Zero-Knowledge Protocols with PVC Security: Striking the Balance between Security and Efficiency

Yi Liu[0000−0003−1722−6746], Yipeng Song[0009−0008−0406−362X], Anjia Yang[0000−0002−7958−6571], and Junzuo Lai[0000−0001−5780−8463]

College of Cyber Security, Jinan University, Guangzhou 510632, China
liuyi@jnu.edu.cn, yipeng93857@gmail.com, anjiayang@gmail.com, laijunzuo@gmail.com

Abstract. Zero-knowledge protocols allow a prover to prove possession of a witness for an NP-statement without revealing any information about the witness itself. This kind of protocol has found extensive applications in various fields, including secure computation and blockchain. However, in certain scenarios (*e.g.*, when the statements are complicated), existing zero-knowledge protocols may not be well-suited due to their limited applicability or high computational overhead.

We address these limitations by incorporating the notion of *publicly verifiable covert (PVC) security* into zero-knowledge protocols. PVC security, recently emerging from secure computation, effectively balances security and efficiency in practical scenarios. With PVC security, while a malicious party may attempt to cheat, such cheating will be detected and become publicly verifiable with a significant probability (called deterrence factor, *e.g.*, >90%). This notion is well-suited for practical scenarios involving reputation-conscious parties (*e.g.*, companies) and offers substantial efficiency improvements.

In this paper, we present the *first* definition of *zero-knowledge protocols with PVC security*. We then propose a generic transformation to convert Sigma protocols with 1-bit challenge, a kind of protocol widely used for zero-knowledge, into efficient zero-knowledge protocols with PVC security. By applying our transformation, we can substantially improve the efficiency of existing protocols for reputation-sensitive parties. For instance, applying the transformation to achieve a deterrence factor of 93.75% incurs a cost of only around 20% compared to the original protocol. Therefore, our results contribute to significant advancements in practical zero-knowledge protocols.

Keywords: Publicly verifiable covert security · Zero-knowledge · Generic transformation.

1 Introduction

The notion of zero-knowledge proofs was introduced by Goldwasser, Micali, and Rackoff in 1985 [16]. This cryptographic paradigm enables a prover to convinc-

ingly demonstrate the validity of a statement to a verifier without revealing any information beyond the truth of the statement itself.

Zero-knowledge *proof of knowledge* is a widely applicable notion closely related to zero-knowledge proofs. It enables the prover to demonstrate *possession of specific knowledge* (referred to as a witness) regarding a statement while avoiding the disclosure of any information about the knowledge itself. Zero-knowledge proof of knowledge protocols have found extensive applications in various fields, including secure computation [14], identification [28], and blockchain [4]. For simplicity and conciseness, we will use the term "zero-knowledge protocols" hereinafter to refer to zero-knowledge proof of knowledge protocols[1].

Despite their broad applicability, zero-knowledge protocols can be ineffective in certain contexts. Specific zero-knowledge protocols, such as those based on the structure of Schnorr protocols [28], may achieve high efficiency but are constrained to proving a limited range of statements. In contrast, generic zero-knowledge protocols, such as zk-SNARKs [12, 27, 17] and Bulletproofs [5], can become computationally intensive when dealing with complex representations of statements. Notably, many interesting and useful zero-knowledge protocols that are not efficient enough follow a similar repetitive structure, where a sub-protocol is invoked multiple times, and in each iteration, the prover responds to a 1-bit challenge $e \in \{0, 1\}$ chosen by the verifier, *e.g.*, quadratic residues [16], graph isomorphism [15], twin-ciphertext proofs [7, 25], etc. On the one hand, unlike Schnorr-like protocols with large challenge spaces, the challenge spaces for this kind of protocol are only 1 bit, and thus, repeating the sub-protocol many times can be computationally expensive. On the other hand, the statements for this kind of protocol are complicated and often ineffective in proving the statements directly via generic zero-knowledge protocols.

In cases where efficient specific zero-knowledge proofs are inapplicable and existing generic zero-knowledge protocols are inefficient, we, fortunately, find that modest relaxation of security requirements could lead to significant efficiency improvements in some certain real-life scenarios, such as the protocols with the repetitive structure mentioned above.

Our approach builds on the notion of publicly verifiable covert (PVC) security that has recently emerged in secure computation. Introduced by Asharov and Orlandi in 2012 [2], PVC security strikes a balance between semi-honest and malicious security while providing a publicly verifiable cheating detection mechanism. A protocol with PVC security guarantees that any deviation of cheaters will be detected by an honest party with a fixed probability ϵ, called the *deterrence factor*. Upon detection, PVC protocols enable the honest party to generate a *publicly verifiable certificate* as *proof of cheating* with respect to the cheater. This certificate can convince all entities and holds utility in legal proceedings or smart contracts that financially penalize the cheater. This PVC security prop-

[1] In this paper, we do not explicitly differentiate between the notions of *proof* and *argument*, and we confine our considerations to scenarios where both the prover and the verifier run within probabilistic polynomial-time.

erty is particularly compelling in practice due to its efficiency[2]. Moreover, it permanently exposes the cheater's misbehavior publicly, imposing reputational risk and accountability. Due to this practical property, the notion of PVC security has gained increasing attention in recent years, leading to advances in protocol design [2, 22, 21, 9, 29, 10, 3, 24] and many applications, including financially backed protocols [30, 11], secure computation involving commitment on the blockchain [1], and private function evaluation [26]. In this work, we find that the notion of PVC security is also applicable to certain zero-knowledge protocol scenarios in addition to secure computation. We outline relevant scenario characteristics as follows.

Reputation-conscious parties. In many secure computation protocols, zero-knowledge protocols may serve as sub-protocols to prove the correctness of data or operations. If parties in the protocols are reputation-conscious (*e.g.*, companies like Google and Microsoft), though they may have the incentive to cheat in order to gain advantages, they are also concerned about the risk of being caught and publicly exposed. In these scenarios, having a noticeable probability (*e.g.*, 93.75%) of being detected and publicly exposed is sufficient to deter malicious behaviors.

Repeated sub-protocols. Many interesting zero-knowledge protocols are constructed by repeating a sub-protocol sequentially or in parallel, including protocols for quadratic residues [16], graph isomorphism [15], and twin-ciphertext proofs [7, 25]. Typically, the sub-protocol involved has high knowledge error (*i.e.*, the challenge space is small), necessitating many repetitions to prevent a malicious party from guessing the challenge. However, for reputation-conscious parties, if cheating would lead to detection and public exposure, the number of sub-protocol repetitions could be significantly reduced (*e.g.*, from 40 to 4). PVC security is thus applicable.

The primary objective of this paper is to explore the approach of integrating the notion of PVC security into zero-knowledge protocols with respect to reputation-conscious parties. By leveraging the deterrent effect of publicly verifiable cheating detection, this incorporation aims to improve efficiency.

1.1 Our Contributions

In this paper, we provide a formal definition of zero-knowledge protocols with PVC security and propose a generic transformation method to convert protocols with specific structures into zero-knowledge protocols with PVC security. Our main contributions can be summarized as follows.

New notion. To address scenarios involving reputation-conscious parties where efficient protocols are lacking for certain statements, we introduce the notion

[2] Recent studies have demonstrated that a two-party PVC protocol with a deterrence factor of 50% incurs only $20-40\%$ overhead compared to state-of-the-art semi-honest protocols based on garbled circuits [21].

of *zero-knowledge protocols with PVC security*. This novel notion strikes a balance between security and efficiency, and we provide a formal definition for it.

