Explainable Arguments

Lucjan Hanzlik[1] and Kamil Kluczniak[1,2]

[1] CISPA Helmholtz Center for Information Security
[2] Stanford University
lucjan.hanzlik@{cispa.saarland}
[3] kamil.kluczniak@{cispa.saarland,stanford.edu}

Abstract. We introduce an intriguing new type of argument systems with the additional property of being explainable. Intuitively by explainable, we mean that given any argument under a statement, and any witness, we can produce the random coins for which the Prove algorithm outputs the same bits of the argument.
This work aims at introducing the foundations for the interactive as well as the non-interactive setting. We show how to build explainable arguments from witness encryption and indistinguishability obfuscation. Finally, we show applications of explainable arguments. Notably we construct deniable chosen-ciphertext secure encryption. Previous deniable encryption scheme achieved only chosen plaintext security.

1 Introduction

Deniability, first introduced by Dolev, Dwork, and Naor [30], is a notion that received a considerable amount of attention because of its application to authentication protocols. This property allows the user to argue against a third party that it did not take part in a protocol execution. The usual argument made by the user to the third party is that the server could simulate a valid communication transcript without actually interacting with the user.

A variant of deniability was considered in the case of encryption schemes [15,62,16], where a public Expl algorithm allows anyone to open any ciphertext to any message without the secret key. Since we can publicly open ciphertexts, the random coins cannot serve as proof that a particular message is encrypted.

A similar concept was recently introduced to ring signatures [57] and called unclaimability. The property states that no one can claim to be the signer of a particular ring signature σ. The premise is similar. There exists an Expl algorithm that allows any of the ring members to generate random coins that can be used to receive the same σ.

Deniability and unclaimability are related notions. In the former, we consider the server malicious because it tries to gain an undeniable proof of an interaction. In the latter, the malicious party is a different user that tries to make it impossible for honest users to explain an interaction/signature. Interestingly, the deniability and unclaimability definitions studied in the literature only consider scenarios where the party producing a transcript/signature/ciphertext is honest, but may eventually become corrupt in the future.

1.1 Contribution

We introduce a new property for argument systems called explainability. Explainability informally resembles deniability and unclaimability. We consider interactive and non-interactive variants of such systems. We show that achieving strong explainability is hard and requires very strong primitives like witness encryption (WE) and indistinguishability obfuscation (iO). Our contribution can be summarized as follows.

New Definitions. We introduce a new property for argument systems that we call explainability, i.e., the ability for anyone with a valid witness wit to compute the random coins coins that "explain" a given argument arg. By "explain," we mean that the witness and coins result in the same argument string $\mathsf{arg} = \mathsf{Prove}(\mathsf{stmt}, \mathsf{wit}; \mathsf{coins})$ or the same transcript of an interaction, given the same instance of the verifier. Thus if one can explain an argument for all witnesses and all coins, then such argument/transcript cannot serve as proof that a particular witness was used. We accounted for certain subtle differences between interactive and non-interactive arguments. In both cases, we consider *malicious prover* explainability, where a prover tries to create a proof that other provers cannot explain with a different but valid witness. In this case, we require the protocol to be unique, in the sense that it is infeasible for a malicious prover to produce two different arguments (or transcripts) that the verifier accepts given the same statement and random coins. For the interactive case, we also consider a *malicious verifier* (similar to deniability) that can abort the protocol execution or send corrupt messages to make it impossible for provers with a different witness to explain the current interaction. Since, in the non-interactive case, there is no interaction with a verifier, we consider a scenario where the common reference string (if used) is maliciously generated. We refer to this case as *malicious setup explainability*. Additionally, we call a (non-)interactive argument system fully explainable, when it is explainable even if both the setup/verifier and the prover are malicious.

Implications. To study the power of explainable arguments we prove several interesting implications of explainable arguments.

- We show that when an argument system is malicious verifier explainable, then it is also witness indistinguishable.
- We show that non-interactive malicious prover explainable arguments and one-way functions imply witness encryption (WE). This result serves us as evidence that constructing such arguments is difficult and requires strong cryptographic primitives.

Constructions of Interactive Explainable Arguments. We introduce new properties for witness encryption that we call *robustness* and *plaintext awareness*. Informally, robustness ensures that decryption is independent of which witness is

used. In other words, there do not exist two valid witnesses for which a ciphertext decrypts to a different message (or $\perp$). Plaintext awareness ensures that an encrypter must know the plaintext it encrypted. We then show how to leverage robust witness encryption to construct interactive explainable arguments. The resulting protocol is round-optimal, predictable, and can be instantiated to yield an optimally laconic argument. Given the witness encryption is plaintext aware, we can show that the protocol is zero-knowledge. Finally, assuming the witness encryption is extractably secure, we can show that our protocol is a proof of knowledge.

Constructions of Non-Interactive Explainable Arguments. We show how to construct malicious setup and malicious prover explainable arguments from indistinguishability obfuscation. While malicious prover explainable arguments can trivially be build using techniques from Sahai and Waters [62], the case of malicious setup explainable arguments is more involved and requires us to use dual-mode witness indistinguishable proofs. Furthermore, we show how to build fully explainable arguments, additionally assuming NIZK.

Why Study Explainable Arguments? Argument systems are fundamental primitives in cryptography. While some privacy properties like zero-knowledge already give a strong form of deniability, our notion of explainability is much stronger as it considers the extreme case where the provers' coins are leaked or are chosen maliciously. For example, using our explainable arguments, we can show explainable interactive anonymous authentication schemes, where anonymity is defined similarly as in ring-signature schemes (see Supplementary Material B.1). Notably, we can construct CCA-1 secure encryption with deniability as defined by Sahai and Waters [62], from CPA secure deniable encryption and our explainable arguments assuming random oracles. Our deniable encryption is a variant of the Naor-Yung transform [55], but only rely on witness indistinguishability instead of zero-knowledge which allows us to instantiate this transformation using our explainable arguments.

Malicious Verifier/Setup Explainability. We consider adversaries that are substantially more powerful than what is usually studied in the literature, e.g., in deniable authentication schemes or ring-signatures. In particular, in our case, the user can deny an argument even when the adversary asks to reveal the user's random coins used to produce the argument. Immediate real-world examples of such powerful adversaries are rogue nation-state actors that might have the right to confiscate a user's hardware and apply effectual forensics techniques to obtain the random seeds as evidence material against the user. We believe that the threat posed by such potent adversaries may prevent the use of e.g., ring-signatures by whistleblowers, as the anonymity notions provided might be insufficient.

Malicious Prover Explainability. The main application we envision for malicious prover explainability is internet voting. An essential part of a sound and fair

voting scheme is to prevent the selling of votes by malicious voters. We note that the "selling votes" issue isn't limited to actual bribery but, perhaps more critically, addresses the issue of forcing eligible voters to vote on a particular candidate. In this case, an authoritarian forces others to deliver evidence that they voted on a particular option or participate in a specific digital event. An authoritarian here may be an abusive family member, corrupt supervisor, or employer. Our strong unclaimability notion is essential to handle such drastic cases, mainly because users might be coerced or bribed to use specific coins in the protocol.

1.2 Related Work

Explainability of the verifier was used by Bitansky and Choudhuri [8] as a step in proving the existence of deterministic-prover zero-knowledge proofs. In their definition they used the fact that the choices of a verifier can be "explained" by outputting random coins that will lead to the same behaviour. This later can be used to transform the system to be secure even against a malicious verifier. In contrary, we consider the explainability of the prover. While arguments with our type of explainability have not been studied before, there exists some related concepts. Here we give an overview of the related literature.

Deniable Authentication. Dolev, Dwork, and Naor [30] first introduced the concept of deniability. The first formal definition is due to Dwork, Naor, and Sahai [32]. Deniability was studied in numerous works [54,47,25] in the context of authentication protocols. The concept was later generalized to authenticated key exchange and was first formally defined by Di Raimondo, and Genaro [26]. Since then deniable key exchange protocols got much attention from the community [24,11,48,28,68,67,45,27,65,50,64,66]. In such protocols, deniability is informally defined as a party's ability to simulate the transcript of interaction without actually communicating with another party. Since each party can generate a transcript itself, the transcript cannot be used as proof to a third party that the interaction took place. At a high level, deniability is very similar to zero-knowledge, but it is important to mention that Pass [58] showed some subtle differences between both notions.

Deniable Encryption. Deniable encryption was first introduced by Canetti, Dwork, Naor, and Ostrovsky [15]. Here we deal with a "post" compromise situation, where an honest encrypter may be forced to "open" a ciphertext. In other words, given a ciphertext, it should be possible to show a message and randomness that result in the given ciphertext. Deniable encryption was intensively studied [7,56,62,20,21,1,22,41]. Very recently, Canetti, Park, and Poburinnaya [16] generalize deniable encryption to the case where multiple parties are compromised and show constructions also assuming indistinguishability obfuscation.

Ring Signatures. Early forms of deniability were the main motivation for the work of Rivest, Shamir, and Tauman [60], which introduces the concept of ring

signatures. This early concept took into account a relaxed form of deniability where only the secret key of a user may leak. Very recently [57] extended ring signatures with additional deniability properties. For example, they show a signer deniable ring signature where any signer may generate random coins that, together with its secret key, will result in the given signature. However, they require to assume the prover is honest at the moment of signature generation. In our argument setting, we do not make such assumptions.

We are the first to study arguments with unclaimability and deniability properties that allow denying executing a protocol even when the prover is forced to reveal all its random coins or where the prover chooses its coins maliciously. Previous works mostly address a post-compromise setting, whereas some of our explainability notions take into account malicious prover. We believe that our primitives may find applications in protocols as a means of providing consistency checks or anonymous authentication of the votes. For example, the protocols from [17,61] rely on a trusted party to verify a voter's signature. That party knows the user's vote. Using our explainable arguments, we can build (see full paper) a simple anonymous authentication protocol without degrading receipt freeness of the voting scheme, and in effect, remove the trust assumption in terms of privacy.

Receipt Freeness and Coertion Resistance in Voting Schemes. Some of our definitions and potential application are tightly connected to voting schemes. In particular, our definition of malicious prover explainability poses the same requirements, at a high level, for an argument system as receipt freeness or coercion resistance in voting schemes [6,63,46,53]. Since we focus on a single primitive, our definitions are much simpler in comparison to complex voting systems. For example, the definition from [17] involves numerous oracles, and defines a set of parties, and assumes trusted parties. Our definition for malicious prover explainability is simple and says that it is infeasible to produce two different arguments under the same statement that verify incorrectly.

Outline of the Paper. In Section 3 we give definitions of explainable argument systems. In Section 4 we construct non-interactive explainable arguments. In Section 5 we introduce robust witness encryption, and apply it to build interactive explainable arguments. Finally, in Section 6, we show how to apply explainable arguments to construct deniable CCA-secure encryption. In the full paper, we recall all definitions for the primitives in the preliminaries section, show an explainable anonymous authentication protocol, and all security proofs.

2 Preliminaries

Notation We denote execution of an algorithm Alg on input x as $a \leftarrow \mathsf{Alg}(x)$ were the output is assigned to a. Unless said otherwise, we will assume that algorithms are probabilistic and choose some random coins internally. In some cases, however, we will write $\mathsf{Alg}(.; r)$ to denote that Alg proceeds deterministically on

input a seed $r \in \{0,1\}^s$ for some integer s. We denote an execution of a protocol between parties V and P, by $\langle \mathsf{Prove}(.) \rightleftharpoons \mathsf{Verify}(.) \rightarrow x \rangle = \mathsf{trans}$, where x is the output of Verify after completion of the protocol, and trans is the transcript of the protocol. A transcript trans contains all messages send between Prove and Verify and the input of Verify. We write $\mathsf{View}(\mathsf{Prove}(.) \rightleftharpoons \mathsf{Verify}(.))$ to denote the view of Verify. The view contains the transcript, all input to Verify including its random coins and its internal state. W say that a function $\mathsf{negl} : \mathbb{N} \mapsto \mathbb{R}^+$ is negligible if for every constant $c > 0$ there exists a integer $N_c \in \mathbb{N}$ such that for all $\lambda > N_c$ we have $\mathsf{negl}(\lambda) < \lambda^{-c}$.

Standard Definitions We use a number of standard cryptographic tools throughout the paper, including: pseudorandom generators and Goldreich-Levin hardcore bits [39], existential unforgeable and unique signature schemes [42,37], zero-knowledge (ZK) and witness-indistinguishable (WI) argument systems, non-interactive ZK arguments from non-falsifiable assumptions [35], dual-mode witness-indistinguishable proofs [43], CCA1 secure and publicly deniable encryption [62], witness encryption [36] and extractable witness encryption [40], indistinguishability obfuscation [3], and punctured pseudorandom functions [13,14,49].

3 Explainable Arguments

In this section, we introduce the security notions for explainable arguments.

3.1 Interactive Explainable Arguments

In an interactive argument system, the prover uses a witness wit for statement stmt to convince the verifier that the statement is true. The communication between the prover and the verifier creates a transcript trans that contains all the exchanged messages. An interactive explainable argument system allows a prover with a different witness wit^* to generate random coins coins for which $\mathsf{Prove}(\mathsf{stmt}, \mathsf{wit}^*; \mathsf{coins})$ interacting with the same instance of the verifier (i.e., the verifier uses the same random coins) creates the same transcript trans. In other words, this means that any prover with a valid witness can provide random coins that would explain the interaction in trans. More formally.

Definition 1 (Interactive Explainable Arguments). *An interactive argument system $\Pi_{\mathcal{R}} = (\mathsf{Prove}, \mathsf{Verify})$ for language $\mathcal{L}_{\mathcal{R}}$ is an interactive explainable argument system if there exists an additional* Expl *algorithm:*

- $\mathsf{Expl}(\mathsf{stmt}, \mathsf{wit}, \mathsf{trans})$*: takes as input a statement* stmt*, any valid witness* wit *(i.e.* $\mathcal{R}(\mathsf{stmt}, \mathsf{wit}) = 1$*) and transcript* trans*, and outputs* $\mathsf{coins} \in \mathbf{Coin}_{\mathsf{Prove}}$ *(i.e. coins that are in the space of the randomness used in* Prove*),*

which satisfies the correctness definition below.

Definition 2 (Correctness). *For all security parameter λ, for all statements $\mathsf{stmt} \in \mathcal{L}_{\mathcal{R}}$, for all $\mathsf{wit}, \mathsf{wit}^*$ such that $\mathcal{R}(\mathsf{stmt}, \mathsf{wit}) = \mathcal{R}(\mathsf{stmt}, \mathsf{wit}^*) = 1$, we have*

$$\langle \mathsf{Verify}(\mathsf{stmt}) \rightleftharpoons \mathsf{Prove}(\mathsf{stmt}, \mathsf{wit}) \rangle =$$
$$\langle \mathsf{Verify}'(\mathsf{stmt}; \mathsf{trans}) \rightleftharpoons \mathsf{Prove}(\mathsf{stmt}, \mathsf{wit}^*; \mathsf{coins}_E) \rangle = \mathsf{trans},$$

where $\mathsf{coins}_E \leftarrow \mathsf{Expl}(\mathsf{stmt}, \mathsf{wit}^*, \mathsf{trans})$ and $\mathsf{coins}_E \in \mathbf{Coin}_{\mathsf{Prove}}$ and Verify' sends its messages as in trans as long as Prove answers as is trans. If the output of Prove do not match trans, then Verify' aborts and outputs $\perp$.

Remark 1. Note that a naive way to implement the Expl algorithm would be to set coins_E and make the Prove algorithm to "replay" the messages. However, this is obviously a scheme that would not be desirable, since an adversary could easily distinguish such coins from honest ones. Therefore we require that $\mathsf{coins}_E \in \mathbf{Coin}_{\mathsf{Prove}}$ to ensure that coins_E can be given as input to an honest Prove algorithm.

The above definition constitutes a correctness definition for explainable arguments and assumes that all parties are honest. Informally, we require that given a witness and a transcript of an interaction between a verifier and a prover (with a possibly different witness), Expl generates coins such that a honest prover returns the same messages given that the verifier send its messages as in trans.

Below we describe explainability of a malicious verifier. Roughly speaking, this property says that a transcript produced during an execution with a malicious verifier, and a honest prover P, should be explainable. The goal of a verifier, is to send such messages to the prover P, that P sends such responses that no other prover (with a different witness) would send. If the adversary succeeds then the transcript (possibly with P's random coins) can be used as a proof to a third party, that P indeed took part in the communication. Remind that P may be forced to reveal its random coins after completing the protocol.

Definition 3 (Malicious Verifier Explainability). *For a security parameter λ, we define the advantage $\mathsf{Adv}^{\mathsf{MVExpl}}_{\mathcal{A}}(\lambda)$ of an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ as*

$$1 - \Pr[\langle \mathcal{A}_3(\mathsf{stmt}; \mathsf{coins}_{\mathcal{A}}) \rightleftharpoons \mathsf{Prove}(\mathsf{stmt}, \mathsf{wit}^*; \mathsf{coins}_P) \rangle = \mathsf{trans}], \text{ where}$$

$$\begin{aligned} (\mathsf{stmt}, \mathsf{wit}, \mathsf{wit}^*, \mathsf{st}) &\leftarrow \mathcal{A}_1(\lambda), \\ \mathsf{trans} = \langle \mathsf{coins}_{\mathcal{A}} \leftarrow \mathcal{A}_2(\mathsf{stmt}; \mathsf{st}) &\rightleftharpoons \mathsf{Prove}(\mathsf{stmt}, \mathsf{wit}) \rangle, \\ \mathsf{coins}_P &\leftarrow \mathsf{Expl}(\mathsf{stmt}, \mathsf{wit}^*, \mathsf{trans}), \\ \mathsf{wit} \neq \mathsf{wit}^*, \quad \mathcal{R}(\mathsf{stmt}, \mathsf{wit}) &= \mathcal{R}(\mathsf{stmt}, \mathsf{wit}^*) = 1, \end{aligned}$$

where the probability is taken over the random coins of Prove. Furthermore, $\mathcal{A}_3$ sends the same messages to Prove as in trans as long as the responses from the prover are as in trans.

