(Commit-and-Prove) Predictable Arguments with Privacy

Hamidreza Khoshakhlagh

Aarhus University, Denmark

Abstract. Predictable arguments introduced by Faonio, Nielsen and Venturi (PKC17) are private-coin argument systems where the answer of the prover can be predicted in advance by the verifier. In this work, we study predictable arguments with additional privacy properties. While the authors in [PKC17] showed compilers for transforming PAs into PAs with zero-knowledge property, they left the construction of witness indistinguishable predictable arguments (WI-PA) in the plain model as an open problem. In this work, we first propose more efficient constructions of zero-knowledge predictable arguments (ZK-PA) based on trapdoor smooth projective hash functions (TSPHFs). Next, we consider the problem of WI-PA construction in the plain model and show how to transform PA into WI-PA using non-interactive witness-indistinguishable proofs.
As a relaxation of predictable arguments, we additionally put forth a new notion of predictability called *Commit-and-Prove Predictable Argument* (CPPA), where except the first (reusable) message of the prover, all the prover's responses can be predicted. We construct an efficient zero-knowledge CPPA in the non-programmable random oracle model for the class of all polynomial-size circuits. Finally, following the connection between predictable arguments and witness encryption, we show an application of CPPAs with privacy properties to the design of witness encryption schemes, where in addition to standard properties, we also require some level of privacy for the decryptors who own a valid witness for the statement used during the encryption process.

Keywords: predictable arguments, zero-knowledge, witness indistinguishability, witness encryption

1 Introduction

Interactive proofs (IPs) and arguments introduced by Goldwasser, Micali, and Rackoff [GMR89] are cryptographic protocols that allow a prover to convince a verifier about the veracity of a public statement $\mathsf{x} \in \mathcal{L}$, where $\mathcal{L}$ is an NP language. The interaction may consist of several rounds of communication, at the end of which the verifier decides to accept or reject the prover's claim on the membership of x in $\mathcal{L}$. There are two properties required for an IP, namely *completeness* and *soundness*. Completeness means that if $\mathsf{x} \in \mathcal{L}$, the honest prover can always convince the honest verifier. Soundness means that for $\mathsf{x} \notin \mathcal{L}$

no (even unbounded) malicious prover can convince the honest verifier that $\mathsf{x} \in \mathcal{L}$. Argument systems are like IPs, except they are only computationally sound; i.e., it should be *computationally hard* (and not impossible) for a malicious prover to convince the verifier that $\mathsf{x} \in \mathcal{L}$. An interactive proof is called *public-coin* if the verifier messages are uniformly and independently random, and *private-coin* otherwise.

Recently, Faonio, Nielsen and Venturi [FNV17] introduced a new property for argument systems called *predictability.* Predictable arguments (PA) are private-coin argument systems where the answer of the prover can be predicted efficiently, given the honest verifier's (private) random coins. The prover in such arguments is deterministic and must be consistent with the unique accepting transcript throughout the entire protocol. Faonio et al. [FNV17] formalized this notion and provided several constructions based on various cryptographic assumptions. They also considered PAs with additional privacy properties, namely a *zero-knowledge* (ZK) property, and showed two transformations from PAs into ZK-PAs, the first in the common reference string (CRS) model, and the second in the non-programmable random oracle (NPRO) model.

1.1 Our Contribution

In this paper, we study predictable arguments with privacy properties in more detail. Our results are three-fold:

First, we provide a more efficient construction of ZK-PA in the CRS model. Compared to the generic transformation of [FNV17], the resulting argument is much more efficient although it works only for a restricted class of languages; i.e., all languages that admit SPHFs. This includes all algebraic languages described in section 2.0.2.

Second, we answer an open problem raised in [FNV17] and show how to construct witness indistinguishable PAs (WI-PA) in the plain model by using non-interactive witness indistinguishable (NIWI) proofs in the plain model. Informally, in order to ensure that the verifier's challenge in the first round is well-formed, we force the verifier to provide a NIWI proof for the statement that "the produced challenge is well-formed". Witness-indistinguishability follows from the soundness of the underlying NIWI and the predictability of the argument. Moreover, we provide a reduction that shows how an adversary breaking the soundness of the WI-PA can be exploited in order to violate the WI property of the underlying NIWI proof system.

Third, motivated by the fact that predictable argument (even without privacy properties) is a strong notion [1], we put forward a relaxation of predictable arguments, namely, commit-and-prove [2] predictable arguments (CPPA) that,

[1] This follows by the fact that predictable arguments and witness encryption (that only exists based on strong primitives like indistinguishability obfuscation) are equivalent.

[2] We call our notion *commit-and-prove* PA because, roughly speaking, a prover first commits to an input (once and for all) and later proves that an opening for the commitment satisfies some properties of interest. Our name is also inspired by the phrase "commit-and-prove schemes" used in some papers, e.g., [CFQ19].

except the first message of the prover, all the prover's responses can be predicted. We formalize this notion for the language of dynamic statements of form $\mathsf{x} = (\mathsf{cm}, C, y)$, where cm is the prover's first message, and C is an arbitrary polynomial-size circuit possibly specified by the verifier. In particular, we consider a case where the prover publishes a first message cm, after which the prover can run an unbounded number of predictable arguments for different but correlated statements (cm, C_i, y_i). In contrast to PAs for which efficient construction based on standard assumptions (even without ZK) seems out of reach, we give a construction of ZK-CPPA for any polynomial-size circuit $C \in \mathsf{P}$ in the NPRO model using garbled circuits (GC) and oblivious transfer (OT). Our construction is very similar to the three-round zero-knowledge argument of [GKPS18] with the main difference being the reusability of the prover's first message and providing ZK in the non-UC model under milder assumptions.

Applications. To demonstrate the usefulness of (CP)PA with privacy properties, we will give its application in the context of witness encryption. We consider witness encryption schemes with a strong notion of privacy for the decryptor, wherein a malicious encryptor should not learn any information about the decryptor's witness, even after the decryptor reveals the decrypted message. Our motivating applications for this scenario are dark pools and over-the-counter (OTC) markets in which an investor (the encrypting party) is interested to communicate with only those trading parties (potential decryptors) whose financial conditions satisfy some constraint. To realize this application, a recent work by Ngo et al. [NMKW21] introduced the notion of *witness key agreement* (WKA) which allows the two sides to agree on a secret key k, given that the trading parties hold a witness that satisfies the desired relation. We show in section 6.1 that the witness encryption (WE) interpretation of our ZK-CPPA construction can be used to realize this application with an efficiency improvement in some aspects.

1.2 Related work

This paper is a follow-up to the work of Faonio et al. [FNV17] that introduced the notion of predictable arguments of knowledge (PAoK) systems. While PAs are always honest-verifier zero-knowledge, providing zero-knowledge or even the weaker notion of witness-indistinguishability is quite challenging. In [FNV17], the authors show a compiler for constructing ZK-PA in the CRS model and leave the construction of WI-PA in the plain model as an open problem. We answer the open problem and propose more efficient ZK-PAs in the CRS model. A related work is that of Bitansky and Choudhuri [BC20] who recently constructed deterministic-prover ZK arguments for NP and showed that such arguments imply ZK-PA for NP. Different from [BC20] who mainly focus on feasibility results and require strong assumptions (e.g., indistinguishability obfuscation) in their construction, our work considers practical solutions in the CRS model. In another related work, Dahari and Lindell [DL20] studied deterministic-prover honest verifier ZK arguments in the plain model. In the same work, they also constructed full ZK arguments given that the prover has access to a pair of witnesses one of which can be used as a basis for the prover's randomness. This differs from our ZK-PA construction wherein the prover is "truly deterministic" although at the cost of requiring a trusted setup. The recent work of [CPW20] introduced the notion of *Witness Maps.* A Unique Witness Map (UWM) is a cryptographic notion that maps all the witnesses for an NP statement to a single witness in a deterministic way. While UWMs can be seen as deterministic-prover NIWI arguments, they differ from WI-PA in several respects, making the two concepts incomparable. First, WI-PA does not require a trusted setup in the form of a common reference string, whereas UWMs are in the CRS model. Second, we consider WI-PA as an interactive protocol, whereas UWMs are non-interactive. Lastly, although UWMs are deterministic-prover, they are not necessarily predictable.

2 Preliminaries

Let PPT denote probabilistic polynomial-time. All adversaries throughout this work will be stateful. By $y \leftarrow \mathcal{A}(\mathsf{x}; r)$ we denote that $\mathcal{A}$, given input x and randomness r, outputs y. Let $\lambda \in \mathbb{N}$ be the security parameter and $\mathsf{negl}(\lambda)$ be an arbitrary negligible function. We write $a \approx_\lambda b$ if $|a - b| \leq \mathsf{negl}(\lambda)$.

2.0.1 Pairings.

A pairing is defined by a tuple $\mathsf{bp} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, g_1, g_2)$ where $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are (additive) groups of prime order p, g_1 is a generator of $\mathbb{G}_1$, g_2 is a generator of $\mathbb{G}_2$, and $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an efficient, non-degenerate bilinear map. In particular, $\hat{e}(a \cdot g_1, b \cdot g_2) = (ab) \cdot \hat{e}(g_1, g_2)$ for any $a, b \in \mathbb{Z}_p$. We denote $[a]_t := a \cdot g_t$ for $t \in \{1, 2, T\}$ where we define $g_T = \hat{e}(g_1, g_2)$. The same notation naturally extends to matrices $[M]_t$ for $M \in \mathbb{Z}_p^{n \times m}$.

2.0.2 Algebraic languages.

We refer to algebraic languages as the set of languages associated to a relation that can be described by algebraic equations over abelian groups. To be more precise, let $\mathtt{gpar}$ be some global parameters, generated by a probabilistic polynomial-time algorithm $\mathsf{setup}.\mathtt{gpar}$ which takes the security parameter λ as input. These global parameters can correspond to the description of groups involved in the construction and usually includes the description of a bilinear group. Throughout the paper, we suppose that these global parameters are implicitly given as input to each algorithm.

Let $\mathsf{lpar} = (\mathbf{M}, \boldsymbol{\theta})$ be a set of language parameters generated by a polynomial-time algorithm $\mathsf{setup}.\mathsf{lpar}$ which takes $\mathtt{gpar}$ as input. Here, $\mathbf{M} : \mathbb{G}^\ell \mapsto \mathbb{G}^{n \times k}$ and $\boldsymbol{\theta} : \mathbb{G}^\ell \mapsto \mathbb{G}^n$ are linear maps such that their different coefficients are not necessarily in the same algebraic structures. Namely, in the most common case, given a bilinear group $\mathtt{gpar} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, [1]_1, [2]_2)$, they can belong to either $\mathbb{Z}_p$, $\mathbb{G}_1$, $\mathbb{G}_2$, or $\mathbb{G}_T$ as long as the equation $\boldsymbol{\theta}(\mathbf{x}) = \mathbf{M}(\mathbf{x}) \cdot \mathbf{w}$ is "well-consistent".

