1$, $\mathcal{D}_2$ is input the randomness output by $\mathsf{DEN.Exp}(pp_{\mathsf{DEN}}, pk_A, \sigma_1, 1)$ which is identical to the input to $\mathcal{A}$ in Game 1. Therefore, $|\Pr[S_1] - \Pr[S_2]| \leq \epsilon_{\mathsf{IND\text{-}EXP\text{-}mult}}$ where $\epsilon_{\mathsf{IND\text{-}EXP\text{-}mult}}$ is the advantage in breaking the IND-EXP for multiple messages property of deniable encryption scheme DEN. By a standard argument, $|\Pr[S_1] - \Pr[S_2]| \leq k\epsilon_{\mathsf{IND\text{-}EXP}}$ where $\epsilon_{\mathsf{IND\text{-}EXP}}$ is the advantage in breaking the IND-EXP property of DEN.

In Game 2, the inputs to $\mathcal{A}$ are identical for $b = 0$ and $b = 1$. In fact, for a signer $id \in \mathsf{corL}$ or $id \in \mathsf{L} \setminus (\mathsf{crcL} \cup \mathsf{corL})$ (i.e., the signer is corrupt or honest respectively), the inputs to $\mathcal{A}$ are independent of b. If $id \in \mathsf{crcL}$ (i.e., the signer is coerced), the signer can be input to oracles ADDU, SIGN, COERCE and CRCSIG. When $b = 1$, all signatures generated by algorithms RF.Sign and RF.FakeSign contain an encryption of '1'. Moreover, the fake transcript is updated with the real encryption randomness. As such, the outputs of oracles COERCE and CRCSIG are identical for $b = 0$ and $b = 1$. Therefore, $\Pr[S_2] = 1/2$. We now have that $|\Pr[S_0] - 1/2| \leq k(\epsilon_{\mathsf{IND\text{-}CPA}} + \epsilon_{\mathsf{IND\text{-}EXP}})$ and conclude that the advantage of the adversary in Game 0 is negligible as required.

Lemma 11 (Soundness). RF.SIG *satisfies soundness for a strong receipt-free incoercible signature scheme if deniable encryption scheme* DEN *is correct and signature scheme* SIG *is correct.*

Proof. Consider a signature $\sigma = (\sigma_1, \sigma_2)$ and fake transcript $\mathsf{trans}_{\mathsf{fake}} = \{(r_{\mathsf{SIG}}, pk_S, sk_S), (m, r'_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2)\}$ output by algorithm RF.FakeSign with respect to message m for keys sk_S and pk_A generated by algorithms RF.SKGen and RF.AKGen respectively and parameters pp output by algorithm RF.Setup. Then, by definition of soundness, $\mathsf{RF.Authenticate}(pp, pk_S, sk_A, m, \sigma)$ returns 0

with overwhelming probability. Assume that RF.Authenticate does not return 0. Then, it must be the case that $\mathsf{SIG.Vf}(pp_{\mathsf{SIG}}, pk_S, \mathcal{H}(m \parallel \sigma_1), \sigma_2) = 1$ and $\mathsf{DEN.Dec}(pp_{\mathsf{DEN}}, sk_A, \sigma_1) = 1$. By correctness of the signature scheme, SIG.Vf returns 1 with overwhelming probability. However, deniable encryption scheme DEN.Dec returns 0 where $\sigma_1 \leftarrow \mathsf{DEN.Enc}(pp_{\mathsf{DEN}}, pk_A, 0)$. Then, by contradiction, RF.SIG is sound.

3.4 Efficiency of RF.SIG and CR.SIG

The efficiency of our constructions is determined by the efficiency of the sender-deniable encryption scheme. Since deniable encryption was first introduced in [6], several deniable encryption constructions have been presented in the literature, for example, [1, 5, 8, 10, 15, 31, 34]. Many constructions have improved upon the efficiency of Canetti *et al.*'s constructions [6]. Specifically, [34] proposes an efficient construction of deniable encryption based on indistinguishable obfuscation, and, more recently, an efficient construction has been proposed in the quantum setting [10]. We are hopeful that further advancement in this space can lead to even more efficient incoercible signature schemes.

RF.SIG and CR.SIG use deniable encryption in distinct ways, which leads to efficiency differences. We now provide a brief efficiency comparison of our constructions. With respect to authenticator key generation and public verification, the efficiency of both constructions is identical. Certainly, the efficiency corresponds to the key generation for the deniable encryption scheme and public verification of the underlying signature scheme, respectively. However, our coercion-resistant construction requires additional computation for signer key generation, signing and authentication. In our strong receipt-free construction, the efficiency of the signer's key generation maps directly to the efficiency of key generation for the underlying signature scheme. Our coercion-resistant construction additionally requires the computation of a deniable encryption of a string that is included in the signer's public key. Furthermore, during signing, both constructions require the computation of a single deniable encryption and a signature. However, our strong receipt-free construction only requires an encryption of a single bit, but our coercion-resistant construction requires the encryption of a string. Finally, to authenticate, our strong receipt-free construction requires the decryption of a single bit, whereas our coercion-resistant construction requires the decryption of two ciphertexts that each encrypt a string. Therefore, it is clear that our strong receipt-free construction is more efficient, though this comes at the cost of a weaker notion of security.

3.5 Related Constructions

Our incoercible signature scheme constructions are closely related to the constructions of embedded secret signature scheme constructions presented in [16, 27]. Here, we present a brief comparison of these constructions.

In [16], an embedded secret signature scheme construction is presented is similar to our coercion-resistant construction, and provides identical efficiency in terms of the sizes of signatures and computation during signing, verification and authentication. Indeed, the construction in [16] and both our receipt-free and coercion-resistant constructions use deniable encryption and require that a secret is shared between the authenticator and signer. Nevertheless, our constructions can be distinguished with respect to how the secret is transmitted. The construction in [16] assumes that the authenticator and signer can privately share a secret during key generation. We do not require such an assumption and, indeed, our syntax models key generation as non-interactive. Instead, our constructions allow the signer to generate a secret during key generation (coercion-resistant construction) or simply encrypt a bit during signing (receipt-free construction).

Furthermore, our contributions differ from those in [16] with respect to the security models. Overall, our security model has a similar approach to that of embedded secret signature schemes [16]. In fact, embedded secret signatures must satisfy an indistinguishability requirement and a soundness requirement. However, we distinguish our security model in the following ways. Firstly, our security model captures a spectrum of incoercibility notions that reflects the understanding of incoercibility established in the literature. Secondly, our indistinguishability notions are stronger. More specifically, the security model in [16] captures an indistinguishability experiment similar to our IND1 indistinguishability property. However, the attacker is assumed to only access a real or fake secret key of a coerced signer, rather than a full transcript. By contrast, in our security model, the adversary is given the signer's full transcript. Finally, we highlight that the notion of soundness

introduced in [16], called embedded secret unforgeability, ensures that an attacker with a fake secret key cannot output a signature without an embedded warning. Our notion of coercion-resistance captures a similar soundness property, in addition to a stronger indistinguishability property.

We note that, in [27], a very efficient construction for an embedded secret signature scheme is also given. In fact, this construction is more efficient than both our strong receipt-free and coercion-resistant constructions. However, it does not come with an accompanying security model, and, in fact, does not consider an attacker that demands a signer's secret key. Our constructions, on the other hand, are accompanied with rigorous proofs under suitable security definitions.

4 On Incoercibility and Deniability

Deniability is closely linked to coercion. In fact, both our strong receipt-free and coercion-resistant constructions make use of deniable encryption. This raises the question: how are deniable encryption and incoercible signatures related? In this section, we answer the question by showing that given a weak receipt-free incoercible signature scheme we can build a *partial* deniable encryption scheme. First, we provide a formal definition for a partial deniable encryption scheme. This is a deniable encryption scheme that only encrypts a single bit, i.e., the message space is $\{0,1\}$, and can only explain one of two messages, e.g., the message $m = 0$. That is, given a ciphertext c, $\mathsf{DEN.Exp}$ can only generate randomness such that c appears to encrypt 0, regardless of the message it encrypts. We then show that a secure partial deniable encryption construction can be built from a weak receipt-free incoercible signature scheme. Therefore, we formally show that any construction of a weak receipt-free incoercible signature scheme will either make use of partial deniable encryption as a building block, or (if more efficient than a construction using partial deniable encryption) lead to efficiency improvements for partial deniable encryption. We leave as an open question whether partial deniable encryption schemes can be built more efficiently than standard deniable encryption schemes, leading to efficiency improvements for weak receipt-free constructions.

4.1 Partial Deniable Encryption

We adapt the definition of public-key sender-deniable encryption [6, 34] such that the message space is $\{0,1\}$ and the explanation algorithm no longer takes as input a message, because the only message that can be explained is 0. Additionally, we modify definitions of correctness, $\mathsf{IND\text{-}CPA}$ and $\mathsf{IND\text{-}EXP}$ to the partial deniability setting. In particular, the $\mathsf{IND\text{-}CPA}$ experiment does not require that the adversary output two messages because the only possible messages are 0 and 1. In the $\mathsf{IND\text{-}EXP}$ experiment, as only the message 0 can be explained, the adversary does not output a message.

Definition 15 (Partial Deniable Encryption Scheme). *A* partial deniable encryption scheme (PDEN) *is a tuple of PPT algorithms* ($\mathsf{PDEN.Setup}, \mathsf{PDEN.KGen}, \mathsf{PDEN.Enc}, \mathsf{PDEN.Dec}, \mathsf{PDEN.Exp}$) *such that:*

$\mathsf{PDEN.Setup}(1^\lambda)$ On input of security parameter 1^λ, $\mathsf{PDEN.Setup}$ outputs public parameters pp_{PDEN}.

$\mathsf{PDEN.KGen}(pp_{\mathsf{PDEN}})$ On input of public parameters pp_{PDEN}, $\mathsf{PDEN.KGen}$ outputs a key pair $(pk_{\mathsf{PDEN}}, sk_{\mathsf{PDEN}})$ where pk_{PDEN} is the public encryption key and sk_{PDEN} is the private decryption key.

$\mathsf{PDEN.Enc}(pp_{\mathsf{PDEN}}, pk_{\mathsf{PDEN}}, m)$ On input of public parameters pp_{PDEN}, public key pk_{PDEN} and message $m \in \{0,1\}$, $\mathsf{PDEN.Enc}$ outputs a ciphertext c.

$\mathsf{PDEN.Dec}(pp_{\mathsf{PDEN}}, sk_{\mathsf{PDEN}}, c)$ On input of public parameters pp_{PDEN}, private key sk_{PDEN} and ciphertext c, $\mathsf{PDEN.Dec}$ outputs a message m.

$\mathsf{PDEN.Exp}(pp_{\mathsf{PDEN}}, pk_{\mathsf{PDEN}}, c)$ On input of public parameters pp_{PDEN}, public key pk_{PDEN}, and ciphertext c, $\mathsf{PDEN.Exp}$ outputs a string u.

Definition 16 (Correctness). *A partial deniable encryption scheme* PDEN *satisfies* correctness *if, for any message* $m \in \{0,1\}$*, there exists a negligible function* negl *such that*

$$\Pr\left[\begin{matrix} pp_{\mathsf{PDEN}} \leftarrow \mathsf{PDEN.Setup}(1^\lambda); \\ (pk_{\mathsf{PDEN}}, sk_{\mathsf{PDEN}}) \leftarrow \mathsf{PDEN.KGen}(pp_{\mathsf{PDEN}}); \\ c \leftarrow \mathsf{PDEN.Enc}(pp_{\mathsf{PDEN}}, pk_{\mathsf{PDEN}}, m) \end{matrix} : \begin{matrix} \mathsf{PDEN.Dec}(pp_{\mathsf{PDEN}}, \\ sk_{\mathsf{PDEN}}, c) = m \end{matrix}\right] \geq 1 - \mathsf{negl}(\lambda).$$

Definition 17 (IND-CPA). *A partial deniable encryption scheme* PDEN *satisfies* indistinguishability under a chosen plaintext attack (IND-CPA) *if, for any PPT adversary* $\mathcal{A}$*, there exists a negligible function* negl *such that*

$$\left| \Pr\left[\mathsf{Exp}^{\mathsf{IND\text{-}CPA},0}_{\mathcal{A},\mathsf{PDEN}}(\lambda) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{IND\text{-}CPA},1}_{\mathcal{A},\mathsf{PDEN}}(\lambda) = 1\right] \right| \leq \mathsf{negl}(\lambda)$$

where $\mathsf{Exp}^{\mathsf{IND\text{-}CPA},b}_{\mathcal{A},\mathsf{PDEN}}(\lambda)$ *is the experiment defined in Figure 18 for* $b \in \{0,1\}$.

Definition 18 (IND-EXP). *A partial deniable encryption scheme* PDEN *satisfies* indistinguishability of explanation (IND-EXP) *if, for any PPT adversary* $\mathcal{A}$*, there exists a negligible function* negl *such that*

$$\left| \Pr\left[\mathsf{Exp}^{\mathsf{IND\text{-}EXP},0}_{\mathcal{A},\mathsf{PDEN}}(\lambda) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{IND\text{-}EXP},1}_{\mathcal{A},\mathsf{PDEN}}(\lambda) = 1\right] \right| \leq \mathsf{negl}(\lambda)$$

where $\mathsf{Exp}^{\mathsf{IND\text{-}EXP},b}_{\mathcal{A},\mathsf{PDEN}}(\lambda)$ *is the experiment defined in Figure 18 for* $b \in \{0,1\}$.

$\mathsf{Exp}^{\mathsf{IND\text{-}CPA},b}_{\mathcal{A},\mathsf{PDEN}}(\lambda)$	$\mathsf{Exp}^{\mathsf{IND\text{-}EXP},b}_{\mathcal{A},\mathsf{PDEN}}(\lambda)$
$pp_{\mathsf{PDEN}} \leftarrow \mathsf{PDEN.Setup}(1^\lambda)$	$pp_{\mathsf{PDEN}} \leftarrow \mathsf{PDEN.Setup}(1^\lambda)$
$(pk_{\mathsf{PDEN}}, sk_{\mathsf{PDEN}}) \leftarrow \mathsf{PDEN.KGen}(pp_{\mathsf{PDEN}})$	$(pk_{\mathsf{PDEN}}, sk_{\mathsf{PDEN}}) \leftarrow \mathsf{PDEN.KGen}(pp_{\mathsf{PDEN}})$
$c \leftarrow \mathsf{PDEN.Enc}(pp_{\mathsf{PDEN}}, pk_{\mathsf{PDEN}}, b)$	$c \leftarrow \mathsf{PDEN.Enc}(pp_{\mathsf{PDEN}}, pk_{\mathsf{PDEN}}, 0; u_0)$;
$b' \leftarrow \mathcal{A}(pp_{\mathsf{PDEN}}, pk_{\mathsf{PDEN}}, c)$	$u_1 \leftarrow \mathsf{PDEN.Exp}(pp_{\mathsf{PDEN}}, pk_{\mathsf{PDEN}}, c)$;
return b'	$b' \leftarrow \mathcal{A}(pp_{\mathsf{PDEN}}, pk_{\mathsf{PDEN}}, c, u_b)$
	return b'

Fig. 18: Experiments for indistinguishability under a chosen plaintext attack and indistinguishability of explanation for a partial deniable encryption scheme.

4.2 Constructing a Partial Deniable Encryption Scheme

We show that given an incoercible signature scheme INC-SIG that satisfies weak receipt-freeness, we can build a secure partial deniable encryption scheme PDEN. That is, we show that PDEN (Figure 19) satisfies correctness, IND-CPA and IND-EXP security, as defined in section 4.1, if INC-SIG is a weak receipt-free incoercible signature scheme.

Intuitively, our result holds because a "real" signature can be used in an encryption of 0, and a "fake" signature can be used in an encryption of 1. The authenticator can differentiate between such signatures and so decrypt the ciphertext. Then, when explaining a ciphertext that encrypts 0 (resp., 1) and hence contains a real (resp., fake) signature, the real (resp., fake) transcript can be output. Only a *partial* decryption scheme can be built from incoercible signatures because an incoercible signature does not allow a transcript to be generated such that a ciphertext encrypting 0 and containing a real signature can be explained for message $m = 1$, as if it contains a fake signature.

Correctness. We first show that for $m = 0$, decryption will always return 0. The ciphertext c that encrypts 0 is distributed as follows: $(pk_S, sk_S, \mathsf{trans}) \leftarrow \mathsf{INC\text{-}SIG.SKGen}(pp_{\mathsf{PDEN}}, pk_{\mathsf{PDEN}}; r)$; $(\sigma, \mathsf{trans}) \leftarrow \mathsf{INC\text{-}SIG.Sign}(pp_{\mathsf{PDEN}}, sk_S, pk_{\mathsf{PDEN}}, 0; r')$; $c \leftarrow \mathsf{trans}$. By completeness of INC-SIG, $\mathsf{INC\text{-}SIG.Authenticate}(pp_{\mathsf{PDEN}}, pk_S, sk_{\mathsf{PDEN}}, 0, \sigma) = 0$ with at most negligible probability $\mathsf{negl}(\lambda)$. Therefore, the ciphertext decrypts to 0 with probability at least $1 - \mathsf{negl}(\lambda)$.

We next show that for $m = 1$, decryption will always return 1. The ciphertext c that encrypts 1 is distributed as follows: $(pk_S, sk_S, \mathsf{trans}) \leftarrow \mathsf{INC\text{-}SIG.SKGen}(pp_{\mathsf{PDEN}}, pk_{\mathsf{PDEN}}; r)$; $\mathsf{trans}_{\mathsf{fake}} \leftarrow \mathsf{FakeTrans}(pp_{\mathsf{PDEN}}, pk_{\mathsf{PDEN}}, \mathsf{trans}, \perp)$; $\sigma, \mathsf{trans}_{\mathsf{fake}}) \leftarrow \mathsf{INC\text{-}SIG.FakeSign}(pp_{\mathsf{PDEN}}, sk_S, pk_{\mathsf{PDEN}}, 0, \mathsf{trans}, \mathsf{trans}_{\mathsf{fake}})$; $c \leftarrow \mathsf{trans}_{\mathsf{fake}}$. By soundness of INC-SIG, $\mathsf{INC\text{-}SIG.Authenticate}(pp_{\mathsf{PDEN}}, pk_S, sk_{\mathsf{PDEN}}, 0, \sigma) = 1$ with at most negligible probability $\mathsf{negl}(\lambda)$. Therefore, the ciphertext decrypts to 1 with probability at least $1 - \mathsf{negl}(\lambda)$.

