Steganography-Free Zero-Knowledge

Behzad Abdolmaleki[1], Nils Fleischhacker[2], Vipul Goyal[3], Abhishek Jain[4], and Giulio Malavolta[1]

[1] Max Planck Institute for Security and Privacy, Bochum, Germany
{behzad.abdolmaleki,giulio.malavolta}@mpi-sp.org
[2] Ruhr University Bochum, Germany
mail@nilsfleischhacker.de
[3] NTT Research and Carnegie Mellon University, USA
vipul@cmu.edu
[4] Johns Hopkins University, Baltimore, USA
abhishek@cs.jhu.edu

Abstract. We revisit the well-studied problem of preventing steganographic communication in multi-party communications. While this is known to be a provably impossible task, we propose a new model that allows circumventing this impossibility. In our model, the parties first publish a single message during an honest *non-interactive* pre-processing phase and then later interact in an execution phase. We show that in this model, it is indeed possible to prevent any steganographic communication in zero-knowledge protocols. Our solutions rely on standard cryptographic assumptions.

1 Introduction

Consider the following scenario: a computer at a government agency storing highly classified data has been infected with a stealthy malware. The malware's main purpose is to communicate the classified data to an attacker on the Internet. To minimize the possibility of being detected and quarantined, the malware has been designed to stealthily "encode" the secret data in ordinary communication between the infected computer and the outside world. This may include communication with "honest" entities on the Internet or potentially even the attacker (disguised as an honest user). An intriguing question, which forms the basis of the present work, is whether it is possible to detect such communication?

The above scenario is representative of a broader theme concerning *steganographic communication*, where a party A wants to transmit a secret message to another party B by communicating over a public broadcast channel without being detected by an external *observer* who is listening on the channel. Since the use of an encrypted channel can be easy to detect, A may instead try to embed its message in an innocuous-looking conversation. For example, [34], it may send a photograph of a person to securely transmit bit 0 if the 30th hair from the left is white, and 1 otherwise.

A sequence of works [14, 31, 29, 3] have established that such steganographic communication is always possible in any system with some entropy, and is provably *impossible* to detect. As such, it may seem that the answer to the aforementioned question is negative.

A New Model for Preventing Steganography. In this work, we propose a new model for circumventing the aforementioned impossibility result. In our model, any communication (via an interactive protocol) proceeds in two phases: a *non-interactive pre-processing* phase and an *execution* phase. Each party publishes a single message during the pre-processing phase, while the execution phase corresponds to the actual protocol execution. We assume that the parties are honest during the pre-processing phase, but may be completely malicious during the execution phase. *Our main goal is to ensure that any attempts at steganographic communication during the execution phase will be detected by the external observer.*

We, in fact, consider a stronger model where *only one of the parties is required to be honest* during the pre-processing phase. In this case, the malicious parties may be able to subliminally embed information in their pre-processing messages. However, we require that such subliminal communication is limited to the (non-interactive) pre-processing and that no steganographic communication can be performed during the execution phase. Our model is meaningful in our motivating example: if the pre-processing step is executed before the computer is infected, then it ensures that no information can be later leaked by the malware without being detected.

Let us now explain why the pre-processing model can help in preventing steganography. As observed in many prior works, the key source of the problem is that the parties' algorithms may be *randomized*, which opens an avenue for subliminal communication. Removing the use of randomness altogether does not yield a solution since randomness is necessary for most of cryptography [21]. The pre-processing model helps resolve this dilemma. The main insight is that the pre-processing step can be used to "fix" the randomness of the parties, thereby forcing them to become *deterministic* during the execution phase. If the parties deviate from the prescribed strategy, they can be detected by the observer.

A common method to detect deviation from prescribed strategy in any protocol is to use zero-knowledge (ZK) proofs [27], à la Goldreich, Micali, Wigderson (GMW) compiler [26]. However, ZK proofs themselves require randomness [21]. As such, a priori, it might not be clear how to implement the above idea.

1.1 Our Contribution

We present a general method for preventing steganographic communication in interactive protocols.

Defining Steganography Freeness. We start by defining *steganography freeness* for generic interactive protocols (S, R) in the non-interactive pre-processing model. Intuitively, our notion requires that no adversarial sender S can steganographically communicate even a single bit of information to the receiver R during

the execution phase as long as at least one of them was honest during the pre-processing phase. We formalize this via a game-based definition (Section 3) where at the start of the execution phase, the adversarial sender is given a randomly chosen bit b. We require that at the end of the execution phase, the probability that the receiver correctly guesses b and the execution transcript is accepted by the observer is only negligibly more than one half.

Steganography-Free Zero-Knowledge. Our main tool for achieving steganography freeness in a generic interactive protocol is a new notion of *steganography-free zero-knowledge* (SF-ZK). An SF-ZK argument proceeds in two phases: first, the prover and the verifier participate in a non-interactive pre-processing step where they send a single message to each other. This step is executed *before* the prover receives the statement and the witness. Next, the prover and the verifier participate in the execution phase where the prover proves the validity of the statement.

An SF-ZK argument system must satisfy the standard completeness, soundness, and ZK properties. In particular, soundness (resp. ZK) must hold even if the prover (resp. verifier) is malicious both during the pre-processing as well as the execution phase. Further, SF-ZK must satisfy two new security properties:

- *Observer Soundness:* This property states that for any *false* statement, no coalition of prover and verifier can produce a transcript that will be accepted by the external observer as long as either the prover or the verifier was honest during the pre-processing phase.
- *Computationally Unique Transcripts (CUT):* We define this property w.r.t. languages $\mathcal{L}$ with unique witnesses; however, it can be naturally extended to the multiple witnesses case. Intuitively, it states that once the pre-processing phase has been executed (where either the prover or the verifier was honest), then for any statement $x \in \mathcal{L}$, two different sets of efficient prover and verifier strategies cannot produce two different transcripts of the execution phase that will both be accepted by the observer.

We show that the CUT property implies steganography freeness. Further, we note that the observer soundness property is crucial in natural applications of SF-ZK. Indeed, if we use SF-ZK to implement a GMW-style compiler for constructing steganography-free protocols, then observer soundness would be necessary to ensure that an adversarial party cannot deviate from a prescribed strategy in the underlying protocol and therefore cannot use the execution transcript to perform steganographic communication.

We refer the reader to Section 3.1 for a formal definition of SF-ZK.

Positive Results. We construct an SF-ZK argument system with black-box simulation for all languages in NP. We, in fact, provide two constructions: first, assuming sub-exponentially hard injective one-way functions, we devise a solution in the *single-execution* setting, where the pre-processing phase can only be used once. Then, assuming the existence of fully homomorphic encryption [24], we present a solution in the *multi-execution* setting, where the pre-processing can be refreshed to allow for an unbounded number of execution phases.

Our construction of SF-ZK directly works for circuit satisfiability and avoids any use of expensive NP reductions. In Section 4, we provide a construction of SF-ZK in the single-execution setting. While this protocol follows a conceptually clean approach, it involves a computationally expensive sub-protocol where the prover is required to give a "proof of proof," namely, proof of honest behavior in the execution of another proof. To obtain a more efficient solution, we also present another construction that follows the same key ideas as in our first construction but avoids the expensive sub-protocol by instead using *cut-and-choose* techniques [37].

In the full version of the paper, we extend our construction of SF-ZK to the multi-execution setting.

Optimality of our Model. In the full version, we show that our adversarial model is "tight". Specifically, we show that when *both* the prover and the verifier are malicious during the pre-processing, SF-ZK is *impossible*, except for languages in BPP.

1.2 Applications

In the following we highlight a few interesting applications of SF-ZK.

Online Games. Imagine a group of players that want to engage in a game of poker without a trusted dealer. The standard solution for this is to use a multi-party computation (MPC) protocol to simulate a dealer by combining the randomness of all players. MPC is however an inherently randomized machinery and the same randomness could be used by colluding players to communicate information (say, about their hands) in an undetectable way. This problem was considered in [34], where the authors proposed a solution based on generic MPC together with unique ZK proof.[5] Their solution relies on players physically exchanging sealed envelopes prior to the execution of the protocol and hence cannot be used over the internet (see Section 1.4 for a more detailed comparison).

In contrast, using SF-ZK allows us to bypass any physical interaction among participants at the cost of a non-interactive pre-processing phase. The resulting protocol is sanitized from any covert communication, since transmitting information covertly via SF-ZK is computationally hard.

Private Classifier. Consider the scenario where a server holds a trained classifier and wants to give clients oracle access to the prediction without revealing the logic implemented by the predictor. At the same time, the client wants to be assured that the answers of the server are consistent and indeed correspond to the output of the classifier. An obvious solution to this problem is to augment the client-server interaction with a standard ZK proof of correctness.

Consider the event the server gets infected by a virus. The malicious program might instruct the machine to simply output the full description of the classifier. However, such behavior is easy to detect for anyone observing the network traffic.

[5] In unique ZK only a single valid proof exists for a given statement-witness pair.

What if the virus implements a more clever strategy: use the ZK proof as a vector to slowly exfiltrate secret information? Since ZK proofs must be randomized, there is plenty of room to transmit information in an undetectable manner.

One solution is to use SF-ZK instead: the (computational) uniqueness of the transcripts ensures that the virus cannot embed information in the randomness of the protocol and observer soundness forces the server to behave correctly. That is, whatever the client can learn from an infected machine he can also learn by honest queries to the non-corrupted server. Note that in this scenario we can assume that the server is not infected during the training of the model, which can be paired with the computation of the honest prover pre-processing.

A similar argument applies to any interaction in the client-server setting where the server holds some amount of secret data (e.g., a password file, or, classified emails) and might get infected with a virus.

1.3 Our Techniques

In this section, we provide an overview of the main ideas underlying our constructions of SF-ZK, both in the single-execution and multi-execution settings.

How to Simulate? We start by describing a key conceptual challenge in constructing SF-ZK. Recall that a black-box simulator works by rewinding the adversarial verifier potentially multiple times. This involves creating multiple protocol transcripts which are necessarily different (for the rewinding to be "successful"). This seems to be at odds with the computationally unique transcripts (CUT) property of SF-ZK; indeed, since the simulator is also an efficient algorithm, intuitively, it should also not be able to produce multiple transcripts of the execution phase. This presents a catch-22: how can we achieve ZK property without violating the CUT property (or vice-versa)?

Towards resolving this conundrum, recall that the CUT property is required to hold against two *different* pairs of prover and verifier strategies (P_1, V_1) and (P_2, V_2), who cannot communicate with each other. This rules out *oblivious* black-box simulation strategies that involve running multiple execution threads (with a common prefix) in parallel since such a strategy implies multiple transcript choices during an honest execution. However, it does *not* rule out *non-oblivious* black-box simulation strategies. In particular, *a non-oblivious simulator can potentially create a transcript, and then use information learned from that transcript to create another one.* This does not violate the CUT property but opens up an avenue for black-box simulation.

Starting Approach. To explain our approach, let us first recall the notion of *delayed-input* witness indistinguishable (WI) proofs, where the statement and the witness is only required for computing the last prover message. Such proofs are known in three rounds with a public-coin verifier based on one-way func-

tions [33]. In particular, a recent work of [28] constructed such proofs for circuit satisfiability[6] based on garbled circuits.

Now consider the following template for SF-ZK: during the pre-processing phase, the prover publishes the first message α of the delayed-input WI and additionally commits to some randomness (say) r. The verifier commits in advance to the second (public-coin) message β of the WI and additionally publishes a "trapdoor" statement with a (verifiably) unique witness. Both the prover and verifier use a non-interactive commitment scheme with unique decommitment[7] to compute their respective commitments.

