

# Explainable Arguments

Lucjan Hanzlik<sup>1</sup> and Kamil Kluczniak<sup>1,2</sup>

<sup>1</sup> CISA Helmholtz Center for Information Security

<sup>2</sup> Stanford University

lucjan.hanzlik@{cisa.saarland}

kamil.kluczniak@{cisa.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 [15,63,16], 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 [58] and called unclaimability. The property states that no one can claim to be the signer of a particular ring signature  $\sigma$ . 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  $\sigma$ .

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  $arg = \text{Prove}(stmt, wit; coins)$  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 [63], 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 full paper [45]). Notably, we can construct CCA-1 secure encryption with deniability as defined by Sahai and Waters [63], from CPA secure deniable encryption and our explainable arguments assuming random oracles. Our deniable encryption is a variant of the Naor-Yung transform [56], 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 [55,48,25] 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 [24,11,49,28,69,68,46,27,66,51,65,67]. 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 [59] 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 [7,57,63,20,21,1,22,41]. 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 [61], 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 [58] 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 [17,62] 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 Coercion 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 [6,64,47,54]. 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 [45], 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  $a \leftarrow \text{Alg}(x)$  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  $\text{Alg}(\cdot; r)$  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  $\langle \text{Prove}(\cdot) \rightleftharpoons \text{Verify}(\cdot) \rightarrow x \rangle = \text{trans}$ , where  $x$  is the output of  $\text{Verify}$  after completion of the protocol, and  $\text{trans}$  is the transcript of the protocol. A transcript  $\text{trans}$  contains all messages send between  $\text{Prove}$  and  $\text{Verify}$  and the input of  $\text{Verify}$ . We write  $\text{View}(\text{Prove}(\cdot) \rightleftharpoons \text{Verify}(\cdot))$  to denote the view of  $\text{Verify}$ . The view contains the transcript, all input to  $\text{Verify}$  including its random coins and its internal state. We say that a function  $\text{negl} : \mathbb{N} \mapsto \mathbb{R}^+$  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  $\text{negl}(\lambda) < \lambda^{-c}$ .

*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 [63], witness encryption [36] and extractable witness encryption [40], indistinguishability obfuscation [3], and punctured pseudorandom functions [13,14,50].

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

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

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

*which satisfies the correctness definition below.*

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

$$\begin{aligned} \langle \text{Verify}(\text{stmt}) \rightrightarrows \text{Prove}(\text{stmt}, \text{wit}) \rangle = \\ \langle \text{Verify}'(\text{stmt}; \text{trans}) \rightrightarrows \text{Prove}(\text{stmt}, \text{wit}^*; \text{coins}_E) \rangle = \text{trans}, \end{aligned}$$

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

*Remark 1.* Note that a naive way to implement the  $\text{Expl}$  algorithm would be to set  $\text{coins}_E$  and make the  $\text{Prove}$  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  $\text{coins}_E \in \mathbf{Coin}_{\text{Prove}}$  to ensure that  $\text{coins}_E$  can be given as input to an honest  $\text{Prove}$  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),  $\text{Expl}$  generates coins such that a honest prover returns the same messages given that the verifier send its messages as in  $\text{trans}$ .

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  $\lambda$ , we define the advantage  $\text{Adv}_{\mathcal{A}}^{\text{MVExpl}}(\lambda)$  of an adversary  $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$  as

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

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

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

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  $\text{negl}(\cdot)$  such that  $\text{Adv}_{\mathcal{A}}^{\text{MVEspl}}(\lambda) \leq \text{negl}(\lambda)$ . 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  $\text{Adv}_{\mathcal{A}}^{\text{MPEspl}}(\lambda)$  of an adversary  $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$  as

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

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

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

**Theorem 1.** If  $(\text{Prove}, \text{Verify}, \text{Expl})$  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  $\Pi_{\mathcal{R}} = (\text{Setup}, \text{Prove}, \text{Verify})$  for language  $\mathcal{L}_{\mathcal{R}}$  is a non-interactive explainable argument system if there exists an additional  $\text{Expl}$  algorithm:



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

which satisfies the correctness definition below.

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

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

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

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  $\text{Expl}$  algorithm to output  $\text{coins}$  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  $\text{Adv}_{\mathcal{A}}^{\text{MSExp}}(\lambda)$  of an adversary  $\mathcal{A}$  by the following probability

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

where the probability is taken over the random coins of the prover  $\text{Prove}$ . We say that a non-interactive argument is malicious setup explainable if for all PPT adversaries  $\mathcal{A}$  there exists a negligible function  $\text{negl}(\cdot)$  such that  $\text{Adv}_{\mathcal{A}}^{\text{MSExp}}(\lambda) \leq \text{negl}(\lambda)$ . We say 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  $\text{Adv}_{\mathcal{A}}^{\text{MSExp}}(\lambda) = 0$ .