We say that an interactive argument system is malicious verifier explainable *if for all adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ such that $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$ are PPT algorithms*

there exists a negligible function $\mathsf{negl}(.)$ *such that* $\mathsf{Adv}^{\mathsf{MVExpl}}_{\mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda)$. *We say that the argument system is* malicious verifier statistically explainable *if the above holds for an unbounded adversary* $\mathcal{A}$.

Let us now consider a scenario where proving ownership of an argument is beneficial to the prover, but at the same time, the system requires the proof to be explainable. A malicious prover tries to prove the statement in a way that makes it impossible for others to "claim" the generated proof. For this property, it is easy to imagine a malicious prover that sends such messages to the verifier, that the verifier accepts, and no other honest prover would ever send such messages. In practice, we may imagine that an adversary runs a different implementation of the prover, for which the distribution of the sent messages deviate from the distribution of the original implementation. Later to "claim" the transcript that adversary may prove that the transcript is indeed the result of the different algorithm, not the honest one. Note that such a "claim" is sound if an honest prover would never produce such messages. To prevent such attacks, we require that there is only one (computationally feasible to find) valid way a prover can respond to the messages from an honest verifier.

Definition 4 (Uniqueness/Malicious Prover Explainability). *We define the advantage* $\mathsf{Adv}^{\mathsf{MPExpl}}_{\mathcal{A}}(\lambda)$ *of an adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ *as*

$$1 - \Pr\left[\begin{array}{l}\langle 1 = \mathsf{Verify}(\mathsf{stmt}; \mathsf{coins}_V) \rightleftharpoons \mathcal{A}_2(\mathsf{st}_1) \rightarrow \mathsf{st}_2\rangle \\ \quad \neq \langle 1 = \mathsf{Verify}(\mathsf{stmt}; \mathsf{coins}_V) \rightleftharpoons \mathcal{A}_3(\mathsf{st}_2)\rangle\end{array}\right],$$

where $\mathsf{st}_1, \mathsf{stmt} \leftarrow \mathcal{A}_1(\lambda)$ *and the probability is taken over the coins* coins_V.

We say that an interactive argument system is malicious prover explainable *if for all PPT adversaries* $\mathcal{A}$ *there exists a negligible function* $\mathsf{negl}(.)$ *such that* $\mathsf{Adv}^{\mathsf{MPExpl}}_{\mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda)$. *We say that the system is* malicious prover statistically explainable *if the above holds for an unbounded* $\mathcal{A}$.

Theorem 1. *If* $(\mathsf{Prove}, \mathsf{Verify}, \mathsf{Expl})$ *is a malicious verifier (statistical) explainable argument system then it is also (statistical) witness indistinguishable.*

Definition 5. *We say that an interactive argument system is fully explainable if it is malicious prover explainable and malicious verifier explainable.*

3.2 Non-Interactive Explainable Arguments

Here we present definitions for non-interactive explainable arguments. Similar to the interactive case, we begin by defining what it means that a system is explainable.

Definition 6 (Non-Interactive Explainable Arguments). *A non-interactive argument system* $\Pi_{\mathcal{R}} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ *for language* $\mathcal{L}_{\mathcal{R}}$ *is a non-interactive explainable argument system if there exists an additional* Expl *algorithm:*

– $\mathsf{Expl}(\mathsf{crs}, \mathsf{stmt}, \mathsf{wit}, \arg)$*: takes as input a statement* stmt*, any valid witness* wit *and an argument* $\arg$*, and outputs random coins* coins

which satisfies the correctness definition below.

Definition 7 (Correctness). *For all security parameter* λ*, for all statements* $\mathsf{stmt} \in \mathcal{L}_{\mathcal{R}}$*, for all* $\mathsf{wit}, \mathsf{wit}^*$ *such that* $\mathcal{R}(\mathsf{stmt}, \mathsf{wit}) = \mathcal{R}(\mathsf{stmt}, \mathsf{wit}^*) = 1$*, for all random coins* $\mathsf{coins}_P \in \mathbf{Coin}_{\mathsf{Prove}}$*, we have*

$$\mathsf{Prove}(\mathsf{crs}, \mathsf{stmt}, \mathsf{wit}; \mathsf{coins}_P) = \mathsf{Prove}(\mathsf{crs}, \mathsf{stmt}, \mathsf{wit}^*; \mathsf{coins}_E)$$

where $\mathsf{coins}_E \leftarrow \mathsf{Expl}(\mathsf{crs}, \mathsf{stmt}, \mathsf{wit}^*, \arg)$*,* $\mathsf{coins}_E \in \mathbf{Coin}_{\mathsf{Prove}}$ *and* $\mathsf{crs} \leftarrow \mathsf{Setup}(\lambda)$*.*

Now we define malicious setup explainability. Note that a malicious verifier cannot influence the explainability of an argument because there is no interaction with the prover. Hence, the malicious verifier from the interactive setting is replaced with an untrusted setup. An adversary might generate parameters that result in the Expl algorithm to output coins yielding a different argument or even failing on certain witnesses. In some sense, we can think of the adversary as wanting to subvert the common reference string against deniability of certain "targeted" witnesses.

Definition 8 (Malicious Setup Explainability). *We define the advantage* $\mathsf{Adv}^{\mathsf{MSExpl}}_{\mathcal{A}}(\lambda)$ *of an adversary* $\mathcal{A}$ *by the following probability*

$$1 - \Pr\left[\arg^* = \arg : \begin{array}{c} (\mathsf{stmt}, \mathsf{wit}, \mathsf{wit}^*, \mathsf{crs}) \leftarrow \mathcal{A}(\lambda) \\ \mathsf{wit} \neq \mathsf{wit}^* \\ \mathcal{R}(\mathsf{stmt}, \mathsf{wit}) = \mathcal{R}(\mathsf{stmt}, \mathsf{wit}^*) = 1 \\ \arg \leftarrow \mathsf{Prove}(\mathsf{crs}, \mathsf{stmt}, \mathsf{wit}); \\ \mathsf{coins}_P \leftarrow \mathsf{Expl}(\mathsf{crs}, \mathsf{stmt}, \mathsf{wit}^*, \arg); \\ \arg^* \leftarrow \mathsf{Prove}(\mathsf{crs}, \mathsf{stmt}, \mathsf{wit}^*; \mathsf{coins}_P) \end{array}\right],$$

where the probability is taken over the random coins of the prover Prove*. We say that a non-interactive argument is* malicious setup explainable *if for all PPT adversaries* $\mathcal{A}$ *there exists a negligible function* $\mathsf{negl}(.)$ *such that* $\mathsf{Adv}^{\mathsf{MSExpl}}_{\mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda)$*. We say the that a non-interactive argument is* malicious setup statistically explainable *if the above holds for an unbounded adversary* $\mathcal{A}$*. Moreover, we say that a non-interactive argument is perfectly malicious setup explainable if* $\mathsf{Adv}^{\mathsf{MSExpl}}_{\mathcal{A}}(\lambda) = 0$*.*

Theorem 2. *If there exists a malicious setup explainable non-interactive argument, then there exists a two-move witness-indistinguishable argument, where the verifier's message is reusable. In other words, given a malicious setup explainable non-interactive argument, we can build a private-coin ZAP.*

Malicious prover explainability is defined similarly as in the case of interactive arguments. For the non-interactive setting, it is simpler to formalize the definition, as we simply require the adversary to return two arguments that verify correctly, but their canonical representation is different.

Definition 9 (Uniqueness/Malicious Prover Explainability). *We define the advantage of an adversary $\mathcal{A}$ against malicious prover explainability of* ExArg *as* $\mathsf{Adv}^{\mathsf{MPExpl}}_{\mathcal{A}}(\lambda) = \Pr[\arg_1 \neq \arg_2]$ *where* $\mathsf{crs} \leftarrow \mathsf{Setup}(\lambda)$ *and* $(\mathsf{stmt}, \arg_1, \arg_2) \leftarrow \mathcal{A}(\lambda)$ *are such that* $\mathsf{Verify}(\mathsf{crs}, \mathsf{stmt}, \arg_1) = \mathsf{Verify}(\mathsf{crs}, \mathsf{stmt}, \arg_2)$*, and the probability is over the random coins of* Setup*. We say that a non-interactive argument is* malicious prover explainable *if for all PPT adversaries $\mathcal{A}$ there exists a negligible function* $\mathsf{negl}(.)$ *such that* $\mathsf{Adv}^{\mathsf{MPExpl}}_{\mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda)$*. We say that a non-interactive argument is* malicious prover statistically explainable *if the above holds for an unbounded adversary $\mathcal{A}$. Moreover, we say that an argument system is a perfectly malicious prover explainable if* $\mathsf{Adv}^{\mathsf{MPExpl}}_{\mathcal{A}}(\lambda) = 0$*.*

For full explainability, we combine both malicious prover and malicious verifier explainability.

Definition 10 (Full Explainability). *We define the advantage of an adversary $\mathcal{A}$ against full explainability of* ExArg *by the following probability*

$$\mathsf{Adv}^{\mathsf{FExpl}}_{\mathcal{A}}(\lambda) = \Pr[\arg_1 \neq \arg_2]$$

where $(\mathsf{stmt}, \mathsf{crs}, \arg_1, \arg_2) \leftarrow \mathcal{A}(\lambda)$ *is such that* $\mathsf{Verify}(\mathsf{crs}, \mathsf{stmt}, \arg_1) = \mathsf{Verify}(\mathsf{crs}, \mathsf{stmt}, \arg_2)$*. We say that a non-interactive argument is* full explainable *if for all PPT adversaries $\mathcal{A}$, there exists a negligible function* $\mathsf{negl}(.)$ *such that* $\mathsf{Adv}^{\mathsf{FExpl}}_{\mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda)$*. We say that the non-interactive argument is* full statistically explainable *if the above holds for an unbound adversary $\mathcal{A}$. Moreover, we say that an argument system is perfectly full explainable if* $\mathsf{Adv}^{\mathsf{FExpl}}_{\mathcal{A}}(\lambda) = 0$*.*

Theorem 3. *If* ExArg *is a fully explainable argument, then* ExArg *is a malicious setup and malicious prover explainable argument.*

Theorem 4. *Given that one-way functions and malicious prover selectively sound non-interactive (resp. two-move) arguments for* NP *exist, then there exists a witness encryption scheme for* NP*.*

4 Non-Interactive Explainable Arguments

In this section, we show that it is possible to construct malicious setup explainable non-interactive argument systems from falsifiable assumptions. We also show a fully explainable argument assuming non-interactive zero-knowledge. As both schemes are nearly identical and differ only in several lines, we will denote the lines or specific algorithms with $\circ$ for the malicious setup explainable argument, and with $\dagger$, we denote the code specific for the fully explainable argument.

Scheme 1. (Non-interactive Explainable Argument) Let $\nabla = \circ$ for the malicious setup explainable argument, and $\nabla = \dagger$ for the fully explainable argument. Let DMWI be a dual-mode proof, NIWI be a non-interactive witness indistinguishable proof, Com be an equivocal commitment scheme, Sig be a unique signature scheme, and PRF be a punctured pseudorandom function. We construct the non-interactive argument system $\mathsf{ExArg}^{\nabla} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ as follows.

Circuit for $\mathsf{ProgProve}^1_\circ$ and $\mathsf{ProgProve}^1_\dagger$	Circuit for $\mathsf{ProgVerify}$
Hardwired: $\mathsf{pp}, \mathsf{crs}_{\mathsf{DMWI}}, K$	**Hardwired:** K
Input: $(\mathsf{stmt}, \mathsf{wit})$	**Input:** (stmt)
1°: **if** $\mathsf{DMWI.Verify}(\mathsf{crs}_{\mathsf{DMWI}}, \mathsf{stmt}, \mathsf{wit}) = 0$	1 : $\mathsf{sk}_s \leftarrow \mathsf{PRF.Eval}(K, \mathsf{stmt})$
$1^\dagger$: **if** $R(\mathsf{stmt}, \mathsf{wit}) = 0$	2 : $\mathsf{vk}_s \leftarrow \mathsf{Sig.Setup}(\mathsf{sk}_s)$
2 : **return** $\perp$.	3 : **return** vk_s
3 : **else**	
4 : $\mathsf{sk}_s \leftarrow \mathsf{PRF.Eval}(K, \mathsf{stmt})$	
5 : $\mathsf{arg} \leftarrow \mathsf{Sig.Sign}(\mathsf{sk}_s, \mathsf{stmt})$	
6 : **return** arg	

Fig. 1. Circuits for $\mathsf{ProgProve}^1_\circ$, $\mathsf{ProgProve}^1_\dagger$ and $\mathsf{ProgVerify}$. Note that $\mathsf{ProgProve}$ differ only in line 1.

$\mathsf{Setup}(\lambda, \mathcal{L}_{\mathcal{R}})$**:**

1. Choose $K \leftarrow \mathsf{PRF.Setup}(\lambda)$ and $\mathsf{crs}_{\mathsf{DMWI}} \leftarrow \mathsf{DMWI.Setup}(\lambda, \mathsf{modeSound}; \mathsf{coins}_S)$, where coins_S are random coins.
2. $O_{\mathsf{Prove}} \leftarrow \mathsf{Obf}(\lambda, \mathsf{ProgProve}^1_\nabla[\mathsf{pp}, \mathsf{crs}_{\mathsf{DMWI}}, K]; \mathsf{coins}_P)$, where $\mathsf{ProgProve}^1_\nabla$ is given by Figure 1 and coins_P are random coins.

3°. Define statement $\mathsf{stmt}^\circ_{\mathsf{Setup}}$ as

$$\left\{ \begin{array}{l} \exists_{i\in[2],K,\mathsf{coins}_P} O_{\mathsf{Prove}} \leftarrow \mathsf{Obf}(\lambda, \mathsf{ProgProve}^i_\circ[\mathsf{pp}, \mathsf{crs}_{\mathsf{DMWI}}, K]; \mathsf{coins}_P) \ \ \vee \\ \exists_{\mathsf{mode},\mathsf{coins}_S} \mathsf{crs}_{\mathsf{DMWI}} \leftarrow \mathsf{DMWI.Setup}(\lambda, \mathsf{mode}; \mathsf{coins}_S) \wedge \mathsf{mode} = \mathsf{modeWI} \end{array} \right\}.$$

$3^\dagger$. Define statement $\mathsf{stmt}^\dagger_{\mathsf{Setup}}$ as

$$\{\exists_{K,\mathsf{coins}_P} O_{\mathsf{Prove}} \leftarrow \mathsf{Obf}(\lambda, \mathsf{ProgProve}^1_\dagger[\mathsf{pp}, \mathsf{crs}_{\mathsf{DMWI}}, K]; \mathsf{coins}_P)\}.$$

4. Set $\mathsf{wit}_{\mathsf{Setup}} = (1, K, \mathsf{coins}_P)$.

5°. $\pi \leftarrow \mathsf{NIWI.Prove}(\mathsf{stmt}^\circ_{\mathsf{Setup}}, \mathsf{wit}_{\mathsf{Setup}})$.

$5^\dagger$. $\pi \leftarrow \mathsf{NIZK.Prove}(\mathsf{stmt}^\dagger_{\mathsf{Setup}}, \mathsf{wit}_{\mathsf{Setup}})$.

6. Compute $O_{\mathsf{Verify}} \leftarrow \mathsf{Obf}(\lambda, \mathsf{ProgVerify}[K])$ and output $\mathsf{crs} = (O_{\mathsf{Prove}}, O_{\mathsf{Verify}}, \mathsf{pp}, \mathsf{etd}, \mathsf{crs}_{\mathsf{DMWI}}, \pi)$.

$\mathsf{Prove}(\mathsf{crs}, \mathsf{stmt}, \mathsf{wit}; r)$**:**

1°. Set $\mathsf{stmt}^\circ_{\mathsf{Setup}}$ as in the setup algorithm.

$1^\dagger$. Set $\mathsf{stmt}^\dagger_{\mathsf{Setup}}$ as in the setup algorithm.

2°. If $\mathsf{NIWI.Verify}(\mathsf{stmt}^\circ_{\mathsf{Setup}}, \pi) = 0$ return $\perp$.

$2^\dagger$. If $\mathsf{NIZK.Verify}(\mathsf{stmt}^\dagger_{\mathsf{Setup}}, \pi) = 0$ return $\perp$.

3°. Run $\mathsf{wit}' \leftarrow \mathsf{DMWI.Prove}(\mathsf{crs}_{\mathsf{DMWI}}, \mathsf{stmt}, \mathsf{wit}; r)$ and $\mathsf{arg} \leftarrow O_{\mathsf{Prove}}(\mathsf{stmt}, \mathsf{wit}')$.

$3^\dagger$. Run $\mathsf{arg} \leftarrow O_{\mathsf{Prove}}(\mathsf{stmt}, \mathsf{wit})$.

4. Run $\mathsf{vk}_s \leftarrow O_{\mathsf{Verify}}(\mathsf{stmt})$.
5. If $\mathsf{Sig.Verify}(\mathsf{vk}_s, \mathsf{arg}, \mathsf{stmt}) \neq 1$ return $\perp$.