New protocols. We propose a generic transformation approach to convert protocols with specific structures, namely Sigma protocols with 1-bit challenge, into zero-knowledge protocols with PVC security. This class of protocols encompasses a variety of interesting statements, including quadratic residues [16], graph isomorphism [15], and twin-ciphertext proofs [7, 25]. By applying our approach, we *significantly* improve the efficiency of the original protocols in scenarios involving reputation-conscious parties. For instance, achieving a deterrence factor of 93.75% incurs a cost of only around 20% compared to the original protocol (which typically requires numerous repetitions to achieve low soundness error).

Overall, our contributions make significant advancements in the field of zero-knowledge protocols, providing a practical approach to improving the efficiency of existing protocols with specific structures when reputation-conscious parties are involved.

1.2 Technical Overview

In this subsection, we provide a concise introduction to zero-knowledge protocols with PVC security and outline the fundamental idea of designing such protocols from Sigma protocols with a 1-bit challenge.

Informally, given a statement x and a NP language L (with relation R), zero-knowledge protocols with PVC security allow a prover to prove to a verifier that she possesses a witness w, such that $(x, w) \in R$, while the verifier learns no additional information beyond the fact that the prover knows a witness w satisfying $(x, w) \in R$. Leveraging the notion of PVC security, a malicious prover, lacking the witness w, can convince the verifier that she possesses the witness for x *only with a small probability* (denoted as $1-\epsilon$, *e.g.*, 6.25%). However, if the malicious prover fails to persuade the verifier with a significant probability ϵ (*e.g.*, 93.75%, called *deterrence factor*), the verifier should be capable of generating a publicly verifiable certificate to attribute blame to the malicious prover. This certificate serves as evidence of cheating by the malicious prover and can be verified by any party external to the protocol.

It is crucial to emphasize that once the malicious prover becomes aware that her dishonest actions have been detected by the verifier, she is unable to prevent the verifier from generating and disseminating the certificate. Thus, the malicious prover's decision to proceed with the protocol and cheat without possessing the witness is made independently of the event of being caught. Consequently, the design of zero-knowledge protocols with PVC security presents a significant challenge. Achieving a mechanism that enables the verifier to gather sufficient evidence for blaming cheating before the malicious prover becomes aware of being caught is a highly nontrivial task.

We observe that a class of zero-knowledge protocols constructed by repeating a sub-protocol is well-suited for extension to PVC security in scenarios involving reputation-conscious parties. To illustrate our idea, we consider as an example the well-known zero-knowledge protocol for quadratic residues [16]. In this protocol, a prover aims to prove to the verifier that she knows $w \in \mathbb{Z}_N^*$, such that $x = w^2 \bmod N$, in zero-knowledge. The protocol proceeds as follows.

1. The prover selects a random value r in $\mathbb{Z}_N^*$, computes $y = r^2 \bmod N$, and sends y to the verifier.
2. The verifier returns a random bit $b \in \{0, 1\}$ to the prover.
3. If $b = 0$, the prover sends $z = r$ to the verifier. If $b = 1$, the prover sends $z = wr \bmod N$ to the verifier.
4. The verifier accepts the proof if and only if $z^2 \equiv x^b y \bmod N$.

The insight that the prover possesses w (*i.e.*, the protocol is proof-of-knowledge) is that if we obtain two accepting transcripts for the same y and different b, we can easily extract w. It is also easy to see that if the prover does not possess w, she can still guess a bit $\hat{b}$ in Step 1 and provide $y = z^2 x^{-\hat{b}} \bmod N$ for a random z in $\mathbb{Z}_n^*$. If $b = \hat{b}$, the verifier will accept the proof for the given z. Hence, to ensure the prover's possession of w, the protocol is typically executed multiple times, sequentially or in parallel, with slight modifications [13]. In the setting of reputation-conscious provers, we can introduce PVC security to largely reduce the number of sub-protocol executions. For example, a deterrence factor of 93.75% can be achieved with only 4 instances to be verified by the verifier, and a deterrence factor of approximately 99% requires only 7 instances. This is sufficient for most real-world scenarios, especially when the publicly verifiable cheating detection mechanism is also available.

To achieve PVC security, the challenge bit b must be unknown to the prover in Step 4. If a malicious prover without possessing w is aware of b, she can simply abort the protocol, claiming a network error, and the verifier will be unable to generate a publicly verifiable certificate for the abortion. However, if the prover answers both challenges $b = 0$ and $b = 1$ (without knowing b now), the verifier can extract the witness w, thereby compromising the zero-knowledge property. To address this issue, we employ an oblivious transfer (OT) protocol. Using the OT protocol, the prover's input becomes (z_0, z_1), representing her answers for $b = 0$ and $b = 1$, respectively, while the verifier's input is either $b = 0$ or $b = 1$. The verifier's output in the OT protocol is z_b, allowing for only checking the correctness of z_b, while the prover is not aware of b.

To generate a publicly verifiable certificate for blaming a malicious prover providing incorrect (z_0, z_1), the verifier requires the prover's signature on a cheating message. However, directly having the prover sign z_0 and z_1 and include the signature in the OT protocol is ineffective as the prover can generate an invalid signature for the incorrect z_i. Our basic idea is to use a derandomization approach. Initially, the verifier selects a random seed, commits to it, and sends the commitment to the prover. The verifier then employs the seed to derive the randomness used during the protocol execution. After the execution of the (perfectly correct) OT protocol, the prover, unaware of the challenge bit, signs the complete

transcript of the OT protocol, the messages she has sent, and the commitment of the verifier's seed. In case of detecting cheating, the verifier can include the OT protocol transcript, the prover's signature and sent messages, and the opening of the commitment in the certificate. With the seed, the prover's message, and the OT protocol transcript, any party can now verify the signature, simulate the protocol execution of the verifier, and verify whether the prover was cheating.

We note that the definition and protocol design also involves several technical details. For instance, while zero-knowledge protocols can be seen as a specific class of secure computation, we cannot directly combine the definition of PVC security for secure computation with that of zero-knowledge protocols. Instead, we need to carefully describe the security requirements when the verifier has no input and only outputs a single bit. Another technical detail is that, as our definition of zero-knowledge protocols does not impose restrictions on the deterrence factor, we choose to use a commitment scheme that supports equivocation (*e.g.*, implemented using a random oracle) in the protocol. Consequently, the protocol is modified to let the prover commit to the first messages of the sub-protocols and subsequently open them after the OT protocol. This modification enables us to prove that the protocol achieves the zero-knowledge property when the challenge space of the protocol becomes exponentially large. For a detailed definition, description of the protocol, and its security proof, we refer the reader to Section 3 and 4.

2 Preliminaries

For a set S, we denote the size of S as $|S|$ and use $x \leftarrow_{\$} S$ to represent the uniform sampling of an element x from S. Moreover, we define $[n] = \{1, \ldots, n\}$ for a positive integer n.

Let κ denote the computational security parameter, which is provided as input in unary format to all algorithms. A function f in κ, mapping natural numbers to the interval $[0, 1]$, is *negligible* if $f(\kappa) = \mathcal{O}(\kappa^{-c})$ for every constant $c > 0$.

Given a $\mathsf{seed} \in \{0,1\}^{\kappa}$, we can employ a pseudorandom function with seed as the key in Counter (CTR) mode to generate a sufficient number of pseudorandom numbers, which can be utilized as local randomness in protocol executions.

In our protocol, we utilize a (non-interactive) commitment scheme denoted as Com. This scheme employs random coins decom for both commitment generation and opening. It is required to satisfy (computational) binding and hiding properties while also supporting equivocation. This scheme can be implemented by a random oracle $H\colon \{0,1\}^* \to \{0,1\}^{\kappa}$ via defining $\mathsf{Com}(m) = H(m, \mathsf{decom})$, where $\mathsf{decom} \leftarrow_{\$} \{0,1\}^{\kappa}$. Additionally, our protocol incorporates the signature scheme $(\mathsf{KGen}, \mathsf{Sig}, \mathsf{Verify})$, which is existentially unforgeable under chosen-message attacks (EUF-CMA).