$\mathsf{PDEN.Setup}(1^\lambda)$

$pp_{\mathsf{PDEN}} \leftarrow \mathsf{INC\text{-}SIG.Setup}(1^\lambda)$
return pp_{PDEN}

$\mathsf{PDEN.KGen}(pp_{\mathsf{PDEN}})$

$(pk_{\mathsf{PDEN}}, sk_{\mathsf{PDEN}}) \leftarrow \mathsf{INC\text{-}SIG.AKGen}(pp_{\mathsf{PDEN}})$
return $(pk_{\mathsf{PDEN}}, sk_{\mathsf{PDEN}})$

$\mathsf{PDEN.Dec}(pp_{\mathsf{PDEN}}, sk_{\mathsf{PDEN}}, c)$

parse c as $\{r, pk_S, sk_S, 0, r', \sigma\}$
if $\mathsf{INC\text{-}SIG.Authenticate}(pp_{\mathsf{PDEN}}, pk_S, sk_{\mathsf{PDEN}}, 0, \sigma) = 1$
  return 0
else return 1

$\mathsf{PDEN.Enc}(pp_{\mathsf{PDEN}}, pk_{\mathsf{PDEN}}, m)$

$(pk_S, sk_S, \mathsf{trans}) \leftarrow \mathsf{INC\text{-}SIG.SKGen}(pp_{\mathsf{PDEN}}, pk_{\mathsf{PDEN}}; r)$
if $m = 0$
  $(\sigma, \mathsf{trans}) \leftarrow \mathsf{INC\text{-}SIG.Sign}(pp_{\mathsf{PDEN}}, sk_S, pk_{\mathsf{PDEN}}, 0, \mathsf{trans}; r')$
  $c \leftarrow \mathsf{trans}$
if $m = 1$
  $\mathsf{trans}_{\mathsf{fake}} \leftarrow \mathsf{FakeTrans}(pp_{\mathsf{PDEN}}, pk_{\mathsf{PDEN}}, \mathsf{trans}, \perp)$
  $(\sigma, \mathsf{trans}_{\mathsf{fake}}) \leftarrow \mathsf{INC\text{-}SIG.FakeSign}(pp_{\mathsf{PDEN}}, sk_S, pk_{\mathsf{PDEN}}, 0, \mathsf{trans}, \mathsf{trans}_{\mathsf{fake}})$
  $c \leftarrow \mathsf{trans}_{\mathsf{fake}}$
return c

$\mathsf{PDEN.Exp}(pp_{\mathsf{PDEN}}, pk_{\mathsf{PDEN}}, c)$

parse c as $\{r, pk_S, sk_S, 0, r', \sigma\}$
return (r, r')

Fig. 19: A partial deniable encryption scheme from a weak receipt-free incoercible signature scheme.

IND-CPA *security.* Let $\mathcal{A}$ be an adversary in the $\mathsf{Exp}^{\mathsf{IND\text{-}CPA},b}_{\mathcal{A},\mathsf{PDEN}}(\lambda)$ experiment that is successful with non-negligible probability. We show that we can construct an adversary $\mathcal{A}' = (\mathcal{A}'_1, \mathcal{A}'_2)$ that succeeds in the $\mathsf{Exp}^{\mathsf{IND1},b}_{\mathcal{A}',\mathsf{INC\text{-}SIG}}(\lambda)$ experiment with non-negligible probability. We present $\mathcal{A}'$ in Figure 20. It is clear that inputs to $\mathcal{A}$ are distributed identically to the $\mathsf{Exp}^{\mathsf{IND\text{-}CPA},b}_{\mathcal{A},\mathsf{PDEN}}(\lambda)$ experiment for the bit b chosen in the IND1 indistinguishability experiment. When $b = 0$, trans is distributed identically to the encryption of 0 for the partial deniable encryption scheme PDEN. When $b = 1$, trans is distributed identically to the encryption of 1 in the partial deniable encryption scheme PDEN. Therefore, if $\mathcal{A}$ successfully guesses b in the IND-CPA experiment then $\mathcal{A}'$ will successfully guess b in the IND1 indistinguishability experiment.

$\mathcal{A}'^{\mathsf{ADDU},\mathsf{CRPT},\mathsf{SIGN},\mathsf{CRCSIG}}_1(pp, pk_A)$	$\mathcal{A}'^{\mathsf{COERCE}}_2(st)$
choose any id	$\mathsf{trans} \leftarrow \mathsf{COERCE}(id)$
$pk_S \leftarrow \mathsf{ADDU}(id)$	$b' \leftarrow \mathcal{A}(pp, pk_A, \mathsf{trans})$
$\sigma \leftarrow \mathsf{CRCSIG}(id, 0)$	**return** b'

Fig. 20: Adversary $\mathcal{A}'$ that breaks the IND1 indistinguishability of weak receipt-free incoercible signature scheme INC-SIG given an adversary that can break the IND-CPA security of partial deniable encryption scheme PDEN.

IND-EXP *security.* Indistinguishability of explanation is perfectly satisfied by the PDEN construction. When 0 is encrypted, r, r' is the only randomness chosen and is output in the ciphertext. PDEN.Exp returns this randomness, and inputs to the adversary are independent of b in the IND-EXP security experiment.

5 Incoercible Strong Designated Verifier Signatures

Designated verifier signature schemes [23] allow for signing with respect to a designated verifier. Only the designated verifier is able to verify that the signer generated the signature, and this conviction cannot be transferred to others. This is because the designated verifier is able to simulate signatures with respect to a signer due to the source hiding requirement [35] for such signatures, meaning that all signatures could have been authored by the designated verifier.

In this setting, if the designated verifier can be trusted not to simulate a signature, then that signature must have been authored by the signer. To ensure that signatures cannot be attributed to their signer, even when the designated verifier is trusted not to simulate signatures, this definition was strengthened in *strong* designated verifier signature schemes [23, 28, 33]. This primitive has the

additional requirement that, to all but the designated verifier, a given signature could have been produced by any signer.

As the designated verifier's secret key is necessary to verify signatures, this setting seems well-suited to explore incoerciblity. Indeed, the designated verifier can now take on the role of the authenticator and detect coercion during verification. As signatures are not publicly verifiable, the attacker is not able to detect coercion evasion by verifying signatures. We therefore provide a security model for incoercible strong designated verifier signature schemes and a construction that provably satisfies coercion-resistance.

5.1 Strong Designated Verifier Signature Schemes

Strong designated verifier signature schemes (SDVS) were informally discussed in [23] and the first formal definitions were provided in [33]. In [28], state-of-the-art security definitions for SDVS were provided, based on the model for designated verifier signature schemes given in [35]. SDVS schemes must satisfy *correctness*, *unforgeability*, *source hiding*, *privacy of signer's identity*, as defined in [28].

Definition 19. *A strong designated verifier signature scheme* SDVS *is a tuple of probabilistic polynomial time (PPT) algorithms (*Setup, DVKGen, SKGen, Sign, Verify*) such that:*

$\mathsf{Setup}(1^\lambda)$ On input of 1^λ where λ is a security parameter, algorithm Setup outputs public parameters pp.

$\mathsf{DVKGen}(pp)$ On input of public parameters pp, algorithm DVKGen outputs a public/secret key pair (pk_V, sk_V) of a designated verifier.

$\mathsf{SKGen}(pp)$ On input of public parameters pp, algorithm SKGen outputs a public/secret key pair (pk_S, sk_S) of a signer.

$\mathsf{Sign}(pp, sk_S, pk_V, m)$ On input of public parameters pp, signer's secret key sk_S, public key of the designated verifier pk_V and message m, outputs a signature σ.

$\mathsf{Verify}(pp, pk_S, sk_V, m, \sigma)$ On input of public parameters pp, signer's public key pk_S, secret key of a designated verifier sk_V, message m and signature σ, algorithm Verify outputs 1 if σ verifies, and 0 otherwise.

A strong designated verifier signature scheme must satisfy *correctness*.

Definition 20 (Correctness). *A strong designated verifier signature scheme satisfies* correctness *if, for any message $m \in \{0,1\}^*$, there exists a negligible function* negl *such that*

$$\Pr\left[\begin{array}{l} pp \leftarrow \mathsf{Setup}(1^\lambda); \\ (pk_V, sk_V) \leftarrow \mathsf{DVKGen}(pp); \\ (pk_S, sk_S) \leftarrow \mathsf{SKGen}(pp); \\ \sigma \leftarrow \mathsf{Sign}(pp, sk_S, pk_V, m) \end{array} : \mathsf{Verify}(pp, pk_S, sk_V, m, \sigma) = 1\right] \geq 1 - \mathsf{negl}(\lambda).$$

Source Hiding. Given a message/signature pair, it should be infeasible for any PPT distinguisher to tell whether σ was generated by the signer or simulated by the designated verifier. This ensures that while the designated verifier can verify a signature with respect to a message, they cannot transfer this guarantee.

Definition 21 (Source Hiding). *A strong designated verifier signature scheme is source hiding if there exists a PPT simulation algorithm* Sim*, which takes as input pp, sk_V, pk_V, pk_S and a message m and outputs a simulated signature, such that for any PPT distinguisher $\mathcal{D} = (\mathcal{D}_1, \mathcal{D}_2)$, there exists a negligible function* negl *such that*

$$\Pr\left[\begin{array}{l} b \leftarrow\!\!\$ \{0,1\}, pp \leftarrow \mathsf{Setup}(1^\lambda); \\ (pk_V, sk_V) \leftarrow \mathsf{DVKGen}(pp); \\ (pk_S, sk_S) \leftarrow \mathsf{SKGen}(pp); \\ (m, st) \leftarrow \mathcal{D}_1(pp, pk_S, sk_S, pk_V, sk_V); \\ \sigma_0 \leftarrow \mathsf{Sign}(pp, sk_S, pk_V, m); \\ \sigma_1 \leftarrow \mathsf{Sim}(pp, sk_V, pk_V, pk_S, m); \\ b' \leftarrow \mathcal{D}_2(st, \sigma_b) \end{array} : b' = b\right] \leq 1/2 + \mathsf{negl}(\lambda).$$

Unforgeability. This requires that no one other than the signer or the designated verifier can produce a valid signature. This is sufficient in the designated verifier case, because only the designated verifier can verify the signature.

Definition 22 (Unforgeability). *A strong designated verifier signature scheme satisfies unforgeability if there exists a negligible function* negl *such that*

$$\Pr\left[\begin{array}{l}\mathsf{Query}\leftarrow\emptyset;\\ pp\leftarrow\mathsf{Setup}(1^\lambda);\\ (pk_V,sk_V)\leftarrow\mathsf{DVKGen}(pp);\\ (pk_S,sk_S)\leftarrow\mathsf{SKGen}(pp);\\ (m^*,\sigma^*)\leftarrow\mathcal{A}^{\mathcal{O}}(pp,pk_S,pk_V)\end{array} : \begin{array}{c}\mathsf{Verify}(pp,pk_S,sk_V,m^*,\sigma^*)=1\\ \wedge\ m^*\notin\mathsf{Query}\end{array}\right]\leq\mathsf{negl}(\lambda)$$

where $\mathcal{O}=(\mathsf{SIGN}_{(pp,sk_S,pk_V,\mathsf{Query})}(m)$*, which outputs* $\sigma\leftarrow\mathsf{Sign}(pp,sk_S,pk_V,m)$ *and updates set* Query *to include message* m*,* $\mathsf{SIM}_{(pp,sk_V,pk_V,pk_S,\mathsf{Query})}(m)$*, which outputs* $\sigma\leftarrow\mathsf{Sim}(pp,sk_V,pk_V,pk_S,m)$ *and updates set* Query *to include message* m*,* $\mathsf{VER}_{(pp,pk_S,sk_V,)}(m,\sigma)$*, which outputs* $b\leftarrow\mathsf{Verify}(pp,pk_S,sk_V,m,\sigma)$*).*

We also provide the *strong unforgeability* requirement for strong designated verifier schemes, as this will be necessary when building one of our constructions.

Definition 23 (Strong Unforgeability). *A strong designated verifier signature scheme satisfies strong unforgeability if there exists a negligible function* negl *such that*

$$\Pr\left[\begin{array}{l}\mathsf{Query}\leftarrow\emptyset;\\ pp\leftarrow\mathsf{Setup}(1^\lambda);\\ (pk_V,sk_V)\leftarrow\mathsf{DVKGen}(pp);\\ (pk_S,sk_S)\leftarrow\mathsf{SKGen}(pp);\\ (m^*,\sigma^*)\leftarrow\mathcal{A}^{\mathcal{O}}(pp,pk_S,pk_V)\end{array} : \begin{array}{c}\mathsf{Verify}(pp,pk_S,sk_V,m^*,\sigma^*)=1\\ \wedge\ (m^*,\sigma^*)\notin\mathsf{Query}\end{array}\right]\leq\mathsf{negl}(\lambda)$$

where $\mathcal{O}=(\mathsf{SIGN}_{(pp,sk_S,pk_V,\mathsf{Query})}(m)$*, which outputs* $\sigma\leftarrow\mathsf{Sign}(pp,sk_S,pk_V,m)$ *and updates set* Query *to include* (m,σ)*,* $\mathsf{SIM}_{(pp,sk_V,pk_V,pk_S,\mathsf{Query})}(m)$*, which outputs* $\sigma\leftarrow\mathsf{Sim}(pp,sk_V,pk_V,pk_S,m)$ *and updates set* Query *to include* (m,σ)*,* $\mathsf{VER}_{(pp,pk_S,sk_V,)}(m,\sigma)$*, which outputs* $b\leftarrow\mathsf{Verify}(pp,pk_S,sk_V,m,\sigma)$*).*

Privacy of Signer's Identity. First defined in [28], this requirement ensures that it is not possible to distinguish between signatures generated by signer S_0 for designated verifier V and signatures generated by signer S_1 for designated verifier V, without knowledge of the secret key of V, sk_V. This requirement is the difference between a strong and standard designated verifier signature schemes.

$\underline{\mathsf{SIGN}_{(pp,sk_{S_0},sk_{S_1},pk_V)}(m,d)}$
$\sigma\leftarrow\mathsf{Sign}(pp,sk_{S_d},pk_V,m)$
return σ

$\underline{\mathsf{SIM}_{(pp,sk_V,pk_V,pk_{S_0},pk_{S_1})}(m,d)}$
$\sigma\leftarrow\mathsf{Sim}(pp,sk_V,pk_V,pk_{S_d},m)$
return σ

$\underline{\mathsf{VER}_{(pp,pk_{S_0},pk_{S_1},sk_V,m^*,\sigma^*)}(m,\sigma,d)}$
if $(m,\sigma)=(m^*,\sigma^*)$ **return** $\perp$
$b\leftarrow\mathsf{Verify}(pp,pk_{S_d},sk_V,m,\sigma)$
return b

$\underline{\mathsf{Exp}^{\mathsf{PSI},b}_{\mathcal{A},\mathsf{SDVS}}(\lambda)}$
$pp\leftarrow\mathsf{Setup}(1^\lambda)$
$(pk_V,sk_V)\leftarrow\mathsf{DVKGen}(pp)$
$(pk_{S_0},sk_{S_0})\leftarrow\mathsf{SKGen}(pp)$
$(pk_{S_1},sk_{S_1})\leftarrow\mathsf{SKGen}(pp)$
$(m^*,st)\leftarrow\mathcal{A}_1^{\mathsf{SIGN},\mathsf{VER},\mathsf{SIM}}(pp,pk_{S_0},pk_{S_1},pk_V)$
$\sigma^*\leftarrow\mathsf{Sign}(pp,sk_{S_b},pk_V,m^*)$
$b'\leftarrow\mathcal{A}_2^{\mathsf{SIGN},\mathsf{VER},\mathsf{SIM}}(\sigma^*,st)$
return b'

Fig. 21: Experiment capturing the privacy of signer's identity requirement for strong designated verifier signature schemes.

Definition 24 (Privacy of the Signer's Identity). *A strong designated verifier signature scheme* SDVS *satisfies privacy of the signer's identity if, for any PPT adversary* $\mathcal{A}=(\mathcal{A}_1,\mathcal{A}_2)$*, there exists a negligible function* negl *such that*

$$\left|\Pr\left[\mathsf{Exp}^{\mathsf{PSI},0}_{\mathcal{A},\mathsf{SDVS}}(\lambda)=1\right]-\Pr\left[\mathsf{Exp}^{\mathsf{PSI},1}_{\mathcal{A},\mathsf{SDVS}}(\lambda)=1\right]\right|\leq\mathsf{negl}(\lambda)$$

where $\mathsf{Exp}^{\mathsf{PSI},b}_{\mathcal{A},\mathsf{SDVS}}(\lambda)$ *is the experiment defined in Figure 21 for* $b\in\{0,1\}$*.*

5.2 Incoercible Strong Designated Verifier Signature Schemes

We base our syntax and security model for incoercible strong designated verifier schemes on [28]. Unlike standard signatures, our syntax in the designated verifier setting does not need the Authenticate algorithm because the designated verifier can detect coercion during verification. This is because a secret key is now required to verify a signature and so the attacker cannot detect that a signer has evaded coercion. Unlike in standard SDVS syntax, the signers' secret keys must be bound to a particular designated verifier. Otherwise, the secret key can be used to sign with respect to any designated verifier, and so, clearly, the attacker can detect a fake key by signing with their own designated verifier public key.

Definition 25. *An incoercible strong designated verifier signature scheme* INC-SDVS *is a tuple of probabilistic polynomial time (PPT) algorithms (*Setup, DVKGen, SKGen, FakeTrans, Sign, FakeSign, Verify*) such that:*

$\mathsf{Setup}(1^\lambda)$ On input of 1^λ where λ is a security parameter, outputs public parameters pp.
$\mathsf{DVKGen}(pp)$ Outputs a public/secret key pair (pk_V, sk_V) of a designated verifier.
$\mathsf{SKGen}(pp, pk_V)$ Outputs a public/secret key pair (pk_S^V, sk_S^V) of a signer with respect to a designated verifier with public key pk_V and an initial transcript trans.
$\mathsf{FakeTrans}(pp, pk_V, \mathsf{trans}, \mathsf{trans}_{\mathsf{fake}})$ On input of pp, pk_V, a transcript trans and a fake transcript $\mathsf{trans}_{\mathsf{fake}}$, outputs an updated fake transcript $\mathsf{trans}_{\mathsf{fake}}$. As before $\mathsf{trans}_{\mathsf{fake}} \leftarrow \mathsf{FakeTrans}(pp, pk_V, \mathsf{trans}, \perp)$ generates an initial fake transcript.
$\mathsf{Sign}(pp, sk_S^V, pk_V, m, \mathsf{trans})$ On input of pp, sk_S^V, pk_V, message m and transcript trans, outputs a signature σ and an updated transcript trans.
$\mathsf{FakeSign}(pp, sk_S^V, pk_V, m, \mathsf{trans}, \mathsf{trans}_{\mathsf{fake}})$ On input of pp, sk_S^V, pk_V, message m, the signer's transcript trans and a signer's fake transcript $\mathsf{trans}_{\mathsf{fake}}$, outputs a signature σ and updated fake transcript $\mathsf{trans}_{\mathsf{fake}}$.
$\mathsf{Verify}(pp, pk_S^V, sk_V, m, \sigma)$ On input of pp, pk_S^V, sk_V, message m and signature σ, outputs 1 if σ verifies, and 0 otherwise.

