Publicly Verifiable Zero Knowledge from (Collapsing) Blockchains

Alessandra Scafuro[1(✉)], Luisa Siniscalchi[2], and Ivan Visconti[3]

[1] North Carolina State University, Raleigh, USA
ascafur@ncsu.edu
[2] Concordium Blockchain Research Center, Aarhus University, Aarhus, Denmark
lsiniscalchi@cs.au.dk
[3] DIEM, University of Salerno, Fisciano, Italy
visconti@unisa.it

Abstract. Publicly Verifiable Zero-Knowledge proofs are known to exist only from setup assumptions such as a trusted common reference string or a random oracle. Unfortunately, the former requires a trusted party while the latter does not exist.

Blockchains are distributed systems that already exist and provide certain security properties (under some honest majority assumption), hence, a natural recent research direction has been to use a blockchain as an alternative setup assumption.

In TCC 2017 Goyal and Goyal proposed a construction of a publicly verifiable zero-knowledge (pvZK) proof system for some proof-of-stake blockchains. The zero-knowledge property of their construction however relies on some additional and not fully specified assumptions about the current and future behavior of honest blockchain players.

In this paper we provide several contributions. First, we show that when using a blockchain to design a provably secure protocol, it is dangerous to rely on demanding additional requirements on behaviors of the blockchain players. We do so by showing an "attack of the clones" whereby a malicious verifier can use a smart contract to slyly (not through bribing) clone capabilities of honest stakeholders and use those to invalidate the zero-knowledge property of the proof system by Goyal and Goyal.

Second, we propose a new publicly verifiable zero-knowledge proof system that relies on non-interactive commitments and on an assumption on the min-entropy of some blocks appearing on the blockchain.

Third, motivated by the fact that blockchains are a recent innovation and their resilience in the long run is still controversial, we introduce the concept of collapsing blockchain, and we prove that the zero-knowledge property of our scheme holds even if the blockchain eventually becomes insecure and all blockchain players eventually become dishonest.

Keywords: Publicly verifiable zero knowledge · (Collapsing) blockchain

J. A. Garay (Ed.): PKC 2021, LNCS 12711, pp. 469–498, 2021.
https://doi.org/10.1007/978-3-030-75248-4_17

1 Introduction

Following the success of Bitcoin many other cryptocurrencies based on blockchain technology have been proposed and, despite a few security issues, they are still expanding their networks with gigantic market capitalizations. What is so appealing in decentralized blockchains?

Public Verifiability. One of the most supported answers is the paradigm shift from trust in some entity to "public verifiability". This property allows every one to check that the system works consistently with the pre-specified rules of the game. This makes users willing to be involved in transactions recorded in a blockchain therefore investing their real-world money. In many blockchain applications both anonymity and public verifiability are required, calling for advanced cryptographic primitives such as publicly verifiable zero-knowledge proofs. For example, when the blockchain is used to record payments, confidential transactions are indeed implemented using publicly verifiable zero-knowledge proofs called zk-SNARKs [9,20].

Publicly Verifiable Zero-Knowledge Proofs. Known constructions of publicly verifiable zero-knowledge (pvZK) proofs are instantiated with *non-interactive* zero-knowledge proofs (NIZK) and, as such, require setup assumptions. Indeed, despite a significant effort of the research community, constructions of NIZK proofs either rely on the existence of a trusted common reference string (CRS) computed by a trusted entity or are based on heuristic assumptions (e.g., random oracles). Recent existing work has shown mechanisms to relax the trust assumptions required to generate the CRS [14] or to mitigate the effect of a malicious CRS [26,30]. While this line of work is very promising, it still requires the employment of third entities that help computing the CRS.

Publicly Verifiable Zero-Knowledge Proofs from a "Blockchain Assumption". Since its introduction in 2008 with Nakamoto's protocol [31], blockchain protocols have been scrutinized by many communities, and currently, we have a good understanding of the security properties they provide and the class of adversaries they withstand. In particular, several works from the cryptographic community provided a formalization of the Bitcoin security guarantees [19,32], a formalization of the ideal functionality it implements [5] as well as game-theoretic analysis [3]. Furthermore, new blockchain designs have been proposed, based on different assumptions on the collective power of the adversary. Some prominent examples that are also implemented in practice are Ouroboros [4] and Algorand [22].

Given that blockchains have been formally analyzed and are up and running in practice, a natural question to ask is whether we can use a blockchain as a setup assumption *to replace* trusted setups required for certain cryptographic tasks, particularly, for publicly verifiable zero-knowledge proof systems that are needed the most in blockchain applications.

This question was first investigated by Goyal and Goyal in [23], where they aimed to construct NIZK using as setup the existence of a proof-of-stake (PoS) blockchain. The security of the NIZK proof provided in [23] – that we will denote by GG-NIZK– however is analyzed in a threat model that does not faithfully match the threat model of PoS blockchains, since it considers only static adversaries and additionally requires that honest stakeholder never reveal their secret keys. Specifically, the zero-knowledge property of GG-NIZK is proved in the presence of a *static adversary* who decides in advance which stakeholder will corrupt in its entire attack. This does not match the widely accepted threat model for proof-of-stake blockchains where an adversary is allowed to corrupt stakeholders at any time, and the only restriction is that, at any point, the total amount of stake held by the adversary is a minority of the total stake of the system. Moreover, in the GG-NIZK security analysis, the zero-knowledge property holds under the additional assumption that honest stakeholders will never leak their stakeholder keys, not even when such keys become irrelevant for the blockchain protocol (for example, because there is zero stake associated to them).

It was observed in [34] that the assumption on stakeholder keys further limits the generality of GG-NIZK since it cannot be used in conjunction with *any* proof-of-stake (PoS) blockchain. In particular [34] observes that one could design a PoS blockchain where stakeholders are required to often refresh their stakeholder keys, by regularly publishing new public keys and voiding old keys by posting their secret keys on the blockchain. Such blockchain protocol, while being a potentially valid PoS blockchain protocol, cannot be used to instantiate GG-NIZK.

The full version of [23] has been recently updated [24] adding a section in the appendix where the authors confirm the security of their construction even in light of the counter-example of [34] by stressing that they expect honest stakeholders to delete keys when they lose significance.

In light of the observations of [34] and of the counter-argument of [24], a natural question to ask is whether such additional assumptions/expectations on the behavior of honest stakeholders required in [23,24] could be symptomatic of unexpected security flaws that would manifest when GG-NIZK is executed with a *real blockchain environment*, *even one that complies with all GG-NIZK assumptions/expectations*. In other words, assuming that the additional restrictions on the power of the adversary and the behavior of honest stakeholders are met, would GG-NIZK be actually secure when executed in the presence of a PoS blockchain that complies with them?

A negative answer to the above question would signify that constructing a publicly verifiable zero-knowledge proof that leverages any blockchain assumption is still an open question.

1.1 Our Contribution

In this paper we target the problem of constructing publicly verifiable zero-knowledge proofs leveraging a blockchain assumption and provide the following contributions.

A More Realistic Blockchain Threat Model. We consider a model where the blockchain can potentially be used to post and fulfill arbitrary smart contracts. Since all existent blockchain protocols either already support or aim to support smart contracts capabilities (e.g., Ethereum Casper, Cardano) and, since smart contracts are among the most appealing feature of blockchains, this model is arguably realistic. Within this model, an adversary can consequently also leverage her ability to publish smart contracts just like any party who uses the blockchain.

Within this threat model, we show that the zero-knowledge property of GG-NIZK is easily violated *even assuming that all restrictions required by the security analysis of GG-NIZK are satisfied*, that is, even assuming that the adversary can only perform static corruption and that honest stakeholders will never reveal their keys. Specifically, we present an adversarial strategy that leverages legitimate smart contracts to collect information that are useful to disturb the security of the external cryptographic protocols that use the blockchain as a building block. We name this type of attacks "attack of the clones" to highlight the adversary's aim to *clone* the capability of a honest player to perform computations using her secret key. However, the smart contract posted by the adversary is completely harmless for a honest stakeholder. Indeed, it does not ask the stakeholder to do anything that will make her lose her stake, or perform any operation against the consensus protocol. Yet, it allows the adversary to break the zero knowledge of the GG-NIZK proof. Our attack leverages a specific dangerous use of stakeholder identifiers in the GG-NIZK. The starting point is that the NIZK proof of [23] includes encryptions of shares of the witness under the public keys inferred by the identifiers of the stakeholders. To break the zero-knowledge property of the NIZK of [23] our attack is rather simple: after the NIZK proof π is received, the adversarial verifier posts a smart contract containing ciphertexts (these are the ciphertext contained in π) and a promised reward (e.g., money, raffle tickets for a vacation in Barbados, etc.) in exchange for decryptions.

Notice that an honest stakeholder participating in this smart contract remains fully honest, does not subtract any resource (unlike in bribing attacks against proof-of-work blockchains) from the participation to the consensus protocol and does not reveal her secret keys to anyone. She just plays with smart contracts as contemplated by the blockchain rules and uses her stake for some harmless entertainment. Indeed, the crux of this attack is that a stakeholder is not aware that an external cryptographic protocol is basing its zero-knowledge property on the assumption that stakeholders would not entertain in smart contracts that are harmless for the underlying blockchain protocol.

One might object that it is plausible that a PoS blockchain would simply forbid the execution of such "weird" smart contracts. However, it is not clear what a "weird" smart contract is, and whether the above smart contract could be redesigned in order to look innocent and harmless (furthermore, the well known DAO attack inflicted to Ethereum suggests that it is unclear whether we are able to identify and stop an harmful smart contract too much in advance).

Our attack is obviously a simple example and after-the-fact can possibly be mitigated, for instance by adding specific further restrictions on how the stakeholder should use her secret keys. However, the point of our attack is not prove that there is no blockchain for which GG-NIZK can be secure. Instead, we want to highlight the vulnerabilities arising when the long-term security of a cryptographic protocol relies on the behavior of blockchain players.

The main lesson of our attack is the following: when designing protocols that leverage a blockchain assumption, one has to consider a threat model where the adversary is allowed to perform the *same actions that are allowed on the blockchain* (e.g., run smart contracts[1]). Note that this should be true even when analyzing the consensus protocol itself. However, since this is out of the scope of this paper, we assume that the underlying blockchain consensus protocol is secure in the presence of smart contracts.

Another lesson to be drawn by our attack is that, when using the blockchain as an underlying assumption, one should take into account the unstable and evolving nature of blockchains. Unlike a common reference string, blockchains evolve over time –due to software updates for example, or governance decisions– stake is transferred among players, new smart contracts are installed etc. Last but not least one might take into account the possibility that a blockchain that todays is reliable tomorrow could collapse and could then be completely controlled by an adversary.

The above attack on the ZK of GG-NIZK leaves open the following natural question.

> Can we design a pvZK proof leveraging the existence of blockchains, that makes no particular assumption on the underlying consensus mechanism neither on the way honest keys must be used (for instance, they can still be used in smart contracts)?

Publicly Verifiable Zero Knowledge from a Generic Blockchain in Our Threat Model. As a second contribution we provide a new protocol for pvZK that is secure in the blockchain threat model discussed above even in the presence of adaptive adversaries. To show this security guarantee, we will prove that once the proof (computed using our protocol) is published, it will preserve its security even if the blockchain collapses, that is, even if the adversary corrupts all the players of the blockchain (and gets all the secrets). We now proceed describing our protocol and our blockchain assumption.

A recent work by Choudhuri et al. [16] shows that using a blockchain as a black-box object that provides only a global ledger does not allow to overcome some impossibility results in the plain model and in particular it does not allow

[1] We note that this threat model was never considered before. [34] only made observations about additional limitations that GG-NIZK imposes on their underlying blockchain. Instead, in this work we are showing an attack that works for any PoS blockchain (even the ones that comply with GG-NIZK pre-requirement) allowing the execution of such smart contracts.

to construct NIZK proofs. We notice that their argument can be extended also to pvZK proofs (see Sect. 5 for more details). Therefore, in order to build a publicly verifiable zero-knowledge proof system from a blockchain, it seems that one needs to provide more power to the simulator besides black-box access to a global ledger. Thus, following [23] we will assume that the simulator has the power of controlling the honest players. However, unlike [23] we assume that the adversary can adaptively corrupt players and moreover we want our pvZK proof to remain zero knowledge even in case of blockchain failure, in the sense that in the future the adversary might take full control over the blockchain.

To leverage this simulation power while making no assumption on the consensus protocol underlying the blockchain (i.e., we do not assume that the blockchain is based on proof-of-work, proof-of-stake, etc.), we require that the blockchain satisfies a more nuanced notion of chain quality. Very informally (a formal definition is provided in Assumption 1) we assume the blockchain has the following mild structure. First, every block contains a distinguished field v. For concreteness, the reader can assume that this field is the same as the "coinbase" value of any Bitcoin block, and to ease the discussion, in the text that follows, we will call this field wallet. Our blockchain assumption, very roughly, is that there exists a parameter d, such that, for any sequence of d blocks, considering the new wallets[2] observed in the sequence, we have that a majority of those wallets has been generated by honest players using independent randomnesses. Essentially our blockchain assumption builds on top of the standard chain quality assumption, requiring that the adversary will be the "winning" node that decides the next block using a fresh wallet less often than honest players. Similar assumptions have been leveraged in the literature. For example [25,33] use the assumption that the majority of mined blocks are honest, to select a committee for secure computation. The difference between our blockchain assumption and the standard chain quality property is mainly that we additionally require that many of the honest blocks will additionally have an high-min entropy field. We discuss more extensively our blockchain assumption Sect. 3.1.

We will leverage this blockchain assumption and the simulator's control of the honest majority to build a pvZK proof as follows. The high-level idea is to follow the FLS approach [17] and prove the OR of two statements: either "x in L" or "Previously I have predicted the majority of fresh wallets appeared in the last d blocks". In particular our idea reminds the implementation of the FLS approach proposed by Barak [7] where the trapdoor theorem consists of some unpredictable information that becomes predictable during the straight-line simulation. The soundness of our construction will follow from similar arguments and will actually be simpler. The reason is that we implement the prediction step with perfectly binding commitments and thus, unlike Barak, we will not have to worry about a prover finding collisions in a collision-resistant hash function.