**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  $\text{ExArg}$  as  $\text{Adv}_{\mathcal{A}}^{\text{MPExpl}}(\lambda) = \Pr[\text{arg}_1 \neq \text{arg}_2]$  where  $\text{crs} \leftarrow \text{Setup}(\lambda)$  and  $(\text{stmt}, \text{arg}_1, \text{arg}_2) \leftarrow \mathcal{A}(\lambda)$  are such that  $\text{Verify}(\text{crs}, \text{stmt}, \text{arg}_1) = \text{Verify}(\text{crs}, \text{stmt}, \text{arg}_2)$ , and the probability is over the random coins of  $\text{Setup}$ . We say that a non-interactive argument is malicious prover explainable if for all PPT adversaries  $\mathcal{A}$  there exists a negligible function  $\text{negl}(\cdot)$  such that  $\text{Adv}_{\mathcal{A}}^{\text{MPExpl}}(\lambda) \leq \text{negl}(\lambda)$ . 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  $\text{Adv}_{\mathcal{A}}^{\text{MPExpl}}(\lambda) = 0$ .

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  $\text{ExArg}$  by the following probability

$$\text{Adv}_{\mathcal{A}}^{\text{FExpl}}(\lambda) = \Pr[\text{arg}_1 \neq \text{arg}_2]$$

where  $(\text{stmt}, \text{crs}, \text{arg}_1, \text{arg}_2) \leftarrow \mathcal{A}(\lambda)$  is such that  $\text{Verify}(\text{crs}, \text{stmt}, \text{arg}_1) = \text{Verify}(\text{crs}, \text{stmt}, \text{arg}_2)$ . We say that a non-interactive argument is full explainable if for all PPT adversaries  $\mathcal{A}$ , there exists a negligible function  $\text{negl}(\cdot)$  such that  $\text{Adv}_{\mathcal{A}}^{\text{FExpl}}(\lambda) \leq \text{negl}(\lambda)$ . 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  $\text{Adv}_{\mathcal{A}}^{\text{FExpl}}(\lambda) = 0$ .

**Theorem 3.** If  $\text{ExArg}$  is a fully explainable argument, then  $\text{ExArg}$  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  $\text{ExArg}^{\nabla} = (\text{Setup}, \text{Prove}, \text{Verify})$  as follows.

Circuit for $\text{ProgProve}_o^1$ and $\text{ProgProve}_\dagger^1$	Circuit for $\text{ProgVerify}$
<b>Hardwired:</b> $\text{pp}, \text{crs}_{\text{DMWI}}, K$	<b>Hardwired:</b> $K$
<b>Input:</b> $(\text{stmt}, \text{wit})$	<b>Input:</b> $(\text{stmt})$
1 <sup>o</sup> : <b>if</b> $\text{DMWI.Verify}(\text{crs}_{\text{DMWI}}, \text{stmt}, \text{wit}) = 0$	1 : $\text{sk}_s \leftarrow \text{PRF.Eval}(K, \text{stmt})$
1 <sup>†</sup> : <b>if</b> $R(\text{stmt}, \text{wit}) = 0$	2 : $\text{vk}_s \leftarrow \text{Sig.Setup}(\text{sk}_s)$
2 : <b>return</b> $\perp$ .	3 : <b>return</b> $\text{vk}_s$
3 : <b>else</b>	
4 : $\text{sk}_s \leftarrow \text{PRF.Eval}(K, \text{stmt})$	
5 : $\text{arg} \leftarrow \text{Sig.Sign}(\text{sk}_s, \text{stmt})$	
6 : <b>return</b> $\text{arg}$	

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

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

1. Choose  $K \leftarrow \text{PRF.Setup}(\lambda)$  and  $\text{crs}_{\text{DMWI}} \leftarrow \text{DMWI.Setup}(\lambda, \text{modeSound}; \text{coins}_S)$ , where  $\text{coins}_S$  are random coins.
2.  $O_{\text{Prove}} \leftarrow \text{Obf}(\lambda, \text{ProgProve}_{\nabla}^1[\text{pp}, \text{crs}_{\text{DMWI}}, K]; \text{coins}_P)$ , where  $\text{ProgProve}_{\nabla}^1$  is given by Figure 1 and  $\text{coins}_P$  are random coins.
- 3<sup>o</sup>. Define statement  $\text{stmt}_{\text{Setup}}^o$  as