Security Model Incoercible strong designated verifier signature schemes must satisfy correctness, source hiding, unforgeability and privacy of the signer's identity. Additionally, they must satisfy *incoercibility*, whether weak/ strong receipt-freeness or coercion-resistance. These requirements are defined similarly to those for the publicly verifiable schemes, adapted to the strong designated verifier setting.

Definition 26 (Correctness). *An incoercible strong designated verifier signature scheme satisfies* correctness *if, for any message $m \in \{0,1\}^*$, there exists a negligible function* negl *such that*

$$\Pr\left[\begin{array}{l} pp \leftarrow \mathsf{Setup}(1^\lambda); \\ (pk_V, sk_V) \leftarrow \mathsf{DVKGen}(pp); \\ (pk_S^V, sk_S^V, \mathsf{trans}) \leftarrow \mathsf{SKGen}(pp, pk_V); \\ (\sigma, \mathsf{trans}) \leftarrow \mathsf{Sign}(pp, sk_S^V, pk_V, m, \mathsf{trans}) \end{array} : \mathsf{Verify}(pp, pk_S^V, sk_V, m, \sigma) = 1\right] \geq 1 - \mathsf{negl}(\lambda).$$

Definition 27 (Source Hiding). *An incoercible strong designated verifier signature scheme is source hiding if there exists a PPT simulation algorithm* Sim*, which takes as input pp, sk_V, pk_V, pk_S^V and a message m and outputs a simulated signature, such that for any PPT distinguisher $\mathcal{D} = (\mathcal{D}_1, \mathcal{D}_2)$, there exists a negligible function* negl *such that*

$$\Pr\left[\begin{array}{l} b \leftarrow\$ \{0,1\}, pp \leftarrow \mathsf{Setup}(1^\lambda); \\ (pk_V, sk_V) \leftarrow \mathsf{DVKGen}(pp); \\ (pk_S^V, sk_S^V, \mathsf{trans}) \leftarrow \mathsf{SKGen}(pp, pk_V); \\ (m, st) \leftarrow \mathcal{D}_1(pp, pk_S^V, sk_S^V, pk_V, sk_V); \\ (\sigma_0, \mathsf{trans}) \leftarrow \mathsf{Sign}(pp, sk_S^V, pk_V, m, \mathsf{trans}); \\ \sigma_1 \leftarrow \mathsf{Sim}(pp, sk_V, pk_V, pk_S^V, m); \\ b' \leftarrow \mathcal{D}_2(st, \sigma_b) \end{array} : b' = b\right] \leq 1/2 + \mathsf{negl}(\lambda).$$

Definition 28 (Unforgeability). *An incoercible strong designated verifier signature scheme satisfies unforgeability if there exists a negligible function* negl *such that*

$$\Pr\left[\begin{array}{l} \mathsf{Query} \leftarrow \emptyset; \\ pp \leftarrow \mathsf{Setup}(1^\lambda); \\ (pk_V, sk_V) \leftarrow \mathsf{DVKGen}(pp); \\ (pk_S^V, sk_S^V, \mathsf{trans}) \leftarrow \mathsf{SKGen}(pp, pk_V); \\ (m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}}(pp, pk_S^V, pk_V) \end{array} : \mathsf{Verify}(pp, pk_S^V, sk_V, m^*, \sigma^*) = 1 \wedge m^* \notin \mathsf{Query}\right] \leq \mathsf{negl}(\lambda)$$

where $\mathcal{O} = (\mathsf{SIGN}_{(pp,sk_S^V,pk_V,\mathsf{Query})}(m)$*, which computes* $(\sigma, \mathsf{trans}) \leftarrow \mathsf{Sign}(pp, sk_S^V, pk_V, m, \mathsf{trans})$*, outputs* σ *and updates set* Query *to include message* m*,* $\mathsf{SIM}_{(pp,sk_V,pk_V,pk_S^V,\mathsf{Query})}(m)$*, which outputs* $\sigma \leftarrow \mathit{Sim}(pp, sk_V, pk_V, pk_S^V, m)$ *and updates set* Query *to include message* m*,* $\mathsf{VER}_{(pp,pk_S^V,sk_V,)}(m, \sigma)$*, which outputs* $b \leftarrow \mathsf{Verify}(pp, pk_S^V, sk_V, m, \sigma)$*).*

$\underline{\mathsf{SIGN}_{(pp,sk_{S_0}^V,sk_{S_1}^V,pk_V)}(m,d)}$
$(\sigma, \mathsf{trans}_d) \leftarrow \mathsf{Sign}(pp, sk_{S_d}^V, pk_V, m, \mathsf{trans}_d)$
return σ

$\underline{\mathsf{SIM}_{(pp,sk_V,pk_V,pk_{S_0}^V,pk_{S_1}^V)}(m,d)}$
$\sigma \leftarrow \mathsf{Sim}(pp, sk_V, pk_V, pk_{S_d}^V, m)$
return σ

$\underline{\mathsf{VER}_{(pp,pk_{S_0}^V,pk_{S_1}^V,sk_V,m^*,\sigma^*)}(m,\sigma,d)}$
if $(m,\sigma) = (m^*,\sigma^*)$ **return** $\perp$
$b \leftarrow \mathsf{Verify}(pp, pk_{S_d}^V, sk_V, m, \sigma)$
return b

$\underline{\mathsf{Exp}^{\mathsf{PSI},b}_{\mathcal{A},\mathsf{INC\text{-}SDVS}}(\lambda)}$
$pp \leftarrow \mathsf{Setup}(1^\lambda)$
$(pk_V, sk_V) \leftarrow \mathsf{DVKGen}(pp)$
$(pk_{S_0}^V, sk_{S_0}^V, \mathsf{trans}_0) \leftarrow \mathsf{SKGen}(pp, pk_V)$
$(pk_{S_1}^V, sk_{S_1}^V, \mathsf{trans}_1) \leftarrow \mathsf{SKGen}(pp, pk_V)$
$(m^*, st) \leftarrow \mathcal{A}_1^{\mathsf{SIGN},\mathsf{VER},\mathsf{SIM}}(pp, pk_{S_0}^V, pk_{S_1}^V, pk_V)$
$(\sigma^*, \mathsf{trans}_b) \leftarrow \mathsf{Sign}(pp, sk_{S_b}^V, pk_V, m^*, \mathsf{trans}_b)$
$b' \leftarrow \mathcal{A}_2^{\mathsf{SIGN},\mathsf{VER},\mathsf{SIM}}(\sigma^*, st)$
return b'

Fig. 22: Experiment capturing the privacy of signer's identity requirement for incoercible strong designated verifier signature schemes.

Definition 29 (Privacy of the Signer's Identity). *An incoercible strong designated verifier signature scheme* INC-SDVS *satisfies privacy of the signer's identity if, for any PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, there exists a negligible function* negl *such that*

$$\left| \Pr\left[\mathsf{Exp}^{\mathsf{PSI},0}_{\mathcal{A},\mathsf{INC\text{-}SDVS}}(\lambda) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{PSI},1}_{\mathcal{A},\mathsf{INC\text{-}SDVS}}(\lambda) = 1\right] \right| \leq \mathsf{negl}(\lambda)$$

where $\mathsf{Exp}^{\mathsf{PSI},b}_{\mathcal{A},\mathsf{INC\text{-}SDVS}}(\lambda)$ *is the experiment defined in Figure 22 for* $b \in \{0,1\}$*.*

Incoercibility. As for publicly verifiable signature schemes, we present three variants of incoercibility: weak receipt-freeness, strong receipt-freeness and coercion-resistance. In Figure 23, we define a number of parameterised oracles for our security experiments.

Weak Receipt-freeness As in the publicly verifiable case, we present a definition of weak receipt-freeness that captures two properties: indistinguishability (IND1) and soundness.

Definition 30 (Weak Receipt-Freeness). *An incoercible strong designated verifier signature scheme* INC-SDVS *satisfies* weak receipt-freeness *if the following conditions hold:*

- ***Indistinguishability*** (IND1)*: for any PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, there exists a negligible function* negl *such that*

$$\left| \Pr\left[\mathsf{Exp}^{\mathsf{IND1},0}_{\mathcal{A},\mathsf{INC\text{-}SDVS}}(\lambda) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{IND1},1}_{\mathcal{A},\mathsf{INC\text{-}SDVS}}(\lambda) = 1\right] \right| \leq \mathsf{negl}(\lambda).$$

- ***Soundness****: for any message* $m \in \{0,1\}^*$*, there exists a negligible function* negl *such that*

$$\Pr\left[\mathsf{Exp}^{\mathsf{sound}}_{\mathsf{INC\text{-}SDVS}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda).$$

where $\mathsf{Exp}^{\mathsf{IND1},b}_{\mathcal{A},\mathsf{INC\text{-}SDVS}}(\lambda)$ *for* $b \in \{0,1\}$ *and* $\mathsf{Exp}^{\mathsf{sound}}_{\mathsf{INC\text{-}SDVS}}(\lambda)$ *are the experiments defined in Figure 24.*

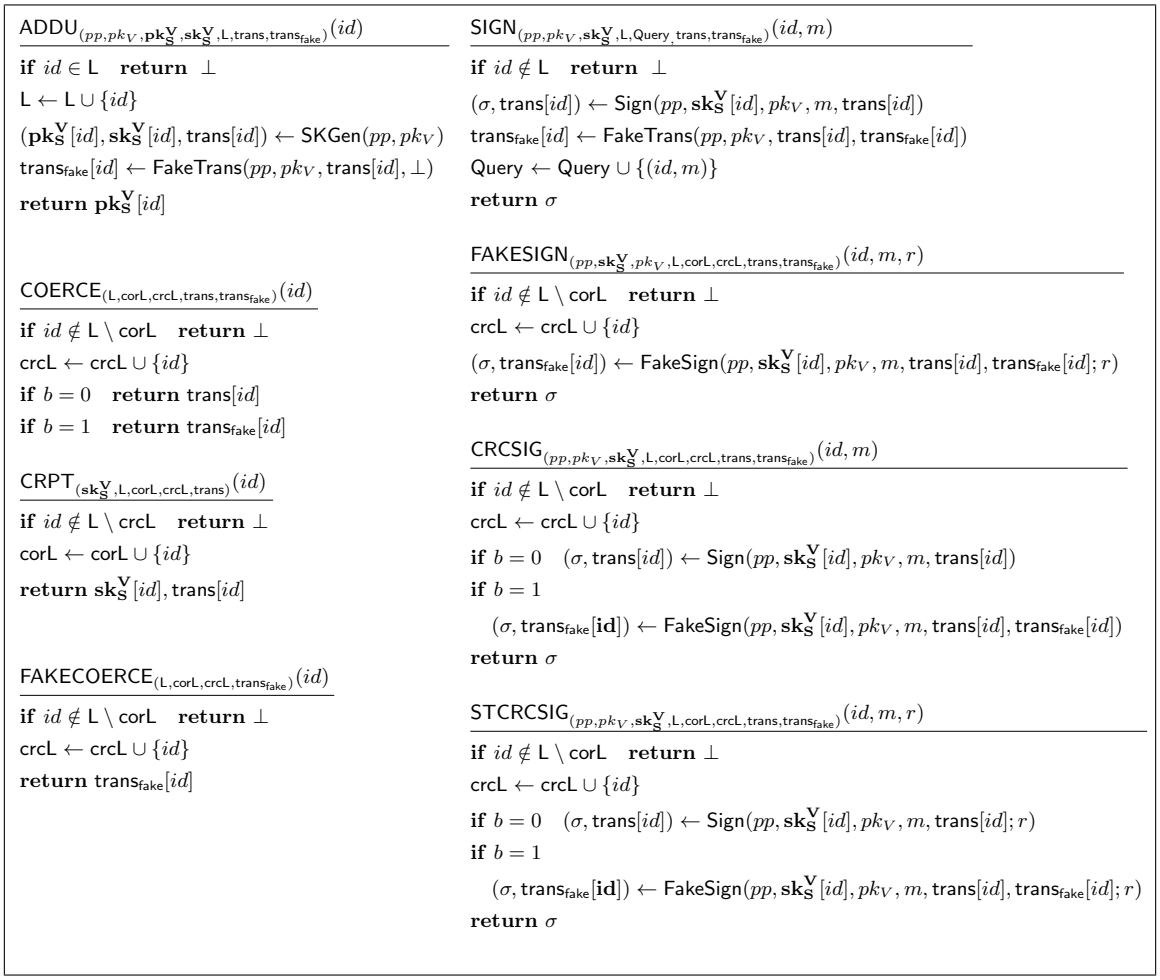

Fig. 23: Oracles used in the experiments for weak receipt-freeness, strong receipt-freeness and coercion-resistance defined in Figure 24.

Strong Receipt-Freeness As in the publicly verifiable case, we present a definition of strong receipt-freeness that captures two properties: (IND2) indistinguishability and soundness.

Definition 31 (Strong Receipt-Freeness). *An incoercible strong designated verifier signature scheme* INC-SDVS *satisfies* strong receipt-freeness *if the following conditions hold:*

- ***Indistinguishability*** (IND2)*: for any PPT adversary $\mathcal{A}$, there exists a negligible function* negl *such that*
$$\left| \Pr\left[\mathsf{Exp}^{\mathsf{IND2},0}_{\mathcal{A},\mathsf{INC\text{-}SDVS}}(\lambda) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{IND2},1}_{\mathcal{A},\mathsf{INC\text{-}SDVS}}(\lambda) = 1\right] \right| \leq \mathsf{negl}(\lambda).$$
- ***Soundness****: for any message $m \in \{0,1\}^*$, there exists a negligible function* negl *such that*
$$\Pr\left[\mathsf{Exp}^{\mathsf{sound}}_{\mathsf{INC\text{-}SDVS}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda).$$

where $\mathsf{Exp}^{\mathsf{IND2},b}_{\mathcal{A},\mathsf{INC\text{-}SDVS}}(\lambda)$ *for* $b \in \{0,1\}$ *and* $\mathsf{Exp}^{\mathsf{sound}}_{\mathsf{INC\text{-}SDVS}}(\lambda)$ *are the experiments defined in Figure 24.*

Coercion-Resistance As in the publicly verifiable case, our definition of *coercion–resistance* is captured with two properties: (IND3) indistinguishability and strong soundness.

Definition 32 (Coercion-Resistance). *An incoercible strong designated verifier signature scheme* INC-SDVS *satisfies* coercion-resistance *if the following conditions hold:*

- ***Indistinguishability*** (IND3)*: For any PPT adversary $\mathcal{A}$, there exists a negligible function* negl *such that*
$$\left| \Pr\left[\mathsf{Exp}^{\mathsf{IND3},0}_{\mathcal{A},\mathsf{INC\text{-}SDVS}}(\lambda) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{IND3},1}_{\mathcal{A},\mathsf{INC\text{-}SDVS}}(\lambda) = 1\right] \right| \leq \mathsf{negl}(\lambda).$$

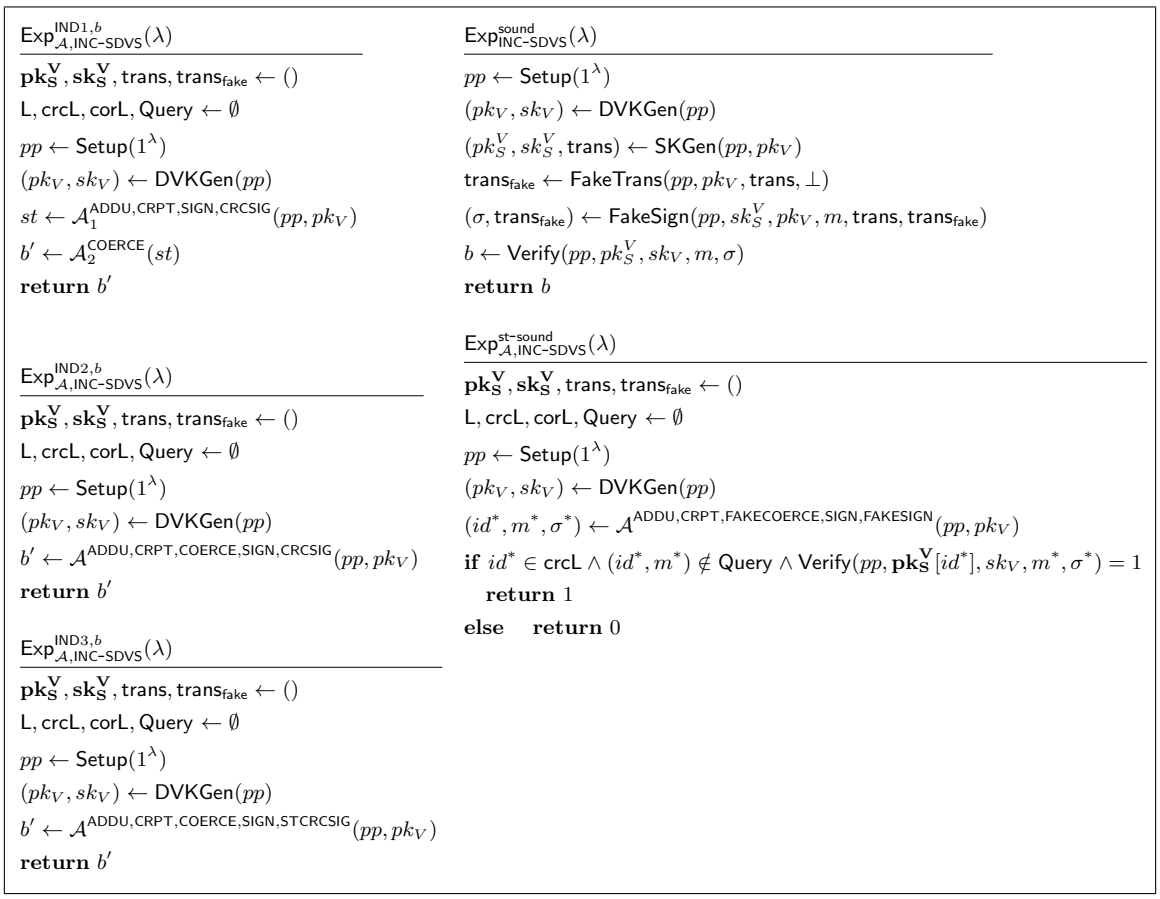

Fig. 24: Experiments for weak receipt-freeness, strong receipt-freeness and coercion-resistance where the adversary has access to oracles defined in Figure 23.