To implement this approach we need two ingredients: a non-interactive commitment scheme (that can be constructed from 1–1 one-way functions) and a

[2] Here we refer to wallets identifying the block leader cashing the reward and not to wallets involved in transactions.

publicly verifiable witness indistinguishable proof system pvWI. We use the pvWI proof system recently constructed in [34] which is the first pvWI proof system from a blockchain assumption. Our blockchain assumption implies the one of [34]. The pvWI proof system of [34] leverages the underlying blockchain assumption by providing an interactive prover and a non-interactive verification function. Concretely, the pvWI proof of [34] builds on a classic 3-round WI proof system where the first two rounds are played by the prover and blockchain: the prover posts the first round of the classic WI proof on the blockchain, then she waits for a few blocks extending the block containing the first message and from those extracts a challenge that corresponds to the second round of a classic 3-round WI proof. The third round of the classic WI proof is then sent to the actual verifier, who can use the blockchain to validate all 3 rounds, non-interactively. If the third round is posted on the blockchain then all verifiers can validate the proof. We need the following 3 properties from the pvWI proof: (1) *delayed-input* completeness, which means that the prover will use the theorem and the witness only for computing the last message of the protocol, which implies that all other messages of the pvWI proof are independent from the witness; (2) *WI in the presence of blockchain failure*, that is, (2.1) the WI property holds even when the prover is the only honest player and therefore the blockchain could be completely controlled by the adversary; (2.2) the WI property is preserved even when, *after* a pvWI proof is computed, the adversary could corrupt the prover; (3) unconditional soundness[3] in the presence of our blockchain assumption (i.e., Assumption 1). Since such properties were not explicitly claimed in [34] we show in the full version of this paper [35] that through minor updates to their protocol those 3 properties are satisfied. The reason why we need the above 3 special properties will be explained later when we will highlight the security proof.

With the above ingredients in hands, our pvZK proof system works as follows. First, the prover, using a non-interactive commitment scheme, commits to $u \cdot d$ strings $\mathtt{com}_1, \ldots, \mathtt{com}_{u \cdot d}$ (u is the blockchain parameters associated to our chain-quality assumption (Assumption 1), more details about u will be provided later) and posts the commitments on the blockchain. Note that the prover securely erases the decommitment information of $\mathtt{com}_1, \ldots, \mathtt{com}_{u \cdot d}$. Then, she waits until the blockchain is extended by a sequence of d blocks $\overline{B}_1, \ldots, \overline{B}_d$, that include n blocks $B_1, \ldots, B_n$ with fresh wallets (that is, with wallets that were not observed before). Let $v_1, \ldots, v_n$ be such fresh wallets observed on the blockchain. In the final step, the prover computes the pvWI proof, for the theorem:"$x \in \mathcal{L}$ or $(\mathtt{com}_1, \ldots, \mathtt{com}_{u \cdot d})$ are commitments of at least $n/2+1$ of the wallets $(v_1, \ldots, v_n)$".

The simulator $\mathsf{S}_{\mathsf{pvZK}}$ uses the same power of the simulator of [23] controlling the honest players in the simulated experiment (in particular, the simulator adds the blocks in the blockchain on behalf of honest players). Therefore $\mathsf{S}_{\mathsf{pvZK}}$ can predict the majority of the unpredictable new wallets associated with a sequence of d future blocks, and can use this knowledge as a trapdoor theorem when computing the messages of the pvWI proof. Notice that the simulator can not tightly predict the future wallets that will be permanently added to the

[3] See the paragraph below about the power of the adversary.

blockchain since there will be several other honest blocks to simulate that will circulate in the network, they might even appear in some forks but eventually will not be part of the blockchain. Since the simulator has no direct power to decide which branch of a fork will remain in the blockchain, we require way more than just d commitments. Indeed we consider the parameter u that measures the upper bound on the amount of valid blocks that honest players propose for each index of the sequence of blocks of the blockchain.

The pvZK that we construct preserves zero knowledge even in case of adaptive corruption during the protocol execution and in case the blockchain completely collapses and the adversary gets the state of all players. To achieve this strong form of zero knowledge, we use secure erasure so that differences in the committed values are not detected. Moreover we rely on the delayed-input pvWI so that the simulator can run the prover procedure of the pvWI except that a different witness is used in last message. Therefore before the last message is played, adaptive corruption is not harmful since the simulator played exactly like a prover of the pvWI. Assuming that the underlying pvWI is secure in case the blockchain collapses (fact that we prove), the proof remains zero-knowledge forever.

A crucial aspect of our construction is that the security of the prover is in the hands of the prover *only* and does not depend on the behavior of the stakeholders. To achieve adaptive security we rely on secure erasure. In contrast, even if the prover of GG-NIZK would erase its randomness, the proof would still suffer of our attack.

For the soundness proof, the main observation is that, as long as our blockchain assumption holds, even an unbounded malicious prover cannot break soundness since it cannot predict enough future wallets. This together with the perfect binding property of the commitment scheme and the unconditional soundness of the pvWI guarantees the soundness of our pvZK.

An additional property of our construction is that all messages except the last one can be computed even before knowing the statement to prove (i.e., it satisfies delayed-input completeness and adaptive-input zero knowledge and soundness).

Finally, we remark that even though messages of our pvZK proof can be very long, therefore exceeding some rule of the blockchain, one can anyway resort to techniques (and assumptions) like IPFS that allow to keep off-chain long message but still accessible by everyone and succinctly notarized on chain.

On the Computational Power of the Adversary and Rationality of Players. In a publicly verifiable proof assuming that an adversarial prover is PPT does not really say much about his limits with respect to the security of the blockchain. Indeed in case of proof-of-work blockchains the limitation of the adversary should be compared to the overall computational capabilities of the network rather than compared to a generic polynomial on input the security parameter. In our definition of soundness we will therefore consider an unbounded prover. When proving the security of our construction we will state explicitly our blockchain

assumption and implicitly we will assume that the constraints on the adversary (see Sect. 3.2) required by the underlying blockchain are maintained.

We remark that this work following [23,24,34] considers either honest or corrupted players, without exploring the game-theoretic scenario where players are instead rational.

1.2 Related Work

The idea of using a blockchain as a trusted setup has been explored already (e.g., fair multi-party computation [12], extraction of week randomness [2]). In [11] a randomness beacon is obtained assuming players to be somehow rational (i.e., they assume that the adversary that will prefer to be honest cashing mining rewards rather than misbehaving compromising the beacon). In our work, as well as the one of [23], we consider zero-knowledge proofs with public verifiability sticking with the traditional setting where security holds against malicious players.

In [16] a blockchain is used as a global setup assumption to obtain concurrent self-composable secure computation protocol, which is impossible in the standard model. We stress that [16] does not provide public verifiability (for the interested reader we expand this discussion in Sect. 5). Recently in [10,25] a blockchain is used to maintain a secret via proactive secrete sharing. As mentioned above[25] requires some chain quality parameters $(\frac{n}{2}+1, n)$ which means that for any sequence of n blocks, the majority of them $\frac{n}{2}+1$ are computed by honest parties. In [10] the adversary controls up to 25% of the stake. However using the technique discussed in [21] one could lift up this requirement to less than 50%.

In [6] the notion of Crowd verifiable zero-knowledge (CVZK) is introduced[4]. In CVZK a prover wants to convince a set of n verifiers of the validity of a certain statement. In more detail, a CVZK is a 3-round protocol where first the prover speaks, then n verifiers compute a private state and send as a second-round a string that may contain some entropy, finally, the prover finishes the proof π. The verification procedure takes as input π the corresponding statement and also the states of the n verifiers. Instead we consider a different notion requiring a zero-knowledge proof that is publicly verifiable (i.e., any verifier with no additional information could check the veracity of the statement). Moreover, the definition of CVZK does not require any setup at the price of allowing the simulator to run in super-polynomial time. Our goal is also to diminish the trust in the setups, however, instead of requiring super-polynomial time simulation, we exploit more realistic setups like the blockchains.

2 The Attack of the Clones to GG-NIZK [23]

A high-level overview of the NIZK presented in [23] was provided in the Introduction. In this section we describe an attack of clones with which a malicious

[4] Our results were publicly announced in [1] way before we have noticed CVZK, therefore the two works are independent.

verifier, using a smart contract, is able to break the zero-knowledge property of GG-NIZK without corrupting any player.

Our attack leverages the fact that, if a blockchain is used as setup assumption for a protocol Π, the security proof of Π must take into account the fact that a player of Π is also a legitimate player of the blockchain protocol. As such, legitimate blockchain activities – such as smart contracts – can be performed by her.

Before describing the attack, we provide a formal description of the GG-NIZK.

Notation for GG-NIZK

- Blockchain $\mathbf{B}$: this is the latest version blockchain which might contain unconfirmed blocks.
- Stable Blockchain $\mathbf{B}'$: this is defined as $\mathbf{B}^{\lceil \ell_1}$, which is the blockchain $\mathbf{B}$ pruned of ℓ_1 blocks (that are possibly unconfirmed blocks).
- Parameter ℓ_2: number of last blocks taken into consideration in $\mathbf{B}'$.
- Stakeholders $\mathcal{M}$: set of public keys associated to the player that have added at least one block in the last ℓ_2 blocks of $\mathbf{B}'$. In [23], such public keys are crucially used for both encryption and signature.
- Chain quality parameters: ℓ_3, ℓ_4 used in the soundness proof.
- $\mathsf{params}:= (1^{\ell_1}, 1^{\ell_2}, 1^{\ell_3}, 1^{\ell_4})$.

GG-NIZK: The Proof. A proof π for theorem x is computed as follows. Let w be the witness s.t. $(x, w) \in \mathcal{R}$.

1. Secret share the witness w using a weighted secret sharing scheme, using as weights the stake of the public keys appearing in $\mathcal{M}$. Do the same with the zero-string.
   Namely, produce the following two sets[5]:

$$\{\mathsf{sh}_{1,i}\}_{i\in\mathcal{M}} = \mathsf{Share}(w, \{\mathsf{stake}_i\}_{i\in\mathcal{M}}, \beta \cdot \mathsf{stake}_{\mathsf{total}}, s_1)$$
$$\{\mathsf{sh}_{2,i}\}_{i\in\mathcal{M}} = \mathsf{Share}(0, \{\mathsf{stake}_i\}_{i\in\mathcal{M}}, \beta \cdot \mathsf{stake}_{\mathsf{total}}, s_2)$$

2. Encrypt each weighted share using the public key of the corresponding player. Namely for all i such that $\mathsf{PK}_i \in \mathcal{M}$, sample random strings $r_{1,i}, r_{2,i}$ and compute: $\mathsf{ctx}_{1,i} = \mathsf{Enc}(\mathsf{PK}_i, \mathsf{sh}_{1,i}, r_{1,i})$ $\quad \mathsf{ctx}_{2,i} = \mathsf{Enc}(\mathsf{PK}_i, \mathsf{sh}_{2,i}, r_{2,i})$.
3. Compute a non-interactive witness indistinguishable proof (NIWI) π_{niwi} for the theorem: (1) either the first set of ciphertexts are correct encryptions under the public keys in $\mathcal{M}$ of shares of the witness w or (2) (trapdoor witness) the second set of ciphertexts is a collection of correct encryptions under the public keys in $\mathcal{M}$ of shares of a valid fork of length $\ell_3 + \ell_4$.

[5] The role of s_1, s_2 and β is not relevant for our discussion and therefore they can be ignored.

Hence, a proof π for theorem $x \in L$ consists of the tuple:

$$\pi = (\mathbf{B}, \{\mathsf{ctx}_{1,i}, \mathsf{ctx}_{2,i}\}_{i \in \mathcal{M}}, \pi_{\mathsf{niwi}}, \mathsf{params})$$

Note that the proof π is not published on the blockchain and it is only sent to the verifier.

Security of GG-NIZK: Intuition. Zero knowledge follows from the assumption of honest majority of stake. Under such assumption, the simulator –controlling all honest players– is able to compute a valid fork that constitutes a valid trapdoor witness for the NIWI. Even if the trapdoor witness is encrypted in $(\mathsf{ctx}_{2,i})_{\mathsf{PK}_i \in \mathcal{M}}$, a malicious verifier cannot detect that the trapdoor witness was used, since it does not control enough secret keys (associated to the public keys in $\mathcal{M}$) that would allow for collection of enough shares.

Soundness is proved by witness extraction: the extractor controls a sufficient fraction of honest secret keys (associated to the public keys in $\mathcal{M}$) and this allows the decryption of enough ciphertexts, that leads to enough shares to reconstruct the witness.

Clearly by obtaining in the future (e.g., when those keys will correspond to a reduced amount of stake) the secrets of the involved stakeholders (through adaptive corruptions or by naturally receiving the keys from honest stakeholders) the adversary would be able to decrypt those ciphertexts therefore breaking the zero knowledge property and without violating the proof-of-stake assumption. This problem imposes the assumptions/limitations of the GG-NIZK discussed previously.

A Simple Smart Contract that Breaks the ZK Property of GG-NIZK. The zero-knowledge property of GG-NIZK crucially relies on the assumption that the malicious verifier – controlling only a minority of stake– does not have enough secret keys for the public keys in $\mathcal{M}$ to be able to decrypt enough ciphertexts and thus reconstruct the witness.

Our main observation is that in order to obtain decryptions of enough ciphertexts, a malicious verifier, does not necessarily need to *own* enough of the stake/secret keys of the honest players. Instead, the malicious verifier can upload a smart contract – that we called `DecryptionForBarbados`– where she promises a reward for a valid decryption of a certain ciphertext ctx under a certain public key PK. Notice that to run such smart contract the adversary does not need to corrupt the stakeholders, or get a stake transfer. So, the attack to works even if no-one is corrupted and even if no-stake is transferred. Obviously, when considering a blockchain with additional restrictions the our attack based on the above smart contract might not work, but still, the potential existence of other attacks should not be overlooked.