At the start of the execution phase, both the prover and the verifier receive the statement x and the prover additionally receives a (unique) witness w. The execution phase proceeds as follows:

- The prover first simply sends a commitment c to 0 using randomness r.
- Next, the verifier decommits to the second message of WI and additionally reveals the (unique) witness for the trapdoor statement.
- Finally, the prover sends the third message γ of the WI proof to prove the statement: "either x is true or I committed to the trapdoor witness in c using randomness r that was committed in the pre-processing".

Let us now see why the above template enables black-box simulation. A simulator can first produce a partial transcript of the execution phase by simply committing to 0 in c and then learn the witness for the trapdoor statement. Now, the simulator can rewind the verifier to the start of the execution phase and generate a new transcript where it commits to the trapdoor witness. It then continues the computation of the second transcript and produces the WI proof using the second branch of the statement. Note that the simulator can use the second branch in the WI because it is now true.

Challenges with CUT. In order to achieve the CUT property, we require the delayed-input WI proof to have a *unique* accepting third message γ for a fixed partial transcript (α, β) and a fixed statement and witness. Towards this, let us briefly recall the construction of [28]. Below, we describe the basic version which achieves soundness one half; the full protocol with negligible soundness error is achieved by parallel repetition of the basic protocol.

- First, the prover computes and sends a garbled circuit for the NP verification circuit. Additionally, it commits to all the wire labels of the garbled circuit.
- Next, the verifier sends a random challenge bit.
- If the challenge bit is 0, the prover "opens" everything by revealing its random tape, otherwise, it decommits to wire labels corresponding to the statement and the witness. In the latter case, the verifier simply evaluates the garbled circuit to check if its output is accepting.

[6] The choice of circuit satisfiability as the language is not arbitrary. We use it to avoid the potential issue of using NP reductions that do not preserve the number of witnesses, which can open up an avenue for subliminal communication.

[7] Such schemes are known based on injective one-way functions.

At a first glance, it may seem that the above construction satisfies the unique third message property if the witness is unique. A closer inspection, however, reveals a subtle problem when we use the above WI in our template for SF-ZK. The issue is that a cheating prover can simply guess in advance, e.g., the first index (among all the parallel repetitions) where the challenge bit is 1. In that repetition, he can choose to garble a trivial circuit that outputs 1 on every input. Clearly, in this case, there are exponentially many accepting third messages. As such, the adversarial prover can violate the CUT property with non-negligible probability.

Towards addressing the above problem, our first observation is that the above protocol can be transformed into one that satisfies the unique third message property *at the cost of losing the delayed input property*. The transformation is simple: for every repetition, the prover pre-commits to both of its possible third messages (one for every challenge bit) in the first round. Now, in the last round, it simply decommits to the appropriate response. Clearly, this protocol satisfies the unique third message property but is no longer delayed input since the prover must know the statement and the witness in order to compute the first message. The latter means that we can not directly use it in our template for SF-ZK.

Nevertheless, as we now describe, the above observation can be used to construct a delayed-input WI with the required property. Our main observation is as follows: the aforementioned attack required the prover to deviate from the honest strategy, namely, sending a garbling of a circuit different from the NP verification circuit (i.e., the circuit which outputs 1 on every input). If we could ensure that the prover garbled the "correct" circuit, then the protocol would indeed satisfy the aforementioned uniqueness property.

Towards this end, we modify the protocol template and now require the prover to additionally prove via a separate three-round proof system that it computed the garbling in the first round message of delayed-input WI "honestly". Crucially, *a non-delayed-input proof with unique third message suffices for this task* since the statement and the witness is known in advance. The first and second messages of this proof are fixed in the pre-processing (in a manner as discussed before in the template); the prover only sends the third message of the proof in the execution phase. The uniqueness of this message ensures that it cannot be used for subliminal communication. More importantly, the soundness of this proof ensures that the prover's first message in the delayed-input WI is well-formed, and therefore, the last message is unique.[8]

Challenges in ZK. The above idea resolves the main challenge in achieving CUT property, but creates a new challenge in achieving the ZK property. Specifically, the main issue is that in order to perform simulation, it seems that we

[8] We remark that our actual protocol slightly differs from the above description in that instead of using delayed-input WI, we introduce and use the notion of (computationally) unique non-interactive WI with honest prover pre-processing. This approach yields a more simplified construction. In this Section, however, we ignore this distinction.

need the non-delayed-input proof to itself be a (steganography-free) ZK proof. However, this is very close to the goal we started with in the first place.

To resolve this seeming circularity, we observe that the non-delayed-input proof does *not* always need to be simulated. In particular, this proof would only need to be simulated when we invoke the WI property of the delayed-input WI inside the hybrids for proving the ZK property of our main SF-ZK construction. Therefore, we do not need this proof to satisfy the standard notion of ZK with polynomial-time simulation, and instead, it suffices to use ZK with *super-polynomial-time* simulation. Indeed, the super-polynomial-time simulator would only be invoked in the "intermediate" hybrids, but not the final one; therefore, the running time of our final simulator for SF-ZK is *unaffected*. Fortunately, the three-round proof system we described earlier indeed satisfies the super-polynomial-time simulation property.

Observer Soundness. While the above solution template resolves the main challenges in achieving ZK and CUT properties, it does not achieve observer soundness property of SF-ZK. Indeed, consider the scenario where the verifier is malicious during the pre-processing phase and uses some a priori fixed randomness (e.g., all 0's). Now, in the execution phase, a malicious prover can use the trapdoor witness (i.e., the witness of the second branch) in the WI proof in the last round.

To address this challenge, we observe that if the verifier is dishonest during pre-processing, then by our assumption that at least one of the parties be honest, we have that the prover must be honest during pre-processing. We use this observation to create an "asymmetry" between a malicious prover and the simulator. Specifically, we require the prover to commit to bit 0 in the pre-processing phase. We also modify the second branch of the WI in the execution phase. Specifically, the second branch will now additionally require the prover to prove that it committed to 1 in the pre-processing phase. Note that since the prover was honest in the pre-processing, it can never execute the second branch since it is always false. However, a simulator can choose to commit to 1 in the pre-processing phase and therefore still use the second branch of the WI.

Other Details. The above discussion is oversimplified and ignores several additional technical issues that we need to address to obtain a secure construction of SF-ZK. For example, we must deal with aborting verifiers who may choose to abort on one of the branches of WI with a high probability to skew the distribution of transcripts generated by the simulator. We also need to enable some mechanism for proving soundness as well as the CUT property via *extraction*, even when the verifier's randomness is fixed during the pre-processing. We resolve these issues by using techniques from [25], and by relying on complexity leveraging in some of our proofs. We refer the reader to the technical Sections for more details.

Multi-Execution SF-ZK. The pre-processing phase of the above construction is *non-reusable*, i.e., it can only be used for a single execution phase. We now describe a strategy to *refresh* the pre-processing phase. Our starting idea is sim-

ple: During the i-th execution phase, the prover and the verifier simply generate new pre-processing messages using pre-committed randomness and give a new SF-ZK proof to establish that the new message was computed honestly. Note, however, that in regular ZK proofs, the size of the prover's message grows with the size of the relation circuit. This means that the size of the i-th pre-processing messages must be larger than the size of the $(i+1)$-th pre-processing messages, at least by a multiplicative overhead of the security parameter. This means that this approach becomes infeasible after a constant number of refreshes.

A plausible approach to allow unlimited refreshing is to use an SF-ZK where the communication complexity does not grow with the size of the relation circuit. Four round ZK arguments (without SF property) that satisfy such a succinctness property are known for all of NP based on collision-resistant hash functions [32]. Unfortunately, it is not clear how to use such argument systems in our setting: first, we need the argument system to be *delayed-input*, namely, where the first message of the prover is independent of the statement. Further, it is unclear how to force uniqueness of last prover message while only relying on *non-interactive* pre-processing.

We instead use a different solution based on (leveled) fully-homomorphic encryption. The main idea is that instead of having the prover perform an "expensive" computation and prove its validity to the verifier, we instead require both the prover and the verifier to perform the expensive computation "locally" on their own. Since the computation involves the private state of the prover, we use FHE to send it to the verifier, who can use the homomorphism property to perform the computation. Now, the prover only needs to prove a simple statement that the resulting encryption (after homomorphic evaluation) decrypts to the "correct" value. The size of this statement (and the corresponding relation circuit) is fixed, and does not cause a blowup as before. Also observe that the maximum size of the circuit to be computed homomorphically is a priori fixed, therefore leveled FHE suffices. We note that this idea has been previously used (see, e.g., [30]) to construct "short" non-interactive zero-knowledge proofs.

1.4 Related Work

Preventing steganographic communication has been the subject of a large body of literature addressing the problem in variety models. We provide a short summary of other directions that address the challenge of protecting cryptosystems against different forms of subversion in below (also refer the reader to [38] for an excellent comprehensive survey).

Collusion-Free Protocols. Our work is closely related to prior work on collusion-free protocols [34] (see also [35]). Roughly speaking, a collusion-free multiparty protocol prevents a group of adversarial parties from colluding with each other to gain an unfair advantage over honest participants, e.g., in a game of poker. As Lepinski *et al.* explain in their work, a key challenge in designing such protocols is preventing steganographic communication between the adversarial parties. They use *physical assumptions*, namely, simultaneous exchange of

sealed envelopes, and an *interactive* pre-processing model to construct collusion-free protocols. While their overall goal is very similar to ours, we note that their constructions require strong physical assumptions (e.g., sealed envelopes) to ensure verifiable determinism.

We further note that our notion of steganography-free ZK is similar in spirit to the notion of "unique ZK" [35], which is used by [34] in their constructions. In particular, unique ZK requires a one-to-one mapping between a proof transcript and the witness used to compute the transcript, which is similar to the CUT property of steganography-free ZK. However, while our notion of steganography-free ZK is a *strengthening* of zero-knowledge, the notion of unique ZK is not. Unique ZK requires a common reference string as well as a pre-processing step where the prover must necessarily be honest. This means that if the prover was dishonest from the beginning, the soundness no longer holds (even if the verifier continues to be honest from the beginning). Unique ZK also does not require the observer soundness property, which makes it harder to use in our applications.

Preventing Steganography via Sanitization. Multiple lines of works have used the approach of using "sanitization" to prevent steganographic communication. The work of Alwen *et al.* [1] considered a mediator model for collusion-free protocols to avoid the use of pre-processing and physical channels. This active mediator has the ability to modify the messages of the protocol participants. This approach is similar in spirit to prior work on subliminal-free ZK and divertible ZK protocols [18, 39, 11, 13, 9, 12] who also use an active "warden" to modify the messages of the prover and the verifier. More recently, Mironov and Stephens-Davidowitz [38] (see also [20]) initiated the study of "reverse firewalls" to prevent steganographic communication in general two-party communication. Roughly speaking, a reverse firewall for a party P is an external entity that sits between P and the outside world and whose scope is to sanitize P's incoming and outgoing messages in the face of subversion of their computer. Later, there has been more efforts on secure computation protocols in this model [15, 23, 16].

Comparison to our model. In the sanitization-based model there is an entity (namely, the reverse firewall) that sits on the network of each participant and has the ability to re-randomize the messages sent by the parties. We note that all of these works differ fundamentally from ours in that they rely on an *active* mediator (or warden, or reverse firewall) who can sanitize the messages of the parties, whereas we consider the classical steganographic communication setting, where there is a *passive* observer who can look at the messages of the parties (but not modify them). This allows one to *detect* steganography by just looking at the communication transcript.