– ***Strong soundness****: for any PPT adversary $\mathcal{A}$, there exists a negligible function* negl *such that*

$$\Pr\left[\mathsf{Exp}^{\mathsf{st\text{-}sound}}_{\mathcal{A},\mathsf{INC\text{-}SDVS}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda).$$

where $\mathsf{Exp}^{\mathsf{IND3},b}_{\mathcal{A},\mathsf{INC\text{-}SDVS}}(\lambda)$ *for* $b \in \{0,1\}$ *and* $\mathsf{Exp}^{\mathsf{st\text{-}sound}}_{\mathcal{A},\mathsf{INC\text{-}SDVS}}(\lambda)$ *are the experiments defined in Figure 24.*

Relation Between Incoercibility and Non–Delegatability. In [29] the]non–delegatability requirement was introduced for SDVS. This ensures a signer cannot delegate their signing rights to another entity. Intuitively, this requires that if an adversary produces a valid signature for signer S and designated verifier V, then it must know either the secret key for S or for V. Although non-delegatability and coercion–resistance seem related at first glance, they address different attack models. Coercion-resistance ensures that coerced signatures are detected, without the attacker discovering that the signer evaded coercion. Non-delegatability prevents a signer that is happy to delegate their signing rights from doing so without revealing their signing key. In both cases delegation is prevented. For coercion-resistance the signer does not wish to delegate their signing rights but is being bribed or blackmailed, whereas for non–delegatability they do wish to delegate their signing rights but not reveal their secret key.

Relation to Deniability. In Section 1.1 we discuss how strong designated verifier signature schemes satisfy elements of deniability. Indeed, in an SDVS, if a signer is being coerced by an attacker *not* to sign a message of their choice, the signer can simply sign the message anyway and claim that it was authored by a different signer or by the designated verifier. Our incoercible strong designated verifier signature schemes inherit this deniability property, as well as provide resistance to the coercive attacks captured by incoercibility, whereby an attacker instructs a signer to sign a message of their choice.

5.3 A Coercion-Resistant Construction

We present in Figure 25 our CR.SDVS construction that satisfies our coercion-resistant security model for incoercible strong designated verifier signature schemes. We use as building blocks a strong designated verifier signature scheme SDVS and a deniable encryption scheme DEN. Additionally, our constructions uses two hash functions $\mathcal{H}_1 : \{0,1\}^* \to \{0,1\}^*$, $\mathcal{H}_2 : \{0,1\}^* \to \mathcal{M}$, where $\mathcal{M}$ is the message space of the strong designated verifier signature scheme SDVS. The first is assumed to model a random oracle, whereas the second is required to be collision resistant.

The idea behind the construction is similar to that used in the public verification setting, but with the strong designated verifier signature scheme SDVS replacing the standard signature scheme SIG. The SDVS building block ensures the properties necessary for a strong designated verifier signature scheme still hold. Coercion-resistance is provided using a similar argument as for the publicly verifiable construction. We note that the SDVS scheme must satisfy *strong unforgeability* [36], to ensure that our construction satisfies the privacy of signer's identity requirement.

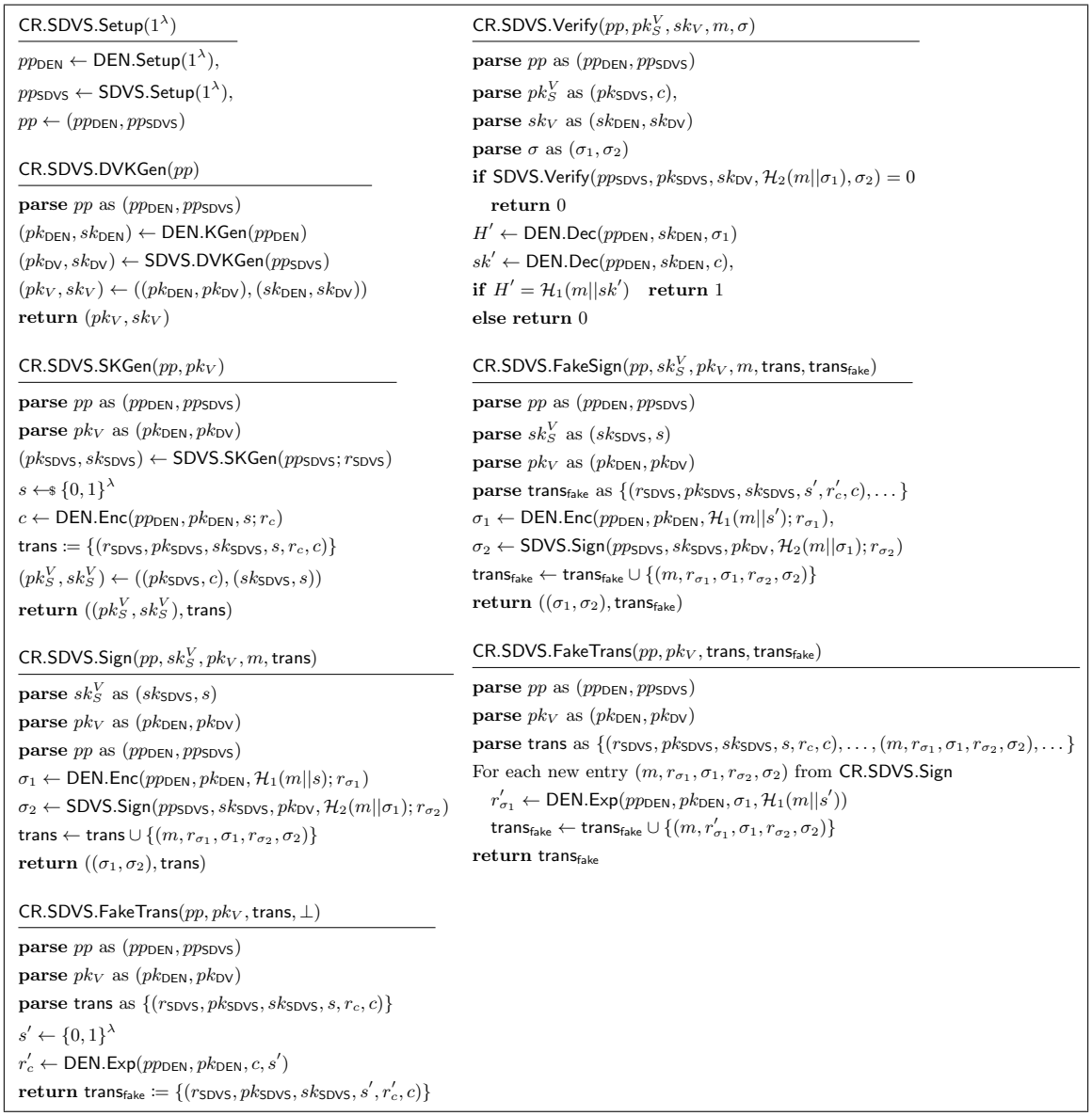

Fig. 25: Our coercion-resistant construction CR.SDVS.

Theorem 3. *Let* SDVS *and* DEN *be a secure strong designated verifier signature scheme and deniable encryption scheme respectively, and the hash functions $\mathcal{H}_1$ and $\mathcal{H}_2$ be modelled as a random oracle model and collision resistant respectively. Then,* CR.SDVS *is a secure construction of a coercion-resistant incoercible strong designated verifier signature scheme. That is,* CR.SDVS *satisfies correctness, source hiding, unforgeability, privacy of the signer's identity and coercion-resistance.*

Lemma 12 (Correctness). CR.SDVS *satisfies* correctness *if strong designated verifier signature scheme* SDVS *and deniable encryption scheme* DEN *are correct.*

Proof. Consider a signature $\sigma = (\sigma_1, \sigma_2)$ output by algorithm CR.SDVS.Sign with respect to message m for keys sk_S^V and pk_V generated by algorithms CR.SDVS.SKGen and CR.SDVS.DVKGen respectively and parameters pp output by algorithm CR.SDVS.Setup. Then, by definition of correctness, CR.SDVS is correct if $\mathsf{CR.SDVS.Verify}(pp, pk_S^V, sk_V, m, \sigma)$, outputs 1 with overwhelming probability. Assume that CR.SDVS.Verify does not return 1. Then, either $\mathsf{SDVS.Vf}(pp_{\mathsf{SDVS}}, pk_{\mathsf{SDVS}}, sk_{\mathsf{DV}}, \mathcal{H}_2(m||\sigma_1), \sigma_2) = 0$ or $\mathsf{DEN.Dec}(pp_{\mathsf{DEN}}, sk_{\mathsf{DEN}}, \sigma_1) \neq \mathcal{H}_1(m||\mathsf{DEN.Dec}(pp_{\mathsf{DEN}}, sk_{\mathsf{DEN}}, c))$. By assumption, SDVS is correct. Therefore, algorithm SDVS.Vf returns 1 with overwhelming probability where pp_{SDVS}, pk_{SDVS}, sk_{DV} are generated according to the construction description and σ_2 is the output of SDVS.Sign for message $\mathcal{H}_2(m||\sigma_1)$. Moreover, by correctness of the deniable encryption scheme, $\mathsf{DEN.Dec}(pp_{\mathsf{DEN}}, sk_{\mathsf{DEN}}, c)$ returns s, and $\mathsf{DEN.Dec}(pp_{\mathsf{DEN}}, sk_{\mathsf{DEN}}, \sigma_1)$ returns $\mathcal{H}_1(m||s)$, where pp_{DEN}, sk_{DEN} are generated according to the construction description, σ_1 is the output of DEN.Enc for message $\mathcal{H}_1(m||s)$ and c is the output of DEN.Enc for message s. Then, by contradiction, CR.SDVS is correct.

Lemma 13 (Source Hiding). CR.SDVS *satisfies* source hiding *if strong designated verifier signature scheme* SDVS *is source hiding.*

Proof. As the strong designated verifier signature scheme is source hiding, there exists a PPT simulator $\mathsf{Sim}_{\mathsf{SDVS}}$ that outputs signatures that are indistinguishable from those output by SDVS.Sign. We now give a simulator for our construction:

$\mathsf{Sim}(pp, sk_V, pk_V, pk_S^V, m)$

parse pp as $(pp_{\mathsf{DEN}}, pp_{\mathsf{SDVS}})$, sk_V as $(sk_{\mathsf{DEN}}, sk_{\mathsf{DV}})$,
pk_V as $(pk_{\mathsf{DEN}}, pk_{\mathsf{DV}})$, pk_S^V as (pk_{SDVS}, c)
$\sigma_1 \leftarrow \mathsf{DEN.Enc}(pp_{\mathsf{DEN}}, pk_{\mathsf{DEN}}, \mathcal{H}_1(m||\mathsf{DEN.Dec}(pp_{\mathsf{DEN}}, sk_{\mathsf{DEN}}, c)))$
$\sigma_2 \leftarrow \mathsf{Sim}_{\mathsf{SDVS}}(pp_{\mathsf{SDVS}}, sk_{\mathsf{DV}}, pk_{\mathsf{DV}}, pk_{\mathsf{SDVS}}, \mathcal{H}_2(m||\sigma_1))$
return (σ_1, σ_2)

Let $\Pr\left[\mathsf{Exp}^{source-hide}_{\mathcal{D},\mathsf{CR.SDVS}}(\lambda) = 1\right]$ be the probability that the distinguisher $\mathcal{D}$ guesses correctly in the source hiding experiment for our construction. We show that if there exists a distinguisher $\mathcal{D} = (\mathcal{D}_1, \mathcal{D}_2)$ such that $\Pr\left[\mathsf{Exp}^{source-hide}_{\mathcal{D},\mathsf{CR.SDVS}}(\lambda) = 1\right] - 1/2 = \epsilon$, where ϵ is non-negligible, then we can can build a distinguisher $\mathcal{D}'$ that breaks the source hiding of the SDVS scheme with non-negligible probability. We give the detailed description of $\mathcal{D}' = (\mathcal{D}'_1, \mathcal{D}'_2)$ in Figure 26, and explain here how $\mathcal{D}'$ works.

Let b be the bit that $\mathcal{D}'$ aims to guess in the source hiding game for the strong designated verifier signature scheme. We show that all inputs that $\mathcal{D}'$ provides to $\mathcal{D}$ are distributed identically to in the source hiding game with bit b for the CR.SDVS construction.

The tuple $(pp, pk_S^V, sk_S^V, pk_V, sk_V)$ is clearly distributed identically to in the source hiding game as they are generated honestly. If $b = 0$, σ_1 is computed identically to in CR.SDVS.Sign and σ_2 is a strong designated verifier signature on $\mathcal{H}_2(m||\sigma_1)$. If $b = 1$, because $\mathsf{DEN.Dec}(pp_{\mathsf{DEN}}, sk_{\mathsf{DEN}}, c) = s$, σ_1 is also distributed identically to the output of Sim. σ_2 is the output of $\mathsf{Sim}_{\mathsf{SDVS}}$ on message $\mathcal{H}_2(m||\sigma_1)$. Therefore, (σ_1, σ_2) are distributed identically in the source hiding experiment. Therefore, if $\mathcal{D}$ successfully guesses b then $\mathcal{D}'$ will also successfully guess b and so break the source hiding of SDVS. Therefore, by contradiction, CR.SDVS satisfies source hiding.

Lemma 14 (Unforgeability). CR.SDVS *satisfies* unforgeability *if strong designated verifier signature scheme* SDVS *satisfies unforgeability and the hash function $\mathcal{H}_2$ is collision resistant.*

$\mathcal{D'}_1(pp_{\mathsf{SDVS}}, pk_{\mathsf{SDVS}}, sk_{\mathsf{SDVS}}, pk_{\mathsf{DV}}, sk_{\mathsf{DV}})$

$pp_{\mathsf{DEN}} \leftarrow \mathsf{DEN.Setup}(1^\lambda)$
$pp \leftarrow (pp_{\mathsf{DEN}}, pp_{\mathsf{SDVS}})$
$(pk_{\mathsf{DEN}}, sk_{\mathsf{DEN}}) \leftarrow \mathsf{DEN.KGen}(pp_{\mathsf{DEN}})$
$(pk_V, sk_V) \leftarrow ((pk_{\mathsf{DEN}}, pk_{\mathsf{DV}}), (sk_{\mathsf{DEN}}, sk_{\mathsf{DV}}))$
$s \leftarrow\!\!\$\ \{0,1\}^\lambda$
$c \leftarrow \mathsf{DEN.Enc}(pp_{\mathsf{DEN}}, pk_{\mathsf{DEN}}, s)$
$(pk_S^V, sk_S^V) \leftarrow ((pk_{\mathsf{SDVS}}, c), (sk_{\mathsf{SDVS}}, s))$
$(st, m) \leftarrow \mathcal{D}_1(pp, pk_S^V, sk_S^V, pk_V, sk_V)$
$\sigma_1 \leftarrow \mathsf{DEN.Enc}(pp_{\mathsf{DEN}}, pk_{\mathsf{DEN}}, \mathcal{H}_1(m||s))$
return $(st', \mathcal{H}_2(m||\sigma_1))$

$\mathcal{D'}_2(st', \sigma_2)$

return $\mathcal{D}_2(st, (\sigma_1, \sigma_2))$

Fig. 26: $\mathcal{D}'$ which breaks the source hiding of SDVS.

Proof. We build an adversary $\mathcal{A}'$ that successfully wins the unforgeability game for SDVS, given $\mathcal{A}$ that wins the unforgeability game for the CR.SDVS construction. We give $\mathcal{A}'$ in Figure 27, and below explain why the simulation input to $\mathcal{A}$ is identically distributed to the unforgeability game for the CR.SDVS construction, and that $\mathcal{A}'$ successfully breaks the unforgeability of SDVS.

$\mathcal{O}_{\mathsf{SIG}}(m)$

$\sigma_1 \leftarrow \mathsf{DEN.Enc}(pp_{\mathsf{DEN}}, pk_{\mathsf{DEN}}, \mathcal{H}_1(m||s))$
$\sigma_2 \leftarrow \mathcal{O}_{\mathsf{SIG}}^{\mathsf{SDVS}}(\mathcal{H}_2(m||\sigma_1))$
$\mathsf{Query} \leftarrow \mathsf{Query} \cup \{m\}$
return (σ_1, σ_2)

$\mathcal{O}_{\mathsf{VER}}(m, (\sigma_1, \sigma_2)$

if $\mathcal{O}_{\mathsf{VER}}^{\mathsf{SDVS}}(\mathcal{H}_2(m||\sigma_1), \sigma_2) = 0$ **return** 0
$H' \leftarrow \mathsf{DEN.Dec}(pp_{\mathsf{DEN}}, sk_{\mathsf{DEN}}, \sigma_1)$
$sk' \leftarrow \mathsf{DEN.Dec}(pp_{\mathsf{DEN}}, sk_{\mathsf{DEN}}, c)$,
if $H' = \mathcal{H}_1(m||sk')$ **return** 1
else return 0

$\mathcal{A'}^{\mathcal{O}_{\mathsf{SIG}}^{\mathsf{SDVS}}, \mathcal{O}_{\mathsf{SIM}}^{\mathsf{SDVS}}, \mathcal{O}_{\mathsf{VER}}^{\mathsf{SDVS}}}(pp_{\mathsf{SDVS}}, pk_{\mathsf{SDVS}}, pk_{\mathsf{DV}})$

$pp_{\mathsf{DEN}} \leftarrow \mathsf{DEN.Setup}(1^\lambda)$
$pp \leftarrow (pp_{\mathsf{DEN}}, pp_{\mathsf{SDVS}})$
$(pk_{\mathsf{DEN}}, sk_{\mathsf{DEN}}) \leftarrow \mathsf{DEN.KGen}(pp_{\mathsf{DEN}})$
$pk_V \leftarrow (pk_{\mathsf{DEN}}, pk_{\mathsf{DV}})$
$s \leftarrow\!\!\$\ \{0,1\}^\lambda$
$c \leftarrow \mathsf{DEN.Enc}(pp_{\mathsf{DEN}}, pk_{\mathsf{DEN}}, s)$
$pk_S^V \leftarrow (pk_{\mathsf{SDVS}}, c)$
$(m^*, (\sigma_1^*, \sigma_2^*)) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{SIG}}, \mathcal{O}_{\mathsf{SIM}}, \mathcal{O}_{\mathsf{VER}}}(pp, pk_S^V, pk_V)$
return $(\mathcal{H}_2(m^*||\sigma_1^*), \sigma_2^*)$

$\mathcal{O}_{\mathsf{SIM}}(m)$

$\sigma_1 \leftarrow \mathsf{DEN.Enc}(pp_{\mathsf{DEN}}, pk_{\mathsf{DEN}}, \mathcal{H}_1(m||s))$
$\sigma_2 \leftarrow \mathcal{O}_{\mathsf{SIM}}^{\mathsf{SDVS}}(\mathcal{H}_2(m||\sigma_1))$
$\mathsf{Query} \leftarrow \mathsf{Query} \cup \{m\}$
return (σ_1, σ_2)

Fig. 27: $\mathcal{A}'$ which breaks the unforgeability of SDVS.