In more details, once the malicious verifier obtains $\pi = (\mathbf{B}, \{\mathsf{ctx}_{1,i}, \mathsf{ctx}_{2,i}\}_{i \in \mathcal{M}}, \pi_{\mathsf{niwi}}, \mathsf{params})$ from an honest prover she can publish a `DecryptionForBarbados` for some of (they could also be rerandomized if useful) the ciphertexts $\mathsf{ctx}_{1,i}$ for which she does not possess the secret key. The malicious verifier using `DecryptionForBarbados` is able to collect enough shares and reconstruct the

witness that is encrypted in $\{\mathsf{ctx}_{1,i}, \mathsf{ctx}_{2,i}\}_{i \in \mathcal{M}}$, thus directly invalidating the ZK property of [23].

In Fig. 1 we give a more detailed description of `DecryptionForBarbados`. In order to keep the smart contract simple we assume that the decryption procedure of the underlying encryption scheme gives in output a pair (m, r) where r is the randomness used to encrypt and m is the message encrypted (see for instance [8]). For the same reason, we also assume that (m, r) are unique (for a public key PK).

Notation (borrowed from [27]).
- Ledger: the blockchain.
- $\mathsf{Ledger}[\mathsf{Pt}_i]$ denotes the amount of money possessed by the secret key of party Pt_i.

`DecryptionForBarbados`

1. Init: Upon receiving (init, $\$\mathsf{reward}, \mathsf{ctx}, \mathsf{PK}_i$) from a contractor $\mathcal{C}$:
   - Assert $\mathsf{Ledger}[\mathcal{C}] > \$\mathsf{reward}$.
   - $\mathsf{Ledger}[\mathcal{C}] := \mathsf{Ledger}[\mathcal{C}] - \$\mathsf{reward}$.
   - Set state := INIT.
2. Claim: On input (claim, v) from a player Pt_i:
   - Parse $v = (m, r)$.
   - If $\mathsf{ctx} = \mathsf{Enc}_{\mathsf{PK}_i}(m, r)$ then set rewards $\mathsf{Ledger}[\mathsf{Pt}] := \mathsf{Ledger}[\mathsf{Pt}] + \$\mathsf{reward}$.
   - Set state := CLAIMED.

Fig. 1. Description of `DecryptionForBarbados`.

Observations on the Smart Contract. We note that a player that uses her secret key to trigger `DecryptionForBarbados` in order to win the reward is not violating any assumption of the underlying PoS protocol or of GG-NIZK. Indeed, he is not exposing his secret key but simply providing a valid decryption of a certain ciphertext. Thus this is legitimate behavior of a honest player, she is simply executing an other application that runs on top of the blockchain.

Our smart contract is not a "bribing attack". Bribing assumes that one is paying somebody to do something wrong/break the rules. Instead in this context an honest player is still behaving honestly and he is not breaking any rule of the underlying PoS protocol.

We also note that since the proof π is not published on the blockchain (and is not required to be), honest players could be not aware that they are helping a malicious verifier to break the security of π.

3 Definitions

Preliminary. We denote the security parameter by λ and use "||" as concatenation operator (i.e., if a and b are two strings then by $a||b$ we denote the

concatenation of a and b). We use the abbreviation PPT that stays for probabilistic polynomial time. We use $\mathsf{poly}(\cdot)$ to indicate a generic polynomial function. A *polynomial-time relation* $\mathcal{R}$ (or *polynomial relation*, in short) is a subset of $\{0,1\}^* \times \{0,1\}^*$ such that membership of (x,w) in $\mathcal{R}$ can be decided in time polynomial in $|x|$. For $(x,w) \in \mathcal{R}$, we call x the *instance* and w a *witness* for x. For a polynomial-time relation $\mathcal{R}$, we define the $\mathcal{NP}$-language $L_{\mathcal{R}}$ as $L_{\mathcal{R}} = \{x$ s.t. $\exists\ w : (x,w) \in \mathcal{R}\}$. Analogously, unless otherwise specified, for an $\mathcal{NP}$-language L we denote by $\mathcal{R}$ the corresponding polynomial-time relation (that is, $\mathcal{R}$ is such that $L = L_{\mathcal{R}}$). We will denote by $\mathcal{P}^{\mathsf{st}}$ a stateful algorithm $\mathcal{P}$ with state st. We will use the notation $r \in_R \{0,1\}^\lambda$ to indicate that r is sampled at random from $\{0,1\}^\lambda$. When we want to specify the randomness r used by an algorithm Al we use the following notation $\mathsf{Al}(\cdot;r)$.

3.1 Blockchain Protocols

In the next two sections we borrow the description of a blockchain protocol of [23,32], moreover we explicitly define the procedure executed by an honest player in order to add a block. A blockchain protocol Γ is parameterized by a validity predicate V that captures the semantics and rules of the blockchain. Γ consists of 4 polynomial-time algorithms ($\mathsf{UpdateState}$, $\mathsf{GetRecords}$, $\mathsf{Broadcast}$, $\mathsf{GenBlock}$) with the following syntax.

- $\mathsf{UpdateState}(1^\lambda, \mathsf{st})$: It takes as input the security parameter λ, state st and outputs the updated state st.
- $\mathsf{GetRecords}(1^\lambda, \mathsf{st})$: It takes as input the security parameter λ and state st. It outputs the longest ordered sequence of valid blocks $\mathbf{B}$ (or simply blockchain) contained in the state variable, where each block in the chain itself contains an unordered sequence of records messages.
- $\mathsf{Broadcast}(1^\lambda, m)$: It takes as input the security parameter λ and a message m, and broadcasts the message over the network to all nodes executing the blockchain protocol. It does not give any output.
- $\mathsf{GenBlock}(\mathsf{st}, \mathbf{B}, x)$: It takes as input a state st, a blockchain $\mathbf{B}$[6], $x \in \{0,1\}^*$ and outputs a candidate block B that contains a string v computed running a function f_{ID} that is defined as follows. The function $f_{\mathsf{ID}}(1^\lambda; r)$ takes as input the security parameter λ and running with $\mathsf{poly}(\lambda)$ bits of randomness r outputs a q bit string v, where $q = \mathsf{poly}(\lambda)$. Moreover every time that f_{ID} runs on input a freshly generated randomness it holds that $H_\infty(f_{\mathsf{ID}}(1^\lambda; \cdot)) \geq \lambda$[7]. The generated block B could satisfy or not the validity predicate V.
  We will denote by B_v a block B that contains the string v computed using f_{ID}.

[6] In order to simplify the notation we make an abuse of notation and we explicitly add the blockchain as input of $\mathsf{GenBlock}$ even though the blockchain can be computed running $\mathsf{GetRecords}$ on input st.

[7] In the existing blockchains the value v could be an identifier of a wallet and f_{ID} is the randomized function that generates it.

Blockchain Notation. With the notation $\mathbf{B} \leq \mathbf{B}'$ we will denote that the blockchain $\mathbf{B}$ is a prefix of the blockchain $\mathbf{B}'$. We denote by $\mathbf{B}^{\lceil n}$ the chain resulting from "pruning" the last n blocks in $\mathbf{B}$. We will denote by Γ^{V} a blockchain protocol Γ that has validate predicate V. A blockchain $\mathbf{B}$ generated by the execution of Γ^{V} is the blockchain obtained by an honest player after calling GetRecords during an execution of Γ^{V}. An *honest execution of* GenBlock is an execution of GenBlock computed by an honest player. A blockchain protocol Γ can satisfy the property of chain-consistency, chain-growth and chain-quality defined in previous works [19,32]. In the rest of the paper we will denote by $\eta(\cdot)$ the chain consistency parameter of Γ^{V}.

Definition 1 (Block Trim Function). *Let B_v be a block generated using* GenBlock *that satisfies the validate predicate* V*. We define a block trim function as a deterministic function* `trim` *that on input B_v outputs v.*

Note that for two blocks B, B' that satisfy V and are generated by an honest execution of GenBlock it could happen that $\texttt{trim}(B) = \texttt{trim}(B')$. For instance this is the case when a honest player Pt runs GenBlock twice and both executions run f_{ID} on input the same randomness stored in the state of Pt.

Definition 2 (Good Execution of GenBlock). *Let $\overline{\mathbf{B}}$ be a blockchain generated by an execution of Γ^{V}. An execution of* GenBlock *is good w.r.t. a blockchain $\mathbf{B}$ if it holds that* GenBlock *runs on input $\overline{\mathbf{B}}$ s.t. $\mathbf{B} \leq \overline{\mathbf{B}}^{\lceil \eta(\lambda)}$, moreover* GenBlock *runs f_{ID} on input fresh randomness and outputs a block that satisfies the validity predicate* V*.*

Definition 3 (Pristine Block). *Let* `trim` *be the block trim function defined in Definition 1. Let $\mathbf{B}$ be a blockchain composed of k blocks generated by an execution of Γ^{V}. The j-th block B_j of $\mathbf{B}$ is* pristine *if for each B_i of $\mathbf{B}$ with $0 < i < j$ it holds that $v \neq v_i$ where $v = \texttt{trim}(B_j)$ and $v_i = \texttt{trim}(B_i)$.*

Assumption 1. *Let $\mathbf{B}$ be a blockchain generated during an execution of Γ^{V}. There exists $d = \mathsf{poly}(\lambda)$ and $u = \mathsf{poly}(\lambda)$ such that for any sequence of d consecutive blocks $B_{i+1}, \ldots, B_{i+d}$ added to $\mathbf{B}$ during the execution of Γ^{V}, let n be the number of pristine blocks in $B_{i+1}, \ldots, B_{i+d}$, it holds that:*

1. *At least $\lfloor n/2 + 1 \rfloor$ of the pristine blocks in the sequence $B_{i+1}, \ldots, B_{i+d}$ have been generated by honest players through good executions of* GenBlock *w.r.t. $\mathbf{B}$;*
2. *For each $j \in \{1, \ldots d\}$, the probability that honest players obtain through honest executions of* GenBlock *w.r.t. $\mathbf{B}$ $u' > u$ different blocks satisfying the validity predicate for the position $i + j$ in the blockchain is negligible in λ.*

We refer to d as the pristine parameter *and to u as the* attempts parameter*.*

Notice that n is a non-constant value that depends on the content of the specific d consecutive blocks taken into account. For the sake of simplifying the description of our construction we will assume wlog that n is also a pristine parameter.

On the Applicability of Our Assumption. It is well known that blockchains need an incentive mechanism and this is typically implemented by assigning a reward each time a block is added to the chain. This process is often implemented as a lottery and some coins are generated and assigned to the winner of the lottery that is also the player that generated the new block added to the chain. In order to get the coin assigned, the winner also includes an identifier of her wallet to the block. Such identifiers usually correspond to public keys of signature schemes and as such they have a significant amount of min-entropy. Therefore, whenever such identifier is selected by a honest blockchain player and has never circulated in the network, it represents an unpredictable string. More concretely one could think in the case of Bitcoin to the coinbase transaction, since sometimes the rewards is cashed on a new wallet.

Our blockchain assumption assumes that given a sufficiently long sequence of blocks, if we restrict to identifiers that appear for the first time on the chain, then a majority of them was unpredictable before the long sequence of blocks started. Obviously an adversary can sometimes be the winner and therefore can use an identifier that is "fresh" in the eyes of others but that she knew already before the long sequence of blocks started. Therefore our assumption requires the adversary to have limited resources so that she places in the chain less blocks than what honest players using fresh identifiers do.

For concreteness, one can consider the current modus operandi of Bitcoin blockchain. To avoid double spending it is in general recommended to wait for 6 more blocks after the block including the spending transaction, this is called confirmation time. The choice of 6 blocks for a confirmation time suggests that it is believed that it would be very unlikely that the adversary could have produced in the meanwhile 7 blocks that cancel the spending transaction. If for instance we quantify "very unlikely" with something less than 2^{-70} then as a consequence the adversary must have probability of being the winner (therefore deciding the next block) less than 2^{-10}. Following this example, if an honest block includes a "fresh" wallet with probability at least 2^{-9} (which is very reasonable), then our assumption clearly holds for a sufficiently large sequence of blocks (i.e., considering a sufficiently large d).

We have considered Bitcoin and the 6-block confirmation rule just because it is the most popular example of blockchain and thus it is a natural target to check the concreteness of our assumption. Indeed, also coinbase transaction is just an example of a field with min-entropy that could be find in the blockchain (see also the examples mentioned in [13]). One could consider, for instance, privacy-preserving blockchains (e.g., [18,28] for the case of PoS blockchains), observing that the cryptographic material used to ensure privacy might imply the presence of fields with high min-entropy in a block.

Our construction is a mere feasibility result aiming at showing that publicly verifiable zero knowledge is possible with generic[8] blockchains.

[8] In the sense of the underlying consensus mechanism.

3.2 Execution of Γ^{V} in an Environment

At a very high level, the execution of the protocol Γ^{V} proceeds in rounds that model time steps. Each participant in the protocol runs the UpdateState algorithm to keep track of the current (latest) blockchain state. This corresponds to listening on the broadcast network for messages from other nodes. The GetRecords algorithm is used to extract an ordered sequence of blocks encoded in the blockchain state variable. The Broadcast algorithm is used by a player when she wants to post a new message m on the blockchain. Note that the message m is accepted by the blockchain protocol only if it satisfies the validity predicate V given the current state, (i.e., the current sequence of blocks).

Following prior works [19,29,32], we define the protocol execution following the activation model of the Universal Composability framework of [15] (though like [23] we will not prove UC-security of our results). For any blockchain protocol $\Gamma^{\mathsf{V}}(\mathsf{UpdateState}, \mathsf{GetRecords}, \mathsf{Broadcast}, \mathsf{GenBlock})$, the protocol execution is directed by the environment $\mathcal{Z}(1^\lambda)$. The environment $\mathcal{Z}$ activates the players as either honest or corrupt and is also responsible for providing inputs/records to all players in each round.