An execution transcript of a two-party protocol is of the form $\mathsf{trans} = (m_1, m_2, m_3, \ldots)$, where the parties send their messages alternately.

Let Π_{OT} represent the protocol that securely realizes a parallel version of the OT functionality $\mathcal{F}_{\mathsf{OT}}$ below with perfect correctness [9], ensuring that the receiver cannot "equivocate" its view by finding a random tape that produces a different output from the output in a real execution.

Functionality $\mathcal{F}_{\mathsf{OT}}$

Private inputs: The sender has input $\{(z_j^0, z_j^1)\}_{j\in[\lambda]}$ and the receiver has input $e \in [\lambda]$.
Upon receiving $\{(z_j^0, z_j^1)\}_{j\in[\lambda]}$ from the sender, send $z_j^{e_j}$ to the receiver, where e_j is the jth bit of e.

3 Zero-Knowledge Protocols with PVC Security

Before providing the definition of zero-knowledge protocols with PVC security, we first define the task of zero-knowledge with the ideal functionality $\mathcal{F}_{\mathsf{zk}}^R$ below, as in [6] (for further details, refer to [19, 20]).

Functionality $\mathcal{F}_{\mathsf{zk}}^R$

Public inputs: Both parties has input an NP statement x.
Private inputs: P has input w.
Upon receiving input $(\texttt{prove}, x, w)$ from P and $(\texttt{verify}, x)$ from V, if $x = x'$ and $(x, w) \in R$, send $(\texttt{accepted})$ to V. Otherwise, send $(\texttt{rejected})$ to V.

This definition approach offers the convenience of defining all the desired properties of a zero-knowledge protocol, including the *proof-of-knowledge* property, in a concise manner. It is worth mentioning that this simulation-based definition aligns with our definition of a zero-knowledge protocol with PVC security (Definition 1), which is better suited for simulation-based definitions.

Now, we modify $\mathcal{F}_{\mathsf{zk}}^R$ to accommodate PVC security and propose the functionality $\mathcal{F}_{\mathsf{pvc\text{-}zk}}^R$ as introduced in Section 1.2.

We would like to highlight that our definition, as described above, differs slightly from the definition of PVC security in the setting of secure two-party computation [2, 22, 21]. In zero-knowledge protocols, verifiers do not possess private input, and their output is constrained to a single bit of information, specifically either `accepted` or `rejected` when no abortion occurs. When cheating goes undetected, the prover can only cheat the verifier to output `accepted`, without gaining knowledge of the honest party's input or modifying the output to an arbitrary value, as is the case in secure computation.

Functionality $\mathcal{F}^{R}_{\mathsf{pvc\text{-}zk}}$ with deterrence ϵ

Public inputs: Both parties have input an NP statement x.
Private inputs: P has input w.
Upon receiving the message $(\texttt{prove}, x, y)$ from P, where $y \in \{0,1\}^n$, $y = \perp$, $y = (\texttt{cheat}, \hat{\epsilon})$ for $\hat{\epsilon} \geq \epsilon$, or $y = (\texttt{stupidCheat}, w, \tilde{\epsilon})$ (where the last three are sent by P* corrupted by the adversary) and $(\texttt{verify}, x')$ from V:

- If $y = \perp$ or $x = x'$, then send $\perp$ to both parties.
- If $y \in \{0,1\}^n$, then if $(x, y) \in R$, send $(\texttt{accepted})$ to V. Otherwise, if $(x, y) \notin R$, send $\perp$ to both parties.
- If $y = (\texttt{cheat}, \hat{\epsilon})$, then:
  - With probability $\hat{\epsilon}$, send $(\texttt{failed})$ to the adversary and $(\texttt{rejected})$ to V.
  - With probability $1 - \hat{\epsilon}$, send $(\texttt{successful})$ to the adversary and $(\texttt{accepted})$ to V.
- If $y = (\texttt{stupidCheat}, w, \tilde{\epsilon})$, such that $(x, w) \in R$, then:
  - With probability $\tilde{\epsilon}$, send $(\texttt{failed})$ to the adversary and $(\texttt{rejected})$ to V.
  - With probability $1 - \tilde{\epsilon}$, send $(\texttt{successful})$ to the adversary and $(\texttt{accepted})$ to V.

The formal definition of zero-knowledge protocols with PVC security is presented below.

Definition 1. *A protocol $\Pi^{R}_{\mathsf{pvc\text{-}zk}}$ along with algorithms* Blame *and* Judge *is a zero-knowledge protocol for the* NP*-relation R that achieves PVC security with deterrence ϵ if the following hold:*

Simulatability *The protocol $\Pi^{R}_{\mathsf{pvc\text{-}zk}}$, where the honest verifier* V *might generate a certificate* cert *if cheating is detected, securely realizes the ideal functionality $\mathcal{F}^{R}_{\mathsf{pvc\text{-}zk}}$ with deterrence ϵ.*

Public Verifiability *If the honest* V *outputs* `rejected` *(together with a certificate* cert*), then* $\mathsf{Judge}(\mathsf{cert}) = 1$*, except for a negligible probability.*

Defamation Freeness *For every* PPT *verifier* V^* *corrupted by the adversary, if the prover is honest, the probability that* V^* *generates a certificate* cert^* *that blames the honest prover and leads to* $\mathsf{Judge}(\mathsf{cert}^*) = 1$ *is negligible.*

Moreover, in the description of secure computation protocols with PVC security, if malicious behaviors are detected, the honest party should generate a certificate and send it to the adversary. Sending the certificate to the adversary indicates that the certificate does not reveal any information about the honest party's private input in the security proof. Conversely, in zero-knowledge protocols, since the verifier has no private input, there is no need to send the generated certificate to the adversary in the protocol description.

Furthermore, we introduce `stupidCheat` message in $\mathcal{F}^{R}_{\mathsf{pvc\text{-}zk}}$ in order to capture the scenario where the prover knows the witness but still intend to provide

invalid responses. It is important to note that this additional instruction does not compromise the security of the zero-knowledge protocol.

Additionally, we observe that setting the deterrence ϵ to be negligible in $\mathcal{F}^R_{\mathsf{pvc\text{-}zk}}$ achieves malicious security. Based on our definition, once the prover P causes the verifier V to output `rejected` in the maliciously secure protocols (*i.e.*, ϵ is negligible), the honest verifier can also generate a certificate to attribute blame to the prover's misconduct.

4 Our Approach

In this section, we introduce our transformation method to convert Sigma protocols with 1-bit challenge into zero-knowledge protocols with PVC security. We first present the definition of Sigma protocols with 1-bit challenge based on [8].

Definition 2. *A two-party protocol Π between a prover P and a verifier V is said to be a Sigma protocol with 1-bit challenge for the NP-relation R if*

- *The protocol Π is of three-move form for a statement x as follows.*

  1. P sends a message a.
  2. V sends a challenge $e \leftarrow\!\!\$\ \{0,1\}$ to P.
  3. P sends a reply message z, and V decides to accept or reject P based on the message (x, a, e, z).

  If $(x, w) \in R$ and P follows the protocol honestly with the correct witness w, then V always accepts.
- *For any statement x and any pair of accepting transcripts for x, $(a, 0, z)$ and $(a, 1, z')$, we can efficiently compute w, such that $(x, w) \in R$. This property is sometimes called* special soundness.
- *There exists a PPT simulator $\mathcal{S}$ that, on input, a statement x and a challenge $e \in \{0,1\}$, outputs an accepting transcript (a, e, z) that is (computationally) indistinguishable from the transcript in a real execution of Π on x. This property is sometimes called* special honest-verifier zero-knowledge.