Inputs to $\mathcal{A}$ are distributed identically to the unforgeability game because (pp, pk_S^V, pk_V) are generated honestly. The oracle $\mathcal{O}_{\mathsf{SIG}}$ is distributed identically as $\mathcal{O}_{\mathsf{SIG}}^{\mathsf{SDVS}}(\mathcal{H}_2(m||\sigma_1))$ returns $\mathsf{SDVS.Sign}(pp_{\mathsf{SDVS}}, sk_{\mathsf{SDVS}}, pk_{\mathsf{DV}}, \mathcal{H}_2(m||\sigma_1))$. The oracle $\mathcal{O}_{\mathsf{SIM}}$ is distributed identically because $\mathcal{O}_{\mathsf{SIM}}^{\mathsf{SDVS}}(\mathcal{H}_2(m||\sigma_1))$ returns $\mathsf{SIM}_{\mathsf{SDVS}}(pp_{\mathsf{SDVS}}, sk_{\mathsf{DV}}, pk_{\mathsf{DV}}, pk_{\mathsf{SDVS}}, \mathcal{H}_2(m||\sigma_1))$. The oracle $\mathcal{O}_{\mathsf{VER}}$ is distributed identically because $\mathcal{O}_{\mathsf{VER}}^{\mathsf{SDVS}}(\mathcal{H}_2(m||\sigma_1), \sigma_2)$ returns $\mathsf{SDVS.Verify}(pp_{\mathsf{SDVS}}, pk_{\mathsf{SDVS}}, sk_{\mathsf{DV}}, \mathcal{H}_2(m||\sigma_1), \sigma_2)$.

We now show that $\mathcal{A}'$ is successful in the unforgeability game, i.e. that σ_2^* is a valid signature on $\mathcal{H}_2(m^*||\sigma_1^*)$, and that $\mathcal{H}_2(m^*||\sigma_1^*)$ has not been input to $\mathcal{O}_{\mathsf{SIG}}^{\mathsf{SDVS}}$ or $\mathcal{O}_{\mathsf{SIM}}^{\mathsf{SDVS}}$. As (σ_1^*, σ_2^*) is a valid CR.SDVS signature, $\mathsf{SDVS.Verify}(pp_{\mathsf{SDVS}}, pk_{\mathsf{SDVS}}, sk_{\mathsf{DV}}, \mathcal{H}_2(m^*||\sigma_1^*), \sigma_2^*) = 1$ and therefore σ_2^* is a valid SDVS signature. By assumption, m^* is not input to $\mathcal{O}_{\mathsf{SIG}}$ or $\mathcal{O}_{\mathsf{SIM}}$ and so $\mathcal{H}_2(m^*||\sigma_1^*)$ is not input to $\mathcal{O}_{\mathsf{SIG}}^{\mathsf{SDVS}}$ or $\mathcal{O}_{\mathsf{SIM}}^{\mathsf{SDVS}}$, unless a hash collision occurs, which occurs with negligible probability. Therefore, $\mathcal{A}'$ succeeds in the unforgeability experiment and, by contradiction, the result holds.

Lemma 15 (Privacy of the Signer's Identity). CR.SDVS *satisfies* privacy of the signer's identity *if strong designated verifier signature scheme* SDVS *satisfies privacy of the signer's identity, strong unforgeability and source hiding and deniable encryption scheme* DEN *satisfies* IND-CPA *security.*

Proof. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary in the $\mathsf{Exp}^{\mathsf{PSI},b}_{\mathcal{A},\mathsf{CR.SDVS}}(\lambda)$ experiment that makes at most k_1, k_2 queries to the SIGN oracle and VER oracle respectively. We proceed through a series of game hops that we show are indistinguishable to the adversary. In the final game, we show that if the advantage of the adversary is non–negligible we can build an adversary that breaks the privacy of the signer's identity for SDVS. We define Game 0 as the experiment $\mathsf{Exp}^{\mathsf{PSI},b}_{\mathcal{A},\mathsf{CR.SDVS}}(\lambda)$ with b chosen randomly. Let S_i be the event that adversary $\mathcal{A}$ correctly guesses b after Game i.

Game 1 is identical to Game 0, except for a change to the VER oracle. If $(m, (\sigma_1, \sigma_2))$ is input such that σ_2 is a valid SDVS signature on $\mathcal{H}_2(m||\sigma_1)$ and (σ_1, σ_2) was not output by the SIGN or SIM oracles on input m, d then the game aborts. We give Game 1 in Figure 28.

Game 1

$pp \leftarrow \mathsf{CR.SDVS.Setup}(1^\lambda)$
$(pk_V, sk_V) \leftarrow \mathsf{CR.SDVS.DVKGen}(pp)$
$(pk^V_{S_0}, sk^V_{S_0}, \mathsf{trans}_0) \leftarrow \mathsf{CR.SDVS.SKGen}(pp, pk_V)$
$(pk^V_{S_1}, sk^V_{S_1}, \mathsf{trans}_1) \leftarrow \mathsf{CR.SDVS.SKGen}(pp, pk_V)$
$(m^*, st) \leftarrow \mathcal{A}_1^{\mathsf{SIGN,VER,SIM}}(pp, pk^V_{S_0}, pk^V_{S_1}, pk_V)$
$(\sigma^*, \mathsf{trans}_b) \leftarrow \mathsf{CR.SDVS.Sign}(pp, sk^V_{S_b}, pk_V, m^*, \mathsf{trans}_b)$
$b' \leftarrow \mathcal{A}_2^{\mathsf{SIGN,VER,SIM}}(\sigma^*, st)$
return b'

SIGN/SIM

as normal

$\mathsf{VER}_{(pp, pk^V_{S_0}, pk^V_{S_1}, sk_V, m^*, \sigma^*)}(m, \sigma, d)$

parse pp as $(pp_{\mathsf{DEN}}, pp_{\mathsf{SDVS}})$ $\wedge$ sk_V as $(sk_{\mathsf{DEN}}, sk_{\mathsf{DV}})$ $\wedge$ σ as (σ_1, σ_2)
parse $pk^V_{S_0}$ as $(pk_{\mathsf{SDVS},0}, c_0))$ $\wedge$ **parse** $pk^V_{S_1}$ as $(pk_{\mathsf{SDVS},1}, c_1))$
if $(m, \sigma) = (m^*, \sigma^*)$ **return** $\perp$
if $\mathsf{SDVS.Verify}(pp_{\mathsf{SDVS}}, pk_{\mathsf{SDVS},d}, sk_{\mathsf{DV}}, \mathcal{H}_2(m||\sigma_1), \sigma_2) = 0$ **return** 0
if σ output by SIGN/SIM on input m, d **return** $\mathsf{CR.SDVS.Verify}(pp, pk^V_{S_d}, sk_V, m, \sigma)$
else Abort Game 1

Fig. 28: Game 1.

We now show that the probability of Game 1 aborting is negligible assuming the strong designated verifier signature scheme SDVS satisfies strong unforgeability. We build an adversary $\mathcal{A}_1'$ that breaks the strong unforgeability of the strong designated verifier signature scheme SDVS, given an adversary $\mathcal{A}$ that causes Game 1 to abort. We give $\mathcal{A}_1'$ in Figure 29.

We first assume that $\mathcal{A}$ causes Game 1 to abort on the q^*th query to VER. This occurs with probability $1/k_2$. We show that assuming q^* was guessed correctly, all inputs to $\mathcal{A}$ are indistinguishable from Game 1. pk_{SDVS}, pk_{DV} are the public keys for the SDVS scheme as in CR.SDVS.DVKGen, CR.SDVS.SKGen and otherwise the public keys $pk_V, pk^V_{S_0}, pk^V_{S_1}$ are generated

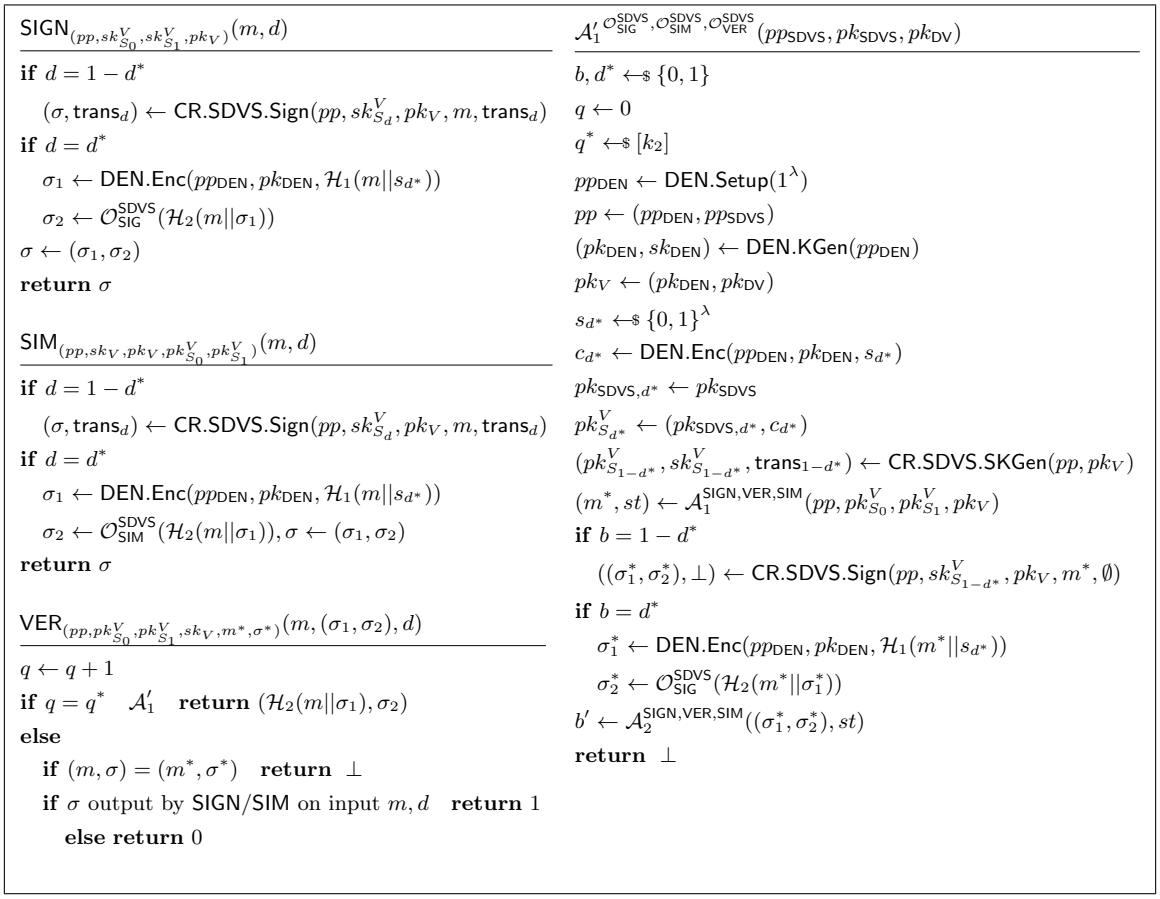

Fig. 29: $\mathcal{A}'_1$ which breaks the strong unforgeability of SDVS.

identically. The challenge signatures are computed identically, except if $b = d^*$ the SDVS signing oracle is used to generate σ^*_2.

Simulating the SIGN/SIM *oracle.* The SIGN oracle is identical to in Game 1, except when $d = d^*$ is input. In this case σ_1 is computed identically and σ_2 can be generated using the SDVS signing oracle. When the SIM oracle is input $d = d^*$, σ_1 is generated identically and the SDVS simulation oracle can be used to generate σ_2. When the SIM oracle in input $1 - d^*$, CR.SDVS.Sign is used to generate the signature. Due to the source hiding of SDVS, this is indistinguishable from the output of CR.SDVS.Sim.

Simulating the VER *oracle.* On the q^*th query $\mathcal{A}'_1$ aborts. Otherwise, if σ was output by SIGN on m, d then clearly σ is valid, due to correctness. Therefore, 1 is the correct output. As it is not yet the q^*th query, if σ was not output by SIGN or SIM then $\mathsf{SDVS.Verify}(pp_{\mathsf{SDVS}}, pk_{\mathsf{SDVS},d}, sk_{\mathsf{DV}}, \mathcal{H}_2(m||\sigma_1), \sigma_2) = 0$ and so 0 is the correct output.

Reduction to the strong unforgeability of SDVS. We first assume that d input to the q^*th verification query is d^*, which occurs with probability $1/2$. For $\mathcal{A}'_1$ to win in the strong unforgeability game, for (σ_1, σ_2) input during the q^*th query, $(\mathcal{H}_2(m||\sigma), \sigma)$ must be a valid SDVS signature under pk_{SDVS} and σ_2 must not have been output by the signing oracle on input $\mathcal{H}_2(m||\sigma_1)$. As Game 1 aborts during this query, then $(\mathcal{H}_2(m||\sigma), \sigma)$ is valid. As σ was not output by SIGN on input m, d^* and the challenge signature cannot be input to VER, σ_2 was not output by the SDVS signing oracle. Therefore, with probability $1/2k_2$, if $\mathcal{A}$ aborts during Game 1, then $\mathcal{A}'_1$ will be successful. Therefore, if ϵ_{SUF} is the advantage in breaking the strong unforgeability of SDVS and A is the event that Game 1 aborts, $\epsilon_{\mathsf{SUF}} \geq A/2k_2$. As Game 1 proceeds identically to Game 0, unless A occurs, $|Pr[S_0] - Pr[S_1]| \leq 2k_2\epsilon_{\mathsf{SUF}}$.

Game 2 is identical to Game 1 except, during the generation of the challenge signature, a freshly sampled value s^* is encrypted instead of s_b. We give Game 2 in Figure 30.

SIGN/VER/SIM

as in Game 1

Game 2

$b \leftarrow\$ \{0,1\}$
$pp \leftarrow \mathsf{CR.SDVS.Setup}(1^\lambda)$
$(pk_V, sk_V) \leftarrow \mathsf{CR.SDVS.DVKGen}(pp)$
$(pk^V_{S_0}, sk^V_{S_0}, \mathsf{trans}_0) \leftarrow \mathsf{CR.SDVS.SKGen}(pp, pk_V)$
$(pk^V_{S_1}, sk^V_{S_1}, \mathsf{trans}_1) \leftarrow \mathsf{CR.SDVS.SKGen}(pp, pk_V)$
parse pp as $(pp_{\mathsf{DEN}}, pp_{\mathsf{SDVS}})$ $\wedge$ pk_V as $(pk_{\mathsf{DEN}}, pk_{\mathsf{DV}})$
parse $sk^V_{S_0}$ as $(sk_{\mathsf{SDVS},0}, s_0))$ $\wedge$ $sk^V_{S_1}$ as $(sk_{\mathsf{SDVS},1}, s_1))$
$(m^*, st) \leftarrow \mathcal{A}_1^{\mathsf{SIGN},\mathsf{VER},\mathsf{SIM}}(pp, pk^V_{S_0}, pk^V_{S_1}, pk_V)$
$s^* \leftarrow\$ \{0,1\}^\lambda$
$\sigma_1^* \leftarrow \mathsf{DEN.Enc}(pp_{\mathsf{DEN}}, pk_{\mathsf{DEN}}, \mathcal{H}_1(m^*||s^*))$
$\sigma_2^* \leftarrow \mathsf{SDVS.Sign}(pp_{\mathsf{SDVS}}, sk_{\mathsf{SDVS},b}, pk_{\mathsf{DV}}, \mathcal{H}_2(m^*||\sigma_1^*))$
$\sigma^* \leftarrow (\sigma_1^*, \sigma_2^*)$
$b' \leftarrow \mathcal{A}_2^{\mathsf{SIGN},\mathsf{VER},\mathsf{SIM}}(\sigma^*, st)$
return b'

Fig. 30: Game 2.

We show that Game 2 and Game 1 are indistinguishable assuming the encryption scheme DEN is IND-CPA secure. We give a distinguishing algorithm $\mathcal{D}_2$ in Figure 31. $\mathcal{D}_2$ aims to guess b' in the IND-CPA game.

SIGN

as normal

$\mathsf{SIM}_{(pp, sk_V, pk_V, pk^V_{S_0}, pk^V_{S_1})}(m, d)$

$\sigma_1 \leftarrow \mathsf{DEN.Enc}(pp_{\mathsf{DEN}}, pk_{\mathsf{DEN}}, \mathcal{H}_1(m||s_d))$
$\sigma_2 \leftarrow \mathsf{Sim}_{\mathsf{SDVS}}(pp_{\mathsf{SDVS}}, sk_{\mathsf{DV}}, pk_{\mathsf{DV}}, pk_{\mathsf{SDVS},d}, \mathcal{H}_2(m||\sigma_1))$
return (σ_1, σ_2)

$\mathsf{VER}_{(pp, pk^V_{S_0}, pk^V_{S_1}, sk_V, m^*, \sigma^*)}(m, \sigma, d)$

parse σ as (σ_1, σ_2)
if $(m, \sigma) = (m^*, \sigma^*)$ **return** $\perp$
if $\mathsf{SDVS.Verify}(pp_{\mathsf{SDVS}}, pk_{\mathsf{SDVS},d}, sk_{\mathsf{DV}}, \mathcal{H}_2(m||\sigma_1), \sigma_2) = 0$ **return** 0
if σ output by SIGN/SIM on input m, d
  return 1
else Abort $\mathcal{D}_2$

$\mathcal{D}_2(\sigma_1^*, st')$

$\sigma_2^* \leftarrow \mathsf{SDVS.Sign}(pp_{\mathsf{SDVS}}, sk_{\mathsf{SDVS},b}, pk_{\mathsf{DV}}, \mathcal{H}_2(m^*||\sigma_1^*))$
$b' \leftarrow \mathcal{A}_2^{\mathsf{SIGN},\mathsf{VER},\mathsf{SIM}}((\sigma_1^*, \sigma_2^*), st)$
if $b' = b$ **return** 1

$\mathcal{D}_2(pp_{\mathsf{DEN}}, pk_{\mathsf{DEN}})$

$b \leftarrow\$ \{0,1\}$
$pp_{\mathsf{SDVS}} \leftarrow \mathsf{SDVS.Setup}(1^\lambda)$
$pp \leftarrow (pp_{\mathsf{DEN}}, pp_{\mathsf{SDVS}})$
$(pk_{\mathsf{DV}}, sk_{\mathsf{DV}}) \leftarrow \mathsf{SDVS.DVKGen}(pp_{\mathsf{SDVS}})$
$pk_V \leftarrow (pk_{\mathsf{DEN}}, pk_{\mathsf{DV}})$
$(pk^V_{S_0}, sk^V_{S_0}, \mathsf{trans}_0) \leftarrow \mathsf{CR.SDVS.SKGen}(pp, pk_V)$
$(pk^V_{S_1}, sk^V_{S_1}, \mathsf{trans}_1) \leftarrow \mathsf{CR.SDVS.SKGen}(pp, pk_V)$
parse $sk^V_{S_d}$ as $(sk_{\mathsf{SDVS},d}, s_d)$
parse $pk^V_{S_d}$ as $(pk_{\mathsf{SDVS},d}, c_d))$ for $d \in \{0,1\}$
$(m^*, st) \leftarrow \mathcal{A}_1^{\mathsf{SIGN},\mathsf{VER},\mathsf{SIM}}(pp, pk^V_{S_0}, pk^V_{S_1}, pk_V)$
$s^* \leftarrow\$ \{0,1\}^\lambda$
return $(\mathcal{H}_1(m^*||s_b), \mathcal{H}_1(m^*||s^*), st')$

Fig. 31: $\mathcal{D}_2$ that distinguishes between Game 2 and Game 1.

