Overcoming Cryptographic Impossibility Results using Blockchains

Rishab Goyal* goyal@utexas.edu
Vipul Goyal vipul@cmu.edu

Abstract

Blockchain technology has the potential to disrupt how cryptography is done. In this work, we propose to view blockchains as an "enabler", much like indistinguishability obfuscation [BGI+12, GGH+13b, SW14] or one-way functions, for building a variety of cryptographic systems. Our contributions in this work are as follows:

1. *A Framework for Proof-of-Stake based Blockchains:* We provide an abstract framework for formally analyzing and defining useful security properties for Proof-of-Stake (POS) based blockchain protocols. Interestingly, for some of our applications, POS based protocols are more suitable. We believe our framework and assumptions would be useful in building applications on top of POS based blockchain protocols even in the future.
2. *Blockchains as an Alternative to Trusted Setup Assumptions in Cryptography:* A trusted setup, such as a common reference string (CRS) has been used to realize numerous systems in cryptography. The paragon example of a primitive requiring trusted setup is a non-interactive zero-knowledge (NIZK) system. We show that already existing blockchains systems including Bitcoin, Ethereum etc. can be used as a foundation (instead of a CRS) to realize NIZK systems.

   The novel aspect of our work is that it allows for utilizing an already existing (and widely trusted) setup rather than proposing a new one. Our construction does not require any additional functionality from the miners over the already existing ones, nor do we need to modify the underlying blockchain protocol. If an adversary can violate the security of our NIZK, it could potentially also take over billions of dollars worth of coins in the Bitcoin, Ethereum or any such cryptocurrency!

   We believe that such a "trusted setup" represents significant progress over using CRS published by a central trusted party. Indeed, NIZKs could further serve as a foundation for a variety of other cryptographic applications such as round efficient secure computation [KO04, HK07].
3. *One-time programs and pay-per use programs:* Goldwasser et al. [GKR08] introduced the notion of one time program and presented a construction using tamper-proof hardware. As noted by Goldwasser et al. [GKR08], clearly a one-time program cannot be solely software based, as software can always be copied and run again. While there have been a number of follow up works [GIS+10, BHR12a, AIKW15], there are indeed *no known constructions of one-time programs which do not rely on self destructing tamper-proof hardware (even if one uses trusted setup or random oracles)*. Somewhat surprisingly, we show that it is possible to base one-time programs on POS based blockchain systems without relying on trusted hardware. Our ideas do not seem to translate over to Proof-of-Work (POW) based blockchains.

   We also introduce the notion of pay-per-use programs which is simply a contract between two parties — service provider and customer. A service provider supplies a program such that if the customer transfers a specific amount of coins to the provider, it can evaluate the program on any input of its choice once, even if the provider is *offline*. This is naturally useful in a subscription based model where your payment is based on your usage.

*Supported by NSF CNS-1228599 and CNS-1414082, DARPA SafeWare.

1 Introduction

The last few years have seen a dramatic rise of cryptocurrencies such as Bitcoin [Nak08] and Ethereum [Woo14]. Some of these cryptocurrencies have a market capitalization running into several billion dollars. This has fuelled a significant interest in the underlying blockchain technology. Blockchain technology has the potential to disrupt how cryptography is done. Much of cryptography can be seen as eliminating the need to trust (and allow for dealing with adversarial parties which can't be trusted). Indeed the purpose of blockchains is something similar: eliminate the central point of trust in cryptocurrencies and possibly other applications. Thus we believe that a sustained effort to bring together "traditional cryptography" with the blockchain technology has the potential to be truly rewarding.

Blockchain Protocols. In a blockchain protocol, the goal of all parties is to maintain a (consistent) global ordered set of records. The set of records is "append only", and publicly visible. Furthermore, records can only be added using a special mechanism to reach consensus on what must be added to the existing blockchain. A protocol can employ any arbitrary technique or mechanism for participants to converge on a uniform and reliable blockchain state.

In most cryptocurrencies instantiated in the blockchain model, the special mechanism to reach consensus is called a *mining* procedure. It is used by all parties to extend the blockchain (i.e., add new blocks) and in turn (potentially) receive rewards for successfully generating a new block consistent with respect to current blockchain state. The mining procedure is meant to simulate a puzzle-solving race between protocol participants and could be run by any party. The rewards mostly consist of freshly generated currency. Presently, the mining procedures employed by most cryptocurrencies could be classified into two broad categories — *Proof-of-Work (POW)* and *Proof-of-Stake (POS)* based puzzles. The basic difference being that in POW puzzles, the probability of successful mining is proportional to the amount of computational power; whereas in POS, it is proportional to the number of coins in possession of a miner. Therefore, POW miners have to spend significant portion of their computational resources (and in turn, monetary resources) to extend the blockchain and in turn get rewarded, whereas POS miners spend significantly less computational resources and just need to have a sufficient balance.

Our Contributions. In this work, we propose to view blockchains as an "enabler", much like indistinguishability obfuscation [BGI$^+$12, GGH$^+$13b, SW14] or one-way functions, for building a variety of cryptographic systems. Basing cryptographic system on blockchains can provide very strong guarantees of the following form: *If an adversary could break the security of the cryptographic system, then it could also break the security of the underlying blockchain allowing it to potentially gain billions of dollars!* Indeed, this perspective is not new. Previous works [ADMM14b, BK14, ADMM14a, KB14, Jag15, LKW15] in this direction include using blockchains to construct fair secure multi-party computation, lottery systems, smart contracts and more. Our contributions in this work include the following:

- *A Framework for Proof-of-Stake based Blockchains:* We provide an abstract framework for formally analyzing and defining useful security properties and hardness relations for POS based blockchain protocols. Interestingly, we observe that for some of our applications, POS based protocols are more suitable than their POW counterparts. Furthermore, we also show how our framework can be instantiated based on existing POS based protocols [KRDO16, BPS16b].

  Previously, various works [GKL15, KP15, PSS16, KRDO16, PS16a, PS16b, BPS16a, BPS16b, GKLP16, GKL16] have analyzed the blockchain consensus protocols (of existing systems like Bitcoin) proving some fundamental properties as well as proposed new blockchain protocols. It is important to note that most of these works consider blockchain protocols with provable security guarantees as an end goal. However, as mentioned before, we consider blockchains as an "enabler". Therefore, we believe our framework and assumptions would be useful in building applications on top of POS based blockchain protocols even in the future.

Recently, it was suggested that blockchains could potentially be used to obtain a common random string as they can be used as a source of public randomness, thereby allowing to generate trusted random parameters [BCG15, BGZ16]. However, the results presented were limited in the sense that either adversaries with bounded budget were assumed or no security analysis was provided. We, on the other hand, proceed in an orthogonal direction by suggesting methods to directly extract cryptographic hardness from blockchains and developing hard-to-compute trapdoors with respect to blockchains.

- *Blockchains as an Alternative to Trusted Setup Assumptions in Cryptography:* A trusted setup, such as a common reference string (CRS) has been used to realize numerous systems in cryptography. Indeed, several of these systems have been shown to be impossible to realize without a trusted setup. In this work, we explore using blockchains as an alternative to a trusted setup (typically performed by a central trusted authority). The paragon example of a primitive requiring trusted setup is a non-interactive zero-knowledge (NIZK) system. Most well-known NIZK constructions are in the so called common reference string (CRS) model where there is a trusted third party which publishes some public parameters. However if the setup is done dishonestly, all security guarantees are lost.

  We show that already existing blockchains systems including Bitcoin, Ethereum etc. could potentially be used as a foundation (instead of a CRS) to realize NIZK systems. Thus, the complex blockchain system consisting of various miners and users can be seen as a "trusted setup". The idea of a decentralized setup for realizing NIZKs is not entirely new: Groth and Ostrovsky [GO07] propose NIZKs with n authorities where a majority of them must be honest. Goyal and Katz [GK08] propose a generalized model which allows for placing "differing levels of trust" in different authorities. However the novel aspect of our work is that it allows for utilizing an already existing (and widely trusted) setup rather than proposing a new one. Our construction does not require any additional functionality from the miners over the already existing ones, nor do we need to modify the underlying blockchain protocol.[1] If an adversary can violate the security of our NIZK, it could potentially also take over billions of dollars worth of coins in the Bitcoin, Ethereum or any such cryptocurrency!

  We believe that such a "trusted setup" represents significant progress over using CRS published by a central trusted party. Indeed, NIZKs could further serve as a foundation for a variety of other cryptographic applications such as round efficient secure computation [KO04, HK07].

- *One-time programs and pay-per use programs:* Say Alice wants to send a program to Bob. The program should run only once and then "self destruct". Is it possible to realize such "one-time programs"? Goldwasser et al. [GKR08] introduced the notion of one time program and presented a construction using tamper-proof hardware. A one-time program can be executed on a single input, whose value can be specified at run time. Other than the result of the computation on this input, nothing else about the program is leaked. One-time programs, for example, lead naturally to electronic cash or token schemes: coins or tokens are generated by a program that can only be run once, and thus cannot be double spent. In the construction of Goldwasser et al. [GKR08], a sender sends a set of very simple hardware tokens to a (potentially malicious) receiver. The hardware tokens allow the receiver to execute a program specified by the sender's tokens exactly once (or, more generally, up to a fixed t times).

  As noted by Goldwasser et al. [GKR08], clearly a one-time program cannot be solely software based, as software can always be copied and run again. While there have been a number of follow up works [GIS$^+$10, BHR12a, AIKW15], there are indeed *no known constructions of one-time programs which do not rely on self destructing tamper-proof hardware (even if one uses trusted setup or random oracles).* Somewhat surprisingly, we show that it is possible to base one-time programs on POS based blockchain systems without relying on trusted hardware. Our ideas do not seem to translate over to POW based blockchains. Our construction assumes the existence of extractable witness encryption (WE)

[1] We would like to point out that (unlike other works like [ADMM14b, BK14, ADMM14a, KB14]) *none* of our applications require the underlying blockchain protocol to provide a sufficiently expressive scripting language. This suggests that our applications could be based on top of almost all existing blockchain protocols.

[GGSW13, GKP+13] (which in turn requires strong knowledge assumptions related over multi-linear maps [GGH13a, CLT13], see also [GGHW14]). However, we stress that our construction does not require WE for all **NP**-relations, instead we only need a WE scheme for very specific blockchain dependent relations. As noted by prior works [Jag15, LKW15], we, for example, already know efficient WE schemes for hash proof system compatible relations [CS02, JR13, BBC+13, Wee10, ABP15] with some works even achieving certain notions of extractability.[2]

We also introduce the notion of pay-per-use programs. Informally, a pay-per-use program is a contract between two parties which we call the service provider and customer. A service provider wants to supply a program (or service) such that if the customer transfers a specific amount of coins to the provider (over the blockchain), it can evaluate the program on any input of its choice once. Additionally, the service provider need not be executing the blockchain protocol after supplying the program, i.e. it could go *offline*. We could also generalize this notion to k-time pay-per-use programs. This is naturally useful in a subscription based model where your payment is based on your usage. The above construction of one-time programs can be easily extended to obtain pay-per-use k-time programs.

1.1 Technical Overview

First, we discuss an abstract model for blockchain protocols as well as the protocol execution model and describe various desirable security properties of blockchains. Next, we outline our NIZK construction based on blockchains and present the main ideas in the security proof. We also overview our construction for OTPs using blockchains and highlight the necessity of a POS based blockchain in the security proof. Finally, we briefly discuss how to extend our idea behind constructing OTPs to building pay-per-use programs.

1.1.1 Proof-of-Stake Protocols: Abstraction and Properties

Informally, a blockchain protocol is a distributed consensus protocol in which each participant (locally) stores an ordered sequence of blocks/records $\mathbf{B}$ (simply called blockchain). The goal of all (honest) parties is to maintain a globally consistent blockchain. Each party can try to include new blocks in their local blockchain as well as attempt to get it added in blockchains of other parties. Such new blocks are created using a special block generation procedure (simply called mining) that depends on the underlying consensus mechanism.

In POS based blockchains, each participant (apart from storing a local blockchain $\mathbf{B}$) is also entitled with some *stake* in the system, which could be measured as a positive rational value.[3] The ideology behind mining in a POS based system is that the probability any party succeeds in generating the next block (i.e., gets to mine a new block) is proportional to its stake. Also, each party that generates a block must provide a *proof-of-stake* which could be used as a certificate by other parties to verify correctness. Such proofs-of-stake are usually provided in the form of signatures, as it prevents unforgeability and permits easy verification. An important aspect in such POS systems is that the stake distribution (among all parties) evolves over time, and is not necessarily static.

Recently, few works [GKL15, KP15, PSS16] initiated the study of formal analysis of blockchain protocols. They formalized and put forth some useful properties for blockchain protocols which were previously discussed only informally [Nak08, mtg10]. The most well-known properties analyzed are chain consistency and chain

[2] At first sight one might ask whether a strong assumption like extractable WE is necessary, or could it be relaxed. It turns out that, to construct one-time programs, it is *sufficient and necessary* to assume a slightly weaker primitive which we call *one-time extractable WE*. A one-time extractable WE is same as a standard extractable WE scheme, except the decryption algorithm could only be run once on each ciphertext. In other words, if we decrypt a one-time WE ciphertext with a bad witness the first time, then next time decryption (on that same ciphertext) will always fail even if we use a correct witness. Again this cannot be solely software based as then ciphertext could always be copied, and thus one-time decryption wouldn't make sense. It is straightforward to verify in our OTP construction that we could instead use such a one-time extractable WE scheme. Additionally, anologous to construction of extractable WE from VBB obfuscation, we could show that a OTP already implies a one-time extractable WE, therefore our assumption of one-time extractable WE for constructing OTPs is both necessary and sufficient.

[3] In cryptocurrencies, stake of any party simply corresponds to the amount of coins it controls.

quality.[4] At a high level, these can be described as follows.

- **ℓ-chain consistency:** blockchains of any two honest parties at any two (possibly different) rounds during protocol execution can differ only in the last ℓ blocks, with all but negligible probability.
- **(μ, ℓ)-chain quality:** fraction of blocks *mined by honest parties* in any sequence of ℓ or more consecutive blocks in an honest party's blockchain is at least μ, with all but negligible probability.

Previous works demonstrated usefulness of the above properties by showing that any blockchain protocol (irrespective of it being POW or POS based) satisfying these properties could be used a public ledger and for byzantine agreement. While the above properties are interesting from the perspective of using blockchains as an *end-goal* or achieving consensus, it is not clear whether these could be used to extract some form of cryptographic hardness. In other words, it does not seem straightforward on how to use these properties if we want to use blockchains as a primitive/enabler. To this end, we introduce several new security properties that are aimed directly at extracting cryptographic hardness from POS blockchains. We exhibit their importance and usability by basing security of all our applications (NIZKs, OTPs and pay-per-use programs) on these properties. At a high level, the properties could be described as follows.

- **(β, ℓ)-sufficient stake contribution:** the combined amount of stake whose proof was provided in any sequence of ℓ or more consecutive blocks in an honest party's blockchain is at least β fraction of the total stake in the system, with all but negligible probability.
- **(β, ℓ)-sufficient honest stake contribution:** the combined amount of *honestly held* stake whose proof was provided in any sequence of ℓ or more consecutive blocks in an honest party's blockchain is at least β fraction of the total stake in the system, with all but negligible probability.
- **(α, ℓ_1, ℓ_2)-bounded stake forking:** no adversary can create a *fork* of length $\ell_1 + \ell_2$ or more such that, in the last ℓ_2 blocks of the fork, the amount of proof-of-stake provided is more than α fraction of the total stake in the system, with all but negligible probability.[5]
- **$(\alpha, \beta, \ell_1, \ell_2)$-distinguishable forking:** with all but negligible probability, any sequence of $\ell_1 + \ell_2$ or more consecutive blocks in an honest party's blockchain could always be *distinguished* from any adversarially generated fork of same length by measuring the amount of proof-of-stake proven in those sequences. The fraction of proof-of-stake proven in the (adversarial) fork will be at most α, and in honest party's blockchain will be at least β. Hence, any fork which is created by the adversary on its own off-line is clearly distinguishable from a real blockchain.

Interestingly, we show that these properties with appropriate parameters are already implied (in an *almost black-box* way) by chain consistency and quality properties if we assume suitable stake distributions among honest parties. Since we already know of POS based blockchain protocols [KRDO16, BPS16b] that fit our abstract framework and satisfy chain consistency and quality, this provides concrete instantiations of our framework and following applications.

We would like to point out that, in our analysis, we make certain simplifying assumptions about the blockchain execution model. First, we require that the number of honest miners who actively participate in mining (i.e., are online) as well as the amount of stake they jointly control does not fall below a certain threshold. In other words, we expect that (honest) miners which control a significant amount of stake do not remain *offline* for arbitrarily long periods. However, we stress that we do not assume that *all* honest parties are online, nor do we assume that *all* honest parties which control a significant fraction of stake are online. We only require that the number of such honest parties does not fall below a reasonable threshold. Second, we also expect each honest party to delete the signing keys after they lose significance, i.e. once the coins associated with a particular key are transferred, then the corresponding signing key must be deleted.

[4] Previous works also define chain growth as a desideratum, however in this work we will only focus on chain consistency and quality properties.

[5] A *fork* is simply a *private* chain of blocks which significantly diverges from global blockchain in honest parties' view.

More details about our proposed properties as well as their reductions to other desideratum is provided later in Sections 4 and 5.

Now our applications give evidence that the above security properties as well as our POS framework are very useful in using POS based blockchains as a primitive, and we believe its scope is beyond this work as well. Also, we would like to point out that our reductions are not completely tight since we do not assume any special structure about underlying POS protocols, but instead work with an abstract model. We hope that future work on POS blockchains will consider these properties as desiderata, thereby proving these properties directly (possibly in a non-black-box way) with better parameters.

1.1.2 Zero-Knowledge Systems based on Blockchains

For ease of exposition, assume that all parties executing the blockchain protocol have the same amount of stake (i.e., each new block contains a proof-of-stake of a fixed amount). Also, the adversary controls only a *minority stake* in the system (say α). Below we describe a simplified construction. A formal treatment is given in the main body.

Defining non-interactive zero-knowledge based on blockchains: We would define the zero-knowledge property as follows. Very informally, we would require the existence of a simulator which should be able to simulate the view of the adversary without having access to the witness. In the real experiment, the adversary interacts with the honest parties: the honest prover, the honest miners and other honest blockchain participants. In the simulated experiment, the adversary interacts with the simulator alone. The simulator in turn emulates all the honest parties: including the honest prover, and the honest miners. We would require the view of the adversarial verifier to be computational indistinguishable in the two experiments. Note that in the simulated experiment, the simulator emulates (or controls) all the honest parties including even the honest blockchain miners. This can be seen as analogous to the simulator emulating the honest party publishing the CRS in the CRS model, or, the simulator controlling a majority of the parties in a secure multi-party computation protocol with honest majority [BGW88], etc.

First, we define the notion of a *fork* with respect to blockchains. Let $\mathbf{B}$ be some blockchain. A fork w.r.t. $\mathbf{B}$ is a sequence of valid blocks that extends some prefix of blockchain $\mathbf{B}$ instead of extending $\mathbf{B}$ directly from its end. In other words, a fork is a sequence of valid blocks that starts extending the chain at some block which is not the most recently added block in $\mathbf{B}$.

The starting point of our construction is the well-known FLS paradigm [FLS99] for transforming proof of the statement $x \in L$ into a witness-indistinguishable proof for the statement — "$x \in L$ OR the common shared random string σ is the output of a pseudorandom generator". Our idea is to use the already established blockchain $\mathbf{B}$ as the CRS σ, and instead of proving that σ is the output of a pseudorandom generator, we will prove some trapdoor information (which is hard to compute) w.r.t. to the current blockchain $\mathbf{B}$. A little more formally, we will generate a witness-indistinguishable proof for the statement — "$x \in L$ OR there exists a long valid fork f w.r.t. blockchain $\mathbf{B}$".

Suppose $\mathsf{Com}(\cdot)$ is a non-interactive statistically binding commitment scheme. Let $\mathbf{B}$ denote the current state of the blockchain and the adversary controls at most α fraction of total stake in the blockchain network. At a high level, the scheme works as follows. The prover constructs the NIZK as:

- Compute commitments $c_1 \leftarrow \mathsf{Com}(w)$ and $c_2 \leftarrow \mathsf{Com}(f)$ where w is the witness for the given statement $x \in L$, and f is simply an all zeros string of appropriate length.
- Compute a non-interactive witness indistinguishable (NIWI) argument using witness w proving that either:
  1. c_1 is a commitment of a valid witness to $x \in L$, or
  2. c_2 is a commitment of a long *fork* w.r.t. blockchain $\mathbf{B}$ (i.e., a different sequence of valid blocks) such that the amount of proof-of-stake present in the fork is a clear majority (of total stake).

Completeness follows directly from the correctness of underlying primitives. To prove the zero-knowledge property, we would need to construct a simulator which would not have the witness w but could still construct

proofs which are indistinguishable from honestly generated proofs. Note that the simulator is permitted to control all honest parties, thus it can access their signing keys. Since honest parties are in (stake) majority, therefore the simulator could efficiently generate a fork of sufficient length that contains a combined proof of majority stake. Hence, it could alternatively compute c_2 as a commitment to the fork, and generate the NIWI using the witness for second condition.

Proving soundness of the above construction is not straightforward and turns out to be more complex. Suppose that an adversary manages to produce a NIZK for a false statement. How could we reduce it to an attack on some reasonable notion of security of the blockchain? For such a reduction, we would have to construct an adversary which controls only a *minority stake* in the system, but it could still generate a fork which contains a proof of majority stake. However, the above NIZK only contains a commitment to such a fork. This problem seems to suggest that some form of extraction (of the fork) would be required for the security reduction to go through. *And yet, we don't have any CRS!* To solve this problem we need to modify our construction such that extraction is possible without any CRS.

Allowing Extraction of f. To this end, we rely on the following idea. Note that each mined block also contains the public key of the corresponding party. At a very high level, our idea is to secret share the fork into ℓ shares, and encrypt i^{th} share under public key of the party that mined i^{th} most recent block (instead of generating a commitment of the fork). If a certain threshold of these ℓ parties are honest, then we could extract the appropriate secret shares and reconstruct the fork.

More formally, let the public keys of the parties who mined at least one block in the last N blocks on blockchain $\mathbf{B}$ be $\mathsf{pk}_1, \ldots, \mathsf{pk}_\ell$ where N is a sufficiently big number and ℓ could be smaller than N (as some party could have mined multiple blocks). Note that in most blockchain protocols, each mined block contains the public (verification) key of its miner. We assume that these public keys could be used for encryption as well.[6] Also, recall that the fraction of total stake controlled by adversary is at most α, and for simplicity we assumed that all parties have the same amount of stake.

Now, the prover uses a $\beta\ell$-out-of-ℓ secret sharing scheme on f to get shares $f_1, \ldots, f_\ell$. For all i, the share f_i will be encrypted under pk_i, where β (< 1) is a scheme parameter such that it is sufficiently higher than α. The second condition (i.e., trapdoor condition) in the NIWI would now be that all these shares lead to a valid reconstruction of a string f which represents a long fork such that it contains a proof of majority stake w.r.t. blockchain $\mathbf{B}$. With this modification, we observe the following:

- Given any $\beta\ell$ secret keys corresponding to these public keys, f can be extracted. This is because the number of blocks a party mines is roughly proportional to its stake. Since we assume that all parties have same amount of stake, this implies that a set of miners controlling approximately β fraction of total stake can now extract f.

- Suppose an adversary is able to prove a false statement. As noted above, a set of miners controlling β fraction of total stake can perform the extraction. Also, these miners can emulate the adversary given more stake (as the adversary controls at most α of total stake), therefore for appropriate values of α and β, this would imply an algorithm using which a set of miners controlling only a minority amount of total stake could generate a sufficiently long fork containing a proof of majority stake. This would contradict the *bounded stake forking property* of the blockchain for suitable values of α, β and N.

- Further, this does not affect the zero-knowledge property since the amount of stake controlled by the adversary is significantly lower than β, therefore the adversary does not learn anything from the secret shares given to it. Also, the simulator, given signing keys of all honest parties (which control majority of stake), can still generate such a fork privately thereby using the fork instead of the actual witness to compute the NIWI.

Tha above construction could be naturally extended to be an *argument of knowledge* by additionally secret sharing the witness w analogous to the fork f. Note that in the above exposition we made a few simplifying assumptions. Thus the current construction does not work as is, and there are a number of

[6]For instance, most blockchain protocols (like Bitcoin, Ethereum etc.) already use ECDSA based signature schemes for which we could directly use ECIES-like integrated encryption schemes [Sho01].

issues which must be resolved. For example, we assumed that the stake distribution was uniform (i.e., all parties had identical stake). Since this may be arbitrary and not necessarily uniform, the idea of a threshold secret sharing does not work in general for extraction. Instead we need to use a weighted threshold secret sharing scheme with the weights being proportional to the respective stakes. Also, it is likely that different honest parties may have a different view of the last few blocks w.r.t. their local blockchains so we need to define the notion of forks with respect to the consistent part of the blockchain. It is also possible that some honest parties might have mined a few blocks in the adversary's fork before converging with other honest participants. To overcome such difficulties due to *small* forks (and other ephemeral consensus problems) in honest parties local blockchains, we need to make some more modifications like only considering the amount of proof-of-stake proven in the last few blocks of the fork etc. Finally, we directly reduce the security of our NIZKAoK construction to chain consistency, sufficient honest stake contribution, and bounded stake forking properties of the underlying blockchain protocols in our framework. More details are provided in Section 6.