Now, given a two-party Sigma protocol Π with a 1-bit challenge, we introduce a protocol $\Pi^R_{\mathsf{pvc\text{-}zk}}$ based on Π, following the idea presented in Section 1.2. Note that we let an honest prover check the correctness of her witness at the beginning of the protocol. This is to facilitate the security proof as in [20]. Additionally, we propose two algorithms, Blame and Judge, for $\Pi^R_{\mathsf{pvc\text{-}zk}}$. We then prove that the protocol $\Pi^R_{\mathsf{pvc\text{-}zk}}$ along with the two algorithms Blame and Judge achieves PVC security.

Protocol $\Pi^R_{\mathsf{pvc\text{-}zk}}$ with deterrence ϵ

Public inputs: Both parties know the two-party Sigma protocol Π with 1-bit challenge for the NP-relation R. Both parties have input an NP statement x. Both parties agree on the parameters κ and λ.
Private inputs: P has input the witness w and a private key sk for the EUF-CMA-secure signature scheme $(\mathsf{KGen}, \mathsf{Sig}, \mathsf{Verify})$. Both parties know the corresponding public key pk of the signature scheme.

1. If $(x, w) \notin R$, P aborts. V chooses $\mathsf{seed} \leftarrow_{\$} \{0,1\}^{\kappa}$ and uses seed to derive a string $e \leftarrow_{\$} \{0,1\}^{\lambda}$ and $\mathsf{seed}_j \leftarrow_{\$} \{0,1\}^{\kappa}$ for all $j \in [\lambda]$. We denote the ith bit of e by e_i. Then V computes the commitment $c_{\mathsf{seed}} \leftarrow \mathsf{Com}(\mathsf{seed})$ and sends c_{seed} to P.
2. For each $j \in [\lambda]$, P initiates an instance of Π and generates the first message a_j. Then P sends $c_a \leftarrow \mathsf{Com}(\{a_j\}_{j\in[\lambda]})$ to V.
3. For each $j \in [\lambda]$, P compute z_j^0 and z_j^1 for different challenge 0 and 1, respectively, according to Π. P and V perform λ executions of Π_{OT} in parallel. In the jth execution of Π_{OT}, the input of P as the sender is (z_j^0, z_j^1) while the input of V as the receiver is e_j. The randomness used by V in the jth execution of Π_{OT} are derived from seed_j. After the executions of Π_{OT}, V obtains $\{z_j^{e_j}\}$. We denote the transcript of the jth execution of Π_{OT} by trans_j.
4. P generates the signature $\sigma \leftarrow \mathsf{Sig}_{\mathsf{sk}}(x, \lambda, c_{\mathsf{seed}}, \{a_j, \mathsf{trans}_j\}_{j\in[\lambda]})$. Then P sends σ, $\{a_j\}_{j\in[\lambda]}$, and the decommitment of c_a (denoted by decom_a) to V.
5. V verifies the signature σ (with respect to $x' = x$) using pk, and the opening of c_a, and aborts with output $\perp$ if σ or the opening is invalid. V checks the correctness of $\{z_j^{e_j}\}$ for all $j \in [\lambda]$ based on (x, a_j, e_j) according to Π. If all $\{z_j^{e_j}\}$ are correct, V outputs `accepted`. Otherwise, V outputs `rejected` and can execute the algorithm Blame to generate the certificate cert as follows.
   (a) Randomly select an index $\hat{\jmath}$ such that $z_j^{e_j}$ is incorrect.
   (b) Generate $\mathsf{cert} = (x, \lambda, \sigma, c_{\mathsf{seed}}, \mathsf{seed}, \mathsf{decom}_{\mathsf{seed}}, \hat{\jmath}, \{a_j, \mathsf{trans}_j\}_{j\in[\lambda]})$.

Theorem 1. *If Π is a Sigma protocol with 1-bit challenge for the NP-relation R, then the protocol $\Pi^R_{\mathsf{pvc\text{-}zk}}$ along with the algorithms Blame and Judge is a zero-knowledge protocol for the NP-relation R that achieves PVC security with deterrence $\epsilon = 1 - 1/2^{\lambda}$.*

Proof. We separately prove that the protocol $\Pi^R_{\mathsf{pvc\text{-}zk}}$ achieves simulatability, public verifiability, and defamation freeness below.

Simulatability. We first focus on the simulatability. Let $\mathcal{A}$ be an adversary corrupting the prover. Given the auxiliary input z, we construct the simulator $\mathcal{S}$ that holds pk and runs $\mathcal{A}(z)$ as a subroutine while playing the role of the verifier V in the ideal world interacting with the ideal functionality $\mathcal{F}^R_{\mathsf{pvc\text{-}zk}}$ as follows.

Algorithm Judge

Inputs: A public key pk and certificate cert.

1. Parse cert as $(x, \lambda, \sigma, c_{\mathsf{seed}}, \mathsf{seed}, \mathsf{decom}_{\mathsf{seed}}, \hat{\jmath}, \{a_j, \mathsf{trans}_j\}_{j\in[\lambda]})$.
2. If $\mathsf{Verify}_{\mathsf{pk}}((x, \lambda, c_{\mathsf{seed}}, \{a_j, \mathsf{trans}_j\}_{j\in[\lambda]}), \sigma) = 0$, output 0.
3. If $c_{\mathsf{seed}} \neq \mathsf{Com}(\mathsf{seed}; \mathsf{decom}_{\mathsf{seed}})$, output 0.
4. Use seed to derive $e \leftarrow\!\!\$\ \{0,1\}^\lambda$ and $\mathsf{seed}_j \leftarrow\!\!\$\ \{0,1\}^\kappa$ for all $j \in [\lambda]$ as V done in $\Pi^R_{\mathsf{pvc\text{-}zk}}$.
5. Use $\mathsf{seed}_{\hat{\jmath}}$ and $e_{\hat{\jmath}}$ to simulate the execution of V in Π_{OT} based on $\mathsf{trans}_{\hat{\jmath}}$ and derive the output $z_{\hat{\jmath}}^{e_{\hat{\jmath}}}$. If the message that should be sent by V does not match the one in $\mathsf{trans}_{\hat{\jmath}}$, output 0.
6. Check the correctness of $z_{\hat{\jmath}}^{e_{\hat{\jmath}}}$ based on $(x, a_{\hat{\jmath}}, e_{\hat{\jmath}})$ according to Π. If it is correct, output 0. Otherwise, output 1.

1. $\mathcal{S}$ computes the commitment $c_{\mathsf{seed}} \leftarrow \mathsf{Com}(0)$ and sends c_{seed} to $\mathcal{A}$.
2. $\mathcal{S}$ receives c_a.
3. $\mathcal{S}$ uses the simulator $\mathcal{S}_{\mathsf{OT}}$ for Π_{OT} and extracts P^*'s input $\{(z_j^0, z_j^1)\}_{j\in[\lambda]}$ from $\mathcal{A}$.
4. $\mathcal{S}$ receives the signature σ, $\{a_j\}_{j\in[\lambda]}$, and decom_a.
5. $\mathcal{S}$ verifies the signature σ using pk and verify the opening of c_a. If one is invalid, $\mathcal{S}$ sends $\perp$ to $\mathcal{F}^R_{\mathsf{pvc\text{-}zk}}$ and simulate the abortion of V.
   $\mathcal{S}$ initiates an empty list $\mathsf{CheatIndex}$. For each $j \in [\lambda]$, $\mathcal{S}$ verifies the correctness of both z_j^0 and z_j^1 based on x, e_j, and a_j.
   - If both z_j^0 and z_j^1 are accepted, compute w_j, such that $(x, w_j) \in R$.
   - If both z_j^0 and z_j^1 are rejected, $\mathcal{S}$ directly sends $(\texttt{prove}, x, (\texttt{cheat}, 1))$ to $\mathcal{F}^R_{\mathsf{pvc\text{-}zk}}$ and halts.
   - If one is accepted while the other is rejected, let $w_j = \perp$ and store j in the list $\mathsf{CheatIndex}$.