Kleptography and Algorithmic-Substitution Attacks. A sequence of works starting from [44, 45], and more recently followed by a series of papers [8, 5, 7, 41, 42, 2], consider the problem of designing cryptographic primitives which retain meaningful security even against adversaries who can tamper with the implementation of the cryptographic algorithm. In particular, these works consider "functionality-preserving" tampering where the adversary does not break the functionality of the cryptographic algorithm to avoid detection. However,

this still leaves open the possibility of the tampered implementation leaking any secret information used by the cryptographic algorithm (e.g., a secret-key for encryption, or a signing key for signature schemes) to the adversary by misusing the randomness. For this reason, these works either avoid the use of randomness altogether (whenever possible), or rely on external sanitizers (such as random oracles) or consider split-state tampering.

There has been another the line of work for protection mechanisms by Dodis *et al.* [19] that studies backdoored pseudorandom generators (BPRGs). In their setting, public parameters are secretly generated together with secret backdoors by a subversive that allows to bypass security, while for any adversary that does not know the backdoor it remains secure.[9] They showed that BPRGs can be immunized by applying a non-trivial function (e.g., a PRF or a seeded extractor) to the outputs of a possibly backdoored pseudorandom generator.

Comparison to our model. Our setting (involving ZK proofs and multi-party computation) necessarily relies on the use of randomness. As such, the solutions we achieve in our model restrict the use of randomness to the pre-processing step, without relying on external sanitizers, or other such means.

Trusted Initialization Phase. Assuming the trust initialization phase setting, Fischlin and Mazaheri [22] proposed an alternative defense mechanism, so-called self-guarding that contrary to the aforementioned approaches that rely on external sanitizers, does not depend on external parties. The security definitions in this model rely on the assumption of having a "secure initialization phase". This assumption makes our problem substantially easier: The NIZK by Sahai and Waters [43] has a deterministic prover and it trivially yields a construction of steganography-free ZK in the common reference string (CRS) model.

Comparison to our model. Self-guarding requires one to rely on a trusted initialization phase where the cryptosystem is unsubverted. In our model, each party runs a local pre-processing, and security is guaranteed if *either of the parties* is honest during the pre-processing phase.

2 Preliminaries

We denote by $n \in \mathbb{N}$ the security parameter that is implicitly given as input to all algorithms in unary representation 1^n. We denote by $\{0,1\}^\ell$ the set of all bit-strings of length ℓ. For a finite set S, we denote the action of sampling x uniformly at random from S by $x \leftarrow\!\!\$\, S$, and we denote the cardinality of S by $|S|$. An algorithm is efficient or PPT if it runs in time polynomial in the security parameter. If $\mathcal{A}$ is randomized then by $y := \mathcal{A}(x; r)$ we denote that $\mathcal{A}$ is run on input x and with random coins r and produces output y. If no randomness is specified, then it is assumed that $\mathcal{A}$ is run with freshly sampled uniform random coins, and we write this as $y \leftarrow \mathcal{A}(x)$. A function $\mathsf{negl}(n)$ is negligible if for all

[9] Parameter subversion has been considered for several primitives, including pseudorandom generators [19, 17], non-interactive zero knowledge [4], and public-key encryption [2].

positive polynomial $\mathsf{poly}(n)$, there exists an $N \in \mathbb{N}$, such that for all $n > N$, $\mathsf{negl}(n) \leq 1/\mathsf{poly}(n)$.

We recall the the notions of projective garbling schemes [6], homomorphic encryption [24], zero-knowledge arguments with super-polynomial simulation (SPS-ZK) [40], and non-Interactive witness indistinguishable arguments with honest pre-processing (HPP-NIWI) [10] in the full version of this paper.

3 Defining Steganography-Freeness

In this section, we introduce the definitions of steganography-free zero-knowledge interactive arguments and steganography-free multi-party computation. Steganography-freeness is generally impossible for regular protocols because without being constrained, a malicious party could always try to correlate its randomness with the secrets it wishes to subliminally communicate. We prevent such attacks by utilizing a non-interactive pre-processing phase. Specifically, we consider protocols that proceed in two phases: A non-interactive pre-processing phase, and an interactive execution phase. As we will see below, our definitions guarantee that no steganographic communication can be performed in the execution phase, once the pre-processing was completed.

We begin by defining steganography-freeness for generic interactive protocols (with pre-processing), which closely matches the intuition behind this notion. Roughly speaking, our notion steganography-free says that no machines can communicate through a protocol execution without being detected. This is captured as a game between a sender and a receiver, where the sender is given a random bit b and interacts with the receiver. In order to win the game the receiver must output b, without raising the suspicion of an external observer. The formal definition is given in the following.

Definition 1 (Steganography-Freeness). *A protocol $\Pi = (\mathsf{S}^1, \mathsf{R}^1, \mathsf{S}^2, \mathsf{R}^2)$ is steganography-free relative to a PPT observer Θ if for all admissible pairs $(\tilde{\mathsf{S}}^1, \tilde{\mathsf{R}}^1)$, and for all PPT algorithms $(\mathsf{S}^*, \mathsf{R}^*)$ it holds that*

$$\Pr\left[\begin{matrix}(s_1, p_1) \leftarrow \tilde{\mathsf{S}}^1(1^\lambda), (s_2, p_2) \leftarrow \tilde{\mathsf{R}}^1(1^\lambda), \\ b \leftarrow\$ \{0,1\}, \mathcal{T} := \langle \mathsf{S}^*(s_1, p_2, b), \mathsf{R}^*(s_2, p_1)\rangle\end{matrix} : \begin{matrix}\Theta(p_1, p_2, \mathcal{T}) = 1 \wedge \\ \mathsf{R}^*(s_2, p_1, \mathcal{T}) = b\end{matrix}\right] \leq \frac{1}{2} + \mathsf{negl}(n)$$

where $(\mathsf{S}^*, \mathsf{R}^*)$ are the (possibly) corrupted versions of $(\mathsf{S}^2, \mathsf{R}^2)$. Both parties ($\mathsf{S}^1$ and R^1) individually compute pre-processing information comprising of a public output and a secret state in the pre-processing stage. In the execution phase, both parties (S^2 and R^2) receive as input their respective secret states as well as the other party's public output from the pre-processing phase.

Note that the definition is *relative* to some observer Θ. Generally, any protocol is steganography-free relative to *some* observer, e.g., the trivial Θ that does not accept *any* transcript. However, this is of course not a useful property. The challenge, therefore, is to achieve steganography-freeness relative to a meaningful observer that accepts honest communication.

It is also important to observe that the definition is conditioned on some admissibility criterion on the behavior of the players in the pre-processing. In this work we are interested in what we call a *partial-honest* pre-processing, i.e., a pair $(\tilde{\mathsf{S}}^1, \tilde{\mathsf{R}}^1)$ is considered admissible if both algorithms are PPT and at least one of them is honest. Note that for this case we consider *rushing* adversaries that sample their pre-processing after the honest one is fixed. We mention that the definition can be extended to capture a bounded amount of covert communication by sampling multiple bits.

3.1 Steganography-Free Zero-Knowledge

Towards defining steganography-free zero-knowledge, we extend the standard definitions in a natural way to accommodate an input-independent pre-processing phase. In the pre-processing stage, both parties (P^1 and V^1) individually compute pre-processing information comprising of a public output and a secret state. In the execution phase, both parties (P^2 and V^2) receive as input their respective secret states as well as the other party's public output from the pre-processing phase, together with the statement x. The prover additionally receives a witness w. At the end of this phase, the honest verifier outputs either 0 or 1. In addition to the standard properties for a zero-knowledge protocol, a steganography-free zero-knowledge protocol must additionally satisfy the following new properties:

1. *Observer Completeness:* There exists an efficient algorithm Θ, that takes as input the protocol transcript and accepts if both parties are honest.
2. *Observer Soundness:* The (possibly colluding) prover and verifier cannot convince the observer to accept a transcript for any $x \notin \mathcal{L}$, as long as either the prover or the verifier executes the pre-processing phase honestly.
3. *Computationally Unique Transcripts:* Given a language with unique witnesses, no two independent coalitions of prover and verifier can produce two different transcripts that are both accepted by the observer. This is again conditioned on the fact that at least one of the parties was honest during the pre-processing.

This set of properties will guarantee that the protocol execution cannot be used as a covert channel. Later we will show that these conditions are indeed sufficient to achieve steganography-freeness. The formal definition is given in the following.

Definition 2 (Steganography-Free Zero-Knowledge Arguments). *Let $\mathcal{L}$ be a language in NP with corresponding relation $\mathcal{R}$. A steganography-free interactive argument system $\Pi = (\mathsf{P}, \mathsf{V})$ for language $\mathcal{L}$ in the non-interactive pre-processing model with observer Θ must satisfy the following properties:*

Completeness. For all $(x, w) \in \mathcal{R}$ it holds that

$$\Pr\left[\begin{matrix}(s_1, p_1) \leftarrow \mathsf{P}^1(1^n), \\ (s_2, p_2) \leftarrow \mathsf{V}^1(1^n)\end{matrix} : 1 \leftarrow \left\langle \mathsf{P}^2(x, w, s_1, p_2), \mathsf{V}^2(x, s_2, p_1) \right\rangle\right] \geq 1 - \mathsf{negl}(n).$$

Computational Non-Adaptive Soundness. For all $x \notin \mathcal{L}$ and all malicious PPT *provers* P^* *it holds that*

$$\Pr\left[\begin{matrix}(s_1,p_1)\leftarrow \mathsf{P}^*(x),\\(s_2,p_2)\leftarrow \mathsf{V}^1(1^n)\end{matrix} : 1 \leftarrow \left\langle \mathsf{P}^*(x,s_1,p_2), \mathsf{V}^2(x,s_2,p_1)\right\rangle\right] \leq \mathsf{negl}(n).$$

Computational Soundness. For all malicious PPT *provers* P^* *it holds that*

$$\Pr\left[\begin{matrix}(s_1,p_1)\leftarrow \mathsf{P}^*(1^n),\\(s_2,p_2)\leftarrow \mathsf{V}^1(1^n), : 1 \leftarrow \left\langle \mathsf{P}^*(x,s_1,p_2), \mathsf{V}^2(x,s_2,p_1)\right\rangle \wedge x \notin \mathcal{L}\\x\leftarrow \mathsf{P}^*(s_1,p_2)\end{matrix}\right] \leq \mathsf{negl}(n).$$

Here, we use the terms computational soundness and adaptive computational soundness interchangeably.