6. Otherwise, return arg.

$\mathsf{Verify}(\mathsf{crs}, \mathsf{stmt}, \mathrm{arg})$**:**

1. Run $\mathsf{vk}_s \leftarrow O_{\mathsf{Verify}}(\mathsf{stmt})$.
2. Output $\mathsf{Sig.Verify}(\mathsf{vk}_s, \mathsf{sig}, \mathsf{msg})$

$\mathsf{Expl}(\mathsf{crs}, \mathsf{stmt}, \mathsf{wit}, \mathrm{arg})$**:**

1. Output 0.

Circuit for $\mathsf{ProgProve}^2_\circ$ and $\mathsf{ProgProve}^2_\dagger$	Circuit for $\mathsf{ProgVerify}^*$
Hardwired: $\mathsf{crs}_{\mathsf{DMWI}}, \mathsf{pp}$	**Hardwired:** $\mathsf{stmt}^*, \mathsf{vk}^*_s,$
$K_{\mathsf{stmt}^*} = \mathsf{PRF.Puncture}(K, \mathsf{stmt}^*)$	$K_{\mathsf{stmt}^*} = \mathsf{PRF.Puncture}(K, \mathsf{stmt}^*)$
Input: $(\mathsf{stmt}, \mathsf{wit}, r)$	**Input:** (stmt)
1°: **if** $\mathsf{DMWI.Verify}(\mathsf{crs}_{\mathsf{DMWI}}, \mathsf{stmt}, \mathsf{wit}) = 0$	1 : **if** $\mathsf{stmt} = \mathsf{stmt}^*$
$1^\dagger$: **if** $R(\mathsf{stmt}, \mathsf{wit}) = 0$	2 : **return** vk^*_s
2 : **return** $\perp$.	3 : **else**
3 : **else**	4 : $\mathsf{sk}_s \leftarrow \mathsf{PRF.Eval}(K_{\mathsf{stmt}^*}, \mathsf{stmt})$
4 : $\mathsf{sk}_s \leftarrow \mathsf{PRF.Eval}(K_{\mathsf{stmt}^*}, \mathsf{stmt})$	5 : $\mathsf{vk}_s \leftarrow \mathsf{Sig.Setup}(\mathsf{sk}_s)$
5 : $\mathrm{arg} \leftarrow \mathsf{Sig.Sign}(\mathsf{sk}_s, \mathsf{stmt})$	6 : **return** vk_s
6 : **return** arg	

Fig. 2. Circuits for $\mathsf{ProgProve}^2_\circ$, $\mathsf{ProgProve}^2_\dagger$ and $\mathsf{ProgVerify}^*$ used in the soundness proof of the non-interactive argument.

Theorem 5. *Let* ExArg° *be the system given by Scheme 1. The system* ExArg° *is computationally sound (in the selective setting) assuming indistinguishability obfuscation of* Obf*, pseudorandomness in punctured points of* PRF*, mode indistinguishability of the* DMWI *scheme, and unforgeability of the signature scheme.*

Theorem 6. *Given that the signature scheme* Sig *is unique,* NIWI *is perfectly sound,* DMWI *is a dual-mode proof, and all primitives are perfectly correct, the argument system* ExArg° *is malicious setup explainable.*

Theorem 7. *Let* $\mathsf{ExArg}^\dagger$ *be the system given by Scheme 1. The system* $\mathsf{ExArg}^\dagger$ *is computationally sound (in the selective setting), assuming indistinguishability obfuscation of* Obf*, pseudorandomness in punctured points of* PRF*, zero-knowledge of the* NIZK *scheme and unforgeability of the signature scheme.*

Theorem 8. *Given that the signature scheme* Sig *is unique,* NIZK *is sound, and all primitives are perfectly correct, argument system* $\mathsf{ExArg}^\dagger$ *is fully explainable.*

Corollary 1. *The scheme is witness indistinguishable against a malicious setup.*

Proof. Witness indistinguishability follows from explainability of the argument system and Theorem 2.

Theorem 9. *Let* ExArg^∇ *be the system given by Scheme 1 for* $\nabla = \circ$ *or* $\nabla = \dagger$. ExArg^∇ *is zero-knowledge in the common reference string model.*

5 Robust-Witness Encryption and Interactive Explainable Arguments

We introduce robust witness encryption and show a generic transformation from any standard witness encryption scheme to a robust witness encryption scheme.

Definition 11 (Robust Witness Encryption). *We call a witness encryption scheme* $\mathsf{WE} = (\mathsf{Enc}, \mathsf{Dec})$ *a robust witness encryption scheme if it is correct, secure and robust as defined below:*

Robustness: *A witness encryption scheme* $(\mathsf{Enc}, \mathsf{Dec})$ *is robust if for all PPT adversaries* $\mathcal{A}$ *there exists a negligible function* $\mathsf{negl}(.)$ *such that*

$$\Pr\left[m_0 \neq m_1 : \begin{array}{c} \mathcal{R}(\mathsf{stmt}, \mathsf{wit}_0) = \mathcal{R}(\mathsf{stmt}, \mathsf{wit}_1) = 1 \wedge \\ (\mathsf{stmt}, \mathsf{ct}, \mathsf{wit}_0, \mathsf{wit}_1) \leftarrow \mathcal{A}(\lambda); \\ m_0 \leftarrow \mathsf{Dec}(\mathsf{stmt}, \mathsf{wit}_0, \mathsf{ct}) \\ m_1 \leftarrow \mathsf{Dec}(\mathsf{stmt}, \mathsf{wit}_1, \mathsf{ct}) \end{array}\right] \leq \mathsf{negl}(\lambda),$$

We call the scheme perfectly robust if the above probability is always zero.

Below we define plaintext awareness [5], but tailored to the case of witness encryption.

Definition 12 (Plaintext Aware Witness Encryption). *Let* $\mathsf{WE} = (\mathsf{Enc}, \mathsf{Dec})$ *be a witness encryption scheme. We extend the scheme with an algorithm* Verify *that on input a ciphertext* ct *and a statement* stmt *outputs a bit indicating whether the ciphertext is in the ciphertext space or not. Additionally we define an algorithm* Setup *that on input the security parameter* λ *outputs a common reference string* crs*, and an algorithm* Setup^* *that additionally outputs* τ*. We say that the witness encryption scheme for a language* $\mathcal{L} \in \mathbf{NP}$ *is plaintext aware if for all PPT adversaries* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}(.)$ *such that*

$$\begin{aligned} &|\Pr[\mathcal{A}(\mathsf{crs}) = 1 : \mathsf{crs} \leftarrow \mathsf{Setup}(\lambda)] \\ &- \Pr[\mathcal{A}(\mathsf{crs}) = 0 : (\mathsf{crs}, \tau) \leftarrow \mathsf{Setup}^*(\lambda)]| \leq \mathsf{negl}(\lambda), \end{aligned}$$

and there exists a PPT extractor Ext *such that*

$$\Pr\left[\mathsf{msg} \leftarrow \mathsf{Ext}(\mathsf{stmt}, \mathsf{ct}, \tau) : \begin{array}{c} (\mathsf{crs}, \tau) \leftarrow \mathsf{Setup}^*(\lambda); \\ (\mathsf{ct}, \mathsf{stmt}) \leftarrow \mathcal{A}(\mathsf{crs}); \\ \mathsf{Verify}(\mathsf{stmt}, \mathsf{ct}) = 1 \end{array}\right] \leq 1 - \mathsf{negl}(\lambda)$$

where for all witnesses wit *such that* $\mathcal{R}(\mathsf{stmt}, \mathsf{wit}) = 1$ *we have* $\mathsf{msg} = \mathsf{Dec}(\mathsf{ct}, \mathsf{wit})$*, and the probability is taken over the random coins of* Setup *and* Setup^**.*

Scheme 2. (Generic Transformation) Let $\mathsf{WE} = (\mathsf{Enc}, \mathsf{Dec})$ be a witness encryption scheme and $\mathsf{NIZK} = (\mathsf{NIZK.Prove}, \mathsf{NIZK.Verify})$ be a proof system. We construct a robust witness encryption scheme WE_{rob} as follows.

$\mathsf{Enc}_{rob}(\lambda, \mathsf{stmt}, \mathsf{msg})$**:**

1. Compute $\mathsf{ct}_{\mathsf{msg}} \leftarrow \mathsf{WE.Enc}(\lambda, \mathsf{stmt}, \mathsf{msg})$
2. Let $\mathsf{stmt}_{\mathsf{NIZK}}$ be defined as $\{\exists_{\mathsf{msg}} \ \mathsf{ct}_{\mathsf{msg}} \leftarrow \mathsf{WE.Enc}(\lambda, \mathsf{stmt}, \mathsf{msg})\}$
3. Compute $\pi \leftarrow \mathsf{NIZK.Prove}(\mathsf{stmt}_{\mathsf{NIZK}}, \mathsf{wit})$ using witness $\mathsf{wit} = (\mathsf{msg})$
4. Return $\mathsf{ct} = (\mathsf{ct}_{\mathsf{msg}}, \pi)$.

$\mathsf{Dec}_{rob}(\mathsf{stmt}, \mathsf{wit}, \mathsf{ct})$**:**

1. Set the statement $\mathsf{stmt}_{\mathsf{NIZK}}$ as $\{\exists_{\mathsf{msg}} \ \mathsf{ct}_{\mathsf{msg}} \leftarrow \mathsf{WE.Enc}(\lambda, \mathsf{stmt}, \mathsf{msg})\}$
2. If $\mathsf{NIZK.Verify}(\mathsf{stmt}_{\mathsf{NIZK}}, \pi) = 0$, then return $\perp$. Otherwise return $\mathsf{WE.Dec}(\mathsf{stmt}, \mathsf{wit}, \mathsf{ct}_{\mathsf{msg}})$

Theorem 10 (Security and Extractability). *Scheme 2 is a (extractably) secure witness encryption if* WE *is a (extractably) secure witness encryption, and* NIZK *is zero-knowledge (in the common reference string or RO model).*

Theorem 11 (Robustness and Plaintext Awareness). *Scheme 2 is robust if the witness encryption scheme* WE *is perfectly correct, and the* NIZK *proof system is perfectly sound (in the common reference string or RO model). If the* NIZK *proof system is a proof of knowledge (in the common string or RO model), then Scheme 2 is plaintext aware.*

5.1 Fully Explainable Arguments from Robust Witness Encryption

In this subsection, we will tackle the problem of constructing fully explainable arguments. The system is described in more detail by scheme 3.

Scheme 3. (Interactive Explainable Argument) The argument system consists of Prove, Verify and Expl, where the protocol between Prove and Verify is specified as follows. Prove takes as input a statement stmt and a witness wit, and Verify takes as input stmt. First Verify chooses $r \leftarrow\!\!\$ \{0,1\}^{\lambda}$, computes $\mathsf{ct} \leftarrow \mathsf{Enc}_{rob}(\lambda, \mathsf{stmt}, r)$ and sends ct to Prove. Then Prove computes $\mathsf{arg} \leftarrow \mathsf{Dec}_{rob}(\mathsf{stmt}, \mathsf{wit}, \mathsf{ct})$ and sends arg to Verify. Finally, Verify returns iff $\mathsf{arg} = r$. The explain algorithm Expl is as follows.

$\mathsf{Expl}(\mathsf{stmt}, \mathsf{wit}, \mathsf{trans})$**:** On input the statement stmt, the witness wit and a transcript trans, output $\perp$.

Theorem 12 (Soundness). *Scheme 3 is an argument system for* NP *language* $\mathcal{L}$ *assuming the witness encryption scheme* WE *for* $\mathcal{L}$ *is secure. Furthermore, if the underlying witness encryption scheme* WE *scheme is extractable, then Scheme 3 is an argument of knowledge.*

Theorem 13 (Zero-Knowledge). *Scheme 3 is zero-knowledge given the underlying witness encryption scheme* WE *is plaintext aware.*

Theorem 14 (Explainability). *Scheme 3 is fully explainable assuming the used witness encryption scheme is robust (or plaintext aware) and correct.*

Remark 2. Scheme 3 is predictable in the sense that the verifier can "predict" the value of the prover's arguments/proof [33]. Furthermore, the protocol is optimally laconic [12], as the verifier can encrypt single bits.

Theorem 15. *Let* WE *be a (non-robust) perfectly correct witness encryption scheme for* NP*. Let* Π *be an interactive public-coin zero-knowledge proof protocol for* NP*. Then there exists a malicious verifier explainable (and witness-indistinguishable) argument for* NP*.*

6 Applications

In this section, we show how to apply explainable arguments. We focus on constructing a CCA1 secure publicly deniable encryption scheme using as a building block malicious verifier explainable arguments. Our transformation is based on the one from Naor and Yung [55] but we replace the NIZK proof system with a NIWI. In the full version we show how to build a deniable anonymous credential scheme from malicious prover explainable arguments. Here we note that the anonymous credential system is a straightforward application of malicious prover explainable arguments and standard signature schemes.

The main idea behind the Naor and Yung construction is to use two CPA secure ciphertexts $\mathsf{ct}_1, \mathsf{ct}_2$ and a NIZK that both contain the same plaintext. The soundness property ensures that a decryption oracle can use either of the secret keys (since the decrypted message would be the same) and zero-knowledge allows the security reduction to change the challenged ciphertext, i.e. change the two CPA ciphertexts. We note that in our approach we replace NIZK with NIWI, that to the best of our knowledge has not been do before.

Scheme 4. (Generic Transformation from CPA to CCA) Let $\mathcal{E} = (\mathsf{KeyGen}_{\mathsf{cpa}}, \mathsf{Enc}_{\mathsf{cpa}}, \mathsf{Dec}_{\mathsf{cpa}})$ be a CPA secure encryption scheme, $(\mathsf{NIWI.Setup}, \mathsf{NIWI.Prove}, \mathsf{NIWI.Verify})$ be a non-interactive witness-indistinguishable proof system. Additionally we define the following statement $\mathsf{stmt}_{\mathsf{cpa}}$ be defined as

$$\{(\exists_{\mathsf{msg}}\, \mathsf{ct}_1 \leftarrow \mathsf{Enc}_{\mathsf{cpa}}(\mathsf{pk}_1, \mathsf{msg}) \ \wedge \ \mathsf{ct}_2 \leftarrow \mathsf{Enc}_{\mathsf{cpa}}(\mathsf{pk}_2, \mathsf{msg})) \ \vee \\ (\exists_{\alpha,\beta} \mathcal{H}_{\mathbb{G}}(\mathsf{ct}_1, \mathsf{ct}_2) = (g^{\alpha}, g^{\beta}, g^{\alpha\cdot\beta}))\},$$

where $\mathcal{H}_{\mathbb{G}}$ is defined as above.

$\mathsf{KeyGen}_{\mathsf{cca1}}(\lambda)$**:**

1. generate CPA secure encryption key pairs $(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{KeyGen}_{\mathsf{cpa}}(\lambda)$ and $(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow \mathsf{KeyGen}_{\mathsf{cpa}}(\lambda)$,
2. generate a common reference string $\mathsf{crs} \leftarrow \mathsf{NIWI.Setup}(\lambda)$,
3. set $\mathsf{pk}_{\mathsf{cca1}} = (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{crs})$ and $\mathsf{sk}_{\mathsf{cca1}} = \mathsf{sk}_1$.

$\mathsf{Enc}_{\mathsf{cca1}}(\mathsf{pk}_{\mathsf{cca1}}, \mathsf{msg})$**:**

1. compute ciphertexts $\mathsf{ct}_1 \leftarrow \mathsf{Enc}_{\mathsf{cpa}}(\mathsf{pk}_1, \mathsf{msg})$ and $\mathsf{ct}_2 \leftarrow \mathsf{Enc}_{\mathsf{cpa}}(\mathsf{pk}_2, \mathsf{msg})$,
2. compute NIWI proof $\Pi \leftarrow \mathsf{NIWI.Prove}(\mathsf{crs}, \mathsf{stmt}_{\mathsf{cpa}}, (\mathsf{msg})$,
3. return ciphertext $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2, \Pi)$.

$\mathsf{Dec}_{\mathsf{cca1}}(\mathsf{sk}_{\mathsf{cca1}}, \mathsf{ct})$**:**

1. return $\perp$ if $\mathsf{NIWI.Verify}(\mathsf{crs}, \mathsf{stmt}_{\mathsf{cpa}}, \Pi) = 0$,
2. return $\mathsf{msg} \leftarrow \mathsf{Dec}_{\mathsf{cpa}}(\mathsf{sk}_1, \mathsf{ct}_1)$.

Theorem 16. *Scheme 4 is an encryption scheme secure against non-adaptive chosen ciphertext attacks (CCA1) in the random oracle model assuming the encryption scheme $\mathcal{E}$ is an encryption scheme secure against chosen plaintext attacks and* NIWI *is a sound and witness indistinguishable proof system.*

Theorem 17. *Scheme 4 is an publicly deniable encryption scheme secure against non-adaptive chosen ciphertext attacks (CCA1) in the random oracle model assuming the encryption scheme $\mathcal{E}$ is an publicly deniable encryption scheme secure against chosen plaintext attacks and* NIWI *is a malicious setup explainable argument system.*

7 Conclusions

Acknowledgements. This work has been partially funded/supported by the German Ministry for Education and Research through funding for the project CISPA-Stanford Center for Cybersecurity (Funding numbers: 16KIS0762 and 16KIS0927).

References

1. D. Apon, X. Fan, and F.-H. Liu. Deniable attribute based encryption for branching programs from LWE. In M. Hirt and A. D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 299–329. Springer, Heidelberg, Oct. / Nov. 2016.
2. L. Babai and S. Moran. Arthur-merlin games: A randomized proof system, and a hierarchy of complexity classes. *Journal of Computer and System Sciences*, 36(2):254 – 276, 1988.
3. B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. P. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. In J. Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 1–18. Springer, Heidelberg, Aug. 2001.
4. B. Barak, S. J. Ong, and S. P. Vadhan. Derandomization in cryptography. In D. Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 299–315. Springer, Heidelberg, Aug. 2003.
5. M. Bellare and P. Rogaway. Optimal asymmetric encryption. In A. D. Santis, editor, *EUROCRYPT'94*, volume 950 of *LNCS*, pages 92–111. Springer, Heidelberg, May 1995.
6. J. C. Benaloh and D. Tuinstra. Receipt-free secret-ballot elections (extended abstract). In *26th ACM STOC*, pages 544–553. ACM Press, May 1994.