   After the verification, if $\mathsf{CheatIndex}$ is empty, $\mathcal{S}$ sends $(\texttt{prove}, x, w_j)$ for a witness w_j with $j \leftarrow\!\!\$\ [\lambda]$ to $\mathcal{F}^R_{\mathsf{pvc\text{-}zk}}$.
   If $\mathsf{CheatIndex}$ is not empty, let $m = |\mathsf{CheatIndex}|$ be the number of elements in $\mathsf{CheatIndex}$. If $m = \lambda$, then $\mathcal{S}$ sends $(\texttt{prove}, x, (\texttt{cheat}, 1 - 1/2^\lambda))$ to $\mathcal{F}^R_{\mathsf{pvc\text{-}zk}}$ and receives $(\texttt{failed})$ or $(\texttt{successful})$ to conclude the simulation. If $m < \lambda$, $\mathcal{S}$ randomly picks $w_j \neq \perp$. Then $\mathcal{S}$ sends $(\texttt{prove}, x, (\texttt{stupidCheat}, w_j, 1 - 1/2^m))$ to $\mathcal{F}^R_{\mathsf{pvc\text{-}zk}}$ and receives $(\texttt{failed})$ or $(\texttt{successful})$ to conclude the simulation.

Now, it remains to show that the joint distribution of the view of the adversary $\mathcal{A}$ and the output of V in the ideal world is computationally indistinguishable from the view of $\mathcal{A}$ and the output of V in the real world. In the following, we consider a sequence of hybrids, where the output of each hybrid is the view of $\mathcal{A}$ and the output of V.

Hybrid$_0$ This is the execution in the ideal world above. In this hybrid, $\mathcal{S}$ and the verifier V interact with the functionality $\mathcal{F}^R_{\mathsf{pvc\text{-}zk}}$. We inline the actions of $\mathcal{S}$, $\mathcal{F}^R_{\mathsf{pvc\text{-}zk}}$, and V, and rewrite the hybrid as follows.

1. Compute the commitment $c_{\mathsf{seed}} \leftarrow \mathsf{Com}(0)$ and send c_{seed} to $\mathcal{A}$.

2. Receive c_a.
3. Use the simulator $\mathcal{S}_{\mathsf{OT}}$ for λ instances of Π_{OT} and extract P^*'s input $\{(z_j^0, z_j^1)\}_{j\in[\lambda]}$ from $\mathcal{A}$.
4. Receive the signature σ, $\{a_j\}_{j\in[\lambda]}$, and decom_a.
5. Verify the signature σ using pk and verify the opening of c_a. If it is invalid, output $\perp$ with abortion.
   Initiate an empty list $\mathsf{CheatIndex}$. For each $j \in [\lambda]$, verify the correctness of both z_j^0 and z_j^1 based on x, e_j, and a_j.
   - If both z_j^0 and z_j^1 are accepted, compute w_j, such that $(x, w_j) \in R$.
   - If both z_j^0 and z_j^1 are rejected, output `rejected`.
   - If one is accepted while the other is rejected, let $w_j = \perp$ and store j in the list $\mathsf{CheatIndex}$.

   After the verification, if $\mathsf{CheatIndex}$ is empty, output `accepted`.
   If $\mathsf{CheatIndex}$ is not empty, let $m = |\mathsf{CheatIndex}|$ be the number of elements in $\mathsf{CheatIndex}$. If $m = \lambda$, then with probability $1 - 1/2^\lambda$ output `rejected` and with the remaining probability output `accepted`. If $m < \lambda$, then with probability $1 - 1/2^m$ output `rejected` and with the remaining probability output `accepted`.

Hybrid$_1$ We modify the operations after the verification of Step 5: let $m = |\mathsf{CheatIndex}|$ be the number of elements in $\mathsf{CheatIndex}$. If $m = 0$, output `accepted`. Otherwise, with probability $1 - 1/2^m$ output `rejected` and with the remaining probability output `accepted`.
It is easy to see that the distribution for the output of **Hybrid**$_1$ is identical to the distribution for the output of **Hybrid**$_0$.

Hybrid$_2$ We modify Step 5 as follows.

5. Verify the signature σ using pk. If it is invalid, output $\perp$ with abortion.
   Initiate an empty list $\mathsf{CheatIndex}$. Pick $e \leftarrow_{\$} \{0,1\}^\lambda$. For each $j \in [\lambda]$, verify the correctness of $z_j^{e_j}$ based on x, e_j, and a_j.
   - If $z_j^{e_j}$ is rejected, store j in the list $\mathsf{CheatIndex}$.

   After the verification, if $\mathsf{CheatIndex}$ is empty, output `accepted`.
   If $\mathsf{CheatIndex}$ is not empty, let $m = |\mathsf{CheatIndex}|$ be the number of elements in $\mathsf{CheatIndex}$. If $m \neq 0$, then output `rejected`. Otherwise, output `accepted`.

The distribution for the output of **Hybrid**$_2$ is identical to the distribution for the output of **Hybrid**$_1$. To check this fact, we consider the corresponding cases in **Hybrid**$_1$ and show that they are identical to the cases in **Hybrid**$_2$ in the following.

- In **Hybrid**$_1$, if all z_j^0 and z_j^1 are accepted, output `accepted`. In **Hybrid**$_2$, the output is also `accepted` in this case.
- In **Hybrid**$_1$, if there exists an index j, such that both z_j^0 and z_j^1 are rejected, output `rejected`. In **Hybrid**$_2$, no matter what e_j is chosen, the output is also `rejected`.
- In **Hybrid**$_1$, if there are m instances of z_j^0 and z_j^1, such that one is accepted while the other is rejected, with probability $1-1/2^m$, the output is `rejected`. In **Hybrid**$_2$, since e is uniformly chosen, the probability that the output is `rejected` is also $1 - 1/2^m$.

Therefore, the distribution for the output of $\mathbf{Hybrid}_2$ is identical to the distribution for the output of $\mathbf{Hybrid}_1$.

$\mathbf{Hybrid}_3$ In this hybrid, we modify Step 1: Choose $\mathsf{seed} \leftarrow_\$ \{0,1\}^\kappa$ and use seed to derive a string $e \leftarrow_\$ \{0,1\}^\lambda$ and $\mathsf{seed}_j \leftarrow_\$ \{0,1\}^\kappa$ for all $j \in [\lambda]$ as in the protocol. Then compute the commitment $c_{\mathsf{seed}} \leftarrow \mathsf{Com}(0)$ and send c_{seed} to $\mathcal{A}$. Since the committed value is unchanged, c_{seed} is identical to that in $\mathbf{Hybrid}_2$. The challenge e is now derived from seed. Since e now is pseudorandom, the output of $\mathbf{Hybrid}_3$ is computationally indistinguishable from the output of $\mathbf{Hybrid}_2$.

$\mathbf{Hybrid}_4$ We modify Step 1 by computing c_{seed} as $c_{\mathsf{seed}} \leftarrow \mathsf{Com}(\mathsf{seed})$. Since the commitment scheme is (computationally) hiding, the output of $\mathbf{Hybrid}_4$ is (computationally) indistinguishable from the output of $\mathbf{Hybrid}_3$.

$\mathbf{Hybrid}_5$ In Step 3 of this hybrid, execute λ instances of Π_{OT} honestly with true randomness instead of using $\mathcal{S}_{\mathsf{OT}}$. Based on the security of Π_{OT}, the output of $\mathbf{Hybrid}_5$ is computationally indistinguishable from the output of $\mathbf{Hybrid}_4$.