Formally, for a set $\mathcal{X}_{\mathsf{lpar}}$ that defines the underlying domain, we define an algebraic language $\mathcal{L}_{\mathsf{lpar}} \subset \mathcal{X}_{\mathsf{lpar}}$ as

$$\mathcal{L}_{\mathsf{lpar}} = \left\{ \mathbf{x} \in \mathbb{G} \middle| \exists \mathbf{w} \in \mathbb{Z}_p^k : \boldsymbol{\theta}(\mathbf{x}) = \mathbf{M}(\mathbf{x}) \cdot \mathbf{w} \right\} \ . \tag{1}$$

An algebraic language where $\mathbf{M}$ is independent of $\mathbf{x}$ and $\boldsymbol{\theta}$ is the identity function is called a *linear language*.

Finally, we note that algebraic languages are as expressive as generic NP languages. This is because every binary circuit can be represented by a set of linear equations.

2.0.3 Smooth Projective Hash Function. Let $\mathcal{L}_{\mathsf{lpar}}$ be a NP language, parametrized by a language parameter lpar, and $\mathcal{R}_{\mathsf{lpar}} \subseteq \mathcal{X}_{\mathsf{lpar}}$ be its corresponding relation. A Smooth projective hash functions (SPHFs, [CS02]) for $\mathcal{L}_{\mathsf{lpar}}$ is a cryptographic primitive with this property that given lpar and a statement x, one can compute a hash of x in two different ways: either by using a projection key hp and $(\mathsf{x}, \mathsf{w}) \in \mathcal{R}_{\mathsf{lpar}}$ as $\mathsf{pH} \leftarrow \mathsf{projhash}(\mathsf{lpar}; \mathsf{hp}, \mathsf{x}, \mathsf{w})$, or by using a hashing key hk and $\mathsf{x} \in \mathcal{X}_{\mathsf{lpar}}$ as $\mathsf{H} \leftarrow \mathsf{hash}(\mathsf{lpar}; \mathsf{hk}, \mathsf{x})$. The formal definition of SPHF follows.

Definition 1. *A SPHF for* $\{\mathcal{L}_{\mathsf{lpar}}\}$ *is a tuple of PPT algorithms* ($\mathsf{setup}, \mathsf{hashkg}, \mathsf{projkg}, \mathsf{hash}, \mathsf{projhash}$)*, which are defined as follows:*

$\mathsf{setup}(1^\lambda)$**:** *Takes in a security parameter* λ *and generates the global parameters* pp *together with the language parameters* lpar*. We assume that all algorithms have access to* pp*.*

$\mathsf{hashkg}(\mathsf{lpar})$**:** *Takes in a language parameter* lpar *and outputs a hashing key* hk*.*

$\mathsf{projkg}(\mathsf{lpar}; \mathsf{hk}, \mathsf{x})$**:** *Takes in a hashing key* hk*,* lpar*, and a statement* x *and outputs a projection key* hp*, possibly depending on* x*.*

$\mathsf{hash}(\mathsf{lpar}; \mathsf{hk}, \mathsf{x})$**:** *Takes in a hashing key* hk*,* lpar*, and a statement* x *and outputs a hash value* H*.*

$\mathsf{projhash}(\mathsf{lpar}; \mathsf{hp}, \mathsf{x}, \mathsf{w})$**:** *Takes in a projection key* hp*,* lpar*, a statement* x*, and a witness* w *for* $\mathsf{x} \in \mathcal{L}$ *and outputs a hash value* pH*.*

A SPHF needs to satisfy the following properties:

Correctness. It is required that $\mathsf{hash}(\mathsf{lpar}; \mathsf{hk}, \mathsf{x}) = \mathsf{projhash}(\mathsf{lpar}; \mathsf{hp}, \mathsf{x}, \mathsf{w})$ for all $\mathsf{x} \in \mathcal{L}$ and their corresponding witnesses w.

Smoothness. It is required that for any lpar and any $\mathsf{x} \notin \mathcal{L}$, the following distributions are statistically indistinguishable:

$$\left\{ (\mathsf{hp}, \mathsf{H}) : \mathsf{hk} \leftarrow \mathsf{hashkg}(\mathsf{lpar}), \mathsf{hp} \leftarrow \mathsf{projkg}(\mathsf{lpar}; \mathsf{hk}, \mathsf{x}), \mathsf{H} \leftarrow \mathsf{hash}(\mathsf{lpar}; \mathsf{hk}, \mathsf{x}) \right\}$$
$$\left\{ (\mathsf{hp}, \mathsf{H}) : \mathsf{hk} \leftarrow \mathsf{hashkg}(\mathsf{lpar}), \mathsf{hp} \leftarrow \mathsf{projkg}(\mathsf{lpar}; \mathsf{hk}, \mathsf{x}), \mathsf{H} \leftarrow_{\$} \mathit{\Omega} \right\} \ .$$

where $\mathit{\Omega}$ is the set of hash values.

2.0.4 Predictable Arguments. Predictable arguments are multi-round interactive protocols where the verifier generates a challenge (which will be sent to the prover) and at the same time it can predict the prover's response to that challenge. Here we recall the formal definition of predictable arguments (PA) [FNV17][3].

Let $\mathcal{RG}$ be a relation generator that takes in a security parameter 1^λ and returns a polynomial-time decidable binary relation $\mathcal{R}_{\mathsf{lpar}}$. For a pair $(\mathsf{x}, \mathsf{w}) \in \mathcal{R}_{\mathsf{lpar}}$, we call x the statement and w the witness. The set of all possible relations $\mathcal{R}_{\mathsf{lpar}}$ that the relation generator $\mathcal{RG}$ (for a given 1^λ) may output is denoted by $\mathcal{RG}_\lambda$. To make the notation simple, we assume that $\mathcal{R}_{\mathsf{lpar}}$ can be described with a language parameter lpar by which λ can be deduced as well.

Definition 2 (Predictable Argument (PA)). *A predictable argument for a relation* $\mathcal{R}_{\mathsf{lpar}}$ *(with the corresponding language parameter* lpar*) is an interactive protocol between a prover* P *and a verifier* V*, which can be specified by two algorithms* $\Pi_{\mathsf{pa}} = (\mathsf{Chall}, \mathsf{Resp})$ *defined as follows:*

(Executed by V*):* $(c, b) \leftarrow \mathsf{Chall}(\mathsf{lpar}, \mathsf{x})$*. The algorithm takes in* lpar *and a statement* x*, and returns a challenge* c *along with a predicted answer* b*.*

(Executed by P*):* $a \leftarrow \mathsf{Resp}(\mathsf{lpar}, \mathsf{x}, \mathsf{w}, c)$*. The algorithm takes in* lpar*, a pair of statement-witness* (x, w) *and a challenge* c*, and returns an answer* a*.*

(Executed by V*): If* $a = b$*,* V *returns* acc*; otherwise it returns* rej*.*

We denote by $\langle \mathsf{P}(\mathsf{lpar}, \mathsf{x}, \mathsf{w}), \mathsf{V}(\mathsf{lpar}, \mathsf{x}) \rangle$ an execution between P and V with common inputs $(\mathsf{lpar}, \mathsf{x})$ and prover's secret input w. The success of the prover in convincing the verifier is denoted by $\langle \mathsf{P}(\mathsf{lpar}, \mathsf{x}, \mathsf{w}), \mathsf{V}(\mathsf{lpar}, \mathsf{x}) \rangle = \mathsf{acc}$. Also, we may call (c, b) as both the output of $\mathsf{Chall}()$, or the output of V running $\mathsf{Chall}()$. The same convention holds for a.

We require two properties for a PA: *completeness* and *soundness.*

- **(Perfect) Completeness.** A predictable argument has perfect completeness if for all $\lambda \in \mathbb{N}$, for all $\mathcal{R}_{\mathsf{lpar}} \in \mathcal{RG}_\lambda$, and for all $(\mathsf{x}, \mathsf{w}) \in \mathcal{R}_{\mathsf{lpar}}$

$$\Pr\left[a = b \ : \ (c, b) \leftarrow \mathsf{Chall}(\mathsf{lpar}, \mathsf{x}); a \leftarrow \mathsf{Resp}(\mathsf{lpar}, \mathsf{x}, \mathsf{w}, c)\right] = 1$$

- **ϵ-Soundness.** For all $\lambda \in \mathbb{N}$, all $\mathsf{x} \notin \mathcal{L}_{\mathsf{lpar}}$, and all PPT adversaries $\mathcal{A}$

$$\Pr\left[a = b \ : \ \mathcal{R}_{\mathsf{lpar}} \leftarrow\!\!\$\ \mathcal{RG}_\lambda; (c, b) \leftarrow \mathsf{Chall}(\mathsf{lpar}, \mathsf{x}); a \leftarrow \mathcal{A}(\mathsf{lpar}, \mathsf{x}, c)\right] \approx_\lambda \epsilon$$

We call a PA sound if $\epsilon \in \mathsf{negl}(\lambda)$. A PA is secure if it is complete and sound. Furthermore, we say that a PA is *zero-knowledge* (ZK-PA) if there exists a PPT algorithm Sim that computes the predicted answer of any valid statement x without knowing the random coins used in $\mathsf{Chall}()$ nor any witness for x, but only knowing the challenge c. In the case of ZK in the CRS model, the algorithm takes in also a CRS trapdoor τ which is generated by a setup algorithm $(\mathsf{crs}_\tau, \tau) \leftarrow \mathsf{setup}(1^\lambda)$. For notational simplicity, we assume that in this case lpar contains crs_τ as well.

[3] We define PAs as one-round protocols. As shown in [FNV17], this is without loss of generality as every ρ-round PA can be squeezed into a one-round PA.

```
Zero-Knowledge
$(\mathsf{crs}_\tau, \tau) \leftarrow \mathsf{setup}(1^\lambda); (\mathsf{x}, \mathsf{w}, c) \leftarrow \mathcal{A}(\mathsf{crs}_\tau, \tau);$
$\textbf{if } \mathcal{R}_{\mathsf{lpar}}(\mathsf{x}, \mathsf{w}) = 0, \textbf{ then return } 0;$
$b \leftarrow\!\!\$ \{0,1\}; \textbf{if } b = 0 \textbf{ then } a \leftarrow \mathsf{Resp}(\mathsf{lpar}, \mathsf{x}, \mathsf{w}, c); \textbf{else } a \leftarrow \mathsf{Sim}(\mathsf{lpar}, \mathsf{x}, \tau, c);$
$b' \leftarrow \mathcal{A}(a);$
$\textbf{return } b = b';$
```

Fig. 1: Experiment for the definition of Zero-knowledge

- **Zero-Knowledge.** A predictable argument Π is zero-knowledge if there exists a PPT simulator Sim such that for all PPT adversary $\mathcal{A}$, $\Pr[\mathsf{Exp}^{\mathsf{zk}}_{\Pi,\mathsf{Sim}}(\mathcal{A}, \lambda) = 1] \approx_\lambda \frac{1}{2}$, where $\mathsf{Exp}^{\mathsf{zk}}_{\Pi,\mathsf{Sim}}(\mathcal{A}, \lambda)$ is depicted in fig. 1.

In this work, we also consider a weaker version of zero-knowledge, called *witness indistinguishability* (WI) [FS90] which informally states that the adversarial verifier cannot identify which witnesses are held by the prover.