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

- 3<sup>†</sup>. Define statement  $\text{stmt}_{\text{Setup}}^\dagger$  as

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

4. Set  $\text{wit}_{\text{Setup}} = (1, K, \text{coins}_P)$ .
- 5<sup>o</sup>.  $\pi \leftarrow \text{NIWI.Prove}(\text{stmt}_{\text{Setup}}^o, \text{wit}_{\text{Setup}})$ .
- 5<sup>†</sup>.  $\pi \leftarrow \text{NIZK.Prove}(\text{stmt}_{\text{Setup}}^\dagger, \text{wit}_{\text{Setup}})$ .
6. Compute  $O_{\text{Verify}} \leftarrow \text{Obf}(\lambda, \text{ProgVerify}[K])$  and output  $\text{crs} = (O_{\text{Prove}}, O_{\text{Verify}}, \text{pp}, \text{etd}, \text{crs}_{\text{DMWI}}, \pi)$ .

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

- 1<sup>o</sup>. Set  $\text{stmt}_{\text{Setup}}^o$  as in the setup algorithm.
- 1<sup>†</sup>. Set  $\text{stmt}_{\text{Setup}}^\dagger$  as in the setup algorithm.
- 2<sup>o</sup>. If  $\text{NIWI.Verify}(\text{stmt}_{\text{Setup}}^o, \pi) = 0$  return  $\perp$ .
- 2<sup>†</sup>. If  $\text{NIZK.Verify}(\text{stmt}_{\text{Setup}}^\dagger, \pi) = 0$  return  $\perp$ .
- 3<sup>o</sup>. Run  $\text{wit}' \leftarrow \text{DMWI.Prove}(\text{crs}_{\text{DMWI}}, \text{stmt}, \text{wit}; r)$  and  $\text{arg} \leftarrow O_{\text{Prove}}(\text{stmt}, \text{wit}')$ .
- 3<sup>†</sup>. Run  $\text{arg} \leftarrow O_{\text{Prove}}(\text{stmt}, \text{wit})$ .
4. Run  $\text{vk}_s \leftarrow O_{\text{Verify}}(\text{stmt})$ .
5. If  $\text{Sig.Verify}(\text{vk}_s, \text{arg}, \text{stmt}) \neq 1$  return  $\perp$ .

6. Otherwise, return arg.
- Verify**(crs, stmt, arg):
1. Run  $vk_s \leftarrow O_{\text{Verify}}(\text{stmt})$ .
  2. Output  $\text{Sig.Verify}(vk_s, \text{sig}, \text{msg})$
- Expl**(crs, stmt, wit, arg):
1. Output 0.

Circuit for $\text{ProgProve}_\circ^2$ and $\text{ProgProve}_\dagger^2$	Circuit for $\text{ProgVerify}^*$
<b>Hardwired:</b> $\text{crs}_{\text{DMWI}}, \text{pp}$	<b>Hardwired:</b> $\text{stmt}^*, vk_s^*$ ,
$K_{\text{stmt}^*} = \text{PRF.Puncture}(K, \text{stmt}^*)$	$K_{\text{stmt}^*} = \text{PRF.Puncture}(K, \text{stmt}^*)$
<b>Input:</b> (stmt, wit, r)	<b>Input:</b> (stmt)
1 <sup>◦</sup> : <b>if</b> $\text{DMWI.Verify}(\text{crs}_{\text{DMWI}}, \text{stmt}, \text{wit}) = 0$	1 : <b>if</b> $\text{stmt} = \text{stmt}^*$
1 <sup>†</sup> : <b>if</b> $R(\text{stmt}, \text{wit}) = 0$	2 : <b>return</b> $vk_s^*$
2 : <b>return</b> $\perp$ .	3 : <b>else</b>
3 : <b>else</b>	4 : $sk_s \leftarrow \text{PRF.Eval}(K_{\text{stmt}^*}, \text{stmt})$
4 : $sk_s \leftarrow \text{PRF.Eval}(K_{\text{stmt}^*}, \text{stmt})$	5 : $vk_s \leftarrow \text{Sig.Setup}(sk_s)$
5 : $\text{arg} \leftarrow \text{Sig.Sign}(sk_s, \text{stmt})$	6 : <b>return</b> $vk_s$
6 : <b>return</b> arg	