We now show that, when $b' = 0$, inputs to $\mathcal{A}$ are identical to Game 1, and, when $b' = 1$, inputs to $\mathcal{A}$ are identical to Game 2. The $pp, pk_V, pk^V_{S_0}, pk^V_{S_1}$ input to $\mathcal{A}$ are identical to both Game 1 and Game 2 as they are honestly generated. Although sk_{DEN} is not known this is never used. If $b' = 0$, $\mathcal{D}_2$ is returned with the encryption of $\mathcal{H}_1(m^*||s_b)$, as in Game 1. If $b' = 1$, $\mathcal{D}_1$ is returned with the encryption of $\mathcal{H}_1(m^*||s^*)$, as in Game 2. Therefore, the challenge signature is distributed correctly. The SIGN oracle is identical to in both Game 1 and Game 2. The SIM oracle is identical to both Game 1 and Game 2, except that instead of decrypting c_d, s_d is used. The VER oracle is identically distributed, because clearly signatures returned from SIGN, SIM are valid. Therefore $|Pr[S_1] - Pr[S_2]| \leq \epsilon_{\mathsf{IND\text{-}CPA}}$, where $\epsilon_{\mathsf{IND\text{-}CPA}}$ is the advantage in breaking the $\mathsf{IND\text{-}CPA}$ security of the deniable encryption scheme.

We now show that given an adversary $\mathcal{A}$ that wins in Game 2, we can build an adversary $\mathcal{A}'_3$ that breaks the privacy of the signer's identity of the SDVS scheme. We give the detailed description of $\mathcal{A}'_3$ in Figure 32, and explain here how $\mathcal{A}'_3$ works.

$\mathsf{SIGN}_{(pp, sk^V_{S_0}, sk^V_{S_1}, pk_V)}(m, d)$

$\sigma_1 \leftarrow \mathsf{DEN.Enc}(pp_{\mathsf{DEN}}, pk_{\mathsf{DEN}}, \mathcal{H}_1(m||s_d))$

$\sigma_2 \leftarrow \mathcal{O}^{\mathsf{SDVS}}_{\mathsf{SIG}}(\mathcal{H}_2(m||\sigma_1), d)$

return (σ_1, σ_2)

$\mathsf{SIM}_{(pp, sk_V, pk_V, pk^V_{S_0}, pk^V_{S_1})}(m, d)$

$\sigma_1 \leftarrow \mathsf{DEN.Enc}(pp_{\mathsf{DEN}}, pk_{\mathsf{DEN}}, \mathcal{H}_1(m||\mathsf{DEN.Dec}(pp_{\mathsf{DEN}}, sk_{\mathsf{DEN}}, c_d)))$

$\sigma_2 \leftarrow \mathcal{O}^{\mathsf{SDVS}}_{\mathsf{SIM}}(\mathcal{H}_2(m||\sigma_1), d)$

return (σ_1, σ_2)

$\mathsf{VER}_{(pp, pk^V_{S_0}, pk^V_{S_1}, sk_V, m^*, \sigma^*)}(m, \sigma, d)$

parse σ as (σ_1, σ_2)

if $(m, \sigma) = (m^*, \sigma^*)$ **return** $\perp$

if $\mathcal{O}^{\mathsf{SDVS}}_{\mathsf{VER}}(\mathcal{H}_2(m||\sigma_1), \sigma_2, d)$ **return** 0

if σ output by $\mathsf{SIGN}/\mathsf{SIM}$ on input m, d **return** 1

else Abort $\mathcal{A}'_3$

$\mathcal{A}'^{\mathcal{O}^{\mathsf{SDVS}}_{\mathsf{SIG}}, \mathcal{O}^{\mathsf{SDVS}}_{\mathsf{VER}}, \mathcal{O}^{\mathsf{SDVS}}_{\mathsf{SIM}}}_3(pp_{\mathsf{SDVS}}, pk_{\mathsf{SDVS},0}, pk_{\mathsf{SDVS},1}, pk_{\mathsf{DV}})$

$pp_{\mathsf{DEN}} \leftarrow \mathsf{DEN.Setup}(1^\lambda)$

$pp \leftarrow (pp_{\mathsf{DEN}}, pp_{\mathsf{SDVS}})$

$(pk_{\mathsf{DEN}}, sk_{\mathsf{DEN}}) \leftarrow \mathsf{DEN.KGen}(pp_{\mathsf{DEN}})$

$pk_V \leftarrow (pk_{\mathsf{DEN}}, pk_{\mathsf{DV}})$

for $d \in \{0, 1\}$

  $s_d \leftarrow\$ \{0, 1\}^\lambda$

  $c_d \leftarrow \mathsf{DEN.Enc}(pp_{\mathsf{DEN}}, pk_{\mathsf{DEN}}, s_d)$

  $pk^V_{S_d} \leftarrow (pk_{\mathsf{SDVS},d}, c_d)$

$(m^*, st) \leftarrow \mathcal{A}^{\mathsf{SIGN,VER,SIM}}_1(pp, pk^V_{S_0}, pk^V_{S_1}, pk_V)$

$s^* \leftarrow\$ \{0, 1\}^\lambda$

$\sigma^*_1 \leftarrow \mathsf{DEN.Enc}(pp_{\mathsf{DEN}}, pk_{\mathsf{DEN}}, \mathcal{H}_1(m^*||s^*))$

return $(\mathcal{H}_2(m^*||\sigma^*_1), st')$

$\mathcal{A}'^{\mathcal{O}^{\mathsf{SDVS}}_{\mathsf{SIG}}, \mathcal{O}^{\mathsf{SDVS}}_{\mathsf{VER}}, \mathcal{O}^{\mathsf{SDVS}}_{\mathsf{SIM}}}_3(\sigma^*_2, st')$

$b' \leftarrow \mathcal{A}^{\mathsf{SIGN,VER,SIM}}_2((\sigma^*_1, \sigma^*_2), st)$

return b'

Fig. 32: $\mathcal{A}'_3$ that breaks the privacy of the signer's identity of the SDVS scheme.

We show that all inputs to $\mathcal{A}$ are distributed identically to in Game 2, such that the bit b is the same as the bit b' chosen in the privacy of the signer's identity game for the SDVS scheme. $pp, pk^V_{S_0}, pk^V_{S_1}, pk_V$ are generated honestly. The challenge signature is identically distributed, because σ^*_1 is generated identically and $\sigma^*_2 = \mathsf{SDVS.Sign}(pp_{\mathsf{SDVS}}, sk_{\mathsf{SDVS},b'}, pk_{\mathsf{DV}}, \mathcal{H}_2(m^*||\sigma^*_1))$. The SIGN and SIM oracles are identical to Game 2, except that the SDVS signing and simulation oracles are used respectively to generate σ_2. The VER oracle is identical, except that the SDVS verification oracle is used to verify $(\mathcal{H}_2(m||\sigma_1), \sigma_2)$ and if (σ_1, σ_2) was output by $\mathsf{SIGN}, \mathsf{SIM}$ then 1 is returned.

If $\mathcal{A}$ guesses b correctly then $\mathcal{A}'_3$ correctly guesses b' in the SDVS game. Therefore, $\Pr[S_2] \leq \epsilon_{\mathsf{psi}} + 1/2$, where ϵ_{psi} is the advantage in breaking the privacy of the signer's identity of the SDVS scheme.

We now have that $|Pr[S_0] - 1/2| \leq 2k_2\epsilon_{\mathsf{SUF}} + \epsilon_{\mathsf{IND\text{-}CPA}} + \epsilon_{\mathsf{psi}}$ and conclude that the advantage of the adversary in Game 0 is negligible as required.

Coercion-Resistance Our $\mathsf{CR.SDVS}$ construction adapts a strong-designated verifier signature scheme to satisfy both ($\mathsf{IND3}$) indistinguishability and strong soundness in the same way that our publicly verifiable construction, $\mathsf{CR.SIG}$, adapts a standard signature scheme. The only difference is that now the authenticator's role is taken by the designated verifier and so, during verification, signatures

are also authenticated. This means that the security proofs of CR.SDVS for both requirements are almost identical to the proofs in the publicly verifiable setting, with only notational differences.

Lemma 16 ((IND3) Indistinguishability). CR.SDVS *satisfies* (IND3) *indistinguishability for a coercion-resistant incoercible strong designated verifier signature scheme if the deniable encryption scheme satisfies* IND-CPA *security and* IND-EXP *security.*

Proof. Let $\mathcal{A}$ be an adversary in the $\mathsf{Exp}^{\mathsf{IND3},b}_{\mathcal{A},\mathsf{CR.SDVS}}(\lambda)$ experiment that makes at most k_1 queries to the ADDU oracle, at most k_2 queries to the SIGN oracle per signer and at most k_3 queries to the STCRCSIG oracle per signer. We proceed through a series of game hops that we show are indistinguishable to the adversary. In the final game, the view of the adversary will be identical for $b = 0$ and $b = 1$ and the adversary will have a negligible chance of guessing b. We define Game 0 as the experiment $\mathsf{Exp}^{\mathsf{IND3},b}_{\mathcal{A},\mathsf{CR.SDVS}}(\lambda)$ with b chosen randomly. Let S_i be the event that adversary $\mathcal{A}$ correctly guesses b after Game i.

Game 1 is identical to Game 0, except that the fake transcript is only generated when a signer is first added to the coerced list. We give the adjusted oracles for this game in Figure 33. As the fake transcript is only used after a signer has been coerced this is only a superficial change and does not affect the distribution of inputs to the adversary. Therefore, $Pr[S_0] = Pr[S_1]$.

$\mathsf{ADDU}_{(pp,pk_V,\mathbf{pk}^{\mathbf{V}}_{\mathbf{S}},\mathbf{sk}^{\mathbf{V}}_{\mathbf{S}},\mathsf{L},\mathsf{trans},\mathsf{trans}_{\mathsf{fake}})}(id)$

if $id \in \mathsf{L}$ **return** $\perp$
$\mathsf{L} \leftarrow \mathsf{L} \cup \{id\}$
$(\mathbf{pk}^{\mathbf{V}}_{\mathbf{S}}[id], \mathbf{sk}^{\mathbf{V}}_{\mathbf{S}}[id], \mathsf{trans}[id]) \leftarrow \mathsf{CR.SDVS.SKGen}(pp, pk_V)$
return $\mathbf{pk}^{\mathbf{V}}_{\mathbf{S}}[id]$

$\mathsf{COERCE}_{(pp,pp_V,\mathsf{L},\mathsf{corL},\mathsf{crcL},\mathsf{trans},\mathsf{trans}_{\mathsf{fake}})}(id)$

if $id \notin \mathsf{L} \setminus \mathsf{corL}$ **return** $\perp$
if $id \notin \mathsf{crcL}$ $\mathsf{crcL} \leftarrow \mathsf{crcL} \cup \{id\}$
  $\mathsf{trans}_{\mathsf{fake}}[id] \leftarrow \mathsf{CR.SDVS.FakeTrans}(pp, pk_V, \mathsf{trans}[id], \perp)$
  $\mathsf{trans}_{\mathsf{fake}}[id] \leftarrow \mathsf{CR.SDVS.FakeTrans}(pp, pk_V, \mathsf{trans}[id], \mathsf{trans}_{\mathsf{fake}}[id])$
if $b = 0$ **return** $\mathsf{trans}[id]$
if $b = 1$ **return** $\mathsf{trans}_{\mathsf{fake}}[id]$

$\mathsf{SIGN}_{(pp,pk_V,\mathbf{sk}^{\mathbf{V}}_{\mathbf{S}},\mathsf{L},\mathsf{crcL},\mathsf{Query},\mathsf{trans},\mathsf{trans}_{\mathsf{fake}})}(id, m)$

if $id \notin \mathsf{L}$ **return** $\perp$
$(\sigma, \mathsf{trans}[id])) \leftarrow \mathsf{CR.SDVS.Sign}(pp, \mathbf{sk}^{\mathbf{V}}_{\mathbf{S}}[id], pk_V, m, \mathsf{trans}[id]))$
if $id \in \mathsf{crcL}$ $\mathsf{trans}_{\mathsf{fake}}[id] \leftarrow \mathsf{CR.SDVS.FakeTrans}(pp, pk_V, \mathsf{trans}[id], \mathsf{trans}_{\mathsf{fake}}[id])$
$\mathsf{Query} \leftarrow \mathsf{Query} \cup \{(id, m)\}$
return σ

$\mathsf{STCRCSIG}_{(pp,pk_V,\mathbf{sk}^{\mathbf{V}}_{\mathbf{S}},\mathsf{L},\mathsf{corL},\mathsf{crcL},\mathsf{trans},\mathsf{trans}_{\mathsf{fake}})}(id, m, r)$

if $id \notin \mathsf{L} \setminus \mathsf{corL}$ **return** $\perp$
if $id \notin \mathsf{crcL}$ $\mathsf{crcL} \leftarrow \mathsf{crcL} \cup \{id\}$
  $\mathsf{trans}_{\mathsf{fake}}[id] \leftarrow \mathsf{CR.SDVS.FakeTrans}(pp, pk_V, \mathsf{trans}[id], \perp)$
  $\mathsf{trans}_{\mathsf{fake}}[id] \leftarrow \mathsf{CR.SDVS.FakeTrans}(pp, pk_V, \mathsf{trans}[id], \mathsf{trans}_{\mathsf{fake}}[id])$
if $b = 0$ $(\sigma, \mathsf{trans}[id]) \leftarrow \mathsf{CR.SDVS.Sign}(pp, \mathbf{sk}^{\mathbf{V}}_{\mathbf{S}}[id], pk_V, m, \mathsf{trans}[id]; r)$
if $b = 1$
  $(\sigma, \mathsf{trans}_{\mathsf{fake}}[\mathbf{id}]) \leftarrow \mathsf{CR.SDVS.FakeSign}(pp, \mathbf{sk}^{\mathbf{V}}_{\mathbf{S}}[id], pk_V, m, \mathsf{trans}[id], \mathsf{trans}_{\mathsf{fake}}[id]; r)$
return σ

Fig. 33: Oracles used in Game 1.

Game 2 is identical to Game 1, except for a change to the COERCE and STCRCSIG oracle. Instead of using s' during the generation of the fake transcript, we will instead use the real key s. We give the COERCE and STCRCSIG oracles used in Game 2 in Figure 34.

This game hop will make use of a hybrid argument. We split the reduction into k_1 steps. We define Game 1.i to behave as in Game 2 only for the first i signers submitted to the ADDU oracle and otherwise behave as in Game 1. Clearly, Game 1.k_1 will be identical to Game 2 and Game 1.0 will be identical to Game 1.

$\mathsf{COERCE}_{(pp,pk_V,\mathsf{L},\mathsf{corL},\mathsf{crcL},\mathsf{trans},\mathsf{trans}_{\mathsf{fake}})}(id)$

parse pp as $(pp_{\mathsf{DEN}}, pp_{\mathsf{SDVS}}), pk_V$ as $(pk_{\mathsf{DEN}}, pk_{\mathsf{DV}})$ $\wedge$ $\mathsf{trans}[id]$ as $\{(r_{\mathsf{SDVS}}, pk_{\mathsf{SDVS}}, sk_{\mathsf{SDVS}}, s, r_c, c), \ldots, (m, r_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2), \ldots\}$
if $id \notin \mathsf{L} \setminus \mathsf{corL}$ **return** $\perp$
if $id \notin \mathsf{crcL}$ $\mathsf{crcL} \leftarrow \mathsf{crcL} \cup \{id\}$
  $r'_c \leftarrow \mathsf{DEN.Exp}(pp_{\mathsf{DEN}}, pk_{\mathsf{DEN}}, c, s)$
  $\mathsf{trans}_{\mathsf{fake}}[id] := \{(r_{\mathsf{SDVS}}, pk_{\mathsf{SDVS}}, sk_{\mathsf{SDVS}}, s, r'_c, c)\}$
  For each entry $(m, r_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2)$ from $\mathsf{CR.SDVS.Sign}$
    $r'_{\sigma_1} \leftarrow \mathsf{DEN.Exp}(pp_{\mathsf{DEN}}, pk_{\mathsf{DEN}}, \sigma_1, \mathcal{H}_1(m||s))$
    $\mathsf{trans}_{\mathsf{fake}}[id] \leftarrow \mathsf{trans}_{\mathsf{fake}}[id] \cup \{(m, r'_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2)\}$
if $b = 0$ **return** $\mathsf{trans}[id]$
if $b = 1$ **return** $\mathsf{trans}_{\mathsf{fake}}[id]$

$\mathsf{STCRCSIG}_{(pp,pk_V,\mathbf{sk}_{\mathbf{S}}^{\mathbf{V}},\mathsf{L},\mathsf{corL},\mathsf{crcL},\mathsf{trans},\mathsf{trans}_{\mathsf{fake}})}(id, m, r)$

parse pp as $(pp_{\mathsf{DEN}}, pp_{\mathsf{SDVS}}), pk_V$ as $(pk_{\mathsf{DEN}}, pk_{\mathsf{DV}})$ $\wedge$ $\mathsf{trans}[id]$ as $\{(r_{\mathsf{SDVS}}, pk_{\mathsf{SDVS}}, sk_{\mathsf{SDVS}}, s, r_c, c), \ldots, (m, r_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2), \ldots\}$
if $id \notin \mathsf{L} \setminus \mathsf{corL}$ **return** $\perp$
if $id \notin \mathsf{crcL}$ $\mathsf{crcL} \leftarrow \mathsf{crcL} \cup \{id\}$
  $r'_c \leftarrow \mathsf{DEN.Exp}(pp_{\mathsf{DEN}}, pk_{\mathsf{DEN}}, c, s)$
  $\mathsf{trans}_{\mathsf{fake}}[id] := \{(r_{\mathsf{SDVS}}, pk_{\mathsf{SDVS}}, sk_{\mathsf{SDVS}}, s, r'_c, c)\}$
  For each entry $(m, r_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2)$ from $\mathsf{CR.SDVS.Sign}$
    $r'_{\sigma_1} \leftarrow \mathsf{DEN.Exp}(pp_{\mathsf{DEN}}, pk_{\mathsf{DEN}}, \sigma_1, \mathcal{H}_1(m||s))$
    $\mathsf{trans}_{\mathsf{fake}}[id] \leftarrow \mathsf{trans}_{\mathsf{fake}}[id] \cup \{(m, r'_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2)\}$
if $b = 0$ $(\sigma, \mathsf{trans}[id]) \leftarrow \mathsf{CR.SDVS.Sign}(pp, \mathbf{sk}_{\mathbf{S}}^{\mathbf{V}}[id], pk_V, m, \mathsf{trans}[id]; r)$
if $b = 1$ $(\sigma, \mathsf{trans}_{\mathsf{fake}}[\mathbf{id}]) \leftarrow \mathsf{CR.SDVS.FakeSign}(pp, \mathbf{sk}_{\mathbf{S}}^{\mathbf{V}}[id], pk_V, m, \mathsf{trans}[id], \mathsf{trans}_{\mathsf{fake}}[id]; r)$
return σ

Fig. 34: COERCE, STCRCSIG oracles in Game 2.