Zero-Knowledge. For all malicious PPT *verifiers* V^* *there exists an expected polynomial time simulator* Sim*, such that for all* PPT *distinguishers* $\mathcal{D}$*, it holds that for all tuples* $(x,w) \in \mathcal{R}$

$$\left|\begin{matrix}\Pr\left[\begin{matrix}(s_1,p_1)\leftarrow \mathsf{P}^1(1^n),\\(s_2,p_2)\leftarrow \mathsf{V}^*(x)\end{matrix} : \mathcal{D}(\left\langle \mathsf{P}^2(x,w,s_1,p_2), \mathsf{V}^*(x,s_2,p_1)\right\rangle) = 1\right]\\ -\Pr\left[\begin{matrix}(s_1,p_1)\leftarrow \mathsf{Sim}(1^n),\\(s_2,p_2)\leftarrow \mathsf{V}^*(x)\end{matrix} : \mathcal{D}(\langle \mathsf{Sim}(x,s_1,p_2), \mathsf{V}^*(x,s_2,p_1)\rangle) = 1\right]\end{matrix}\right| \leq \mathsf{negl}(n).$$

Observer Completeness. For all $(x,w) \in \mathcal{R}$ *it holds that*

$$\Pr\left[\begin{matrix}(s_1,p_1)\leftarrow \mathsf{P}^1(1^n), (s_2,p_2)\leftarrow \mathsf{V}^1(1^n),\\\mathcal{T} := \left\langle \mathsf{P}^2(s_1,p_2,x,w), \mathsf{V}^2(s_2,p_1,x)\right\rangle\end{matrix} : \Theta(p_1,p_2,\mathcal{T},x) = 1\right] \geq 1 - \mathsf{negl}(n).$$

Non-Adaptive Observer Soundness. For all $x \notin \mathcal{L}$*, for all admissible pairs* $(\tilde{\mathsf{P}}^1, \tilde{\mathsf{V}}^1)$*, for all* PPT *algorithms* P^* *and* V^* *it holds that*

$$\Pr\left[\begin{matrix}(s_1,p_1)\leftarrow \tilde{\mathsf{P}}^1(x), (s_2,p_2)\leftarrow \tilde{\mathsf{V}}^1(x),\\\mathcal{T} := \langle \mathsf{P}^*(s_1,p_2,x), \mathsf{V}^*(s_2,p_1,x)\rangle\end{matrix} : \Theta(p_1,p_2,\mathcal{T},x) = 1\right] \leq \mathsf{negl}(n)$$

Observer Soundness. For all admissible pairs $(\tilde{\mathsf{P}}^1, \tilde{\mathsf{V}}^1)$*, for all* PPT *algorithms* P^* *and* V^* *it holds that*

$$\Pr\left[\begin{matrix}(s_1,p_1)\leftarrow \tilde{\mathsf{P}}^1(1^n), (s_2,p_2)\leftarrow \tilde{\mathsf{V}}^1(1^n),\\x\leftarrow \mathsf{P}^*(s_1,p_2); \mathcal{T} := \langle \mathsf{P}^*(s_1,p_2,x), \mathsf{V}^*(s_2,p_1,x)\rangle\end{matrix} : \begin{matrix}\Theta(p_1,p_2,\mathcal{T},x) = 1\\ \wedge \;\; x \notin \mathcal{L}\end{matrix}\right] \leq \mathsf{negl}(n)$$

where a pair $(\tilde{\mathsf{P}}^1, \tilde{\mathsf{V}}^1)$ *is considered admissible if both algorithms are* PPT *and it holds that* $\tilde{\mathsf{P}}^1(w,x) = \mathsf{P}^1(1^n)$ *or* $\tilde{\mathsf{V}}^1(x) = \mathsf{V}^1(1^n)$*. Notice that, we use the terms observer soundness and adaptive observer soundness interchangeably.*

Computationally Unique Transcripts. For all $x \in \mathcal{L}$ *such that there exists a unique* w *such that* $\mathcal{R}(x,w) = 1$*, for all admissible pairs* $(\tilde{\mathsf{P}}^1, \tilde{\mathsf{V}}^1)$*, for all* PPT *algorithms* $(\mathsf{P}^*, \mathsf{V}^*, \hat{\mathsf{P}}^*, \hat{\mathsf{V}}^*)$ *it holds that*

$$\Pr\left[\begin{matrix}(s_1,p_1)\leftarrow \tilde{\mathsf{P}}^1(w,x), (s_2,p_2)\leftarrow \tilde{\mathsf{V}}^1(x),\\\mathcal{T}_1 := \langle \mathsf{P}^*(s_1,p_2,x,w), \mathsf{V}^*(s_2,p_1,x)\rangle,\\\mathcal{T}_2 := \left\langle \hat{\mathsf{P}}^*(s_1,p_2,x,w), \hat{\mathsf{V}}^*(s_2,p_1,x)\right\rangle\end{matrix} : \begin{matrix}\Theta(p_1,p_2,\mathcal{T}_1,x) = 1 \wedge\\ \Theta(p_1,p_2,\mathcal{T}_2,x) = 1 \wedge\\ \mathcal{T}_1 \neq \mathcal{T}_2\end{matrix}\right] \leq \mathsf{negl}(n)$$

where a pair $(\tilde{\mathsf{P}}^1, \tilde{\mathsf{V}}^1)$ *is considered admissible if both algorithms are* PPT *and it holds that* $\tilde{\mathsf{P}}^1(w,x) = \mathsf{P}^1(1^n)$ *or* $\tilde{\mathsf{V}}^1(x) = \mathsf{V}^1(1^n)$.

Observe that, although the honest pre-processing algorithms do not require the statement or the witness as input, we still provide the (possibly) malicious machines with x (and w if the prover is malicious). This guarantees that the properties are preserved even if the algorithm has partial knowledge of the statement (and possibly the witness) ahead of time.

We further remark that our definition of computationally unique transcripts is going to be useful only for languages with unique witnesses, since the prover might be able to produce two accepting transcripts by simply executing the protocol with two different witnesses. While this suffices for our applications, the definition can be naturally extended to the k-witnesses case by requiring the coalitions to output $k+1$ distinct valid transcripts.

Steganography-Freeness. In the following, we argue that our conditions defined above suffice to show that the protocol satisfies steganography-freeness.

Theorem 1 (Steganography-Freeness). *Let* $\mathcal{L}$ *be a language with unique witnesses and let* (P, V) *be an observer sound zero-knowledge protocol for* $\mathcal{L}$ *with computationally unique transcripts. Then* (P, V) *is steganography-free relative to the observer with partially honest pre-processing.*

We defer the proof to the full version.

Multi-Execution SF-ZK. The above definition refers to *single-execution* SF-ZK where all of the properties are required to hold for a single execution phase, after the pre-processing is fixed. In the full version of the paper, we extend the notion of SF-ZK to the *multi-execution* setting.

4 A Steganography-Free ZK Protocol

Let $\tilde{\mathcal{L}}$ be any average-case hard language with unique witnesses and let $f : \{0,1\}^{n_{\mathsf{OWF}}} \to \{0,1\}^{m_{\mathsf{OWF}}}$ be a one-way function with an efficiently checkable range. Let $(\mathsf{WI\text{-}P}, \mathsf{WI\text{-}V})$ be an HPP-NIWI with unique proofs for the following language: $\mathcal{L}_{\mathsf{NIWI}} =$

$$\left\{ \begin{pmatrix} x, y, \tilde{w}, \\ c_0, \bar{c}, \tilde{c} \end{pmatrix} \;\middle|\; \begin{array}{l} \exists (w, s, \tilde{r}) : ((x,w) \in \mathcal{R} \wedge \mathsf{Com}(\tilde{r}; s) = \bar{c} \wedge \mathsf{Com}(0^n; \tilde{r}) = \tilde{c}) \\ \vee\ \exists (r, \tilde{r}) : (\mathsf{Com}(1; r) = c_0 \wedge \mathsf{Com}(\tilde{w}; \tilde{r}) = \tilde{c}) \\ \vee\ \exists (w, r, z) : ((x,w) \in \mathcal{R} \wedge \mathsf{Com}(1; r) = c_0 \wedge f(z) = y) \end{array} \right\}$$

where the first branch (1) is going to be used by the prover and the second branch (2) will allow one to simulate without knowing the witness. Interestingly the third branch (3) is used neither by the honest prover nor by the simulator, but it is only instrumental to prove the indistinguishability of the two. Finally, we let $(\mathsf{SPS\text{-}P}, \mathsf{SPS\text{-}V})$ be a three-round SPS-ZK argument system with unique last messages for the following language:

$$\mathcal{L}_{\mathsf{SPSZK}} = \{\tau \mid \exists u : \mathsf{WI\text{-}P}_1(u) = \tau\}.$$

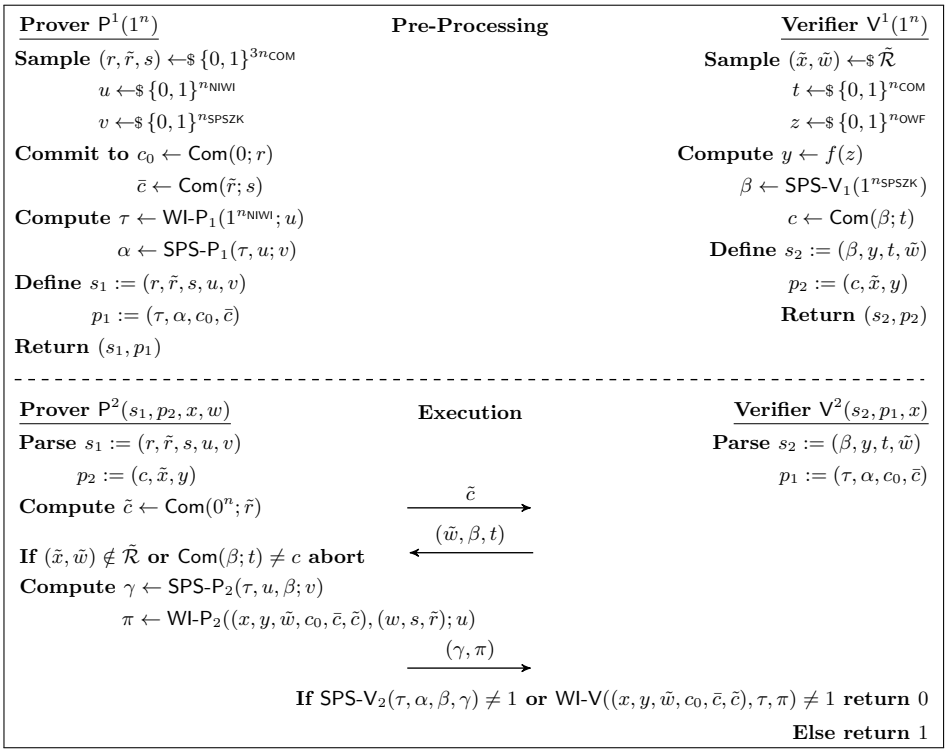

Fig. 1. Our SF-ZK protocol.

4.1 Our Protocol

Our protocol SF-ZK is formally described in Figure 1. We describe extensions to the multi-execution setting in the full version.

Pre-Processing. In the pre-processing phase, the honest prover computes a commitment to 0 and to some random coins $\tilde{r}$. The former guarantees that, if the prover's pre-processing is honest, then it is hard to cheat in the execution phase, whereas the latter fixes the random coins used later in the execution phase. The prover also initializes the pre-processing τ of an HPP-NIWI proof and computes the first message α of an SPS-ZK proof that asserts that τ is well-formed. The public output of the prover's pre-processing consists of the commitments together with the messages (τ, α). The secret state consists of the random coins used in the pre-processing.

On the other hand, the verifier samples a random image y from the domain of the one-way function f and computes a commitment c to a randomly sampled second message β of the SPS-ZK proof. Furthermore, it samples a random instance $\tilde{x}$ of an average-case hard language with unique witnesses. The public output of the verifier's pre-processing consists of $(c, \tilde{x}, y)$, and the secret state consists of the random coins used in the pre-processing.

Execution. The execution phase is started by the prover, who sends a commitment $\tilde{c}$ to 0^n, using the random coins $\tilde{r}$ fixed in the pre-processing. Then the verifier replies with the decommitment (β, t) to c and reveals the unique witness $\tilde{w}$. The prover checks that (β, t) is a valid decommitment for c and computes the last message γ of the SPS-ZK protocol that certifies that τ is well-formed. Finally, it computes the proof π using the first branch (1) thereby proving that $\tilde{c}$ was correctly formed using the random coins committed in the pre-processing and that x is indeed an accepting instance of $\mathcal{L}$. The verifier simply checks whether the transcript (α, β, γ) and the proof π verify correctly.

While $\tilde{c}$ might seem purposeless, it is going to be useful in the simulation: The simulator will spawn a lookahead thread to learn $\tilde{w}$, which will allow it to rewind the execution to compute $\tilde{c}$ as a commitment to $\tilde{w}$. This in turn allows it to compute the proof π using the second branch (2), which does not require knowledge of the witness for x. This is however not a feasible strategy for any malicious prover (which cannot rewind the execution of the protocol), since it requires to know $\tilde{w}$ ahead of time.