All the corrupt players are controlled by the adversary $\mathcal{A}$ that can corrupt players adaptively during the execution of Γ^{V}.

Specifically $\mathcal{A}$ can send a corruption request $\langle \mathtt{corr}, \mathsf{Pt}_\mathtt{i} \rangle$ to player Pt_i at any point during the execution of Γ^{V}. The adversary is also responsible for the delivery of all network messages. Honest players start by executing UpdateState on input 1^λ with an empty state $\mathsf{st} = \epsilon$.

- In round r, each honest player Pt_i potentially receives a message(s) m from $\mathcal{Z}$ and potentially receives incoming network messages (delivered by $\mathcal{A}$). It may then perform any computation, broadcast a message (using Broadcast algorithm) to all other players (which will be delivered by the adversary; see below) and update its state st_i. It could also attempt to "add" a new block to its chain: Pt_i will run the procedure GenBlock, and this execution of GenBlock could use fresh randomness for the function $f_{\mathsf{ID}}(1^\lambda; \cdot)$ if requested by $\mathcal{Z}$.
- $\mathcal{A}$ is responsible for delivering all messages sent by players (honest or corrupted) to all other players. $\mathcal{A}$ cannot modify the content of messages broadcast by honest players, but it may delay or reorder the delivery of a message as long as it eventually delivers all messages within a certain time limit.
- At any point $\mathcal{Z}$ can communicate with adversary $\mathcal{A}$.

Constraints on the Adversary. In order to show that a blockchain enjoys some useful properties (e.g., chain consistency) prior works [19,32] restrict their analysis to compliant executions of Γ^{V} where some specific restrictions[9] are imposed to $\mathcal{Z}$ and $\mathcal{A}$. Those works showed that certain desirable security properties are respected except with negligible probability in any compliant execution. Obviously, when in our work we claim results assuming some properties of the blockchain, we are taking into account compliant executions of the underlying blockchain protocol only. The same is done by [23].

[9] For instance, they require that any broadcasted message is delivered in a maximum number of time steps.

3.3 Publicly Verifiable ZK Proof System from BlockchainS

Here we define delayed-input publicly verifiable zero knowledge w.r.t. blockchain failure over a blockchain protocol $\Gamma^{\mathsf{V}} = (\mathsf{UpdateState}, \mathsf{GetRecords}, \mathsf{Broadcast}, \mathsf{GenBlock})$. We will make use of the following notation.

The view of a blockchain player Pt consists of the messages received during an execution of Γ^{V}, along with its randomness and its inputs. Let $\mathsf{Exec}^{\Gamma^{\mathsf{V}}}(\mathcal{A}, \mathcal{H}, \mathcal{Z}, 1^\lambda)$ be the random variable denoting the joint view of all players in the execution Γ^{V}, with adversary $\mathcal{A}$ and set of honest players $\mathcal{H}$ in environment $\mathcal{Z}$, such a joint view fully determines the execution. Let $\Gamma^{\mathsf{V}}_{\mathsf{view}}(\mathcal{A}, \mathcal{H}, \mathcal{Z}, 1^\lambda)$ denote an execution of $\Gamma^{\mathsf{V}}(\mathcal{A}, \mathcal{H}, \mathcal{Z}, 1^\lambda)$ producing view as joint view.

Definition 4 (Publicly Verifiable Proof System from Blockchain). *A pair of stateful* PPT *algorithms $\Pi = (\mathcal{P}, \mathcal{V})$ over a blockchain protocol Γ^{V} is a publicly verifiable proof system for the $\mathcal{NP}$-language $\mathcal{L}$ with witness relation $\mathcal{R}$ if it satisfies the following properties:*

Completeness. *$\forall\ x, w$ s.t. $(x, w) \in \mathcal{R}$, $\forall$ PPT adversary $\mathcal{A}$ any* PPT *$\mathsf{Pt}_j \in \mathcal{H}$ where $\mathcal{H}$ is the set of honest parties, and for any environment $\mathcal{Z}$, assuming that $\mathcal{P} \in \mathcal{H}$, there exist negligible functions $\nu_1(\cdot)$, $\nu_2(\cdot)$ such that:*

$$\Pr\left[\mathcal{V}(x, \pi, \mathbf{B}) = 1 \;\vdots\; \begin{array}{l} \mathsf{view} \leftarrow \mathsf{Exec}^{\Gamma^{\mathsf{V}}}(\mathcal{A}, \mathcal{H}, \mathcal{Z}, 1^\lambda) \\ \pi \leftarrow \mathcal{P}^{\mathsf{st}_{\mathcal{P}}}(x, w) \\ \mathbf{B} = \mathsf{GetRecords}(1^\lambda, \mathsf{st}_j) \end{array}\right] \geq 1 - \nu_1(|x|) - \nu_2(\lambda)$$

where $\mathsf{st}_{\mathcal{P}}$ denotes the state of $\mathcal{P}$ during the execution $\Gamma^{\mathsf{V}}_{\mathsf{view}}(\mathcal{A}, \mathcal{H}, \mathcal{Z}, 1^\lambda)$. The running time of $\mathcal{P}$ is polynomial in the size of the blockchain $\mathbf{B} = \mathsf{GetRecords}(1^\lambda, \mathsf{st}_j)$ where st_j is the state of Pt_j at the end of the execution $\Gamma^{\mathsf{V}}_{\mathsf{view}}(\mathcal{A}, \mathcal{H}, \mathcal{Z}, 1^\lambda)$.[10]

Soundness. *$\forall\ x \notin \mathcal{L}$, $\forall$ stateful adversary $\mathcal{A}$ and* PPT *honest player $\mathsf{Pt}_j \in \mathcal{H}$ where $\mathcal{H}$ is the set of honest players and for any environment $\mathcal{Z}$, there exist negligible functions $\nu_1(\cdot)$, $\nu_2(\cdot)$ such that:*

$$\Pr\left[\mathcal{V}(x, \pi, \mathbf{B}) = 1 \;\vdots\; \begin{array}{l} \mathsf{view} \leftarrow \mathsf{Exec}^{\Gamma^{\mathsf{V}}}(\mathcal{A}, \mathcal{H}, \mathcal{Z}, 1^\lambda) \\ \pi, x \leftarrow \mathcal{A}^{\mathsf{st}_{\mathcal{A}}} \\ \mathbf{B} = \mathsf{GetRecords}(1^\lambda, \mathsf{st}_j) \end{array}\right] \leq \nu_1(|x|) + \nu_2(\lambda)$$

where $\mathsf{st}_{\mathcal{A}}$ denotes the state of $\mathcal{A}$ during the execution $\Gamma^{\mathsf{V}}_{\mathsf{view}}(\mathcal{A}, \mathcal{H}, \mathcal{Z}, 1^\lambda)$. Furthermore st_j is the state of Pt_j at the end of the execution $\Gamma^{\mathsf{V}}{}_{\mathsf{view}}(\mathcal{A}, \mathcal{H}, \mathcal{Z}, 1^\lambda)$. The proof π might consist of multiple messages, i.e., $\pi = (\pi^1, \ldots, \pi^m)$, in this case, we will say that Π is an m-messages proof system. Moreover if π is composed of m-messages $\pi = (\pi^1, \ldots, \pi^m)$, $\mathcal{A}$ is allowed to choose x just before computing the last message π^m of the proof $\pi = (\pi^1, \ldots, \pi^m)$.

[10] Note that the execution of $\Gamma^{\mathsf{V}}_{\mathsf{view}}(\mathcal{A}, \mathcal{H}, \mathcal{Z}, 1^\lambda)$ could continue even after π is provided by $\mathcal{P}$.

Definition 5 (Delayed-Input Completeness from Blockchain). *An m-messages proof system Π over a blockchain protocol Γ^{V} is* **delayed-input**, *if x, w are not involved before the computation of the last message π^m of the proof $\pi = (\pi^1, \ldots, \pi^m)$.*

Definition 6 (Witness Indistinguishability w.r.t. Blockchain Failure). *A publicly verifiable proof system $\Pi = (\mathcal{P}, \mathcal{V})$ over a blockchain protocol Γ^{V} for the $\mathcal{NP}$-language $\mathcal{L}$ with witness relation $\mathcal{R}$ is witness indistinguishable (WI) w.r.t. blockchain failure if it satisfies the following property:*

$\forall x, w_0, w_1$ such that $(x, w_0) \in \mathcal{R}$ and $(x, w_1) \in \mathcal{R}$, $\forall$ PPT adversary $\mathcal{A}$ and set of PPT *honest players $\mathcal{H}$ and any* PPT *environment $\mathcal{Z}$, where $\mathcal{P} \in \mathcal{H}$ it holds that:*

$$\left\{\mathsf{view}_{\mathcal{A}} : \mathsf{view}_{\mathcal{A}} \leftarrow \mathtt{Exp}^{0}{}_{\mathcal{A},\Pi,\Gamma^{\mathsf{V}}}(\lambda)\right\} \approx \left\{\mathsf{view}_{\mathcal{A}} : \mathsf{view}_{\mathcal{A}} \leftarrow \mathtt{Exp}^{1}{}_{\mathcal{A},\Pi,\Gamma^{\mathsf{V}}}(\lambda)\right\}$$

where $\mathtt{Exp}^{b}{}_{\mathcal{A},\Pi,\Gamma^{\mathsf{V}}}(\lambda)$ is defined below, for $b \in \{0,1\}$.

$\mathtt{Exp}^{b}{}_{\mathcal{A},\Pi,\Gamma^{\mathsf{V}}}(\lambda, x, w_b)$:
- $\mathcal{P}$ *runs on input* 1^λ.
- *An execution of* $\Gamma^{\mathsf{V}}(\mathcal{A}, \mathcal{Z}, \mathcal{H}, 1^\lambda)$ *starts.*
- $\mathcal{P}^{\mathsf{st}_{\mathcal{P}}}$ *outputs messages* $\pi^1, \ldots, \pi^{m-1}$, *where* $\mathsf{st}_{\mathcal{P}}$ *is the state of* $\mathcal{P}$ *in the execution* $\Gamma^{\mathsf{V}}(\mathcal{A}, \mathcal{Z}, \mathcal{H}, 1^\lambda)$.
- *Upon receiving* $(x, w_0) \in \mathcal{R}$, $(x, w_1) \in \mathcal{R}$ *from* $\mathcal{A}$.
- $\mathcal{P}^{\mathsf{st}_{\mathcal{P}}}$ *computes* π^m *on input* π^{m-1}, x, w_b *and outputs* $\pi = (\pi^1, \ldots, \pi^m)$.
- $\mathcal{A}$ *can send a collapse request* $\langle \mathtt{corr}, \mathtt{all} \rangle$ *obtaining:*
The state st_i *from the honest player* $\mathsf{Pt}_i \in \mathcal{H}$, *for* $i = 1, \ldots, \lvert\mathcal{H}\rvert$;
The state $\mathsf{st}_{\mathcal{P}}$ *from* $\mathcal{P}$.
-*The execution of* $\Gamma^{\mathsf{V}}(\mathcal{A}, \mathcal{Z}, \mathcal{H}, 1^\lambda)$ *terminates and* $\mathcal{A}$ *outputs her view* $\mathsf{view}_{\mathcal{A}}$ *and this is the output of the experiment.*

Remark 1. The above definition does not assume that the blockchain satisfies the predicate V, even when $\mathcal{P}$ is the only honest player of Γ^{V}, and thus the blockchain could be completely controlled by the adversary. In this scenario we will say that $\Pi = (\mathcal{P}, \mathcal{V})$ enjoys WI w.r.t. blockchain failure over any blockchain protocol.

Definition 7 (Zero Knowledge w.r.t. Blockchain Failure). *A publicly verifiable proof system $\Pi = (\mathcal{P}, \mathcal{V})$ over a blockchain protocol Γ^{V} for the $\mathcal{NP}$-language $\mathcal{L}$ with witness relation $\mathcal{R}$ is Zero Knowledge (ZK) w.r.t. blockchain failure if there is a stateful* PPT *algorithm S such that $\forall$ x, w s.t. $(x, w) \in \mathcal{R}$, $\forall$* PPT *adversary $\mathcal{A}$ and set of* PPT *honest players $\mathcal{H}$ and for any* PPT *environment $\mathcal{Z}$, where $\mathcal{P} \in \mathcal{H}$ it holds that:*

$$\left\{\mathsf{view}_{\mathcal{A}} : \mathsf{view}_{\mathcal{A}} \leftarrow \mathtt{Exp}^{0}{}_{\mathcal{A},\Pi,\Gamma^{\mathsf{V}}}(\lambda)\right\} \approx \left\{\mathsf{view}_{\mathcal{A}} : \mathsf{view}_{\mathcal{A}} \leftarrow \mathtt{Exp}^{1}{}_{\mathcal{A},\Pi,\mathsf{S},\Gamma^{\mathsf{V}}}(\lambda)\right\}$$

where $\mathtt{Exp}^0_{\mathcal{A},\Pi,\Gamma^{\mathsf{V}}}(\lambda)$ *and* $\mathtt{Exp}^1_{\mathcal{A},\Pi,\mathsf{S},\Gamma^{\mathsf{V}}}(\lambda)$ *are defined below.*

$\mathtt{Exp}^0_{\mathcal{A},\Pi,\Gamma^{\mathsf{V}}}(\lambda)$*:*

- $\mathcal{P}$ *runs on input* 1^λ.
- *An execution of* $\Gamma^{\mathsf{V}}(\mathcal{A},\mathcal{H},\mathcal{Z},1^\lambda)$ *starts.*