- **Witness-Indistinguishability.** A predictable argument Π is statistically witness indistinguishable if for any adversary $\mathcal{A}$, for any common statement x, for any witnesses $\mathsf{w}_1, \mathsf{w}_2$ such that $(\mathsf{x}, \mathsf{w}_1) \in \mathcal{R}_{\mathsf{lpar}}, (\mathsf{x}, \mathsf{w}_2) \in \mathcal{R}_{\mathsf{lpar}}$, the following holds:

$$\langle \mathsf{P}(\mathsf{lpar}, \mathsf{x}, \mathsf{w}_1), \mathcal{A}(\mathsf{lpar}, \mathsf{x}) \rangle \approx_\lambda \langle \mathsf{P}(\mathsf{lpar}, \mathsf{x}, \mathsf{w}_2), \mathcal{A}(\mathsf{lpar}, \mathsf{x}) \rangle$$

2.0.5 Oblivious Transfer. A 2-round oblivious transfer (OT) is a protocol between a receiver and a sender and consists of three polynomial-time algorithms $\Pi_{\mathsf{OT}} = (\Pi^R_{\mathsf{OT}}, \Pi^S_{\mathsf{OT}}, \Pi^O_{\mathsf{OT}})$ defined as follows:

First round. The receiver generates the first message $m^R \leftarrow \Pi^R_{\mathsf{OT}}(b; r^R)$ for the selection bit $b \in \{0,1\}$ and random tape $r^R \in \{0,1\}^{\mathsf{poly}(\lambda)}$.

Second round. For the input messages (x^0, x^1), where $x^l \in \{0,1\}^{\mathsf{poly}(\lambda)}$ for $l \in \{0,1\}$, the sender generates the second message $m^S \leftarrow \Pi^S_{\mathsf{OT}}(m^R, (x^0, x^1); r^S)$ using random tape $r^S \in \{0,1\}^{\mathsf{poly}(\lambda)}$.

Output. The receiver computes the output $x = \Pi^O_{\mathsf{OT}}(m^S, b, r^R)$.

In this work, we are interested in OT protocols that are *correct* and securely implement the ideal functionality in fig. 2 in the presence of malicious adversaries. Moreover, we require an additional property called *sender-extractability* in [GKPS18], which at a high-level means that the randomness of the sender is sufficient to reconstruct its input. The formal definition of this property can be found in appendix A.1.

Choose. On input $(\texttt{receive}, \mathrm{sid}, b)$ from R , where $b \in \{0,1\}$, if no messages of the form $(\texttt{receive}, \mathrm{sid}, b)$ is stored, store $(\texttt{receive}, \mathrm{sid}, b)$ and send $(\texttt{receive}, \mathrm{sid})$ to S.

Transfer. On input $(\texttt{send}, \mathrm{sid}, x^0, x^1)$ from S, with $x^0, x^1 \in \{0,1\}^k$, if no messages of the form $(\texttt{send}, \mathrm{sid}, x^0, x^1)$ is stored and a message of the form $(\texttt{receive}, \mathrm{sid}, b)$ is present, send $(\texttt{sent}, \mathrm{sid}, x^b)$ to R.

Fig. 2: The ideal functionality $\mathcal{F}_{\mathsf{OT}}$ for oblivious transfer

2.0.6 Garbled Circuits. We recall the definition of garbling schemes formalized in [BHR12]. At a high-level, a garbling scheme consists of four algorithms $\mathsf{GC} = (\mathsf{Garble}, \mathsf{Encode}, \mathsf{Eval}, \mathsf{Decode})$ defined as follows: Garble takes a circuit C and outputs a garbled circuit $\mathbf{C}$, encoding information e, and decoding information d. Encode takes an input e and x, and outputs a garbled input X. Eval takes as input a garbled circuit $\mathbf{C}$, and a garbled input X and outputs a garbled output Y. Finally, Decode takes d and a garbled output Y, and outputs a plain output y. In this work, we also assume an extra verification algorithm Verify that takes $(C, \mathbf{C}, e)$ as input and outputs 1 if this triple is valid.

A garbling scheme GC should satisfy *correctness* and the following security properties: *authenticity* which informally captures the unforgeability of the output of a garbled circuit evaluations, and *verifiability* that ensures the existence of an algorithm Verify that takes a circuit C, a (possibly maliciously generated) garbled circuit $\mathbf{C}$, and encoding information e, and outputs 1 if $\mathbf{C}$ is a valid garbling of C. The formal definition of these properties can be found in appendix A.2.

3 More efficient ZK-PA

PAs have deterministic provers and hence by an impossibility result from Goldreich and Oren [GO94] cannot be zero-knowledge in the plain model for non-trivial languages. Faonio *et al.* [FNV17] circumvented this impossibility and provided two constructions by using setup assumptions. Their first construction in the CRS model is based on the natural idea of adding a NIZK proof of knowledge π for the "well-formedness" of the challenge generated by the challenger. Although this gives a generic compiler for constructing ZK-PAs from PAs, here we investigate designing out-of-the-box ZK-PA protocols with concrete efficiency. We give a construction in the CRS model which is based on the notion of Trapdoor Smooth Projective Hash Functions (TSPHFs).

3.1 TSPHF-based ZK-PAs in the CRS model

As shown in [FNV17], PAs can be constructed from SPHFs, but since the projection key in SPHFs can be generated in a malicious way, they can provide

- $\mathsf{Setup}(1^\lambda)$: Run $(\mathsf{crs}_\tau, \tau) \leftarrow \mathsf{tsetup}(1^\lambda)$ and return $(\mathsf{crs}_\tau, \tau)$.
- $\mathsf{Chall}(\mathsf{lpar}, \mathsf{x})$:
  - Run $\mathsf{hk} \leftarrow\!\!\$\ \mathsf{hashkg}(\mathsf{lpar})$ and $\mathsf{hp} \leftarrow \mathsf{projkg}(\mathsf{lpar}; \mathsf{hk}, \mathsf{x})$.
  - Compute $\mathsf{H} \leftarrow \mathsf{hash}(\mathsf{lpar}; \mathsf{hk}, \mathsf{x})$.
  - Return $(c, b) := (\mathsf{hp}, \mathsf{H})$.
- $\mathsf{Resp}(\mathsf{lpar}, \mathsf{x}, \mathsf{w}, c)$: For $c := \mathsf{hp}$, check if $\mathsf{verHP}(\mathsf{crs}_\tau, \mathsf{hp}) = 1$, then run $\mathsf{pH} \leftarrow \mathsf{projhash}(\mathsf{lpar}; \mathsf{hp}, \mathsf{x}, \mathsf{w})$ and return $a := \mathsf{pH}$.
- $\mathsf{Sim}(\mathsf{lpar}, \mathsf{x}, \tau, c)$: Parse $c := \mathsf{hp}$ and return $\mathsf{tH} \leftarrow \mathsf{thash}(\mathsf{lpar}; \mathsf{hp}, \mathsf{x}, \tau)$.

Fig. 3: ZK-PA $\mathit{\Pi}_{\mathsf{zkpa}}$ from TSPHFs.

only honest-verifier zero-knowledge property and it is not clear how to construct ZK-PA from standard SPHFs directly. Benhamouda *et al.* [BBC$^+$13] defined the notion of *trapdoor SPHFs* (TSPHFs) as an extension of SPHF in which one can verify the correctness of the projection key generation. More in details, a TSPHF comes with three additional algorithms $(\mathsf{tsetup}, \mathsf{verHP}, \mathsf{thash})$. tsetup outputs a CRS crs_τ with a trapdoor τ. The trapdoor τ can be used by thash to compute the hash value of any statement x (only by knowing public hp). The algorithm verHP takes in a key hp and the CRS crs_τ, and outputs 1 if hp is a valid projection key. The properties a TSPHF must verify are the same as SPHF, except the smoothness property is no longer statistical but computational as hp should now contain enough information to compute the hash of any statement. Moreover, a TSPHF should satisfy zero-knowledge property which informally states that for any statement x with valid witness w, the projected hash value $\mathsf{pH} \leftarrow \mathsf{projhash}(\mathsf{lpar}; \mathsf{hp}, \mathsf{x}, \mathsf{w})$ should be indistinguishable from the trapdoor hash value $\mathsf{tH} \leftarrow \mathsf{thash}(\mathsf{lpar}; \mathsf{hp}, \mathsf{x}, \tau)$. For a more formal definition of TSPHFs, We refer the reader to [BBC$^+$13].

In this section, we show the connection between ZK-PAs and TSPHFs [BP13], namely we construct ZK-PA for a relation $\mathcal{R}_{\mathsf{lpar}}$ given a TSPHF for the same relation. Different from [FNV17], the relation $\mathcal{R}_{\mathsf{lpar}}$ here is identical. This is because [FNV17] considers the connection for the knowledge-sound PAs (and extractable SPHFs) whereas here we only consider soundness and (computational) smoothness. As a direct result of this, we obtain ZK-PA for all languages that admit TSPHFs (i.e., *algebraic languages*).

3.1.1 Construction of ZK-PA from TSPHFs. We are now ready to present our construction of ZK-PAs from TSPHFs. Let $\mathit{\Pi}_{\mathsf{tsphf}} = (\mathsf{setup}, \mathsf{tsetup}, \mathsf{hashkg}, \mathsf{projkg}, \mathsf{hash}, \mathsf{projhash}, \mathsf{verHP}, \mathsf{thash})$ be a TSPHF for $\mathcal{L}_{\mathsf{lpar}}$. The construction of $\mathit{\Pi}_{\mathsf{zkpa}} = (\mathsf{Setup}, \mathsf{Chall}, \mathsf{Resp}, \mathsf{Sim})$ in the CRS model is given in fig. 3.

Theorem 1. *If the TSPHF $\mathit{\Pi}_{\mathsf{tsphf}}$ is correct, (computationally) smooth and zero-knowledge, then $\mathit{\Pi}_{\mathsf{zkpa}}$ in fig. 3 is secure and zero-knowledge.*

Proof. The correctness of Π_{zkpa} is trivial and follows directly from the correctness of the TSPHF. Here we only give a sketch of the proofs for soundness and ZK.
(Soundness). To show soundness, let $\mathcal{A}$ be a PPT adversary that breaks the soundness, i.e., $\mathcal{A}$ outputs the predicted answer for a chosen $\mathsf{x} \notin \mathcal{L}_{\mathsf{lpar}}$ with non-negligible probability. We construct a PPT algorithm $\mathcal{B}$ that breaks the smoothness of the underlying TSPHF. Given $(\mathsf{hp}, \mathsf{H})$ as input, $\mathcal{B}$ proceeds as follows: it first runs $\mathcal{A}$ on the security parameters to receive x. Next, it runs $\mathcal{A}$ on input $(\mathsf{x}, \mathsf{hp})$ and receives the answer a. Now depending on whether $a = \mathsf{H}$ or not, $\mathcal{B}$ can decide if H is a hash value or random. This breaks the smoothness property.
(ZK). The ZK property can be shown by a straightforward reduction to the ZK property of the underlying TSPHF. To do so, let $\mathcal{A}$ be an efficient adversary against the ZK property of Π_{zkpa}. We construct an efficient algorithm $\mathcal{B}$ against the ZK property of Π_{tsphf} as follows: $\mathcal{B}$ runs $(\mathsf{crs}_\tau, \tau) \leftarrow \mathsf{setup}(1^\lambda)$ and sends $(\mathsf{crs}_\tau, \tau)$ to $\mathcal{A}$. Upon receiving $(\mathsf{x}, \mathsf{w}, c)$ from $\mathcal{A}$, $\mathcal{B}$ sends the same tuple $(\mathsf{x}, \mathsf{w}, c)$ to the challenger. Given the challenge a, $\mathcal{B}$ finally runs $\mathcal{A}(a)$ and return the same guess as $\mathcal{A}$. It is easy to see that $\mathcal{B}$ can now distinguish a real answer from a simulated one with the same advantage as $\mathcal{A}$'s advantage. This completes the proof.