Using POW based blockchains. We note that the above idea could potentially be ported to the POW based blockchains as well with the following caveat: the NIZK proof generated by the prover would be valid for a limited period of time. The main modification will be that now the prover simply proves that c_2 is a commitment to a very long fork instead. The rest of the construction would be mostly identical. However, the proof of security would now rely on the fact that any adversary which controls noticeably less than half of the computational resources can not compute a fork of length much longer than the honest parties blockchain. Intuitively, this can not happen because it would imply that any adversary with only minority voting power could fork the blockchain at any round. It is important to note that unlike the NIZKs based on POS blockchains, NIZKs based on POW blockchains will only be valid for atmost a bounded period of time as any verifier must reject such proofs once the length of its local blockchain is comparable to the length of the fork under c_2.

1.1.3 One-Time Programs using Blockchains

There are two main ideas behind constructing one-time programs (OTPs) using blockchains — (1) the blockchain could be used as a public *immutable* bulletin board, and (2) any adversarially generated fork can be distinguished from the real blockchain state. Informally, the scheme works as follows. To compile a circuit C over blockchain $\mathbf{B}$, the compilation algorithm first garbles the circuit to compute a garbled circuit and wire keys. Suppose we encrypt the wire keys using public key encryption and set the corresponding OTP as the garbled circuit and encrypted wire keys. This suggests that the evaluator must interact with the compiling party to be able to evaluate the program. Since OTPs are not defined in an interactive setting, we need to somehow allow conditional release/decryption of encrypted wire keys for evaluation. Additionally, we need to make sure that the evaluator only learns the wire keys corresponding to exactly *one* input as otherwise it will not satisfy the one-time secrecy condition. To this end, we encrypt the wire keys using witness encryption scheme. At a high level, an OTP for a circuit C is generated as follows:

- First, the circuit C is garbled to output a garbled circuit and corresponding input wire keys. Next, for each input wire, both wire keys are independently encrypted using a witness encryption (WE) scheme such that to decrypt the evaluator needs to produce a blockchain $\mathbf{B}'$ as a witness where $\mathbf{B}'$ must satisfy the following conditions — (1) there exists a block in $\mathbf{B}'$ which contains the input (on which evaluator wants to evaluate), and (2) $\mathbf{B}'$ contains a certain minimum number of blocks, say n, after the block containing input. The OTP for C will simply be this garbled circuit and all the encrypted wire keys.
- To execute the OTP, the evaluator chooses an input x and must commit it to the blockchain. Next, it must wait until its input x is added to the blockchain and is extended by n blocks. Let the resulting blockchain be $\widetilde{\mathbf{B}}$. The evaluator uses $\widetilde{\mathbf{B}}$ as the witness to decrypt the wire keys corresponding to the input x. In particular, for the i^{th} input wire, $\widetilde{\mathbf{B}}$ would serve as the witness to decrypt exactly one of the two wire keys depending upon the i^{th} bit of x. Finally, it could evaluate the garbled circuit using the decrypted wire keys.

There are various technical details we omit in the above sketch. For instance, the n blocks added after the input block must contain a minimum amount of combined proof-of-stake, as otherwise any adversary could simply generate such n blocks by itself. Also, the witness must be valid only if the user has committed to a *single* unique input x, as otherwise the user can commit to multiple inputs in the blockchain and be able to run the OTP on all of them. Mostly these could be dealt with by adding more checks on witness blockchain $\widetilde{\mathbf{B}}$ as part of the relation. Next, we briefly talk about the security.

Suppose that the adversarial user controls only minority stake. The security of this construction relies on the inability of the user to be able to extend the blockchain $\mathbf{B}$ by a sequence of n or more valid blocks without the support of honest parties. To execute this idea, we additionally check that the sequence of n blocks added after the input x contain a minimum amount of combined proof-of-stake. For simplicity, consider that we check whether the sequence of n blocks contain a proof of majority stake. Now the adversary will not be able to extend $\mathbf{B}$ on its own such that it satisfies this constraint. However, during honest execution, for sufficiently large values of n this will always hold. Therefore, the adversary's inability to fork directly reduces the security of the OTP to security of garbling scheme. To formally prove one-time secrecy of above construction, we reduce security of the above scheme to chain consistency and distinguishable forking properties of the underlying blockchain protocols in our framework. More details are provided in Section 7.

We would like to point out that this idea fails in POW based systems. This is because after receiving the OTP, the user can simply go offline and compute multiple forks of the chain starting from $\mathbf{B}$ such that each fork has a different user input. The user can compute such a fork on its own (albeit at a much slower rate compared to the growth of the original blockchain). Thus, unlike NIZKs, we do not know how to port the above idea to POW based blockchains.

Input Hiding. We would also like to note that in the above scheme, the evaluator needs to publicly broadcast its input x. This might not be suitable for applications of one-time programs which want the evaluator's input to be hidden. To this end, the scheme could be modified as follows. The evaluator adds to the blockchain a statistically binding commitment to its input (instead of its actual input x). Now the witness to decrypt the wire keys would also includes opening for the commitment and the witness relation verifies opening as well. We discuss additional such improvements later in Section 7.3.2.

Pay-per-use Programs. Lastly, the above construction of one-time programs can also be easily extended to obtain pay-per-use k-time programs. This can be done by requiring in the witness encryption relation that in the extension of the blockchain $\mathbf{B}$, apart from x, there is also an evidence of cryptocurrency transfer of some pre-specified amount to the service provider. This has been discussed in detail later in Section 8.

Comparison with related work. Recently it was shown that in [Jag15, LKW15] that Bitcoin could be combined with extractable witness encryption to build time-lock encryptions. Their idea was to exploit the fact that it should be hard for an adversary to generate blocks (i.e., extend blockchain) faster than the rest of the network. Very briefly, to encrypt data in their schemes, they encrypted it using WE under the current blockchain such that after say n more blocks have mined, those blocks could be used as a witness to decrypt the corresponding ciphertext. At a high level, they view mining of these n blocks as a proof of time being elapsed. At first sight, it might seem that our OTP construction is a straightforward combination of such time-lock encryptions with garbled circuits, this is not the case. We briefly highlight the important differences. First, time-lock encryptions used blockchain only as a counter/clock. On the other hand, we exploit the fact that blockchains could be used as an immutable public bulletin board. Concretely, in our construction, the evaluator needs to commit its input on the blockchain. Second, in our construction, it is essential that the underlying blockchain protocol is POS based, whereas [Jag15, LKW15] built schemes directy on top of Bitcoin. Lastly, we reduce the security of our construction to fundamental properties over blockchains and give examples of blockchain protocols for which those properties are satisfied, whereas [Jag15, LKW15] only gave an ad hoc analysis arguing that Bitcoin could be used implement such reference clocks.

1.2 Organization

We first provide some background on blockchain protocols in Section 2 and introduce some preliminaries in Section 3. Next, we provide necessary abstractions and useful security properties for POS based protocols in Section 4. In Section 5, we prove our security properties for blockchain protocols assuming only some fundamental properties. We also talk about two POS based protocols which could be used to instantiate our framework. Next we present our NIZKAoK construction in Section 6. And, in Sections 7 and 8, we present our constructions for OTPs and pay-per-use programs, respectively.

2 Background on Blockchain Protocols

In this section, we present an abstract model for blockchain protocols as well as the protocol execution model. Our model is an extension of the model used by Pass et al. [PSS16], which in turn is an extension of [GKL15].

2.1 Blockchain Protocols

A blockchain protocol Γ consists of 3 polynomial-time algorithms ($\mathsf{UpdateState}$, $\mathsf{GetRecords}$, $\mathsf{Broadcast}$) with the following syntax.

- $\mathsf{UpdateState}(1^\lambda)$: It is a stateful algorithm that takes as input the security parameter λ, and maintains a local state st.[7]
- $\mathsf{GetRecords}(1^\lambda, \mathsf{st})$: 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 $\mathbf{m}$.[8]
- $\mathsf{Broadcast}(1^\lambda, m)$: 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.

As in [GKL15, PSS16], the blockchain protocol is also parameterized by a validity predicate V that captures semantics of any particular blockchain application. The validity predicate takes as input a sequence of blocks $\mathbf{B}$ and outputs a bit, where 1 certifies validity of blockchain $\mathbf{B}$ and 0 its invalidity.[9]

In a blockchain protocol, the goal of all parties is to maintain a (consistent) global ordered set of records. The above model tries to abstractly capture the necessary components that any blockchain protocol possesses. However in this paper, we will mostly be interested in the blockchain-based cryptocurrencies like Bitcoin [Nak08], Ethereum [Woo14] and their derivatives. Such blockchain protocols support special types of messages which we call currency transaction messages. Each currency transaction message is associated with two parties — sender and receiver. It is used to transfer money (coins) from a sender to receiver given that a sender has sufficient funds (and can prove their ownership). We consider that each party has at least one *unique* public identity.

Let $\mathcal{A}$ and $\mathcal{B}$ be two parties with public identities $\mathsf{id}_\mathcal{A}$ and $\mathsf{id}_\mathcal{B}$. We will use the message $(\mathsf{id}_\mathcal{A}, \mathsf{id}_\mathcal{B}, q, \mathsf{aux})$ to denote a transaction of q coins from $\mathcal{A}$ to $\mathcal{B}$ with some (possibly empty) auxiliary information aux. Such a message is accompanied with a proof of identity and sufficiency of funds by $\mathcal{A}$ to verify validity, however we choose to avoid it in our abstraction. In Bitcoin as well as most of its derivatives, the identities are cryptographic hashes of a public verification key, and a proof of identity is a signature on the entire message.

[7] The local state should be considered as the entire blockchain (i.e., sequence of mined blocks along with metadata) in Bitcoin and other cryptocurrencies.

[8] The sequence $\mathbf{B}$ should be considered as the entire transaction history in Bitcoin and other cryptocurrencies, where the blocks are ordered in the sequence they were mined.

[9] The validity predicate could be used to capture various fundamental properties. E.g., In Bitcoin and other cryptocurrencies, it could be used to check for double spending, correct mining etc.

To prove sufficiency of funds, Bitcoin requires sender $\mathcal{A}$ to provide prior unspent transaction(s) with $\mathcal{A}$ as receiver, whereas currencies like Ethereum only require $\mathcal{A}$ to have sufficient balance where balance is computed as the difference between total number of coins addressed to $\mathcal{A}$ and number of coins spent by $\mathcal{A}$.

In most cryptocurrencies instantiated in the blockchain model, there is a special *mining* procedure which is used to extend the blockchain (i.e., add new blocks) and in turn rewards the miner who successfully generates a new valid block with freshly generated currency. The mining procedure simulates a puzzle-solving race between protocol participants and could be run by any party executing the blockchain protocol. Presently, different cryptocurrencies employ different types of puzzles for mining (i.e., achieving consensus on next block). These could be classified into two broad categories — Proof-of-Work (POW) and Proof-of-Stake (POS) based puzzles. The basic difference between these is that in POW puzzles the probability of successful mining is proportional to the amount of computational power, whereas in POS it is proportional to the balance of miner. Therefore, the POW miners have to spend significant portion of their computational resources (and in turn, monetary resources/*budget*) to extend the blockchain and in turn get rewarded, whereas POS miners spend significantly less computational resources and earn rewards at a rate proportional to their stake in the system.

We do not provide an explicit procedure for mining in the above abstraction as it is not important for our applications. In Section 4, we will discuss some important properties and assumptions on the blockchain protocol. Next we briefly describe our protocol execution model.

2.2 Blockchain Execution

At a very high level, the protocol execution proceeds as follows. Each participant in the protocol runs the $\mathsf{UpdateState}$ algorithm to keep track of the current (latest) blockchain state. This corresponds to listening on the broadcast network for messages from other nodes. The $\mathsf{GetRecords}$ algorithm is used to extract an ordered sequence of blocks encoded in the blockchain state variable, which is considered as the common public ledger among all the nodes. And, the $\mathsf{Broadcast}$ algorithm is used by a party when it wants to post a new message on the blockchain. In cryptocurrencies, this corresponds to the scenario when either a miner wants to broadcast a successfully mined block, or a transaction between nodes. Note that the messages are accepted by the blockchain protocol only if they satisfy the validity predicate given the current state, i.e. sequence of blocks.

Following prior works [GKL15, KP15, PSS16], we define the protocol execution in the Universal Composability framework of [Can01]. For any blockchain protocol $\Gamma^V = (\mathsf{UpdateState}^V, \mathsf{GetRecords}, \mathsf{Broadcast})$, the protocol execution is directed by the environment $\mathcal{Z}(1^\lambda)$ where λ is the security parameter. The environment $\mathcal{Z}$ activates the parties as either *honest* or *corrupt*, and is also responsible for providing inputs/ records to all parties in each round. All the corrupt parties are controlled by the adversary $\mathcal{A}$. The adversary is also responsible for delivery of all network messages. Honest parties start by executing $\mathsf{UpdateState}^V$ on input 1^λ with an empty local state $\mathsf{st} = \epsilon$. The execution proceeds as follows and the following description is mostly taken verbatim from [PSS16] with appropriate modification.

- The execution proceeds in rounds that model time steps. In round r, each honest player 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 $\mathsf{Broadcast}$ algorithm) to all other players (which will be delivered by the adversary; see below) and update its local state st_i. It could also attempt to "add" a new block to its chain i.e., run the mining proedure.

- $\mathcal{A}$ is responsible for delivering all messages sent by parties (honest or corrupted) to all other parties. $\mathcal{A}$ 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. The identity of the sender is not known to the recipient.

- At any point, $\mathcal{Z}$ can communicate with adversary $\mathcal{A}$ or access $\mathsf{GetRecords}(\mathsf{st}_i)$ where st_i is the local state of player i.

We would like to point out that we only consider *static* corruptions. The scenario of adaptive corruptions *with erasures* has been considered in prior works [GKL15, PSS16], and we believe all our assumptions and analysis over blockchain protocols holds in that case as well. However, for simplicity of analysis we only consider static corruptions. More details about the blockchain protocol properties and assumptions are provided in Sections 4 and 5. We would like to note that it is an interesting open problem to extend our analysis in presence of adaptive corruptions *without erasures*.

Admissible environment and compliant executions. In this work, we do not constrain ourselves to any specific blockchain protocol. Instead we show how to use any blockchain protocol that provides certain security guarantees. We also point out two proposed POS protocols [KRDO16, BPS16b] for which we show that all required security properties hold.

Prior works have defined compliant executions for their blockchain protocols by specifying a set of constraints on the PPT pair $(\mathcal{A}, \mathcal{Z})$. They show that certain desirable security properties are respected except with negligible probability in any compliant execution. In this work, we prove additional security properties in an (almost) black-box way for their protocols assuming only the basic properties which they satisfy. Therefore, we do not provide explicit descriptions of the compliant executions which they consider. However, it is important to note that our theorems also hold only in any compliant execution expected by the underlying blockchain protocol.

3 Preliminaries

Notations. We will use bold letters for vectors (e.g., $\mathbf{v}$). For any finite set S, $x \leftarrow S$ denotes a uniformly random element x from the set S. Similarly, for any distribution $\mathcal{D}$, $x \leftarrow \mathcal{D}$ denotes an element x drawn from distribution $\mathcal{D}$.

Let $\mathsf{EXEC}^{\Gamma^V}(\mathcal{A}(x), \mathcal{Z}, 1^\lambda)$ be the random variable denoting the joint view of all parties in the execution of protocol Γ^V with adversary $\mathcal{A}$ and environment $\mathcal{Z}$ where $\mathcal{A}$ is given an additional private input x. This joint view fully determines the execution. Also, let $\mathsf{view}_{\mathcal{A}}(\mathsf{EXEC}^{\Gamma^V}(\mathcal{A}(x), \mathcal{Z}, 1^\lambda))$ denote the view of adversary $\mathcal{A}$ in the protocol execution.

3.1 Public Key Integrated Encryption-Signature Scheme

First, we define an integrated scheme which works both as a public key encryption scheme as well as public key signature scheme. Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be the message spaces for encryption and signature scheme respectively. A public key integrated encryption-signature scheme HS for message spaces $\mathcal{M}_1$ and $\mathcal{M}_2$ consists of following polynomial-time algorithms.

- $\mathsf{Setup}(1^\lambda)$: The setup algorithm takes as input the security parameter λ, and outputs a master public-secret key pair $(\mathsf{mpk}, \mathsf{msk})$.
- $\mathsf{Enc}(\mathsf{mpk}, m \in \mathcal{M}_1)$: The encryption algorithm takes as input master public key mpk and a message m, and outputs a ciphertext ct.
- $\mathsf{Dec}(\mathsf{msk}, \mathsf{ct})$: The decryption algorithm takes as input master secret key msk and a ciphertext ct, and outputs a message m.
- $\mathsf{Sign}(\mathsf{msk}, m \in \mathcal{M}_2)$: The signing algorithm takes as input master secret key msk and a message m, and outputs a signature σ.
- $\mathsf{Verify}(\mathsf{mpk}, m \in \mathcal{M}_2, \sigma)$: The verification algorithm takes as input master public key mpk, a message m and a signature σ, and outputs a bit.

Correctness. An integrated scheme HS for message spaces $\mathcal{M}_1, \mathcal{M}_2$ is said to be correct if for all λ, $m_1 \in \mathcal{M}_1$, $m_2 \in \mathcal{M}_2$ and $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$, we have that $\mathsf{Dec}(\mathsf{msk}, \mathsf{Enc}(\mathsf{mpk}, m_1)) = m_1$ and $\mathsf{Verify}(\mathsf{mpk}, m_2, \mathsf{Sign}(\mathsf{msk}, m_2)) = 1$.

Security. Informally, an integrated encryption-signature scheme is said to be secure if it is both an unforgeable signature scheme as well as an IND-CPA secure public key encryption scheme. More formally,

Definition 3.1. A public key integrated encryption-signature scheme $\mathsf{HS} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Sign}, \mathsf{Verify})$ is a secure integrated scheme if for every PPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$ there exists a negligible functions $\mathrm{negl}_1(\cdot), \mathrm{negl}_2(\cdot)$, such that for all $\lambda \in \mathbb{N}$, the following holds:

$$\left| \Pr \left[\mathcal{A}_1(\mathsf{ct}, \mathsf{st}) = b \middle| \begin{array}{c} (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda); b \leftarrow \{0,1\} \\ (m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_0^{\mathsf{Sign}(\mathsf{msk}, \cdot)}(\mathsf{mpk}); \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, m_b) \end{array} \right] - \frac{1}{2} \right| \leq \mathrm{negl}_1(\lambda),$$

and

$$\Pr \left[\mathsf{Verify}(\mathsf{msk}, m^*, \sigma^*) = 1 \middle| \begin{array}{c} (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ (m^*, \sigma^*) \leftarrow \mathcal{A}_2^{\mathsf{Sign}(\mathsf{msk}, \cdot)}(\mathsf{mpk}) \end{array} \right] \leq \mathrm{negl}_2(\lambda),$$

where $\mathcal{A}_2$ must never have queried m^* to signing oracle.

While such an integrated scheme could always be generically constructed from any IND-CPA secure public key encryption scheme and any EUF-CMA secure public key signature scheme, we hope that the signature schemes used in current blockchain protocols could be used as integrated encryption-signature schemes as well. For instance, most blockchain protocols (like Bitcoin, Ethereum etc.) already use ECDSA based signature schemes for which we could directly use ECIES-like integrated encryption schemes [Sho01]. However this will be a slightly stronger assumption.

3.2 Weighted Threshold Secret Sharing Schemes

A weighted threshold secret sharing scheme consists of a pair of probabilistic algorithms $(\mathsf{Share}, \mathsf{Rec})$ with following syntax.

- $\mathsf{Share}(s, \{w_i\}_{i \leq n} \in (\mathbb{Q}^+)^n, W \in \mathbb{Q}^+)$: The sharing algorithm takes as input a secret s, a set of weights $\{w_i\}_{i \leq n}$ and a threshold W. It outputs a sequences of n shares $\{\mathsf{sh}_i\}_i$.
- $\mathsf{Rec}(\{\mathsf{sh}_i\}_{i \leq m})$: The reconstruction algorithm takes as input a set of shares $\{\mathsf{sh}_i\}_{i \leq m}$, and either outputs $\perp$ or some string t.

Correctness. A weighted threshold secret sharing scheme SS is said to be correct if for all $n \in \mathbb{N}$, $\{w_i\}_{i \leq n} \in (\mathbb{Q}^+)^n$, $W \in \mathbb{Q}^+$, s, it holds that for all $\mathcal{I} \subseteq \{1, \ldots, n\}$,

$$\Pr \left[\mathsf{Rec}(\{\mathsf{sh}_i\}_{i \in \mathcal{I}}) = s \;\middle|\; \{\mathsf{sh}_i\}_i \leftarrow \mathsf{Share}(s, \{w_i\}_{i \leq n}, W) \bigwedge \sum_{i \in \mathcal{I}} w_i \geq W \right] = 1.$$

Secrecy. Informally, a weighted threshold secret sharing scheme satisfies secrecy if any group of colluding parties can not learn the secret if the total combined weight of their shares is less than threshold.

Definition 3.2. A weighted threshold secret sharing scheme $\mathsf{SS} = (\mathsf{Share}, \mathsf{Rec})$ satisfies secrecy if for every two secrets s_1, s_2, every share size n, all weight distributions $\{w_i\}_{i \leq n}$, threshold W and subset $\mathcal{I} \subseteq \{1, \ldots, n\}$, if $\sum_{i \in \mathcal{I}} w_i < W$, then the following holds:

$$\left\{ \{\mathsf{sh}_i\}_{i \in \mathcal{I}} : \{\mathsf{sh}_i\}_i \leftarrow \mathsf{Share}(s_1, \{w_i\}_{i \leq n}, W) \right\}$$
$$\approx_c$$
$$\left\{ \{\mathsf{sh}_i\}_{i \in \mathcal{I}} : \{\mathsf{sh}_i\}_i \leftarrow \mathsf{Share}(s_2, \{w_i\}_{i \leq n}, W) \right\}.$$

3.3 Non-Interactive Argument Systems

3.3.1 Non-Interactive Witness Indistinguishable Arguments

Witness indistinguishable (WI) proofs were introduced by Feige and Shamir [FS90] as a natural weakening of zero-knowledge (ZK) proofs. At a high level, the witness indistinguishability property says that a proof must not reveal the witness used to prove the underlying statement even if it reveals all possible witnesses corresponding to the statement. Unlike ZK proofs, WI proofs without interaction in the standard model are known to be possible. Barak, Ong and Vadhan [BOV07] provided constructions for one-message (completely non-interactive, with no shared random string or setup assumptions) witness indistinguishable proofs (NIWIs) based on ZAPs (i.e., two-message public-coin witness indistinguishable proofs) and Nisan-Wigderson type pseudorandom generators [NW94]. Groth, Ostrovsky and Sahai [GOS12] gave the first NIWI construction from a standard cryptographic assumption, namely the decision linear assumption. Recently, Bitansky and Paneth [BP15] constructed NIWI proofs assuming iO and one-way permutations.

Definition 3.3. (NIWI) A pair of PPT algorithms $(\mathcal{P}, \mathcal{V})$ is a NIWI argument system for a language $\mathcal{L} \in \mathbf{NP}$ with witness relation $\mathcal{R}$ if it satisfies the following conditions:

- (Completeness) For all (x, w) such that $\mathcal{R}(x, w) = 1$, there exists a negligible function $\text{negl}(\cdot)$ such that

$$\Pr[\mathcal{V}(x, \pi) = 1 : \pi \leftarrow \mathcal{P}(x, w)] = 1 - \text{negl}(|x|).$$

- (Soundness) For every $x \notin \mathcal{L}$ and all PPT adversaries $\mathcal{A}$, there exists a negligible function $\text{negl}(\cdot)$ such that

$$\Pr[\mathcal{V}(x, \pi) = 1 : \pi \leftarrow \mathcal{A}(x)] \leq \text{negl}(|x|).$$

- (Witness Indistinguishability) For any sequence $\mathcal{I} = \{(x, w_1, w_2) : \mathcal{R}(x, w_1) = 1 \wedge \mathcal{R}(x, w_2) = 1\}$

$$\{\pi_1 : \pi_1 \leftarrow \mathcal{P}(x, w_1)\}_{(x,w_1,w_2)\in\mathcal{I}} \approx_c \{\pi_2 : \pi_2 \leftarrow \mathcal{P}(x, w_2)\}_{(x,w_1,w_2)\in\mathcal{I}}$$

3.3.2 Non-Interactive Zero Knowledge Arguments

The notion of Zero Knowledge for interactive protocols was introduced by Goldwasser, Micali and Rackoff [GMR89]. A non-interactive zero knowledge argument system is a one-message ZK protocol. However, it is well known that NIZKs are impossible in the standard model [GO94]. They are usually defined with trusted setup. The formal definition follows.