We show that Game 1.i and Game 1.$(i+1)$ are indistinguishable assuming the encryption scheme DEN is IND-CPA secure. We give a distinguishing algorithm $\mathcal{D}_1$ in Figure 35. $\mathcal{D}_1$ aims to guess a bit b' in the IND-CPA game for multiple encryptions. In this game, instead of receiving one ciphertext, the adversary can make several queries of the form (m_0, m_1) and will receive $\mathsf{DEN.Enc}(pp_{\mathsf{DEN}}, pk_{\mathsf{DEN}}, m_b)$ for each query to an oracle $\mathsf{LR}_{\mathsf{CPA}}$.

We show that, when $b' = 0$, inputs to $\mathcal{A}$ are identical to Game 1.i and, when $b' = 1$, inputs to $\mathcal{A}$ are identical to Game 1.$(i+1)$. Note that pp, pk_V input to $\mathcal{A}$ are honestly generated and are identical in Games 1.i and 1.$(i+1)$. Furthermore, sk_{DEN} is not known but is never used. The ADDU and SIGN oracles are distributed identically to both Games 1.i and 1.$(i+1)$, because when $b = 1, d = 1$ and $id = \hat{id}$, $s^*_{b'}$ is essentially used as the secret key. It is encrypted in both the public key and in honest signatures. The CRPT oracle is distributed identically to both Games 1.i and 1.$(i+1)$, provided it doesn't abort. This is because the real transcript is generated identically provided $id \neq \hat{id}$ or $d = 0$. Games 1.i and 1.$(i+1)$ are identical with the exception of oracles COERCE and STCRCSIG when $id = \hat{id}$ where $\hat{id}$ is the $i+1$th signer created. When $b = 0$, the fake transcript is never used and the real transcript is generated normally. Therefore the outputs of the COERCE and STCRCSIG oracles are distributed identically to both Games 1.i and 1.$(i+1)$. When $d = 0$, the oracles both abort. When $d = 1$ and $b = 1$, if $b' = 0$ the oracle is distributed identically to Game 1 because s^*_0 is used as the secret key and s^*_1 is used in the fake transcript. When $d = 1$ and $b = 1$, if $b' = 1$ the oracle is distributed identically to Game 2 because s^*_1 is used as both the secret key and in the fake transcript. The probability that distinguisher $\mathcal{D}_1$ aborts is at most 1/2. Therefore $|Pr[S_{1.i}] - Pr[S_{1.(i+1)}]| \leq 2\epsilon_{\mathsf{IND\text{-}CPA-mult}}$, where $\epsilon_{\mathsf{IND\text{-}CPA-mult}}$ is the advantage in breaking the IND-CPA for multiple encryptions property of the deniable encryption scheme DEN. By a standard argument, $|Pr[S_{1.i}] - Pr[S_{1.(i+1)}]| \leq 2(1+k_2)\epsilon_{\mathsf{IND\text{-}CPA}}$, where $\epsilon_{\mathsf{IND\text{-}CPA}}$ is the advantage in breaking the IND-CPA property of DEN. Therefore $|Pr[S_1] - Pr[S_2]| \leq 2k_1(1+k_2)\epsilon_{\mathsf{IND\text{-}CPA}}$

Game 3 is identical to Game 2, except for another change to the SIGN, COERCE and STCRCSIG oracles. We include the real randomness used to encrypt in the fake transcript, instead of obtaining

Fig. 35: $\mathcal{D}_1$ that distinguishes between Game 1.i and Game 1.i+1.

fake randomness from algorithm DEN.Exp. Additionally, in STCRCSIG we use the honest signing oracle regardless of b and then update the fake transcript as in the real transcript. We give the SIGN, COERCE and STCRCSIG oracles used in Game 3 in Figure 36.

This game hop will again make use of a hybrid argument. We split the reduction into k_1 steps. We define Game 2.i to behave as in Game 3 only for the first i signers submitted to the ADDU oracle and otherwise behave as in Game 2. Clearly, Game 2.k_1 will be identical to Game 3 and Game 2.0 will be identical to Game 2.

We show that Game 2.i and Game 2.$(i+1)$ are indistinguishable assuming the deniable encryption scheme DEN is IND-EXP secure. We give a distinguishing algorithm $\mathcal{D}_2$ in Figure 37. $\mathcal{D}_2$ aims

$\mathsf{SIGN}_{(pp,pk_V,\mathbf{sk}_{\mathbf{S}}^{\mathbf{V}},\mathsf{L},\mathsf{crcL},\mathsf{Query},\mathsf{trans},\mathsf{trans}_{\mathsf{fake}})}(id,m)$

if $id \notin \mathsf{L}$ **return** $\perp$
$(\sigma, \mathsf{trans}[id]) \leftarrow \mathsf{CR.SDVS.Sign}(pp, \mathbf{sk}_{\mathbf{S}}^{\mathbf{V}}[id], pk_V, m, \mathsf{trans}[id])$
if $id \in \mathsf{crcL}$ $\mathsf{trans}_{\mathsf{fake}}[id] \leftarrow \mathsf{trans}[id]$
$\mathsf{Query} \leftarrow \mathsf{Query} \cup \{(id,m)\}$
return σ

$\mathsf{COERCE}_{(pp,pk_V,\mathsf{L},\mathsf{corL},\mathsf{crcL},\mathsf{trans},\mathsf{trans}_{\mathsf{fake}})}(id)$

if $id \notin \mathsf{L} \setminus \mathsf{corL}$ **return** $\perp$
if $id \notin \mathsf{crcL}$ $\mathsf{crcL} \leftarrow \mathsf{crcL} \cup \{id\}$
$\mathsf{trans}_{\mathsf{fake}}[id] \leftarrow \mathsf{trans}[id]$
if $b = 0$ **return** $\mathsf{trans}[id]$
if $b = 1$ **return** $\mathsf{trans}_{\mathsf{fake}}[id]$

$\mathsf{STCRCSIG}_{(pp,pk_V,\mathbf{sk}_{\mathbf{S}}^{\mathbf{V}},\mathsf{L},\mathsf{corL},\mathsf{crcL},\mathsf{trans},\mathsf{trans}_{\mathsf{fake}})}(id,m,r)$

parse $\mathsf{trans}[id]$ as $\{(r_{\mathsf{SDVS}}, pk_{\mathsf{SDVS}}, sk_{\mathsf{SDVS}}, s, r_c, c), \ldots, (m, r_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2), \ldots\}$
if $id \notin \mathsf{L} \setminus \mathsf{corL}$ **return** $\perp$
$\mathsf{crcL} \leftarrow \mathsf{crcL} \cup \{id\}$
$(\sigma, \mathsf{trans}[id]) \leftarrow \mathsf{CR.SDVS.Sign}(pp, \mathbf{sk}_{\mathbf{S}}^{\mathbf{V}}[id], pk_V, m, \mathsf{trans}[id]; r)$
$\mathsf{trans}_{\mathsf{fake}}[id] \leftarrow \mathsf{trans}[id]$
return σ

Fig. 36: SIGN, COERCE and STCRCSIG oracles in Game 3.

to guess a bit b' in the IND-EXP game, which is adjusted similarly to the IND-CPA experiment for multiple encryptions. That is, the adversary can submit multiple queries to an oracle $\mathsf{LR}_{\mathsf{EXP}}$ that, on input a message, returns a ciphertext and randomness. If $b' = 0$, oracle $\mathsf{LR}_{\mathsf{EXP}}$ returns the randomness used to encrypt and, if $b' = 1$, it returns randomness obtained via the algorithm DEN.Exp.

We show that, when $b' = 0$, inputs to $\mathcal{A}$ are identical to Game $2.(i+1)$ and, when $b' = 1$, inputs to $\mathcal{A}$ are identical to Game $2.i$. Note that pp, pk_V input to $\mathcal{A}$ are honestly generated and are identical in Games $2.i$ and $2.(i+1)$. Furthermore, sk_V is not known but is never used. The outputs of the ADDU and SIGN oracles are distributed identically to both Games $2.i$ and $2.(i+1)$, because when $b = 1, d = 1$ and $id = \hat{id}$, the secret key s is always encrypted in both the public key and in signatures. The CRPT oracle is distributed identically to both Games $2.i$ and $2.(i+1)$, provided it doesn't abort. This is because the real transcript is generated identically provided $id \neq \hat{id}$ or $d = 0$.

Games $2.i$ and $2.(i+1)$ are identical with the exception of oracles SIGN, COERCE and STCRCSIG when $id = \hat{id}$. When $b = 0$, the fake transcript is never used and the real transcript is generated normally. Therefore the outputs of the COERCE and STCRCSIG oracles are distributed identically to both Games $2.i$ and $2.(i+1)$. When $d = 0$, the COERCE and STCRCSIG oracles both abort. When $d = 1$ and $b = 1$, if $b' = 0$ the fake transcript is distributed identically to Game 3 because $\mathcal{D}_2$ is input the real randomness used in the encryption. When $d = 1$ and $b = 1$, if $b' = 1$ the fake transcript is distributed identically to Game 2 because $\mathcal{D}_2$ is input the randomness output by DEN.Exp. In STCRCSIG, as the real key is included in the fake transcript in both Games 2 and Games 3, the signature output is distributed identically to both Game 2 and Game 3. The probability that distinguisher $\mathcal{D}_2$ aborts is at most $1/2$. Therefore $|Pr[S_{2.i}] - Pr[S_{2.(i+1)}]| \leq 2\epsilon_{\mathsf{IND\text{-}EXP\text{-}mult}}$, where $\epsilon_{\mathsf{IND\text{-}EXP\text{-}mult}}$ is the advantage in breaking the IND-EXP for multiple messages property of deniable encryption scheme DEN. By a standard argument, $|Pr[S_{2.i}] - Pr[S_{2.(i+1)}]| \leq 2(1 + k_2 + k_3)\epsilon_{\mathsf{IND\text{-}EXP}}$ where $\epsilon_{\mathsf{IND\text{-}EXP}}$ is the advantage in breaking the IND-EXP property of DEN. Therefore $|Pr[S_2] - Pr[S_3]| \leq 2k_1(1 + k_2 + k_3)\epsilon_{\mathsf{IND\text{-}EXP}}$.

In Game 3, the fake transcript is identical to the real transcript. Also signatures output by STCRCSIG are now always generated by the honest signing algorithm. Therefore, inputs to the adversary $\mathcal{A}$ are independent of b and so $Pr[S_3] = 1/2$.

We now have that

$$|Pr[S_0] - 1/2| \leq 2k_1(1 + k_2)(\epsilon_{\mathsf{IND\text{-}EXP}} + \epsilon_{\mathsf{IND\text{-}CPA}}) + 2k_1k_3\epsilon_{\mathsf{IND\text{-}EXP}}$$

and conclude that the advantage of the adversary in Game 0 is negligible as required.

$\mathsf{ADDU}_{(pp,pk_V,\mathbf{pk_S^V},\mathbf{sk_S^V},\mathsf{L},\mathsf{trans},\mathsf{trans_{fake}})}(id)$

parse pp as $(pp_{\mathsf{DEN}}, pp_{\mathsf{SDVS}})$
if $id \in \mathsf{L}$ **return** $\perp, \mathsf{L} \leftarrow \mathsf{L} \cup \{id\}, q \leftarrow q+1$
if $q = i+1$ $\hat{id} \leftarrow id$
if $b = 1$ and $d = 1$ and $id = \hat{id}$
  $(pk_{\mathsf{SDVS}}, sk_{\mathsf{SDVS}}) \leftarrow \mathsf{SDVS.SKGen}(pp_{\mathsf{SDVS}}; r_{\mathsf{SDVS}})$
  $s \leftarrow\$ \{0,1\}^{\lambda}, (c, r_c) \leftarrow \mathsf{LR}_{\mathsf{EXP}}(s)$
  $\mathsf{trans}[id] \coloneqq \{(r_{\mathsf{SDVS}}, pk_{\mathsf{SDVS}}, sk_{\mathsf{SDVS}}, s, r_c, c)$
  $(\mathbf{pk_S^V}[id], \mathbf{sk_S^V}[id]) \leftarrow ((pk_{\mathsf{SDVS}}, c), (sk_{\mathsf{SDVS}}, s))$
  return $\mathbf{pk_S^V}[id]$
else as in Game 2

$\mathsf{SIGN}_{(pp,pk_V,\mathbf{sk_S^V},\mathsf{L},\mathsf{crcL},\mathsf{Query},\mathsf{trans},\mathsf{trans_{fake}})}(id, m)$

parse pp as $(pp_{\mathsf{DEN}}, pp_{\mathsf{SDVS}}), \mathbf{sk_S^V}[id]$ as (sk_{SDVS}, s)
pk_V as $(pk_{\mathsf{DEN}}, pk_{\mathsf{DV}})$
if $b = 1$ and $d = 1$ and $id = \hat{id}$
  $(\sigma_1, r_{\sigma_1}) \leftarrow \mathsf{LR}_{\mathsf{EXP}}(\mathcal{H}_1(m||s))$
  $\sigma_2 \leftarrow \mathsf{SDVS.Sign}(pp_{\mathsf{SDVS}}, sk_{\mathsf{SDVS}}, pk_{\mathsf{DV}}, \mathcal{H}_2(m||\sigma_1); r_{\sigma_2})$
  $\sigma \leftarrow (\sigma_1, \sigma_2)$
  $\mathsf{trans}[id] \leftarrow \mathsf{trans}[id] \cup \{(m, r_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2)\}$
  if $id \in \mathsf{crcL}$
    $\mathsf{trans_{fake}}[id] \leftarrow \mathsf{trans_{fake}}[id] \cup \{(m, r_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2)\}$
  $\mathsf{Query} \leftarrow \mathsf{Query} \cup \{(id, m)\}$ **return** σ
else as in Game 2

$\mathsf{CRPT}_{(\mathbf{sk_S^V},\mathsf{L},\mathsf{corL},\mathsf{crcL},\mathsf{trans})}(id)$

if $id \notin \mathsf{L} \setminus \mathsf{crcL}$ **return** $\perp$
$\mathsf{corL} \leftarrow \mathsf{corL} \cup \{id\}$
if $d = 1$ and $id = \hat{id}$ $\mathcal{D}_1$ aborts
else return $\mathbf{sk_S^V}[id], \mathsf{trans}[id]$

$\mathsf{COERCE}_{(pp,pk_V,\mathsf{L},\mathsf{corL},\mathsf{crcL},\mathsf{trans},\mathsf{trans_{fake}})}(id)$

if $id \notin \mathsf{L} \setminus \mathsf{corL}$ **return** $\perp$
if $id = \hat{id}$
  if $b = 0$ $\mathsf{crcL} \leftarrow \mathsf{crcL} \cup \{id\}$, **return** $\mathsf{trans}[id]$
  if $d = 0$ $\mathcal{D}_1$ aborts
  parse $\mathsf{trans}[id]$ as $\{(r_{\mathsf{SDVS}}, pk_{\mathsf{SDVS}}, sk_{\mathsf{SDVS}}, s, r_c, c), \ldots, (m, r_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2), \ldots\}$
  if $id \notin \mathsf{crcL}$ $\mathsf{crcL} \leftarrow \mathsf{crcL} \cup \{id\}$
    $\mathsf{trans_{fake}}[id] \coloneqq \{(r_{\mathsf{SDVS}}, pk_{\mathsf{SDVS}}, sk_{\mathsf{SDVS}}, s, r_c, c), \ldots, (m, r_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2), \ldots\}$
  return $\mathsf{trans_{fake}}[id]$
else As in Game 1.i