1. *At any point* $\mathcal{A}$ *can send a corruption request* $\langle \mathtt{ZK}_{\mathtt{corr}}(x,w)\rangle$ *(s.t.* $(x,w)\in\mathcal{R}$*) to* $\mathcal{P}$ *obtaining from* $\mathcal{P}$ *her state* $\mathsf{st}_{\mathcal{P}}$.
2. $\mathcal{P}^{\mathsf{st}_{\mathcal{P}}}$ *outputs messages* $\pi^1,\ldots,\pi^{m-1}$.
3. *If* $\mathcal{A}$ *did not compute Step 1* $\mathcal{P}$ *receives* $(x,w')\in\mathcal{R}$ *from* $\mathcal{A}$.
4. $\mathcal{P}^{\mathsf{st}_{\mathcal{P}}}$ *outputs* $\pi=(\pi^1,\ldots,\pi^m)$.
5. *If* $\mathcal{A}$ *sends a collapse request* $\langle \mathtt{corr},\mathtt{all}\rangle$ *obtains:*
   The state st_i *from honest player* $\mathsf{Pt}_i\in\mathcal{H}$*, for* $i=1,\ldots,|\mathcal{H}|$*;*
   The state $\mathsf{st}_{\mathcal{P}}$ *from* $\mathcal{P}$*, if* $\mathcal{A}$ *did not compute Step 1.*

-The execution of $\Gamma^{\mathsf{V}}(\mathcal{A},\mathcal{Z},\mathcal{H},1^\lambda)$ *terminates and* $\mathcal{A}$ *outputs her view* $\mathsf{view}_{\mathcal{A}}$ *and this is the output of the experiment.*

$\mathtt{Exp}^1_{\mathcal{A},\Pi,\mathsf{S},\Gamma^{\mathsf{V}}}(\lambda)$*:*

- S *runs on input* 1^λ.
- *An execution of* $\Gamma^{\mathsf{V}}(\mathcal{A},\mathsf{S},\mathcal{Z},1^\lambda)$ *starts.*

1. *At any point* $\mathcal{A}$ *can send a corruption request* $\langle \mathtt{ZK}_{\mathtt{corr}}(x,w)\rangle$ *(s.t.* $(x,w)\in\mathcal{R}$*) to* S *obtaining from* S *a state* $\mathsf{st}_{\mathcal{P}}$.
2. S *outputs messages* $\pi^1,\ldots,\pi^{m-1}$.
3. *If* $\mathcal{A}$ *did not compute Step 1:* S *receives* $(x,w')\in\mathcal{R}$ *from* $\mathcal{A}$*,* S *ignores* w'.
4. S *outputs* $\pi=(\pi^1,\ldots,\pi^m)$.
5. *If* $\mathcal{A}$ *sends a collapse request* $\langle \mathtt{corr},\mathtt{all}\rangle$ *obtains from* S*:*
   The state st_i *for each honest player* $\mathsf{Pt}_i\in\mathcal{H}$*, for* $i=1,\ldots,|\mathcal{H}|$*;*
   The state $\mathsf{st}_{\mathcal{P}}$ *for the honest prover of* Π*, if* $\mathcal{A}$ *did not compute Step 1.*

-The execution of $\Gamma^{\mathsf{V}}(\mathcal{A},\mathcal{Z},\mathcal{H},1^\lambda)$ *terminates and* $\mathcal{A}$ *outputs her view* $\mathsf{view}_{\mathcal{A}}$ *and this is the output of the experiment.*

4 Publicly Verifiable ZK w.r.t. Blockchain Failure

We construct a delayed-input publicly verifiable zero-knowledge proof system w.r.t. blockchain failure $\Pi_{\mathsf{pvZK}}=(\mathcal{P}_{\mathsf{pvZK}},\mathcal{V}_{\mathsf{pvZK}})$ over any blockchain protocol $\Gamma^{\mathsf{V}}=(\mathsf{UpdateState},\ \mathsf{GetRecords},\ \mathsf{Broadcast},\mathsf{GenBlock})$ satisfying chain-consistency property, chain-growth property and Assumption 1. The parameters of Π_{pvZK} are reported in Table 1. We assume wlog that in a sequence of d blocks, n of them are pristine, where n is an even non-negative integer. Π_{pvZK} for the $\mathcal{NP}$-language $\mathcal{L}$ makes use of the following tools:

– The block trim function $\mathtt{trim}$ defined in Definition 1, that on input a block B outputs a q-bits long string v.

– A non-interactive statistically binding commitment scheme $\Pi_{\mathsf{Com}} = (\mathsf{Com}, \mathsf{VrfyOpen})$.
– A delayed-input publicly verifiable proof system $\Pi_{\mathsf{pvWI}} = (\mathcal{P}_{\mathsf{pvWI}}, \mathcal{V}_{\mathsf{pvWI}})$ over any blockchain protocol $\Gamma^{\mathsf{V}} = (\mathsf{UpdateState}, \mathsf{GetRecords}, \mathsf{Broadcast}, \mathsf{GenBlock})$ that satisfies chain-consistency property, chain-growth property and Assumption 1. Moreover Π_{pvWI} enjoys WI w.r.t. blockchain failure over any blockchain protocol. Π_{pvWI} is for $\mathcal{NP}$-language $\mathcal{L}_{\mathsf{pvWI}}$ which is associated to the relation $\mathcal{R}_{\mathsf{pvWI}} = \left\{((x, x_{\mathtt{com}}), w) : (x, w) \in \mathcal{R} \ \vee \ (x_{\mathtt{com}}, w) \in \mathcal{R}_{\mathtt{com}} \right\}$, where $\mathcal{R}$ is the relation associated to the $\mathcal{NP}$-language $\mathcal{L}$ and $\mathcal{R}_{\mathtt{com}}$ is the relation associated to the following $\mathcal{NP}$-language:

$$\mathcal{L}_{\mathtt{com}} = \Big\{ \{\mathtt{com}_j\}_{j=1}^{u \cdot d}, \{v_i\}_{i=1}^{n} : \exists\, 1 \le j_1 < \cdots < j_{n/2+1} \le n,\ \{\mathtt{open}_{j_k}\}_{k=1}^{n/2+1}$$
$$\text{s.t. } \mathsf{VrfyOpen}(\mathtt{com}_{j_k}, \mathtt{open}_{j_k}, v_{j_k}) = 1\ \forall k = 1, \ldots, n/2+1 \Big\}$$

Loosely speaking the relation $\mathcal{R}_{\mathtt{com}}$ is satisfied if the message committed in $\mathtt{com}_{j_k}$ is v_{j_k} for at least $n/2+1$ distinct values of j_k. The instance length of $\mathcal{L}_{\mathsf{pvWI}}$ is ℓ and the size of the proof generated by $\mathcal{P}_{\mathsf{pvWI}}$ is of m messages.

Our delayed-input publicly verifiable zero-knowledge proof system w.r.t. blockchain failure $\Pi_{\mathsf{pvZK}} = (\mathcal{P}_{\mathsf{pvZK}}, \mathcal{V}_{\mathsf{pvZK}})$ is described in Fig. 2.

Table 1. Parameters of Π_{pvZK}.

Table of notation	
ℓ	Size of the theorem for $\mathcal{L}_{\mathsf{pvWI}}$
m	Number of messages of Π_{pvWI}
q	Output-length of the block trim function $\mathtt{trim}$. See Definition 1
η	Chain consistency parameter of Γ^{V}
d, n	Pristine parameters of Γ^{V}. See Assumption 1
u	Attempts parameter of Γ^{V}. See Assumption 1

Theorem 1. *Let $\Gamma^{\mathsf{V}} = (\mathsf{UpdateState}, \mathsf{GetRecords}, \mathsf{Broadcast}, \mathsf{GenBlock})$ be any blockchain protocol that satisfies chain-consistency property, chain-growth property and Assumption 1. Let $\Pi_{\mathsf{Com}} = (\mathsf{Com}, \mathsf{VrfyOpen})$ be a non-interactive statistically binding commitment scheme. Let $\Pi_{\mathsf{pvWI}} = (\mathcal{P}_{\mathsf{pvWI}}, \mathcal{V}_{\mathsf{pvWI}})$ be a delayed-input publicly verifiable proof system over Γ^{V} for $\mathcal{NP}$-language $\mathcal{L}_{\mathsf{pvWI}}$. Moreover Π_{pvWI} enjoys WI w.r.t. blockchain failure over any blockchain protocol. Assuming secure erasure, $\Pi_{\mathsf{pvZK}} = (\mathcal{P}_{\mathsf{pvZK}}, \mathcal{V}_{\mathsf{pvZK}})$ (described in Fig. 2) is a delayed-input publicly verifiable zero-knowledge proof system w.r.t. blockchain failure over Γ^{V} for $\mathcal{NP}$.*

We note that a pvWI proof system that satisfies delayed-input completeness can be instantiated from OWPs using the work of [34]. In the full version [35] we

Publicly Verifiable ZK proof system w.r.t. blockchain failure $\Pi_{\mathsf{pvZK}} = (\mathcal{P}_{\mathsf{pvZK}}, \mathcal{V}_{\mathsf{pvZK}})$
Parameters are defined in Table 4.

Prover Procedure: $\mathcal{P}_{\mathsf{pvZK}}$. Input: instance x, witness w s.t. $(x, w) \in \mathcal{R}$.

— **First step.**

1. Compute $(\mathtt{com}_j, \mathtt{open}_j) \leftarrow \mathsf{Com}(0^q)$ and erase $\mathtt{open}_j$ for $j = 1, \ldots, d \cdot u$.

— **Blockchain Interaction.**

2. Set $\mathsf{st} = \epsilon$. Post $\mathtt{com}_1, \ldots, \mathtt{com}_{u \cdot d}$ on the blockchain by running $\mathsf{Broadcast}(1^\lambda, \mathtt{com}_1, \ldots, \mathtt{com}_{u \cdot d})$ and then monitor the blockchain by running $\mathsf{st} = \mathsf{UpdateState}(1^\lambda, \mathsf{st})$, $\mathbf{B} = \mathsf{GetRecords}(1^\lambda, \mathsf{st})$, until $\mathtt{com}_1, \ldots, \mathtt{com}_{u \cdot d}$ followed by d additional blocks $B_1, \ldots, B_d$ are posted on the blockchain $\mathbf{B}^{\lceil \eta}$ (i.e., we consider the blockchain $\mathbf{B}$ pruned of the last η blocks). Let $\overline{B_1}, \ldots, \overline{B_n}$ be the n pristine blocks in the sequence $B_1, \ldots, B_d$.

— **Second step.**

3. Compute $v_j = \mathtt{trim}(\overline{B_j})$ for $j = 1, \ldots, n$ and set $\mathtt{com} = \{\mathtt{com}_j\}_{j=1}^{u \cdot d}$, $\mathtt{val} = \{v_j\}_{j=1}^{n}$, $x_{\mathtt{com}} = (\mathtt{com}, \mathtt{val})$, $x_{\mathsf{pvWI}} = (x, x_{\mathtt{com}})$.
4. Obtain π^1_{pvWI} with randomness r_1 executing $\mathcal{P}_{\mathsf{pvWI}}$ on input $1^\lambda, \ell$ and interacting with the blockchain if it is required by $\mathcal{P}_{\mathsf{pvWI}}$.
5. For $i = 2, \ldots, m-1$:
   Obtain π^i_{pvWI} with randomness r_i executing $\mathcal{P}_{\mathsf{pvWI}}$ on input $\pi^{i-1}_{\mathsf{pvWI}}$ and interacting with the blockchain if it is required by $\mathcal{P}_{\mathsf{pvWI}}$.
6. Obtain π^m_{pvWI} executing $\mathcal{P}_{\mathsf{pvWI}}$ on input $\pi^{m-1}_{\mathsf{pvWI}}, x_{\mathsf{pvWI}}, w$ and interacting with the blockchain if it is required by $\mathcal{P}_{\mathsf{pvWI}}$.
7. Set $\pi_{\mathsf{pvWI}} = (\pi^1_{\mathsf{pvWI}}, \ldots, \pi^m_{\mathsf{pvWI}})$ and $\pi = (x_{\mathsf{pvWI}}, \{\mathtt{com}_j\}_{j=1}^{u \cdot d}, \pi_{\mathsf{pvWI}})$ erase any randomness that $\mathcal{P}_{\mathsf{pvWI}}$ requests to erase and output π.

Verifier Procedure: $\mathcal{V}_{\mathsf{pvZK}}$. Input: x, $\pi = (x_{\mathsf{pvWI}}, \{\mathtt{com}_j\}_{j=1}^{u \cdot d}, \pi_{\mathsf{pvWI}})$, and a blockchain $\tilde{\mathbf{B}}$ s.t. $\mathbf{B}^{\lceil \eta} \leq \tilde{\mathbf{B}}$ works as follows.

— **Check Blockchain.** If the messages $\{\mathtt{com}_j\}_{j=1}^{u \cdot d}$ are not posted on the blockchain $\tilde{\mathbf{B}}^{\lceil \eta}$ then $\mathcal{V}_{\mathsf{pvZK}}$ outputs 0. Otherwise, let B^* be the block of the blockchain $\tilde{\mathbf{B}}^{\lceil \eta}$ where the messages $\{\mathtt{com}_j\}_{j=1}^{u \cdot d}$ are posted. Let $\overline{B_1}, \ldots, \overline{B_n}$ be the n pristine blocks of the blockchain $\tilde{\mathbf{B}}^{\lceil \eta}$ after B^*. $\mathcal{V}_{\mathsf{pvZK}}$ computes $v'_j = \mathtt{trim}(\overline{B_j})$ for $j = 1, \ldots, n$ and parses x_{pvWI} as instance x, commitments $\{\mathtt{com}_j\}_{j=1}^{u \cdot d}$, and strings $\{v_j\}_{j=1}^{n}$.

— **Check Proof.** Accept if all the following conditions are satisfied.

- $v'_j = v_j$ for all $j \in \{1, \ldots, n\}$;
- $\mathcal{V}_{\mathsf{pvWI}}(x_{\mathsf{pvWI}}, \pi_{\mathsf{pvWI}}, \tilde{\mathbf{B}}) = 1$.