Definition 3.4. (NIZK with CRS) A pair of PPT algorithms $(\mathcal{K}, \mathcal{P}, \mathcal{V})$ is a NIZK argument of knowledge for a language $\mathcal{L} \in \mathbf{NP}$ with witness relation $\mathcal{R}$ if it satisfies the following conditions:

- (Completeness) For all (x, w) such that $\mathcal{R}(x, w) = 1$, there exists a negligible function $\text{negl}(\cdot)$ such that

$$\Pr[\mathcal{V}(\mathsf{crs}, x, \pi) = 1 : \mathsf{crs} \leftarrow \mathcal{K}(1^\lambda);\ \pi \leftarrow \mathcal{P}(\mathsf{crs}, x, w)] \geq 1 - \text{negl}(\lambda).$$

- (Soundness) For every $x \notin \mathcal{L}$ and all PPT adversaries $\mathcal{A}$, there exists a negligible function $\text{negl}(\cdot)$ such that

$$\Pr[\mathcal{V}(\mathsf{crs}, x, \pi) = 1 : \mathsf{crs} \leftarrow \mathcal{K}(1^\lambda);\ \pi \leftarrow \mathcal{A}(\mathsf{crs}, x)] \leq \text{negl}(\lambda).$$

- (Knowledge Extractor) There is a PPT algorithm $\mathcal{E} = (\mathcal{E}_1, \mathcal{E}_2)$, such that for all PPT adversaries $\mathcal{A}$, there exists a negligible function $\text{negl}(\cdot)$ such that

$$\Pr[\mathcal{A}(\mathsf{crs}) = 1 : \mathsf{crs} \leftarrow \mathcal{K}(1^\lambda)] = \Pr[\mathcal{A}(\mathsf{crs}) = 1 : (\mathsf{crs}, \tau) \leftarrow \mathcal{E}_1(1^\lambda)],$$

and

$$\Pr\left[\mathcal{V}(\mathsf{crs}, x, \pi) = 0 \vee \mathcal{R}(x, w) = 1 : \begin{array}{c}(\mathsf{crs}, \tau) \leftarrow \mathcal{E}_1(1^\lambda);\ (x, \pi) \leftarrow \mathcal{A}(\mathsf{crs}) \\ w \leftarrow \mathcal{E}_2(\mathsf{crs}, \tau, x, \pi)\end{array}\right] \geq 1 - \text{negl}(\lambda).$$

- (Zero Knowledge) There is a PPT algorithm Sim for the argument system such that for (x, w) subject to $\mathcal{R}(x, w) = 1$, the following holds

$$\left\{(\mathsf{crs}, \pi) : \mathsf{crs} \leftarrow \mathcal{K}(1^\lambda);\ \pi \leftarrow \mathcal{P}(x, w)\right\} \approx_c \left\{\mathsf{Sim}(x)\right\}$$

NIZKs are a well studied and fundamental cryptographic primitive. There are numerous constructions with different setup assumptions. In this work, we construct NIZKs over blockchain protocols without any additional setup assumption. Below we provide the formal definition.

Definition 3.5. (NIZK over Blockchains) A pair of PPT algorithms $(\mathcal{P}, \mathcal{V})$ over a blockchain protocol Γ^V is a NIZK argument of knowledge for a language $\mathcal{L} \in \mathbf{NP}$ with witness relation $\mathcal{R}$ if it satisfies the following conditions:

- (Completeness) For all (x, w) such that $\mathcal{R}(x, w) = 1$, all PPT adversaries $\mathcal{A}$ and players i, j in environment $\mathcal{Z}$, there exists negligible functions $\mathrm{negl}_1(\cdot), \mathrm{negl}_2(\cdot)$ such that

$$\Pr\left[\mathcal{V}(\widetilde{\mathbf{B}}, x, \pi) = 1 : \begin{array}{c} \mathsf{view} \leftarrow \mathsf{EXEC}^{\Gamma^V}\left(\mathcal{A}, \mathcal{Z}, 1^\lambda\right) \\ \mathbf{B} = \mathsf{GetRecords}(\mathsf{view}_i);\ \widetilde{\mathbf{B}} = \mathsf{GetRecords}(\mathsf{view}_j) \\ \pi \leftarrow \mathcal{P}(\mathbf{B}, x, w) \end{array}\right] \geq 1 - \mathrm{negl}_1(|x|) - \mathrm{negl}_2(\lambda),$$

  where view_i and view_j denote the view of players i and j, and both i, j are honest.[10]

- (Soundness) For every $x \notin \mathcal{L}$ and all stateful PPT adversaries $\mathcal{A}$ and each player i in environment $\mathcal{Z}$, there exists negligible functions $\mathrm{negl}_1(\cdot), \mathrm{negl}_2(\cdot)$ such that

$$\Pr\left[\mathcal{V}(\mathbf{B}, x, \pi) = 1 : \begin{array}{c} \mathsf{view} \leftarrow \mathsf{EXEC}^{\Gamma^V}\left(\mathcal{A}(x), \mathcal{Z}, 1^\lambda\right) \\ \mathbf{B} = \mathsf{GetRecords}(\mathsf{view}_i);\ \pi \leftarrow \mathcal{A} \end{array}\right] \leq \mathrm{negl}_1(|x|) + \mathrm{negl}_2(\lambda),$$

  where view_i denotes the view of player i, and i is honest.

- (Knowledge Extractor) There is a stateful PPT algorithm $\mathcal{E}$, such that for all stateful PPT adversaries $\mathcal{A}$ and each player i in environment $\mathcal{Z}$, there exists negligible functions $\mathrm{negl}_1(\cdot), \mathrm{negl}_2(\cdot)$ such that

$$\left\{\mathsf{view}_{\mathcal{A}}\left(\mathsf{EXEC}^{\Gamma^V}\left(\mathcal{A}, \mathcal{Z}, 1^\lambda\right)\right)\right\} \approx_c \left\{\mathsf{view}_{\mathcal{A}}\left(\mathsf{EXEC}^{\Gamma^V}\left(\mathcal{A}, \mathcal{E}, \mathcal{Z}, 1^\lambda\right)\right)\right\}$$

  and

$$\Pr\left[\begin{array}{l} \mathcal{V}(\mathbf{B}, x, \pi) = 0 \\ \vee\ \mathcal{R}(x, w) = 1 \end{array} : \begin{array}{c} \mathsf{view} \leftarrow \mathsf{EXEC}^{\Gamma^V}\left(\mathcal{A}, \mathcal{E}, \mathcal{Z}, 1^\lambda\right);\ (x, \pi) \leftarrow \mathcal{A} \\ \mathbf{B} = \mathsf{GetRecords}(\mathsf{view}_i);\ w \leftarrow \mathcal{E}(x, \pi) \end{array}\right] \geq 1 - \mathrm{negl}_1(|x|) - \mathrm{negl}_2(\lambda),$$

  where view_i denotes the view of player i and i is honest, and $\mathsf{EXEC}^{\Gamma^V}\left(\mathcal{A}, \mathcal{E}, \mathcal{Z}, 1^\lambda\right)$ is the random variable denoting the joint view of all parties in the blockchain execution where adversary $\mathcal{A}$ controls all the corrupt parties, and $\mathcal{E}$ controls all the honest parties.

- (Zero Knowledge) There is a stateful PPT algorithm Sim for the argument system such that for all (x, w) subject to $\mathcal{R}(x, w) = 1$ and all stateful PPT adversaries $\mathcal{A}$ and each player i in environment $\mathcal{Z}$, the following holds

$$\left\{(\pi, \mathsf{view}_{\mathcal{A}}) \ :\ \begin{array}{c} \mathsf{view} \leftarrow \mathsf{EXEC}^{\Gamma^V}\left(\mathcal{A}, \mathsf{Sim}, \mathcal{Z}, 1^\lambda\right) \\ \pi \leftarrow \mathsf{Sim}(x) \end{array}\right\}$$

$$\approx_c$$

$$\left\{(\pi, \mathsf{view}_{\mathcal{A}}) \ :\ \begin{array}{c} \mathsf{view} \leftarrow \mathsf{EXEC}^{\Gamma^V}\left(\mathcal{A}, \mathcal{Z}, 1^\lambda\right) \\ \mathbf{B} = \mathsf{GetRecords}(\mathsf{view}_i);\ \pi \leftarrow \mathcal{P}(\mathbf{B}, x, w) \end{array}\right\}$$

[10] We have overloaded the notation by using $\mathsf{GetRecords}$ algorithm to take as input the view of a party instead of its state. This is still well defined since the state of any party is part of its view.

where view_i denotes the view of player i and i is honest, and $\mathsf{EXEC}^{\Gamma^V}\left(\mathcal{A}, \mathsf{Sim}, \mathcal{Z}, 1^\lambda\right)$ is the random variable denoting the joint view of all parties in the blockchain execution where adversary $\mathcal{A}$ controls all the corrupt parties, and Sim controls all the honest parties.

3.4 Garbled Circuits

Our definition of garbled circuits [Yao86] is based upon the work of Bellare et al. [BHR12b]. Let $\{\mathcal{C}_n\}_n$ be a family of circuits where each circuit in $\mathcal{C}_n$ takes n bit inputs. A garbling scheme GC for circuit family $\{\mathcal{C}_n\}_n$ consists of polynomial-time algorithms Garble and Eval with the following syntax.

- $\mathsf{Garble}(1^\lambda, C \in \mathcal{C}_n)$: The garbling algorithm takes as input the security parameter λ and a circuit $C \in \mathcal{C}_n$. It outputs a garbled circuit G, together with $2n$ wire keys $\{w_{i,b}\}_{i\leq n, b\in\{0,1\}}$.
- $\mathsf{Eval}(G, \{w_i\}_{i\leq n})$: The evaluation algorithm takes as input a garbled circuit G and n wire keys $\{w_i\}_{i\leq n}$ and outputs $y \in \{0,1\}$.

Correctness. A garbling scheme GC for circuit family $\{\mathcal{C}_n\}_n$ is said to be correct if for all λ, n, $x \in \{0,1\}^n$ and $C \in \mathcal{C}_n$, $\mathsf{Eval}(G, \{w_{i,x_i}\}_{i\leq n}) = C(x)$, where $(G, \{w_{i,b}\}_{i\leq n, b\in\{0,1\}}) \leftarrow \mathsf{Garble}(1^\lambda, C)$.

Security. Informally, a garbling scheme is said to be secure if for every circuit C and input x, the garbled circuit G together with input wires $\{w_{i,x_i}\}_{i\leq n}$ corresponding to some input x reveals only the output of the circuit $C(x)$, and nothing else about the circuit C or input x.

Definition 3.6. A garbling scheme $\mathsf{GC} = (\mathsf{Garble}, \mathsf{Eval})$ for a class of circuits $\mathcal{C} = \{\mathcal{C}_n\}_n$ is said to be a selectively secure garbling scheme if there exists a polynomial-time simulator Sim such that for all λ, n, $C \in \mathcal{C}_n$ and $x \in \{0,1\}^n$, the following holds:

$$\left\{\mathsf{Sim}\left(1^\lambda, 1^n, 1^{|C|}, C(x)\right)\right\} \approx_c \left\{(G, \{w_{i,x_i}\}_{i\leq n}) \;:\; \left(G, \{w_{i,b}\}_{i\leq n, b\in\{0,1\}}\right) \leftarrow \mathsf{Garble}(1^\lambda, C)\right\}.$$

While this definition is not as general as the definition in [BHR12b], it suffices for our construction.

3.5 Witness Encryption

The notion of witness encryption was introduced by Garg et al. [GGSW13]. The following description is provided as it appears in prior work [GGHW14].

Let $\mathcal{L}$ be an **NP** language with witness relation $\mathcal{R}$. A witness encryption scheme WE for language $\mathcal{L}$ with message space $\mathcal{M} \subseteq \{0,1\}^*$ consists of polynomial-time algorithms Enc and Dec with following syntax.

- $\mathsf{Enc}(1^\lambda, x \in \{0,1\}^*, m \in \mathcal{M})$: The encryption algorithm takes as input the security parameter λ, an unbounded length string x and a message m. It outputs a ciphertext ct.
- $\mathsf{Dec}(\mathsf{ct}, w \in \{0,1\}^*)$: The decryption algorithm takes as input a ciphertext ct and an unbounded length string w. It either outputs a message $m \in \mathcal{M}$ or a special symbol $\perp$.

Correctness. A witness encryption scheme WE for language $\mathcal{L}$ is said to be correct if for all λ, $(x, w) \in \mathcal{R}$, $m \in \mathcal{M}$, $\mathsf{Dec}(\mathsf{ct}, w) = m$, where $\mathsf{ct} \leftarrow \mathsf{Enc}(1^\lambda, x, m)$.

Security. In this work we will require the witness encryption scheme to provide extractable security. Informally, such a witness encryption scheme is said to be secure if an adversary can learn some non-trivial information about the encrypted message only if it knows a *witness* for the instance used during encryption. More formally,

Definition 3.7. A witness encryption scheme $\mathsf{WE} = (\mathsf{Enc}, \mathsf{Dec})$ for a language $\mathcal{L}$ with witness relation $\mathcal{R}$ is extractable secure if for every security parameter λ, every PPT adversary $\mathcal{A}$ and polynomial $p(\cdot)$, there exists a PPT extractor E and polynomial $q(\cdot)$ such that for every pair of messages $m_0, m_1 \in \mathcal{M}$, string $x \in \{0,1\}^*$,

$$\Pr\left[\mathcal{A}(1^\lambda, \mathsf{ct}) = b \;\middle|\; b \leftarrow \{0,1\}; \mathsf{ct} \leftarrow \mathsf{Enc}(1^\lambda, x, m_b)\right] \geq \frac{1}{2} + \frac{1}{p(\lambda)}$$
$$\implies \Pr\left[(x,w) \in \mathcal{R} \;\middle|\; w \leftarrow E(1^\lambda, x, m_0, m_1)\right] \geq \frac{1}{q(\lambda)}.$$

3.6 One-Time Programs and Compilers

The notion of one-time programs was introduced by Goldwasser et al. [GKR08]. Let $\{\mathcal{C}_n\}_n$ be a family of circuits where each circuit in $\mathcal{C}_n$ takes n bit inputs. A one-time compiler OTC for circuit family $\{\mathcal{C}_n\}_n$ consists of polynomial-time algorithms $\mathsf{Compile}$ and Eval with the following syntax.

- $\mathsf{Compile}(1^\lambda, C \in \mathcal{C}_n)$: The compilation algorithm takes as input the security parameter λ and a circuit $C \in \mathcal{C}_n$. It outputs a compiled circuit CC.
- $\mathsf{Eval}(CC, x \in \{0,1\}^n)$: The evaluation algorithm takes as input a compiled circuit CC and an n-bit input x, and outputs $y \in \{0,1\} \cup \perp$.

Correctness. A one-time compiler OTC for circuit family $\{\mathcal{C}_n\}_n$ is said to be correct if for all λ, n, $x \in \{0,1\}^n$ and $C \in \mathcal{C}_n$,

$$\Pr[\mathsf{Eval}(CC, x) = C(x) \mid CC \leftarrow \mathsf{Compile}(1^\lambda, C)] \geq 1 - \mathrm{negl}(\lambda),$$

where evaluation is run only once, and $\mathrm{negl}(\cdot)$ is a negligible function.

One-Time Secrecy. Traditionally, security for one-time compilers have been defined in presence of secure hardware or memory devices. The following definition is based upon the works of [GKR08, GIS+10].

Definition 3.8. A one-time compiler $\mathsf{OTC} = (\mathsf{Compile}, \mathsf{Eval})$ for a class of circuits $\mathcal{C} = \{\mathcal{C}_n\}_n$ is said to be a H/S-secure one-time compiler if for every PPT adversary $\mathcal{A}$, there exists a PPT simulator Sim such that for all λ, n, $C \in \mathcal{C}_n$, the following holds:

$$\left\{\mathsf{Sim}^{\mathcal{O}_C(\cdot)}\left(1^\lambda, 1^n, 1^{|C|}\right)\right\} \approx_c \left\{\mathcal{A}^{CC_{\mathsf{hrdw}}}(1^\lambda, CC_{\mathsf{sftw}}) \;:\; \begin{array}{c} CC \leftarrow \mathsf{Compile}(1^\lambda, C) \\ (CC_{\mathsf{hrdw}}, CC_{\mathsf{sftw}}) = CC \end{array}\right\}$$

where $\mathcal{O}_C(\cdot)$ provides one-time access to the circuit C, and CC_{hrdw}, CC_{sftw} are the hardware and software components of the one-time program CC.

Intuitively, the above definition requires that for any one-time program, the output of an adversary given full access to the software component and black-box access to the hardware component should be indistinguishable from the output of a simulator with one-time oracle access to the circuit. In this work we adapt the traditional definition of one-time compilers from a combination of *hardware-software* setting to only *software* setting, but in presence of a blockchain protocol.

Definition 3.9. A one-time compiler $\mathsf{OTC} = (\mathsf{Compile}, \mathsf{Eval})$ for a class of circuits $\mathcal{C} = \{\mathcal{C}_n\}_n$ is said to be a B/C-secure one-time compiler if for every PPT adversary $\mathcal{A}$, there exists a PPT simulator Sim such that for all λ, n, $C \in \mathcal{C}_n$, the following holds:

$$\left\{\mathsf{view}_{\mathsf{Sim}}\left(\mathsf{EXEC}^{\Gamma^V}\left(\mathsf{Sim}^{\mathcal{O}_C(\cdot)}\left(1^n, 1^{|C|}\right), \mathcal{Z}, 1^\lambda\right)\right)\right\}$$
$$\approx_c$$
$$\left\{\mathsf{view}_{\mathcal{A}}\left(\mathsf{EXEC}^{\Gamma^V}\left(\mathcal{A}\left(CC\right), \mathcal{Z}, 1^\lambda\right)\right) \;:\; CC \leftarrow \mathsf{Compile}(1^\lambda, C)\right\}$$

where $\mathcal{O}_C(\cdot)$ provides one-time access to the circuit C.

The above definition permits the adversary $\mathcal{A}$ to adaptively choose an input on which it could run the one-time program CC. In other words, $\mathcal{A}$ can select the input after it receives the one-time program. This definition guarantees one-time secrecy in a strong sense. We also introduce a slightly weaker notion of selective one-time secrecy as follows.

Definition 3.10. A one-time compiler $\mathsf{OTC} = (\mathsf{Compile}, \mathsf{Eval})$ for a class of circuits $\mathcal{C} = \{\mathcal{C}_n\}_n$ is said to be a B/C-*selectively*-secure one-time compiler if for every *admissible* PPT adversary $\mathcal{A}$, there exists a PPT simulator Sim such that for all λ, n, $C \in \mathcal{C}_n$ and $x \in \{0,1\}^n$, the following holds:

$$\left\{\mathsf{view}_{\mathsf{Sim}}\left(\mathsf{EXEC}^{\Gamma^V}\left(\mathsf{Sim}\left(1^n, 1^{|C|}, x, C(x)\right), \mathcal{Z}, 1^\lambda\right)\right)\right\}$$
$$\approx_c$$
$$\left\{\mathsf{view}_{\mathcal{A}}\left(\mathsf{EXEC}^{\Gamma^V}\left(\mathcal{A}\left(CC\right), \mathcal{Z}, 1^\lambda\right)\right) \;:\; CC \leftarrow \mathsf{Compile}(1^\lambda, C)\right\}$$

where adversary $\mathcal{A}$ is *admissible* if it evaluates the one-time program CC on x before evaluating on any other input.

4 Proof-of-Stake Protocols: Abstraction and Definitions

In this paper, we work in the execution model for proof-of-stake based protocols described in previous section. It is reasonable to assume that any adversary in this model would have full access to the blockchain as well as could possibly affect the protocol execution by adversarially mining for blocks or deviating from the protocol. It also seems reasonable to assume that no real-world adversary could run with a majority stake, or in other words majority voting power, as otherwise such an adversary could possibly affect the protocol execution arbitrarily, thereby destroying any guarantee that we could hope to get. All such restrictions could be captured by defining the adversary and environment to be sufficiently restrictive by considering appropriate compliant executions as discussed in previous section.

In this section, we define various security properties for proof-of-stake based blockchain protocols. We would like to point out that prior works [GKL15, KP15, PSS16, KRDO16, PS16a, PS16b, BPS16a, BPS16b, GKLP16, GKL16] have mostly considered only chain consistency, chain quality and chain growth as desiderata for blockchain protocols. We, on the other hand, also introduce many new security properties inspired by the notions of stake contribution and adversarial forking in POS based protocols. Later we also show that existing POS based protocols [KRDO16, BPS16b] already satisfy these stronger security properties. We believe that these new properties will have wider applicability as already suggested by our NIZK, one-time program and pay-per-use program constructions.

We also extend the abstraction for blockchain protocols to introduce additional POS specific abstracts. Below we introduce some necessary notations and definitions.

Notations. We denote by $\mathbf{B}^{\lceil \ell}$ the chain resulting from the "pruning" last ℓ blocks in $\mathbf{B}$. Note that for $\ell \geq |\mathbf{B}|$, $\mathbf{B}^{\lceil \ell} = \epsilon$. Also, if $\mathbf{B}_1$ is a prefix of $\mathbf{B}_2$, then we write $\mathbf{B}_1 \preceq \mathbf{B}_2$. We also use $\mathbf{B}^{\ell\rceil}$ to denote the chain containing last ℓ blocks in $\mathbf{B}$, i.e. $\mathbf{B}^{\ell\rceil} = \mathbf{B} \setminus \mathbf{B}^{\lceil \ell}$. Note that for $\ell \geq |\mathbf{B}|$, $\mathbf{B}^{\ell\rceil} = \mathbf{B}$.

Let $\mathsf{EXEC}^{\Gamma}(\mathcal{A}, \mathcal{Z}, 1^{\lambda})$ be the random variable denoting the joint view of all parties in the protocol execution. This fully determines the execution. Recall that each blockchain protocol is also associated with a validity predicate, however we avoid explicitly mentioning it whenever possible.

For any POS based blockchain protocol Γ, there exists a polynomial time algorithm $\mathsf{stake} : \{0,1\}^* \times \{0,1\}^* \to \mathbb{Q}^+$ which takes as inputs the blockchain $\mathbf{B}$ and a public identity id, and outputs a rational value. Concretely, consider a party P with public identity id, we use $\mathsf{stake}(\mathbf{B}, \mathsf{id})$ to denote the stake of party P as per the blockchain $\mathbf{B}$. For an adversary $\mathcal{A}$ that controls all parties with public identities in the set $\mathcal{X}$, its total stake as per blockchain $\mathbf{B}$ can computed as $\sum_{\mathsf{id} \in \mathcal{X}} \mathsf{stake}(\mathbf{B}, \mathsf{id})$. We overload the notation and use $\mathsf{stake}(\mathbf{B}, \mathcal{A})$ to denote $\mathcal{A}$'s total stake, and $\mathsf{stake}_{\mathsf{total}}$ to denote the combined stake of all parties i.e. $\mathsf{stake}_{\mathsf{total}} = \sum_{\mathsf{id}} \mathsf{stake}(\mathbf{B}, \mathsf{id})$. Also, we will simply write $\mathsf{stake}_{\mathcal{A}}$ whenever $\mathbf{B}$ is clear from context.

For any PPT adversary $\mathcal{A}$, the adversarial stake ratio $\mathsf{stake\text{-}ratio}_{\mathcal{A}}(\mathbf{B})$ w.r.t. blockchain $\mathbf{B}$ is defined as the ratio of $\mathcal{A}$'s total stake over combined stake of all parties. More formally,

$$\mathsf{stake\text{-}ratio}_{\mathcal{A}}(\mathbf{B}) = \frac{\mathsf{stake}_{\mathcal{A}}}{\mathsf{stake}_{\mathsf{total}}}.$$

We will drop dependence of $\mathsf{stake\text{-}ratio}_{\mathcal{A}}$ on blockchain $\mathbf{B}$ whenever clear from context.

Also, let $\mathsf{miner} : \{0,1\}^* \times \mathbb{N} \to \{0,1\}^*$ be a function that takes as input the blockchain $\mathbf{B}$ and an index i, and returns the public identity of the party that mined the i^{th} block, where blocks are counted from the head of the blockchain.[11] We overload the notation and use $\mathsf{miner}(\mathbf{B}, [\ell])$ to denote the *set* of public identities of all parties that mined at least one block in the last ℓ blocks of the blockchain $\mathbf{B}$.[12]

4.1 Chain Consistency

First, we define the chain consistency property for blockchain protocols Γ with environment $\mathcal{Z}$ and adversary $\mathcal{A}$. At a very high level, it states that the blockchains of any two honest parties at any two (possibly different) rounds during protocol execution can differ only in the last ℓ blocks with all but negligible probability, where ℓ parameterizes strength of the property. In other words, this suggests that if any party is honestly executing the blockchain protocol, then it could always assert that any block which is at least ℓ blocks deep in its blockchain is immutable.

A more general definition appears in [PSS16] which is an extension of the common prefix property by Garay et al. [GKL15]. As in prior works, we first define the consistency predicate and then use it to define the chain consistency property for blockchain protocols.

Predicate 1. (Consistency) Let $\mathsf{consistent}$ be the predicate such that $\mathsf{consistent}^{\ell}(\mathsf{view}) = 1$ iff for all rounds $r \leq \widetilde{r}$, and all players i, j (potentially the same) in view such that i is *honest* at round r with blockchain $\mathbf{B}$ and j is *honest* at round $\widetilde{r}$ with blockchain $\widetilde{\mathbf{B}}$, we have that $\mathbf{B}^{\lceil \ell} \preceq \widetilde{\mathbf{B}}$.