Instantiation and Efficiency Evaluation. Given the above connection, one can now obtain a secure ZK-PA for any algebraic language $\mathcal{L}_{\mathsf{lpar}}$ with $\mathsf{lpar} = (\mathbf{M}, \boldsymbol{\theta})$ (see eq. (1)) in the bilinear setting based on the efficient construction of TSPHF in [BP13]. For the sake of completeness, we provide the construction in fig. 8 in appendix B. The resulting ZK-PA is sound under the DDH assumption in $\mathbb{G}_2$ (See [BP13], Appendix E.3 for the security proof). To evaluate efficiency, we note that compared to the original construction of ZK-PA in [FNV17], the above construction is more efficient as it only has one more group element in the challenge c (compared to the non-zk construction of PA), whereas the idea of adding a NIZK proof for the well-formedness of c in [FNV17] has at least a linear overhead in the size of c [4].

Remark 1 (Non-Blackbox Construction in the plain model). Recently, Abdolmaleki et al. [AKL21] show how one can use non-blackbox techniques to construct a subversion-resistant variant of smooth projective hash functions. Following a similar approach directly yields the construction of ZK-PA in the plain model, thus giving another way to circumvent the [GO94] impossibility using non-blackbox techniques. The key idea is to rely on the existence of an efficient (non-blackbox) extractor that—after checking the well-formedness of c—can extract a function of the verifier's randomness (e.g., $[\boldsymbol{\alpha}]_2$) by which one can efficiently compute the predictable answer b.

Remark 2. In their recent work, Bitansky and Choudhuri [BC20] also construct ZK-PA for all NP. Their construction, however, mainly focuses on a *feasibility*

[4] Here we are assuming that the security of the construction should remain under standard and falsifiable assumptions as it is easy to construct succinct NIZKs based on non-falsifiable assumptions.

result rather than efficiency, and requires strong assumptions such as indistinguishability obfuscation. Moreover, while the zero-knowledge simulator in their construction is non-black-box which is inherent in the plain model, we rather focus on more efficient constructions in the CRS model.

4 Witness-Indistinguishable Predictable Arguments

Due to a classical impossibility result [GO94], a prerequisite for constructing 2-message ZK proof systems based on black-box techniques is a common reference string (CRS)—a string generated by a trusted party to which both prover and verifier have access. Requiring such a trust model may however be overkill for some applications where a weaker notion of privacy such as witness indistinguishability (WI) is sufficient. Weaker than ZK property, this property states that for any two possible witnesses $\mathsf{w}_1, \mathsf{w}_2$, an adversary cannot distinguish proofs generated by w_1 from the proofs generated by w_2. Given a PA $\Pi_{\mathsf{pa}} = (\mathsf{Chall}, \mathsf{Resp})$ for an NP language $\mathcal{L}$, we show how to construct a WI-PA $\Pi_{\mathsf{wipa}} = (\mathsf{Chall}', \mathsf{Resp}')$ for the same language. At first it may seem that regardless of which witness is used by the prover when running Resp, it has the same functionality since all the witnesses return the same (predicted) answer. This argument is however not true: while for an honestly-generate challenge, Resp behaves the same regardless of which valid witness is used, this might not be true for maliciously generated challenges. To circumvent this issue, the key idea is to require the verifier to prove that the challenge is indeed generated from a proper run of Chall with some randomness. This should be done without breaking the soundness, meaning the secret coins of the verifier should be kept hidden from the prover. To this end, we will use a NIWI proof system as an ingredient, through which the verifier proves the following statement: there exists a random string α, such that $c = \mathsf{Chall}(\mathsf{lpar}, \mathsf{x}; \alpha)$. The prover first checks if the NIWI proof verifies and if so, computes the predicted answer as before.

Since we use a NIWI proof system in the plain model as an ingredient of our construction, below we recall the definition of WI for such proof systems. We note that a construction of NIWI in the plain model for all NP languages and based on standard assumptions is presented in [GOS06].

Definition 3. *Let $\Pi_{\mathsf{niwi}} = (\mathcal{P}_{\mathsf{niwi}}, \mathcal{V}_{\mathsf{niwi}})$ be a non-interactive proof system for a language $\mathcal{L}_{\mathsf{lpar}}$. We say that Π_{niwi} is computationally witness-indistinguishable if for all $(\mathsf{x}, \mathsf{w}_1, \mathsf{w}_2)$ such that $(\mathsf{x}, \mathsf{w}_1) \in \mathcal{R}_{\mathsf{lpar}}$ and $(\mathsf{x}, \mathsf{w}_2) \in \mathcal{R}_{\mathsf{lpar}}$, and for all PPT adversaries $\mathcal{A}$,*

$$\Pr\left[\mathcal{A}(\pi) = 1 \ : \ \pi \leftarrow \mathcal{P}_{\mathsf{niwi}}(\mathsf{lpar}, \mathsf{x}, \mathsf{w}_1)\right] \approx_\lambda \Pr\left[\mathcal{A}(\pi) = 1 \ : \ \pi \leftarrow \mathcal{P}_{\mathsf{niwi}}(\mathsf{lpar}, \mathsf{x}, \mathsf{w}_2)\right]$$

4.1 Our Construction

Let $\Pi_{\mathsf{pa}} = (\mathsf{Chall}, \mathsf{Resp})$ be a predictable argument for language $\mathcal{L}_{\mathsf{lpar}}$, and Π_{niwi} be a non-interactive computational WI proof system in the plain model for the language of statements c for which there exists α such that $c = \mathsf{Chall}(\mathsf{lpar}, \mathsf{x}; \alpha)$.

We construct a WI-PA $\mathit{\Pi}_{\mathsf{wipa}} = (\mathsf{Chall}', \mathsf{Resp}')$ for $\mathcal{L}_{\mathsf{lpar}}$ as depicted in fig. 4. The completeness of the construction follows straightforwardly from the completeness of $\mathit{\Pi}_{\mathsf{pa}}$. We prove soundness and WI in the next theorem.

- $\mathsf{Chall}'(\mathsf{lpar}, \mathsf{x})$: the verifier computes $(c, b) \leftarrow \mathsf{Chall}(\mathsf{lpar}, \mathsf{x}; \alpha)$ and sends the challenge c along with a NIWI proof π for the existence of α such that c is the first output of $\mathsf{Chall}(\mathsf{lpar}, \mathsf{x}; \alpha)$.
- $\mathsf{Resp}'(\mathsf{lpar}, \mathsf{x}, \mathsf{w}, c, \pi)$: the prover first checks the NIWI proof π. If π verifies, the prover computes $a \leftarrow \mathsf{Resp}(\mathsf{lpar}, \mathsf{x}, \mathsf{w}, c)$ and returns a.

Fig. 4: Construction of WI-PA

Theorem 2. *The construction in fig. 4 is a statistical witness-indistinguishable predictable argument in the plain model.*

Proof. **Soundness.** Let $\mathsf{x} \notin \mathcal{L}_{\mathsf{lpar}}$ and $\mathcal{A}$ be an efficient adversary that breaks soundness of $\mathit{\Pi}_{\mathsf{wipa}}$ by convincing the honest verifier V with non-negligible probability ε. I.e., $\varepsilon(n) \geq \frac{1}{p(n)}$ for some polynomial p and for infinitely many n's. Denoting this set by N, we restrict ourselves to $n \in N$ from now on. This indicates that there exists a first message c from V on which $\mathcal{A}$ convinces V with probability at least ε. Fix this challenge c and the corresponding answer b computed by V. Define a set S as follows:

$$S = \Big\{ b : \Pr\big[\mathcal{A}(\mathsf{lpar}, \mathsf{x}, c) = b | (c, b) \leftarrow \mathsf{Chall}(\mathsf{lpar}, \mathsf{x})\big] \geq \frac{\varepsilon}{2} \Big\}$$

Fix some $b_0 \in S$ and define $S_0 \subseteq S$ as

$$S_0 = \Big\{ b \in S : \Pr\big[\mathcal{A}(\mathsf{lpar}, \mathsf{x}, c) = b | (c, b_0) \leftarrow \mathsf{Chall}(\mathsf{lpar}, \mathsf{x})\big] \geq \frac{\varepsilon}{4} \Big\}.$$

Since $b_0 \in S$, we have that $\Pr\big[\mathcal{A}(\mathsf{lpar}, \mathsf{x}, c) = b_0 | (c, b_0) \leftarrow \mathsf{Chall}(\mathsf{lpar}, \mathsf{x})\big] \geq \frac{\varepsilon}{2}$ and therefore $|S_0| \cdot \frac{\varepsilon}{4} \leq 1 - \frac{\varepsilon}{2}$, which consequently implies that $|S_0| \leq \frac{4}{\varepsilon}$. Now, the fact that ε is non-negligible indicates that S_0 is bounded by a polynomial. On the other hand, we have that $\Pr[b \in S] \geq \frac{\varepsilon}{2}$, and that S is exponential in the security parameter λ. This means that there should exist $b_1 \in S$ such that $b_1 \notin S_0$. We now construct a non-uniform PPT adversary $\mathcal{B}$ that breaks the witness-indistinguishability of $\mathit{\Pi}_{\mathsf{niwi}}$. Let $aux = (\alpha_0, \alpha_1, b_0, b_1)$ be such that $(c, b_0) \leftarrow \mathsf{Chall}(\mathsf{lpar}, \mathsf{x}; \alpha_0)$ and $(c, b_1) \leftarrow \mathsf{Chall}(\mathsf{lpar}, \mathsf{x}; \alpha_1)$. Given aux as advice, $\mathcal{B}$ proceeds as follows: it first returns $(c, (b_0, \alpha_0), (b_1, \alpha_1))$ to the WI challenger and obtains a proof π. Next, $\mathcal{B}$ calls $\mathcal{A}$ on input (π, c) and returns i when it receives b_i from $\mathcal{A}$. Note that for π that is computed using (r_0, b_0), $\mathcal{A}$ returns b_1 with probability at most $\frac{\varepsilon}{4}$, whereas for π computed by (r_1, b_1), $\mathcal{A}$ returns b_1 with

probability at least $\frac{\varepsilon}{2}$. This makes $\mathcal{B}$ a successful adversary in breaking WI.
WI. Let V^* be an adversary against WI property of Π_{wipa} and $(\mathsf{x}, \mathsf{w}_1, \mathsf{w}_2)$ be such that $(\mathsf{x}, \mathsf{w}_1), (\mathsf{x}, \mathsf{w}_2) \in \mathcal{R}_{\mathsf{lpar}}$. It follows from (statistical) soundness of the NIWI proof that V^*'s first message is computed correctly with overwhelming probability. This together with predictability of the argument indicates that the answer from the prover is unique regardless of which witness is used and thus completes the proof. □

5 Commit-and-Prove Predictable Arguments

We study a relaxed notion of predictability in interactive argument systems which consists of two phases: In phase 1 (commitment phase), the prover commits to its witness once for all and sends the commitment to the verifier. In phase 2 (challenge-response phase), the prover and the verifier engage in a predictable argument protocol, where the verifier's challenges may depend on the commitment in such a way that the prover's responses can be predicted by the verifier. The type of relations we consider are of the following form: a statement $\mathsf{x} = (\mathsf{cm}, C, y)$ and a witness (w, d) are in the relation (i.e., $(\mathsf{x}, (\mathsf{w}, \mathsf{d})) \in \mathcal{R}$) iff "$\mathsf{cm}$ commits to w by randomness d, and $C(\mathsf{w}) = y$". Here C is a circuit in some polynomial-size circuit class $\mathcal{C}$ and y is the expected output of the circuit.