Execution of Γ^{V} by honest player Pt_j:

Pt_j acts as described in Section 3.2, in particular, upon receiving a request of an execution of $\mathsf{GenBlock}$ using fresh randomness for the function $f_{\mathsf{ID}}(1^\lambda; \cdot)$ by $\mathcal{Z}$:

Pt_j picks r at random from $\{0,1\}^{\mathsf{poly}(\lambda)}$;
Pt_j runs $\mathsf{GenBlock}$ and uses the randomness r to execute f_{ID}.

If $\mathcal{A}$ sends a collapse request $\langle \mathtt{corr}, \mathtt{all} \rangle$, $\mathcal{A}$ obtains $\mathsf{st}_{\mathsf{Pt}_i}$ from honest player Pt_i, for all $i = 1, \ldots, |\mathcal{H}|$, moreover $\mathcal{A}$ obtains the state $\mathsf{st}_{\mathcal{P}_{\mathsf{pvZK}}}$ of $\mathcal{P}_{\mathsf{pvZK}}$ (if $\mathcal{A}$ did not send a corruption request to $\mathcal{P}_{\mathsf{pvZK}}$ before).

Fig. 2. Description of $\Pi_{\mathsf{pvZK}} = (\mathcal{P}_{\mathsf{pvZK}}, \mathcal{V}_{\mathsf{pvZK}})$.

prove that Π_{pvWI} satisfies Definitions 4 and 6. Therefore we have the following corollary.

Corollary 1. *Let* $\Gamma^{\mathsf{V}} = (\mathsf{UpdateState}, \mathsf{GetRecords}, \mathsf{Broadcast}, \mathsf{GenBlock})$ *be a blockchain protocol that satisfies chain-consistency property, chain-growth property and Assumption 1. Assuming secure erasure, if one-way permutations exists, then* $\Pi_{\mathsf{pvZK}} = (\mathcal{P}_{\mathsf{pvZK}}, \mathcal{V}_{\mathsf{pvZK}})$ *is a delayed-input publicly verifiable zero-knowledge proof system w.r.t. blockchain failure over* Γ^{V} *for* $\mathcal{NP}$.

The proof of the Theorem 1 and the description of the simulator $\mathsf{S}_{\mathsf{pvZK}}$ for Π_{pvZK} can be found in the next subsections.

Note that the inputs of Π_{pvZK} (i.e., the statement x and the witness w) are used only in the last message of the protocol. Therefore the prover can *pre-process* all the other messages ahead of time (even without knowing the statement) and complete the last message whenever the statement becomes available.

4.1 Delayed-Input Completeness (Definition 5)

Let st and $\mathsf{st}_{\mathsf{Pt}_i}$ be respectively the states of $\mathcal{P}$ and of an honest player Pt_i after Step 7 of Π_{pvZK} (that is, after the proof has been computed). Since both $\mathcal{P}$ and $\mathcal{V}$ are running the protocol honestly, from the chain-consistency property follows that $\mathbf{B}^{\lceil \eta} \leq \tilde{\mathbf{B}}$ (with overwhelming probability), where $\mathbf{B} = \mathsf{GetRecords}(\mathsf{st})$ and $\tilde{\mathbf{B}} = \mathsf{GetRecords}(\mathsf{st}_{\mathsf{Pt}_i})$. Therefore $\mathcal{V}$ performs all the blockchain checks on $\tilde{\mathbf{B}}$ successfully. After that $\mathcal{P}$ posts the commitments $\{\mathtt{com}_j\}_{j=1}^{u \cdot d}$ in the blockchain $\mathbf{B}$ we are guaranteed by the chain growth property of Γ^{V} and by Assumption 1 that new d blocks will be added to $\mathbf{B}$ and among them n will be pristine. Therefore $\mathcal{P}$ can construct the instance $x_{\mathtt{com}}$ (as defined in Step 3 of Fig. 2) in order to complete her execution running Π_{pvWI}.

Finally the completeness of Π_{pvZK} follows from the completeness of Π_{pvWI} and the correctness of Π_{Com}.

4.2 Soundness (Definition 4)

Claim 1. *If Assumption 1 holds for* Γ^{V} *then* Π_{pvZK} *is sound.*

Proof. Let $\mathcal{P}^{\star}_{\mathsf{pvZK}}$ be a successful adversary. Recall that $\mathcal{P}^{\star}_{\mathsf{pvZK}}$ is successful if it produces with non-negligible probability an accepting π of Π_{pvZK} w.r.t. $x \notin \mathcal{L}$, where x is adaptively chosen by $\mathcal{P}^{\star}_{\mathsf{pvZK}}$ before the last message of π.

Let B^* be the block in the blockchain $\mathbf{B}$ where the last commitment of the set of the commitments $\mathtt{com}_1, \ldots, \mathtt{com}_{u \cdot d}$ is posted by $\mathcal{P}^{\star}_{\mathsf{pvZK}}$, and let $B_1, \ldots, B_n$ be the n pristine blocks (in a sequence of d blocks) appeared in $\mathbf{B}$ after the block B^*.

From Assumption 1 it follows that in a sequence of n pristine blocks $B_1, \ldots, B_n$ at least $n/2 + 1$ are generated by honest players through good executions of $\mathsf{GenBlock}$ w.r.t. $\mathbf{B}$. Let $\overline{B}_1, \ldots, \overline{B}_{n/2+1}$ be the $n/2 + 1$ blocks generated by honest players through good executions of $\mathsf{GenBlock}$ w.r.t. $\mathbf{B}$ in the

sequence of pristine blocks $B_1, \ldots, B_n$, and the value $\overline{v}_j$ be s.t. $\overline{v}_j = \mathtt{trim}(\overline{B}_j)$, for $j = 1, \ldots, n/2+1$. When $\mathcal{P}^{\star}_{\mathsf{pvZK}}$ posts $\mathtt{com}_1, \ldots, \mathtt{com}_{u \cdot d}$, it has no information about the values $\overline{v}_1, \ldots, \overline{v}_{n/2+1}$, because when $\mathcal{P}^{\star}_{\mathsf{pvZK}}$ posts $\mathtt{com}_1, \ldots, \mathtt{com}_{u \cdot d}$ each value $\overline{v}_j$ (for $j = 1, \ldots, n/2+1$) can be guessed with probability $2^{-\lambda}$ (since Assumption 1 holds and each $\overline{v}_j$ has at least λ bits of min-entropy). Moreover, since Π_{Com} is a perfectly binding commitment scheme, the committed message is uniquely identified in the commitment phase. Therefore the probability that $\mathcal{P}^{\star}_{\mathsf{pvZK}}$ correctly commits the values $\overline{v}_1, \ldots, \overline{v}_{n/2+1}$ is negligible. It follows that the values $\overline{v}_1, \ldots, \overline{v}_{n/2+1}$ are committed in $\mathtt{com}_1, \ldots, \mathtt{com}_{u \cdot d}$ only with negligible probability, therefore $x_{\mathtt{com}} \notin \mathcal{L}_{\mathtt{com}}$. Since by contradiction we are assuming that $\mathcal{P}^{\star}_{\mathsf{pvZK}}$ is successful w.r.t. $x \notin \mathcal{L}$, it follows that with non-negligible probability $x_{\mathsf{pvWI}} = (x_{\mathtt{com}}, x) \notin \mathcal{L}_{\mathsf{pvWI}}$. This contradicts the soundness property of Π_{pvWI}

4.3 Zero Knowledge w.r.t. Blockchain Failure (Definition 7)

Simulator $\mathsf{S}_{\mathsf{pvZK}}$. The simulator $\mathsf{S}_{\mathsf{pvZK}}$ is presented in Fig. 3, the red steps denote the steps of $\mathsf{S}_{\mathsf{pvZK}}$ that are different from the one of $\mathcal{P}_{\mathsf{pvZK}}$.

Zero Knowledge w.r.t. Blockchain Failure. Let $\mathcal{A}$ be the adversary as defined in Definitions 7. Intuitively, we want to prove that even if the blockchain collapses, the zero-knowledge property of Π_{pvZK} is still preserved.

In order to show that Π_{pvZK} satisfies zero knowledge w.r.t. blockchain failure we will consider the following hybrid experiments.

- Hybrid H_0. In hybrid experiment $H_0(\lambda)$ the simulator $\mathsf{S}'_{\mathsf{pvZK}}$ follows the honest prover procedure of $\mathcal{P}_{\mathsf{pvZK}}$.
- Hybrid H_1. Experiment $H_1(\lambda)$ is described as $H_0(\lambda)$ except that the simulator $\mathsf{S}'_{\mathsf{pvZK}}$ emulates the honest players in the execution of Γ^{V}, more precisely $\mathsf{S}'_{\mathsf{pvZK}}$ follows Step 3 and Steps 14 and 15 of Fig. 3.
  Note that after that the commitments are posted in the blockchain in $H_0(\lambda)$ when an honest player $\mathsf{Pt}_j \in \mathcal{H}$ receives a request from $\mathcal{Z}$ of an execution of $\mathsf{GenBlock}$ using fresh randomness for $f_{\mathsf{ID}}(1^{\lambda}; \cdot)$ Pt_j runs f_{ID} on input freshly generated randomness obtaining a freshly generated value v. It easy to see that in $H_1(\lambda)$ the value v is generated in the same way as $\mathsf{Pt}_j \in \mathcal{H}$ does in $H_0(\lambda)$ except that v is computed at the start of Π_{pvZK}. Since (1) the values $v_j \leftarrow f_{\mathsf{ID}}(1^{\lambda}; r_j)$ for $j = 1, \ldots, d \cdot u$ are identically distributed in the two hybrid experiments and (2) $\mathsf{S}'_{\mathsf{pvZK}}$ is behaving in the same way of the honest players in an execution of Γ^{V}, we have that $H_1 \equiv H_0$.
- Hybrid H_2. If a corruption of the form $\langle \mathtt{ZK}_{\mathtt{corr}}(x, w) \rangle$ occurs when Π_{pvZK} starts, $H_2(\lambda)$ corresponds to $H_1(\lambda)$, otherwise we consider a series of hybrid experiments $H_2^0(\lambda), \ldots, H_2^{u \cdot d}(\lambda)$ where $H_2^0(\lambda) = H_1(\lambda)$ and $H_2(\lambda) = H_2^{u \cdot d}(\lambda)$ and they are described as follows.

  Hybrid H_2^k with $k \in \{1, \ldots, u \cdot d\}$. The hybrid experiment $H_2^k(\lambda)$ is describe ad $H_2^{k-1}(\lambda)$ except that $\mathsf{S}'_{\mathsf{pvZK}}$ computes the k-th commitment following Steps 2–4 of Fig. 3. Indeed, $\mathsf{S}'_{\mathsf{pvZK}}$ computes $(\mathtt{com}_j, \mathtt{open}_j) \leftarrow \mathsf{Com}(v_j)$ for $j = 1, \ldots, k$ (where $v_j \leftarrow f_{\mathsf{ID}}(1^{\lambda}; r_j)$) and it computes $(\mathtt{com}_j, \mathtt{open}_j) \leftarrow \mathsf{Com}(0^q)$ for $j = k+1, \ldots, u \cdot d$.

Simulator Procedure: $\mathsf{S}_{\mathsf{pvZK}}$. Parameters are defined in Table 4.

— **First step.**

1. If a corruption request $\langle \mathtt{ZK}_{\mathtt{corr}}(x,w)\rangle$ is received, then execute the steps of $\mathcal{P}_{\mathsf{pvZK}}$ on input x, w. Else continue with the following steps.
2. For $j = 1, \ldots, u \cdot d$:
3. Pick r_j randomly in $\{0,1\}^{\mathsf{poly}(\lambda)}$ compute $v_j \leftarrow f_{\mathsf{ID}}(1^\lambda; r_j)$ and set $R = R||r_j$.
4. Compute $(\mathtt{com}_j, \mathtt{open}_j) \leftarrow \mathsf{Com}(v_j)$.

— **Blockchain Interaction.**

5. Set $\mathtt{st} = \epsilon$. Post $\mathtt{com}_1, \ldots, \mathtt{com}_{u\cdot d}$ on the blockchain by running $\mathsf{Broadcast}(1^\lambda, \mathtt{com}_1, \ldots, \mathtt{com}_{u\cdot d})$ and then monitor the blockchain by running $\mathtt{st} = \mathsf{UpdateState}(1^\lambda, \mathtt{st})$, $\mathbf{B} = \mathsf{GetRecords}(1^\lambda, \mathtt{st})$, until $\mathtt{com}_1, \ldots, \mathtt{com}_{u\cdot d}$ followed by d additional blocks $B_1, \ldots, B_d$ are posted on the blockchain $\mathbf{B}^{\lceil \eta}$. Let $\overline{B_1}, \ldots, \overline{B_n}$ be the n pristine blocks in the sequence $B_1, \ldots, B_d$.