Definition 4.1. (Chain Consistency) A blockchain protocol Γ satisfies $\ell_0(\cdot)$-consistency with adversary $\mathcal{A}$ in environment $\mathcal{Z}$, if there exists negligible function $\mathrm{negl}(\cdot)$ such that for every $\lambda \in \mathbb{N}$, $\ell > \ell_0(\lambda)$ the following holds:

$$\Pr\left[\mathsf{consistent}^{\ell}(\mathsf{view}) = 1 \;\middle|\; \mathsf{view} \leftarrow \mathsf{EXEC}^{\Gamma}\left(\mathcal{A}, \mathcal{Z}, 1^{\lambda}\right)\right] \geq 1 - \mathrm{negl}(\lambda).$$

4.2 Defining Stake Fraction

For any POS based blockchain protocol, we could define special quantitative measures for a blockchain analogous to the *combined difficulty* or 'length' measure as in case of POW based protocols. For example, in Bitcoin 'length' of a chain of blocks is computed as the sum of *difficulty* of all individual blocks where difficulty is measured as the hardness of puzzle solved.

Note that in any POS based protocol, ideally the number of blocks mined by any party directly depends on its stake, or in other words, voting power is proportional to the amount of stake with a party. Also, each

[11] The rightmost (i.e., most recently added) block is called the head of the blockchain.

[12] Note that a party could potentially mine more than one block in a sequence of ℓ blocks.

new block added to the blockchain contains an efficiently verifiable proof of stake provided by a miner in the form of digital signatures. So, for POS based protocols, we could measure difficulty in terms of the amount of stake proven per block. The analogy being that solving POW puzzles with high difficulty requires more work (higher voting power) from a miner, and since voting power in POS based protocols is measured in terms of stake ratio, so for such protocols difficulty is measured as the amount of stake proven. Below we formally define such a measure.

Definition 4.2. (Proof-of-Stake Fraction) The proof-of-stake fraction $\mathsf{u\text{-}stakefrac}(\mathbf{B}, \ell)$ w.r.t. blockchain $\mathbf{B}$ is defined as the combined amount of *unique* stake whose proof is provided in last ℓ mined blocks. More formally, let $\mathcal{M} = \mathsf{miner}(\mathbf{B}, [\ell])$,

$$\mathsf{u\text{-}stakefrac}(\mathbf{B}, \ell) = \frac{\sum_{\mathsf{id} \in \mathcal{M}} \mathsf{stake}(\mathbf{B}, \mathsf{id})}{\mathsf{stake}_{\mathsf{total}}}.$$

In the above definition, it is important to note that we are only interested in the amount of *unique stake* proven. To understand this, first note that if some party added proof of its stake on the blockchain (i.e., mined a new block), then it would increase the probability of other parties mining on top of the newly mined block instead of mining on top of the previous block. However, if a certain single party with a low total stake is mining an unreasonably high proportion of blocks in a short span of rounds (or for simplicity all the blocks) on some chain, then other parties might not want to extend on top of such a blockchain as it could possibly correspond to an adversarial chain of blocks. So, by considering only unique stake we could use proof-of-stake fraction to (approximately) distinguish between (possibly) adversarial and honest blockchains as a higher proof-of-stake fraction increases confidence in that chain.

For some applications, we also need to consider only the amount of stake whose proof was provided by the honest parties in the blockchain. Below we define the proof-of-honest-stake fraction.

Definition 4.3. (Proof-of-*Honest*-Stake Fraction) The proof-of-honest-stake fraction $\mathsf{u\text{-}honest\text{-}stakefrac}(\mathbf{B}, \ell)$ w.r.t. blockchain $\mathbf{B}$ is defined as the combined amount of *unique* stake held by the *honest* parties whose proof is provided in last ℓ mined blocks. More formally, let $\mathcal{M} = \mathsf{miner}(\mathbf{B}, [\ell])$ and $\mathcal{M}_{\mathsf{honest}}$ denote the honest parties in $\mathcal{M}$, then

$$\mathsf{u\text{-}honest\text{-}stakefrac}(\mathbf{B}, \ell) = \frac{\sum_{\mathsf{id} \in \mathcal{M}_{\mathsf{honest}}} \mathsf{stake}(\mathbf{B}, \mathsf{id})}{\mathsf{stake}_{\mathsf{total}}}.$$

4.3 Stake Contribution Properties

In the previous section, we defined the notion of proof-of-stake fraction and proof-of-honest-stake fraction. Now, we define some useful properties for POS based blockchain protocols inspired by the above stake abstraction. We know that in any POS based protocol each mined block contains a proof of stake. At a very high level, the sufficient stake contribution property says that in a sufficiently long sequence of valid blocks, a significant amount of stake has been proven.

In other words, it says that after sufficiently many rounds, the amount of proof-of-stake added in mining the ℓ most recent blocks is a fairly high fraction (at least β) of the total stake in the system, where ℓ and β are property parameters denoting the length of chain and minimum amount of stake fraction in it (respectively). More formally, we define it as follows.

Predicate 2. (Sufficient Stake Contribution) Let $\mathsf{suf\text{-}stake\text{-}contr}$ be the predicate such that $\mathsf{suf\text{-}stake\text{-}contr}^{\ell}(\mathsf{view}, \beta) = 1$ iff for every round $r \geq \ell$, and each player i in view such that i is honest at round r with blockchain $\mathbf{B}$, we have that last ℓ blocks in blockchain $\mathbf{B}$ contain a *combined* proof of stake of more than $\beta \cdot \mathsf{stake}_{\mathsf{total}}$, i.e. $\mathsf{u\text{-}stakefrac}(\mathbf{B}, \ell) > \beta$.

Below we define the sufficient stake contribution property for blockchain protocols.

Definition 4.4. (Sufficient Stake Contribution) A blockchain protocol Γ satisfies $(\beta(\cdot), \ell_0(\cdot))$-sufficient stake contribution property with adversary $\mathcal{A}$ in environment $\mathcal{Z}$, if there exists a negligible function $\mathrm{negl}(\cdot)$ such that for every $\lambda \in \mathbb{N}$, $\ell \geq \ell_0(\lambda)$ the following holds:

$$\Pr\left[\mathsf{suf\text{-}stake\text{-}contr}^{\ell}(\mathsf{view}, \beta(\lambda)) = 1 \;\middle|\; \mathsf{view} \leftarrow \mathsf{EXEC}^{\Gamma}\left(\mathcal{A}, \mathcal{Z}, 1^{\lambda}\right)\right] \geq 1 - \mathrm{negl}(\lambda).$$

Previously we defined the notion of proof-of-honest-stake fraction along the lines of proof-of-stake fraction in which only the amount of honestly held stake was measured. Analogously, we could define the sufficient honest stake contribution property which says that in a sufficiently long sequence of valid blocks, a significant amount of *honestly held* stake has been proven.

Predicate 3. (Sufficient *Honest* Stake Contribution) Let $\mathsf{honest\text{-}suf\text{-}stake\text{-}contr}$ be the predicate such that $\mathsf{honest\text{-}suf\text{-}stake\text{-}contr}^{\ell}(\mathsf{view}, \beta) = 1$ iff for every round $r \geq \ell$, and each player i in view such that i is honest at round r with blockchain $\mathbf{B}$, we have that last ℓ blocks in blockchain $\mathbf{B}$ contain a *combined* proof of *honest* stake of more than $\beta \cdot \mathsf{stake}_{\mathsf{total}}$, i.e. $\mathsf{u\text{-}honest\text{-}stakefrac}(\mathbf{B}, \ell) > \beta$.

Definition 4.5. (Sufficient *Honest* Stake Contribution) A blockchain protocol Γ satisfies $(\beta(\cdot), \ell_0(\cdot))$-sufficient honest stake contribution property with adversary $\mathcal{A}$ in environment $\mathcal{Z}$, if there exists a negligible function $\mathrm{negl}(\cdot)$ such that for every $\lambda \in \mathbb{N}$, $\ell \geq \ell_0(\lambda)$ the following holds:

$$\Pr\left[\mathsf{honest\text{-}suf\text{-}stake\text{-}contr}^{\ell}(\mathsf{view}, \beta(\lambda)) = 1 \;\middle|\; \mathsf{view} \leftarrow \mathsf{EXEC}^{\Gamma}\left(\mathcal{A}, \mathcal{Z}, 1^{\lambda}\right)\right] \geq 1 - \mathrm{negl}(\lambda).$$

4.4 Bounded Forking Properties

Note that during protocol execution, any adversary could possibly generate a *private* chain of blocks which may or may not satisfy blockchain validity predicate, and may significantly diverge from the local blockchain in the view of honest parties. We call such a private chain of blocks, created by the adversary, a *fork*. In this work, we consider the following bounded forking properties which (at a very high level) require that no polytime adversary can create a sufficiently long fork containing valid blocks such that the combined amount of proof of stake proven in that fork is higher than certain threshold.

We start by defining the bounded stake forking property which says that if an adversary creates a fork of length at least $\ell_1 + \ell_2$ then the proof-of-stake fraction in the last ℓ_2 blocks of the fork is not more than α, where α, ℓ_1, ℓ_2 are property parameters with α being the threshold and $\ell_1 + \ell$ denoting the fork length. More formally, we first define the bounded stake fork predicate and then use it to define the bounded stake forking property.

Predicate 4. (Bounded Stake Fork) Let $\mathsf{bd\text{-}stake\text{-}fork}$ be the predicate such that $\mathsf{bd\text{-}stake\text{-}fork}^{(\ell_1,\ell_2)}(\mathsf{view}, \alpha) = 1$ iff for all rounds $r \geq \widetilde{r}$, for each pair of players i, j in view such that i is honest at round r with blockchain $\mathbf{B}$ and j is corrupt in round $\widetilde{r}$ with blockchain $\widetilde{\mathbf{B}}$, if there exists $\ell' \geq \ell_1 + \ell_2$ such that $\widetilde{\mathbf{B}}^{\lceil \ell'} \preceq \mathbf{B}$ and for all $\widetilde{\ell} < \ell'$, $\widetilde{\mathbf{B}}^{\lceil \widetilde{\ell}} \not\preceq \mathbf{B}$, then $\mathsf{u\text{-}stakefrac}(\widetilde{\mathbf{B}}, \ell' - \ell_1) \leq \alpha$.

Definition 4.6. (Bounded Stake Forking) A blockchain protocol Γ satisfies $(\alpha(\cdot), \ell_1(\cdot), \ell_2(\cdot))$-bounded stake forking property with adversary $\mathcal{A}$ in environment $\mathcal{Z}$, if there exists a negligible functions $\mathrm{negl}(\cdot), \delta(\cdot)$ such that for every $\lambda \in \mathbb{N}$, $\ell \geq \ell_1(\lambda)$, $\widetilde{\ell} \geq \ell_2(\lambda)$ the following holds:

$$\Pr\left[\mathsf{bd\text{-}stake\text{-}fork}^{(\ell,\widetilde{\ell})}(\mathsf{view}, \alpha(\lambda) + \delta(\lambda)) = 1 \;\middle|\; \mathsf{view} \leftarrow \mathsf{EXEC}^{\Gamma}\left(\mathcal{A}, \mathcal{Z}, 1^{\lambda}\right)\right] \geq 1 - \mathrm{negl}(\lambda).$$

The above property only stipulates that the proof-of-stake fraction of any adversarially generated fork is bounded. However, we additionally might expect a POS based blockchain protocol to satisfy the sufficient stake contribution property which states that any honest party's blockchain will have sufficiently high proof-of-stake fraction. Therefore, combining both these properties, we could define a stronger property for blockchain protocols which states that a sufficiently long chain of blocks generated during an honest protocol execution could always be *distinguished* from any adversarially generated fork. Also, the combined amount of stake proven in those sequences (i.e., its proof-of-stake fraction), which could be computed in polynomial time, could be used to distinguish such sequences. Formally, we could define it as follows.

Definition 4.7. (Distinguishable Forking) A blockchain protocol Γ satisfies $(\alpha(\cdot), \beta(\cdot), \ell_1(\cdot), \ell_2(\cdot))$-distinguishable forking property with adversary $\mathcal{A}$ in environment $\mathcal{Z}$, if there exists a negligible functions $\mathrm{negl}(\cdot), \delta(\cdot)$ such

that for every $\lambda \in \mathbb{N}$, $\ell \geq \ell_1(\lambda)$, $\widetilde{\ell} \geq \ell_2(\lambda)$ the following holds:

$$\Pr\left[\begin{array}{c} \alpha(\lambda)+\delta(\lambda)<\beta(\lambda) \ \wedge \ \mathsf{suf\text{-}stake\text{-}contr}^{\widetilde{\ell}}(\mathsf{view},\beta(\lambda))=1 \\ \wedge \quad \mathsf{bd\text{-}stake\text{-}fork}^{(\ell,\widetilde{\ell})}(\mathsf{view},\alpha(\lambda)+\delta(\lambda))=1 \end{array} \middle| \mathsf{view} \leftarrow \mathsf{EXEC}^{\Gamma}\left(\mathcal{A},\mathcal{Z},1^{\lambda}\right)\right] \geq 1-\mathrm{negl}(\lambda).$$

5 Instantiating our Framework

In this section, we show that the proposed proof-of-stake based blockchain protocols of [KRDO16] and [BPS16b] satisfy all the properties described in Section 4 for suitable parameters. We start by defining some additional properties for POS based blockchain protocols and then discuss relations among all these.

5.1 Chain Quality and Bounded Length Forking

Chain Quality. Another important property defined in prior works is of chain quality which was initially informally discussed on the Bitcoin forum [mtg10], and formally defined by [GKL15]. At a high level, it says that the number of blocks contributed by the adversary should not be very large, or in other words its contribution must be proportional to its voting power. Alternatively, this could be interpreted as a measure of fairness in the protocol and used to define a lower bound on the number of blocks contributed by honest parties. To be consistent with prior works, we define chain quality predicate with respect to the fraction of honest blocks.

Predicate 5. (Quality) Let $\mathsf{quality}$ be the predicate such that $\mathsf{quality}^{\ell}_{\mathcal{A}}(\mathsf{view},\mu)=1$ iff for every round $r \geq \ell$, and each player i in view such that i is *honest* at round r with blockchain $\mathbf{B}$, we have that out of ℓ blocks in $\mathbf{B}^{\lceil \ell}$ at least μ fraction of blocks are "honest".

Note that a block is said to be *honest* iff it is mined by an honest party. Below we recall the chain quality property for blockchain protocols as it appears in prior works.

Definition 5.1. (Chain Quality) A blockchain protocol Γ satisfies $(\mu(\cdot),\ell_0(\cdot))$-chain quality with adversary $\mathcal{A}$ in environment $\mathcal{Z}$, if there exists a negligible function $\mathrm{negl}(\cdot)$ such that for every $\lambda \in \mathbb{N}$, $\ell \geq \ell_0(\lambda)$ the following holds:

$$\Pr\left[\mathsf{quality}^{\ell}_{\mathcal{A}}(\mathsf{view},\mu(\lambda))=1 \mid \mathsf{view} \leftarrow \mathsf{EXEC}^{\Gamma}\left(\mathcal{A},\mathcal{Z},1^{\lambda}\right)\right] \geq 1-\mathrm{negl}(\lambda).$$

Bounded Length Forking. Additionally, we would expect a POS based blockchain protocol to satisfy the property that — no PPT adversary should be able to generate (with non-negligible probability) a sufficiently long *fork* that satisfies all validity conditions and *the last block in that fork was mined by an honest party.* The intuition behind this is that if the adversary can generate such a sufficiently long chain, then it would mean that it could prevent consensus between honest parties for a sufficiently long time. To formally capture this, we define the bounded length forking property over blockchain protocols as follows.

Predicate 6. (Bounded Length Fork) Let $\mathsf{bd\text{-}length\text{-}fork}$ be the predicate such that $\mathsf{bd\text{-}length\text{-}fork}^{\ell}(\mathsf{view})=1$ iff there exists rounds $r,\widetilde{r}$, players i,j in view such that i is honest at round r with blockchain $\mathbf{B}$ and j is corrupt at round $\widetilde{r}$ with blockchain $\widetilde{\mathbf{B}}$, and there exists $\ell' \geq \ell$ such that $\widetilde{\mathbf{B}}^{\lceil \ell'} \preceq \mathbf{B}$ and for all $\widetilde{\ell} < \ell'$, $\widetilde{\mathbf{B}}^{\lceil \widetilde{\ell}} \not\preceq \mathbf{B}$, and *the last block in chain* $\widetilde{\mathbf{B}}$ *is honest* (i.e., not mined by the adversary).

Definition 5.2. (Bounded Length Forking) A blockchain protocol Γ satisfies $\ell_0(\cdot)$-bounded length forking property with adversary $\mathcal{A}$ in environment $\mathcal{Z}$, if there exists a negligible function $\mathrm{negl}(\cdot)$ such that for every $\lambda \in \mathbb{N}$, $\ell \geq \ell_0(\lambda)$ the following holds:

$$\Pr\left[\mathsf{bd\text{-}length\text{-}fork}^{\ell}(\mathsf{view})=1 \mid \mathsf{view} \leftarrow \mathsf{EXEC}^{\Gamma}\left(\mathcal{A},\mathcal{Z},1^{\lambda}\right)\right] \leq \mathrm{negl}(\lambda).$$

5.2 Proof-of-Stake Blockchain Protocols: Snowwhite, Ouroboros

Very recently, Kiayias et al. [KRDO16] and Pass et al. [BPS16b] proposed POS based blockchain protocols and proved that their protocols satisfy consistency, chain quality and chain growth properties for suitable parameters against certain reasonable adversary-environment pairs. Pass et al. proved the following theorem about their blockchain protocol $\Gamma_{\mathsf{snowwhite}}$.

Theorem 5.1. ([BPS16b, Theorem 1], Paraphrased) Let n be the number of nodes executing the blockchain protocol $\Gamma_{\mathsf{snowwhite}}$, p be the probability that a node is elected leader in a given round[13], and α, β be the respective probabilities of the elected node being honest or corrupt, and γ be the discounted version of α in presence of adversarial network delays. For any constant $\epsilon_1, \epsilon_2 > 0$, any $\ell_0 \geq \epsilon_1 \lambda$, $\Gamma_{\mathsf{snowwhite}}$ satisfies:

1. ℓ_0-consistency,
2. $((1-\epsilon_2)(1-\beta/\alpha), \ell_0)$-chain quality,
3. $((1-\epsilon_2)\gamma, (1-\epsilon_2)np, \ell_0)$-chain growth

against any $\Gamma_{\mathsf{snowwhite}}$-compliant adversary-environment pair $(\mathcal{A}, \mathcal{Z})$.

A similar result was given by Kiayias et al. [KRDO16] as well for their blockchain protocol $\Gamma_{\mathsf{ouroboros}}$. Note that both works only prove the consistency, chain quality and chain growth properties for their proposed protocols. However, in this work we also want the underlying blockchain protocol to satisfy certain stronger properties — sufficient stake contribution (Definition 4.4), sufficient honest stake contribution (Definition 4.5), bounded stake forking (Definition 4.6) and distinguishable forking (Definition 4.7) for suitable parameters.

We show in the next section that these properties are already satisfied by both the proposed protocols and more importantly, we show that sufficient honest stake contribution and bounded stake forking properties are already implied by consistency and chain quality under simple assumptions about the distribution of stake between honest parties.

5.3 Relating blockchain properties

In this section, we show how the different blockchain properties described in Section 5.1 are related. First, we discuss the sufficient honest stake contribution and chain quality properties, and show that if a blockchain protocol satisfies chain quality (Definition 5.1), then it also satisfies sufficient honest stake contribution property (Definition 4.5) for appropriate parameters under suitable stake distributions. Next, we also show that the consistency property (Definition 4.1) implies the bounded length forking property (Definition 5.2).

Finally, we show that the sufficient honest stake contribution and bounded length forking properties along with the assumption that proof-of-stake is unforgeable imply the bounded stake forking and distinguishable forking properties with appropriate parameters.

5.3.1 Chain Quality to Sufficient Honest Stake Quantity

Recall that the chain quality property suggests that in a sufficiently long sequence of valid blocks, a significant fraction of blocks would have been mined by honest parties. In other words, this suggests that no adversary could generate (with non-negligible probability) an unfair proportion of blocks in a contiguous sequence of ℓ or more valid blocks. Suppose that in any ℓ consecutive blocks at least $\ell/2$ blocks were mined by honest parties. Also, for simplicity assume that there are n honest parties and each of them hold *exactly* $\frac{c}{n}$ fraction of total stake where c is some constant less than 1. Additionally, assume for simplicity that all these n parties are online/ actively mining. Now since we know that the probability an honest party gets to mine the next block is proportional to its stake, thus in this case any honest party is equally likely to mine the next block. Therefore, if $\ell/2 \geq n \log(n)$, then in expectation each of the n honest parties must have mined at least one

[13] In any round, the leader could be regarded as the node that mines the new block.

block in the sequence of ℓ blocks.[14] A similar claim could also be made with all but negligible probability with a slightly larger ℓ. Thus, this suggests that after say n^2 rounds, all the honest parties must have mined at least one block which implies that the fraction of total stake proved in these rounds by honest parties is at least $n \cdot \frac{c}{n} = c$.

In the general case the distribution of stake among honest parties could be arbitrarily skewed (i.e., not necessarily uniform or even close to uniform), so we can not apply the bounds from the standard coupon collecting problem directly. However, we show that for most natural stake distributions similar bounds could be obtained by using appropriate concentration inequalities. Below we formally describe the stake distribution for which we connect the chain quality and sufficient stake contribution properties.

(m, β, γ)-Stake distribution. Consider the scenario where polynomially many parties are executing the blockchain protocol Γ with adversary-environment pair $(\mathcal{A}, \mathcal{Z})$. If there are at least m honest parties such that each of these have at least γ fraction of total stake separately and they have β stake ratio altogether, then such a distribution is considered to be an (m, β, γ)-stake distribution. We would like to note that this only assumes that each of the m honest parties have at least γ stake ratio, and the actual distribution among these may be arbitrary.

For such stake distributions, we can claim the following without making any additional assumption.

Theorem 5.2. If a blockchain protocol Γ satisfies (μ, ℓ_0)-chain quality property over a (m, β, γ)-stake distribution, then it also satisfies (β, ℓ)-sufficient *honest* stake contribution property for all $\ell \geq \dfrac{\log(m) + \omega(\log(\lambda))}{\mu\gamma}$.

Proof. Consider an honest party i with blockchain $\mathbf{B}$ at round $r > \ell_0$. Let $B_1, \ldots, B_\ell$ be the sequence of last $\ell > \ell_0$ (consecutive) blocks in $\mathbf{B}$. Note that (μ, ℓ_0)-chain quality property implies that, with all but negligible probability, in any sequence of ℓ_0 or more consecutive blocks the fraction of blocks mined by honest parties is at least μ in each honest party's private blockchain. Thus, we know that at least $\mu\ell$ blocks are honest in $B_1, \ldots, B_\ell$. Let $p_1, \ldots, p_m$ be the honest parties that have stake ratio at least γ. We know that the probability party p_i mines a block is at least γ. Therefore, we can say that

$$\Pr[p_i \text{ did not mine any block in } B_1, \ldots, B_\ell] \leq (1-\gamma)^{\mu\ell} \leq e^{-\gamma\mu\ell}.$$

Using union bound, we can write that

$$\Pr[\exists\ i \text{ such that } p_i \text{ did not mine any block in } B_1, \ldots, B_\ell] \leq m \cdot e^{-\gamma\mu\ell}.$$

Substituting ℓ with $\dfrac{\log(m) + \omega(\log(\lambda))}{\mu\gamma}$, we get that the above probability is negligible. Therefore, the protocol also satisfies (β, ℓ)-sufficient honest stake contribution property for all such ℓ. Thus, the theorem follows. ∎

In the above proof, we would obtain tighter bounds if model the random variables as sub-gamma random variables and use corresponding concentration bounds. However we only want the parameter ℓ to be polynomially bounded, thus we avoided a more complicated analysis. Additionally, we could obtain even tighter lower bounds if we modify the underlying validity predicate of the blockchain protocols, but in this work we restrict ourselves from make any blockchain protocol specific assumptions.

Also, (by definition) we already know that if a blockchain protocol satisfies (β, ℓ)-sufficient *honest* stake contribution property for some parameters (β, ℓ) then it also satisfies (β, ℓ)-sufficient stake contribution property. Thus, we could also state the following.

Corollary 5.1. If a blockchain protocol Γ satisfies (μ, ℓ_0)-chain quality property over a (m, β, γ)-stake distribution, then it also satisfies (β, ℓ)-sufficient stake contribution property for all $\ell \geq \dfrac{\log(m) + \omega(\log(\lambda))}{\mu\gamma}$.

[14] The bound of $n \log(n)$ follows from the coupon collecting problem.

We would like to point out that we can not prove sufficient stake contribution property with better parameters, i.e. higher β values, when compared with sufficient honest stake contribution properties unless we make additional assumptions about the underlying blockchain protocol. This is because in the adversarially mined blocks, the adversary could potentially provide proof of extremely little stake fraction. In the worst case, it could even simply not interact in the protocol execution, thereby not providing any proof-of-stake. In such a scenario, clearly the sufficient honest stake contribution and sufficient stake contribution properties will share the same parameters.

Also, note that in the above proof, we assume that *at least* m honest parties that have stake at least γ are actively mining, i.e. are online. In other words, we do not assume that *all* honest parties controlling at least γ stake are online. Also, in scenarios where honest parties might stop mining (i.e., go offline) for say δ rounds, we could instead define the set of parties $p_1, \ldots, p_m$ as those that are online. Thus, we could still prove sufficient stake contribution property but with either lower values of parameter β or larger values of parameter ℓ. Concretely, the parameter ℓ will then also depend on δ. We leave further analysis for future work.