7. R. Bendlin, J. B. Nielsen, P. S. Nordholt, and C. Orlandi. Lower and upper bounds for deniable public-key encryption. In D. H. Lee and X. Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 125–142. Springer, Heidelberg, Dec. 2011.
8. N. Bitansky and A. R. Choudhuri. Characterizing deterministic-prover zero knowledge. In R. Pass and K. Pietrzak, editors, *TCC 2020, Part I*, volume 12550 of *LNCS*, pages 535–566. Springer, Heidelberg, Nov. 2020.
9. N. Bitansky and O. Paneth. ZAPs and non-interactive witness indistinguishability from indistinguishability obfuscation. In Y. Dodis and J. B. Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 401–427. Springer, Heidelberg, Mar. 2015.
10. M. Blum, P. Feldman, and S. Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *20th ACM STOC*, pages 103–112. ACM Press, May 1988.
11. J.-M. Bohli and R. Steinwandt. Deniable group key agreement. In P. Q. Nguyen, editor, *Progress in Cryptology - VIETCRYPT 06*, volume 4341 of *LNCS*, pages 298–311. Springer, Heidelberg, Sept. 2006.
12. D. Boneh, Y. Ishai, A. Sahai, and D. J. Wu. Quasi-optimal SNARGs via linear multi-prover interactive proofs. In J. B. Nielsen and V. Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 222–255. Springer, Heidelberg, Apr. / May 2018.
13. D. Boneh and B. Waters. Constrained pseudorandom functions and their applications. In K. Sako and P. Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 280–300. Springer, Heidelberg, Dec. 2013.
14. E. Boyle, S. Goldwasser, and I. Ivan. Functional signatures and pseudorandom functions. In H. Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 501–519. Springer, Heidelberg, Mar. 2014.
15. R. Canetti, C. Dwork, M. Naor, and R. Ostrovsky. Deniable encryption. In B. S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 90–104. Springer, Heidelberg, Aug. 1997.
16. R. Canetti, S. Park, and O. Poburinnaya. Fully deniable interactive encryption. In D. Micciancio and T. Ristenpart, editors, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*, pages 807–835. Springer, Heidelberg, Aug. 2020.
17. P. Chaidos, V. Cortier, G. Fuchsbauer, and D. Galindo. BeleniosRF: A non-interactive receipt-free electronic voting scheme. In E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, editors, *ACM CCS 2016*, pages 1614–1625. ACM Press, Oct. 2016.
18. S. Chakraborty, M. Prabhakaran, and D. Wichs. Witness maps and applications. In A. Kiayias, M. Kohlweiss, P. Wallden, and V. Zikas, editors, *PKC 2020, Part I*, volume 12110 of *LNCS*, pages 220–246. Springer, Heidelberg, May 2020.
19. R. Cramer and V. Shoup. Signature schemes based on the strong RSA assumption. In J. Motiwalla and G. Tsudik, editors, *ACM CCS 99*, pages 46–51. ACM Press, Nov. 1999.
20. D. Dachman-Soled. On the impossibility of sender-deniable public key encryption. Cryptology ePrint Archive, Report 2012/727, 2012. https://eprint.iacr.org/2012/727.
21. D. Dachman-Soled. A black-box construction of a CCA2 encryption scheme from a plaintext aware (sPA1) encryption scheme. In H. Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 37–55. Springer, Heidelberg, Mar. 2014.

22. A. De Caro, V. Iovino, and A. O'Neill. Deniable functional encryption. In C.-M. Cheng, K.-M. Chung, G. Persiano, and B.-Y. Yang, editors, *PKC 2016, Part I*, volume 9614 of *LNCS*, pages 196–222. Springer, Heidelberg, Mar. 2016.
23. A. De Santis, S. Micali, and G. Persiano. Non-interactive zero-knowledge proof systems. In C. Pomerance, editor, *CRYPTO'87*, volume 293 of *LNCS*, pages 52–72. Springer, Heidelberg, Aug. 1988.
24. M. Di Raimondo and R. Gennaro. New approaches for deniable authentication. In V. Atluri, C. Meadows, and A. Juels, editors, *ACM CCS 2005*, pages 112–121. ACM Press, Nov. 2005.
25. M. Di Raimondo and R. Gennaro. New approaches for deniable authentication. *Journal of Cryptology*, 22(4):572–615, Oct. 2009.
26. M. Di Raimondo, R. Gennaro, and H. Krawczyk. Deniable authentication and key exchange. In A. Juels, R. N. Wright, and S. De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 400–409. ACM Press, Oct. / Nov. 2006.
27. Y. Dodis and D. Fiore. Interactive encryption and message authentication. In M. Abdalla and R. D. Prisco, editors, *SCN 14*, volume 8642 of *LNCS*, pages 494–513. Springer, Heidelberg, Sept. 2014.
28. Y. Dodis, J. Katz, A. Smith, and S. Walfish. Composability and on-line deniability of authentication. In O. Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 146–162. Springer, Heidelberg, Mar. 2009.
29. Y. Dodis and A. Yampolskiy. A verifiable random function with short proofs and keys. In S. Vaudenay, editor, *PKC 2005*, volume 3386 of *LNCS*, pages 416–431. Springer, Heidelberg, Jan. 2005.
30. D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography (extended abstract). In *23rd ACM STOC*, pages 542–552. ACM Press, May 1991.
31. C. Dwork and M. Naor. Zaps and their applications. In *41st FOCS*, pages 283–293. IEEE Computer Society Press, Nov. 2000.
32. C. Dwork, M. Naor, and A. Sahai. Concurrent zero-knowledge. In *30th ACM STOC*, pages 409–418. ACM Press, May 1998.
33. A. Faonio, J. B. Nielsen, and D. Venturi. Predictable arguments of knowledge. In S. Fehr, editor, *PKC 2017, Part I*, volume 10174 of *LNCS*, pages 121–150. Springer, Heidelberg, Mar. 2017.
34. U. Feige, D. Lapidot, and A. Shamir. Multiple noninteractive zero knowledge proofs under general assumptions. *SIAM J. Comput.*, 29(1):1–28, Sept. 1999.
35. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A. M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Heidelberg, Aug. 1987.
36. S. Garg, C. Gentry, A. Sahai, and B. Waters. Witness encryption and its applications. In D. Boneh, T. Roughgarden, and J. Feigenbaum, editors, *45th ACM STOC*, pages 467–476. ACM Press, June 2013.
37. R. Gennaro, S. Halevi, and T. Rabin. Secure hash-and-sign signatures without the random oracle. In J. Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 123–139. Springer, Heidelberg, May 1999.
38. O. Goldreich. *Basing Non-Interactive Zero-Knowledge on (Enhanced) Trapdoor Permutations: The State of the Art*, pages 406–421. Springer-Verlag, 2011.
39. O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. In *21st ACM STOC*, pages 25–32. ACM Press, May 1989.
40. S. Goldwasser, Y. T. Kalai, R. A. Popa, V. Vaikuntanathan, and N. Zeldovich. How to run turing machines on encrypted data. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 536–553. Springer, Heidelberg, Aug. 2013.

41. S. Goldwasser, S. Klein, and D. Wichs. The edited truth. In Y. Kalai and L. Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 305–340. Springer, Heidelberg, Nov. 2017.
42. S. Goldwasser and R. Ostrovsky. Invariant signatures and non-interactive zero-knowledge proofs are equivalent (extended abstract). In E. F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 228–245. Springer, Heidelberg, Aug. 1993.
43. J. Groth, R. Ostrovsky, and A. Sahai. Non-interactive zaps and new techniques for NIZK. In C. Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 97–111. Springer, Heidelberg, Aug. 2006.
44. J. Groth, R. Ostrovsky, and A. Sahai. Perfect non-interactive zero knowledge for NP. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 339–358. Springer, Heidelberg, May / June 2006.
45. L. Hanzlik, K. Kluczniak, M. Kutyłowski, and L. Krzywiecki. Mutual restricted identification. In S. Katsikas and I. Agudo, editors, *Public Key Infrastructures, Services and Applications*, pages 119–133, Berlin, Heidelberg, 2014.
46. M. Hirt and K. Sako. Efficient receipt-free voting based on homomorphic encryption. In B. Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 539–556. Springer, Heidelberg, May 2000.
47. M. Jakobsson, K. Sako, and R. Impagliazzo. Designated verifier proofs and their applications. In U. M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 143–154. Springer, Heidelberg, May 1996.
48. S. Jiang and R. Safavi-Naini. An efficient deniable key exchange protocol (extended abstract). In G. Tsudik, editor, *FC 2008*, volume 5143 of *LNCS*, pages 47–52. Springer, Heidelberg, Jan. 2008.
49. A. Kiayias, S. Papadopoulos, N. Triandopoulos, and T. Zacharias. Delegatable pseudorandom functions and applications. In A.-R. Sadeghi, V. D. Gligor, and M. Yung, editors, *ACM CCS 2013*, pages 669–684. ACM Press, Nov. 2013.
50. L. Krzywiecki, K. Kluczniak, P. Kozieł, and N. Panwar. Privacy-oriented dependency via deniable sigma protocol. *Computers & Security*, 79:53 – 67, 2018.
51. A. Lysyanskaya. Unique signatures and verifiable random functions from the DH-DDH separation. In M. Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 597–612. Springer, Heidelberg, Aug. 2002.
52. S. Micali, M. O. Rabin, and S. P. Vadhan. Verifiable random functions. In *40th FOCS*, pages 120–130. IEEE Computer Society Press, Oct. 1999.
53. T. Moran and M. Naor. Receipt-free universally-verifiable voting with everlasting privacy. In C. Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 373–392. Springer, Heidelberg, Aug. 2006.
54. M. Naor. Deniable ring authentication. In M. Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 481–498. Springer, Heidelberg, Aug. 2002.
55. M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd ACM STOC*, pages 427–437. ACM Press, May 1990.
56. A. O'Neill, C. Peikert, and B. Waters. Bi-deniable public-key encryption. In P. Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 525–542. Springer, Heidelberg, Aug. 2011.
57. S. Park and A. Sealfon. It wasn't me! - Repudiability and claimability of ring signatures. In A. Boldyreva and D. Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 159–190. Springer, Heidelberg, Aug. 2019.
58. R. Pass. On deniability in the common reference string and random oracle model. In D. Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 316–337. Springer, Heidelberg, Aug. 2003.

59. C. Peikert and S. Shiehian. Noninteractive zero knowledge for NP from (plain) learning with errors. In A. Boldyreva and D. Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 89–114. Springer, Heidelberg, Aug. 2019.
60. R. L. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In C. Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 552–565. Springer, Heidelberg, Dec. 2001.
61. P. Y. A. Ryan, P. B. Rønne, and V. Iovino. Selene: Voting with transparent verifiability and coercion-mitigation. In J. Clark, S. Meiklejohn, P. Y. Ryan, D. Wallach, M. Brenner, and K. Rohloff, editors, *Financial Cryptography and Data Security*, pages 176–192, Berlin, Heidelberg, 2016.
62. A. Sahai and B. Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In D. B. Shmoys, editor, *46th ACM STOC*, pages 475–484. ACM Press, May / June 2014.
63. K. Sako and J. Kilian. Receipt-free mix-type voting scheme - a practical solution to the implementation of a voting booth. In L. C. Guillou and J.-J. Quisquater, editors, *EUROCRYPT'95*, volume 921 of *LNCS*, pages 393–403. Springer, Heidelberg, May 1995.
64. N. Unger and I. Goldberg. Improved strongly deniable authenticated key exchanges for secure messaging. *Proceedings on Privacy Enhancing Technologies*, 2018(1):21 – 66, 01 Jan. 2018.
65. N. Unger and I. Goldberg. Deniable key exchanges for secure messaging. In I. Ray, N. Li, and C. Kruegel, editors, *ACM CCS 2015*, pages 1211–1223. ACM Press, Oct. 2015.
66. N. Vatandas, R. Gennaro, B. Ithurburn, and H. Krawczyk. On the cryptographic deniability of the Signal protocol. In M. Conti, J. Zhou, E. Casalicchio, and A. Spognardi, editors, *ACNS 20, Part II*, volume 12147 of *LNCS*, pages 188–209. Springer, Heidelberg, Oct. 2020.
67. S. Yamada, N. Attrapadung, B. Santoso, J. C. N. Schuldt, G. Hanaoka, and N. Kunihiro. Verifiable predicate encryption and applications to CCA security and anonymous predicate authentication. In M. Fischlin, J. Buchmann, and M. Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 243–261. Springer, Heidelberg, May 2012.
68. A. C.-C. Yao and Y. Zhao. Deniable internet key exchange. In J. Zhou and M. Yung, editors, *ACNS 10*, volume 6123 of *LNCS*, pages 329–348. Springer, Heidelberg, June 2010.

Supplementary Material

A Preliminaries

Pseudorandom Generators We recall the definitions of pseudorandom generators.

Definition 13. *A pseudorandom generator consists of two algorithms* $\mathsf{SeedGen}$, PRG *with the following syntax.*

$\mathsf{SeedGen}(\lambda)$**:** *On input a security parameter* λ *outputs a seed* s.
$\mathsf{PRG}(\mathsf{s}, i)$**:** *On input a seed* s *and an index* $i \in \mathbb{N}$*, outputs* r.

We define the following property.

Pseudorandomness: *We define the advantage of an adversary* $\mathcal{A}$ *against pseudorandomness of a PRG as follows.*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{A}}(\lambda, n) = |\Pr[1 \leftarrow \mathcal{A}(\lambda, [\mathsf{PRG}(\mathsf{s}, i), i)]_{i=1}^{n}); \mathsf{s} \leftarrow \mathsf{SeedGen}(\lambda)] \\ - \Pr[1 \leftarrow \mathcal{A}(\lambda, [r_i, i]_{i=1}^{n}); [r_i \leftarrow\!\!\$\, U]_{i=1}^{n}]|,\end{aligned}$$

where U *denotes the uniform distribution. We say that a PRG is pseudorandom if for all all PPT adversaries* $\mathcal{A}$ *there exists a negligible function* $\mathsf{negl}(.)$ *such that* $\mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda)$.

Signature Schemes We will recall the standard definition of signature schemes, with a slight twist that the key generation algorithm takes as input a secret key and outputs a verification key. For notational convenience, in our constructions, we will sample secret keys from a secret key space, and compute the verification keys based on those secret keys. We will denote the secret key space $\{0,1\}^p$ where $p = \mathsf{poly}(\lambda)$, where λ is the security parameter.

Definition 14. *A signature scheme* Sig *consists of PPT algorithms* (KeyGen, $\mathsf{Sign}, \mathsf{Verify}$) *with the following syntax.*

$\mathsf{KeyGen}(\mathsf{sk})$**:** *This deterministic algorithm takes as input a secret key* sk *and outputs a verification key* vk.
$\mathsf{Sign}(\mathsf{sk}, \mathsf{m})$**:** *This algorithm takes as input a signing key* sk *and a message* m *and outputs a signature* sig.
$\mathsf{Verify}(\mathsf{vk}, \mathsf{m}, \mathsf{sig})$**:** *This deterministic algorithm takes as input a verification key* vk*, a message* m*, and a signature* sig *and outputs either* 0 *or* 1.

We define the following properties of a signature scheme.

Correctness: *It holds for every security parameter* $\lambda \in \mathbb{N}$*, all* $\mathsf{sk} \in \{0,1\}^p$*, and every message* $\mathsf{m} \in \{0,1\}^n$*, where* $p = \mathsf{poly}(\lambda)$ *and* $n = \mathsf{poly}(\lambda)$*, that given* $\mathsf{vk} \leftarrow \mathsf{KeyGen}(\mathsf{sk})$*,* $\mathsf{sig} \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{m})$ *it holds that* $\mathsf{Verify}(\mathsf{vk}, \mathsf{m}, \mathsf{sig}) = 1$.

Existential Unforgeability under Chosen Message Attacks: *Let $\lambda \in \mathbb{N}$ be a security parameter. We define the advantage of an adversary $\mathcal{A}$ against unforgeability under chosen message attack as follows:*

$$\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{A}}(\lambda) = \Pr\left[\mathsf{Verify}(\mathsf{vk}, \mathsf{m}^*, \mathsf{sig}^*) = 1 \;:\; \begin{array}{c} \mathsf{sk} \leftarrow\!\!\$\, \{0,1\}^p; \\ \mathsf{vk} \leftarrow \mathsf{Setup}(\mathsf{sk}); \\ (\mathsf{sig}^*, \mathsf{m}^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{sk},\cdot)} \end{array}\right]$$

where m^* was not queried to the $\mathsf{Sign}(\mathsf{sk}, \cdot)$ oracle, and the probability is taken over the random coins of Setup and the random coins of $\mathcal{A}$. We say that a signature scheme is unforgeable if for all PPT adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that $\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda)$.

Uniqueness: *We say that a signature scheme is unique if given a verification key vk for all msg, if sig_0 and sig_1 are such that $\mathsf{Verify}(\mathsf{vk}, \mathsf{sig}_0, \mathsf{msg}) = \mathsf{Verify}(\mathsf{vk}, \mathsf{sig}_1, \mathsf{msg}) = 1$, then $\mathsf{sig}_0 = \mathsf{sig}_1$.*

Unique signatures [42] in the standard model can be constructed from the strong RSA assumption [37,19,52] and bilinear maps [51,29].