Definition 4 (Commit-and-Prove Predictable Arguments). *Let $\mathcal{C}$ be a class of polynomial-sized circuits. A commit-and-prove predictable argument for $\mathcal{C}$ is a multi-round protocol (between a prover P and a verifier V) which consists of three algorithms $\Pi_{\mathsf{cppa}} = (\mathsf{Commit}, \mathsf{Chall}, \mathsf{Resp})$:*

Commitment phase *(**executed by** P): $\mathsf{cm} \leftarrow \mathsf{Commit}(\mathsf{w}; \mathsf{d})$ on input a value w, generates a commitment cm by using some randomness d.*
Interaction phase. *Each round proceeds as follows:*
- *(**Executed by** V): $(c, b) \leftarrow \mathsf{Chall}(\mathsf{cm}, C, y)$ on input a statement (cm, C, y) such that $C \in \mathcal{C}$, generates a challenge c and a predicted answer b.*
- *(**Executed by** P): $a \leftarrow \mathsf{Resp}(\mathsf{cm}, C, \mathsf{w}, \mathsf{d}, c)$ on input a commitment cm, a circuit $C \in \mathcal{C}$, the committed value w, the randomness d, returns a response a.*

V accepts the proof iff $a = b$ in all rounds.

We call a CPPA as a ρ-round CPPA if the interaction phase consists of ρ rounds. A CPPA should satisfy *completeness* and *soundness* as defined below:

(Perfect) Completeness. An honest prover with a statement $\mathsf{x} = (\mathsf{cm}, C, y)$ and witness (w, d) such that (w, d) opens the commitment (i.e., $\mathsf{cm} = \mathsf{Commit}(\mathsf{w}; \mathsf{d})$), and $C(\mathsf{w}) = y$ can always convince the verifier with overwhelming probability. More precisely, a CPPA has perfect completeness if for all $\lambda \in \mathbb{N}$, for all $C \in \mathcal{C}$, and for all $(\mathsf{x} = (\mathsf{cm}, C, y), (\mathsf{w}, \mathsf{d})) \in \mathcal{R}$

$$\Pr\left[a = b \;:\; (c, b) \leftarrow \mathsf{Chall}(\mathsf{cm}, C, y); a \leftarrow \mathsf{Resp}(\mathsf{cm}, C, \mathsf{w}, \mathsf{d}, c)\right] = 1$$

ϵ-Soundness. For all $\lambda \in \mathbb{N}$, and all (stateful) PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$

$$\Pr\left[\begin{matrix} a = b \wedge \\ C(\mathsf{w}) \neq y \end{matrix} : \begin{matrix} (\mathsf{w}, \mathsf{d}, C, y) \leftarrow \mathcal{A}_1(1^\lambda); \mathsf{cm} \leftarrow \mathsf{Commit}(\mathsf{w}; \mathsf{d}) \\ (c, b) \leftarrow \mathsf{Chall}(\mathsf{cm}, C, y); a \leftarrow \mathcal{A}_2(\mathsf{w}, \mathsf{d}, C, y, c) \end{matrix}\right] \approx_\lambda \epsilon$$

We call a CPPA sound if $\epsilon \in \mathsf{negl}(\lambda)$. A CPPA is secure if it is correct and sound. Similar to PAs, one can show that CPPAs can also be made extremely laconic in terms of both round complexity and proof complexity. Specifically, the same technique in [FNV17] can be used to collapse any ρ-round CPPA into a single round CPPA.

In this work, we only focus on CPPA protocols with the zero-knowledge property. A CPPA is *zero-knowledge* (ZK-CPPA) if there exists a PPT algorithm Sim that computes the predicted answer of any valid statement x without knowing the random coins used by $\mathsf{Chall}()$ nor any witness for x, but only knowing the challenge c. Since our construction of ZK-CPPA is in the non-programmable random oracle (NPRO) model, we define this property in this model.

Definition 5 (Zero-knowledge CPPA in the NPRO model). *We say that a CPPA* ($\mathsf{Commit}, \mathsf{Chall}, \mathsf{Resp}$) *for a class of circuits $\mathcal{C}$ satisfies the* zero-knowledge *property in the NPRO model if for any PPT adversary $\mathcal{A}$, there exists a PPT simulator* Sim *such that for all PPT distinguisher $\mathcal{D}$, for all* $(\mathsf{x}, \mathsf{w}) \in \mathcal{R}$, *and all auxiliary inputs $z \in \{0,1\}^*$, we have:*

$$\max_{\mathcal{D}, z} \Big| \Pr[\mathcal{D}^H(\mathsf{x}, \tau, z) = 1 : \tau \leftarrow (\mathsf{P}^H(\mathsf{x}, \mathsf{w}) \leftrightarrows \mathcal{A}^H(\mathsf{x}, z))] \\ - \Pr[\mathcal{D}^H(\mathsf{x}, \tau, z) = 1 : \tau \leftarrow \mathsf{Sim}^H(\mathsf{x}, z)] \Big| \leq \mathsf{negl}(|\mathsf{x}|)$$

Where P *and* $\mathcal{A}$ *are respectively the prover and the (malicious) verifier running the CPPA protocol, and* $\mathsf{P}^H(\mathsf{x}, \mathsf{w}) \leftrightarrows \mathcal{A}^H(\mathsf{x}, z)$ *denotes the random variable corresponding to a protocol transcript on input* (x, w).

We now give our construction of ZK-CPPA for all polynomial-size circuits P in the NPRO model. The construction is similar to the three-round ZK protocol of [GKPS18], with the difference that the first message in our protocol is reusable. Moreover, here we only focus on providing ZK property as defined above, whereas the construction of [GKPS18] shows ZK in the UC model.

5.1 ZK-CPPA based on garbled circuits and oblivious transfer

Let $\mathsf{GC} = (\mathsf{Garble}, \mathsf{Encode}, \mathsf{Eval}, \mathsf{Decode}, \mathsf{Verify})$ be a garbled circuit with correctness, authenticity, and verifiability, and $\Pi_{\mathsf{OT}} = (\Pi^R_{\mathsf{OT}}, \Pi^S_{\mathsf{OT}}, \Pi^O_{\mathsf{OT}})$ be a sender-extractable oblivious transfer protocol that realizes $\mathcal{F}_{\mathsf{OT}}$. At a high level, the construction proceeds as follows. The prover P with witness $\mathsf{w} = (\mathsf{w}_1, \ldots, \mathsf{w}_n) \in \{0,1\}^n$ plays the role of the receiver in n instances of the OT protocol and commits to its witness bits by providing w_j as input to the j-th instance of Π_{OT}. Let $m^R_j \leftarrow \Pi^R_{\mathsf{OT}}(\mathsf{w}_j; r^R_j)$ and define cm and d as the set of $\{m^R_j\}_{j \in [n]}$ and $\{r^R_j\}_{j \in [n]}$, respectively. For a circuit-value pair (C, y) of the verifier's choice, let $\hat{C}$ be a circuit that realizes the following relation $\mathcal{R}$: $\mathcal{R}(\mathsf{x} = (\mathsf{cm}, C, y), (\mathsf{w}, \mathsf{d})) = 1$ iff (w, d) open cm and $C(\mathsf{w}) = y$. The verifier V constructs a GC $\mathbf{C}$ for $\hat{C}$ and sends it along with the second message of the OT as the challenge c. Moreover, V sets the predicted answer b to be the output 1-key k^1 of the final gate in the circuit. Now, P with a valid witness (w, d) evaluates $\mathbf{C}$ and sends the obtained garbled output $a = k^1$ as the predicted answer. It is not hard to see that this construction results in a CPPA. To additionally ensure ZK property, we follow the same approach as [GKPS18] by enforcing V to also provide a ciphertext $ct = H(\mathsf{k}^1) \oplus r$, where H is a random oracle and r is the randomness used by V to produce the second message of the OT. When P computes k^1, she first recovers r and then computes all the labels by executing the extractor Ext guaranteed by the sender-extractability property. Finally, P verifies if the garbled circuit has been constructed correctly and if so, she sends the predicted answer $a = \mathsf{k}^1$ to V. The resulting protocol Π_{cppa} is described in fig. 5. The proof idea is similar in spirit to the proof of Theorem 4.2 in [GKPS18]. We give a proof sketch here.

Theorem 3. *Let GC be a correct, authentic, and verifiable garbling scheme, Π_{OT} be a sender-extractable OT protocol that securely implements $\mathcal{F}_{\mathsf{OT}}$, and H be a random oracle. The protocol Π_{cppa} in fig. 5 is a secure and zero-knowledge commit-and-prove predictable argument as defined in definitions 4 and 5.*

Proof (Sketch). Completeness follows straightforwardly by the correctness property of the underlying OT and the garbling scheme.

In order to show soundness, let us consider a PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and assume that $(\mathsf{w}, \mathsf{d}, C, y)$ is a tuple returned by $\mathcal{A}_1$ that corresponds to a false statement. That is, $\mathsf{x} = (\mathsf{cm}, C, y)$, where $\mathsf{cm} = \mathsf{Commit}(\mathsf{w}; \mathsf{d})$ and $C(\mathsf{w}) \neq y$. We show that for $(c, b) \leftarrow \mathsf{Chall}(\mathsf{cm}, C, y)$, if $\mathcal{A}_2$ having c can compute the predicted answer b, then one can either break the sender security of the underlying OT protocol, or the authenticity of the garbling scheme. To show this reduction, we first note that b is the correct label k^1. Now, given that $C(\mathsf{w}) \neq y$, there can be two cases where $\mathcal{A}_2$ can output k^1 with non-negligible probability. In the first case, $\mathcal{A}_2$ outputs k^1 by the ability of computing invalid labels $k_j^{1-\mathsf{w}_j}$ that does not correspond to its committed value. It is not hard to see that such $\mathcal{A}_2$ can be used to break OT sender security. The reduction $\mathcal{B}$ proceeds as follows: $\mathcal{B}$ first computes a garbled circuit $\mathbf{C}$ and sends the labels to the OT challenger. Next, it extracts $\mathcal{A}_2$'s input w and forwards it as the choice bits of the receiver. The OT challenger computes the sender's message either by invoking a real sender, or by invoking the simulator, and sends it to the reduction who further forwards to $\mathcal{A}_2$ together with $\mathbf{C}$ and a random T. Now, since $\mathcal{A}_2$ can compute k^1 only in the real execution of Π_{OT}, a successful $\mathcal{A}_2$ with non-negligible probability ϵ implies that $\mathcal{B}$ can distinguish the real and simulated view of the OT protocol with probability at least ϵ. In the second case, where $\mathcal{A}_2$ does not use invalid labels but computes the correct k^1, it is straightforward to construct an adversary $\mathcal{B}$ that breaks the authenticity of the underlying garbling scheme by forging k^1 for a given garbled circuit $\mathbf{C}$.