Next we discuss the consistency and bounded length forking properties.

5.3.2 Consistency to Bounded Length Forking

Recall that if a blockchain protocol satisfies chain consistency property, then that suggests that there exists a polynomial ℓ such that chains of any two honest parties differ only in the last ℓ blocks. Alternatively, it could be interpreted as follows. Let $\mathbf{B}$ be the longest most consistent chain (among honest parties) after say r rounds. We call this as the *central* blockchain because (with all but negligible probability) it is immutable. Let us assume that during protocol execution, the adversary was able to create a forked blockchain $\mathbf{B}'$ with a fork of length $\ell' > \ell$ w.r.t. to $\mathbf{B}$ such that the last block in $\mathbf{B}'$ was mined by some honest party i in round $r' \leq r$. Since $\mathbf{B}$ is the central blockchain, therefore we know that after r rounds $\mathbf{B}$ is a prefix of the private blockchain of party i. This would imply that the private blockchain of party i is not self-consistent as its chain after rounds r' and r differ in more than ℓ blocks which contradicts the consistency property. Thus, we could conclude the following.

Lemma 5.1. If a blockchain protocol Γ satisfies ℓ-consistency property, then it also satisfies ℓ-bounded length forking property.

Next we discuss distinguishable forking property and show that if proof-of-stake is unforgeable, then bounded length forking and sufficient stake contribution properties imply distinguishable forking property.

5.3.3 Distinguishable Forking Property

Suppose that a blockchain protocol satisfies bounded length forking property with parameter ℓ_1. Then this would imply that if the adversary creates a fork of length ℓ' ($\geq \ell_1$), then either the last $\ell' - \ell_1$ blocks in its private blockchain are adversarial (i.e., mined by the adversary), or the adversary was able to prove stake of *honest* coins (i.e., prove stake of some coins which belong to an honest party). This is because bounded length forking says that the length of a fork which ends at an honest block can be at most ℓ_1 (with all but negligible probability), and since the adversary created a fork of length more than ℓ_1, we could conclude that the last $\ell' - \ell_1$ blocks can not have been mined by any honest party. Now we know that in cryptocurrencies, proof-of-stake by any party is provided in the form of a digital signature. Therefore, assuming that no PPT adversary can forge a signature[15], this would imply that (with all but negligible probability) the last $\ell' - \ell_1$ blocks can only contain proof of stake which the adversary controls (i.e., adversary's coins).

Therefore, we could conclude the following two facts — (1) if a blockchain protocol Γ satisfies ℓ_1-bounded length forking property against all PPT adversaries with at most α stake ratio, then no PPT adversary can

[15]We would like to point out that here we assume that honest parties delete signing keys after they lose significance/stake, i.e. once the coins associated are transferred. Now the same analysis does not seem to work if parties are rational not honest as such parties could potentially sell their *old* keys. It is an interesting problem to formalize what guarantees can be provided in such a setting.

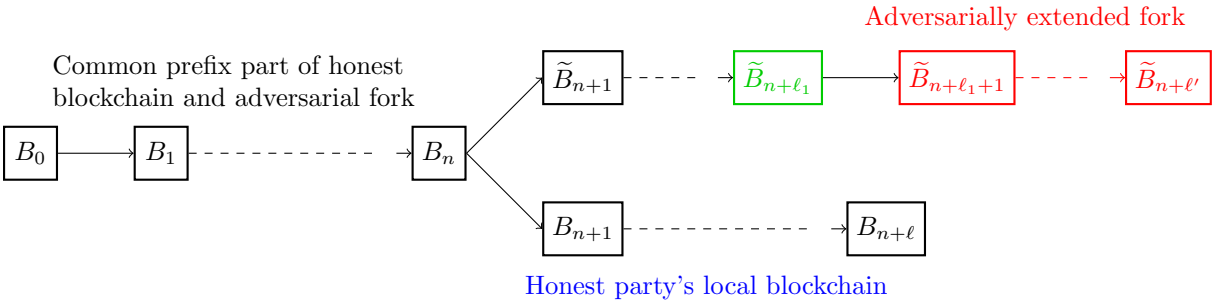

Figure 1: Visualizing Distinguishability Forking property.

create a fork of length ℓ' such that the last $\ell' - \ell_1$ blocks contain a proof of more than α stake ratio unless the adversary could forge on the underlying signature scheme, and (2) if it also satisfies (β, ℓ_2)-sufficient stake contribution property states that, with all but negligible probability, proof of at least β fraction of stake will be contained in any sequence of ℓ_2 consecutive blocks in an honest party's blockchain.

For an illustrative explanantion, see Figure 1. Consider a party, controlled by the adversary, with private blockchain $\widetilde{\mathbf{B}} = (B_0, \ldots, B_n, \widetilde{B}_{n+1}, \ldots, \widetilde{B}_{n+\ell'})$ and an honest party with private blockchain $\mathbf{B} = (B_0, \ldots, B_{n+\ell})$. Now ℓ_1-bounded length forking says that block $\widetilde{B}_{n+\ell_1}$ (highlighted as green in Figure 1) will be the last block in $\widetilde{\mathbf{B}}$ which was mined by some honest party. Therefore, all the blocks after $\widetilde{B}_{n+\ell_1}$ (highlighted as red) must have been created by the adversary on its own, thus the amount of stake proved in these latter (red) blocks could be at most the adversary's total stake (if forging proof-of-stake is hard). On the other hand, the amount of stake proved in the latter blocks of an honest party's blockchain will be significantly higher as suggested by the sufficient stake contribution property.

Combining the above facts, we could make a stronger claim which intuitively suggests that a sufficiently long chain of blocks in an honest execution can be distinguished from an adversarially created fork of sufficient length. This is because the combined amount of stake proven in such a sequence of blocks could be calculated in polynomial time and it could be used to distinguish such sequences.

Therefore, we could conclude the following.

Theorem 5.3. If a POS based blockchain protocol Γ satisfies $\ell_1(\cdot)$-bounded length forking property and $(\beta(\cdot), \ell_2(\cdot))$-sufficient stake contribution property against all $\alpha(\cdot)$-stake bounded PPT adversaries, and underlying signature scheme is secure (i.e., proof of stake is unforgeable), and the following constraint holds:

$$\exists\, c > 0, \forall \lambda \in \mathbb{N}, \quad \beta(\lambda) - \alpha(\lambda) \geq \frac{1}{\lambda^c},$$

then Γ also satisfies the $(\alpha(\cdot), \beta(\cdot), \ell_1(\cdot), \ell_2(\cdot))$-distinguishable forking property.

This follows directly from the discussion at the start of the section, therefore we skip the proof.

Additionally, observe that if a blockchain protocol satisfies distinguishable forking property with parameters $(\alpha, \beta, \ell_1, \ell_2)$ then it also satisfies bounded stake forking property with same parameters. However, for bounded stake forking property to hold, the protocol need not satisfy the sufficient stake contribution property. Concretely, we could state the following.

Corollary 5.2. If a POS based blockchain protocol Γ satisfies $\ell_1(\cdot)$-bounded length forking property against all $\alpha(\cdot)$-stake bounded PPT adversaries, and underlying signature scheme is secure (i.e., proof of stake is unforgeable), then Γ also satisfies the $(\alpha(\cdot), \ell_1(\cdot), \ell_2(\cdot))$-distinguishable forking property.

Combining all the above lemmas, theorems and corollaries, we could state the following stronger statement.

Theorem 5.4. Let n be the number of nodes executing the blockchain protocol $\Gamma_{\mathsf{snowwhite}}$, p be the probability that a node is elected leader in a given round, and δ_h, δ_c be the respective probabilities of the elected node

being honest or corrupt, and δ_d be the discounted version of δ_h in presence of adversarial network delays. If the stake is distributed as a (m, β, γ)-stake distribution and the adversary is α-stake bounded and proof of stake is unforgeable, then for any constant $\epsilon_1, \epsilon_2 > 0$, any $\ell_1 \geq \epsilon_1\lambda, \ell_2 \geq \dfrac{\log(m) + \omega(\log(\lambda))}{\mu\gamma}$ where $\mu = (1-\epsilon_2)(1-\delta_c/\delta_h)$, $\Gamma_{\mathsf{snowwhite}}$ satisfies:

1. ℓ_1-consistency,
2. $((1-\epsilon_2)(1-\delta_c/\delta_h), \ell_1)$-chain quality,
3. $((1-\epsilon_2)\delta_d, (1-\epsilon_2)np, \ell_1)$-chain growth,
4. (β, ℓ_2)-sufficient stake contribution,
5. (β, ℓ_2)-sufficient *honest* stake contribution,
6. ℓ_1-bounded length forking,
7. (α, ℓ_1, ℓ_2)-bounded stake forking,
8. $(\alpha, \beta, \ell_1, \ell_2)$-distinguishable forking

against any $\Gamma_{\mathsf{snowwhite}}$-compliant adversary-environment pair $(\mathcal{A}, \mathcal{Z})$.

A similar theorem could also be stated for $\Gamma_{\mathsf{ouroboros}}$.

6 NIZKs over Blockchain

In this section, we provide our construction for NIZKs from NIWIs and weighted threshold secret sharing scheme over any POS based blockchain protocol under an additional assumption that each miner's signing-verification key pair could be used as an decryption-encryption key pair. In other words, we assume that the blockchain protocol uses a public key integrated encryption-signature scheme.[16] Below we describe the main ideas.

Outline. Suppose the blockchain protocol satisfies ℓ_1-chain consistency, (β, ℓ_2)-sufficient honest stake contribution and $(1-\alpha, \ell_3, \ell_4)$-bounded stake forking properties. By chain consistency property, we know that all honest parties agree on all but last ℓ_1 (or less) blocks of blockchain $\mathbf{B}$. Also, bounded stake forking property suggests that no PPT adversary can generate a fork of length $\geq \ell_3 + \ell_4$ such that the proof-of-stake fraction after the first ℓ_3 blocks of the fork is more than $1-\alpha$.

At a high level, the scheme works as follows. An honest prover takes as input an instance-witness pair (x, w) and a blockchain $\mathbf{B}$. It starts by extracting, from its blockchain, the public identities (thereby public keys) of all the parties who mined a block in last ℓ_2 blocks of blockchain $\mathbf{B}^{\lceil \ell_1}$. In other words, it selects a *committee* of miners from the most recent part of its blockchain which has become globally persistent. Now, the NIZK proof of the statement $x \in \mathcal{L}$ consists of — (1) a set of ciphertexts $\{\mathsf{ct}_{\mathsf{id}}\}$ (one for each miner selected as part of the *committee*), and (2) a witness-indistinguishable proof for the statement "$x \in \mathcal{L}$ *OR the ciphertexts* $\{\mathsf{ct}_{\mathsf{id}}\}$ *together encrypt a fork of length more than* $\ell_3 + \ell_4$ *such that the proof-of-stake fraction after the first* ℓ_3 *blocks of the fork is more than* $1-\alpha$". In short, the witness-indistinguishable proof proves that either $x \in \mathcal{L}$ or the prover can break the bounded stake forking property. Since the above language is in $\mathbf{NP}$, an honest prover simply encrypts random values in ciphertexts $\{\mathsf{ct}_{\mathsf{id}}\}$ and uses witness w for the witness-indistinguishable proof. The prover outputs its blockchain $\mathbf{B}$, ciphertexts $\{\mathsf{ct}_{\mathsf{id}}\}$, witness-indistinguishable proof and all the blockchain property parameters.

[16] As we mentioned before, most blockchain protocols (like Bitcoin, Ethereum etc.) use ECDSA based signature schemes for which we could directly use ECIES-like integrated encryption schemes [Sho01]. Thus, our NIZKs are instantiable over existing blockchain protocols.

The verifier on input an instance x, proof π and blockchain $\mathbf{B}$ performs two checks — (1) the prover's blockchain is consistent with its local blockchain, and (2) the witness-indistinguishable proof gets verified. The completeness follows directly from the correctness properties of underlying primitives. Intuitively, the soundness is guaranteed by the fact that the blockchain protocol satisfies the $(1-\alpha, \ell_3, \ell_4)$-bounded stake forking property, and the system is zero-knowledge because a simulator can generate a witness for the trapdoor part of the statement (i.e., it could generate a long fork satisfying the minimum proof-of-stake constraint) as it controls all the honest parties executing the blockchain, therefore it could use their signing keys to compute such a fork privately. For making the system an argument of knowledge as well, we could additionally make the prover secret share the witness and encrypt a share to each member of the committee it selected. It will be crucial that the secret sharing scheme be a weighted threshold scheme as will become clearer later in this section.

Below we start by describing the $\mathsf{valid\text{-}fork}$ predicate which will be used later while defining the trapdoor part of the statement.

Predicate 7. Let $\mathsf{valid\text{-}fork}$ be the predicate such that it is satisfied iff the blockchain $\widetilde{\mathbf{B}}$ contains a fork of length at least $\ell_1 + \ell_2$ such that the fork satisfies the blockchain validity predicate as well as the the proof-of-stake fraction in the last ℓ_2 blocks of the fork is at least γ. More formally, $\mathsf{valid\text{-}fork}^V(\mathbf{B}, \widetilde{\mathbf{B}}, \ell_1, \ell_2, \gamma) = 1$ iff there exists $\ell' \geq \ell_1 + \ell_2$ such that $\widetilde{\mathbf{B}}^{\lceil \ell'} \preceq \mathbf{B}$ and for all $\widetilde{\ell} < \ell'$, $\widetilde{\mathbf{B}}^{\lceil \widetilde{\ell}} \not\preceq \mathbf{B}$, and $\mathsf{u\text{-}stakefrac}(\widetilde{\mathbf{B}}, \ell' - \ell_1) \geq \gamma$.

6.1 Construction

Let $\Gamma^V = (\mathsf{UpdateState}^V, \mathsf{GetRecords}, \mathsf{Broadcast})$ be a blockchain protocol, and $(\mathcal{P}_{\mathsf{NIWI}}, \mathcal{V}_{\mathsf{NIWI}})$ is a NIWI argument system for $\mathbf{NP}$, and $\mathsf{SS} = (\mathsf{Share}, \mathsf{Rec})$ be a weighted threshold secret sharing scheme, and $\mathsf{HS} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Sign}, \mathsf{Verify})$ be a public key integrated encryption-signature scheme. Below we describe our NIZK construction for an $\mathbf{NP}$ language $\mathcal{L}$ over blockchains.

- $\mathcal{P}\left(\mathsf{params} = (1^{\ell_1}, 1^{\ell_2}, 1^{\ell_3}, 1^{\ell_4}, \alpha, \beta), \mathbf{B}, x, w\right)$: The prover algorithm takes as input the length parameters $\ell_1, \ell_2, \ell_3, \ell_4$, stake fraction parameters α, β, a blockchain $\mathbf{B}$, an instance x and a witness w such that $\mathcal{R}(x, w) = 1$ where $\mathcal{R}$ is the instance-witness relation for language $\mathcal{L}$.

  Let $\mathbf{B}'$ correspond to the blockchain $\mathbf{B}$ with last ℓ_1 blocks pruned, i.e. $\mathbf{B}' = \mathbf{B}^{\lceil \ell_1}$. Let $\mathcal{M}$ denote the set of miners who mined at least one block in the last ℓ_2 blocks of the blockchain $\mathbf{B}'$, i.e. $\mathcal{M} = \mathsf{miner}(\mathbf{B}', [\ell_2])$. Also, let $\mathsf{stake}_{\mathsf{id}} = \mathsf{stake}(\mathbf{B}', \mathsf{id})$ and $\mathsf{pk}_{\mathsf{id}}$ be the stake and public key of party id, respectively.[17] First, it secret shares the witness w and an all zeros string (separately) into $|\mathcal{M}|$ shares with weights $\{\mathsf{stake}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{M}}$ and threshold $\beta \cdot \mathsf{stake}_{\mathsf{total}}$ as follows

  $$\{\mathsf{sh}_{\mathsf{id},1}\}_{\mathsf{id}} = \mathsf{Share}(w, \{\mathsf{stake}_{\mathsf{id}}\}_{\mathsf{id}}, \beta \cdot \mathsf{stake}_{\mathsf{total}}; s_1), \quad \{\mathsf{sh}_{\mathsf{id},2}\}_{\mathsf{id}} = \mathsf{Share}(0, \{\mathsf{stake}_{\mathsf{id}}\}_{\mathsf{id}}, \beta \cdot \mathsf{stake}_{\mathsf{total}}; s_2).$$

  Next, it encrypts all these shares as follows

  $$\forall\ \mathsf{id} \in \mathcal{M}, \quad \mathsf{ct}_{\mathsf{id},1} = \mathsf{Enc}(\mathsf{pk}_{\mathsf{id}}, \mathsf{sh}_{\mathsf{id},1}; r_{\mathsf{id},1}), \quad \mathsf{ct}_{\mathsf{id},2} = \mathsf{Enc}(\mathsf{pk}_{\mathsf{id}}, \mathsf{sh}_{\mathsf{id},2}; r_{\mathsf{id},2}).$$

  Finally, it computes a NIWI proof π' for the following statement

  $$\exists \{\mathsf{sh}_i, r_i\}_{i \in \mathcal{M}}, s \text{ such that } \left(\begin{array}{c} \{\mathsf{sh}_i\}_i = \mathsf{Share}(w, \{\mathsf{stake}_{\mathsf{id}}\}_{\mathsf{id}}, \beta \cdot \mathsf{stake}_{\mathsf{total}}; s) \ \wedge \\ \forall\ i,\ \mathsf{ct}_{i,1} = \mathsf{Enc}(\mathsf{pk}_i, \mathsf{sh}_i; r_i) \ \wedge\ \mathcal{R}(x, w) = 1 \end{array} \right)$$
  $$\bigvee \left(\begin{array}{c} \{\mathsf{sh}_i\}_i = \mathsf{Share}(\widetilde{\mathbf{B}}, \{\mathsf{stake}_{\mathsf{id}}\}_{\mathsf{id}}, \beta \cdot \mathsf{stake}_{\mathsf{total}}; s) \ \wedge \\ \forall\ i,\ \mathsf{ct}_{i,2} = \mathsf{Enc}(\mathsf{pk}_i, \mathsf{sh}_i; r_i) \ \wedge\ \mathsf{valid\text{-}fork}^V(\mathbf{B}', \widetilde{\mathbf{B}}, \ell_3, \ell_4, 1-\alpha) \end{array} \right)$$

  using the NIWI prover algorithm $\mathcal{P}_{\mathsf{NIWI}}$ with $\left(\{\mathsf{sh}_{\mathsf{id},1}, r_{\mathsf{id},1}\}_{\mathsf{id}}, s_1\right)$ as the witness. Finally, it sets the proof π as

  $$\pi = \left(\pi', \mathbf{B}, \{\mathsf{ct}_{\mathsf{id},1}, \mathsf{ct}_{\mathsf{id},2}\}_{\mathsf{id}}, \mathsf{params} = (1^{\ell_1}, 1^{\ell_2}, 1^{\ell_3}, 1^{\ell_4}, \alpha, \beta)\right).$$

[17] Observe that since HS is an integrated encryption-signature scheme, therefore the public verification keys of all parties executing the blockchain protocol could be used for encryption as well.

- $\mathcal{V}(\mathbf{B}, x, \pi)$: Let $\pi = \left(\pi', \overline{\mathbf{B}}, \{\mathsf{ct}_{i,1}, \mathsf{ct}_{i,2}\}_i, \mathsf{params} = (1^{\ell_1}, 1^{\ell_2}, 1^{\ell_3}, 1^{\ell_4}, \alpha, \beta)\right)$. The verifier starts by checking that blockchains $\overline{\mathbf{B}}$ and $\mathbf{B}$ are ℓ_1-consistent, i.e. $\overline{\mathbf{B}}^{\lceil \ell_1} \preceq \mathbf{B}$, as well as verifier's blockchain $\mathbf{B}$ is at least as long as prover's blockchain $\overline{\mathbf{B}}$, i.e. $|\overline{\mathbf{B}}| \leq |\mathbf{B}|$. If these check fail, then verifier rejects the proof and outputs 0. Otherwise, it runs the NIWI verifier algorithm $\mathcal{V}_{\mathsf{NIWI}}$ to verify proof π' and outputs same as the NIWI verifier.

6.2 Security Proof

We will now show that the NIZKs described in Section 6.1 is NIZK argument of knowledge as per Definition 3.5. More formally, we prove the following theorem where all the parameters are polynomials in the security parameter λ.

Theorem 6.1. If $(\mathcal{P}_{\mathsf{NIWI}}, \mathcal{V}_{\mathsf{NIWI}})$ is a NIWI argument system (Definition 3.3) for **NP**, SS is a weighted threshold secret sharing scheme (Definition 3.2), HS is a secure integrated public key encryption-signature scheme (Definition 3.1), and blockchain protocol Γ^V satisfies ℓ_1-chain consistency, (β, ℓ_2)-sufficient honest stake contribution properties against all PPT adversaries with at most α stake ratio, and $(1-\alpha, \ell_3, \ell_4)$-bounded stake forking property against all PPT adversaries with at most $\alpha + \beta$ stake ratio, then $(\mathcal{P}, \mathcal{V})$ with parameters $\alpha, \beta, \ell_1, \ell_2, \ell_3, \ell_4$ is a NIZK argument of knowledge for any **NP** language $\mathcal{L}$ over blockchain protocol Γ^V against all PPT adversaries with at most α stake ratio.

Proof. Below we provide the proofs of completeness, soundness, zero-knowledge and argument of knowledge.

Completeness. Fix any ℓ_1, ℓ_2, ℓ_3, ℓ_4, α, β, instance x and witness w. Suppose $\mathbf{B}, \overline{\mathbf{B}}$ be the respective blockchains of the prover and verifier. Let π be the NIZK proof created by prover with parameters ℓ_1, ℓ_2, ℓ_3, ℓ_4, α and β, and blockchain $\mathbf{B}$. We know that π is of the form $(\pi', \mathbf{B}, \{\mathsf{ct}_{i,1}, \mathsf{ct}_{i,2}\}_i, \mathsf{params})$ where π' is a NIWI proof of the statement described before.

Since both prover and verifier are running the blockchain protocol honestly, therefore blockchains $\mathbf{B}$ and $\overline{\mathbf{B}}$ are ℓ_1-consistent (with all but negligible probability). Thus the consistency checks during verification pass. Additionally, since $\mathcal{R}(x, w) = 1$ the prover computes the NIWI proof π' using secret shares of witness w, thus the verifier accepts the NIWI proof π' as it follows directly from the correctness of the NIWI argument system. Therefore, $(\mathcal{P}, \mathcal{V})$ satisfies the NIZK correctness condition.

Soundness and Argument of Knowledge. Suppose $x \notin \mathcal{L}$ and a cheating prover succeeds with non-negligible probability to cause the verifier to accept. Then either the $(1-\alpha, \ell_3, \ell_4)$-bounded stake forking property or the (β, ℓ_2)-sufficient honest stake contribution property is not satisfied by the blockchain protocol Γ^V, or the NIWI argument system is not sound.

Consider a cheating prover which runs with at most α stake ratio (i.e., a cheating prover with stake ratio bounded by α). Assume that it outputs a proof π of the form $(\pi', \overline{\mathbf{B}}, \{\mathsf{ct}_{i,1}, \mathsf{ct}_{i,2}\}_i, \mathsf{params})$ that gets verified with non-negligible probability with respect to a blockchain $\mathbf{B}$, where $\mathbf{B}$ is some honest party's blockchain and $\mathsf{params} = (1^{\ell_1}, 1^{\ell_2}, 1^{\ell_3}, 1^{\ell_4}, \alpha, \beta)$. First, we know that since π gets verified, $\mathbf{B}$ and $\overline{\mathbf{B}}$ are consistent (i.e., $\overline{\mathbf{B}}^{\lceil \ell_1} \preceq \mathbf{B}$ and $|\overline{\mathbf{B}}| \leq |\mathbf{B}|$). Now assuming NIWI soundness condition holds, we know that if π' gets verified with non-negligible probability, then the statement it proves must also be true with non-negligible probability. Since $x \notin \mathcal{L}$, this suggests that the cheating prover used the witness $\{\mathsf{sh}_i, r_i\}_{i \in \mathcal{M}}, s$ such that

$$\{\mathsf{sh}_i\}_i = \mathsf{Share}(\widetilde{\mathbf{B}}, \{\mathsf{stake}_{\mathsf{id}}\}_{\mathsf{id}}, \beta \cdot \mathsf{stake}_{\mathsf{total}}; s) \wedge$$
$$\forall\ i,\ \mathsf{ct}_{i,2} = \mathsf{Enc}(\mathsf{pk}_i, \mathsf{sh}_i; r_i)\ \wedge\ \mathsf{valid\text{-}fork}^V(\overline{\mathbf{B}}^{\lceil \ell_1}, \widetilde{\mathbf{B}}, \ell_3, \ell_4, 1-\alpha)$$

Therefore, the predicate $\mathsf{valid\text{-}fork}^V(\overline{\mathbf{B}}^{\lceil \ell_1}, \widetilde{\mathbf{B}}, \ell_3, \ell_4, 1-\alpha)$ must be satisfied. Now, since $\overline{\mathbf{B}}^{\lceil \ell_1} \preceq \mathbf{B}$ and $|\overline{\mathbf{B}}| \leq |\mathbf{B}|$, we could also write that $\mathsf{valid\text{-}fork}^V(\mathbf{B}^{\lceil \ell_1}, \widetilde{\mathbf{B}}, \ell_3, \ell_4, 1-\alpha)$ is also satisfied. This suggests that $\widetilde{\mathbf{B}}$ contains a valid fork w.r.t. to an honest party's blockchain $\mathbf{B}$. Thus, if a cheating prover with only α stake

ratio succeeds with non-negligible probability, then the blockchain protocol does not satisfy the $(1-\alpha, \ell_3, \ell_4)$-bounded stake forking property. More formally, below we sketch an extractor that controls at most $\alpha + \beta$ stake ratio, and it runs the prover to either learn the witness or break the assumption that the blockchain protocol satisfies bounded stake forking property.