Argument Systems

Definition 15. *We define an NP-relation to be a polynomial-time binary function $\mathcal{R}$, which takes as input a statement stmt and witness wit. Let $\mathcal{L}_{\mathcal{R}} = \{\mathsf{stmt} : \exists \mathsf{wit}\ \mathcal{R}(\mathsf{stmt}, \mathsf{wit}) = 1\}$ denote the language defined by $\mathcal{R}$.*

Definition 16. *An interactive argument $\Pi_{\mathcal{R}} = (\mathsf{Prove}, \mathsf{Verify})$ for an NP-relation $\mathcal{R}(\mathsf{stmt}, \mathsf{wit})$ is consists of interactive PPT algorithms, a prover Prove and a verifier Verify. The prover is given a statement stmt and a witness wit $s.t\ \mathcal{R}(\mathsf{stmt}, \mathsf{wit}) = 1$ and the verifier is given the statement stmt. where $b \in \{0,1\}$ is the output of the verifier Verify after the protocol completes.*

Perfect Completeness: *We say that an interactive argument system $\Pi_{\mathcal{R}} = (\mathsf{Prove}, \mathsf{Verify})$ for a NP-relation $\mathcal{R}$ is perfectly complete if for all security parameters $\lambda \in \mathbb{N}$ we have*

$$\Pr[\mathsf{Prove}(\mathsf{stmt}, \mathsf{wit}) \rightleftharpoons \mathsf{Verify}(\mathsf{stmt}) = 1 : \mathcal{R}(\mathsf{stmt}, \mathsf{wit}) = 1] = 1.$$

Soundness: *Let $\mathcal{R}$ be a NP-relation defining by the language $\mathcal{L}_{\mathcal{R}} = \{\mathsf{stmt} : \exists_{\mathsf{wit}} \mathcal{R}(\mathsf{stmt}, \mathsf{wit}) = 1\}$. We say that a argument is sound if for all PPT adversaries $\mathcal{A}$ there exists a negligible function $\mathsf{negl}(\lambda)$ such that*

$$\mathsf{Adv}^{\mathsf{Sound}}_{\mathcal{A}}(\lambda) = \Pr[\mathcal{A} \rightleftharpoons \Pi_R.\mathsf{Verify}(\mathsf{stmt}) = 1 : \mathsf{stmt} \notin \mathcal{L}_{\mathcal{R}}] \leq \mathsf{negl}(\lambda),$$

where the probability is taken of the random coins of Verify.

Argument of Knowledge: *Let $\mathcal{R}$ be a NP-relation defining by the language $\mathcal{L}_{\mathcal{R}} = \{\mathsf{stmt} : \exists_{\mathsf{wit}} \mathcal{R}(\mathsf{stmt}, \mathsf{wit}) = 1\}$. We say that a argument is an argument of knowledge if for all PPT adversaries $\mathcal{A}$, all statements stmt and all sufficiently large $\lambda \in \mathbb{N}$, there exists a PPT algorithm Ext such that if*

$$\mathcal{A} \rightleftharpoons \Pi_R.\mathsf{Verify}(\mathsf{stmt}) = 1$$

then

$$\Pr[\mathsf{wit}^* \leftarrow \mathsf{Ext}^{\mathcal{O}}] \leq 1 - \mathsf{negl}(\lambda)$$

where $\mathcal{R}(\mathsf{stmt}, \mathsf{wit}^*) = 1$ *and the oracle* $\mathcal{O}$ *is a communication interface with* $\mathcal{A}$*.*

Witness Indistinguishability: *We say that the protocol for language* $\mathcal{L} \in \mathsf{NP}$ *is witness indistinguishable if all PPT adversaries* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that*

$$\mathsf{Adv}^{\mathsf{WI}}_{\mathcal{A}}(\lambda) = \Pr\left[\begin{array}{c} \hat{b} = b \ : \\ (\mathsf{stmt}^*, \mathsf{wit}_0, \mathsf{wit}_1) \leftarrow \mathcal{A}(\lambda, \mathcal{L}) \\ b \leftarrow\!\!\$\, \{0,1\} \\ \hat{b} \leftarrow \mathcal{A}^{\mathsf{Prove}(\mathsf{stmt}^*, \mathsf{wit}_b)}(\mathsf{stmt}^*, \mathsf{wit}_0, \mathsf{wit}_1) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

where $\mathcal{R}(\mathsf{stmt}^*, \mathsf{wit}_0) = \mathcal{R}(\mathsf{stmt}^*, \mathsf{wit}_1) = 1$ *and the probability is taken over the random coins of* Prove*, and random choice of* b*. Note that it is enough to consider one interaction between the adversary and the prover. Full security then follows from hybrid arguments. We call the proof system statistically witness indistinguishable if the above holds for all adversaries.*

Zero-Knowledge: *We denote the view of a verifier* Verify *interacting with a prover* Prove *as* $\mathsf{View}(\mathsf{Verify}(\mathsf{stmt}) \rightleftharpoons \mathsf{Prove})$*, where* stmt *is a statement. The view includes the verifiers' input* stmt*, coin tosses, and all incoming messages. We say that a proof protocol is zero-knowledge if there exists a PPT simulator* Simul *that for every malicious PPT verifier* Verify *and every PPT distinguisher* Disti

$$\begin{aligned} |\Pr[\mathsf{Disti}(\mathsf{View}(\mathsf{Verify}(\mathsf{stmt}) \rightleftharpoons \mathsf{Prove}(\mathsf{stmt}, \mathsf{wit})))] - \\ \Pr[\mathsf{Disti}(\mathsf{View}(\mathsf{Verify}(\mathsf{stmt}) \rightleftharpoons \mathsf{Simul}))]| \leq \mathsf{negl}(\lambda). \end{aligned}$$

Definition 17. *A interactive argument protocol is called public coin [2] if the verifier sends all his internal coin tosses to the prover*[4]*.*

Definition 18 (Non-Interactive Argument System). *Let* $\mathcal{R}$ *be an* NP*-relation and* $\mathcal{L}_{\mathcal{R}}$ *be the language defined by* $\mathcal{R}$*. A non-interactive argument for* $\mathcal{L}_{\mathcal{R}}$ *consists of algorithms* $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ *with the following syntax.*

$\mathsf{Setup}(\lambda)$**:** *Takes as input a security parameter* λ*, and outputs a common reference string* crs*.*

$\mathsf{Prove}(\mathsf{crs}, \mathsf{stmt}, \mathsf{wit})$**:** *Takes as input a common reference string* crs*, a statement* stmt *and a witness* wit*, and outputs either an argument* arg *or* $\perp$*.*

$\mathsf{Verify}(\mathsf{crs}, \mathsf{stmt}, \mathrm{arg})$**:** *Takes as input the common reference string* crs*, a statement* stmt*, an argument* arg*, and outputs either* 0 *or* 1*.*

We define the following properties.

[4] Public coin protocols are also called Arthur-Merlin protocols.

Perfect Completeness: *It holds for all security parameters* $\lambda \in \mathbb{N}$*, all statements* $\mathsf{stmt} \in \mathcal{L}_{\mathcal{R}}$ *and all witnesses* wit *that if* $\mathcal{R}(\mathsf{stmt}, \mathsf{wit}) = 1$*,* $\mathsf{crs} \leftarrow \mathsf{Setup}(\lambda)$*, and* $\arg \leftarrow \mathsf{Prove}(\mathsf{crs}, \mathsf{stmt}, \mathsf{wit})$*, then* $\mathsf{Verify}(\mathsf{crs}, \mathsf{stmt}, \arg) = 1$*.*

Computational Soundness: *We define the advantage* $\mathsf{Adv}^{\mathsf{Sound}}_{\mathcal{A}}(\lambda)$ *of an adversary* $\mathcal{A}$ *as follows.*

$$\Pr\left[\mathsf{Verify}(\mathsf{crs}, \mathsf{stmt}, \arg) = 1 \;:\; \begin{array}{c} \mathsf{crs} \leftarrow \mathsf{Setup}(\lambda); \\ (\arg, \mathsf{stmt}) \leftarrow \mathcal{A}^{\mathsf{Verify}(\mathsf{crs},\cdot,\cdot)}(\mathsf{crs}); \\ \mathsf{stmt} \notin \mathcal{L}_{\mathcal{R}} \end{array}\right]$$

where the probability is taken over the random coins of Verify*.*
We say that an argument is sound if for all PPT adversaries $\mathcal{A}$ *there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that* $\mathsf{Adv}^{\mathsf{Sound}}_{\mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda)$*.*

Perfect Soundness: *For all adversaries* $\mathcal{A}$ *we have*

$$\Pr\left[\mathsf{Verify}(\mathsf{crs}, \mathsf{stmt}, \arg) = 1 \;:\; \begin{array}{c} (\arg, \mathsf{crs}, \mathsf{stmt}) \leftarrow \mathcal{A}(\mathcal{L}_{\mathcal{R}}); \\ \mathsf{stmt} \notin \mathcal{L}_{\mathcal{R}} \end{array}\right] = 0$$

We will call a perfectly sound system a proof system.

Argument of Knowledge: *Let* $\mathcal{R}$ *be a* NP*-relation defining by the language* $\mathcal{L}_{\mathcal{R}} = \{\mathsf{stmt} : \exists_{\mathsf{wit}} \mathcal{R}(\mathsf{stmt}, \mathsf{wit}) = 1\}$*. Let* Setup^* *be an algorithm that is as* Setup *but outputs additionally a trapdoor* τ*. We say that an argument is an argument of knowledge if for all PPT adversaries* $\mathcal{A}$ *we have*

$$\begin{aligned} &|\Pr[\mathcal{A}(\mathsf{crs}) = 1 : \mathsf{crs} \leftarrow \mathsf{Setup}(\lambda)] - \\ &\Pr[\mathcal{A}(\mathsf{crs}) = 0 : (\mathsf{crs}, \tau) \leftarrow \mathsf{Setup}(\lambda)]| \leq \mathsf{negl}(\lambda) \end{aligned}$$

and there exists a PPT algorithm Ext *and a negligible function* $\mathsf{negl}(.)$ *such that if*

$$(\mathsf{crs}, \tau) \leftarrow \mathsf{Setup}^*(\lambda), \quad \textit{and} \quad (\arg, \mathsf{stmt}) \leftarrow \mathcal{A}(\mathsf{crs}),$$

then

$$\Pr[\mathsf{wit}^* \leftarrow \mathsf{Ext}(\mathsf{crs}, \tau, \arg, \mathsf{stmt})] \leq 1 - \mathsf{negl}(\lambda)$$

where $\mathcal{R}(\mathsf{stmt}, \mathsf{wit}^*) = 1$*.*

Remark 3 (Selective Soundness). One can also refer to a selective secure variant of the soundness definition, where the adversary defines the statement stmt before obtaining the common reference string.

Computational Witness-Indistinguishability: *Let* λ *be a security parameter. We define the advantage of* $\mathcal{A}$ *against witness indistinguishability as follows:*

$$\mathsf{Adv}^{\mathsf{WI}}_{\mathcal{A}}(\lambda) = \left|\Pr\left[b = \hat{b} : \begin{array}{c} \mathsf{crs} \leftarrow \mathsf{Setup}(\lambda); \\ (\mathsf{stmt}, \mathsf{wit}_0, \mathsf{wit}_1) \leftarrow \mathcal{A}(\mathsf{crs}); \\ \mathcal{R}(\mathsf{stmt}, \mathsf{wit}_0) = \mathcal{R}(\mathsf{stmt}, \mathsf{wit}_1) = 1; \\ b \leftarrow\!\!\$\, \{0,1\}; \\ \arg^* \leftarrow \mathsf{Prove}(\mathsf{crs}, \mathsf{stmt}, \mathsf{wit}_b); \\ \hat{b} \leftarrow \mathcal{A}(\arg^*) \end{array}\right] - \frac{1}{2}\right|,$$

where the probability is taken over the random coins of Prove*. We say that the argument system is witness indistinguishable if for all PPT adversaries* $\mathcal{A}$ *there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that* $\mathsf{Adv}^{\mathsf{WI}}_{\mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda)$.

Perfect Witness-Indistinguishability: *We say that a argument system for language* $\mathcal{L}_{\mathcal{R}}$ *is perfectly witness indistinguishable if all adversaries* $\mathcal{A}$ *the following is* 0*:*

$$\left| \Pr \left[b = \hat{b} \ : \begin{array}{c} (\mathsf{crs}, \mathsf{stmt}, \mathsf{wit}_0, \mathsf{wit}_1) \leftarrow \mathcal{A}(\lambda); \\ \mathcal{R}(\mathsf{stmt}, \mathsf{wit}_0) = \mathcal{R}(\mathsf{stmt}, \mathsf{wit}_1) = 1; \\ b \leftarrow\!\!\$ \, \{0,1\}; \\ \arg^* \leftarrow \mathsf{Prove}(\mathsf{crs}, \mathsf{stmt}, \mathsf{wit}_b); \\ \hat{b} \leftarrow \mathcal{A}(\arg^*) \end{array} \right] - \frac{1}{2} \right|,$$

Zero-Knowledge: *Let* λ *be a security parameter. We define the advantage of* $\mathcal{A}$ *against zero-knowledge as follows:*

$$\begin{aligned} \mathsf{Adv}^{\mathsf{ZK}}_{\mathcal{A}}(\lambda) = \Bigg| \Pr \left[\mathcal{A}(\arg^*) = 1 \ : \begin{array}{c} \mathsf{crs} \leftarrow \mathsf{Setup}(\lambda); \\ (\mathsf{stmt}, \mathsf{wit}) \leftarrow \mathcal{A}(\mathsf{crs}); \\ \mathcal{R}(\mathsf{stmt}, \mathsf{wit}) = 1; \\ \arg^* \leftarrow \mathsf{Prove}(\mathsf{crs}, \mathsf{stmt}, \mathsf{wit}_b) \end{array} \right] - \\ \Pr \left[\mathcal{A}(\arg^*) = 1 \ : \begin{array}{c} (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Simul}_1(\lambda); \\ (\mathsf{stmt}, \mathsf{wit}) \leftarrow \mathcal{A}(\mathsf{crs}); \\ \mathcal{R}(\mathsf{stmt}, \mathsf{wit}) = 1; \\ \arg^* \leftarrow \mathsf{Simul}_2(\mathsf{crs}, \mathsf{stmt}, \mathsf{td}) \end{array} \right] \Bigg|, \end{aligned}$$

where the probability is taken over random coins of Prove*. We say that the argument system is zero-knowledge if for all PPT adversaries* $\mathcal{A}$ *there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that* $\mathsf{Adv}^{\mathsf{ZK}}_{\mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda)$.

Remark 4 (Non-interactive zero-knowledge). We will refer to non-interactive zero-knowledge arguments as NIZK. Zero-knowledge proofs can be constructed from bilinear maps [44], the learning with errors assumption [59], obfuscation [62] and (doubly-enhanced) trapdoor permutations[10,23,34,38]. We will also refer to NIZK without a common reference string, however such can only constructed under non-falsifiable assumptions [35].

Definition 19 (Non-Interactive Witness Indistinguishable Proofs). *We call that a non-interactive argument system is a* NIWI *if it is perfectly sound and computationally witness indistinguishable.*

Remark 5. Note that since NIWI are perfectly sound, these proofs systems do not require a setup and a common reference string. Thus we will omit to execute the setup and including the crs in the argument in our protocol specifications.

Definition 20 (Dual-Mode Witness Indistinguishable Proofs). *To define a dual-mode witness indistinguishable proofs* DMWI*, we will redefine the setup algorithm as follows.*

$\mathsf{Setup}(\lambda, \mathsf{mode})$**:** *Takes as input a security parameter* λ*, and and a mode* $\mathsf{mode} \in \{\mathsf{modeSound}, \mathsf{modeWI}\}$*, and outputs a common reference string* crs*.*

We require that for $\mathsf{mode} = \mathsf{modeSound}$ *the system satisfies perfect soundness and for* $\mathsf{mode} = \mathsf{modeWI}$ *the system satisfies perfect witness indistinguishability. Furthermore, we define the following property.*

Mode Indistinguishability: *For all* λ *we define the advantage of* $\mathcal{A}$ *against mode indistinguishability as follows:* $\mathsf{Adv}^{\mathsf{modeIND}}_{\mathcal{A}}(\lambda) =$

$$\left| \Pr\left[\mathsf{mode} = \mathsf{mode}^* : \begin{array}{c} \mathsf{mode} \leftarrow\!\!\$\, \{\mathsf{modeSound}, \mathsf{modeWI}\}; \\ (\mathsf{crs}) \leftarrow \mathsf{Setup}(\lambda, \mathsf{mode}); \\ \mathsf{mode}^* \leftarrow A(\lambda, \mathsf{crs}) \end{array} \right] - \frac{1}{2} \right|,$$

where the probability is taken over the random choice of mode *and the random coins of* Setup*. We say that the proof system is mode indistinguishable if for all PPT adversaries* $\mathcal{A}$ *there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that* $\mathsf{Adv}^{\mathsf{modeIND}}_{\mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda)$*.*

Definition 21. *We call an argument system private-coin ZAP if witness indistinguishability holds against a malicious setup. That is in the definition of witness indistinguishability, the adversary runs as follows* $(\mathsf{crs}, \mathsf{stmt}, \mathsf{wit}_0, \mathsf{wit}_1) \leftarrow \mathcal{A}(\lambda)$*.*

Non-interactive witness-indistinguishable proofs can be constructed from NIZK proofs and derandomization assumptions [31,4], from bilinear pairings [43] and indistinguishablity obfuscation [9].

(Publicly Deniable) Public Key Encryption We recall the standard definition of public key encryption.

Definition 22. *A public key encryption scheme* $\mathcal{E}$ *consists of algorithms* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *with the following syntax.*

$\mathsf{KeyGen}(\lambda)$: *On input a security parameter* λ*, outputs keypair* $(\mathsf{sk}, \mathsf{pk})$*.*
$\mathsf{Enc}(\mathsf{pk}, \mathsf{m})$: *On input a public key* pk *and message* m*, outputs ciphertext* ct*.*
$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$: *On input a secret key* sk *and ciphertext* ct*, outputs a message* m *or* $\perp$ *if decryption fails.*

We define the following properties.