4.2 Analysis

Parameters. Let n be the security parameter of our scheme, we consider the following parameters that are (implicitly) given as input to each algorithm of our building blocks:

- n_{SPSZK} : The security parameter for the SPS-ZK argument $(\mathsf{SPS\text{-}P}, \mathsf{SPS\text{-}V})$.
- n_{NIWI} : The security parameter for the non-interactive witness indistinguishable argument $(\mathsf{WI\text{-}P}, \mathsf{WI\text{-}V})$.
- n_{COM} : The security parameter for the perfectly binding commitment scheme Com with unique openings.
- n_{L} : The security parameter for the average-case hard language with unique witnesses $\tilde{\mathcal{L}}$.
- n_{OWF} : The security parameter for the one-way function f.

We require that the parameters satisfy the following relation

$$2^{n_{\mathsf{SPSZK}}} \ll 2^{n_{\mathsf{OWF}}} \ll 2^{n_{\mathsf{COM}}} \ll 2^{n_{\mathsf{NIWI}}} = 2^{n_{\mathsf{L}}},$$

where $a \ll b$ means that for all polynomial functions $a \cdot \mathsf{poly}(n) < b$. In particular we require the SPS-ZK argument to be sound against an adversary that runs in time $\mathsf{poly}(n_{\mathsf{SPSZK}})$ and to be simulatable in time $O(2^{n_{\mathsf{SPSZK}}})$. By setting the security parameter of the underlying perfectly binding commitment scheme to be also n_{SPSZK}, then one can find the committed message in time $O(2^{n_{\mathsf{SPSZK}}})$ by exhaustive search.[10] We require the one-way function to be hard to invert in time $O(2^{n_{\mathsf{SPSZK}}})$ but easy to invert in time $O(2^{n_{\mathsf{OWF}}})$, similarly the commitment scheme is hiding against $O(2^{n_{\mathsf{OWF}}})$ bounded machines but extractable in time $O(2^{n_{\mathsf{COM}}})$. Finally, the HPP-NIWI and the average-case hard language shall be hard even for adversaries running in time $O(2^{n_{\mathsf{COM}}}) \gg O(2^{n_{\mathsf{OWF}}}) \gg O(2^{n_{\mathsf{SPSZK}}})$.

[10] This instantiation of the perfectly binding commitment scheme used inside the SPS-ZK protocol is different from the perfectly binding commitment scheme Com used in our protocol. In particular, we use different security levels for these schemes.

Security proof. In the following, we state our main theorems:

Theorem 2 (Soundness). *If* $(\mathsf{WI\text{-}P}, \mathsf{WI\text{-}V})$ *is an HPP-NIWIs with unique proofs,* $\tilde{\mathcal{L}}$ *is an average-case hard language with unique witnesses,* $(\mathsf{SPS\text{-}P}, \mathsf{SPS\text{-}V})$ *is an SPS-ZK argument, and the commitment scheme* Com *is perfectly binding, then the argument system SF-ZK in Figure 1 is computationally sound.*

Proof. The proof consists of two steps. In the first step, we prove that it in present of non-adaptive (selective) security notation in a way that the adversary is not allow to adaptively choose the statement. In the second step, we invoke complexity leveraging to lift the reduction to the adaptive settings.

Non-adaptive soundness. Assume that there exists an $x^* \notin \mathcal{L}$ and a malicious PPT prover P^* such that the verifier on input x^* and interaction with P^* will accept with probability ϵ. Let $x_{\mathsf{NIWI}} := (x, y, \tilde{w}, c_0, \bar{c}, \tilde{c})$ We can split this probability into two parts: Either P^* cheats in such away that $x_{\mathsf{NIWI}} \notin \mathcal{L}_{\mathsf{NIWI}}$ (in which case we will be able to use the soundness of the HPP-NIWI to show that P^* would not be successful) or P^* cheats in such away that $x_{\mathsf{NIWI}} \in \mathcal{L}_{\mathsf{NIWI}}$. In this case, we show that this event can only occur with negligible probability due to the average-case hardness of $\tilde{\mathcal{L}}$. Let cheat be the event that a malicious prover causes the honest verifier to accept x^*.

$$
\begin{aligned}
\epsilon &= \Pr[\mathsf{cheat}] \\
&= \underbrace{\Pr[\mathsf{cheat} | x_{\mathsf{NIWI}} \notin \mathcal{L}_{\mathsf{NIWI}}] \cdot \Pr[x_{\mathsf{NIWI}} \notin \mathcal{L}_{\mathsf{NIWI}}]}_{\epsilon'} \\
&+ \underbrace{\Pr[\mathsf{cheat} | x_{\mathsf{NIWI}} \in \mathcal{L}_{\mathsf{NIWI}}] \cdot \Pr[x_{\mathsf{NIWI}} \in \mathcal{L}_{\mathsf{NIWI}}]}_{\epsilon''}
\end{aligned}
$$

Bounding ϵ'. We will first bound ϵ' using the soundness of the HPP-NIWI and the super-polynomial extractability of the SPS-ZK. Assume towards contradiction, that $\epsilon' \geq 1/\mathsf{poly}(n)$. We then construct a malicious $\mathsf{WI\text{-}P}^*$ as follows: $\mathsf{WI\text{-}P}^*$ engages with P^* in a protocol execution where it impersonates the verifier and computes all of the messages honestly. Let (α, β, γ) be the variables determined by the transcript of the execution. Then $\mathsf{WI\text{-}P}^*$ checks that $\mathsf{SPS\text{-}V}(\tau, \alpha, \beta, \gamma) = 1$ and extracts the witness u from (α, β, γ) in time $O(2^{n_{\mathsf{SPSZK}}})$ (recall the choice of parameters from Section 4.2) if this is the case. If the extraction fails or the transcript does not verify, then $\mathsf{WI\text{-}P}^*$ aborts. Finally, $\mathsf{WI\text{-}P}^*$ outputs $(x_{\mathsf{NIWI}}, \tau, \pi, u)$.

It is easy to see that $\mathsf{WI\text{-}P}^*$ perfectly simulates the verifier's preprocessing as well as the execution phase for P^*. $\mathsf{WI\text{-}P}^*$ successfully cheats, if (τ, π) verifies, extraction is successful, and $x_{\mathsf{NIWI}} \notin \mathcal{L}_{\mathsf{NIWI}}$.

Note that $1 \leftarrow \langle \mathsf{P}^*(x^*, s_1, p_2), \mathsf{V}^1(x^*, s_2, p_1) \rangle$ implies that both (α, β, γ) as well as (τ, π) verify correctly. Assume for the moment that the extraction from

(α, β, γ) is successful with probability $1 - \mathsf{negl}(n)$. Then it holds that

$$\begin{aligned}
&\Pr\left[(x_{\mathsf{NIWI}}, \tau, \pi, u) \leftarrow \mathsf{WI\text{-}P}^*(1^n) : \begin{matrix} x_{\mathsf{NIWI}} \notin \mathcal{L}_{\mathsf{NIWI}} \wedge \mathsf{WI\text{-}P}_1(u) = \tau \\ \wedge\ \mathsf{WI\text{-}V}(x_{\mathsf{NIWI}}, \tau, \pi) = 1 \end{matrix}\right] \\
\geq &\Pr[\mathsf{cheat} | x_{\mathsf{NIWI}} \notin \mathcal{L}_{\mathsf{NIWI}}] \cdot \Pr[x_{\mathsf{NIWI}} \notin \mathcal{L}_{\mathsf{NIWI}}] \cdot (1 - \mathsf{negl}(n)) \\
= &\epsilon' - \mathsf{negl}(n) = 1/\mathsf{poly}(n) - \mathsf{negl}(n).
\end{aligned}$$

Since $\mathsf{WI\text{-}P}^*$ runs in time $O(2^{n_{\mathsf{SPSZK}}}) + \mathsf{poly}(n)$ this would contradict the soundness of the HPP-NIWI. What is left to be shown is that the probability that the extraction from (α, β, γ) is not successful is bounded by a negligible function. If this was not the case, then α and the randomness used to compute it would uniquely determine β (recall the properties of SPS-ZK from Section 2). Therefore we could find the randomness in time $O(2^{n_{\mathsf{SPSZK}}}) + \mathsf{poly}(n)$ and use it together with α, to break the hiding property of $c = \mathsf{Com}(\beta)$. It follows that the extraction must succeed with all but negligible probability. We can conclude that $\epsilon' \leq \mathsf{negl}(n)$.

Bounding ϵ''. Assume towards contradiction that $\epsilon'' \geq 1/\mathsf{poly}(n)$. Since $x^* \notin \mathcal{L}$, the definition of $\mathcal{L}_{\mathsf{NIWI}}$ implies that for an $x_{\mathsf{NIWI}} \in \mathcal{L}_{\mathsf{NIWI}}$ there exists an $(r, \tilde{r})$ such that $\mathsf{Com}(1; r) = c_0$ and $\mathsf{Com}(\tilde{w}; \tilde{r}) = \tilde{c}$. However, we can show that this would allow us to decide $\tilde{\mathcal{L}}$ in the average case as follows.

Given a random instance $\tilde{x}$, compute a verifier preprocessing honestly using $\tilde{x}$ as the random instance of the average-case hard language. The prover P^* returns its pre-processing and the commitment $\tilde{c}$. Then extract the content of $\tilde{c}$ in time $O(2^{n_{\mathsf{COM}}})$. If it contains a valid witness for $\tilde{x}$ return 1, else return a random bit. Note that if $\tilde{x} \notin \tilde{\mathcal{L}}$ then $\tilde{w}$ does not exist and therefore the algorithm described above will always output a random bit. On the other hand, if $\tilde{x} \in \tilde{\mathcal{L}}$ then we can lower bound the probability of the algorithm outputting 1 by $1/2 + \epsilon'' = 1/2 + 1/\mathsf{poly}(n)$. Since the described algorithm runs in time $O(2^{n_{\mathsf{COM}}}) + \mathsf{poly}(n)$ this clearly contradicts the average case hardness of $\tilde{\mathcal{L}}$ as specified in Section 4.2. We have thus established that $\epsilon = \epsilon' + \epsilon'' \leq \mathsf{negl}(n)$ and SF-ZK is therefore computationally sound.

From selective to adaptive . For the second step of the proof, we rely on complexity leveraging. Let l_x be the domain size of the statement $l_x = |x|$. Let $\mathcal{B}$ against the adaptive security. We set l_x to be

$$2^{l_x} \ll 2^{n_{\mathsf{SPSZK}}} \ll 2^{n_{\mathsf{OWF}}} \ll 2^{n_{\mathsf{COM}}} \ll 2^{n_{\mathsf{NIWI}}} = 2^{n_{\mathsf{L}}}.$$

We construct a reduction which behaves identically as the non-adaptive case, except that it guesses a statement x and aborts if $x \neq x^*$. The analysis is identical to what described above, except that the advantage drops by a factor at most $1/2^{l_x}$.