— **Second step.**

6. Compute $v_j = \mathtt{trim}(\overline{B_j})$ for $j = 1, \ldots, n$ and set $\mathtt{com} = \{\mathtt{com}_j\}_{j=1}^{u\cdot d}$, $\mathtt{val} = \{v_j\}_{j=1}^{n}$, $x_{\mathtt{com}} = (\mathtt{com}, \mathtt{val})$, $\pi^0_{\mathsf{pvWI}} = (1^\lambda, \ell)$.
7. Let $B_{j_1}, \ldots, B_{j_k}$ be the pristine blocks generated by honest players in the sequence $B_1, \ldots, B_d$ set $w_{\mathtt{com}} = \mathtt{open}_{j_1}, \ldots, \mathtt{open}_{j_k}$ (where $k \geq n/2 + 1$ by Assumption 1).
8. Obtain π^1_{pvWI} with randomness r_1 executing $\mathcal{P}_{\mathsf{pvWI}}$ on input $1^\lambda, \ell$ and interacting with the blockchain if it is required by $\mathcal{P}_{\mathsf{pvWI}}$. If a corruption request $\langle \mathtt{ZK}_{\mathtt{corr}}(x,w)\rangle$ is received: erase the values $\{\mathtt{open}_j\}_{j=1}^{u\cdot d}$ and output $\mathtt{st}_{\mathcal{P}_{\mathsf{pvZK}}} = r^1$ and π^1.
9. For $i = 2, \ldots, m-1$:
   Obtain $r^i, \pi^i_{\mathsf{pvWI}}$ executing $\mathcal{P}_{\mathsf{pvWI}}$ on input r^{i-1}, and $\pi^{i-1}_{\mathsf{pvWI}}$ interacting with the blockchain if it is required by $\mathcal{P}_{\mathsf{pvWI}}$.
   If a corruption request $\langle \mathtt{ZK}_{\mathtt{corr}}(x,w)\rangle$ is received.
   Erase the values $\{\mathtt{open}_j\}_{j=1}^{u\cdot d}$. Output $\mathtt{st}_{\mathcal{P}_{\mathsf{pvZK}}} = r'||r^i$ and $\pi^1, \ldots, \pi^i$.
10. If a corruption request $\langle \mathtt{ZK}_{\mathtt{corr}}(x,w)\rangle$ was not received, then:
11. Upon receiving x from $\mathcal{A}$, set $x_{\mathsf{pvWI}} = (x, x_{\mathtt{com}})$
    Obtain π^m_{pvWI} executing $\mathcal{P}_{\mathsf{pvWI}}$ with randomness on input $\pi^{m-1}_{\mathsf{pvWI}}, x_{\mathsf{pvWI}}, w_{\mathtt{com}}$ and interacting with the blockchain if it is required by $\mathcal{P}_{\mathsf{pvWI}}$.
    Set $\pi_{\mathsf{pvWI}} = (\pi^1_{\mathsf{pvWI}}, \ldots, \pi^m_{\mathsf{pvWI}})$ and $\pi = (x_{\mathsf{pvWI}}, \{\mathtt{com}_j\}_{j=1}^{n}, \pi_{\mathsf{pvWI}})$.
    Obtain $\mathtt{st}_{\mathcal{P}_{\mathsf{pvWI}}}$ from $\mathcal{P}_{\mathsf{pvWI}}$ set $\mathtt{st}_{\mathcal{P}_{\mathsf{pvZK}}} = \mathtt{st}_{\mathsf{pvWI}}$ and erase $\{\mathtt{open}\}_{i=1}^{u\cdot d}$. Output π.
12. If a corruption request $\langle \mathtt{ZK}_{\mathtt{corr}}(x,w)\rangle$ is received: output $\mathtt{st}_{\mathcal{P}_{\mathsf{pvZK}}}$.

— Execution of Γ^{V} simulating honest player Pt_j. Act on behalf of Pt_j as described in Section 3.2, in particular, upon receiving a request of an execution of $\mathsf{GenBlock}$ using fresh randomness for the function $f_{\mathsf{ID}}(1^\lambda; \cdot)$:

13. Run $\mathbf{B} = \mathsf{GetRecords}(1^\lambda, \mathtt{st}_j)$, let np be number of pristine blocks posted after $\mathtt{com}_1, \ldots, \mathtt{com}_{u\cdot d}$ in the blockchain $\mathbf{B}^{\lceil \eta}$. Let K be the number of blocks added in the blockchain $\mathbf{B}^{\lceil \eta}$. Let nb be the number of honest executions of $\mathsf{GenBlock}$ already executed for the block B_{K+1}.
14. If $0 \leq np < n$: Parse R as $r_1, \ldots, r_{u\cdot d}$ and run an execution of $\mathsf{GenBlock}$ on behalf of honest player Pt_j with randomness r_{np+nb} to execute f_{ID}.
15. Else: Pick r at random from $\{0,1\}^{\mathsf{poly}(\lambda)}$ and $\mathsf{GenBlock}$ on behalf of honest player Pt_j with randomness r to execute f_{ID}.
16. If $\mathcal{A}$ sends a collapse request $\langle \mathtt{corr}, \mathtt{all}\rangle$ compute the following steps: Disclose state $\mathtt{st}_{\mathsf{Pt}_i}$ of honest player Pt_i, for all $i = 1, \ldots, |\mathcal{H}|$. If a corruption request $\langle \mathtt{ZK}_{\mathtt{corr}}(x,w)\rangle$ did not occur obtain $\mathtt{st}_{\mathcal{P}_{\mathsf{pvWI}}}$ from $\mathcal{P}_{\mathsf{pvWI}}$ set $\mathtt{st}_{\mathcal{P}_{\mathsf{pvZK}}} = \mathtt{st}_{\mathsf{pvWI}}$ and disclose $\mathtt{st}_{\mathcal{P}_{\mathsf{pvZK}}}$.

Fig. 3. Simulator $\mathsf{S}_{\mathsf{pvZK}}$ of Π_{pvZK}.

Assuming secure erasure, from Claim 2 it holds that $H_2^{k-1} \approx H_2^k$ for all $k = 1, \ldots, u \cdot d$, therefore since H_1 corresponds to H_2^0 and H_2 corresponds to $H_2^{u \cdot d}$ we conclude that $H_1(\lambda) \approx H_2(\lambda)$.

– Hybrid H_3. If a corruption of the form $\langle \mathtt{ZK}_{\mathtt{corr}}(x, w) \rangle$ occurs during the computation of the first $m-1$ messages of Π_{pvWI}, we have that $H_2(\lambda)$ corresponds to $H_3(\lambda)$. Indeed due to the delayed-input property of Π_{pvWI}, $\mathsf{S}'_{\mathsf{pvZK}}$ computes the first $m-1$ messages of Π_{pvWI} as $\mathcal{P}_{\mathsf{pvZK}}$ does. Note that the decommitment information $\{\mathtt{open}_j\}_{j=1}^{u \cdot d}$ are securely erased by $\mathcal{P}_{\mathsf{pvZK}}$, therefore if $\mathsf{S}'_{\mathsf{pvZK}}$ receives a corruption request during the computation of the first $m-1$ messages of Π_{pvWI} she is able to exhibit randomness that is identically distributed to the one that $\mathcal{P}_{\mathsf{pvZK}}$ would have in her state.
If a corruption of the form $\langle \mathtt{ZK}_{\mathtt{corr}}(x, w) \rangle$ does not occur during the computation of the first $m-1$ messages of Π_{pvWI}, then H_3 is defined as follow.
The hybrid experiment $H_3(\lambda)$ is described exactly as $H_2(\lambda)$ except for the witness used to compute the last message π^m_{pvWI} generated using Π_{pvWI}, for which $\mathsf{S}'_{\mathsf{pvZK}}$ is acting as $\mathsf{S}_{\mathsf{pvZK}}$. In more details, for the computation of the message π^m_{pvWI} $\mathsf{S}'_{\mathsf{pvZK}}$ is behaving as described in Steps 11 of Fig. 3. Assuming secure erasure, since Π_{pvWI} satisfies WI w.r.t. blockchain failure it follows that $H_2(\lambda) \approx H_3(\lambda)$ (see Claim 3).

$H_0(\lambda)$ corresponds to the experiment where $\mathcal{P}_{\mathsf{pvZK}}$ is interacting with $\mathcal{A}$ and $H_3(\lambda)$ corresponds to the experiment where $\mathsf{S}_{\mathsf{pvZK}}$ is interacting with $\mathcal{A}$. Since $H_3(\lambda) \approx H_0(\lambda)$ it follows that $\mathcal{A}$ distinguishes the two experiments only with negligible probability.

Claim 2. *Assume that $\Pi_{\mathtt{com}}$ satisfies computationally hiding secure erasure, and the blockchain protocol Γ^{V} satisfies Assumption 1, then for every pair of messages $m_0, m_1 \in \{0,1\}^q$ it holds that $H_2^{k-1}(\lambda) \approx H_2^k(\lambda)$ for $k \in \{1, \ldots, u \cdot d\}$.*

Proof. Suppose by contradiction that the above claim does not hold, this implies that there exists an adversary $\mathcal{A}$ that is able to distinguish between $H_2^{k-1}(\lambda)$ and $H_2^k(\lambda)$. Note that $\mathcal{A}$ could wait until the protocol Π_{pvZK} ends and then can send a collapse request $\langle \mathtt{corr}, \mathtt{all} \rangle$. Using $\mathcal{A}$ it is possible to construct a malicious receiver $\mathcal{A}_{\mathsf{Com}}$ that breaks the hiding of Π_{Com} with non-negligible probability.
Let $\mathcal{CH}$ be the challenger of the hiding game of Π_{Com}. $\mathcal{A}_{\mathsf{Com}}$ computes the following steps:

1. Compute v_k running $f_{\mathsf{ID}}(1^\lambda; r)$ where r is an uniformly chosen randomness and sends the messages $m_0 = 0^q$ and $m_1 = v_k$ to $\mathcal{CH}$.
2. Upon receiving $\tilde{\mathtt{com}}_k$ from $\mathcal{CH}$, $\mathcal{A}_{\mathsf{Com}}$ interacts with $\mathcal{A}$ computing all the messages of $\mathsf{S}'_{\mathsf{pvZK}}$ following the steps described in $H_2^k(\lambda)$ (and in $H_2^{k-1}(\lambda)$) except for the k-th commitment for which she uses $\tilde{\mathtt{com}}_k$.
3. Emulation of the state $\mathtt{st}_{\mathcal{P}_{\mathsf{pvZK}}}$ of $\mathcal{P}_{\mathsf{pvZK}}$ after π is compute: acts as $\mathsf{S}'_{\mathsf{pvZK}}$ in $H_2^k(\lambda)$ (and in $H_2^{k-1}(\lambda)$) and securely erase the decommitment information $\{\mathtt{open}_j\}_{j=1}^{u \cdot d}$ (except for $\tilde{\mathtt{open}}_k$ that was never available to $\mathcal{A}_{\mathsf{Com}}$), set the state $\mathtt{st}_{\mathcal{P}_{\mathsf{pvZK}}}$ as described $H_2^k(\lambda)$ (and in $H_2^{k-1}(\lambda)$) that is as described in Step 12 of Fig. 3.

4. execution of Γ^{V} :
   4.1. Emulate the honest players acting as the honest player of Γ^{V} (as described in Section $H_2^k(\lambda)$ (and in $H_2^{k-1}(\lambda)$)).
   4.2. After π of Π_{pvZK} is computed if $\mathcal{A}$ sends a collapse request $\langle \mathtt{corr}, \mathtt{all} \rangle$, disclose the states of all the honest players $\mathsf{st}_{\mathsf{P}_{\mathsf{t}_1}}, \ldots, \mathsf{st}_{\mathsf{P}_{\mathsf{t}_{|\mathcal{H}|}}}$ and $\mathsf{st}_{\mathcal{P}_{\mathsf{pvZK}}}$.
5. When $\mathcal{A}$ stops, $\mathcal{A}_{\mathsf{Com}}$ outputs the outcome of $\mathcal{A}$.

$\mathcal{A}_{\mathsf{Com}}$ emulates the states of all the honest players $\mathsf{st}_{\mathsf{P}_{\mathsf{t}_1}}, \ldots, \mathsf{st}_{\mathsf{P}_{\mathsf{t}_{|\mathcal{H}|}}}$ in a perfect manner, since $\mathcal{A}_{\mathsf{Com}}$ just acts as the honest players in the execution of Γ^{V}. Moreover, $\mathsf{st}_{\mathcal{P}_{\mathsf{pvZK}}}$ after π is computed in Step 3 of the above procedure, corresponds to the state of an honest $\mathcal{P}_{\mathsf{pvZK}}$ in $H_2^k(\lambda)$ (and in $H_2^{k-1}(\lambda)$). The proof is concluded observing that if $\mathcal{CH}$ uses the message m_0 to compute $\tilde{\mathtt{com}}_k$ then the reduction is distributed as H_2^{k-1} and as H_2^k otherwise.

Claim 3. *Assume that Π_{pvWI} satisfies WI w.r.t. blockchain failure as in Definition 6 over any blockchain protocol, secure erasure, and the blockchain protocol Γ^{V} satisfies Assumption 1, then for every $x_{\mathsf{pvWI}}, w_0, w_1$ s.t. $(x_{\mathsf{pvWI}}, w_0) \in \mathcal{R}_{\mathsf{pvWI}}$ and $(x_{\mathsf{pvWI}}, w_1) \in \mathcal{R}_{\mathsf{pvWI}}$ it holds that $H_2(\lambda) \approx H_3(\lambda)$.*

Proof. Suppose by contradiction that the above claim does not hold, this implies that there exists an adversary $\mathcal{A}$ that is able to distinguish between $H_2(\lambda)$ and $H_3(\lambda)$. Note that $\mathcal{A}$ could wait until the protocol Π_{pvZK} ends and then can send a collapse request $\langle \mathtt{corr}, \mathtt{all} \rangle$. Using $\mathcal{A}$ it is possible to construct a malicious verifier $\mathcal{A}_{\mathsf{pvWI}}$ that breaks the WI w.r.t. blockchain failure w.r.t. any blockchain protocol property of Π_{pvWI}. We remark that Π_{pvWI} enjoys WI w.r.t. blockchain failure w.r.t. any blockchain protocol (i.e., even w.r.t. a blockchain protocol where $\mathcal{P}_{\mathsf{pvWI}}$ is the only honest player of the blockchain protocol). Let $\mathcal{CH}$ be the challenger of the WI w.r.t. blockchain failure game of Π_{pvWI}. $\mathcal{A}_{\mathsf{pvWI}}$ computes the following steps.