$\mathbf{Hybrid}_6$ In this hybrid, the randomness used in the execution Π_{OT} are derived from seed_j. Since the randomness used in Π_{OT} is now pseudorandom instead of true randomness, the output of $\mathbf{Hybrid}_6$ is computationally indistinguishable from the output of $\mathbf{Hybrid}_5$. We state $\mathbf{Hybrid}_6$ as follows.

1. Choose $\mathsf{seed} \leftarrow_\$ \{0,1\}^\kappa$ and use seed to derive a string $e \leftarrow_\$ \{0,1\}^\lambda$ and $\mathsf{seed}_j \leftarrow_\$ \{0,1\}^\kappa$ for all $j \in [\lambda]$. Compute $c_{\mathsf{seed}} \leftarrow \mathsf{Com}(\mathsf{seed})$ and send c_{seed} to $\mathcal{A}$.
2. Receive c_a.
3. Execute λ instances of Π_{OT} with $\mathcal{A}$ using input $\{e_j\}_{j\in[\lambda]}$ with randomness derived from seed_j and receive $\{z_j^{e_j}\}_{j\in[\lambda]}$.
4. Receive the signature σ, $\{a_j\}_{j\in[\lambda]}$, and decom_a.
5. Verify the signature σ using pk and verify the opening of c_a. If it is invalid, output $\perp$ with abortion. For each $j \in [\lambda]$, verify the correctness of $z_j^{e_j}$ based on x, e_j, and a_j. If all $\{z_j^{e_j}\}$ are correct, output `accepted`. Otherwise, output `rejected`.

It is easy to see that $\mathbf{Hybrid}_6$ is identical to the execution of the protocol for an honest verifier V. Since $\mathbf{Hybrid}_6$ is computationally indistinguishable from $\mathbf{Hybrid}_0$, the joint distribution of the view of the adversary $\mathcal{A}$ and the output of V in the ideal world is computationally indistinguishable from the view of $\mathcal{A}$ and the output of V in the real world.

We now focus on the case where the verifier V is corrupted by the adversary $\mathcal{S}$. Given the auxiliary input z, we construct the simulator $\mathcal{S}$ that runs $\mathcal{A}(z)$ as a subroutine while playing the role of the prover P in the ideal world interacting with the ideal functionality $\mathcal{F}^R_{\mathsf{pvc\text{-}zk}}$ as follows.

0. $\mathcal{S}$ generates a pair of keys $(\mathsf{pk}, \mathsf{sk})$ for the signature scheme using KGen and sends pk to $\mathcal{A}$.
1. If $\mathcal{S}$ receives $\perp$ from $\mathcal{F}^R_{\mathsf{pvc\text{-}zk}}$, $\mathcal{S}$ simulates the abortion of P in the protocol. Otherwise, $\mathcal{S}$ receives c_{seed} from $\mathcal{A}$.
2. $\mathcal{S}$ generates $c_a \leftarrow \mathsf{Com}(0)$ and sends it to $\mathcal{A}$.

3. $\mathcal{S}$ uses the simulator $\mathcal{S}_{\mathsf{OT}}$ for Π_{OT} and extracts the challenge $\{e_j\}_{j\in[\lambda]}$. Given e_j, $\mathcal{S}$ uses the simulator $\mathcal{S}_\Sigma$ for the protocol Π to generate the accepting transcript (a_j, e_j, z_j). Then $\mathcal{S}$ returns $\{z_j\}_{j\in[\lambda]}$ as output of Π_{OT} to $\mathcal{A}$.
4. $\mathcal{S}$ generates the signature σ as in the protocol. $\mathcal{S}$ equivocates the commitment c_a to be the commitment to $\{a_j\}_{j\in[\lambda]}$. Then $\mathcal{S}$ sends σ, $\{a_j\}_{j\in[\lambda]}$, and the decommitment of c_a to $\mathcal{A}$ to conclude the simulation.

Now it remains to show that the joint distribution of the view of the adversary $\mathcal{A}$ and the output of P (P may output $\perp$) in the ideal world is computationally indistinguishable from the view of $\mathcal{A}$ and the output of P in the real world. In the following, we consider a sequence of hybrids, where the output of each hybrid is the view of $\mathcal{A}$ and the output of P. We first denote the simulation in the ideal world above by $\mathbf{Hybrid}_0$.

$\mathbf{Hybrid}_1$ Assume that $\mathcal{S}$ is given the witness w. Then, if $(x, w) \notin R$, $\mathcal{S}$ simulates the abortion of the prover in Step 1. It is easy to see that the distribution of the output for $\mathbf{Hybrid}_1$ is identical to the distribution for the output of $hybrid_0$.

$\mathbf{Hybrid}_2$ In Step 4, $\mathcal{S}$ generates the accepting transcripts for the cases $e_j = 0$ and $e_j = 1$, *i.e.*, $(a_j^0, 0, z_j^0)$ and $(a_j^1, 1, z_j^1)$, and returns the corresponding $\{z_j^{e_j}\}_{j\in[\lambda]}$ to $\mathcal{A}$. Then in Step 4, $\mathcal{S}$ equivocates c_a to be the commitment to $\{a_j^{e_j}\}_{j\in[\lambda]}$. Since the transcripts for $1-e_j$'s are not used, it is easy to see that the distribution of the output for $\mathbf{Hybrid}_2$ is identical to the distribution for the output of $hybrid_1$.

$\mathbf{Hybrid}_3$ For this hybrid, $\mathcal{S}$ generates a_j for all $j \in [\lambda]$. Then in Step 3, $\mathcal{S}$ computes z_j^0 and z_j^1 as in the protocol based on a_j. Since for the protocol Π, each accepting transcript (a_j, e_j, z_j) simulated by $\mathcal{S}_\Sigma$ is (computationally) indistinguishable from that in a real execution and $\mathcal{A}$ only receive $z_j^{e_j}$ for each $j \in [\lambda]$, the output of $\mathbf{Hybrid}_3$ is (computationally) indistinguishable from the output of $\mathbf{Hybrid}_2$.

$\mathbf{Hybrid}_4$ $\mathcal{S}$ generates c_a honestly as in the protocol based on $\{a_j\}_{j\in[\lambda]}$. Now, $\mathcal{S}$ does not need to equivocate c_a in Step 4. Based on the security of the commitment scheme, the output of $\mathbf{Hybrid}_4$ is computationally indistinguishable from the output of $\mathbf{Hybrid}_3$.

$\mathbf{Hybrid}_5$ In this hybrid, $\mathcal{S}$ honestly executes the protocol Π_{OT}. We restate this hybrid as follows.

0. Generate a pair of keys $(\mathsf{pk}, \mathsf{sk})$ for the signature scheme using KGen and send pk to $\mathcal{A}$.
1. If $(x, w) \notin R$, abort. Otherwise, receive c_{seed} from $\mathcal{A}$.
2. For each $j \in [\lambda]$, initiate an instance of Π and generate a_j. Compute $c_a \leftarrow \mathsf{Com}(\{a_j\}_{j\in[\lambda]})$ and sends it to $\mathcal{A}$.
3. For each $j \in [\lambda]$, compute z_j^0 and z_j^1 for different challenge 0 and 1, respectively, according to Π. Execute λ Π_{OT} with $\mathcal{A}$ using input (z_j^0, z_j^1).
4. Generate the signature σ as in the protocol. Then send σ, $\{a_j\}_{j\in[\lambda]}$, and the decommitment of c_a to $\mathcal{S}$ to conclude the simulation.