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

**Theorem 5.** *Let  $\text{ExArg}^\circ$  be the system given by Scheme 1. The system  $\text{ExArg}^\circ$  is computationally sound (in the selective setting) assuming indistinguishability obfuscation of  $\text{Obf}$ , 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  $\text{Sig}$  is unique, NIWI is perfectly sound, DMWI is a dual-mode proof, and all primitives are perfectly correct, the argument system  $\text{ExArg}^\circ$  is malicious setup explainable.*

**Theorem 7.** *Let  $\text{ExArg}^\dagger$  be the system given by Scheme 1. The system  $\text{ExArg}^\dagger$  is computationally sound (in the selective setting), assuming indistinguishability obfuscation of  $\text{Obf}$ , 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  $\text{Sig}$  is unique, NIZK is sound, and all primitives are perfectly correct, argument system  $\text{ExArg}^\dagger$  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  $\text{ExArg}^\nabla$  be the system given by Scheme 1 for  $\nabla = \circ$  or  $\nabla = \dagger$ .  $\text{ExArg}^\nabla$  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  $\text{WE} = (\text{Enc}, \text{Dec})$  a robust witness encryption scheme if it is correct, secure and robust as defined below:*

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

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

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

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

*and there exists a PPT extractor  $\text{Ext}$  such that*

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

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

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

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

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

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

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

**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  $\text{stmt}$  and a witness  $\text{wit}$ , and Verify takes as input  $\text{stmt}$ . First Verify chooses  $r \leftarrow_{\$} \{0, 1\}^\lambda$ , computes  $\text{ct} \leftarrow \text{Enc}_{rob}(\lambda, \text{stmt}, r)$  and sends  $\text{ct}$  to Prove. Then Prove computes  $\text{arg} \leftarrow \text{Dec}_{rob}(\text{stmt}, \text{wit}, \text{ct})$  and sends  $\text{arg}$  to Verify. Finally, Verify returns iff  $\text{arg} = r$ . The explain algorithm Expl is as follows.

$\text{Expl}(\text{stmt}, \text{wit}, \text{trans})$ : On input the statement  $\text{stmt}$ , the witness  $\text{wit}$  and a transcript  $\text{trans}$ , 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 II 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 [56] 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  $ct_1, ct_2$  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 done before.

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

$$\{(\exists_{\text{msg}} ct_1 \leftarrow \text{Enc}_{\text{cpa}}(\text{pk}_1, \text{msg}) \wedge ct_2 \leftarrow \text{Enc}_{\text{cpa}}(\text{pk}_2, \text{msg})) \vee (\exists_{\alpha, \beta} \mathcal{H}_{\mathbb{G}}(ct_1, ct_2) = (g^\alpha, g^\beta, g^{\alpha \cdot \beta}))\},$$

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

$\text{KeyGen}_{\text{cca1}}(\lambda)$ :

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

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

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

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

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

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

## 7 Conclusions

In this paper, we introduce new security definitions for interactive and non-interactive argument systems that formally capture a property called explainability. Such arguments can be used to construct CCA1 deniable encryption and deniable anonymous authentication. We also introduced a new property for witness encryption called robustness which can be of independent interest. An interesting open question is whether such argument systems can be constructed from simpler primitives or we need such strong primitives because malicious prover explainability implies uniqueness of the proof.

**Acknowledgements.** This work has been partially funded/supported by the German Ministry for Education and Research through funding for the project CISP-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 and K. Kluczniak. Explainable arguments. To Appear in Cryptology ePrint Archive, Report 2021/xxxx, 2021. <https://ia.cr/2021/xxxx>.
46. 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.
47. 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.
48. 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.
49. 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.
50. 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.
51. L. Krzywiecki, K. Kluczniak, P. Kozieł, and N. Panwar. Privacy-oriented dependency via deniable sigma protocol. *Computers & Security*, 79:53 – 67, 2018.
52. 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.
53. S. Micali, M. O. Rabin, and S. P. Vadhan. Verifiable random functions. In *40th FOCS*, pages 120–130. IEEE Computer Society Press, Oct. 1999.

54. 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.
55. M. Naor. Deniable ring authentication. In M. Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 481–498. Springer, Heidelberg, Aug. 2002.
56. 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.
57. 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.
58. 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.
59. 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.
60. 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.
61. 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.
62. 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.
63. 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.
64. 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.
65. 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.
66. 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.
67. N. Vatasdas, R. Gennaro, B. Ithurburn, and H. Krawczyk. On the cryptographic deniability of the Signal protocol. In M. Conti, J. Zhou, E. Casalichio, and A. Spognardi, editors, *ACNS 20, Part II*, volume 12147 of *LNCS*, pages 188–209. Springer, Heidelberg, Oct. 2020.
68. 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.
69. 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.