For the rest of the analysis we condition on the event that in any ℓ_2 consecutive blocks the amount of honest stake contribution is always more than β. Recall that the (β, ℓ_2)-sufficient honest stake contribution property states that this condition is satisfied with all but negligible probability.

The extractor runs the prover with α stake ratio. The prover outputs a proof π of the form $(\pi', \overline{\mathbf{B}}, \{\mathsf{ct}_{i,1}, \mathsf{ct}_{i,2}\}_i, \mathsf{params})$ where $\mathsf{params} = (1^{\ell_1}, 1^{\ell_2}, 1^{\ell_3}, 1^{\ell_4}, \alpha, \beta)$. Let $\mathcal{M} = \mathsf{miner}(\overline{\mathbf{B}}^{\lceil \ell_1}, [\ell_2])$. By sufficient honest stake contribution property, we know that there exists a set $\mathcal{M}' \subseteq \mathcal{M}$ such that each party in $\mathcal{M}'$ is an honest party executing the blockchain protocol and the combined stake ratio of $\mathcal{M}'$ is at least β. Let $\mathcal{M}'$ be such a minimal set. The extractor using secret keys of all parties in $\mathcal{M}'$ decrypts the ciphertexts $\{\mathsf{ct}_{i,1}, \mathsf{ct}_{i,2}\}_i$ for all $i \in \mathcal{M}'$ to get the corresponding shares $\mathsf{sh}_{i,1}$ and $\mathsf{sh}_{i,2}$. Let $w = \mathsf{Rec}(\{\mathsf{sh}_{i,1}\}_{i \in \mathcal{M}'})$ and $\widetilde{\mathbf{B}} = \mathsf{Rec}(\{\mathsf{sh}_{i,2}\}_{i \in \mathcal{M}'})$. Now if π' gets verified, then either $\mathcal{R}(x, w) = 1$ or $\mathsf{valid\text{-}fork}^V(\overline{\mathbf{B}}^{\lceil \ell_1}, \widetilde{\mathbf{B}}, \ell_3, \ell_4, 1-\alpha)$ is true. Thus, either the extractor can extract the witness to prove that $x \in \mathcal{L}$, or it could generate a fork of length more than $\ell_3 + \ell_4$ such that proof-of-stake fraction *after* the first ℓ_3 blocks of the fork is at least $1 - \alpha$.

Thus if a cheating prover can prove a false statement (i.e., π verifies but $x \notin \mathcal{L}$), then a set of honest parties which control only β stake ratio can decrypt the corresponding ciphertexts $\mathsf{ct}_{i,2}$ and reconstruct the secret shares to obtain a forked blockchain $\widetilde{\mathbf{B}}$ thereby contradicting the fact that the blockchain protocol satisfies $(1-\alpha, \ell_3, \ell_4)$-bounded stake forking property. Therefore, assuming that bounded stake forking and sufficient honest stake contribution properties hold and the NIWI system is sound, then the NIZK system satisfies soundness as well.

Finally, the fact that the system is an argument of knowledge is immediate from the extraction procedure described above.

Zero-Knowledge. We start by sketching the simulator Sim. Let $\mathcal{A}$ be any real world adversary such that its stake ratio is at most α. Also, let $\mathbf{B}$ be the local blockchain of some honest party. Note that the simulator controls all honest parties executing the blockchain protocol. On input x and parameters $\mathsf{params} = (1^{\ell_1}, 1^{\ell_2}, 1^{\ell_3}, 1^{\ell_4}, \alpha, \beta)$, simulator Sim runs as follows.

- Let $\mathbf{B}' = \mathbf{B}^{\lceil \ell_1}$ and $\mathcal{M} = \mathsf{miner}(\mathbf{B}', [\ell_2])$. Sim extends the blockchain $\mathbf{B}'$ with ℓ $(\geq \ell_1 + \ell_2)$ *dummy* blocks such that each new block satisfies the validity predicate V and is accompanied by a proof-of-stake generated by Sim using signing keys of the honest parties such that the proof-of-stake fraction in the last $\ell - \ell_1$ blocks is at least $1 - \alpha$. Let $\widetilde{\mathbf{B}}$ be the extended blockchain generated by Sim.
- It secret shares an all zeros string and blockchain $\widetilde{\mathbf{B}}$ into $|\mathcal{M}|$ shares with weights $\{\mathsf{stake}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{M}}$ and threshold $\beta \cdot \mathsf{stake}_{\mathsf{total}}$ as follows

$$\{\mathsf{sh}_{\mathsf{id},1}\}_{\mathsf{id}} = \mathsf{Share}(0, \{\mathsf{stake}_{\mathsf{id}}\}_{\mathsf{id}}, \beta \cdot \mathsf{stake}_{\mathsf{total}}; s_1), \quad \{\mathsf{sh}_{\mathsf{id},2}\}_{\mathsf{id}} = \mathsf{Share}(\widetilde{\mathbf{B}}, \{\mathsf{stake}_{\mathsf{id}}\}_{\mathsf{id}}, \beta \cdot \mathsf{stake}_{\mathsf{total}}; s_2).$$

- It encrypts all these shares as follows

$$\forall\ \mathsf{id} \in \mathcal{M}, \quad \mathsf{ct}_{\mathsf{id},1} = \mathsf{Enc}(\mathsf{pk}_{\mathsf{id}}, \mathsf{sh}_{\mathsf{id},1}; r_{\mathsf{id},1}), \quad \mathsf{ct}_{\mathsf{id},2} = \mathsf{Enc}(\mathsf{pk}_{\mathsf{id}}, \mathsf{sh}_{\mathsf{id},2}; r_{\mathsf{id},2}).$$

- Next it computes a NIWI proof π' as done by an honest prover $\mathcal{P}$, but instead using $\left(\{\mathsf{sh}_{\mathsf{id},2}, r_{\mathsf{id},2}\}_{\mathsf{id}}, s_2\right)$ as the witness.
- Finally, it outputs the proof π as

$$\pi = \left(\pi', \mathbf{B}, \{\mathsf{ct}_{\mathsf{id},1}, \mathsf{ct}_{\mathsf{id},2}\}_{\mathsf{id}}, \mathsf{params} = (1^{\ell_1}, 1^{\ell_2}, 1^{\ell_3}, 1^{\ell_4}, \alpha, \beta)\right).$$

Remark 6.1. Since the adversarial stake ratio is at most α, thus the amount of stake controlled by honest parties is at least $1 - \alpha$. Therefore, the simulator Sim could, in polynomial time, extend $\mathbf{B}'$ with a sequence of ℓ dummy blocks as described above as it controls all the honest parties executing the protocol.

Next, we describe a sequence of games. The output of each game is the proof π and the adversary $\mathcal{A}$'s transcript.

In the first game, the proof π is honestly generated as in real execution, i.e. by running the NIZK prover algorithm $\mathcal{V}$ with parameters params, blockchain $\mathbf{B}$, instance x and witness w. In the last game, π is generated as in the case of simulated execution. In the following games, let $\mathbf{B}' = \mathbf{B}^{\lceil \ell_1}$ and $\mathcal{M} = \mathsf{miner}(\mathbf{B}', [\ell_2])$.

Game 1: In this game, the proof π is generated as follows.

1. The prover secret shares the witness w and an all zeros string into $|\mathcal{M}|$ shares with weights $\{\mathsf{stake}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{M}}$ and threshold $\beta \cdot \mathsf{stake}_{\mathsf{total}}$ as follows

$$\{\mathsf{sh}_{\mathsf{id},1}\}_{\mathsf{id}} = \mathsf{Share}(w, \{\mathsf{stake}_{\mathsf{id}}\}_{\mathsf{id}}, \beta \cdot \mathsf{stake}_{\mathsf{total}}; s_1), \quad \{\mathsf{sh}_{\mathsf{id},2}\}_{\mathsf{id}} = \mathsf{Share}(0, \{\mathsf{stake}_{\mathsf{id}}\}_{\mathsf{id}}, \beta \cdot \mathsf{stake}_{\mathsf{total}}; s_2).$$

2. It encrypts all these shares as follows

$$\forall\ \mathsf{id} \in \mathcal{M}, \quad \mathsf{ct}_{\mathsf{id},1} = \mathsf{Enc}(\mathsf{pk}_{\mathsf{id}}, \mathsf{sh}_{\mathsf{id},1}; r_{\mathsf{id},1}), \quad \mathsf{ct}_{\mathsf{id},2} = \mathsf{Enc}(\mathsf{pk}_{\mathsf{id}}, \mathsf{sh}_{\mathsf{id},2}; r_{\mathsf{id},2}).$$

3. Next it computes a NIWI proof π' using $\left(\{\mathsf{sh}_{\mathsf{id},1}, r_{\mathsf{id},1}\}_{\mathsf{id}}, s_1\right)$ as the witness.

4. Finally, it outputs the proof π as

$$\pi = \left(\pi', \mathbf{B}, \{\mathsf{ct}_{\mathsf{id},1}, \mathsf{ct}_{\mathsf{id},2}\}_{\mathsf{id}}, \mathsf{params} = (1^{\ell_1}, 1^{\ell_2}, 1^{\ell_3}, 1^{\ell_4}, \alpha, \beta)\right).$$

In the following games, let $\widetilde{\mathbf{B}}$ be the blockchain such that $\mathsf{valid\text{-}fork}^V(\mathbf{B}^{\lceil \ell_1}, \widetilde{\mathbf{B}}, \ell_3, \ell_4, 1-\alpha)$ and $\mathcal{M}_{\mathcal{A}}$ be the subset of corrupt parties in $\mathcal{M}$, i.e. those which are controlled by the adversary $\mathcal{A}$.

Game 2: This game is same as the previous game, except the prover computes ciphertexts $\mathsf{ct}_{\mathsf{id},2}$ as encryptions of zero instead of $\mathsf{sh}_{\mathsf{id},2}$ for all $\mathsf{id} \in \mathcal{M} \setminus \mathcal{M}_{\mathcal{A}}$.

2. It encrypts all these shares as follows

$$\begin{aligned} &\forall\ \mathsf{id} \in \mathcal{M}_{\mathcal{A}}, && \mathsf{ct}_{\mathsf{id},1} = \mathsf{Enc}(\mathsf{pk}_{\mathsf{id}}, \mathsf{sh}_{\mathsf{id},1}; r_{\mathsf{id},1}), \quad \mathsf{ct}_{\mathsf{id},2} = \mathsf{Enc}(\mathsf{pk}_{\mathsf{id}}, \mathsf{sh}_{\mathsf{id},2}; r_{\mathsf{id},2}), \\ &\forall\ \mathsf{id} \in \mathcal{M} \setminus \mathcal{M}_{\mathcal{A}}, && \mathsf{ct}_{\mathsf{id},1} = \mathsf{Enc}(\mathsf{pk}_{\mathsf{id}}, \mathsf{sh}_{\mathsf{id},1}; r_{\mathsf{id},1}), \quad \mathsf{ct}_{\mathsf{id},2} = \mathsf{Enc}(\mathsf{pk}_{\mathsf{id}}, 0; r'_{\mathsf{id},2}). \end{aligned}$$

Game 3: This game is same as the previous game, except the prover computes the shares $\mathsf{sh}_{\mathsf{id},2}$ as secret shares for blockchain $\widetilde{\mathbf{B}}$ instead.

1. The prover secret shares the witness w and blockchain $\widetilde{\mathbf{B}}$ into $|\mathcal{M}|$ shares with weights $\{\mathsf{stake}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{M}}$ and threshold $\beta \cdot \mathsf{stake}_{\mathsf{total}}$ as follows

$$\{\mathsf{sh}_{\mathsf{id},1}\}_{\mathsf{id}} = \mathsf{Share}(w, \{\mathsf{stake}_{\mathsf{id}}\}_{\mathsf{id}}, \beta \cdot \mathsf{stake}_{\mathsf{total}}; s_1), \quad \{\mathsf{sh}_{\mathsf{id},2}\}_{\mathsf{id}} = \mathsf{Share}(\widetilde{\mathbf{B}}, \{\mathsf{stake}_{\mathsf{id}}\}_{\mathsf{id}}, \beta \cdot \mathsf{stake}_{\mathsf{total}}; s_2).$$

Game 4: This game is same as the previous game, except the prover computes ciphertexts $\mathsf{ct}_{\mathsf{id},2}$ as encryptions of $\mathsf{sh}_{\mathsf{id},2}$ instead of zero for all $\mathsf{id} \in \mathcal{M} \setminus \mathcal{M}_{\mathcal{A}}$.

2. It encrypts all these shares as follows

$$\forall\ \mathsf{id} \in \mathcal{M}, \quad \mathsf{ct}_{\mathsf{id},1} = \mathsf{Enc}(\mathsf{pk}_{\mathsf{id}}, \mathsf{sh}_{\mathsf{id},1}; r_{\mathsf{id},1}), \quad \mathsf{ct}_{\mathsf{id},2} = \mathsf{Enc}(\mathsf{pk}_{\mathsf{id}}, \mathsf{sh}_{\mathsf{id},2}; r_{\mathsf{id},2}).$$

Game 5: This game is same as the previous game, except the witness used for generating the NIWI proof is switched.

3. Next it computes a NIWI proof π' using $\left(\{\mathsf{sh}_{\mathsf{id},2}, r_{\mathsf{id},2}\}_{\mathsf{id}}, s_2\right)$ as the witness.

Game 6: This game is same as the previous game, except the prover computes ciphertexts $\mathsf{ct}_{\mathsf{id},1}$ as encryptions of zero instead of $\mathsf{sh}_{\mathsf{id},1}$ for all $\mathsf{id} \in \mathcal{M} \setminus \mathcal{M}_{\mathcal{A}}$.

2. It encrypts all these shares as follows

$$\begin{aligned} &\forall\ \mathsf{id} \in \mathcal{M}_{\mathcal{A}}, && \mathsf{ct}_{\mathsf{id},1} = \mathsf{Enc}(\mathsf{pk}_{\mathsf{id}}, \mathsf{sh}_{\mathsf{id},1}; r_{\mathsf{id},1}), \quad \mathsf{ct}_{\mathsf{id},2} = \mathsf{Enc}(\mathsf{pk}_{\mathsf{id}}, \mathsf{sh}_{\mathsf{id},2}; r_{\mathsf{id},2}), \\ &\forall\ \mathsf{id} \in \mathcal{M} \setminus \mathcal{M}_{\mathcal{A}}, && \mathsf{ct}_{\mathsf{id},1} = \mathsf{Enc}(\mathsf{pk}_{\mathsf{id}}, 0; r'_{\mathsf{id},1}), \quad \mathsf{ct}_{\mathsf{id},2} = \mathsf{Enc}(\mathsf{pk}_{\mathsf{id}}, \mathsf{sh}_{\mathsf{id},2}; r_{\mathsf{id},2}). \end{aligned}$$

Game 7: This game is same as the previous game, except the prover computes the shares $\mathsf{sh}_{\mathsf{id},1}$ as secret shares for an all zeros string instead of witness w.

1. The prover secret shares an all zeros string and blockchain $\widetilde{\mathbf{B}}$ into $|\mathcal{M}|$ shares with weights $\{\mathsf{stake}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{M}}$ and threshold $\beta \cdot \mathsf{stake}_{\mathsf{total}}$ as follows

$$\{\mathsf{sh}_{\mathsf{id},1}\}_{\mathsf{id}} = \mathsf{Share}(0, \{\mathsf{stake}_{\mathsf{id}}\}_{\mathsf{id}}, \beta \cdot \mathsf{stake}_{\mathsf{total}}; s_1), \quad \{\mathsf{sh}_{\mathsf{id},2}\}_{\mathsf{id}} = \mathsf{Share}(\widetilde{\mathbf{B}}, \{\mathsf{stake}_{\mathsf{id}}\}_{\mathsf{id}}, \beta \cdot \mathsf{stake}_{\mathsf{total}}; s_2).$$

Game 8: This game is same as the previous game, except the prover computes ciphertexts $\mathsf{ct}_{\mathsf{id},1}$ as encryptions of $\mathsf{sh}_{\mathsf{id},1}$ for all $\mathsf{id} \in \mathcal{M} \setminus \mathcal{M}_{\mathcal{A}}$.

2. It encrypts all these shares as follows

$$\forall\ \mathsf{id} \in \mathcal{M}, \quad \mathsf{ct}_{\mathsf{id},1} = \mathsf{Enc}(\mathsf{pk}_{\mathsf{id}}, \mathsf{sh}_{\mathsf{id},1}; r_{\mathsf{id},1}), \quad \mathsf{ct}_{\mathsf{id},2} = \mathsf{Enc}(\mathsf{pk}_{\mathsf{id}}, \mathsf{sh}_{\mathsf{id},2}; r_{\mathsf{id},2}).$$

■

6.2.1 Analysis

We now establish via a sequence of claims that no PPT adversary can distinguish between any two adjacent games with non-negligible advantage. Since the last game is identical to the simulated execution, therefore establishing non-negligible gap between adjacent games is sufficient.

Let $\mathcal{D}$ be any successful PPT adversary that tries to distinguish between any two successive games. Let $\mathsf{Adv}^i_{\mathcal{D}}$ denote the advantage of algorithm $\mathcal{D}$ in distinguishing between Games i and $i+1$. We show via a sequence of claims that $\mathsf{Adv}^i_{\mathcal{D}}$ is negligible for all $i \leq 7$. Below we discuss our claims in detail.

Claim 6.1. If HS is a secure integrated public key encryption-signature scheme, then for any PPT adversary $\mathcal{D}$, $\mathsf{Adv}^1_{\mathcal{D}}$ is negligible in the security parameter λ.

Proof. For proving indistinguishability of Games 1 and 2, we need to sketch n intermediate hybrid games between these two, where $n = |\mathcal{M} \setminus \mathcal{M}_{\mathcal{A}}|$. Observe that in Game 1, ciphertexts $\mathsf{ct}_{\mathsf{id},2}$ for all $\mathsf{id} \notin \mathcal{M}_{\mathcal{A}}$ are encryptions of shares $\mathsf{sh}_{\mathsf{id},2}$; however, in Game 2, they are encryptions of $\mathsf{sh}'_{\mathsf{id},2}$. The high-level proof idea is to switch $\mathsf{ct}_{\mathsf{id},2}$ from encryptions of $\mathsf{sh}_{\mathsf{id},2}$ to encryptions of $\mathsf{sh}'_{\mathsf{id},2}$ one-at-a-time. Intuitively, this could be done because the adversary $\mathcal{A}$ does not control the parties not in $\mathcal{M}_{\mathcal{A}}$, therefore it does not know the corresponding secret keys and hence must break security of the underlying encryption scheme.

Concretely, i^{th} intermediate hybrid between Game 1 and 2 proceeds same as Game 1 except that the (lexicographically) first i ciphertexts $\mathsf{ct}_{\mathsf{id},2}$ are computed as $\mathsf{ct}_{\mathsf{id},2} \leftarrow \mathsf{Enc}(\mathsf{pk}_{\mathsf{id}}, \mathsf{sh}'_{\mathsf{id},2})$, and remaining as $\mathsf{ct}_{\mathsf{id},2} \leftarrow \mathsf{Enc}(\mathsf{pk}_{\mathsf{id}}, \mathsf{sh}_{\mathsf{id},2})$. For the analysis, Game 1 is regarded as 0^{th} intermediate hybrid, and Game 2 is regarded as n^{th} intermediate hybrid. $\mathcal{D}$'s advantage in distinguishing any pair of consecutive intermediate hybrid is negligibly small as otherwise it directly reduces to an attack on IND-CPA security of encryption scheme. Thus, the claim follows. ■

Claim 6.2. If SS is a weighted threshold secret sharing scheme, then for any PPT adversary $\mathcal{D}$, $\mathsf{Adv}^2_{\mathcal{D}}$ is negligible in the security parameter λ.

Proof. The proof of this claim follows directly from the secrecy property of the weighted threshold secret sharing scheme. Note that the adversary only learns the shares encrypted under public keys corresponding to set $\mathcal{M}_{\mathcal{A}}$. Since $\mathcal{A}$'s stake ratio is at most $\alpha < \beta$ and the combined weight of shares addressed to $\mathcal{M}_{\mathcal{A}}$ is $\sum_{\mathsf{id} \in \mathcal{M}_{\mathcal{A}}} \mathsf{stake}_{\mathsf{id}} \leq \mathsf{stake}_{\mathcal{A}} \leq \alpha$, therefore using secrecy property of weighted threshold secret sharing scheme, we can conclude that

$$\left\{ \{\mathsf{sh}_{\mathsf{id},2}\}_{\mathsf{id} \in \mathcal{M}_{\mathcal{A}}} : \{\mathsf{sh}_{\mathsf{id},2}\}_{\mathsf{id}} \leftarrow \mathsf{Share}(0, \{\mathsf{stake}_{\mathsf{id}}\}_{\mathsf{id}}, \beta \cdot \mathsf{stake}_{\mathsf{total}}) \right\}$$
$$\approx_c$$
$$\left\{ \{\mathsf{sh}_{\mathsf{id},2}\}_{\mathsf{id} \in \mathcal{M}_{\mathcal{A}}} : \{\mathsf{sh}_{\mathsf{id},2}\}_{\mathsf{id}} \leftarrow \mathsf{Share}(\widetilde{\mathbf{B}}, \{\mathsf{stake}_{\mathsf{id}}\}_{\mathsf{id}}, \beta \cdot \mathsf{stake}_{\mathsf{total}}) \right\}.$$

Thus, the claim follows. ∎

Claim 6.3. If HS is a secure integrated public key encryption-signature scheme, then for any PPT adversary $\mathcal{D}$, $\mathsf{Adv}^3_{\mathcal{D}}$ is negligible in the security parameter λ.

Proof. The proof of this claim is identical to the proof of Claim 6.1. ∎

Claim 6.4. If $(\mathcal{P}_{\mathsf{NIWI}}, \mathcal{V}_{\mathsf{NIWI}})$ is a NIWI argument system for **NP**, then for any PPT adversary $\mathcal{D}$, $\mathsf{Adv}^4_{\mathcal{D}}$ is negligible in $|x|$.

Proof. Note that in Games 4 and 5, both $\left(\{\mathsf{sh}_{\mathsf{id},1}, r_{\mathsf{id},1}\}_{\mathsf{id}}, s_1\right)$ and $\left(\{\mathsf{sh}_{\mathsf{id},2}, r_{\mathsf{id},2}\}_{\mathsf{id}}, s_2\right)$ are valid witnesses. Therefore, the proof of this claim follows directly from the witness indistinguishability property of the NIWI argument system. ∎

Claim 6.5. If HS is a secure integrated public key encryption-signature scheme, then for any PPT adversary $\mathcal{D}$, $\mathsf{Adv}^5_{\mathcal{D}}$ is negligible in the security parameter λ.

Proof. The proof of this claim is identical to the proof of Claim 6.1. ∎

Claim 6.6. If $(\mathsf{Share}, \mathsf{Rec})$ is a weighted threshold secret sharing scheme, then for any PPT adversary $\mathcal{D}$, $\mathsf{Adv}^6_{\mathcal{D}}$ is negligible in the security parameter λ.

Proof. The proof of this claim is identical to the proof of Claim 6.2. ∎

Claim 6.7. If HS is a secure integrated public key encryption-signature scheme, then for any PPT adversary $\mathcal{D}$, $\mathsf{Adv}^7_{\mathcal{D}}$ is negligible in the security parameter λ.

Proof. The proof of this claim is identical to the proof of Claim 6.1. ∎

7 One-Time Programs over Blockchain

In this section, we provide our construction for one-time compilers from garbled circuits and extractable witness encryption over any POS based blockchain protocol. Below we describe the main ideas.

Outline. Suppose the blockchain protocol satisfies $(\alpha, \beta, \ell_1, \ell_2)$-distinguishable forking property. We know that distinguishable forking property suggests that no PPT adversary can generate a fork of length $\geq \ell_1 + \ell_2$ such that the proof-of-stake fraction after the first ℓ_1 blocks of the fork is more than α. Additionally, it also implies that the proof-of-stake fraction in any ℓ_2 consecutive blocks in an honest party's blockchain will be at least β, with β being non-negligibly higher than α.