It is easy to see that $\mathbf{Hybrid}_5$ is identical to the execution of the protocol for an honest prover P. Since $\mathbf{Hybrid}_5$ is computationally indistinguishable from

$\mathbf{Hybrid}_0$, the joint distribution of the view of the adversary $\mathcal{A}$ and the output of P in the ideal world is computationally indistinguishable from the view of $\mathcal{A}$ and the output of P in the real world. Therefore, the simulatability of the protocol is proved.

Public Verifiability. We now argue that whenever an honest verifier V outputs `rejected`, V is capable of generating a valid certificate for the malicious prover P. In Step 5 of the protocol, without receiving a valid signature and a valid opening of c_a, V will not continue the execution of the protocol. At this point, given the security of Π_{OT}, a malicious P cannot make her decision to continue the protocol (*i.e.*, provide a valid signature and opening) or abort based on the challenge e. When the malicious P chooses not to abort and proceeds to provide the signature and opening, V now possesses sufficient information to generate the certificate. Subsequently, it can be readily verified that the algorithm Judge is applicable to a certificate cert generated by an honest V in the protocol using Blame. This verification is feasible since the algorithm Judge simulates the operations carried out by V and employs the same verification procedure as V in the protocol.

Defamation Freeness. We show that the protocol achieves defamation freeness by contradiction. We assume that an honest prover P is accused by a verifier V corrupted by the adversary via a valid certificate. This implies that the adversary can provide a valid signature of P for the message

$$(x, \lambda, c_{\mathsf{seed}}, \{a_j, \mathsf{trans}_j\}_{j\in[\lambda]}) .$$

Since P is honest, she only generates a signature for the message she has seen and generated. Considering the EUF-CMA-secure signature scheme, the adversary cannot forge the signature except with negligible probability. Furthermore, as the commitment scheme is binding, the adversary cannot provide a different opening for c_{seed} from the one used in Step 1 to generate the signature. Therefore, the simulation performed in the algorithm Judge will employ the exact seed. Given that Π_{OT} is perfectly correct, with the signed transcript of Π_{OT}, there exists exactly one valid output for V consistent with $\mathsf{trans}_{\hat{\jmath}}$, regardless of V's randomness and input. Thus, the simulated transcript in the algorithm Judge will match $\mathsf{trans}_{\hat{\jmath}}$ only if $\mathsf{seed}_{\hat{\jmath}}$ is the one used by V to derive randomness during the execution of Π_{OT}. In this scenario, the simulation will yield the same $z_{\hat{\jmath}}^{e_{\hat{\jmath}}}$ generated by the honest P, and consequently, the algorithm Judge will not output 1. Hence, the protocol achieves defamation freeness, and a malicious verifier cannot generate a valid certificate cert to blame an honest prover except with negligible probability.

In summary, if Π is a Sigma protocol with 1-bit challenge for the NP-realation R, then the protocol $\Pi^R_{\mathsf{pvc\text{-}zk}}$ along with the algorithms Blame and Judge is a zero-knowledge protocol for the NP-realation R that achieves PVC security with deterrence $\epsilon = 1 - 1/2^{\lambda}$.

Remark 1. It is worth mentioning that extending the challenge space of a Sigma protocol Π in Definition 2 to accommodate other small challenge spaces can be easily accomplished, along with slight modifications to $\Pi^R_{\mathsf{pvc\text{-}zk}}$ to support such Sigma protocols.

5 Performance

In this section, we conduct a performance analysis of our protocol $\Pi^R_{\mathsf{pvc\text{-}zk}}$ in comparison to an original protocol, denoted as Π_o, which repetitively executes the Sigma protocol Π with 1-bit challenge to minimize the knowledge error.

It is important to note that traditionally, Π_o requires the execution of at least 40 instances of Π. However, in scenarios involving reputation-conscious parties, our $\Pi^R_{\mathsf{pvc\text{-}zk}}$ could significantly reduce the number of instances required. Table 1 presents the required number of instances for Π to achieve specific deterrence factors. Although our protocol requires the prover to perform the OT protocol, sending two answers for the challenge bit (while the verifier only derives one answer), the communication cost of each instance in comparison to Π is at most doubled, considering that only the number of answers is doubled. Moreover, it is important to mention that the commitment and signature have constant sizes, as we employ a random oracle to instantiate the commitment scheme and compress the signed messages. Additionally, the OT protocol is quite efficient nowadays [18]. For the number of rounds, assuming that the random OTs have already been generated[3], our protocol consists of 5 rounds. In comparison, the constant-round Goldreich-Kahan protocol [13, 23] also consists of 5 rounds. Thus, the cost of our protocol is primarily determined by the cost of Π and the number of instances required for repetition.

Table 1. Number of instances for Π required to achieve the specified deterrence factors.

Deterrence Factors	50%	75%	87.5%	93.75%	96.88%	98.44%	99.22%
Number of Instances	1	2	3	4	5	6	7

Therefore, we observe that even achieving a deterrence factor of 99.22%, our protocol only requires 7 instances, resulting in a cost increase of at most 14× compared to Π. In contrast, the traditional protocol incurs a cost of at least 40×. Consequently, our protocol $\Pi^R_{\mathsf{pvc\text{-}zk}}$ costs less than 35% of Π_o in the scenarios involving reputation-conscious parties. For a deterrence factor of 93.75%, the cost of $\Pi^R_{\mathsf{pvc\text{-}zk}}$ is approximately 20% of the cost of Π_o. This substantial reduction in cost proves particularly advantageous in real-world scenarios.

[3] It is common practice to pre-generate random OTs before executing the protocol. Since zero-knowledge protocols are often used as sub-protocols, it is reasonable to assume that random OTs have been generated.

6 Conclusion

In this paper, we explore the approach of integrating zero-knowledge protocols into the PVC security model to address the significant overhead challenges present in many useful and interesting zero-knowledge protocols. Specifically, we introduce a new notion called zero-knowledge protocols with PVC security, which allows honest parties to generate publicly verifiable certificates against detected misbehaviors. This approach offers the potential to simultaneously achieve high efficiency and strong security guarantees, effectively balancing efficiency and security. Building on the notion, we propose a generic transformation that converts any Sigma protocol with 1-bit challenge into a zero-knowledge protocol with PVC security. Our transformation achieves a high cheating detection accuracy (e.g. 99.22%) while significantly reducing overhead (e.g. by approximately 65%) compared to the original protocol, thereby contributing to significant advancements in the practical application of zero-knowledge protocols.

7 Acknowledgements

This work was supported in part by the National Key Research and Development Program of China under Grant No. 2021ZD0112802, in part by the National Natural Science Foundation of China under Grant Nos. 62302194, 62072215 and 62472198, in part by Guangdong Basic and Applied Basic Research Foundation under Grant Nos. 2023B1515040020 and 2019B030302008, in part by Guangzhou Basic and Applied Basic Research Foundation under Grant Nos. 2025A04J2146 and 2024A04J3458, in part by the Guangzhou Basic Research Plan City-School Joint Funding Project under Grant No. 2024A03J0405, and in part by National Joint Engineering Research Center of Network Security Detection and Protection Technology, Guangdong Key Laboratory of Data Security and Privacy Preserving, Guangdong Hong Kong Joint Laboratory for Data Security and Privacy Protection, and Engineering Research Center of Trustworthy AI, Ministry of Education.