We now argue that Π_{cppa} is zero-knowledge in the NPRO model. Let V^* be a PPT adversary against the ZK property. We construct an efficient simulator Sim that simulates the protocol as follows. Sim observes V^*'s calls to the random oracle, so that for every query $H(u)$ made by V^*, Sim records u in a set L. To simulate the first message, Sim invokes the simulator of Π_{OT} for the corrupt receiver. Upon receiving V^*'s message c, Sim parses c as $(\mathbf{C}, \{m_j^S\}_{j\in[n]}, T)$ and defines the set $\tilde{R} = \{H(u) \oplus T | u \in L\}$. For any $r \in \tilde{R}$ parsed as $r = r_1||\dots||r_n$, Sim computes $(k_j^0, k_j^1) \leftarrow \mathsf{Ext}(m_j^R, m_j^S, r_j^S)$ for $j \in [n]$ and checks if $\mathsf{Verify}(\hat{C}, \mathbf{C}, \{k_j^0, k_j^1\}_{j\in[n]}) = 1$. If there exists such $r \in \tilde{R}$, the simulator sends Y to V^*, where $Y \in L$ is so that $r = H(Y) \oplus T$. Otherwise, Sim aborts the protocol. It should be clear that the output of the simulator is perfectly indistinguishable from the real distribution. This completes the proof. □

6 Applications: Witness Encryption with Decryptor Privacy

Besides being a notion of theoretical interest, we also show the applications of (commit-and-prove) predictable arguments with zero-knowledge or witness-indistinguishability property in the context of witness encryption. Witness encryption (WE) is a powerful notion of encryption introduced by Garg et al. [GGSW13]. A WE scheme for an NP relation $\mathcal{R}_{\mathsf{lpar}}$ allows to encrypt a message m with respect to a statement x as $ct \leftarrow \mathsf{WE.Enc}(\mathsf{lpar}, m, \mathsf{x})$. The ciphertext can be decrypted as $m \leftarrow \mathsf{WE.Dec}(ct, \mathsf{w})$ for any w such that $(\mathsf{x}, \mathsf{w}) \in \mathcal{R}_{\mathsf{lpar}}$. Security guarantees that no adversary should learn any non-trivial information about m if $\mathsf{x} \notin \mathcal{L}_{\mathsf{lpar}}$, where $\mathcal{L}_{\mathsf{lpar}}$ is the language corresponding to $\mathcal{R}_{\mathsf{lpar}}$. More formally, we say that a WE is secure if it is complete and sound as defined below:

- **Completeness.** A WE has completeness if for all $\lambda \in \mathbb{N}$, for all $\mathcal{R}_{\mathsf{lpar}} \in \mathcal{RG}_\lambda$, for all m, and for all $(\mathsf{x}, \mathsf{w}) \in \mathcal{R}_{\mathsf{lpar}}$

$$\Pr[\mathsf{Dec}(\mathsf{Enc}(\mathsf{lpar}, \mathsf{x}, m), \mathsf{w}) = m] \geq 1 - \mathsf{negl}(\lambda)$$

  If the probability is 1, we say WE is perfectly complete.
- **Soundness.** A WE has soundness if for all $\lambda \in \mathbb{N}$ and all PPT adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that for any m_0, m_1

$$\Pr\left[\begin{matrix}\mathcal{R}_{\mathsf{lpar}} \leftarrow\!\$\ \mathcal{RG}_\lambda; \mathsf{x} \leftarrow \mathcal{A}(\mathsf{lpar}); b \leftarrow\!\$\ \{0,1\};\\ ct \leftarrow \mathsf{Enc}(\mathsf{lpar}, \mathsf{x}, m_b); b' \leftarrow \mathcal{A}(\mathsf{lpar}, \mathsf{x}, ct)\end{matrix} : \begin{matrix} b = b' \wedge \mathsf{x} \notin \mathcal{L}_{\mathsf{lpar}} \\ \wedge |m_0| = |m_1|\end{matrix}\right] \approx_\lambda \mathsf{negl}(\lambda)$$

While being a very powerful notion, existing constructions of WE are not satisfactory, as they are either based on strong assumptions such as indistinguishability obfuscation and multilinear maps [GGH+13, GGSW13, GKW17, CVW18], or based on new and unexplored algebraic structures [BIJ+20].

As noted in [FNV17], predictable arguments imply witness encryption as one can encrypt a bit m by generating a challenge-answer pair (c, b) for the PA and

- **Oracles and Primitives**: A correct, authentic, and verifiable garbling scheme $\mathsf{GC} = (\mathsf{Garble}, \mathsf{Encode}, \mathsf{Eval}, \mathsf{Decode})$, a sender-extractable 2-round OT Π_{OT}, and a hash function $H : \{0,1\}^* \to \{0,1\}^{\mathsf{poly}(\lambda)}$ modeled as a random oracle.
- **P's private input**: $\mathsf{w} \in \{0,1\}^n$, where $n = \mathsf{poly}(\lambda)$.
- **Commitment Phase**: P plays the role of the receiver in n instances of Π_{OT} and computes $(\mathsf{cm}, \mathsf{d})$ as follows:
  1. Sample uniformly random r_j^R from $\{0,1\}^\lambda$, and compute $m_j^R \leftarrow \Pi_{\mathsf{OT}}^R(\mathsf{w}_j; r_j^R)$ for $j \in [n]$.
  2. Define $\mathsf{cm} = \{m_j^R\}_{j\in[n]}$ and $\mathsf{d} = \{r_j^R\}_{j\in[n]}$.
- **Common inputs**: A security parameter λ, and a statement $\mathsf{x} = (\mathsf{cm}, C, y)$, where C is a polynomial-size circuit.
- **Challenge**: Let $\hat{C}$ be a circuit that realizes the following relation $\mathcal{R}$: $\mathcal{R}(\mathsf{x} = (\mathsf{cm}, C, y), (\mathsf{w}, \mathsf{d})) = 1$ iff (w, d) opens cm and $C(\mathsf{w}) = y$. V plays the role of the sender in n instances of Π_{OT} and computes a pair (c, b) of challenge-predicted answer as follows:
  1. Compute $(\mathbf{C}, e, d) \leftarrow \mathsf{Garble}(1^\lambda, \hat{C})$, where $e := \{k_j^0, k_j^1\}_{j\in[n]}$, and $d := (k^0, k^1)$.
  2. For $j \in [n]$, sample uniformly random r_j^S from $\{0,1\}^\lambda$, and compute $m_j^S = \Pi_{\mathsf{OT}}^S(k_j^0, k_j^1, m_j^R; r_j^S)$.
  3. Compute $T = H(k^1) \oplus r^S$, where $r^S = r_1^S || \dots || r_n^S$.
  4. Define $c = (\mathbf{C}, \{m_j^S\}_{j\in[n]}, T)$ and $b = k^1$, and send c to P.
- **Response**: P proceeds as follows:
  1. Execute $k_j^{\mathsf{w}_j} = \Pi_{\mathsf{OT}}^O(m_j^S, \mathsf{w}_j, r_j^R)$ for $j \in [n]$.
  2. Execute $Y = \mathsf{Eval}(\mathbf{C}, \{k_j^{\mathsf{w}_j}\}_{j\in[n]})$.
  3. Recover $r^S = H(Y) \oplus T$, and parse $r^S = r_1^S || \dots || r_n^S$.
  4. Reconstruct sender's inputs $(k_j^0, k_j^1) \leftarrow \mathsf{Ext}(m_j^R, m_j^S, r_j^S)$ for $j \in [n]$. Abort if the extractor fails for some $j \in [n]$.
  5. Send the predicted answer $a = Y$ if $\mathsf{Verify}(\hat{C}, \mathbf{C}, \{k_j^0, k_j^1\}_{j\in[n]}) = 1$; and abort otherwise.
- V accepts the proof iff $a = b$.

Fig. 5: ZK-CPPA Π_{cppa} based on GC and OT

define the ciphertext as $(c, b \oplus m)$. Viceversa, a PA can be constructed from WE by encrypting a random bit m and then asking the prover to return m. Furthermore, it is not hard to show that commit-and-prove predictable arguments are also equivalent to a variant of witness encryption studied in [BL20, CDK+21]. It is therefore interesting to see the applications of predictable arguments with privacy in the context of witness encryption. While the standard definition of witness encryption requires the above properties, for some applications explained

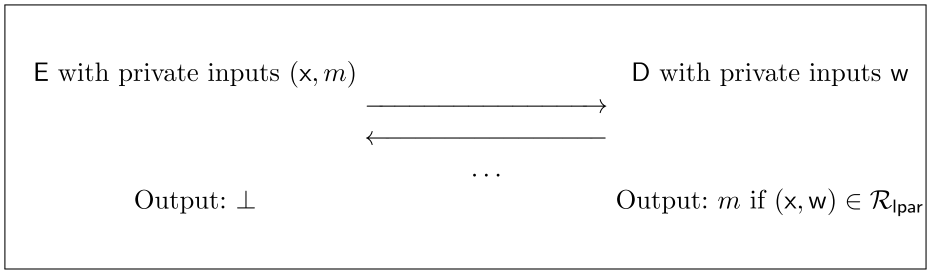
E with private inputs (x, m)
D with private inputs w
$\dots$
Output: $\perp$
Output: m if $(\mathsf{x}, \mathsf{w}) \in \mathcal{R}_{\mathsf{lpar}}$

Fig. 6: Functionality of a WE scheme with decryptor privacy for a relation $\mathcal{R}_{\mathsf{lpar}}$

below, we may require some level of privacy for the decryptor as well. In other words, we may ask for a WE scheme that mimics the following functionality (See fig. 6): the functionality is parameterized by a message space $\mathcal{M}$ and an NP relation $\mathcal{R}_{\mathsf{lpar}}$. An encryptor E with private inputs $m \in \mathcal{M}$ and bitstring x interacts with a decryptor D with private input w, at the end of which D outputs m iff $(\mathsf{x}, \mathsf{w}) \in \mathcal{R}$. Note that this is different from standard WE wherein the decryptor aims to obtain the message internally without revealing it to the environment. Here instead, the decrypted message is revealed to the encryptor which may break the privacy of the decryptor. Since the encryptor knows the plaintext when running the encryption algorithm, one may wonder how the decrypted message can leak some information about the decryptor's witness. In appendix C, we provide an example to illustrate this scenario.