Theorem 3 (Observer Soundness). *If* $(\mathsf{WI\text{-}P}, \mathsf{WI\text{-}V})$ *is an HPP-NIWI with unique proofs,* $\tilde{\mathcal{L}}$ *is an average-case hard language with unique witnesses,* Com *is a perfectly binding commitment scheme with unique openings, and* $(\mathsf{SPS\text{-}P}, \mathsf{SPS\text{-}V})$ *is an SPS-ZK argument, then the argument system SF-ZK in Figure 1 is observer sound.*

$\Theta(p_1, p_2, \mathcal{T}, x)$

1 : **if** $\mathsf{Com}(\beta; t) \neq c$ **or** $(\tilde{x}, \tilde{w}) \notin \tilde{\mathcal{R}}$
2 : **return** 0
3 : **elseif** $\mathsf{SPS\text{-}V}(\tau, \alpha, \beta, \gamma) = 0$ **or** $\mathsf{WI\text{-}V}((x, y, \tilde{w}, c_0, \bar{c}, \tilde{c}), \tau, \pi) = 0$
4 : **return** 0
5 : **else return** 1

Fig. 2. The observer algorithm Θ

Proof. We describe the observer algorithm in Figure 2. Recall that the observer soundness definition considers two cases. In one case the prover acts honestly during the pre-processing phase ($\tilde{\mathsf{P}} = \mathsf{P}^1$), in the other case the verifier does ($\tilde{\mathsf{V}} = \mathsf{V}^1$). We analyze the two cases separately.

Honest P^1. Assume towards contradiction, that there exists an $x^* \notin \mathcal{L}$, a malicious prover P^*, and a malicious verifier V^* such that

$$\frac{1}{\mathsf{poly}(n)} \leq \Pr\left[\begin{matrix}(s_1, p_1) \leftarrow \mathsf{P}^1(1^n), (s_2, p_2) \leftarrow \mathsf{V}^*(x^*), \\ \mathcal{T} := \langle \mathsf{P}^*(x^*, s_1, p_2), \mathsf{V}^*(x^*, s_2, p_1)\rangle\end{matrix} : \Theta(p_1, p_2, \mathcal{T}, x) = 1\right].$$

From this it follows that

$$\frac{1}{\mathsf{poly}(n)} \leq \Pr\left[\begin{matrix}(r, \tau) \leftarrow \mathsf{P}^1(1^n), \\ \pi \in \mathcal{T}\end{matrix} : \mathsf{WI\text{-}V}((x, y, \tilde{w}, c_0, \bar{c}, \tilde{c}), \tau, \pi) = 1\right]. \tag{1}$$

Where Equation 1 stems from the fact that the prover's pre-processing is honest and the observer always verifies the proof π. Recall that the statement $(x, y, \tilde{w}, c_0, \bar{c}, \tilde{c}) \in \mathcal{L}_{\mathsf{NIWI}}$ if and only if

$$\begin{aligned} &\exists(s, \tilde{r}) : (\underline{x^* \in \mathcal{L}} \wedge \mathsf{Com}(\tilde{r}; s) = \bar{c} \wedge \mathsf{Com}(0^n, \tilde{r}) = \tilde{c}) \\ \vee\ &\exists(r, \tilde{r}) : (\underline{\mathsf{Com}(1; r) = c_0} \wedge \mathsf{Com}(\tilde{w}, \tilde{r}) = \tilde{c}) \\ \vee\ &\exists(r, z) : (x^* \in \mathcal{L} \wedge \underline{\mathsf{Com}(1; r) = c_0} \wedge f(z) = y) \end{aligned} \tag{2}$$

By assumption, $x^* \notin \mathcal{L}$ and $\mathsf{Com}(0; r) = c_0$, since the prover's pre-processing is generated honestly and the commitment scheme is perfectly binding. Therefore each of the parts underlined in Equation 2 is false. By extensions, this makes the conjunction in each of the three branches false. It follows then that π is a proof for a false statement given an honestly generated τ, which contradicts the soundness of the HPP-NIWI.

Honest V^1. For this case we can bootstrap the verifier's honest preprocessing into a fully honest verifier execution and then simply reduce observer soundness to regular soundness.

Assume towards contradiction that there exists an $x^* \notin \mathcal{L}$, a malicious prover P^*, and a malicious verifier V^* such that

$$\frac{1}{\mathsf{poly}(n)} \leq \Pr\left[\begin{matrix}(s_1, p_1) \leftarrow \mathsf{P}^*(x^*), (s_2, p_2) \leftarrow \mathsf{V}^1(1^n), \\ \mathcal{T} := \langle \mathsf{P}^*(x^*, s_1, p_2), \mathsf{V}^*(x^*, s_2, p_1)\rangle\end{matrix} : \Theta(p_1, p_2, \mathcal{T}, x) = 1\right].$$

From this it follows that

$$\frac{1}{\mathsf{poly}(n)} \leq \Pr\left[\begin{array}{l}(s_1,p_1) \leftarrow \mathsf{P}^*(x^*),\\ (s_2,p_2) \leftarrow \mathsf{V}^1(1^n),\\ \mathcal{T} := \langle \mathsf{P}^*(x^*,s_1,p_2), \mathsf{V}^2(x^*,s_2,p_1)\rangle\end{array} : \Theta(p_1,p_2,\mathcal{T},x)=1\right] \quad (3)$$

$$= \Pr\left[\begin{array}{l}(s_1,p_1) \leftarrow \mathsf{P}^*(x^*),\\ (s_2,p_2) \leftarrow \mathsf{V}^1(1^n)\end{array} : 1 \leftarrow \langle \mathsf{P}^*(x^*,s_1,p_2), \mathsf{V}^2(x^*,s_2,p_1)\rangle\right] \quad (4)$$

To see why Equation 3 holds, first note that the commitment scheme is perfectly binding and the language $\tilde{\mathcal{L}}$ has unique witnesses. Since Θ verifies in line 1 that (β, t) is a valid decommitment of c and that $\tilde{w}$ is indeed a witness of $\tilde{x}$, it follows that given the verifier's honest pre-processing there exists only a unique verifier message that does *not* cause the observer to output 0. For every possible transcript of the interaction between P^* and V^* consider the following two possibilities. Either the message sent by V^* is exactly that unique message or it sends any other message. In the first case, the malicious verifier behaves identically to the honest verifier and replacing V^* by V^2 does not change the resulting transcript or the output of Θ at all. In the latter case, Θ already outputs 0 for this transcript anyway and the only change could be that Θ now outputs 1. Thus we can conclude that the probability of Θ outputting 1 can only increase. Thus Equation 3 must hold.

To see that Equation 4 must hold we simply need to consider the checks performed by Θ in line 3. It's easy to see that $\Theta(p_1,p_2,\mathcal{T},x) = 1$ implies that $\mathsf{SPS\text{-}V}(\tau,\alpha,\beta,\gamma) = 1$ and $\mathsf{WI\text{-}V}((x,y,\tilde{w},c_0,\bar{c},\tilde{c}),\tau,\pi) = 1$, since the protocols are public-coin (and therefore publicly verifiable). However, these coincide with all checks performed by the honest verifier. Therefore, in an execution between the malicious prover and the honest verifier, the honest verifier accepts if and only if the transcript is accepted by the observer. Equation 4 therefore holds. We've thus shown that

$$\frac{1}{\mathsf{poly}(n)} \leq \Pr\left[\begin{array}{l}(s_1,p_1) \leftarrow \mathsf{P}^*(x^*),\\ (s_2,p_2) \leftarrow \mathsf{V}^1(1^n)\end{array} : 1 \leftarrow \langle \mathsf{P}^*(x^*,s_1,p_2), \mathsf{V}^2(x^*,s_2,p_1)\rangle\right]$$

which would contradict the soundness of SF-ZK. Therefore, an x^* and P^* as assumed above cannot exist and SF-ZK must also be (selective) observer sound. The proof for the adaptive observer sound is the same as above.

Theorem 4 (Zero Knowledge). *If* $(\mathsf{WI\text{-}P}, \mathsf{WI\text{-}V})$ *is an HPP-NIWIs with unique proofs,* $(\mathsf{SPS\text{-}P}, \mathsf{SPS\text{-}V})$ *is an SPS-ZK argument with unique last message,* f *is a one-way function with efficiently checkable range, and* Com *is a perfectly binding and computationally hiding commitment scheme, then the argument system SF-ZK in Figure 1 is computationally zero knowledge.*

Proof. We specify the zero-knowledge simulator Sim in the following. The simulator keeps a record of its running time and aborts if the number of steps exceeds 2^n.

1. During the preprocessing phase the simulator acts exactly like the honest prover, except that it commits to 1 in $c_0 \leftarrow \mathsf{Com}(1,r)$.

2. In the execution phase, it initializes a counter $i = 0$ and runs the following lookahead thread.
   (a) Commit to 0^n in $\tilde{c}$ using fresh randomness and send $\tilde{c}$.
   (b) As a response V^* either aborts or sends a response $(\tilde{w}, \beta, t)$.
   (c) If $i = 0$ check whether the verifier aborts or $(\tilde{x}, \tilde{w}) \notin \tilde{\mathcal{R}}$ and abort the whole simulation if any of these conditions are met, outputting whatever V^* outputs. Otherwise set $i := 1$ and return to step 2a.
   (d) If $i \neq 0$ check whether the verifier aborts or $(\tilde{x}, \tilde{w}) \notin \tilde{\mathcal{R}}$ and return to step 2a if this is the case. Otherwise set $i := i + 1$; if $i = 12n$ exit the loop, otherwise return to step 2a.
3. Let T be the number of iterations of the previous loop. Let $\tilde{p} := 12n/T$. Then the simulator enters in the following loop up to $(n^2/\tilde{p})$-many times.
   (a) Use the alternative witness $\tilde{w}$ to compute $\tilde{c} := \mathsf{Com}(\tilde{w}; r^*)$, using fresh random coins r^*, and send $\tilde{c}$ to the verifier.
   (b) As a response V^* either aborts or sends a second message $(\tilde{w}, \beta, t)$.
   (c) If the verifier aborts or the second message is invalid, return to step 3a, else exit the loop.
4. If $n^2/\tilde{p}$ iterations were reached without a valid $\tilde{w}$ being output by the verifier, output fail. Else use the alternative witness (r, r^*) to compute π using the second branch (2) of the HPP-NIWI proof and compute γ honestly. Send (γ, π) to the verifier.
5. The simulator outputs whatever V^* outputs.

We first bound the running time of the simulator and the probability of the simulator outputting fail.

Lemma 1. Sim *runs in expected polynomial time in* n.

Proof. Let $p(n)$ be the probability that V^* outputs a well-formed response given $\tilde{c}$ computed as in step 2a. Observe that the work of the simulator is strictly polynomial time except for the number of rewindings, therefore it is sufficient to bound the number of iterations. Note that from [36] the expected number of iterations of the first loop is exactly $\frac{12n}{p(n)}$. With this observation in mind, we distinguish between two cases

1. $\frac{p(n)}{\tilde{p}} \neq O(1)$. In this case, we use the trivial bound 2^n. However, this case can be shown to happen with negligible probability by the Chernoff bound.
2. $\frac{p(n)}{\tilde{p}} = O(1)$. In this case we can bound the running time by

$$\mathsf{poly}(n) \cdot p(n) \cdot \left(\frac{12n}{p(n)} + \frac{n^2}{\tilde{p}} \right) = \mathsf{poly}(n) \cdot \frac{p(n)}{\tilde{p}} = \mathsf{poly}(n)$$

which concludes our analysis.

Next we bound the probability that the simulator outputs fail.

Claim. The probability that Sim outputs fail is negligible in n.

Let $q(n)$ be the probability that V^* outputs a well-formed response given $\tilde{c}$ (computed as in step 3). We state and prove the following helping lemma.

Lemma 2. *There exists a negligible function such that* $q(n) \geq p(n) - \mathsf{negl}(n)$.

Proof. If $p(n)$ is negligible than it is trivial. Else it can be easily shown via a two-step argument. Let us define $\mathfrak{q}(n)$ as $q(n)$ except that in the simulation the commitment $\bar{c}$ is computed as the commitment to a random string. Note that in the real protocol the corresponding opening s is used only after the last message of V^* and therefore $q(n) = \mathfrak{q}(n) - \mathsf{negl}(n)$ by the hiding of the commitment scheme.