At a high level, the scheme works as follows. To compile a circuit C over blockchain $\mathbf{B}$, the compilation algorithm first garbles the circuit to compute a garbled circuit and wire keys. Suppose we encrypt the wire

keys using public key encryption and set the corresponding one-time program as the garbled circuit and encrypted wire keys. This suggests that the evaluator must interact with the compiling party to be able to evaluate the program. However, one-time programs are not defined in an interactive setting. Therefore, we need to somehow allow conditional release/ conditional decryption of encrypted wire keys for evaluation. Additionally, we need to make sure that the evaluator only learns the wire keys corresponding to exactly *one* input as otherwise it will not satisfy the one-time secrecy condition. To this end, we encrypt the wire keys using witness encryption scheme such that, to decrypt the wire keys, the evaluator needs to produce a blockchain $\mathbf{B}'$ as a witness where $\mathbf{B}'$ must satisfy the following conditions — (1) there exists a block in $\mathbf{B}'$ which contains the input (on which evaluator wants to evaluate the circuit), (2) there are at least $\ell_1 + \ell_2$ more blocks after the input block such that the proof-of-stake fraction in the last ℓ_2 blocks of $\mathbf{B}'$ is more than β, and (3) there does not exists any other block which posts a different input.

To evaluate such a compiled program, the evaluator needs to post its input on the blockchain, and then wait for it to get added to blockchain and get extended by $\ell_1 + \ell_2$ blocks. Afterwards, it could simply use its blockchain as a witness to decrypt appropriate wire keys and then evaluate the garbled circuit using those keys. Intuitively, this would satisfy the one-time secrecy property because in order to evaluate the program on a second input the adversary needs to fork the blockchain before the input block. Now, since the distinguishable forking property guarantees that no PPT adversary can generate such a fork (of length more than $\ell_1 + \ell_2$) with non-negligible probability, therefore one-time secrecy follows.

We start by describing the **NP** language for which we assume existence of a secure extractable witness encryption scheme. Next we develop our one-time compilers on top of a blockchain protocol, and finally show our construction satisfies one-time secrecy property.

7.1 NP Relation on Blockchain Protocols

Let $\Gamma = (\mathsf{UpdateState}, \mathsf{GetRecords}, \mathsf{Broadcast})$ be a blockchain protocol with validity predicate V. Consider the following relation.

Definition 7.1. Let $\mathcal{R}_{\Gamma^V}$ be a relation on the blockchain protocol Γ^V. The instances and witnesses satisfying the relation are of the form

$$x = (1^\lambda, \mathsf{st}, 1^{\ell_1}, 1^{\ell_2}, 1^n, \beta, i, b, \mathsf{uid}), \quad w = \widetilde{\mathsf{st}}.$$

Let $\mathbf{B} = \mathsf{GetRecords}(1^\lambda, \mathsf{st})$ and $\widetilde{\mathbf{B}} = \mathsf{GetRecords}(1^\lambda, \widetilde{\mathsf{st}})$. The instance-witness pair satisfies the relation $((x, w) \in \mathcal{R}_{\Gamma^V})$ *if and only if* all the following properties are satisfied:

- Blockchains $\mathbf{B}$ and $\widetilde{\mathbf{B}}$ are valid, i.e. $V(\mathbf{B}) = V(\widetilde{\mathbf{B}}) = 1$
- $\mathbf{B}$ is a prefix of $\widetilde{\mathbf{B}}$, i.e. they are consistent[18]
- There exists a *unique* block $B^* \in \widetilde{\mathbf{B}} \setminus \mathbf{B}$ such that the following are satisfied
  - There exists a *unique* record m^* in B^* such that $m^* = (\mathsf{uid}, y)$, y is an n-bit string and $y_i = b$
  - Let ℓ' be the number of blocks in blockchain $\widetilde{\mathbf{B}}$ after block B^*, i.e. $B^* \in \widetilde{\mathbf{B}}^{\lceil \ell'}$. It should hold that $\ell' \geq \ell_1 + \ell_2$ and $\mathsf{u\text{-}stakefrac}(\widetilde{\mathbf{B}}, \ell' - \ell_1) > \beta$

Remark 7.1. The *uniqueness* of block B^* and record m^* is defined in the following way. There must not exist any other block (i.e., apart from B^*) in the entire witness blockchain $\widetilde{\mathbf{B}}$ such that it contains a record m of the form (uid, z) where z is any n-bit string. Similarly, there must not exist any record m other than m^* in block B^* that satisfies the same property.

Let $\mathcal{L}_{\Gamma^V}$ be the language specified by the relation $\mathcal{R}_{\Gamma^V}$. This language is in **NP** because verifying validity of blockchains take only polynomial time and all the properties in Definition 7.1 could also be verified simultaneously.

[18] Formally, the consistency should be checked as $\mathbf{B}^{\lceil \kappa} \preceq \widetilde{\mathbf{B}}$ for an appropriate value of parameter κ (Definition 4.1), however for ease of exposition we avoid it.

7.2 One-Time Compilers

Let $\Gamma^V = (\mathsf{UpdateState}^V, \mathsf{GetRecords}, \mathsf{Broadcast})$ be a blockchain protocol, and $\mathsf{GC} = (\mathsf{GC.Garble}, \mathsf{GC.Eval})$ be a garbling scheme for circuit family $\mathcal{C} = \{\mathcal{C}_n\}_n$, and $\mathsf{WE} = (\mathsf{Enc}, \mathsf{Dec})$ be a witness encryption scheme for language $\mathcal{L}_{\Gamma^V}$. Below we describe our one-time compilers $\mathsf{OTC} = (\mathsf{Compile}, \mathsf{Eval})$ for circuit family $\mathcal{C} = \{\mathcal{C}_n\}_n$ in the blockchain model.

- $\mathsf{Compile}(1^\lambda, 1^{\ell_1}, 1^{\ell_2}, \beta, C \in \mathcal{C}_n)$: The compilation algorithm first garbles the circuit C by computing $(G, \{w_{i,b}\}_{i\le n, b\in\{0,1\}}) \leftarrow \mathsf{GC.Garble}(1^\lambda, C)$. Next, it encrypts each of the wire keys $w_{i,b}$ separately under instances $x_{i,b}$ as follows:

  $$\forall i \le n, b \in \{0,1\}, \quad x_{i,b} = (1^\lambda, \mathsf{st}, 1^{\ell_1}, 1^{\ell_2}, 1^n, \beta, i, b, \mathsf{uid} = G), \quad \mathsf{ct}_{i,b} \leftarrow \mathsf{Enc}(1^\lambda, x_{i,b}, w_{i,b}),$$

  where st is its local blockchain state. Finally, it sets the compiled circuit as $CC = \left(1^\lambda, 1^{\ell_1}, 1^{\ell_2}, G, \{\mathsf{ct}_{i,b}\}_{i\le n, b\in\{0,1\}}\right)$.

- $\mathsf{Eval}(CC, y \in \{0,1\}^n)$: Let $CC = \left(1^\lambda, 1^{\ell_1}, 1^{\ell_2}, G, \{\mathsf{ct}_{i,b}\}_{i\le n, b\in\{0,1\}}\right)$. It first posts input y on the blockchain by running $\mathsf{Broadcast}$ algorithm as $\mathsf{Broadcast}(1^\lambda, (G, y))$.

  It runs the $\mathsf{UpdateState}$ algorithm, and waits for message (G, y) to be posted on the blockchain and further the chain to be extended by $\ell_1 + \ell_2$ blocks. After the blockchain gets extended, it uses its own local state st as a witness to decrypt the wire keys corresponding to input y as

  $$\forall i \le n, \quad w_i = \mathsf{Dec}(\mathsf{ct}_{i,y_i}, \mathsf{st}).$$

  It then uses these n wire keys to evaluate the garbled circuit, and outputs $\mathsf{GC.Eval}(G, \{w_i\}_{i\le n})$. If the witnes decryption fails (outputs $\perp$), then it also outputs $\perp$.

Correctness. Fix any λ, n, ℓ_1, ℓ_2, β, and circuit $C \in \mathcal{C}_n$. Let $(G, \{w_{i,b}\}) \leftarrow \mathsf{GC.Garble}(1^\lambda, C)$, $x_{i,b} = (1^\lambda, \mathsf{st}, 1^{\ell_1}, 1^{\ell_2}, 1^n, \beta, i, b, G)$, and $\mathsf{ct}_{i,b} \leftarrow \mathsf{Enc}(1^\lambda, x_{i,b}, w_{i,b})$.

For any input $y \in \{0,1\}^n$, consider that an evaluator runs $\mathsf{Broadcast}$ algorithm to post (G, y) on the blockchain. Let $\widetilde{\mathsf{st}}$ be the local state of the evaluator after message (G, y) is posted on blockchain and it is extended by $\ell_1+\ell_2$ blocks. Assuming that evaluator and compiler's blockchain are consistent (Definition 4.1), then with all but negligible probability for all $i \le n$, $\widetilde{\mathsf{st}}$ could be used as the witness to decrypt ciphertexts ct_{i,y_i} as $(x_{i,y_i}, \widetilde{\mathsf{st}}) \in \mathcal{R}_{\Gamma^V}$. This is true because consistency property guarantees that, with all but negligible probability, the blockchains $\mathbf{B}$ and $\widetilde{\mathbf{B}}$ will be consistent. Additionally, the stake quantity property (Definition 4.4) guarantees that (with all but negligible probability) the condition $\mathsf{u\text{-}stakefrac}(\widetilde{\mathbf{B}}, \ell' - \ell_1) > \beta$ will be satisfied. Therefore, $\mathsf{Dec}(\mathsf{ct}_{i,y_i}, \mathsf{st}) = w_{i,y_i}$ which follows from correctness of the witness encryption scheme. Finally, $\mathsf{GC.Eval}(G, \{w_{i,y_i}\}_{i\le n}) = C(y)$ as it follows from correctness of the garbling scheme. Therefore, OTC satisfies the one-time compiler correctness condition.

Remark 7.2. Our one-time compiler takes additional parameters ℓ_1, ℓ_2 and β as inputs, which we refer to as the *hardness parameters*. The primary purpose of ℓ_1, ℓ_2 and β is to connect the efficiency of our compiled circuit to an appropriate hardness assumption on the blockchain protocol. Informally, increasing value of ℓ_1 and ℓ_2 reduces efficiency of our compiled circuit as the evaluator needs to wait for longer time (more blocks) in order to evaluate the circuit. At the same time, reducing ℓ_1 and ℓ_2 increases the strength of the assumption on the blockchain. The latter will get highlighted in the security proof. The effect of choice of β has an indirect impact on efficiency, although it affects the same way as ℓ_1, ℓ_2.

7.3 Security Proof

We will now show that the one-time compiler described in Section 7.2 achieves one-time secrecy as per Definition 3.10. More formally, we prove the following theorem where all the parameters are polynomials in the security parameter λ.

Theorem 7.1. If GC is a secure garbling scheme (Definition 3.6) for circuit family $\{\mathcal{C}_n\}_n$, WE is extractable secure witness encryption scheme (Definition 3.7) for language $\mathcal{L}_{\Gamma^V}$, and $(\alpha, \beta, \ell_1, \ell_2)$-distinguishable forking property (Definition 4.7) holds for blockchain protocol Γ^V, then OTC with hardness parameters ℓ_1, ℓ_2 and β is a B/C-*selectively*-secure one-time compiler (Definition 3.10) for circuit family $\{\mathcal{C}_n\}_n$ against all PPT adversaries with at most α stake ratio.

To prove above theorem, we first sketch the simulator Sim. Let $\mathcal{A}$ be any real world adversary.

- Sim obtains the input y^* from $\mathcal{A}$, and also obtains $(1^n, 1^{|C|}, y')$ during initialization. It aborts if $y' \neq y^*$.
- It queries the (one-time access) oracle at input y^* to receive $C(y^*)$.
- It computes the garbled circuit G as $(G, \{w_i\}_{i\leq n}) \leftarrow \mathsf{GC.Sim}(1^\lambda, 1^n, 1^{|C|}, C(y^*))$.
- Next, it computes ciphertexts $\mathsf{ct}_{i,b}$ as follows:

$$\forall i \leq n, b = y^*_i, \quad x_{i,b} = (1^\lambda, \mathsf{st}, 1^{\ell_1}, 1^{\ell_2}, 1^n, \beta, i, b, G), \quad \mathsf{ct}_{i,b} \leftarrow \mathsf{Enc}(1^\lambda, x_{i,b}, w_i),$$
$$\forall i \leq n, b \neq y^*_i, \quad x_{i,b} = (1^\lambda, \mathsf{st}, 1^{\ell_1}, 1^{\ell_2}, 1^n, \beta, i, b, G), \quad \mathsf{ct}_{i,b} \leftarrow \mathsf{Enc}(1^\lambda, x_{i,b}, \mathbf{0}),$$

  where st is its local blockchain state.
- It sets the compiled circuit as $CC = \big(1^\lambda, 1^{\ell_1}, 1^{\ell_2}, G, \{\mathsf{ct}_{i,b}\}_{i\leq n, b\in\{0,1\}}\big)$, and sends CC to $\mathcal{A}$.
- Finally, it outputs whatever $\mathcal{A}$ outputs.

Remark 7.3. Adversary $\mathcal{A}$ could also interact with other parties in the blockchain execution. Sim would simply forward all of $\mathcal{A}$'s network messages during simulation and honestly simulate the blockchain network, except if $\mathcal{A}$ broadcasts any message of form (G, y) with $y \neq y^*$ *before* broadcasting (G, y^*), then it aborts.

Next, we describe a sequence of games. In the following games, $\mathcal{A}$ can also participate in the blockchain execution with the only restriction that it is not allowed to broadcast any message of form (G, y) with $y \neq y^*$ *before* broadcasting (G, y^*).

Game 1: In this game, the real world adversary $\mathcal{A}$ interacts with the one-time compiler challenger.

1. $\mathcal{A}$ sends its input y^* to the challenger.
2. The challenger garbles the circuit C by computing $(G, \{w_{i,b}\}_{i\leq n, b\in\{0,1\}}) \leftarrow \mathsf{GC.Garble}(1^\lambda, C)$. Next, it encrypts each of the wire keys $w_{i,b}$ as follows:

$$\forall i \leq n, b \in \{0,1\}, \quad x_{i,b} = (1^\lambda, \mathsf{st}, 1^{\ell_1}, 1^{\ell_2}, 1^n, \beta, i, b, G), \quad \mathsf{ct}_{i,b} \leftarrow \mathsf{Enc}(1^\lambda, x_{i,b}, w_{i,b}),$$

   where st is its local blockchain state. Finally, it sets the compiled circuit as $CC = \big(1^\lambda, 1^{\ell_1}, 1^{\ell_2}, G, \{\mathsf{ct}_{i,b}\}_{i\leq n, b\in\{0,1\}}\big)$, and sends CC to $\mathcal{A}$.
3. $\mathcal{A}$ receives the compiled circuit CC from the challenger, and outputs its complete transcript.

Game 2: This game is same as previous game, except the challenger correctly encrypts the wire keys corresponding to input y^* only and creates rest of the n ciphertexts as encryptions of zeros.

1. $\mathcal{A}$ sends its input y^* to the challenger.

2. The challenger garbles the circuit C by computing $(G, \{w_{i,b}\}_{i\leq n, b\in\{0,1\}}) \leftarrow \mathsf{GC.Garble}(1^\lambda, C)$. Next, it computes ciphertexts $\mathsf{ct}_{i,b}$ as follows:

$$\forall i \leq n, b = y_i^*, \quad x_{i,b} = (1^\lambda, \mathsf{st}, 1^{\ell_1}, 1^{\ell_2}, 1^n, \beta, i, b, G), \quad \mathsf{ct}_{i,b} \leftarrow \mathsf{Enc}(1^\lambda, x_{i,b}, w_{i,b}),$$
$$\forall i \leq n, b \neq y_i^*, \quad x_{i,b} = (1^\lambda, \mathsf{st}, 1^{\ell_1}, 1^{\ell_2}, 1^n, \beta, i, b, G), \quad \mathsf{ct}_{i,b} \leftarrow \mathsf{Enc}(1^\lambda, x_{i,b}, \mathbf{0}),$$

where st is its local blockchain state. Finally, it sets the compiled circuit as $CC = \left(1^\lambda, 1^{\ell_1}, 1^{\ell_2}, G, \{\mathsf{ct}_{i,b}\}_{i\leq n, b\in\{0,1\}}\right)$, and sends CC to $\mathcal{A}$.

3. $\mathcal{A}$ receives the compiled circuit CC from the challenger, and outputs its complete transcript.

Game 3: This game is same as previous game, except the challenger simulates the circuit instead of garbling.

1. $\mathcal{A}$ sends its input y^* to the challenger.

2. The challenger computes the garbled circuit G as $(G, \{w_i\}_{i\leq n}) \leftarrow \mathsf{GC.Sim}(1^\lambda, 1^n, 1^{|C|}, C(y^*))$. Next, it computes ciphertexts $\mathsf{ct}_{i,b}$ as follows:

$$\forall i \leq n, b = y_i^*, \quad x_{i,b} = (1^\lambda, \mathsf{st}, 1^{\ell_1}, 1^{\ell_2}, 1^n, \beta, i, b, G), \quad \mathsf{ct}_{i,b} \leftarrow \mathsf{Enc}(1^\lambda, x_{i,b}, w_i),$$
$$\forall i \leq n, b \neq y_i^*, \quad x_{i,b} = (1^\lambda, \mathsf{st}, 1^{\ell_1}, 1^{\ell_2}, 1^n, \beta, i, b, G), \quad \mathsf{ct}_{i,b} \leftarrow \mathsf{Enc}(1^\lambda, x_{i,b}, \mathbf{0}),$$

where st is its local blockchain state. Finally, it sets the compiled circuit as $CC = \left(1^\lambda, 1^{\ell_1}, 1^{\ell_2}, G, \{\mathsf{ct}_{i,b}\}_{i\leq n, b\in\{0,1\}}\right)$, and sends CC to $\mathcal{A}$.

3. $\mathcal{A}$ receives the compiled circuit CC from the challenger, and outputs its complete transcript.

7.3.1 Analysis

We now establish via a sequence of claims that no PPT adversary with at most α stake ratio can distinguish between any two adjacent games with non-negligible advantage. Since the last game is identical to the simulated execution, therefore establishing non-negligible gap between adjacent games is sufficient.

Let $\mathcal{D}$ be any successful PPT adversary with α-bounded stake ratio that tries to distinguish between any two successive games. Let $\mathsf{Adv}^i_{\mathcal{D}}$ denote the advantage of algorithm $\mathcal{D}$ in distinguishing between Games i and $i+1$. We show via a sequence of claims that $\mathsf{Adv}^1_{\mathcal{D}}$ and $\mathsf{Adv}^2_{\mathcal{D}}$ both are negligible. Below we discuss our claims in detail.

Claim 7.1. If WE is extractable secure witness encryption scheme for language $\mathcal{L}_{\Gamma^V}$, and $(\alpha, \beta, \ell_1, \ell_2)$-distinguishable forking property holds for blockchain protocol Γ^V, then for any PPT adversary $\mathcal{D}$ with at most α stake ratio, $\mathsf{Adv}^1_{\mathcal{D}}$ is negligible in the security parameter λ.

Proof. For proving indistinguishability of Games 1 and 2, we need to sketch n intermediate hybrid games between these two, where n is the input length in circuit C. Observe that in Game 1, ciphertexts $\mathsf{ct}_{i,b}$ are encryptions of garbled circuit input wire keys $w_{i,b}$ for both values of bit b; however, in Game 2, ciphertexts $\mathsf{ct}_{i,b}$ are encryptions of $w_{i,b}$ *if and only if* $b = y_i^*$, and they are encryptions of zeros otherwise. The high-level proof idea is to switch $\mathsf{ct}_{i,b}$ from encryptions of $w_{i,b}$ to encryptions of $\mathbf{0}$ one-at-a-time by using extractable security of the witness encryption scheme. Intuitively, this could be done because the adversary $\mathcal{A}$ is only allowed to broadcast (G, y^*) before broadcasting any other message of the form (G, y), therefore if $\mathcal{A}$ does not try to fork the blockchain, then it could only generate the witnesses corresponding to instances x_{i,y_i^*}. Concretely, i^{th} intermediate hybrid between Game 1 and 2 proceeds same as Game 1 except that the first i ciphertexts $\mathsf{ct}_{j,b}$ are computed as $\mathsf{ct}_{j,b} \leftarrow \mathsf{Enc}(1^\lambda, x_{j,b}, \mathbf{0})$ if $b \neq y_j^*$, i.e. for $j \leq i$ and $b \neq y_j^*$, $\mathsf{ct}_{j,b}$ are encryptions of zero, and for $j > i$ or $b = y_j^*$, $\mathsf{ct}_{j,b}$ are encryptions of wire keys $w_{j,b}$ under instances $x_{j,b}$. For the analysis, Game 1 is regarded as 0^{th} intermediate hybrid, and Game 2 is regarded as n^{th} intermediate

hybrid. Below we show that $\mathcal{D}$'s advantage in distinguishing any pair of consecutive intermediate hybrid is negligibly small.

We describe a reduction algorithm $\mathcal{B}$ which breaks the bounded forking/mining assumption, if $\mathcal{D}$ distinguishes between intermediate hybrids $i-1$ and i with non-negligible advantage. Since WE is an extractable secure witness encryption scheme for language $\mathcal{L}_{\Gamma^V}$, therefore for every distinguisher $\mathcal{D}$ with at most α stake ratio, there exists a PPT algorithm E that extracts the witness given the challenge messages and the instance. The reduction algorithm $\mathcal{B}$ simply runs the extractor E on inputs security parameter λ, instance $x_{i,1-y_i^*}$, and messages $\mathbf{0}$ and $w_{i,1-y_i^*}$. Since E runs in polynomial time, therefore if $\mathsf{Adv}^1_{\mathcal{D}}$ is non-negligible, then either WE is not a secure witness encryption scheme for language $\mathcal{L}_{\Gamma^V}$, or $(\alpha, \beta, \ell_1, \ell_2)$-distinguishable forking property does not hold for blockchain protocol Γ^V. ■

Claim 7.2. If GC is a secure garbling scheme for circuit family $\{\mathcal{C}_n\}_n$, then for any PPT adversary $\mathcal{D}$, $\mathsf{Adv}^2_{\mathcal{D}}$ is negligible in the security parameter λ.

Proof. The proof of this claim follows directly from the security of our garbling scheme. For any circuit $C \in \mathcal{C}_n$, if $\mathcal{D}$ distinguishes between Game 2 and 3 with non-negligible probability, then it also breaks security of garbling scheme GC.

Suppose there exists an adversary $\mathcal{D}$ such that $\mathsf{Adv}^2_{\mathcal{D}}$ is non-negligible in λ. We construct an algorithm $\mathcal{B}$ that can distinguish between a simulated garbled circuit and an honestly generated garbled circuit, therefore break security of garbling scheme GC. Fix any circuit $C \in \mathcal{C}_n$ and input $y^* \in \{0,1\}^n$. $\mathcal{B}$ received a garbled circuit G along with wire keys w_i for $i \leq n$. It computes ciphertexts $\mathsf{ct}_{i,b}$, and sets the compiled circuit CC as in Game 3. It send the compiled circuit CC to $\mathcal{A}$, and gives $\mathcal{A}$'s output transcript to distinguisher $\mathcal{D}$. Finally, $\mathcal{B}$ outputs whatever $\mathcal{D}$ outputs.

Note that if the garbled circuit G was honestly computed, then $\mathcal{B}$ exactly simulates the view of Game 2 to $\mathcal{A}$. Otherwise the view is of Game 3. As a result, if $\mathsf{Adv}^2_{\mathcal{D}}$ is non-negligible in λ, then $\mathcal{B}$ breaks the garbling scheme's security with non-negligible advantage. ■

7.3.2 Discussion

Output Hiding. In the above one-time compiler construction any party executing the blockchain protocol can evaluate the circuit since the witness used for decrypting the wire keys becomes publicly known. If we only want the evaluator to learn the output, then the above scheme could be modified as follows — evaluator broadcasts an instance of a hard language (for which it knows the witness) in addition to the current message, and the compiler encrypts the wire keys for a modified witness relation such that now the witness for decrypting the wire keys must also include a witness for the hard to decide instance. For e.g., the instance could be a random element in the image set of a one-way function and its preimage would be a witness.

Input Hiding. We would also like to note that in current construction the evaluator needs to publicly broadcast its input. This might not be suitable for applications of one-time programs which want the evaluator's input to be hidden. To this end, the scheme could be modified as follows — evaluator broadcasts a binding commitment to its input instead of its actual input, and compiler encrypts the wire keys under same instances but for a modified witness relation such that now the witness also includes opening for the commitment and the witness relation verifies opening as well.

Efficiency. To reduce evaluation time, the compiler instead of using its local state (i.e., entire blockchain) as part of the instance could use only the last block in its blockchain and then the evaluator would only need to provide the blocks posted after that as its witness. Additionally, the size of the instance and broadcast message could be reduced by using a shorter unique identifier instead of the garbled circuit G. For instance, the identifier could be the hash of garbled circuit. Note that a unique identifier is tied with each input only to allow more than one program to be evaluated over the blockchain.

Witness Encryption. We would like to point out that in our one-time compiler construction, if we use only WE for circuits then the evaluator is constrained to run the program within an a-priori bounded as the witness length will be bounded. However, if the WE scheme supports decryption even for unbounded length witnesses, then the same scheme will result in general (unconstrained) one-time programs.

On the necessity of Extractable Witness Encryption. One might ask whether a strong assumption like extractable WE is necessary for constructing one-time programs, or could it be relaxed. It turns out that, to construct one-time programs, it is *sufficient and necessary* to assume a slightly weaker primitive which we call *one-time extractable WE*. A one-time extractable WE is same as a standard extractable WE scheme, except on each ciphertext the decryption algorithm could only be run once. In other words, if we decrypt a one-time WE ciphertext with a bad witness the first time, then next time decryption (on that same ciphertext) will always fail even if we use a correct witness. Again this cannot be solely software based as then ciphertext could always be copied, and thus one-time decryption wouldn't make sense. It is straightforward to verify in our OTP construction that we could instead use such a one-time extractable WE scheme. Additionally, anologous to construction of extractable WE from VBB obfuscation, we could show that a OTP already implies a one-time extractable WE, therefore our assumption of one-time extractable WE for constructing OTPs is both necessary and sufficient.