Correctness: *We say that a encryption scheme* $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is correct if for all all security parameters* $\lambda \in \mathbb{N}$*, all key pairs* $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(\lambda)$

$$\Pr[\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, \mathsf{m})) \neq \mathsf{m}] = 0,$$

where the probability is taken over the random coins of the Enc *algorithm, as well as the random choice of* m*.*

Chosen Plaintext Attack Security (IND-CPA): *We define the advantage of an adversary $\mathcal{A}$ against IND-CPA as follows.*

$$\mathsf{Adv}^{\mathsf{cpa}}_{\mathcal{A}}(\lambda) = \left| \Pr \left[\hat{b} = b \ : \begin{array}{c} (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(\lambda); \\ (\mathsf{m}_0, \mathsf{m}_1) \leftarrow \mathcal{A}(\mathsf{pk}); \\ b \leftarrow\!\$ \{0,1\}; \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{m}_b); \\ \hat{b} \leftarrow \mathcal{A}(\mathsf{ct}) \end{array} \right] - \frac{1}{2} \right|,$$

where the probability is taken over the random coins of Enc *and random choice of b. We say that a public key encryption is CPA secure if for all PPT adversaries $\mathcal{A}$ there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that* $\mathsf{Adv}^{\mathsf{cpa}}_{\mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda)$.

Non-Adaptive Chosen Ciphertext Attack Security (IND-CCA1): *We define the advantage of an adversary $\mathcal{A}$ against the IND-CCA1 as*

$$\mathsf{Adv}^{\mathsf{cca1}}_{\mathcal{A}}(\lambda) = \left| \Pr \left[\hat{b} = b \ : \begin{array}{c} (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(\lambda); \\ (\mathsf{m}_0, \mathsf{m}_1) \leftarrow \mathcal{A}^{\mathsf{Dec}(\mathsf{sk},\cdot)}(\mathsf{pk}); \\ b \leftarrow\!\$ \{0,1\}; \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{m}_b); \\ \hat{b} \leftarrow \mathcal{A}(\mathsf{ct}) \end{array} \right] - \frac{1}{2} \right|,$$

where the probability is taken over the random coins of Enc *and random choice of b. We say that a public key encryption is CCA1 secure if for all PPT adversaries $\mathcal{A}$ there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that* $\mathsf{Adv}^{\mathsf{cca1}}_{\mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda)$.

Deniability (Indistinguishability of Explanation [62]): *Encryption scheme $\mathcal{E}$ is a publicly deniable encryption if there exists an additional algorithm* $\mathsf{Expl}(\mathsf{pk}, \mathsf{ct}, \mathsf{m}; r)$ *that outputs a string e and the indistinguishability of explanation defined below holds. We define the advantage of an adversary $\mathcal{A}$ against deniablility as follows:*

$$\mathsf{Adv}^{\mathsf{IND-EX}}_{\mathcal{A}}(\lambda) = \left| \Pr \left[\hat{b} = b \ : \begin{array}{c} (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(\lambda); \\ (\mathsf{m}) \leftarrow \mathcal{A}(\mathsf{pk}); \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{m}; u_0); \\ u_1 \leftarrow \mathsf{Expl}(\mathsf{pk}, \mathsf{ct}, \mathsf{m}; r); \\ b \leftarrow\!\$ \{0,1\}; \\ \hat{b} \leftarrow \mathcal{A}(\mathsf{ct}, u_b) \end{array} \right] - \frac{1}{2} \right|,$$

where the probability is taken over the random coins of Enc*, and random choice of b, u_0, r. We say that a public key encryption is deniable if for all PPT adversaries $\mathcal{A}$ there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that* $\mathsf{Adv}^{\mathsf{IND-EX}}_{\mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda)$.

Witness Encryption We recall the standard definition of witness encryption introduced by Garg et al. [36].

Definition 23. *A witness encryption scheme* WE *consists of algorithms* (Enc, Dec) *with the following syntax.*

$\mathsf{Enc}(\lambda, \mathsf{stmt}, \mathsf{m})$: *On input a security parameter* λ*, a statement* stmt *and message* m*, outputs ciphertext* ct.

$\mathsf{Dec}(\mathsf{stmt}, \mathsf{wit}, \mathsf{ct})$: *On input a statement* stmt*, witness* wit *and ciphertext* ct*, outputs a message* m *or* $\perp$ *if decryption fails.*

We define the following properties.

Correctness: *We say that a witness encryption scheme* $\mathsf{WE} = (\mathsf{Enc}, \mathsf{Dec})$ *for language* $\mathcal{L}$ *is correct if for all security parameters* $\lambda \in \mathbb{N}$*, statements* $\mathsf{stmt} \in \mathcal{L}$*, witnesses* wit*, s.t.* $\mathcal{R}(\mathsf{stmt}, \mathsf{wit}) = 1$ *and messages* m*,*

$$\Pr[\mathsf{Dec}(\mathsf{stmt}, \mathsf{wit}, \mathsf{Enc}(\lambda, \mathsf{stmt}, \mathsf{m})) \neq \mathsf{m}] \approx 0,$$

where the probability is taken over the random coins of the Enc *algorithm, and the random choice of* m*. We say that the witness encryption scheme is perfectly correct if the above probability is zero.*

Security: *We define the advantage of* $\mathcal{A}$ *against the security of the witness encryption scheme as* $\mathsf{Adv}^{\mathsf{WE}}_{\mathcal{A}}(\lambda) =$

$$\left| \Pr \left[\mathsf{stmt} \notin \mathcal{L} \ \wedge \ \hat{b} = b \ : \ \begin{array}{c} (\mathsf{stmt}, \mathsf{m}_0, \mathsf{m}_1) \leftarrow \mathcal{A}(\lambda, \mathcal{L}); \\ b \leftarrow\!\!\$ \, \{0,1\}; \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\lambda, \mathsf{stmt}, \mathsf{m}_b); \\ \hat{b} \leftarrow \mathcal{A}(\mathsf{ct}) \end{array} \right] - \frac{1}{2} \right|,$$

where the probability is taken over the random coins of Enc *and random choice of* b*. We say that a witness encryption is secure if for all PPT adversaries* $\mathcal{A}$ *there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that* $\mathsf{Adv}^{\mathsf{WE}}_{\mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda)$.

Extractable Security: *We say that a witness encryption for language* $\mathcal{L}$ *defined by a* **NP***-relation* $\mathcal{R}$ *is extractably secure if for all PPT adversaries* $\mathcal{A}$*, there exists a PPT extractor* Ext *and polynomials* p *and* q*, such that for all auxiliary inputs* aux *and for all* $\mathsf{stmt} \in \{0,1\}^*$*, the following holds:*

$$\Pr \left[\begin{array}{l} \mathsf{m} \leftarrow_R \{0,1\}, \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\lambda, \mathsf{stmt}, \mathsf{m}) \end{array} : \mathcal{A}(\mathsf{stmt}, \mathsf{ct}, \mathsf{aux}) = \mathsf{m} \right] \geq 1/2 + 1/q(|\mathsf{stmt}|)$$
$$\implies \Pr[\mathsf{Ext}(\mathsf{stmt}, \mathsf{aux}) = \mathsf{wit} : (\mathsf{stmt}, \mathsf{wit}) \in \mathcal{R}] \geq 1/p(|x|)$$

Indistinguishability Obfuscation We recall the definition of indistinguishability obfuscation first introduced by Barak et al. [3].

Definition 24. *An indistinguishability obfuscation algorithm for the class of circuits* $\{C_\lambda\}$ *is a PPT algorithm* Obf *with the following syntax.*

$\mathsf{Obf}(\lambda, C)$**:** *Takes as input a security parameter* λ *and circuit* C*, and outputs an obfuscated program* O.

Furthermore, we define the following properties.

Correctness: *For all security parameters $\lambda \in \mathbb{N}$, for all $C \in C_\lambda$, and for all inputs x, there exists a negligible function* $\mathsf{negl}(.)$ *such that*

$$\Pr[O(x) = C(x) : O \leftarrow \mathsf{Obf}(\lambda, C)] = 1 - \mathsf{negl}(\lambda)$$

Indistinguishability: *We say that an obfuscation scheme is indistinguishable if all PPT adversaries $\mathcal{A}$, there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that*

$$\mathsf{Adv}^{\mathsf{iO}}_{\mathcal{A}}(\lambda) = \Pr\left[b = \hat{b} : \begin{array}{c}(C_0, C_1) \leftarrow \mathcal{A}(\lambda);\\ b \leftarrow\!\$\, \{0,1\};\\ O \leftarrow \mathsf{Obf}(\lambda, C_b); \hat{b} \leftarrow \mathcal{A}(O)\end{array}\right] \leq \mathsf{negl}(\lambda)$$

where for all inputs x, $C_0(x) = C_1(x)$ and the probability is taken over the random coins of Obf.

Punctured Pseudorandom Functions Finally, we will need to rely on puncturable pseudorandom functions [13,14,49].

Definition 25. *A puncturable PRF consists of algorithms* $\mathsf{PRF} = (\mathsf{Setup}, \mathsf{Eval}, \mathsf{Puncture})$ *with the following syntax.*

$\mathsf{Setup}(\lambda)$**:** *Takes as input a security parameter λ and outputs a key K. The algorithm also sets the parameters $n = \mathsf{poly}(\lambda)$ and $d = \mathsf{poly}(\lambda)$.*

$\mathsf{Puncture}(K, x)$**:** *Takes as input a key K and a string $x \in \{0,1\}^n$, and outputs a punctured key K_x.*

$\mathsf{Eval}(K, x)$**:** *On input a key K and a string $x \in \{0,1\}^n$, outputs $y \in \{0,1\}^d$.*

We define the following properties.

Functionality preserved under puncturing: *For every PPT adversary $\mathcal{A}$ such that $\mathcal{A}$ outputs $x^* \in \{0,1\}^n$, for all $x \in \{0,1\}^n$, where $x \neq x^*$, we have that*

$$\Pr[\mathsf{Eval}(K, x) = \mathsf{Eval}(K_{x^*}, x)) : K \leftarrow \mathsf{Setup}(\lambda), K_{x^*} \leftarrow \mathsf{Puncture}(K, x^*)] = 1$$

Pseudorandom at punctured points: *We define the advantage of an adversary $\mathcal{A}$ against pseudorandomness at punctured points as follows.*

$$\mathsf{Adv}^{\mathsf{PRand}}_{\mathcal{A}}(\lambda) = \Pr\left[b = \hat{b} \ : \begin{array}{c}x^* \leftarrow \mathcal{A}(\lambda); K \leftarrow \mathsf{Setup}(\lambda),\\ K_{x^*} \leftarrow \mathsf{Puncture}(K, x^*)\\ r_0 \leftarrow \mathsf{Eval}(K, x^*); r_1 \leftarrow\!\$\, \{0,1\}^n\\ b \leftarrow\!\$\, \{0,1\}; \hat{b} \leftarrow \mathcal{A}(\sigma, K_{x^*}, r_b)\end{array}\right],$$

where the probability is taken over random coins of Setup *and random choice of r_1. We say that a punctured function is pseudorandom at punctured points if for every PPT adversary $\mathcal{A}$ there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that* $\mathsf{Adv}^{\mathsf{PRand}}_{\mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda)$.

B Full Proofs

Theorem 1. *If* $(\mathsf{Prove}, \mathsf{Verify}, \mathsf{Expl})$ *is a malicious verifier (statistical) explainable argument system then it is also (statistical) witness indistinguishable.*

Proof. An adversary $\mathcal{A}$ against witness indistinguishability is given oracle access to a prover that uses the witness wit_b, where $\mathcal{A}$'s goal is to output b. The idea behind the proof is for each oracle query of $\mathcal{A}$ to first run the prover according to the protocol to get a transcript trans, then rewind the adversary, compute $\mathsf{coins}_{\mathcal{R}} \leftarrow \mathsf{Expl}(\mathsf{stmt}, \mathsf{wit}_{\hat{b}}, \mathsf{trans})$ for some random bit $\hat{b}$ and run $\mathsf{Prove}(\mathsf{stmt}, \mathsf{wit}_{\hat{b}}; \mathsf{coins}_{\mathcal{R}})$ in interaction with the adversary. Note that due to malicious verifier explainability $\mathcal{A}$ will only notice this change with negligible probability (in such a case we could use $\mathcal{A}$ to break this property). What is more important, the adversary's oracle queries now only consists of interactions with a prover that uses $\mathsf{wit}_{\hat{b}}$ instead of wit_b. Thus, the only way for $\mathcal{A}$ to win the WI experiment is by guessing the bit b.

Theorem 2. *If there exists a malicious setup explainable non-interactive argument, then there exists a two-move witness-indistinguishable argument, where the verifier's message is reusable. In other words, given a malicious setup explainable non-interactive argument, we can build a private-coin ZAP.*

Proof. Let $\mathsf{ExArg}_{\mathcal{L}} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ be an malicious setup explainable non-interactive argument for $\mathcal{L}$. The ZAP verifier runs the setup algorithm $\mathsf{crs} \leftarrow \mathsf{Setup}(\lambda)$ and sends the common reference string crs to the prover. When a prover is given a statement stmt and witness wit, runs $\mathrm{arg} \leftarrow \mathsf{Prove}(\mathsf{crs}, \mathsf{stmt}, \mathsf{wit})$, sends arg to the ZAP verifier which accepts the argument if $\mathsf{Verify}(\mathsf{crs}, \mathsf{stmt}, \mathrm{arg}) = 1$.

Soundness immediately follows from the soundness of the non-interactive argument. To show witness-indistinguishability, let stmt be the statement and $\mathsf{wit}_0, \mathsf{wit}_1$ the witnesses chosen by the adversary $\mathcal{A}$. The challenger responds with $\mathrm{arg} \leftarrow \mathsf{Prove}(\mathsf{crs}, \mathsf{stmt}, \mathsf{wit}_b)$ for some bit b. Note that the argument for all witnesses in $\mathcal{L}$ can be explained as $\mathsf{coins}_0 \leftarrow \mathsf{Expl}(\mathsf{crs}, \mathsf{wit}_0, \mathrm{arg})$ and $\mathsf{coins}_1 \leftarrow \mathsf{Expl}(\mathsf{crs}, \mathsf{wit}_1, \mathrm{arg})$, such that

$$\begin{aligned} \mathrm{arg} &= \mathsf{Prove}(\mathsf{crs}, \mathsf{stmt}, \mathsf{wit}_0; \mathsf{coins}_0) \\ &= \mathsf{Prove}(\mathsf{crs}, \mathsf{stmt}, \mathsf{wit}_1; \mathsf{coins}_1). \end{aligned}$$

Thus any adversary that has non-negligible advantage against witness indistinguishability.

To show the implication to witness encryption, let us recall a generalized version of the Goldreich-Levin theorem [39], from [18].

Lemma 1 (Generalized Goldreich-Levin Theorem). *If for any* $\mathcal{A}$ *and any* $(\alpha, \beta) \in \{0,1\}^k \times \{0,1\}^\ell$ *such that* $p(\alpha) = \Pr[\mathcal{A}(\alpha, r) = \langle \beta, r \rangle : r \leftarrow\$ \{0,1\}^\ell]$*, then then there exists a PPT inverter* $\mathcal{A}'$ *and a non-zero polynomial* $q(.)$ *such that* $\Pr[\mathcal{A}'^{\mathcal{A}(\alpha,.)}(1^\ell, \alpha) = \beta] \geq q(p(\alpha) - 1/2)$.

Theorem 3. *If* ExArg *is a fully explainable argument, then* ExArg *is a malicious setup and malicious prover explainable argument.*

Proof. The proof follows immediately from the definitions.

Theorem 4. *Given that one-way functions and malicious prover selectively sound non-interactive (resp. two-move) arguments for* NP *exist, then there exists a witness encryption scheme for* NP*.*

In the proof below we use terminology from non-interactive arguments but we can rewrite the same proof for the case of two-move arguments, where we use the verifier's first message instead of the CRS and the provers response as the argument.

Proof. First note, that from malicious prover explainability of the argument, we have that for all coins $\mathsf{coins}_{\mathcal{A}}$ and $\mathsf{arg} \leftarrow \mathsf{Prove}(\mathsf{crs}, \mathsf{stmt}, \mathsf{wit}; \mathsf{coins}_{\mathcal{A}})$ an honest prover can output coins coins_P, which explains an argument. Assuming malicious prover explainability/uniqueness, Expl must return the same coins as the adversary, and the prove algorithm must return the same argument under the same coins but with a different witness. In other words for all $\mathsf{coins}_{\mathcal{A}}$ it must be that $\mathsf{coins}_P = \mathsf{coins}_{\mathcal{A}}$. Therefore, we have that the distribution of arguments for a given statement is determined only by the random coins. Thus if we set the coins, then there exists only one argument for a statement.

Having established that, we construct a witness encryption scheme as follows. Let stmt be the statement refined by a relation R for the witness encryption, and PRG be a pseudorandom number generator. The encrypter generates a random seed $z \leftarrow \mathsf{SeedGen}(\lambda)$ and computes $y \leftarrow \mathsf{PRG}(z, 0)$. Then, the encrypter constructs a statement $\mathsf{stmt}' = \{\exists_{\mathsf{wit}}\ R(\mathsf{stmt}, \mathsf{wit}) = 1 \vee y = \mathsf{PRG}(\mathsf{wit})\}$, for a given y. Finally, to encrypt a message $\mathsf{msg} \in \{0, 1\}$, the encrypter first generates the common reference string crs. If the arguments system is selectively secure, the encrypter generates the crs for stmt'. Then the encrypter computes $\mathsf{arg} \leftarrow \mathsf{Prove}(\mathsf{crs}, \mathsf{stmt}', z; \mathsf{coins})$, and computes $c = \mathsf{msg} \oplus H(\mathrm{arg}, r)$, where H is a Goldreich-Levin hardcore bit and r is uniformly random. The ciphertext is $\mathsf{ct} = (\mathsf{crs}, c, \mathsf{coins}, y, r)$. A decrypter with a witness wit^* such that $R(\mathsf{stmt}, \mathsf{wit}^*) = 1$, computes $\mathrm{arg}^* \leftarrow \mathsf{Prove}(\mathsf{crs}, \mathsf{stmt}', \mathsf{wit}^*; \mathsf{coins})$ and $\mathsf{msg}^* \leftarrow c \oplus H(\mathrm{arg}^*, r)$.