Recall that $p(n)$ is defined as the probability of V^* to abort given $\tilde{c} = \mathsf{Com}(0)$ using fresh randomness and $\mathfrak{q}(n)$ is defined as the probability of V^* to abort given $\tilde{c} = \mathsf{Com}(\tilde{w})$ using fresh randomness. Thus we can use V^* as a distinguisher for the commitment scheme and it will succeed with probability $p(n) - \mathfrak{q}(n)$. Since this value can be bound by a negligible function by the computational hiding of Com, we have that

$$p(n) - (q(n) + \mathsf{negl}(n)) = p(n) - \mathfrak{q}(n) \leq \mathsf{negl}(n)$$

which implies that $q(n) \geq p(n) - \mathsf{negl}(n)$ and concludes our proof.

We are now in the position of proving our claim.

Proof. Recall that the simulator outputs fail if all $\frac{n^2}{\tilde{p}}$ iterations in step 3 are not successful. We consider two cases.

1. $p(n) \leq 2 \cdot \mathsf{negl}(n)$. In this case the simulator reaches step 3 with negligible probability and therefore fail happens with the same probability.
2. $p(n) > 2 \cdot \mathsf{negl}(n)$. For conceptual simplicity we split the loop in step 3 to n independent rewinds, each upper-bounded by $\frac{n}{\tilde{p}}$ steps. Then fail happens if all of the rewinds as not successful. By a routine calculation we obtain that the expected number of iterations of each rewinding until a successful instance is found is

$$\frac{1}{q(n)} \leq \frac{1}{p(n) - \mathsf{negl}(n)} < \frac{2}{p(n)} = O\left(\frac{1}{\tilde{p}}\right),$$

   where the first inequality is by Lemma 2 and last equality is discussed above. By Markov's inequality the probability that the simulator tries more than $\frac{n}{\tilde{p}}$ iterations is at most $O(1/n)$. Since we consider n independent instances, the total probability is bounded by $O(1/n)^n$.

Finally, we show that the distribution induced by the output of the simulator is computationally indistinguishable from the honest one. Consider the following sequence of hybrids.

Hybrid H_1: The first hybrid is the interaction between the simulator Sim and the malicious verifier V^*.

Hybrid H_2: The last message γ of the SPS-ZK is simulated in time $O(2^{n_{\mathsf{SPSZK}}})$.

Hybrid H_3: The simulator inverts the one-way function to obtain $\tilde{z}$ such that $f(\tilde{z}) = y$ and uses it, together with the original witness w and the randomness r, to compute π by satisfying the third branch (3).

Hybrid H_4: The simulator computes $\tilde{c}$ as $\mathsf{Com}(0^n)$ using fresh random coins.
Hybrid H_5: The simulator no longer rewinds the verifier and simply executes the protocol in a single thread.
Hybrid H_6: The commitment $\tilde{c}$ is computed using the committed randomness $\tilde{r}$, instead of a fresh r^*.
Hybrid H_7: The simulator computes π using the original witness $(w, s, \tilde{r})$, without inverting f.
Hybrid H_8: The SPS-ZK is no longer simulated and instead computed honestly.
Hybrid H_9: The simulator now commits to 0 in $c_0 = \mathsf{Com}(0; r)$.

It is easy to see that the last hybrid exactly matches the honest execution. We will show that each δ_i defined as

$$\delta_i := |\Pr[\mathcal{D}(\langle \mathsf{H}_i(x,w), \mathsf{V}^*(x)\rangle) = 1] - \Pr[\mathcal{D}(\langle \mathsf{H}_{i+1}(x,w), \mathsf{V}^*(x)\rangle) = 1]|$$

is negligible in n. Note that the simulator Sim runs in expected (super) polynomial-time, whereas all of the following reductions must terminate in strict (super) polynomial-time. This issue can be dealt with by truncating Sim to twice its expected running time. By Markov's inequality, this reduces its success probability by at most $1/2$.

Observe that the difference in the first hybrid is that the SPS-ZK protocol is simulated in super-polynomial time. The simulator simply guesses the challenge of the verifier ahead of time and restarts the whole execution if the guess was not correct. The expected number of attempts is in the order of $O(2^{n_{\mathsf{SPSZK}}})$, however, when the simulator is successful, the transcript of the execution is statistically close to the transcript of an honest run. This bounds the value of δ_1 (and analogously of δ_7) to a negligible function.

The differences δ_2 and δ_6 can be shown to be negligible with a reduction to the witness indistinguishability of the HPP-NIWI arguments: The reduction simply sets π to be the challenge proof and returns the output of the distinguisher. Note that the random coins of the setup are not required for the simulation. The reduction runs in time $O(2^{n_{\mathsf{OWF}}}) + O(2^{n_{\mathsf{SPSZK}}}) + \mathsf{poly}(n)$ and therefore the differences among these hybrids can be bound by a negligible function (recall the parameter setup from Section 4.2).

Note that the fifth hybrid differs from the fourth only in case fail happens, however by Lemma 4.2 this happens with negligible probability and the bound on δ_4 follows. δ_3 and δ_5 can be shown to be negligible with a trivial reduction to the hiding property of the commitment scheme. Note that the reduction runs in time $O(2^{n_{\mathsf{OWF}}})+O(2^{n_{\mathsf{SPSZK}}})+\mathsf{poly}(n)$, however the commitment scheme is assumed to be hiding for machines bounded by such a runtime. The bound on δ_8 uses an identical argument except that now the reduction runs in (strict) polynomial time. We can conclude that

$$|\Pr[\mathcal{D}(\langle \mathsf{P}(x,w), \mathsf{V}^*(x)\rangle) = 1] - \Pr[\mathcal{D}(\mathsf{Sim}(x)) = 1]| \leq \sum_{i=1}^{9} \delta_i \leq \mathsf{negl}(n).$$

Theorem 5 (Computationally Unique Transcripts). *If* $(\mathsf{WI\text{-}P}, \mathsf{WI\text{-}V})$ *is an HPP-NIWI with unique proofs,* $\tilde{\mathcal{L}}$ *is an average-case hard language with unique*

witnesses, f is a one-way function, $(\mathsf{SPS\text{-}P}, \mathsf{SPS\text{-}V})$ *is an SPS-ZK argument with unique last messages, and the commitment scheme* Com *is perfectly binding and has unique openings, then the argument system SF-ZK in Figure 1 has computationally unique transcripts.*

Proof. Recall that a pair of machines $(\tilde{\mathsf{P}}^1, \tilde{\mathsf{V}}^1)$ is admissible if at least one of the two is identical to an honest generation algorithm. We treat the two cases separately.

Honest P^1. First observe that the verifier only sends the decommitment (β, t) and the witness $\tilde{w}$. Since the commitment scheme is perfectly binding and has unique decommitments and Θ verifies that the decommitment is correct, then (β, t) is uniquely determined by the preprocessing. Further, $\tilde{\mathcal{L}}$ has unique witnesses, therefore $\tilde{w}$ is also fixed by the preprocessing, for any choice of $\tilde{x}$.

On the prover's side the tuple $(\tilde{c}, \gamma, \pi)$ collects all messages sent in the execution. Since the prover's preprocessing phase is honest, c_0 is a commitment to 0. Since the commitment scheme is perfectly binding and has unique decommitments, then $\bar{c}$ from the pre-processing fixes both $(s, \tilde{r})$. If we assume towards contradiction that there exists two different accepting $\tilde{c}$ and $\hat{c}$, by the soundness of π we have that $\tilde{c} = \mathsf{Com}(0^n, \tilde{r})$ and $\hat{c} = \mathsf{Com}(0^n, \hat{r})$, where $\bar{c} = \mathsf{Com}(\tilde{r}, s)$ and $\bar{c} = \mathsf{Com}(\hat{r}, s)$. However this is a contradiction since the commitment has unique openings. It follows that $\tilde{r} = \hat{r}$ and therefore $\tilde{c}$ is unique. Recall that both the HPP-NIWI and the SPS-ZK have unique last messages, and therefore (γ, π) are uniquely determined by the pre-processing.

Honest V^1. Given an honest verifier pre-processing p_2 and a (possibly malicious) prover pre-processing p_1 for a certain statement x with unique witnesses, let $\mathcal{T}_1 := (\tilde{c}, \beta, t, \tilde{w}, \pi, \gamma)$ and $\mathcal{T}_2 := (\hat{c}, \hat{\beta}, \hat{t}, \hat{w}, \hat{\pi}, \hat{\gamma})$ be the two transcripts such that $\Theta(p_1, p_2, \mathcal{T}_1, x) = \Theta(p_1, p_2, \mathcal{T}_2, x) = 1$. We shall prove that $\mathcal{T}_1 = \mathcal{T}_2$ with all but negligible probability.

$\underline{(\beta, t) = (\hat{\beta}, \hat{t})}$: Since the commitment scheme is perfectly binding and has unique openings and $c = \mathsf{Com}(\beta; t)$ is fixed in the pre-processing, this equality must hold.

$\underline{\tilde{w} = \hat{w}}$: The witness is uniquely determined by the statement $\tilde{x}$, since $\tilde{\mathcal{L}}$ has unique witnesses.

$\underline{\gamma = \hat{\gamma}}$: Fix τ and (α, β), which are all part of the pre-processing, then γ is unique since the SPS-ZK has unique last messages.

$\underline{\pi = \hat{\pi}}$: First note that there must exist some u such that $\mathsf{WI\text{-}P}_1(u) = \tau$. To see why this is the case, recall either the transcript of the SPS-ZK proof uniquely determines the witness or α (together with the randomness used to compute it) uniquely determines the challenge β. If a valid u does not exist, then we are left with the latter case, which implies that we can guess the content of c running in time $O(2^{n_{\mathsf{SPSZK}}})$. This contradicts the hiding property of Com (refer to Section 2 for further discussion). It follows that τ is well-formed except with negligible probability.

Therefore both π and $\hat{\pi}$ are generated using the witness for one of the following branches:

$$\begin{aligned} & \exists(s,\tilde{r}) : (x^* \in \mathcal{L} \wedge \mathsf{Com}(\tilde{r};s) = \bar{c} \wedge \mathsf{Com}(0^n,\tilde{r}) = \tilde{c}) \\ \vee\ & \exists(r,\tilde{r}) : (\mathsf{Com}(1;r) = c_0 \wedge \mathsf{Com}(\tilde{w},\tilde{r}) = \tilde{c}) \\ \vee\ & \exists(r,z) : (x^* \in \mathcal{L} \wedge \mathsf{Com}(1;r) = c_0 \wedge f(z) = y) \end{aligned}$$

We bound the probability that π or $\hat{\pi}$ is a valid proof for the second branch (2) in the following. Assume without loss of generality that such proof is π. Since the commitment scheme is perfectly binding we can extract $\tilde{w}$ from $\tilde{c}$ (by exhaustive search) in time $O(2^{n_{\mathsf{COM}}})$. Note that the extraction is successful with probability 1 since $\tilde{c}$ is perfectly binding. Recall that $\tilde{\mathsf{V}}^1$ is honest by assumption and therefore we can plug in a hard instance $\tilde{x}$ and break the average-case hardness of $\tilde{\mathcal{L}}$ from the first message of the prover.

On the other hand, if the third branch (3) is proven with non-negligible probability then we can invert y in time $O(2^{n_{\mathsf{SPSZK}}}) + \mathsf{poly}(n)$ by extracting u from (α, β, γ) and running the polynomial-time extractor of the HPP-NIWI proof. It follows that both proofs are for the first branch (1), which implies that they are identical.