$\mathsf{STCRCSIG}_{(pp,pk_V,\mathbf{sk_S^V},\mathsf{L},\mathsf{corL},\mathsf{crcL},\mathsf{trans},\mathsf{trans_{fake}})}(id, m, r)$

if $id \notin \mathsf{L} \setminus \mathsf{corL}$ **return** $\perp$
if $id = \hat{id}$
  if $b = 0$ $\mathsf{crcL} \leftarrow \mathsf{crcL} \cup \{id\}, (\sigma, \mathsf{trans}[id]) \leftarrow \mathsf{CR.SDVS.Sign}(pp, \mathbf{sk_S^V}[id], pk_V, m, \mathsf{trans}[id]; r)$
    return σ
  if $d = 0$ $\mathcal{D}_1$ aborts
  parse pp as $(pp_{\mathsf{DEN}}, pp_{\mathsf{SDVS}}), pk_V$ as $(pk_{\mathsf{DEN}}, pk_{\mathsf{DV}}), r$ as $(r_{\sigma_1}, r_{\sigma_2})$
  $\mathsf{trans}[id]$ as $\{(r_{\mathsf{SDVS}}, pk_{\mathsf{SDVS}}, sk_{\mathsf{SDVS}}, s, r_c, c), \ldots, (m, r_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2), \ldots\}$
  if $id \notin \mathsf{crcL}$ $\mathsf{crcL} \leftarrow \mathsf{crcL} \cup \{id\}$
    $\mathsf{trans_{fake}}[id] \coloneqq \{(r_{\mathsf{SDVS}}, pk_{\mathsf{SDVS}}, sk_{\mathsf{SDVS}}, s, r_c, c), \ldots, (m, r_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2), \ldots\}$
  $\sigma_1 \leftarrow \mathsf{DEN.Enc}(pp_{\mathsf{DEN}}, pk_V, \mathcal{H}_1(m||s); r_{\sigma_1})$,
  $\sigma_2 \leftarrow \mathsf{SDVS.Sign}(pp_{\mathsf{SDVS}}, sk_{\mathsf{SDVS}}, pk_{\mathsf{DV}}, \mathcal{H}_2(m||\sigma_1); r_{\sigma_2}), \sigma \leftarrow (\sigma_1, \sigma_2)$
  $\mathsf{trans_{fake}}[id] \leftarrow \mathsf{trans_{fake}}[id] \cup \{(m, r_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2; r)\}$
  return σ
else As in Game 1.i

$\mathcal{D}_2^{\mathsf{LR}_{\mathsf{EXP}}}(pp_{\mathsf{DEN}}, pk_{\mathsf{DEN}})$

$b \leftarrow\$ \{0,1\}, q \leftarrow 0, d \leftarrow\$ \{0,1\}, \mathbf{pk_S^V}, \mathbf{sk_S^V}, \mathsf{trans}, \mathsf{trans_{fake}} \leftarrow (), \mathsf{L}, \mathsf{crcL}, \mathsf{corL}, \mathsf{Query} \leftarrow \emptyset$
$pp_{\mathsf{SDVS}} \leftarrow \mathsf{SDVS.Setup}(1^{\lambda}), pp \leftarrow (pp_{\mathsf{DEN}}, pp_{\mathsf{SDVS}}), (pk_{\mathsf{DV}}, sk_{\mathsf{DV}}) \leftarrow \mathsf{SDVS.DVKGen}(pp_{\mathsf{SDVS}}), pk_V \leftarrow (pk_{\mathsf{DEN}}, pk_{\mathsf{DV}})$
$b' \leftarrow \mathcal{A}^{\mathsf{ADDU},\mathsf{CRPT},\mathsf{COERCE},\mathsf{SIGN},\mathsf{STCRCSIG}}(pp, pk_V)$
if $b' = b$ **return** 1

Fig. 37: $\mathcal{D}_2$ that distinguishes between Game 2.i and Game 2.$(i+1)$.

Lemma 17 (Strong Soundness). CR.SDVS *satisfies strong soundness for a coercion-resistant incoercible strong designated verifier signature scheme if* $\mathcal{H}_1$ *is a random oracle and if the deniable encryption scheme satisfies* IND-CPA *security.*

Proof. First we show that if there exists an adversary $\mathcal{A}$ that makes k_1, k_2 queries to the SIGN oracle and ADDU oracle respectively, such that $\Pr\left[\mathsf{Exp}^{\mathsf{st\text{-}sound}}_{\mathcal{A},\mathsf{CR.SDVS}}(\lambda) = 1\right] = \epsilon$, where ϵ is non-negligible, then we can can build an adversary $\mathcal{A}'$ that breaks the IND-CPA for multiple encryptions security of the deniable encryption scheme DEN with non-negligible probability. We describe the IND-CPA multiple encryptions security game in the proof of Lemma 5. We give the detailed description of $\mathcal{A}'$ in Figure 38, and explain here how $\mathcal{A}'$ works.

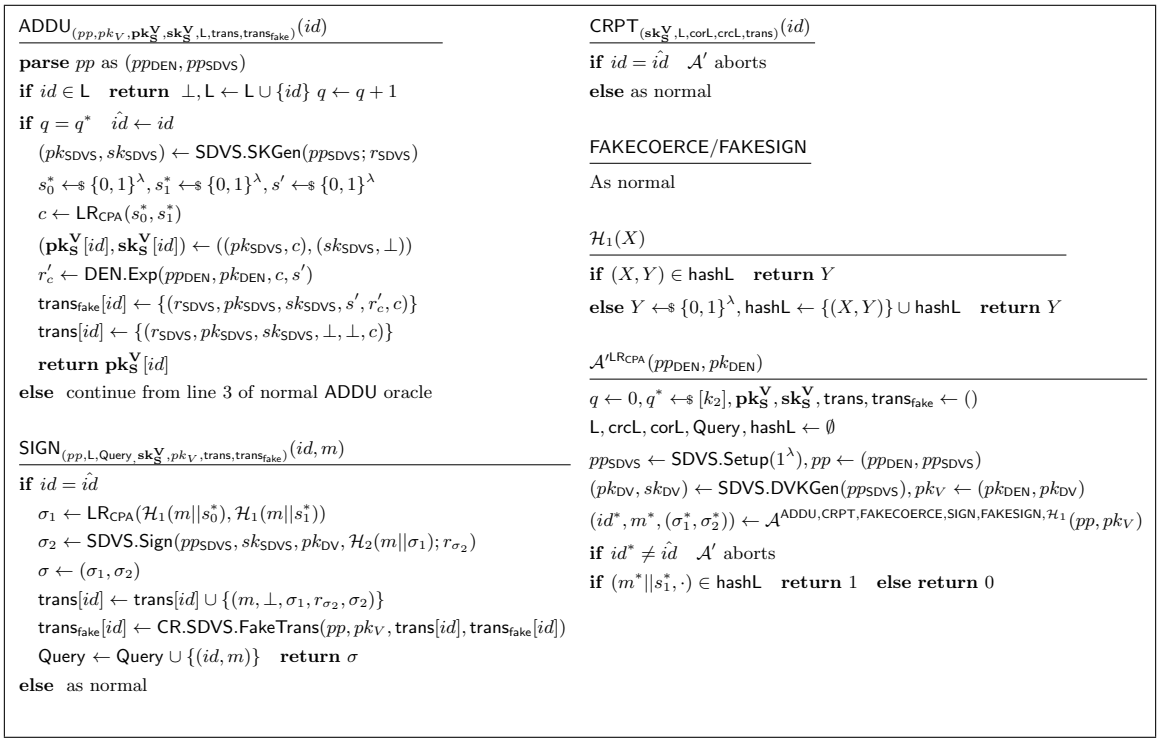

Fig. 38: $\mathcal{A}'$ that breaks the IND-CPA security of the deniable encryption scheme.

First note that all inputs that $\mathcal{A}'$ provides to $\mathcal{A}$ are distributed identically to in the strong soundness experiment for the CR.SDVS construction. The pp, pk_V input to $\mathcal{A}$ are honestly generated and are identical to in the strong soundness experiment. sk_{DEN} is not known, but is never used.

Simulating the ADDU *oracle.* For all signers other than $\hat{id}$, the ADDU oracle is identical to in the strong soundness experiment. For the signer $\hat{id}$, sk_{SDVS} and pk_{SDVS} are generated identically. c is the encryption of $s^*_{b'}$ under pk_{DEN}, where b' is the bit chosen in the IND-CPA experiment. As both s_1^*, s_0^* are chosen identically to s, then c is distributed correctly. The second part of the secret key and real transcript is not known, but we will show that this is not used throughout. The fake transcript is generated as normal using the fake key and the deniable encryption explanation algorithm.

Simulating the SIGN *oracle.* For all signers other than $\hat{id}$, this is identical to in the strong soundness experiment. For the signer $\hat{id}$, σ_1 is the encryption of $\mathcal{H}_1(m||s^*_{b'})$ under pk_A, where b' is the bit chosen in the IND-CPA experiment. This is consistent with the ADDU oracle, because $s^*_{b'}$ is encrypted in the public key. σ_2 and the fake transcript are generated identically to in the strong soundness experiment. Again, the real transcript is not known in full, but is never used.

Simulating the other oracles. If CRPT is input $\hat{id}$, then $\mathcal{A}'$ will abort. For all other signers, the ADDU and SIGN oracles were performed as normal. FAKECOERCE and FAKESIGN do not require

s or the real transcript as the fake key or fake transcript are used instead. The hash oracle $\mathcal{H}_1$ is distributed identically to the random oracle model.

Reduction to IND-CPA *security.* Assume $\mathcal{A}$ is successful. We assume with probability $1/k_2$ that $\mathcal{A}'$ guesses correctly and $\hat{id} = id^*$. Then $\mathcal{A}'$ will not abort because, for $\mathcal{A}$ to be successful, $id^* \in \mathsf{crcL}$ and so cannot have been input to CRPT.

Parse $\mathbf{pk_S^V}[id^*] = (pk_{\mathsf{SDVS}}, c)$. For $\mathcal{A}$ to be successful, $\mathsf{DEN.Dec}(pp_{\mathsf{DEN}}, sk_{\mathsf{DEN}}, \sigma_1^*) = \mathcal{H}_1(m^*||\mathsf{DEN.Dec}(pp_{\mathsf{DEN}}, sk_{\mathsf{DEN}}, c))$. Therefore, $(m^*||\mathsf{DEN.Dec}(pp_{\mathsf{DEN}}, sk_{\mathsf{DEN}}, c), \cdot)$ must be included in hashL, otherwise the probability of this occurring is negligible. We know that $\mathsf{DEN.Dec}(pp_{\mathsf{DEN}}, sk_{\mathsf{DEN}}, c)$ is either s_0^* or s_1^*, and so either $m^*||s_0^*$ or $m^*||s_1^*$ must be included in hashL. As $(id^*, m^*) \notin \mathsf{Query}$, we have not queried either to the signing oracle. As such, the adversary must have queried $\mathcal{H}_1$ directly. All inputs to the adversary are independent of $s_{1-b'}^*$. Therefore, except for the negligible probability of the adversary guessing $s_{1-b'}^*$, they must have queried only $m^*||s_{b'}^*$ to the hash oracle. Therefore, we successfully guess b' in the IND-CPA game.

Therefore, the probability that $\mathcal{A}'$ outputs 1 given $b' = 1$ is greater than $(1 - \mathsf{negl}(\lambda))\epsilon$. The $(1 - \mathsf{negl}(\lambda))$ factor is due to the negligible chance that $\mathcal{A}$ wins without querying $m^*||s_1^*$ to the hash oracle. The probability that $\mathcal{A}'$ outputs 1 given $b' = 0$ is $\leq \mathsf{negl}(\lambda)$, as all inputs to $\mathcal{A}$ are independent of s_1^* and so the chance that $\mathcal{A}$ has input this to the hash oracle is negligible. Therefore, the IND-CPA multiple encryptions advantage is $\geq k_2((1 - \mathsf{negl}(\lambda))\epsilon - \mathsf{negl}(\lambda))$, which is non-negligible. By contradiction, our CR.SDVS construction satisfies strong soundness.

6 Conclusion

We introduced and defined incoercible signatures, and presented an accompanying security model. Specifically, our security model captures a strong notion of incoercibility, coercion-resistance, and we contributed an incoercible signature scheme construction that provably satisfies coercion-resistance. Additionally, our security model captures both strong and weak receipt-freeness. Though these are weaker notions of security than coercion-resistance, we note that they may be sufficient in some application scenarios. For example, strong receipt-freeness is sufficient if it can be assumed that the attacker will not attempt to produce signatures on behalf of coerced signers. Moreover, our strong receipt-freeness construction is more efficient than our coercion-resistant construction, demonstrating that efficiency/security trade-offs are possible. We comment that even more efficient weak receipt-free constructions may be possible, and leave this as an open problem.

In this work, we explore incoercibility in the context of standard digital signature schemes, and additionally show that our syntax and security model can be extended in an intuitive way to the designated verifier signature scheme setting. We also present a construction that satisfies our security model in this setting. An interesting area of future research is to consider incoercibility in the context of other signing and anonymity protocols, for example, group and ring signatures.

References

1. Shweta Agrawal, Shafi Goldwasser, and Saleet Mossel. Deniable fully homomorphic encryption from learning with errors. In *CRYPTO'21*, pages 641–670, 2021.
2. Joël Alwen, Rafail Ostrovsky, Hong-Sheng Zhou, and Vassilis Zikas. Incoercible multi-party computation and universally composable receipt-free voting. In *CRYPTO'15*, pages 763–780, 2015.
3. Mihir Bellare and Sara K. Miner. A forward-secure digital signature scheme. In Michael Wiener, editor, *CRYPTO'99*, pages 431–448, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.
4. Josh Benaloh and Dwight Tuinstra. Receipt-free secret-ballot elections (extended abstract). In *STOC'94*, STOC '94, pages 544–553. Association for Computing Machinery, 1994.
5. Rikke Bendlin, Jesper Buus Nielsen, Peter Sebastian Nordholt, and Claudio Orlandi. Lower and upper bounds for deniable public-key encryption. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT'11*, pages 125–142. Springer Berlin Heidelberg, 2011.
6. Ran Canetti, Cynthia Dwork, Moni Naor, and Rafail Ostrovsky. Deniable encryption. In Burton S. Kaliski, editor, *CRYPTO'97*, pages 90–104, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.
7. Ran Canetti and Rosario Gennaro. Incoercible multi-party computation. In *FOCS'96*, pages 504–513. IEEE, 1996.

8. Ran Canetti, Sunoo Park, and Oxana Poburinnaya. Fully deniable interactive encryption. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO'20*, pages 807–835, Cham, 2020.
9. Pyrros Chaidos, Véronique Cortier, Georg Fuchsbauer, and David Galindo. Beleniosrf: a non-interactive receipt-free electronic voting scheme. In *CCS'16– ACM SIGSAC Conference on Computer and Communications Security*, pages 1614–1625, 2016.
10. Andrea Coladangelo, Shafi Goldwasser, and Umesh Vazirani. Deniable encryption in a quantum world. In *STOC'22*, pages 1378–1391, 2022.
11. Stéphanie Delaune, Steve Kremer, and Mark Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17(4):435–487, 2009.
12. Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In Gilles Brassard, editor, *CRYPTO'89*, pages 307–315, New York, NY, 1990. Springer New York.
13. Yevgeniy Dodis, Jonathan Katz, Shouhuai Xu, and Moti Yung. Key-insulated public key cryptosystems. In Lars R. Knudsen, editor, *EUROCRYPT'02*, pages 65–82, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.
14. Yevgeniy Dodis, Jonathan Katz, Shouhuai Xu, and Moti Yung. Strong key-insulated signature schemes. In Yvo G. Desmedt, editor, *PKC'03*, pages 130–144, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.
15. Markus Dürmuth and David Mandell Freeman. Deniable encryption with negligible detection probability: An interactive construction. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, pages 610–626, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.
16. Konrad Durnoga, Jacek Pomykała, and Tomasz Trabszys. Digital signature with secretly embedded warning. *Control and Cybernetics*, 42(4):805–824, 2013.
17. Shafi Goldwasser, Silvio Micali, and Ronald L Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.
18. Johan Håstad, Jakob Jonsson, Ari Juels, and Moti Yung. Funkspiel schemes: an alternative to conventional tamper resistance. In *CCS'00*, pages 125–133. Association for Computing Machinery, 2000.
19. Amir Herzberg, Markus Jakobsson, Stanisław Jarecki, Hugo Krawczyk, and Moti Yung. Proactive public key and signature systems. In *CCS'97*, pages 100–110. Association for Computing Machinery, 1997.
20. Gene Itkis. Cryptographic tamper evidence. In *CCS'03*, CCS '03, pages 355–364, New York, NY, USA, 2003. Association for Computing Machinery.
21. Gene Itkis and Leonid Reyzin. Sibir: Signer-base intrusion-resilient signatures. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO'02*, pages 499–514, Cham, 2002. Springer International Publishing.
22. Gene Itkis and Peng Xie. Generalized key-evolving signature schemes or how to foil an armed adversary. In Jianying Zhou, Moti Yung, and Yongfei Han, editors, *ACNS'03*, pages 151–168, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.
23. Markus Jakobsson, Kazue Sako, and Russell Impagliazzo. Designated verifier proofs and their applications. In *EUROCRYPT'96*, pages 143–154. Springer, 1996.
24. Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-resistant electronic elections. In *WPES'05*, pages 61–70, 2005.
25. Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography*. CRC press, 2014.
26. Aggelos Kiayias, Thomas Zacharias, and Bingsheng Zhang. End-to-end verifiable elections in the standard model. In *EUROCRYPT'15–International Conference on the Theory and Applications of Cryptographic Techniques*, pages 468–498, 2015.
27. M. Kutyłowski and P. Kubiak. Lightweight digital signature with secretly embedded warning. *Control and Cybernetics*, 42(4):825–827, 2013.
28. Fabien Laguillaumie and Damien Vergnaud. Designated verifier signatures: Anonymity and efficient construction from any bilinear map. In *SCN'04*, pages 105–119. Springer, 2004.
29. Helger Lipmaa, Guilin Wang, and Feng Bao. Designated verifier signature schemes: attacks, new security notions and a new construction. In *International Colloquium on Automata, Languages, and Programming*, pages 459–471, 2005.
30. David Naccache, David Pointcheval, and Christophe Tymen. Monotone signatures. In Paul Syverson, editor, *Financial Cryptography*, pages 305–318, Berlin, Heidelberg, 2002.
31. Adam O'Neill, Chris Peikert, and Brent Waters. Bi-deniable public-key encryption. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO'11*, pages 525–542, Berlin, Heidelberg, 2011.
32. Rafail Ostrovsky and Moti Yung. How to withstand mobile virus attacks. In *PODC'91*, pages 51–59. Association for Computing Machinery, 1991.
33. Shahrokh Saeednia, Steve Kremer, and Olivier Markowitch. An efficient strong designated verifier signature scheme. In *ICISC'03*, pages 40–54. Springer, 2003.
34. Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *STOC'14*, pages 475–484, 2014.

35. Ron Steinfeld, Huaxiong Wang, and Josef Pieprzyk. Efficient extension of standard schnorr/rsa signatures into universal designated-verifier signatures. In *PKC'04*, pages 86–100. Springer, 2004.
36. Haibo Tian, Zhengtao Jiang, Yi Liu, and Baodian Wei. A systematic method to design strong designated verifier signature without random oracles. *Cluster computing*, 16(4):817–827, 2013.
37. Dominique Unruh and Jörn Müller-Quade. Universally composable incoercibility. In Tal Rabin, editor, *CRYPTO'10*, pages 411–428, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.