For correctness, note that from malicious prover explainability we are guaranteed that $\mathrm{arg}^* = \mathrm{arg}$, hence $\mathsf{msg}^* = \mathsf{msg}$.

To argue the security of the witness encryption, we use the distinguisher $\mathcal{D}$ against the encryption scheme to construct an adversary against the selective soundness of the argument system. First, we show how to use $\mathcal{D}$ to build a predictor $\mathcal{A}'$ for the generalized Goldreich-Levin theorem. Let $\mathcal{D}$ distinguish the message bit msg, with non-negligible probability for $\mathsf{stmt} \notin \mathcal{L}$. We set $\alpha = (\mathsf{crs}, \mathsf{coins}, y)$ and $\beta = \mathrm{arg}$ from the generalized Goldreich-Levin theorem 1. So we have that $\mathcal{D}$ on input crs, y, r and some random coins can distinguish $H(\mathrm{arg}, r)$ from random with non-negligible probability. By lemma 1 we can construct $\mathcal{A}'$ who on input α outputs $\beta = \mathrm{arg}$ with non-negligible probability. Moreover, we have that $\mathrm{arg} = \mathsf{Prove}(\mathsf{crs}, \mathsf{stmt}', z; \mathsf{coins})$ because otherwise, as argued

earlier, the argument system wouldn't be malicious prover explainable. Now, we will choose y uniformly at random. If an adversary $\mathcal{A}'$ notices this change with non-negligible probability, then we can use $\mathcal{A}'$ as a distinguisher for the pseudorandomness property of the PRG. Finally, we have that with overwhelming probability stmt is not in the language, and arg is a counterexample for selective soundness of the argument system.

Theorem 5. *Let* ExArg° *be the system given by Scheme 1. The system* ExArg° *is computationally sound (in the selective setting) assuming indistinguishability obfuscation of* Obf*, pseudorandomness in punctured points of* PRF*, mode indistinguishability of the* DMWI *scheme, and unforgeability of the signature scheme.*

Proof. We will prove the soundness of the non-interactive arguments system by changing the obfuscated programs according to the hybrid experiments below. Then we argue that any adversary that produces an argument for a statement stmt^* can be used to break $\mathsf{EUF\text{-}CMA}$ of the underlying signature scheme. For clarity, we will give detailed proofs of indistinguishability between the hybrid experiments at the end of the proof.

$\mathcal{H}_0$**:** This is the original scheme.

$\mathcal{H}_1$**:** We change the parameters $\mathsf{crs}_{\mathsf{DMWI}}$ to $\mathsf{crs}_{\mathsf{DMWI}} \leftarrow \mathsf{DMWI.Setup}(\lambda, \mathsf{modeWI}; \mathsf{coins}_S)$.

Claim (Informal). Indistinguishability between $\mathcal{H}_0$ and $\mathcal{H}_1$ follows from mode indistinguishability of DMWI.

$\mathcal{H}_2$**:** We change the witness $\mathsf{wit}_{\mathsf{Setup}}$ to (modeWI).

Claim (Informal). Indistinguishability between $\mathcal{H}_1$ and $\mathcal{H}_2$ follows from witness indistinguishability of the NIWI scheme.

$\mathcal{H}_3$**:** We use $\mathsf{ProgProve}^2_{\nabla}$ instead of $\mathsf{ProgProve}^1_{\nabla}$.

Claim (Informal). Indistinguishability between $\mathcal{H}_2$ and $\mathcal{H}_3$ follows from indistinguishability of Obf.

$\mathcal{H}_4$**:** We use $\mathsf{ProgVerify}^*$ instead of $\mathsf{ProgVerify}$.

Claim (Informal). Indistinguishability between $\mathcal{H}_3$ and $\mathcal{H}_4$ follows from indistinguishability of Obf.

$\mathcal{H}_5$**:** We switch the witness $\mathsf{wit}_{\mathsf{Setup}}$ to $(2, K, \mathsf{coins}_P)$.

Claim (Informal). Indistinguishability between $\mathcal{H}_4$ and $\mathcal{H}_5$ follows from witness indistinguishability of the NIWI scheme.

$\mathcal{H}_6$**:** We change the parameters $\mathsf{crs}_{\mathsf{DMWI}}$ to binding mode, i.e. we reverse the changes made in $\mathcal{H}_2$.

Claim (Informal). Indistinguishability between $\mathcal{H}_5$ and $\mathcal{H}_6$ follows from mode indistinguishability of DMWI.

$\mathcal{H}_7$**:** We choose the key sk_s^* at random.

Claim (Informal). Indistinguishability between $\mathcal{H}_6$ and $\mathcal{H}_7$ follows from pseudorandomness at punctured points.

Finally, we set vk_s^* from the $\mathsf{EUF\text{-}CMA}$ challenger. Now, assume an adversary outputs an argument arg^* for stmt^* which verifies correctly. Let us remind that the program $\mathsf{ProgVerify}^*$ outputs vk_s^*. Given that the argument verifies correctly we have that $\mathsf{Sig.Verify}(\mathsf{vk}_s^*, \mathsf{arg}^*, \mathsf{stmt}^*) = 1$. We return $(\mathsf{arg}^*, \mathsf{stmt}^*)$ as a forge in the existential unforgeability under chosen message attack experiment.

Claim. If there exists an adversary $\mathcal{A}$, which distinguishes between the hybrids $\mathcal{H}_0$ and $\mathcal{H}_1$, then there exists a reduction $\mathcal{S}$ which breaks mode indistinguishability of the DMWI scheme.

Proof. Both hybrids differ only in how the reference string $\mathsf{crs}_{\mathsf{DMWI}}$ is computed. The reduction $\mathcal{S}$ will obtain the $\mathsf{crs}_{\mathsf{DMWI}}$ from the mode indistinguishability challenger. If $\mathsf{crs}_{\mathsf{DMWI}}$ is a hiding string, then the simulation by $\mathcal{S}$ is as in $\mathcal{H}_1$, and otherwise, if the string is binding, then as in $\mathcal{H}_0$. Thus, if $\mathcal{A}$'s advantage is non-negligible, so is the reduction advantage.

Claim. If there exists an adversary $\mathcal{A}$, which distinguishes between the hybrids $\mathcal{H}_1$ and $\mathcal{H}_2$, then there exists a reduction $\mathcal{S}$ which breaks witness indistinguishability of NIWI.

Proof. Both hybrids differ only in which witness the setup algorithm is using. The reduction $\mathcal{S}$, will then set the witness wit_0 as in $\mathcal{H}_1$ and wit_1 as in $\mathcal{H}_2$, sent both witnesses to the witness indistinguishability challenger, and obtain a proof π. $\mathcal{S}$ will use π in the crs.

If $\mathcal{A}$ has any noticeable advantage in distinguishing between $\mathcal{H}_1$ and $\mathcal{H}_2$, then $\mathcal{S}$ has a non-negligible advantage in distinguishing the witness used to compute π.

Claim. If there exists an adversary $\mathcal{A}$, which distinguishes between the hybrids $\mathcal{H}_2$ and $\mathcal{H}_3$, then there exists a reduction $\mathcal{S}$ which breaks indistinguishability obfuscation.

Proof. The difference is that the PRF key is punctured on stmt^*, but the PRF evaluation $\mathsf{PRF.Eval}(K, \mathsf{stmt}^*)$ will never be called, thus the input-output behavior of both programs will be the same. The reduction $\mathcal{S}$ will set the program to be obfuscated to $\mathsf{ProgProve}^1_\nabla$ for $\mathcal{H}_0$, and for $\mathcal{H}_1$ it sets the program to be obfuscated to $\mathsf{ProgProve}^2_\nabla$. Now any adversary $\mathcal{A}$ that can distinguish between the hybrids leads to $\mathcal{S}$ successfully answering the indistinguishability challenge.

Claim. If there exists an adversary $\mathcal{A}$, which distinguishes between the hybrids $\mathcal{H}_3$ and $\mathcal{H}_4$, then there exists a reduction $\mathcal{S}$ which breaks indistinguishability obfuscation.

Proof. The difference here is that for stmt^* the signature verification key vk_s^* is pre-computed and hardwired into the program. As earlier, the input-output behavior of both programs is the same. Thus, we can construct a reduction $\mathcal{S}$, which queries the indistinguishability challenger on both programs, and forwards the challenge to the adversary in the soundness experiment.

Claim. If there exists an adversary $\mathcal{A}$, which distinguishes between the hybrids $\mathcal{H}_4$ and $\mathcal{H}_5$, then there exists a reduction $\mathcal{S}$ which breaks witness indistinguishability of the NIWI scheme.

Proof. The proof follows the reasoning as in the proof of for $\mathcal{H}_2$. The hybrids $\mathcal{H}_4$ and $\mathcal{H}_5$ differ only in which witness the setup algorithm is using.

Claim. If there exists an adversary $\mathcal{A}$, which distinguishes between the hybrids $\mathcal{H}_5$ and $\mathcal{H}_6$, then there exists a reduction $\mathcal{S}$ which breaks mode indistinguishability of the DMWI scheme.

Proof. The reasoning is analogical to the reasoning of the proof for $\mathcal{H}_1$ The difference is that we switch $\mathsf{crs}_{\mathsf{DMWI}}$ from a hiding string back to a binding string.

Claim. If there exists an adversary $\mathcal{A}$, which distinguishes between the hybrids $\mathcal{H}_6$ and $\mathcal{H}_7$, then there exists a reduction $\mathcal{S}$ which breaks pseudorandomness at punctured points of the PRF. What follows is that the corresponding verification key vk_s^* is chosen uniformly at random from the key space.

Proof. We will construct a reduction $\mathcal{S}$, which gives stmt^* to the challenger in the punctured points experiment for the punctured PRF. The challenger returns K_{stmt^*} and the challenge r. Now, $\mathcal{S}$ sets $\mathsf{sk}^* = r$, generates vk_s based on sk^* and proceeds by constructing the proving and verification programs as in $\mathcal{H}_6$. Now, if the challenge r given to $\mathcal{S}$ is $\mathsf{PRF.Eval}(K_{\mathsf{stmt}^*}, \mathsf{stmt})$ then $\mathcal{S}$ simulates $\mathcal{H}_6$ and otherwise if r is random then $\mathcal{H}_7$. Thus, if $\mathcal{A}$ has a non-negligible advantage to distinguish between $\mathcal{H}_6$ and $\mathcal{H}_7$, then $\mathcal{S}$ has the same advantage in the punctured points experiment.

Theorem 6. *Given that the signature scheme* Sig *is unique,* NIWI *is perfectly sound,* DMWI *is a dual-mode proof, and all primitives are perfectly correct, the argument system* ExArg° *is malicious setup explainable.*

Proof. Note that a verification key vk_s can be computed using the program O_{Verify} from only the statement stmt. Let arg be the argument input to the Expl algorithm and assume $\mathsf{Verify}(\mathsf{crs}, \mathsf{stmt}, \mathrm{arg}) = 1$. From the verification algorithm, we have that vk_s is a unique signature scheme, and arg is a valid signature on stmt with respect to vk_s. There exists only one signature arg for stmt, which verifies with the verification key vk_s. Thus the bits of arg depend only on the statement signed, as otherwise, it would contradict uniqueness of the signature scheme.

Now we have to argue that for every wit^*, where $\mathcal{R}(\mathsf{stmt}, \mathsf{wit}^*) = 1$, the Prove algorithm will return arg. From perfect soundness of the NIWI we have that there are three cases:

- The program O_{Prove} is $\mathsf{ProgProve}^1[\mathsf{pp}, \mathsf{crs}_{\mathsf{DMWI}}, K]$ for some K. In other words, O_{Prove} is a correct obfuscation of $\mathsf{ProgProve}^1$, the signature arg is determined only by the message stmt and the secret key sk. The secret key is, in turn, determined by the statement and key K.
- The program O_{Prove} is $\mathsf{ProgProve}^2$. The argument is analogical to the previous, as the only difference is that the key K is punctured on stmt^*.
- The common reference string $\mathsf{crs}_{\mathsf{DMWI}}$ is hiding. In this case, from perfect hiding we have that for all wit^* and $\pi \leftarrow \mathsf{DMWI.Prove}(\mathsf{crs}_{\mathsf{DMWI}}, \mathsf{stmt}, \mathsf{wit}^*)$, the program O_{Prove} must have the same output. Thus either O_{Prove} returns a correct arg, or an incorrect one for all wit^*. In case the program O_{Prove} would abort or return an incorrect argument, it would have abort on arg as well.

Theorem 7. *Let* $\mathsf{ExArg}^\dagger$ *be the system given by Scheme 1. The system* $\mathsf{ExArg}^\dagger$ *is computationally sound (in the selective setting), assuming indistinguishability obfuscation of* Obf*, pseudorandomness in punctured points of* PRF*, zero-knowledge of the* NIZK *scheme and unforgeability of the signature scheme.*

Proof (Sketch). The proof follows the proof of Theorem 5 with the exception that instead of hybrids $\mathcal{H}_1$, $\mathcal{H}_2$, $\mathcal{H}_5$ and $\mathcal{H}_6$, in $\mathcal{H}_1$ we run the simulator of the NIZK scheme. To preserve the numbering of hybrids, we set $\mathcal{H}_2$ to be equal to $\mathcal{H}_1$, $\mathcal{H}_5$ to be equal to $\mathcal{H}_4$, and $\mathcal{H}_6$ to be equal to $\mathcal{H}_5$.

Theorem 8. *Given that the signature scheme* Sig *is unique,* NIZK *is sound, and all primitives are perfectly correct, the argument system* $\mathsf{ExArg}^\dagger$ *is fully explainable.*

Proof. The proof follows mostly the reasoning of the proof of Theorem 6. Here, however, we need to consider only the case where O_{Prove} is correct. Similarly, as above, if the program O_{Verify} is correct, the verification key vk_s is determined by the statement. Let arg such that $\mathsf{Verify}(\mathsf{crs}, \mathsf{stmt}, \mathrm{arg}) = 1$ be the argument and $\mathsf{coins}_{\mathcal{A}}$ be the coins returned by the adversary. From the verification we know that $\mathsf{Sig.Verify}(\mathsf{vk}_s, \mathrm{arg}, \mathsf{stmt}) = 1$. From the uniqueness of the signature scheme, we know that there exists only one arg that verifies with stmt under vk_s.

The Expl algorithm given a witness wit^* will return $\gamma = 0$. Now we have to argue that $\mathsf{Prove}(\mathsf{crs}, \mathsf{stmt}, \mathsf{wit}^*; \gamma)$ will output arg. From perfect soundness of the NIZK we have that the obfuscated program O_{Prove} is correctly implementing $\mathsf{ProgProve}^1[\mathsf{pp}, \mathsf{crs}_{\mathsf{DMWI}}, K]$ for some K. In this case, O_{Prove} is a correct obfuscation of $\mathsf{ProgProve}^1[\mathsf{pp}, \mathsf{crs}_{\mathsf{DMWI}}, K]$, thus the program must return a signature on stmt that verifies with vk_s, as otherwise arg would not be a correct argument as well.

Theorem 9. *Let* ExArg^∇ *be the system given by Scheme 1 for* $\nabla = \circ$ *or* $\nabla = \dagger$*.* ExArg^∇ *is zero-knowledge in the common reference string model.*

Proof. To show the zero-knowledge property, we need to build a simulator $\mathcal{S}$, that outputs a common reference string crs and then on input a statement stmt outputs arg such that $\mathsf{Verify}(\mathsf{crs}, \mathsf{stmt}, \mathrm{arg}) = 1$. The simulator $\mathcal{S}$ will generate the crs as specified by the ExArg^∇ scheme, but keeps the secret key of the

PRF in his memory. Then given a statement stmt it $\mathcal{S}$ generates the secret key $\mathsf{sk} \leftarrow \mathsf{PRF.Eval}(K, \mathsf{stmt})$. Finally, $\mathcal{S}$ signs stmt using sk and outputs the signature together with the message.

Theorem 10 (Security and Extractability). *Scheme 2 is a (extractably) secure witness encryption if* WE *is a (extractably) secure witness encryption scheme, and* NIZK *is zero-knowledge (in the common reference string or RO model).*

Proof (Sketch). The proof is pretty straightforward, as the NIZK can be simulated. The rest of the proof follows from security or extractable security of the WE scheme.

Theorem 11 (Robustness and Plaintext Awareness). *Scheme 2 is robust if the witness encryption scheme* WE *is perfectly correct, and the* NIZK *proof system is perfectly sound (in the common reference string or RO model). If the* NIZK *proof system is a proof of knowledge (in the common reference string or RO model), then Scheme 2 is plaintext aware.*

Proof (Sketch). Let us first address robustness. Note that the ciphertext $\mathsf{ct}_{\mathsf{msg}}$ is generated honestly because of the perfect soundness of the NIZK. Therefore, by perfect correctness of witness encryption, it follows that there cannot exist an adversary that will output two valid witnesses for stmt for which $\mathsf{ct}_{\mathsf{msg}}$ will decrypt to different messages.

For plaintext awareness, note that we can run the extractor to obtain the message msg from NIZK proof, and from perfect correctness of the underlying WE scheme we have that for all valid witnesses wit, the decryption function will return the message msg.