References

1. Agrawal, N., Bell, J., Gascón, A., Kusner, M.J.: Mpc-friendly commitments for publicly verifiable covert security. In: Kim, Y., Kim, J., Vigna, G., Shi, E. (eds.) CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021. pp. 2685–2704. ACM, New York, New York, USA (2021)
2. Asharov, G., Orlandi, C.: Calling out cheaters: Covert security with public verifiability. In: Wang, X., Sako, K. (eds.) Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Proceedings. Lecture Notes in Computer Science, vol. 7658, pp. 681–698. Springer, Beijing, China (2012)

3. Attema, T., Dunning, V., Everts, M.H., Langenkamp, P.: Efficient compiler to covert security with public verifiability for honest majority MPC. In: Ateniese, G., Venturi, D. (eds.) Applied Cryptography and Network Security - 20th International Conference, ACNS 2022, Proceedings. Lecture Notes in Computer Science, vol. 13269, pp. 663–683. Springer, Rome, Italy (2022)
4. Ben-Sasson, E., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., Virza, M.: Zerocash: Decentralized anonymous payments from bitcoin. In: 2014 IEEE Symposium on Security and Privacy, SP. pp. 459–474. IEEE Computer Society, Berkeley, CA, USA (2014)
5. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: Short proofs for confidential transactions and more. In: 2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings. pp. 315–334. IEEE Computer Society, San Francisco, California, USA (2018)
6. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: 42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001. pp. 136–145. IEEE Computer Society, Las Vegas, Nevada (2001)
7. Couteau, G., Peters, T., Pointcheval, D.: Encryption switching protocols. In: Robshaw, M., Katz, J. (eds.) Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9814, pp. 308–338. Springer, Santa Barbara, CA, USA (2016)
8. Damgård, I.: On σ-protocols. Lecture Notes, University of Aarhus, Department for Computer Science (2002)
9. Damgård, I., Orlandi, C., Simkin, M.: Black-box transformations from passive to covert security with public verifiability. In: Micciancio, D., Ristenpart, T. (eds.) Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Proceedings, Part II. Lecture Notes in Computer Science, vol. 12171, pp. 647–676. Springer, Santa Barbara, CA, USA (2020)
10. Faust, S., Hazay, C., Kretzler, D., Schlosser, B.: Generic compiler for publicly verifiable covert multi-party computation. In: Canteaut, A., Standaert, F. (eds.) Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Proceedings, Part II. Lecture Notes in Computer Science, vol. 12697, pp. 782–811. Springer, Zagreb, Croatia (2021)
11. Faust, S., Hazay, C., Kretzler, D., Schlosser, B.: Financially backed covert security. In: Hanaoka, G., Shikata, J., Watanabe, Y. (eds.) Public-Key Cryptography - PKC 2022 - 25th IACR International Conference on Practice and Theory of Public-Key Cryptography, Virtual Event, March 8-11, 2022, Proceedings, Part II. Lecture Notes in Computer Science, vol. 13178, pp. 99–129. Springer, Berlin (2022)
12. Gennaro, R., Gentry, C., Parno, B., Raykova, M.: Quadratic span programs and succinct nizks without pcps. In: Johansson, T., Nguyen, P.Q. (eds.) Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Proceedings. Lecture Notes in Computer Science, vol. 7881, pp. 626–645. Springer, Athens, Greece (2013)
13. Goldreich, O., Kahan, A.: How to construct constant-round zero-knowledge proof systems for NP. J. Cryptol. **9**(3), 167–190 (1996)
14. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or A completeness theorem for protocols with honest majority. In: Aho, A.V. (ed.) Proceedings of the 19th Annual ACM Symposium on Theory of Computing. pp. 218–229. ACM, New York, New York, USA (1987)

15. Goldreich, O., Micali, S., Wigderson, A.: Proofs that yield nothing but their validity for all languages in NP have zero-knowledge proof systems. J. ACM **38**(3), 691–729 (1991)
16. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems (extended abstract). In: Sedgewick, R. (ed.) Proceedings of the 17th Annual ACM Symposium on Theory of Computing. pp. 291–304. ACM, Providence, Rhode Island, USA (1985)
17. Groth, J.: On the size of pairing-based non-interactive arguments. In: Fischlin, M., Coron, J. (eds.) Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Proceedings, Part II. Lecture Notes in Computer Science, vol. 9666, pp. 305–326. Springer, Vienna, Austria (2016)
18. Guo, C., Katz, J., Wang, X., Weng, C., Yu, Y.: Better concrete security for half-gates garbling (in the multi-instance setting). In: Micciancio, D., Ristenpart, T. (eds.) Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Proceedings, Part II. Lecture Notes in Computer Science, vol. 12171, pp. 793–822. Springer, Santa Barbara, CA, USA (2020)
19. Hazay, C., Lindell, Y.: Efficient Secure Two-Party Protocols - Techniques and Constructions. Information Security and Cryptography, Springer, Berlin (2010)
20. Hazay, C., Lindell, Y.: A note on zero-knowledge proofs of knowledge and the ZKPOK ideal functionality. IACR Cryptol. ePrint Arch. (2010), http://eprint.iacr.org/2010/552
21. Hong, C., Katz, J., Kolesnikov, V., Lu, W., Wang, X.: Covert security with public verifiability: Faster, leaner, and simpler. In: Ishai, Y., Rijmen, V. (eds.) Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Proceedings, Part III. Lecture Notes in Computer Science, vol. 11478, pp. 97–121. Springer, Darmstadt, Germany (2019)
22. Kolesnikov, V., Malozemoff, A.J.: Public verifiability in the covert model (almost) for free. In: Iwata, T., Cheon, J.H. (eds.) Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Proceedings, Part II. Lecture Notes in Computer Science, vol. 9453, pp. 210–235. Springer, Auckland, New Zealand (2015)
23. Lindell, Y.: How to simulate it - A tutorial on the simulation proof technique. In: Lindell, Y. (ed.) Tutorials on the Foundations of Cryptography, pp. 277–346. Springer International Publishing, Berlin (2017)
24. Liu, Y., Lai, J., Wang, Q., Qin, X., Yang, A., Weng, J.: Robust publicly verifiable covert security: Limited information leakage and guaranteed correctness with low overhead. In: Guo, J., Steinfeld, R. (eds.) Advances in Cryptology - ASIACRYPT 2023 - 29th International Conference on the Theory and Application of Cryptology and Information Security, Proceedings, Part I. Lecture Notes in Computer Science, vol. 14438, pp. 272–301. Springer, Guangzhou, China (2023)
25. Liu, Y., Wang, Q., Yiu, S.M.: Blind polynomial evaluation and data trading. In: Conti, M., Zhou, J., Casalicchio, E., Spognardi, A. (eds.) Applied Cryptography and Network Security - 19th International Conference, ACNS 2021. Lecture Notes in Computer Science, Springer, Kamakura, Japan (2021)
26. Liu, Y., Wang, Q., Yiu, S.: Making private function evaluation safer, faster, and simpler. In: Hanaoka, G., Shikata, J., Watanabe, Y. (eds.) Public-Key Cryptography - PKC 2022 - 25th IACR International Conference on Practice and Theory of Public-Key Cryptography, Virtual Event, March 8-11, 2022, Proceedings, Part

I. Lecture Notes in Computer Science, vol. 13177, pp. 349–378. Springer, Berlin (2022)
27. Parno, B., Howell, J., Gentry, C., Raykova, M.: Pinocchio: Nearly practical verifiable computation. In: 2013 IEEE Symposium on Security and Privacy, SP 2013. pp. 238–252. IEEE Computer Society, Berkeley, CA, USA (2013)
28. Schnorr, C.: Efficient identification and signatures for smart cards. In: Brassard, G. (ed.) Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Proceedings. Lecture Notes in Computer Science, vol. 435, pp. 239–252. Springer, Santa Barbara, California, USA (1989)
29. Scholl, P., Simkin, M., Siniscalchi, L.: Multiparty computation with covert security and public verifiability. Cryptology ePrint Archive, Report 2021/366 (2021), https://ia.cr/2021/366
30. Zhu, R., Ding, C., Huang, Y.: Efficient publicly verifiable 2pc over a blockchain with applications to financially-secure computations. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019. pp. 633–650. ACM, London, UK (2019)