6.1 Application: Dark Pools

We now justify our model of WE with decryptor privacy. In our model, we are assuming that the decryptor D sends back the decrypted message to the encryptor E whereas in all previous works, the communication is non-interactive (i.e., "one-shot") in the sense that there is only one message ct from E to D. Our motivating applications are *dark pools* and *over-the-counter* markets. Dark pools are anonymized trading platforms that allow parties to place invisible orders such that each party can only know their own orders. Such pools allow the investors to communicate only to those whose transaction conditions satisfy some constraints. At the same time, they should also guarantee that investors do not learn any information about traders' secret information.

In a recent work, Ngo et al. [NMKW21] introduced a new cryptographic primitive called *Witness Key Agreement* (WKA) as a tool to make this possible. In the dark pool scenario, a WKA allows a party E to securely agree on a secret key with another party D who owns a secret witness satisfying some arithmetic relation. More precisely, in the presence of a public bulletin board or a public blockchain, a WKA addresses the following problem: given n parties who have committed to their secret inputs w, and published the commitments cm anonymously on the blockchain, an investor E wants to agree on a key k

with any party whose committed secret w satisfies some relation; i.e., $C(\mathsf{w}) = y$, where C is an arbitrary arithmetic circuit specified by E. Similar to NP relations defined in section 5, one can set $\mathsf{x} = (\mathsf{cm}, C, y)$ and let $\mathcal{R}$ be defined such that $\mathcal{R}(\mathsf{x}, (\mathsf{w}, \mathsf{d})) = 1$ iff cm commits to w (with decommitment d) and $C(\mathsf{w}) = y$. Once the secret key k is recovered by the legitimate party (i.e., any party with valid witness (w, d) such that $\mathcal{R}(\mathsf{x}, (\mathsf{d}, \mathsf{w})) = 1$), they together with the investor can secure their communication from any external party by using k.

We now demonstrate how our construction of ZK-CPPA can be used as a drop-in replacement for a witness key agreement. At a high level, the protocol proceeds as follows. All parties first commit to their secret values w via $\mathsf{cm} \leftarrow \mathsf{Commit}(\mathsf{w}; \mathsf{d})$, and publish the resulting commitments cm. Later, an investor who wish to communicate only with participants whose secret satisfy $C(\mathsf{w}) = y$ (for some arbitrarily chosen circuit C and value y) considers the following relation: $\mathcal{R}(\mathsf{x} = (\mathsf{cm}, C, y), (\mathsf{w}, \mathsf{d})) = 1$ iff $C(\mathsf{w}) = y$, and $\mathsf{cm} = \mathsf{Commit}(\mathsf{w}; \mathsf{d})$. Let us assume that $\mathsf{x}_i = (\mathsf{cm}_i, C, y)$ is the statement corresponding to party i. The investor now encrypts the secret key k under all such statements x_i [5]. It is not hard to see that only the prover with the valid witness $(\mathsf{w}_i, \mathsf{d}_i)$ can decrypt the ciphertext. Moreover, since the construction is ZK, the decrypted message k says nothing about $(\mathsf{w}_i, \mathsf{d}_i)$, even if the ciphertext is generated maliciously.

Efficiency and Comparison with [NMKW21]. In [NMKW21], the authors propose a WKA construction based on a type of *Succinct Zero-Knowledge Non-Interactive Argument of Knowledge Proof System* (zk-SNARK) from non-interactive linear proof systems (NILP), where the verifier is designated. The construction at a high-level is as follows. A designated verifier—playing the role of the investor— first broadcasts a CRS as a challenge for the relation $\mathcal{R}$ of interest. Next, a prover publishes a partial zk-SNARK proof as a response for the committed value that satisfies $\mathcal{R}$. Finally, the verifier using the partial proof can derive a shared secret key with the prover.

We now compare our proposed construction for WKA with that of [NMKW21]. In contrast to our scheme which is ZK, the construction of [NMKW21] only provides honest-verifier ZK. Moreover, the WKA in [NMKW21] requires an expensive trusted setup which should be invoked every time an investor E_i asks for the preprocessing of a new CRS corresponding to the relation $\mathcal{R}_i$ of E_i's interest. On the other hand, the major downside of our scheme is that the size of the ciphertext grows linearly with the number of parties in the system as the investor should encrypt the message under every existing commitment in the system, whereas the size of ciphertext in [NMKW21] is independent of the number of parties. This suggests that there might well be a trade-off between the size of the ciphertext and the required number of trusted setups and our construction performs better when the number of parties is small.

[5] We again emphasize that we see the notions of PA and WE (and their "commit-and-prove" variants) interchangeably here, as the implication from one to another is straightforward and shown in [FNV17].

7 Conclusion and Open Problems

In this work, we study predictable arguments with privacy properties and show their application to the construction of witness encryption schemes that require decryptor's privacy. We also introduce CPPAs that provide a weakening of predictability and give an efficient construction using garbled circuits techniques. While we construct CPPA in the random oracle model, an interesting open question is whether PA also exists in this model. Another theoretical question left open by our work is to show if WI deterministic-prover argument (WI-DA) implies WI-PA. While zero-knowledge deterministic-prover argument (ZK-DA) was characterized in a recent work by Bitansky and Choudhuri in [BC20], where they showed that ZK-DA implies ZK-PA, it would be interesting to do the same characterization for the weaker notion of witness indistinguishability. Finally, finding more applications for CPPA would be an interesting question.

Acknowledgment

We thank Matteo Campanelli and Jesper Buus Nielsen for helpful comments. We also thank the anonymous reviewers of ACNS'22 for their valuable feedback. The author was funded by the Concordium Foundation under Concordium Blockchain Research Center, Aarhus.

References

ACP09. Michel Abdalla, Céline Chevalier, and David Pointcheval. Smooth projective hashing for conditionally extractable commitments. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 671–689. Springer, Heidelberg, August 2009.

AKL21. Behzad Abdolmaleki, Hamidreza Khoshakhlagh, and Helger Lipmaa. Smooth zero-knowledge hash functions. *IACR Cryptol. ePrint Arch.*, 2021:653, 2021.

BBC$^+$13. Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. New techniques for SPHFs and efficient one-round PAKE protocols. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 449–475. Springer, Heidelberg, August 2013.

BC20. Nir Bitansky and Arka Rai Choudhuri. Characterizing deterministic-prover zero knowledge. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part I*, volume 12550 of *LNCS*, pages 535–566. Springer, Heidelberg, November 2020.

BHR12. Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *ACM CCS 2012*, pages 784–796. ACM Press, October 2012.

BIJ$^+$20. James Bartusek, Yuval Ishai, Aayush Jain, Fermi Ma, Amit Sahai, and Mark Zhandry. Affine determinant programs: A framework for obfuscation and witness encryption. In Thomas Vidick, editor, *ITCS 2020*, volume 151, pages 82:1–82:39. LIPIcs, January 2020.

BL20. Fabrice Benhamouda and Huijia Lin. Mr NISC: Multiparty reusable non-interactive secure computation. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part II*, volume 12551 of *LNCS*, pages 349–378. Springer, Heidelberg, November 2020.

BP13. Fabrice Benhamouda and David Pointcheval. Trapdoor smooth projective hash functions. Cryptology ePrint Archive, Report 2013/341, 2013. `https://eprint.iacr.org/2013/341`.

CDK$^+$21. Matteo Campanelli, Bernardo David, Hamidreza Khoshakhlagh, Anders K. Kristensen, and Jesper Buus Nielsen. Encryption to the future: A paradigm for sending secret messages to future (anonymous) committees. *IACR Cryptol. ePrint Arch.*, page 1423, 2021.

CFQ19. Matteo Campanelli, Dario Fiore, and Anaïs Querol. LegoSNARK: Modular design and composition of succinct zero-knowledge proofs. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2075–2092. ACM Press, November 2019.

CPW20. Suvradip Chakraborty, Manoj Prabhakaran, and Daniel Wichs. Witness maps and applications. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part I*, volume 12110 of *LNCS*, pages 220–246. Springer, Heidelberg, May 2020.

CS02. Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 45–64. Springer, Heidelberg, April / May 2002.

CVW18. Yilei Chen, Vinod Vaikuntanathan, and Hoeteck Wee. GGH15 beyond permutation branching programs: Proofs, attacks, and candidates. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 577–607. Springer, Heidelberg, August 2018.

DL20. Hila Dahari and Yehuda Lindell. Deterministic-prover zero-knowledge proofs. Cryptology ePrint Archive, Report 2020/141, 2020. `https://eprint.iacr.org/2020/141`.

FNV17. Antonio Faonio, Jesper Buus Nielsen, and Daniele Venturi. Predictable arguments of knowledge. In Serge Fehr, editor, *PKC 2017, Part I*, volume 10174 of *LNCS*, pages 121–150. Springer, Heidelberg, March 2017.

FS90. Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *22nd ACM STOC*, pages 416–426. ACM Press, May 1990.

GGH$^+$13. Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pages 40–49. IEEE Computer Society Press, October 2013.

GGSW13. Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 467–476. ACM Press, June 2013.

GKPS18. Chaya Ganesh, Yashvanth Kondi, Arpita Patra, and Pratik Sarkar. Efficient adaptively secure zero-knowledge from garbled circuits. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part II*, volume 10770 of *LNCS*, pages 499–529. Springer, Heidelberg, March 2018.

GKW17. Rishab Goyal, Venkata Koppula, and Brent Waters. Lockable obfuscation. In Chris Umans, editor, *58th FOCS*, pages 612–621. IEEE Computer Society Press, October 2017.

GMR89. Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

GO94. Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7(1):1–32, December 1994.

GOS06. Jens Groth, Rafail Ostrovsky, and Amit Sahai. Non-interactive zaps and new techniques for NIZK. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 97–111. Springer, Heidelberg, August 2006.

Ham16. Fabrice Ben Hamouda-Guichoux. *Diverse modules and zero-knowledge.* PhD thesis, École Normale Supérieure, Paris, France, 2016.

NMKW21. Chan Nam Ngo, Fabio Massacci, Florian Kerschbaum, and Julian Williams. Practical witness-key-agreement for blockchain-based dark pools financial trading. In *Financial Cryptography and Data Security - 25th International Conference, FC 2021, Virtual Event, March 1-5, 2021, Revised Selected Papers, Part II*, pages 579–598, 2021.

Yao86. Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986.

Choose. On input $(\texttt{receive}, \text{sid}, b)$ from R , where $b \in \{0,1\}$, if no messages of the form $(\texttt{receive}, \text{sid}, b)$ is stored, store $(\texttt{receive}, \text{sid}, b)$ and send $(\texttt{receive}, \text{sid})$ to S.

Transfer. On input $(\texttt{send}, \text{sid}, x^0, x^1)$ from S, with $x^0, x^1 \in \{0,1\}^k$, if no messages of the form $(\texttt{send}, \text{sid}, x^0, x^1)$ is stored and a message of the form $(\texttt{receive}, \text{sid}, b)$ is present, send $(\texttt{sent}, \text{sid}, x^b)$ to R.