Theorem 12 (Soundness). *Scheme 3 is an argument system for* NP *language* $\mathcal{L}$ *assuming the witness encryption scheme* WE *for* $\mathcal{L}$ *is secure. Furthermore, if the underlying witness encryption scheme* WE *scheme is extractable, then Scheme 3 is an argument of knowledge.*

Proof. If the witness encryption scheme WE is extractable, then knowledge soundness directly follows from from the extractability of WE. Thus we will put our focus on the case where WE is only secure.

Without loss of generality, let us assume that we have a statement $\mathsf{stmt} \notin \mathcal{L}$. We will show that for any PPT adversary $\mathcal{A}$ the probability that an honest execution of the Verify algorithm interacting with $\mathcal{A}$ outputs 1 is negligible. To do this, we will use two hybrids.

$\mathcal{H}_0$**:** This is the real soundness experiment.
$\mathcal{H}_1$**:** We replace ct with a ciphertext encrypting the message 0.

Claim. Hybrids $\mathcal{H}_0$ and $\mathcal{H}_1$ are computationally indistinguishable assuming the security of the witness encryption scheme. More specifically, there exists a reduction $\mathcal{R}_1$ such that

$$\mathsf{Adv}^{\mathsf{WE}}_{\mathcal{R}_1}(\lambda) = |\Pr[\mathcal{H}_1(\mathcal{A}) = 1] - \Pr[\mathcal{H}_0(\mathcal{A}) = 1]|.$$

Proof (Sketch). Since stmt is false, any adversary $\mathcal{A}$ that is able to distinguish between those hybrids can be used to construct a reduction $\mathcal{R}_1$ that breaks security of the witness encryption scheme.

Claim. For hybrid $\mathcal{H}_1$ we have $\Pr[\mathcal{H}_1(\mathcal{A}) = 1] = \frac{1}{2^\lambda}$.

Proof. It is easy to see that since the ciphertext ct send to the prover is independent of the challenge r, it follows that the prover has only a chance to guess the correct argument arg. However, since the challenge space is of size 2^λ, it follows that $\mathcal{A}$ advantage in breaking soundness in hybrid $\mathcal{H}_1$ is $\frac{1}{2^\lambda}$, which is negligible in the security parameter.

We conclude that $\Pr[\mathcal{H}_0(\mathcal{A}) = 1] = \mathsf{Adv}^{\mathsf{WE}}_{\mathcal{R}_1}(\lambda) + \frac{1}{2^\lambda}$.

Theorem 13 (Zero-Knowledge). *Scheme 3 is zero-knowledge given the underlying witness encryption scheme* WE *is plaintext aware.*

Proof (Sketch). The proof follows immediately from plaintext awareness of the WE scheme. The simulator runs the extractor of the plaintext aware WE scheme and responds with the extracted message.

Theorem 14 (Explainability). *Scheme 3 is fully explainable assuming the used witness encryption scheme is robust (or plaintext aware) and correct.*

Proof (Sketch). It is easy to see that the scheme 3 is malicious prover explainable. This follows from the fact that there are no random coins used on the side of the prover and the verifier is honest, so malicious prover explainability follows from the correctness of the witness encryption.

From perfect soundness of NIZK we have that there exists a message msg such that $\mathsf{ct} = \mathsf{WE.Enc}(\lambda, \mathsf{stmt}, \mathsf{msg})$. Furthermore, from perfect correctness, of the WE scheme we have that ct decrypts to msg for all valid witnesses. If the witness encryption scheme is plaintext aware and perfectly correct, then the reduction runs the extractor to obtain the plaintext arg. Furthermore, if for any wit or wit^*, the decryption result would differ msg then the fact would contradict perfect correctness of the witness encryption.

Theorem 15. *Let* WE *be a (non-robust) perfectly correct witness encryption scheme for* NP*. Let* Π *be an interactive public-coin zero-knowledge proof protocol for* NP*. Then there exists a malicious verifier explainable (and witness-indistinguishable) argument for* NP*.*

Proof (Sketch). The idea for the protocol is as earlier, but we need to show consistency of the ciphertext. The idea is to run the public-coin zero-knowledge protocol, where the prover acts as the verifier, and the verifier proofs correctness of the witness encryption ct. From soundness of the proof protocol we have consistency of the WE scheme just as in case of robust encryption. Malicious verifier explainability holds because all prover coins are included in the transcript. Therefore, to explain a an argument it is sufficient to return the messages from the given transcript.

Note that the construction given in the proof of Theorem 15 is not malicious prover explainable, because, the prover can choose some random coins.

Theorem 16. *Scheme 4 is an encryption scheme secure against non-adaptive chosen ciphertext attacks (CCA1) in the random oracle model assuming the encryption scheme $\mathcal{E}$ is an encryption scheme secure against chosen plaintext attacks and* NIWI *is a sound and witness indistinguishable proof system.*

Proof. We will prove this theorem using a series of hybrid arguments. Let $\mathsf{ct}^* = (\mathsf{ct}_1^*, \mathsf{ct}_2^*, \Pi^*)$ denote the challenged ciphertext for messages $\mathsf{msg}_0^*, \mathsf{msg}_1^*$. We assume the adversary will make at most q_h random oracle queries.

$\mathcal{H}_0$**:** This is the original CCA1 experiment where the challenger encrypts message msg_0^* (bit $b = 0$).

$\mathcal{H}_1$**:** We program the random oracle $\mathcal{H}_{\mathbb{G}}$ to output $(g^\alpha, g^\beta, g^{\alpha\cdot\beta})$ on input $(\mathsf{ct}_1^*, \mathsf{ct}_2^*)$ and retain (α, β). Additionally, we ensure that for all other random oracle queries the output is a non-DDH tuple, i.e. for the i-th query we choose (a_i, b_i, c_i) at random and output $(g^{a_i}, g^{b_i}, g^{c_i})$ and abort in the unlikely event of $a_i \cdot b_i = c_i$.

$\mathcal{H}_2$**:** We replace the way the proof Π^* is computed. Instead using witness (msg) we use the trapdoor witness (α, β) from hybrid $\mathcal{H}_1$.

$\mathcal{H}_3$**:** We change the ciphertext ct_2^* is generated and use $\mathsf{ct}_2 \leftarrow \mathsf{Enc}_{\mathsf{cpa}}(\mathsf{pk}_2, r)$ for some random r.

Claim. $\mathcal{H}_1$ aborts with probability at most $(q_h - 1)/p$ which is a negligible factor if the $\mathbb{G}$ order p is exponential in the security parameter.

Proof (Sketch). This follows from simple calculations.

Claim. Hybrids $\mathcal{H}_1$ and $\mathcal{H}_2$ are computationally indistinguishable assuming witness indistinguishability of NIWI. More specifically, there exists a reduction $\mathcal{R}_1$ such that

$$\mathsf{Adv}^{\mathsf{WI}}_{\mathcal{R}_1}(\lambda) = |\Pr[\mathcal{H}_2(\mathcal{A}) = 1] - \Pr[\mathcal{H}_1(\mathcal{A}) = 1]|.$$

Proof (Sketch). Any adversary $\mathcal{A}$ that is able to distinguish between those hybrids can be used to construct a reduction $\mathcal{R}_1$ that breaks witness indistinguishability.

Claim. Hybrids $\mathcal{H}_2$ and $\mathcal{H}_3$ are computationally indistinguishable assuming CPA security of $\mathcal{E}$. More specifically, there exists a reduction $\mathcal{R}_2$ such that

$$\mathsf{Adv}^{\mathsf{cpa}}_{\mathcal{R}_2}(\lambda) = |\Pr[\mathcal{H}_3(\mathcal{A}) = 1] - \Pr[\mathcal{H}_2(\mathcal{A}) = 1]|.$$

Proof (Sketch). Any adversary $\mathcal{A}$ that is able to distinguish between those hybrids can be used to construct a reduction $\mathcal{R}_2$ that distinguishes plaintexts in the CPA experiment. It is worth noting that in the simulation of the CCA1 experiment we use the secret key sk_1 to answer decryption queries. Therefore the public key pk_2 can be set as the public key given by the challenger in the CPA experiment and we can use the challenged ciphertext from the CPA experiment to compute ct_2^*.

Claim. The advantage of an adversary $\mathcal{A}$ in breaking CCA1 security in hybrid 3 is at most 2 times the advantage in breaking CPA security of $\mathcal{E}$ and breaking the soundness property of NIWI. More formally, there exists a reduction $\mathcal{R}_3$ such that:

$$\mathsf{Adv}^{\mathsf{cca1}}_{\mathcal{A}}(\lambda) \leq 2 \cdot (\mathsf{Adv}^{\mathsf{cpa}}_{\mathcal{R}_3}(\lambda) + \mathsf{Adv}^{\mathsf{Sound}}_{\mathcal{R}_3}(\lambda))$$

Proof. The idea is as follows. With probability 1/2 the reduction $\mathcal{R}_3$ will work as in hybrid 3 but will abort in case $\mathcal{A}$ proves a false statement. To this end for every decryption query $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2, \Pi)$ the reduction will decrypt ct_1, ct_2 and compare the plaintexts. In case they do not match $\mathcal{A}$ was able to break the soundness property of NIWI which can be used by $\mathcal{R}_3$.
In the other case we do not abort and allow for false statements. However, when this happens we just assume that the reduction will not be able to break the CPA security of $\mathcal{E}$ and with probability 1/2 $\mathcal{R}_3$ will still be able to use $\mathcal{A}$ to break soundness of NIWI as described above. This simple trick allows us to replace the secret key we use inside the decryption oracle from sk_1 to sk_2. This way $\mathcal{R}_3$ can set pk_1 to a public key received from the challenger in a CPA experiment. What is more, since in the previous hybrid we replaced ct_2 with a ciphertext of a random message then if $\mathcal{A}$ is able to distinguish messages in the CCA1 experiment then $\mathcal{R}_3$ can use $\mathcal{A}$ to distinguish messages for a CPA experiment.

Theorem 17. *Scheme 4 is an publicly deniable encryption scheme secure against non-adaptive chosen ciphertext attacks (CCA1) in the random oracle model assuming the encryption scheme $\mathcal{E}$ is an publicly deniable encryption scheme secure against chosen plaintext attacks and NIWI is a malicious setup explainable argument system.*

Proof (Sketch). Both ciphertexts are created using publicly deniable encryption, so they can be easily explained. The problem in scheme 4 is the proof Π. However, if we use a malicious setup explainable argument system then we can easily explain also Π.

B.1 Explainable Anonymous Authentication

In this section we show a simple interactive authentication scheme that is anonymous in a similar manner to ring signatures. We build the scheme from a standard signature scheme and a explainable zero-knowledge argument of knowledge. We focus on the interactive scheme since we can instantiate the argument

with our scheme given in Section 5. We note that we may similarly construct a non-interactive ring-signature, but we would need to have a non-interactive zero-knowledge argument of knowledge.

Definition 26. *We define an anonymous interactive authentication scheme to consist of algorithms* $\mathsf{KeyGen}, \mathsf{Prove}, \mathsf{Verify}$. *The* KeyGen *algorithm takes as input a secret key* $\mathsf{sk} \in \{0,1\}^p$ *and outputs a pair* $(\mathsf{sk}, \mathsf{pk})$, *where* $p = \mathsf{poly}(\lambda)$ *and* λ *is the security parameter. Then prove algorithm take as input* sk *and a set of public keys* R, *and a message* msg. *The verifier takes as input the set public keys* R, *and engages into a interactive protocol with the prover. Finally, the verifier returns a bit.*

Correctness: *We say that an interactive authentication scheme is perfectly correct if for any security parameter* $\lambda \in \mathbb{N}$, *any set* $R = [\mathsf{pk}_i]_{i=1}^n$ *where* $n = \mathsf{poly}(\lambda)$, $\mathsf{sk}_i \in \{0,1\}^p$, $p = \mathsf{poly}(\lambda)$, $(\mathsf{pk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{sk}_i)$, *for all* $i \in [n]$, *all messages* msg *in the message space and any* $j \in [n]$ *we have*

$$\Pr[\langle \mathsf{Prove}(R, \mathsf{sk}, \mathsf{msg}) \rightleftharpoons \mathsf{Verify}(R, \mathsf{msg}) \rangle = 1] = 1.$$

Unforgeability: *We say that an interactive authentication scheme is unforgeable if for all PPT adversaries* $\mathcal{A} = (A_0, A_1)$, *there exists a negligible function* $\mathsf{negl}(.)$ *such that*

$$\Pr\left[\mathcal{A}_1(\mathsf{st}) \rightleftharpoons \mathsf{Verify}(R, \mathsf{msg}) = 1 : \begin{array}{c}(R, \mathsf{msg}, \mathsf{st}) \leftarrow \mathcal{A}_0^{\mathcal{O}}([\mathsf{pk}_i]_{i=1}^n); \\ R \cap C = \emptyset; (R, \mathsf{msg}) \notin \Sigma\end{array}\right] \leq \mathsf{negl}(\lambda),$$

where $n = \mathsf{poly}(\lambda)$ *and for all* $i \in [n]$ $\mathsf{sk}_i \in \{0,1\}^p$, $p = \mathsf{poly}(\lambda)$, $(\mathsf{pk}_i) \leftarrow \mathsf{Setup}(\mathsf{sk}_i)$, *and* $\mathcal{O}$ *is an oracle that is defined as follows: On input* $(\mathit{corrupt}, i)$, $\mathcal{O}$ *outputs* sk_i *and stores* pk_i *in the set* C. *On input* $(\mathit{interact}, i, \mathsf{msg}, R)$, $\mathcal{O}$ *acts as a interface to the* i*th prover on input* msg *and a set* R *and stores* (R, msg) *in the set* Σ. *The probability is taken over the random choice of* $[\mathsf{sk}_i]_{i=1}^n$ *and random coins of* Verify.

Anonymity: *We say that an interactive authentication scheme is anonymous if for all PPT adversaries* $\mathcal{A} = (A_0, A_1)$ *we have*

$$\Pr\left[\hat{b} = b : \begin{array}{c}(R, i_0, i_1, \mathsf{msg}, \mathsf{st}) \leftarrow A_0([(\mathsf{pk}_i, \mathsf{sk}_i)]_{i=1}^n); \\ i_0, i_1 \in R; b \leftarrow_R \{0,1\}; \\ \hat{b} \leftarrow A_1(\mathsf{st}) \rightleftharpoons \mathsf{Prove}(\mathsf{sk}_b, R, \mathsf{msg})\end{array}\right] \leq \mathsf{negl}(\lambda),$$

where $n = \mathsf{poly}(\lambda)$, *and for all* $i \in [n]$ $\mathsf{sk}_i \in \{0,1\}^p$, $p = \mathsf{poly}(\lambda)$, *and* $(\mathsf{pk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{sk}_i)$. *The probability is taken over the random coins of* Prove *and choice of* b.

Explainability: *Since the anonymous authentication scheme is an interactive protocol with a prover and verifier we can use any of the definitions of explainability for interactive arguments given in Section 3.*

Definition 27 (A Simple Anonymous Authentication Scheme). *Let* $\mathsf{Sig} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *be a signature scheme, and let* $\Pi_{\mathcal{R}} = (\mathsf{Prove}, \mathsf{Verify})$ *be a*

argument system. We construct a anonymous authentication scheme as follows. The KeyGen *algorithm generates a public and secret key for the signature scheme. Denote as* pk, sk *the public/secret key of the prover. The prover on input a set R, first check whether its public key is in R, and if so, the prover computes* $\mathsf{sig} \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{msg}||R)$. *Finally, the prover and verifer run a interactive argument for the statement* stmt*:*

$$\{\exists_{\mathsf{pk},\mathsf{sk},\mathsf{sig}} \mathsf{pk} \in R \quad \textit{and} \quad \mathsf{sig} \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{msg})\}$$

The verifier accepts if the argument verifier accepts.

Theorem 18 (Unforgeability). *Let* Sig *be a unforgeable signature scheme and Π be a zero-knowledge argument of knowledge. Then the scheme from Definition 27 is unforgeable.*

Proof (Sketch). Let sk, pk be a secret and public key from the unforgeability experiment of the signature scheme. We choose $j \in [n]$ uniformly at random, and set $\mathsf{pk}_j = \mathsf{pk}$. All other secret keys are generated by the reduction. Given a corruption query for i, the reduction responds with sk_i given that $i \neq j$. If the adversary asks to corrupt j, the reduction aborts, what may happen with probability $1/n$. On input a interaction query, the reduction runs the simulator of the zero-knowledge argument. Finally, suppose the adversary runs a interaction for the set R^* and message msg^* such that $R^* \cup C = \emptyset$ and $(R^*, \mathsf{msg}^*) \notin \Sigma$. Then the reduction runs the extractor of the argument system and obtains $\mathsf{pk}', \mathsf{sk}', \mathsf{sig}'$ such that $\mathsf{pk}' \in R$ and $\mathsf{sig}' \leftarrow \mathsf{Sign}(\mathsf{sk}', \mathsf{msg}')$. With probability $1/n$ we have that $pk' = \mathsf{pk}$, in which case we return msg^* and sig as a forge in the unforgeability experiment of the signature scheme.

Theorem 19. *Let Sig be a signature scheme and Π be a zero-knowledge argument system. Then the scheme from Definition 27 is anonymous.*

Proof (Sketch). The proof trivially follows from the zero-knowledge property of the argument system.

Theorem 20 (Explanability). *Let* Sig *be a signature scheme. If Π is malicious verifier explainable, then the scheme from Definition 27 is malicious verifier explainable. If Π is malicious prover explainable, then the scheme from Definition 27 is malicious prover explainable. If Π is fully explainable, then the scheme from Definition 27 is fully explainable.*

Proof (Sketch). The proof is a straightforward consequence of the fact the the entire protocol transcript consists only of the messages of the argument system.