$\underline{\tilde{c} = \hat{c}}$: As we argued above, π must be a proof for the first branch. Since $(s, \tilde{r})$ are fixed in the pre-processing by $\bar{c}$, then $\tilde{c}$ is also uniquely determined, unless π is a proof for a false statement. This happens only with negligible probability. □

Acknowledgements. Behzad Abdolmaleki and Giulio Malavolta were supported by the German Federal Ministry of Education and Research BMBF (grant 16K15K042, project 6GEM). Nils Fleischhacker and Giulio Malavolta were supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2092 CASA - 390781972. Vipul Goyal was supported by the NSF award 1916939, DARPA SIEVE program under Agreement No. HR00112020025, a gift from Ripple, a DoE NETL award, a JP Morgan Faculty Fellowship, a PNC center for financial services innovation award, and a Cylab seed funding award. Abhishek Jain was supported in part by NSF CNS-1814919, NSF CAREER 1942789, Johns Hopkins University Catalyst award, AFOSR Award FA9550-19-1-0200 and the Office of Naval Research Grant N00014-19-1-2294.

References

1. Joël Alwen, abhi shelat, and Ivan Visconti. Collusion-free protocols in the mediated model. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 497–514, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Heidelberg, Germany.
2. Benedikt Auerbach, Mihir Bellare, and Eike Kiltz. Public-key encryption resistant to parameter subversion and its realization from efficiently-embeddable groups. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018: 21st International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 10769 of *Lecture Notes in Computer Science*, pages 348–377, Rio de Janeiro, Brazil, March 25–29, 2018. Springer, Heidelberg, Germany.

3. Michael Backes and Christian Cachin. Public-key steganography with active attacks. In Joe Kilian, editor, *TCC 2005: 2nd Theory of Cryptography Conference*, volume 3378 of *Lecture Notes in Computer Science*, pages 210–226, Cambridge, MA, USA, February 10–12, 2005. Springer, Heidelberg, Germany.
4. Mihir Bellare, Georg Fuchsbauer, and Alessandra Scafuro. NIZKs with an untrusted CRS: Security in the face of parameter subversion. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016, Part II*, volume 10032 of *Lecture Notes in Computer Science*, pages 777–804, Hanoi, Vietnam, December 4–8, 2016. Springer, Heidelberg, Germany.
5. Mihir Bellare and Viet Tung Hoang. Resisting randomness subversion: Fast deterministic and hedged public-key encryption in the standard model. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 627–656, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany.
6. 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: 19th Conference on Computer and Communications Security*, pages 784–796, Raleigh, NC, USA, October 16–18, 2012. ACM Press.
7. Mihir Bellare, Joseph Jaeger, and Daniel Kane. Mass-surveillance without the state: Strongly undetectable algorithm-substitution attacks. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *ACM CCS 2015: 22nd Conference on Computer and Communications Security*, pages 1431–1440, Denver, CO, USA, October 12–16, 2015. ACM Press.
8. Mihir Bellare, Kenneth G. Paterson, and Phillip Rogaway. Security of symmetric encryption against mass surveillance. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 1–19, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany.
9. Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible protocols and atomic proxy cryptography. In Kaisa Nyberg, editor, *Advances in Cryptology – EUROCRYPT'98*, volume 1403 of *Lecture Notes in Computer Science*, pages 127–144, Espoo, Finland, May 31 – June 4, 1998. Springer, Heidelberg, Germany.
10. Zvika Brakerski, Sanjam Garg, and Rotem Tsabary. Fhe-based bootstrapping of designated-prover nizk. In *Pass R., Pietrzak K. (eds) Theory of Cryptography. TCC 2020. Lecture Notes in Computer Science, vol 12550. Springer, Cham*, 2020.
11. Mike Burmester and Yvo Desmedt. Broadcast interactive proofs (extended abstract). In Donald W. Davies, editor, *Advances in Cryptology – EUROCRYPT'91*, volume 547 of *Lecture Notes in Computer Science*, pages 81–95, Brighton, UK, April 8–11, 1991. Springer, Heidelberg, Germany.
12. Mike Burmester, Yvo Desmedt, Toshiya Itoh, Kouichi Sakurai, and Hiroki Shizuya. Divertible and subliminal-free zero-knowledge proofs for languages. *Journal of Cryptology*, 12(3):197–223, June 1999.
13. Mike Burmester, Yvo Desmedt, Toshiya Itoh, Kouichi Sakurai, Hiroki Shizuya, and Moti Yung. A progress report on subliminal-free channels. In Ross J. Anderson, editor, *Information Hiding, First International Workshop, Cambridge, UK, May 30 - June 1, 1996, Proceedings*, volume 1174 of *Lecture Notes in Computer Science*, pages 157–168. Springer, 1996.
14. Christian Cachin. An information-theoretic model for steganography. *Inf. Comput.*, 192(1):41–56, July 2004.

15. Suvradip Chakraborty, Stefan Dziembowski, and Jesper Buus Nielsen. Reverse firewalls for actively secure mpcs. In *Advances in Cryptology, 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA*, pages 732–762, 2020.
16. Suvradip Chakraborty, Chaya Ganesh, Mahak Pancholi, and Pratik Sarkar. Reverse firewalls for adaptively secure mpc without setup. In *Advances in Cryptology, ASIACRYPT 2021, Tibouchi, Mehdi and Wang, Huaxiong, Springer International Publishing*, pages 335–364, 2021.
17. Jean Paul Degabriele, Kenneth G. Paterson, Jacob C. N. Schuldt, and Joanne Woodage. Backdoors in pseudorandom number generators: Possibility and impossibility results. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 403–432, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany.
18. Yvo Desmedt. Abuses in cryptography and how to fight them. In Shafi Goldwasser, editor, *Advances in Cryptology – CRYPTO'88*, volume 403 of *Lecture Notes in Computer Science*, pages 375–389, Santa Barbara, CA, USA, August 21–25, 1990. Springer, Heidelberg, Germany.
19. Yevgeniy Dodis, Chaya Ganesh, Alexander Golovnev, Ari Juels, and Thomas Ristenpart. A formal treatment of backdoored pseudorandom generators. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 101–126, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany.
20. Yevgeniy Dodis, Ilya Mironov, and Noah Stephens-Davidowitz. Message transmission with reverse firewalls—secure communication on corrupted machines. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 341–372, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany.
21. Yevgeniy Dodis, Shien Jin Ong, Manoj Prabhakaran, and Amit Sahai. On the (im)possibility of cryptography with imperfect randomness. In *45th Annual Symposium on Foundations of Computer Science*, pages 196–205, Rome, Italy, October 17–19, 2004. IEEE Computer Society Press.
22. Marc Fischlin and Sogol Mazaheri. Self-guarding cryptographic protocols against algorithm substitution attacks. In *IEEE 31st Computer Security Foundations Symposium (CSF)*, 2018.
23. Chaya Ganesh, Bernardo Magri, and Daniele Venturi. Cryptographic reverse firewalls for interactive proof systems. Cryptology ePrint Archive, Report 2020/204, 2020.
24. Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st Annual ACM Symposium on Theory of Computing*, pages 169–178, Bethesda, MD, USA, May 31 – June 2, 2009. ACM Press.
25. Oded Goldreich and Ariel Kahan. How to construct constant-round zero-knowledge proof systems for NP. *Journal of Cryptology*, 9(3):167–190, June 1996.
26. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th Annual ACM Symposium on Theory of Computing*, pages 218–229, New York City, NY, USA, May 25–27, 1987. ACM Press.
27. Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *17th Annual ACM Symposium*

on Theory of Computing, pages 291–304, Providence, RI, USA, May 6–8, 1985. ACM Press.
28. Carmit Hazay and Muthuramakrishnan Venkitasubramaniam. On the power of secure two-party computation. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 397–429, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany.
29. Nicholas J. Hopper, John Langford, and Luis von Ahn. Provably secure steganography. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 77–92, Santa Barbara, CA, USA, August 18–22, 2002. Springer, Heidelberg, Germany.
30. Yael Tauman Kalai, Bhavana Kanukurthi, and Amit Sahai. Cryptography with tamperable and leaky memory. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 373–390, Santa Barbara, CA, USA, August 14–18, 2011. Springer, Heidelberg, Germany.
31. Stefan Katzenbeisser and Fabien A.P. Petitcolas. Defining security in steganographic systems, 2002.
32. Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *24th Annual ACM Symposium on Theory of Computing*, pages 723–732, Victoria, BC, Canada, May 4–6, 1992. ACM Press.
33. Dror Lapidot and Adi Shamir. Publicly verifiable non-interactive zero-knowledge proofs. In Alfred J. Menezes and Scott A. Vanstone, editors, *Advances in Cryptology – CRYPTO'90*, volume 537 of *Lecture Notes in Computer Science*, pages 353–365, Santa Barbara, CA, USA, August 11–15, 1991. Springer, Heidelberg, Germany.
34. Matt Lepinski, Silvio Micali, and abhi shelat. Collusion-free protocols. In Harold N. Gabow and Ronald Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 543–552, Baltimore, MA, USA, May 22–24, 2005. ACM Press.
35. Matt Lepinski, Silvio Micali, and abhi shelat. Fair-zero knowledge. In Joe Kilian, editor, *TCC 2005: 2nd Theory of Cryptography Conference*, volume 3378 of *Lecture Notes in Computer Science*, pages 245–263, Cambridge, MA, USA, February 10–12, 2005. Springer, Heidelberg, Germany.
36. Yehuda Lindell. How to simulate it - a tutorial on the simulation proof technique. Cryptology ePrint Archive, Report 2016/046, 2016. https://ia.cr/2016/046.
37. Yehuda Lindell and Benny Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. In Moni Naor, editor, *Advances in Cryptology – EUROCRYPT 2007*, volume 4515 of *Lecture Notes in Computer Science*, pages 52–78, Barcelona, Spain, May 20–24, 2007. Springer, Heidelberg, Germany.
38. Ilya Mironov and Noah Stephens-Davidowitz. Cryptographic reverse firewalls. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 657–686, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany.
39. Tatsuaki Okamoto and Kazuo Ohta. How to utilize the randomness of zero-knowledge proofs. In Alfred J. Menezes and Scott A. Vanstone, editors, *Advances in Cryptology – CRYPTO'90*, volume 537 of *Lecture Notes in Computer Science*, pages 456–475, Santa Barbara, CA, USA, August 11–15, 1991. Springer, Heidelberg, Germany.
40. Rafael Pass. Simulation in quasi-polynomial time, and its application to protocol composition. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*,

volume 2656 of *Lecture Notes in Computer Science*, pages 160–176, Warsaw, Poland, May 4–8, 2003. Springer, Heidelberg, Germany.
41. Alexander Russell, Qiang Tang, Moti Yung, and Hong-Sheng Zhou. Cliptography: Clipping the power of kleptographic attacks. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016, Part II*, volume 10032 of *Lecture Notes in Computer Science*, pages 34–64, Hanoi, Vietnam, December 4–8, 2016. Springer, Heidelberg, Germany.
42. Alexander Russell, Qiang Tang, Moti Yung, and Hong-Sheng Zhou. Generic semantic security against a kleptographic adversary. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017: 24th Conference on Computer and Communications Security*, pages 907–922, Dallas, TX, USA, October 31 – November 2, 2017. ACM Press.
43. Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *46th Annual ACM Symposium on Theory of Computing*, pages 475–484, New York, NY, USA, May 31 – June 3, 2014. ACM Press.
44. Adam Young and Moti Yung. Kleptography: Using cryptography against cryptography. In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 62–74, Konstanz, Germany, May 11–15, 1997. Springer, Heidelberg, Germany.
45. Adam Young and Moti Yung. The prevalence of kleptographic attacks on discrete-log based cryptosystems. In Burton S. Kaliski Jr., editor, *Advances in Cryptology – CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 264–276, Santa Barbara, CA, USA, August 17–21, 1997. Springer, Heidelberg, Germany.