Fig. 7: The ideal functionality $\mathcal{F}_{\mathsf{OT}}$ for oblivious transfer

Supporting Material

A Additional Preliminaries

A.1 Oblivious Transfer

A 2-round oblivious transfer (OT) is a protocol between a receiver R and a sender S and consists of three polynomial-time algorithms $\mathit{\Pi}_{\mathsf{OT}} = (\mathit{\Pi}^R_{\mathsf{OT}}, \mathit{\Pi}^S_{\mathsf{OT}}, \mathit{\Pi}^O_{\mathsf{OT}})$ defined as follows:

First round. R generates the first message $m^R \leftarrow \mathit{\Pi}^R_{\mathsf{OT}}(b; r^R)$ for the selection bit $b \in \{0,1\}$ and random tape $r^R \in \{0,1\}^{\mathsf{poly}(\lambda)}$.

Second round. For the input messages (x^0, x^1), where $x^l \in \{0,1\}^{\mathsf{poly}(\lambda)}$ for $l \in \{0,1\}$, S generates the second message $m^S \leftarrow \mathit{\Pi}^S_{\mathsf{OT}}(m^R, (x^0, x^1); r^S)$ using random tape $r^S \in \{0,1\}^{\mathsf{poly}(\lambda)}$.

Output. R computes the output $x = \mathit{\Pi}^O_{\mathsf{OT}}(m^S, b, r^R)$.

We require an OT protocol to securely implement the ideal functionality in fig. 7 in the presence of malicious adversaries.

For our construction, we also require another property called *sender-extractability* in [GKPS18], which informally states that the randomness of the sender is sufficient to reconstruct its input with high probability.

– **Sender-Extractability.** For any security parameter $\lambda \in \mathbb{N}$, for any $b \in \{0,1\}$, for any messages (x^0, x^1), where $x^l \in \{0,1\}^{\mathsf{poly}(\lambda)}$ for $l \in \{0,1\}$, there exist a PPT algorithm Ext such that for $m^R \leftarrow \mathit{\Pi}^R_{\mathsf{OT}}(b; r^R)$, and $m^S \leftarrow \mathit{\Pi}^S_{\mathsf{OT}}(m^R, (x^0, x^1); r^S)$, where $r^R, r^S \in \{0,1\}^{\mathsf{poly}(\lambda)}$, we have

$$\Pr\left[(\bar{x}^0, \bar{x}^1) \neq (x^0, x^1) \ : \ (\bar{x}^0, \bar{x}^1) \leftarrow \mathsf{Ext}(m^R, m^S, r^S)\right] \approx_\lambda 0$$

A.2 Garbled Circuit

Garbled circuit introduced by [Yao86] is a cryptographic technique that enables two-party secure computation in which two parties do not trust each other and want to jointly evaluate a function over their private inputs. The following definition is from the *garbling schemes* abstraction introduced by Bellare et al. in [BHR12].

Definition 6 (Garbling Scheme). *Let $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ be a polynomial-size circuit class. A garbled circuit scheme* GC *for $\mathcal{C}$ consists of four polynomial-time algorithms* $\mathsf{GC} = (\mathsf{Garble}, \mathsf{Encode}, \mathsf{Eval}, \mathsf{Decode})$*:*

$(\mathbf{C}, e, d) \leftarrow \mathsf{Garble}(1^\lambda, C)$**:** *On input a boolean circuit $C \in \mathcal{C}_\lambda$, outputs $(\mathbf{C}, e, d)$, where $\mathbf{C}$ is a garbled circuit, e is encoding information, and d is decoding information.*

$X \leftarrow \mathsf{Encode}(e, x)$**:** *On input e and x, where x is a suitable input for C, outputs a garbled input X.*

$Y = \mathsf{Eval}(\mathbf{C}, X)$**:** *On input $(\mathbf{C}, X)$ as above, outputs a garbled output Y.*

$y \leftarrow \mathsf{Decode}(d, Y)$**:** *On input (d, Y) as above, outputs a plain output y.*

A garbling scheme should satisfy the following correctness and security properties:

Correctness. For any security parameter $\lambda \in \mathbb{N}$, for any circuit $C \in \mathcal{C}_\lambda$, for $(\mathbf{C}, e, d) \leftarrow \mathsf{Garble}(1^\lambda, C)$, and for all suitable input x:

$$\mathsf{Decode}(d, \mathsf{Eval}(\mathbf{C}, \mathsf{Encode}(e, x))) = C(x)$$

Authenticity. For all circuits $C : \{0,1\}^n \to \{0,1\}$, inputs $x \in \{0,1\}^n$, where $n = \mathsf{poly}(\lambda)$, and for all PPT adversaries $\mathcal{A}$,

$$\Pr\left[\begin{matrix}\hat{Y} \neq \mathsf{Eval}(\mathbf{C}, X) \wedge \\ \mathsf{Decode}(d, \hat{Y}) \neq \perp\end{matrix} : \begin{matrix} (\mathbf{C}, e, d) \leftarrow \mathsf{Garble}(1^\lambda, C) \\ X = \mathsf{Encode}(e, x);\ \hat{Y} \leftarrow \mathcal{A}(C, x, \mathbf{C}, X)\end{matrix}\right] \approx_\lambda 0$$

Verifiability. There exists a PPT algorithm Verify such that for all circuits $C : \{0,1\}^{\mathsf{poly}(\lambda)} \to \{0,1\}$,

$$\Pr\left[\mathsf{Verify}(C, \mathbf{C}, e) = 1\ :\ (\mathbf{C}, e, d) \leftarrow \mathsf{Garble}(1^\lambda, C)\right] = 1$$

B Details on ZK-PA

The construction of ZK-PA for algebraic languages $\mathcal{L}_{\mathsf{lpar}}$ with $\mathsf{lpar} = (\mathbf{M}, \boldsymbol{\theta})$ is depicted in fig. 8.

– $\mathsf{Setup}(1^\lambda)$: $\tau \leftarrow\!\!\$\ \mathbb{Z}_p$; **return** $(\mathsf{crs}_\tau = [\tau]_2, \tau)$.
– $\mathsf{Chall}(\mathsf{lpar}, \mathbf{x})$: $\boldsymbol{\alpha} \leftarrow\!\!\$\ \mathbb{Z}_p^n$; $[\gamma]_1 \leftarrow \boldsymbol{\alpha}^\top[\mathbf{M}(\mathbf{x})]_1$; $[\xi]_2 \leftarrow \boldsymbol{\alpha}[\tau]_2; c \leftarrow ([\gamma]_1, [\xi]_2); b \leftarrow \hat{e}(\boldsymbol{\alpha}^\top[\boldsymbol{\theta}(\mathbf{x})]_1, [1]_2)$; **return** (c, b).
– $\mathsf{Resp}(\mathsf{lpar}, \mathbf{x}, \mathbf{w}, c)$: parse c as $([\gamma]_1, [\xi]_2)$; **if** $([\xi]_2 \notin \mathbb{G}_2^n \vee \hat{e}([\gamma]_1, \mathsf{crs}_\tau) \neq \hat{e}([\mathbf{M}]_1, [\xi]_2))$ **return** $\perp$; **else return** $a \leftarrow \hat{e}([\gamma]_1\mathbf{w}, [1]_2)$.
– $\mathsf{Sim}(\mathsf{lpar}, \mathbf{x}, \tau, c)$: parse c as $([\gamma]_1, [\xi]_2)$; **return** $\tau^{-1}\hat{e}([\boldsymbol{\theta}(\mathbf{x})]_1, [\xi]_2)$.

Fig. 8: Construction of ZK-PA from [BP13].

C Witness Encryption with Decryptor's privacy: An example

To clarify how a decrypted message which is already known by the encryptor can leak some information about the decryptor's witness, let us consider a WE scheme for a concrete disjunction language defined as follows: the language parameter $\mathsf{lpar} = (\mathbb{G}, g, \mathsf{pk})$ includes a group $\mathbb{G}$ of order p with generator g, and an ElGamal public key pk. A statement x is in the language iff x is the ElGamal encryption of a bit under pk. More formally, for $\mathsf{lpar} = (\mathbb{G}, g, \mathsf{pk})$, we define

$$\mathcal{L}_{\mathsf{lpar}} = \left\{ \mathsf{x} \mid \exists r \in \mathbb{Z}_p, \exists b \in \{0,1\} : \mathsf{x} = (g^r, \mathsf{pk}^r g^b) \right\}$$

We denote the witness for $\mathsf{x} = (g^r, \mathsf{pk}^r g^b)$ as $\mathsf{w} = (r, b)$. Using generic techniques for the disjunctions of languages([ACP09, Ham16]), one can encrypt a message $m \in \mathbb{G}$ under a statement $\mathsf{x} = (\mathsf{x}_0, \mathsf{x}_1)$ as follows:

1. select $\alpha_0, \alpha_1, \alpha_2, \alpha_3 \leftarrow\!\!\$\ \mathbb{Z}_p$.
2. compute $\mathsf{aux}_0 = g^{\alpha_0}\mathsf{pk}^{\alpha_1}$, $\mathsf{aux}_1 = g^{\alpha_1}\mathsf{x}_0^{\alpha_2}(\frac{\mathsf{x}_1}{g})^{\alpha_3}$ and $\mathsf{aux}_2 = g^{\alpha_2}\mathsf{pk}^{\alpha_3}$. Define $\mathsf{aux} = (\mathsf{aux}_0, \mathsf{aux}_1, \mathsf{aux}_2)$.
3. compute $\pi = \mathsf{x}_0^{\alpha_0}\mathsf{x}_1^{\alpha_1}$ and $\tilde{\pi} = \pi\mathsf{x}_0^{\alpha_0}(\frac{\mathsf{x}_1}{g})^{\alpha_1}g^{\alpha_1}$. Define $ct = \pi m$.
4. return $ct = (c, \mathsf{aux}, \tilde{\pi})$.

Having a witness $\mathsf{w} = (r, 0)$ for $\mathsf{x} = (g^r, \mathsf{pk}^r)$, one can decrypt the ciphertext by first computing $\pi = \mathsf{aux}_0^r$ and then recovering the message by computing $\pi^{-1}ct$. On the other hand, if the decryptor has a witness $\mathsf{w} = (r, 1)$ for $\mathsf{x} = (g^r, \mathsf{pk}^r g)$, he first obtains π from dividing $\tilde{\pi}$ by $\mathsf{aux}_0^r\mathsf{aux}_1^1\mathsf{aux}_2^{-r} = \mathsf{x}_0^{\alpha_0}(\frac{\mathsf{x}_1}{g})^{\alpha_1}g^{\alpha_1}$ and then computes $m = \pi^{-1}ct$ as before. While for an honestly generated ciphertext ct, this construction does not leak any information about the witness, it is not hard to see that a malicious encryptor can learn part of the witness (and thus distinguish them) by simply defining $\tilde{\pi}$ to be a random group element. In this case (i.e., $b = 1$), the decryptor fails to decrypt m correctly, hence making the two cases $b = 0$ and $b = 1$ distinguishable for the encryptor.