1. $\mathcal{A}_{\mathsf{pvWI}}$ acts as described in $H_2(\lambda)$ and $H_3(\lambda)$ until Step 6 of Fig. 3. In particular, $\mathcal{A}_{\mathsf{pvWI}}$ computes the instance $x_{\mathtt{com}}$ and the witness $w_{\mathtt{com}}$ as explained, respectively, in Step 6 and in Steps 7, 14 and 15 of Fig. 3.
2. $\mathcal{A}_{\mathsf{pvWI}}$ interacts as a proxy between $\mathcal{CH}$ and $\mathcal{A}$ for the messages π^1_{pvWI}, $\ldots, \pi^{m-1}_{\mathsf{pvWI}}$, and interacting with the blockchain as a $\mathcal{P}_{\mathsf{pvWI}}$ would do upon request of $\mathcal{CH}$.
3. $\mathcal{A}$ chooses $(x, w) \in \mathcal{R}$ before the last message of Π_{pvZK} and therefore $\mathcal{A}_{\mathsf{pvWI}}$ (that is acting as $\mathcal{P}_{\mathsf{pvZK}}$) will obtain w s.t. $(x, w) \in \mathcal{R}$ and sends $x_{\mathsf{pvWI}} = (x, x_{\mathtt{com}}), w, w_{\mathtt{com}}$ to $\mathcal{CH}$ before the message π^m_{pvWI}. $\mathcal{A}_{\mathsf{pvWI}}$ completes the proof π of Π_{pvZK} using π^m_{pvWI} and interacting with the blockchain as a $\mathcal{P}_{\mathsf{pvWI}}$ would do upon request of $\mathcal{CH}$.
   3.1. Emulation of the state $\mathsf{st}_{\mathcal{P}_{\mathsf{pvZK}}}$ of $\mathcal{P}_{\mathsf{pvZK}}$ after π is computed:
      i. $\mathcal{A}_{\mathsf{pvWI}}$ sends a collapse request $\langle \mathtt{corr}, \mathtt{all} \rangle$ to $\mathcal{CH}$ obtaining $\mathsf{st}_{\mathcal{P}_{\mathsf{pvWI}}}$ from the challenger $\mathcal{CH}$.
      ii. $\mathcal{A}_{\mathsf{pvWI}}$ is acting as $\mathsf{S}'_{\mathsf{pvZK}}$ in $H_2(\lambda)$ (and in $H_3(\lambda)$) and securely erases the decommitment information $\{\mathtt{open}_j\}_{j=1}^{u \cdot d}$, set $\mathsf{st}_{\mathcal{P}_{\mathsf{pvZK}}} = \mathsf{st}_{\mathcal{P}_{\mathsf{pvWI}}}$.

4. execution of Γ^{V} :
   4.1. $\mathcal{A}_{\mathsf{pvWI}}$ emulates the honest players acting as the honest players of Γ^{V} as described in $H_3(\lambda)$ (and in $H_2(\lambda)$).
   4.2. After π is computed if $\mathcal{A}$ sends a collapse request $\langle \mathtt{corr}, \mathtt{all} \rangle$, $\mathcal{A}_{\mathsf{pvWI}}$ discloses the states of all the honest players $\mathsf{st}_{\mathsf{P}_{\mathsf{t}_1}}, \ldots, \mathsf{st}_{\mathsf{P}_{\mathsf{t}_{|\mathcal{H}|}}}$ and $\mathsf{st}_{\mathcal{P}_{\mathsf{pvZK}}}$.
5. When $\mathcal{A}$ stops, $\mathcal{A}_{\mathsf{pvWI}}$ outputs the outcome of $\mathcal{A}$.

We note that $\mathcal{A}_{\mathsf{pvWI}}$ simulates the states of all the honest players $\mathsf{st}_{\mathsf{P}_{\mathsf{t}_1}}, \ldots, \mathsf{st}_{\mathsf{P}_{\mathsf{t}_{|\mathcal{H}|}}}$ in a perfect way, this is because in the execution of Γ^{V}, $\mathcal{A}_{\mathsf{pvWI}}$ is behaving in the same way of the honest players of an execution of Γ^{V} (as described in $H_3(\lambda)$ (and in $H_2(\lambda)$)). The proof is concluded observing that if $\mathcal{CH}$ uses the witness w to compute π^m_{pvWI} then the reduction is distributed as H_2, and as H_3 otherwise.

5 On Public Verifiability in [16]

A recent work [16] models the blockchain as a global ledger functionality $\mathcal{G}_{\mathtt{ledger}}$ available to all the participants of a cryptographic protocol. [16] constructs concurrent self-composable secure computation protocol for general functionalities in such global ledger model. The protocols constructed in [16] are not publicly verifiable, and therefore do not satisfy the main feature that we study and achieve in this work. Indeed the authors of [16] already notice in their work that non-interactive zero knowledge for NP is impossible in their model. We remark that actually the impossibility extends also to publicly verifiable zero knowledge for languages that are not in BPP and we give now a high-level intuition. First of all, note that in the model of [16], since the blockchain is modeled as a global ledger, the simulator S of the zero-knowledge property has the same power of the adversary while accessing $\mathcal{G}_{\mathtt{ledger}}$. Suppose now by contradiction that it is possible to construct a publicly verifiable zero-knowledge argument $\Pi = (\mathcal{P}, \mathcal{V})$ for the $\mathcal{NP}$-language $\mathcal{L}$ in the $\mathcal{G}_{\mathtt{ledger}}$ model. This means that there exists a simulator S that having access to $\mathcal{G}_{\mathtt{ledger}}$ on input any instance $x \in \mathcal{L}$ outputs an accepting proof π w.r.t. x that is (computationally) indistinguishable from a proof generated by a honest prover $\mathcal{P}$. Let us now consider a malicious polynomial-time prover $\mathcal{P}^*$ that in the $\mathcal{G}_{\mathtt{ledger}}$-model wants to prove a false statement x^* to an honest verifier $\mathcal{V}$. We will show that $\mathcal{P}^*$ proves a false theorem with non-negligible probability, $\mathcal{P}^*$ works as follows. $\mathcal{P}^*$ internally runs S on input x^*. Moreover, each interaction that S wants to do with $\mathcal{G}_{\mathtt{ledger}}$ is emulated by $\mathcal{P}^*$ and this is possible since S and $\mathcal{P}^*$ are accessing $\mathcal{G}_{\mathtt{ledger}}$ in the same way. At the end of the execution, S outputs π^* w.r.t. x^*. $\mathcal{P}^*$ forwards π^* to $\mathcal{V}$. Note that we are guaranteed by the zero-knowledge property that π^* is accepting and the view of an honest verifier that receives π^* from $\mathcal{P}^*$ is (computationally) indistinguishable from the view that $\mathcal{V}$ has when she receives a proof from an honest prover. Finally we note that public verifiability guarantees that π^* can be accepted by any verifier non-interactively, The only caveat in the above reasoning can concern the fact that S might refuse to produce an accepting proof when $x \notin \mathcal{L}$. However this immediately shows that the language $\mathcal{L}$ is in BPP.

Acknowledgments. Research supported in part by NSF grants #1012798,#1764025, and in part by the European Union's Horizon 2020 research and innovation programme under grant agreement No 780477 (project PRIViLEDGE).

References

1. Pencil - workshop on privacy enhancing cryptography in ledgers (2019). https://priviledge-project.eu/pencil
2. Aggarwal, D., Obremski, M., Ribeiro, J., Siniscalchi, L., Visconti, I.: How to extract useful randomness from unreliable sources. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020. LNCS, vol. 12105, pp. 343–372. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45721-1_13
3. Badertscher, C., Garay, J., Maurer, U., Tschudi, D., Zikas, V.: But why does it work? A rational protocol design treatment of bitcoin. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10821, pp. 34–65. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78375-8_2
4. Badertscher, C., Gazi, P., Kiayias, A., Russell, A., Zikas, V.: Ouroboros genesis: composable proof-of-stake blockchains with dynamic availability. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, 15–19 October 2018, pp. 913–930 (2018)
5. Badertscher, C., Maurer, U., Tschudi, D., Zikas, V.: Bitcoin as a transaction ledger: a composable treatment. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10401, pp. 324–356. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_11
6. Baldimtsi, F., Kiayias, A., Zacharias, T., Zhang, B.: Crowd verifiable zero-knowledge and end-to-end verifiable multiparty computation. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020. LNCS, vol. 12493, pp. 717–748. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64840-4_24
7. Barak, B.: How to go beyond the black-box simulation barrier. In: 42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, , Las Vegas, Nevada, USA, 14–17 October 2001, pp. 106–115. IEEE Computer Society (2001)
8. Bellare, M., Rogaway, P.: Optimal asymmetric encryption. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 92–111. Springer, Heidelberg (1995). https://doi.org/10.1007/BFb0053428
9. Ben-Sasson, E., Chiesa, A., Gabizon, A., Virza, M.: Quasi-linear size zero knowledge from linear-algebraic PCPs. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016. LNCS, vol. 9563, pp. 33–64. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49099-0_2
10. Benhamouda, F., et al.: Can a public blockchain keep a secret? In: Pass, R., Pietrzak, K. (eds.) TCC 2020. LNCS, vol. 12550, pp. 260–290. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64375-1_10
11. Bentov, I., Gabizon, A., Zuckerman, D.: Bitcoin beacon. CoRR abs/1605.04559 (2016)
12. Bentov, I., Kumaresan, R.: How to use bitcoin to design fair protocols. IACR Crypto. ePrint Arch. **2014**, 129 (2014)
13. Bonneau, J., Clark, J., Goldfeder, S.: On bitcoin as a public randomness source. Cryptology ePrint Archive, Report 2015/1015 (2015). https://eprint.iacr.org/2015/1015

14. Bowe, S., Gabizon, A., Green, M.D.: A multi-party protocol for constructing the public parameters of the Pinocchio zk-SNARK. In: Zohar, A., et al. (eds.) FC 2018. LNCS, vol. 10958, pp. 64–77. Springer, Heidelberg (2019). https://doi.org/10.1007/978-3-662-58820-8_5
15. Canetti, R.: Universally composable security: a new paradigm for cryptographic protocols. In: 42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, Las Vegas, Nevada, USA, 14–17 October 2001, pp. 136–145. IEEE Computer Society (2001)
16. Choudhuri, A.R., Goyal, V., Jain, A.: Founding secure computation on blockchains. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019. LNCS, vol. 11477, pp. 351–380. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17656-3_13
17. Feige, U., Lapidot, D., Shamir, A.: Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In: 31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, 22–24 October 1990, vol. 1, pp. 308–317. IEEE Computer Society (1990)
18. Ganesh, C., Orlandi, C., Tschudi, D.: Proof-of-stake protocols for privacy-aware blockchains. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019. LNCS, vol. 11476, pp. 690–719. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17653-2_23
19. Garay, J., Kiayias, A., Leonardos, N.: The bitcoin backbone protocol: analysis and applications. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 281–310. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46803-6_10
20. Gennaro, R., Gentry, C., Parno, B., Raykova, M.: Quadratic span programs and succinct NIZKs without PCPs. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 626–645. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38348-9_37
21. Gentry, C., Halevi, S., Magri, B., Nielsen, J.B., Yakoubov, S.: Random-index PIR with applications to large-scale secure MPC. Cryptology ePrint Archive, Report 2020/1248 (2020). https://eprint.iacr.org/2020/1248
22. Gilad, Y., Hemo, R., Micali, S., Vlachos, G., Zeldovich, N.: Algorand: scaling Byzantine agreements for cryptocurrencies. In: Proceedings of the 26th Symposium on Operating Systems Principles, Shanghai, China, 28–31 October 2017, pp. 51–68 (2017)
23. Goyal, R., Goyal, V.: Overcoming cryptographic impossibility results using blockchains. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017. LNCS, vol. 10677, pp. 529–561. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70500-2_18
24. Goyal, R., Goyal, V.: Overcoming cryptographic impossibility results using blockchains. Cryptology ePrint Archive, Report 2017/935 (2017). https://eprint.iacr.org/2017/935
25. Goyal, V., Kothapalli, A., Masserova, E., Parno, B., Song, Y.: Storing and retrieving secrets on a blockchain. IACR Cryptol. ePrint Arch. **2020**, 504 (2020)
26. Groth, J., Kohlweiss, M., Maller, M., Meiklejohn, S., Miers, I.: Updatable and universal common reference strings with applications to zk-SNARKs. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018. LNCS, vol. 10993, pp. 698–728. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96878-0_24
27. Juels, A., Kosba, A.E., Shi, E.: The ring of gyges: investigating the future of criminal smart contracts. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016, pp. 283–295 (2016)

28. Kerber, T., Kiayias, A., Kohlweiss, M., Zikas, V.: Ouroboros crypsinous: privacy-preserving proof-of-stake. In: 2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, 19–23 May 2019, pp. 157–174. IEEE (2019)
29. Kiayias, A., Panagiotakos, G.: Speed-security tradeoffs in blockchain protocols. IACR Cryptology ePrint Archive: Report 2015/1019 (2015). http://eprint.iacr.org/2015/1019
30. Maller, M., Bowe, S., Kohlweiss, M., Meiklejohn, S.: Sonic: zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, 11–15 November 2019, pp. 2111–2128. ACM (2019)
31. Nakamoto, S.: Bitcoin: a peer-to-peer electionic cash system (2008, unpublished)
32. Pass, R., Seeman, L., Shelat, A.: Analysis of the blockchain protocol in asynchronous networks. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10211, pp. 643–673. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56614-6_22
33. Pass, R., Shi, E.: The sleepy model of consensus. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017. LNCS, vol. 10625, pp. 380–409. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70697-9_14
34. Scafuro, A., Siniscalchi, L., Visconti, I.: Publicly verifiable proofs from blockchains. In: Lin, D., Sako, K. (eds.) PKC 2019. LNCS, vol. 11442, pp. 374–401. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17253-4_13
35. Scafuro, A., Siniscalchi, L., Visconti, I.: Publicly verifiable zero knowledge from (collapsing) blockchains. Cryptology ePrint Archive, Report 2020/1435. https://eprint.iacr.org/2020/1435