8 Pay-Per-Use Programs

In this section we introduce the notion of pay-per-use programs over blockchains. Informally, a pay-per-use program is a contract between two parties which we call the service provider and customer in the following exposition. A service provider wants to supply a program (or service) such that any customer who transfers a specific amount of coins to the provider (over the blockchain) can evaluate the program on any input of its choice once. Additionally, the service provider need not be executing the blockchain protocol after supplying the program, i.e. it could go *offline*.

8.1 Definition

Let $\{\mathcal{C}_n\}_n$ be a family of circuits where each circuit in $\mathcal{C}_n$ takes n bit inputs, and Γ be a blockchain protocol with validity predicate V. A pay-per-use compiler PPC for circuit family $\{\mathcal{C}_n\}_n$ over blockchain protocol Γ^V consists of polynomial-time algorithms $\mathsf{Compile}$ and Eval with the following syntax.

- $\mathsf{Compile}(1^\lambda, C \in \mathcal{C}_n, \mathsf{id} \in \{0,1\}^*, q \in \mathbb{Q})$: The compilation algorithm takes as input the security parameter λ, a circuit $C \in \mathcal{C}_n$, a public identity id and a rational number q. It outputs a compiled circuit CC.
- $\mathsf{Eval}(CC, x \in \{0,1\}^n, \widetilde{\mathsf{id}} \in \{0,1\}^*)$: The evaluation algorithm takes as input a compiled circuit CC, an n-bit input x and a public identity $\widetilde{\mathsf{id}}$. It outputs $y \in \{0,1\} \cup \perp$.

Before formally defining correctness and security properties for pay-per-use compilers, we would like to point out that a pay-per-use program whose functionality remains hidden even after polynomially many and apriori unbounded number of evaluations is very powerful. For instance, consider a pay-per-use program in which the provider sets run-time cost to be 0 i.e., the customer does not need to transfer any coin to evaluate the program. Such a primitive that provides security for an unbounded number of evaluations will be equivalent to an appropriate notion of obfuscation. In this work we will be interested in pay-per-use programs that can be evaluated only at one point as these could be easily generalized to run more than once. The correctness and security properties are defined below.

Correctness. A pay-per-use compiler PPC for circuit family $\{\mathcal{C}_n\}_n$ over blockchain protocol Γ^V is said to be correct if for all λ, n, $q \geq 0$, $\mathsf{id}, \widetilde{\mathsf{id}} \in \{0,1\}^*$, $x \in \{0,1\}^n$ and $C \in \mathcal{C}_n$,

$$\Pr[\mathsf{Eval}(CC, x, \widetilde{\mathsf{id}}) = C(x) \mid CC \leftarrow \mathsf{Compile}(1^\lambda, C, \mathsf{id}, q)] \geq 1 - \mathrm{negl}(\lambda),$$

where a transaction of q coins was made over Γ^V from identity $\widetilde{\mathsf{id}}$ to id, evaluation is run only once, and $\mathrm{negl}(\cdot)$ is a negligible function.

Definition 8.1. A pay-per-use compiler $\mathsf{PPC} = (\mathsf{Compile}, \mathsf{Eval})$ for a class of circuits $\mathcal{C} = \{\mathcal{C}_n\}_n$ over blockchain protocol Γ^V is said to be a *selectively*-secure pay-per-use compiler if for every *admissible* PPT adversary $\mathcal{A}$, there exists a PPT simulator Sim such that for all λ, n, $q \geq 0$, $C \in \mathcal{C}_n$ and $x \in \{0,1\}^n \cup \bot$, the following holds:

$$\left\{\mathsf{view}_{\mathsf{Sim}}\left(\mathsf{EXEC}^{\Gamma^V}\left(\mathsf{Sim}\left(1^n, 1^{|C|}, x, C_\bot(x), \mathsf{id}, q\right), \mathcal{Z}, 1^\lambda\right)\right)\right\}$$
$$\approx_c$$
$$\left\{\mathsf{view}_{\mathcal{A}}\left(\mathsf{EXEC}^{\Gamma^V}\left(\mathcal{A}\left(CC\right), \mathcal{Z}, 1^\lambda\right)\right) \;:\; CC \leftarrow \mathsf{Compile}(1^\lambda, C, \mathsf{id}, q)\right\}$$

where $C_\bot(x) = C(x)$ for $x \in \{0,1\}^n$ and $C_\bot(\bot) = \bot$, and adversary $\mathcal{A}$ is *admissible* if:

- $x = \bot$ and $\mathcal{A}$ never transfers q or more coins to identity id during blockchain execution, or
- $x \neq \bot$ and $\mathcal{A}$ transfers q coins to identity id and evaluates the program CC on x before evaluating on any other input.

As before we could also define full security for pay-per-use compilers, as a strengthening of selective security, by removing the restriction on the adversary to provide its input (if any) on which it will evaluate CC before it sees the compiled circuit.

Next, we provide our construction for pay-per-use compilers from garbled circuits and extractable witness encryption over any POS based blockchain protocol. The high level idea is similar to that for one-time compilers. As before, we start by describing the **NP** language for which we assume existence of a secure extractable witness encryption scheme.

8.2 NP Relation on Blockchain Protocols

Let $\Gamma = (\mathsf{UpdateState}, \mathsf{GetRecords}, \mathsf{Broadcast})$ be a blockchain protocol with validity predicate V. Consider the following relation.

Definition 8.2. Let $\widetilde{\mathcal{R}}_{\Gamma^V}$ be a relation on the blockchain protocol Γ^V. The instances and witnesses satisfying the relation are of the form

$$x = (1^\lambda, \mathsf{st}, 1^{\ell_1}, 1^{\ell_2}, 1^n, \beta, i, b, \mathsf{id}, q), \quad w = \widetilde{\mathsf{st}}.$$

Let $\mathbf{B} = \mathsf{GetRecords}(1^\lambda, \mathsf{st})$ and $\widetilde{\mathbf{B}} = \mathsf{GetRecords}(1^\lambda, \widetilde{\mathsf{st}})$. The instance-witness pair satisfies the relation $((x,w) \in \widetilde{\mathcal{R}}_{\Gamma^V})$ *if and only if* all the following properties are satisfied:

- Blockchains $\mathbf{B}$ and $\widetilde{\mathbf{B}}$ are valid, i.e. $V(\mathbf{B}) = V(\widetilde{\mathbf{B}}) = 1$
- $\mathbf{B}$ is a prefix of $\widetilde{\mathbf{B}}$, i.e. they are consistent[19]
- There exists a *unique* block $B^* \in \widetilde{\mathbf{B}} \setminus \mathbf{B}$ such that the following are satisfied

[19]Formally, the consistency should be checked as $\mathbf{B}^{\lceil\kappa} \preceq \widetilde{\mathbf{B}}$ for an appropriate value of parameter κ (Definition 4.1), however for ease of exposition we avoid it.

– There exists a *unique* currency transaction message m^* in B^* such that $m^* = (\widetilde{\mathsf{id}}, \mathsf{id}, q, \mathsf{aux} = y)$, y is an n-bit string, and either $i = 0$ or $y_i = b$

– Let ℓ' be the number of blocks in blockchain $\widetilde{\mathbf{B}}$ after block B^*, i.e. $B^* \in \widetilde{\mathbf{B}}^{\lceil \ell'}$. It should hold that $\ell' \geq \ell_1 + \ell_2$ and $\mathsf{u\text{-}stakefrac}(\widetilde{\mathbf{B}}, \ell' - \ell_1) > \beta$

Remark 8.1. The *uniqueness* of block B^* and record m^* is defined analogous to that in Remark 7.1.

Let $\widetilde{\mathcal{L}}_{\Gamma^V}$ be the language specified by the relation $\widetilde{\mathcal{R}}_{\Gamma^V}$. This language is in **NP** because verifying validity of blockchains take only polynomial time and the properties in Definition 8.2 could also be verified simultaneously.

8.3 Construction

Let $\Gamma^V = (\mathsf{UpdateState}^V, \mathsf{GetRecords}, \mathsf{Broadcast})$ be a blockchain protocol, and $\mathsf{GC} = (\mathsf{GC.Garble}, \mathsf{GC.Eval})$ be a garbling scheme for circuit family $\mathcal{C} = \{\mathcal{C}_n\}_n$, and $\mathsf{WE} = (\mathsf{Enc}, \mathsf{Dec})$ be a witness encryption scheme for language $\widetilde{\mathcal{L}}_{\Gamma^V}$. Below we describe our pay-per-use compilers $\mathsf{PPC} = (\mathsf{Compile}, \mathsf{Eval})$ for circuit family $\mathcal{C} = \{\mathcal{C}_n\}_n$ in the blockchain model.

- $\mathsf{Compile}(1^\lambda, 1^{\ell_1}, 1^{\ell_2}, \beta, C \in \mathcal{C}_n, \mathsf{id} \in \{0,1\}^*, q \in \mathbb{Q})$: The compilation algorithm first garbles the circuit C by computing $(G, \{w_{i,b}\}_{i \leq n, b \in \{0,1\}}) \leftarrow \mathsf{GC.Garble}(1^\lambda, C)$. Next, it encrypts each of the wire keys $w_{i,b}$ separately under instances $x_{i,b}$ as follows:

$$\forall i \leq n, b \in \{0,1\}, \quad x_{i,b} = (1^\lambda, \mathsf{st}, 1^{\ell_1}, 1^{\ell_2}, 1^n, \beta, i, b, \mathsf{id}, q), \quad \mathsf{ct}_{i,b} \leftarrow \mathsf{Enc}(1^\lambda, x_{i,b}, w_{i,b}),$$

  where st is its local blockchain state. It also encrypts the garbled circuit G as follows:

$$x_0 = (1^\lambda, \mathsf{st}, 1^{\ell_1}, 1^{\ell_2}, 1^n, \beta, 0, 0, \mathsf{id}, q), \quad \mathsf{ct}_0 \leftarrow \mathsf{Enc}(1^\lambda, x_0, G),$$

  Finally, it sets the compiled circuit as $CC = \left(1^\lambda, 1^{\ell_1}, 1^{\ell_2}, \mathsf{id}, q, \mathsf{ct}_0, \{\mathsf{ct}_{i,b}\}_{i \leq n, b \in \{0,1\}}\right)$.

- $\mathsf{Eval}(CC, y \in \{0,1\}^n, \widetilde{\mathsf{id}} \in \{0,1\}^*)$: Let $CC = \left(1^\lambda, 1^{\ell_1}, 1^{\ell_2}, \mathsf{id}, q, \mathsf{ct}_0, \{\mathsf{ct}_{i,b}\}_{i \leq n, b \in \{0,1\}}\right)$. The evaluation algorithm transfers q coins under its public identity $\widetilde{\mathsf{id}}$ to id by broadcasting a currency transfer message with its input y as the auxiliary information. Concretely, it runs $\mathsf{Broadcast}$ algorithm as $\mathsf{Broadcast}(1^\lambda, m)$ where $m = (\widetilde{\mathsf{id}}, \mathsf{id}, q, \mathsf{aux} = y)$.

  Next, it runs the $\mathsf{UpdateState}$ algorithm, and waits for message m to be posted on the blockchain and further the chain to be extended by $\ell_1 + \ell_2$ blocks. After the blockchain gets extended, it uses its own local state st as a witness to decrypt the garbled circuit and wire keys corresponding to input y as

$$G = \mathsf{Dec}(\mathsf{ct}_0, \mathsf{st}), \qquad \forall i \leq n, \quad w_i = \mathsf{Dec}(\mathsf{ct}_{i,y_i}, \mathsf{st}).$$

  It then uses these n wire keys to evaluate the garbled circuit, and outputs $\mathsf{GC.Eval}(G, \{w_i\}_{i \leq n})$. If the witnes decryption fails (outputs $\perp$), then it also outputs $\perp$.

Correctness. Fix any λ, n, q, ℓ_1, ℓ_2, β, id, and circuit $C \in \mathcal{C}_n$. Let $(G, \{w_{i,b}\}) \leftarrow \mathsf{GC.Garble}(1^\lambda, C)$, $x_0 = (1^\lambda, \mathsf{st}, 1^{\ell_1}, 1^{\ell_2}, 1^n, \beta, 0, 0, \mathsf{id}, q)$, $x_{i,b} = (1^\lambda, \mathsf{st}, 1^{\ell_1}, 1^{\ell_2}, 1^n, \beta, i, b, \mathsf{id}, q)$, and $\mathsf{ct}_0 \leftarrow \mathsf{Enc}(1^\lambda, x_0, G)$, $\mathsf{ct}_{i,b} \leftarrow \mathsf{Enc}(1^\lambda, x_{i,b}, w_{i,b})$.

For any input $y \in \{0,1\}^n$, consider that an evaluator runs $\mathsf{Broadcast}$ algorithm to post the transaction message $m = (\widetilde{\mathsf{id}}, \mathsf{id}, q, y)$ on the blockchain with public identity $\widetilde{\mathsf{id}}$. Let $\widetilde{\mathsf{id}}$ has a balance of more than q coins, and $\widetilde{\mathsf{st}}$ be the local state of the evaluator after message m is posted on blockchain and is extended by $\ell_1 + \ell_2$ blocks. Assuming that evaluator and compiler's blockchain are consistent (Definition 4.1), then with all but negligible probability for all $i \leq n$, $\widetilde{\mathsf{st}}$ could be used as the witness to decrypt ciphertexts ct_0 and ct_{i,y_i} as $(x_0, \widetilde{\mathsf{st}}) \in \widetilde{\mathcal{R}}_{\Gamma^V}$ and $(x_{i,y_i}, \widetilde{\mathsf{st}}) \in \widetilde{\mathcal{R}}_{\Gamma^V}$. This is true because consistency property guarantees that, with all but

negligible probability, the blockchains $\mathbf{B}$ and $\widetilde{\mathbf{B}}$ will be consistent. Additionally, the stake quantity property (Definition 4.4) guarantees that (with all but negligible probability) the condition $\mathsf{u\text{-}stakefrac}(\widetilde{\mathbf{B}}, \ell' - \ell_1) > \beta$ will be satisfied. Therefore, $\mathsf{Dec}(\mathsf{ct}_0, \mathsf{st}) = G$ and $\mathsf{Dec}(\mathsf{ct}_{i,y_i}, \mathsf{st}) = w_{i,y_i}$ which follows from correctness of the witness encryption scheme. Finally, $\mathsf{GC.Eval}(G, \{w_{i,y_i}\}_{i \le n}) = C(y)$ as it follows from correctness of the garbling scheme. Therefore, PPC satisfies the pay-per-use compiler correctness property.

Remark 8.2. As in the one-time compilers, our pay-per-use compiler also takes the hardness parameters ℓ_1, ℓ_2 and β as additional inputs. Refer to Remark 7.2 for more discussion.

8.4 Security Proof

We will now show that the pay-per-use compiler described in Section 8.3 achieves one-time secrecy as per Definition 8.1. More formally, we prove the following theorem where all the parameters are polynomials in the security parameter λ.

Theorem 8.1. If GC is a secure garbling scheme (Definition 3.6) for circuit family $\{\mathcal{C}_n\}_n$, WE is extractable secure witness encryption scheme (Definition 3.7) for language $\widetilde{\mathcal{L}}_{\Gamma^V}$, and $(\alpha, \beta, \ell_1, \ell_2)$-distinguishable forking property (Definition 4.7) holds for blockchain protocol Γ^V, then OTC with hardness parameters ℓ_1, ℓ_2 and β is a *selectively*-secure pay-per-use compiler (Definition 8.1) for circuit family $\{\mathcal{C}_n\}_n$ against all PPT adversaries with at most α stake ratio.

The proof of above theorem is divided in two parts.

8.4.1 Adversary transfers q coins

In this scenario, we show that the adversary does not learn anything more about the circuit other than its size and output on the specified point. This proof is identical to that of Theorem 7.1, therefore we only provide a brief sketch. First, we sketch the simulator Sim. Let $\mathcal{A}$ be any real world adversary.

- Sim obtains the input y^* from $\mathcal{A}$, and also obtains $(1^n, 1^{|C|}, y', C(y'), \mathsf{id}, q)$ during initialization. It aborts if $y' \neq y^*$.
- It computes the garbled circuit G as $(G, \{w_i\}_{i \le n}) \leftarrow \mathsf{GC.Sim}(1^\lambda, 1^n, 1^{|C|}, C(y'))$.
- Next, it computes ciphertexts $\mathsf{ct}_{i,b}$ as follows:

$$\forall i \le n, b = y^*_i, \quad x_{i,b} = (1^\lambda, \mathsf{st}, 1^{\ell_1}, 1^{\ell_2}, 1^n, \beta, i, b, \mathsf{id}, q), \quad \mathsf{ct}_{i,b} \leftarrow \mathsf{Enc}(1^\lambda, x_{i,b}, w_i),$$
$$\forall i \le n, b \neq y^*_i, \quad x_{i,b} = (1^\lambda, \mathsf{st}, 1^{\ell_1}, 1^{\ell_2}, 1^n, \beta, i, b, \mathsf{id}, q), \quad \mathsf{ct}_{i,b} \leftarrow \mathsf{Enc}(1^\lambda, x_{i,b}, \mathbf{0}),$$

  where st is its local blockchain state. It also encrypts the garbled circuit G as

$$x_0 = (1^\lambda, \mathsf{st}, 1^{\ell_1}, 1^{\ell_2}, 1^n, \beta, i, b, \mathsf{id}, q), \quad \mathsf{ct}_0 \leftarrow \mathsf{Enc}(1^\lambda, x_0, G).$$

- It sets the compiled circuit as $CC = \big(1^\lambda, 1^{\ell_1}, 1^{\ell_2}, \mathsf{id}, q, \mathsf{ct}_0, \{\mathsf{ct}_{i,b}\}_{i \le n, b \in \{0,1\}}\big)$, and sends CC to $\mathcal{A}$.
- Finally, it outputs whatever $\mathcal{A}$ outputs.

Next, we describe a sequence of games. In the following games, $\mathcal{A}$ can also participate in the blockchain execution with the only restriction that it is not allowed to broadcast any message of form $(\widetilde{\mathsf{id}}, \mathsf{id}, q, y)$ with $y \neq y^*$ *before* broadcasting $(\widetilde{\mathsf{id}}, \mathsf{id}, q, y^*)$.

Game 1: In this game, the real world adversary $\mathcal{A}$ interacts with the pay-per-use compiler challenger.

1. $\mathcal{A}$ sends its input y^* to the challenger.
2. The challenger garbles the circuit C by computing $(G, \{w_{i,b}\}_{i\leq n, b\in\{0,1\}}) \leftarrow \mathsf{GC.Garble}(1^\lambda, C)$. Next, it encrypts each of the wire keys $w_{i,b}$ as follows:

$$\forall i \leq n, b \in \{0,1\}, \quad x_{i,b} = (1^\lambda, \mathsf{st}, 1^{\ell_1}, 1^{\ell_2}, 1^n, \beta, i, b, \mathsf{id}, q), \quad \mathsf{ct}_{i,b} \leftarrow \mathsf{Enc}(1^\lambda, x_{i,b}, w_{i,b}),$$

   where st is its local blockchain state. It also encrypts the garbled circuit G as follows:

$$x_0 = (1^\lambda, \mathsf{st}, 1^{\ell_1}, 1^{\ell_2}, 1^n, \beta, 0, 0, \mathsf{id}, q), \quad \mathsf{ct}_0 \leftarrow \mathsf{Enc}(1^\lambda, x_0, G),$$

   Finally, it sets the compiled circuit as $CC = \left(1^\lambda, 1^{\ell_1}, 1^{\ell_2}, \mathsf{id}, q, \mathsf{ct}_0, \{\mathsf{ct}_{i,b}\}_{i\leq n, b\in\{0,1\}}\right)$, and sends CC to $\mathcal{A}$.
3. $\mathcal{A}$ receives the compiled circuit CC from the challenger, and outputs its complete transcript.

Game 2: This game is same as previous game, except the challenger encrypts only the wire keys corresponding to input y^* and creates rest of the n ciphertexts as encryptions of zeros. Concretely, the challenger generates the ciphertexts $\mathsf{ct}_{i,b}$ as follows

$$\begin{aligned}
&\forall i \leq n, b = y^*_i, \quad x_{i,b} = (1^\lambda, \mathsf{st}, 1^{\ell_1}, 1^{\ell_2}, 1^n, \beta, i, b, \mathsf{id}, q), \quad \mathsf{ct}_{i,b} \leftarrow \mathsf{Enc}(1^\lambda, x_{i,b}, w_{i,b}),\\
&\forall i \leq n, b \neq y^*_i, \quad x_{i,b} = (1^\lambda, \mathsf{st}, 1^{\ell_1}, 1^{\ell_2}, 1^n, \beta, i, b, \mathsf{id}, q), \quad \mathsf{ct}_{i,b} \leftarrow \mathsf{Enc}(1^\lambda, x_{i,b}, \mathbf{0}),
\end{aligned}$$

Game 3: This game is same as previous game, except the challenger simulates the circuit instead of garbling. Concretely, the challenger generates the garbled circuit and wire keys as $(G, \{w_i\}_{i\leq n}) \leftarrow \mathsf{GC.Sim}(1^\lambda, 1^n, 1^{|C|}, C(y^*))$.

The proof of indistinguishability between any two successive games is identical to that in Section 7.3.

8.4.2 Adversary does not transfer q coins

Here we show that the adversary does not learn anything about the circuit at all. First, we sketch the simulator Sim. Let $\mathcal{A}$ be any real world adversary.

- Sim obtains $(1^n, 1^{|C|}, \perp, \perp, \mathsf{id}, q)$ during initialization.
- Next, it computes ciphertexts ct_0 and $\mathsf{ct}_{i,b}$ as encryptions of all zeros:

$$\begin{aligned}
\forall i \leq n, b \in \{0,1\}, \quad x_{i,b} &= (1^\lambda, \mathsf{st}, 1^{\ell_1}, 1^{\ell_2}, 1^n, \beta, i, b, \mathsf{id}, q), \quad \mathsf{ct}_{i,b} \leftarrow \mathsf{Enc}(1^\lambda, x_{i,b}, \mathbf{0}),\\
x_0 &= (1^\lambda, \mathsf{st}, 1^{\ell_1}, 1^{\ell_2}, 1^n, \beta, 0, 0, \mathsf{id}, q), \quad \mathsf{ct}_0 \leftarrow \mathsf{Enc}(1^\lambda, x_0, \mathbf{0}),
\end{aligned}$$

  where st is its local blockchain state.
- It sets the compiled circuit as $CC = \left(1^\lambda, 1^{\ell_1}, 1^{\ell_2}, \mathsf{id}, q, \mathsf{ct}_0, \{\mathsf{ct}_{i,b}\}_{i\leq n, b\in\{0,1\}}\right)$, and sends CC to $\mathcal{A}$.
- Finally, it outputs whatever $\mathcal{A}$ outputs.

Next, we describe a sequence of games. In the following games, $\mathcal{A}$ can also participate in the blockchain execution with the only restriction that it is not allowed to broadcast any message of form $(\widetilde{\mathsf{id}}, \mathsf{id}, q, y)$.

Game 1: In this game, the real world adversary $\mathcal{A}$ interacts with the pay-per-use compiler challenger.

1. The challenger garbles the circuit C by computing $(G, \{w_{i,b}\}_{i\leq n, b\in\{0,1\}}) \leftarrow \mathsf{GC.Garble}(1^\lambda, C)$. Next, it encrypts each of the wire keys $w_{i,b}$ as follows:

$$\forall i \leq n, b \in \{0,1\}, \quad x_{i,b} = (1^\lambda, \mathsf{st}, 1^{\ell_1}, 1^{\ell_2}, 1^n, \beta, i, b, \mathsf{id}, q), \quad \mathsf{ct}_{i,b} \leftarrow \mathsf{Enc}(1^\lambda, x_{i,b}, w_{i,b}),$$

   where st is its local blockchain state. It also encrypts the garbled circuit G as follows:

$$x_0 = (1^\lambda, \mathsf{st}, 1^{\ell_1}, 1^{\ell_2}, 1^n, \beta, 0, 0, \mathsf{id}, q), \quad \mathsf{ct}_0 \leftarrow \mathsf{Enc}(1^\lambda, x_0, G),$$

   Finally, it sets the compiled circuit as $CC = \left(1^\lambda, 1^{\ell_1}, 1^{\ell_2}, \mathsf{id}, q, \mathsf{ct}_0, \{\mathsf{ct}_{i,b}\}_{i\leq n, b\in\{0,1\}}\right)$, and sends CC to $\mathcal{A}$.

2. $\mathcal{A}$ receives the compiled circuit CC from the challenger, and outputs its complete transcript.

Game 2: This game is same as previous game, except the challenger computes all ciphertexts as encryptions of zeros. Concretely, the challenger generates the ciphertexts ct_0 and $\mathsf{ct}_{i,b}$ as follows

$$\forall i \leq n, b \in \{0,1\}, \quad x_{i,b} = (1^\lambda, \mathsf{st}, 1^{\ell_1}, 1^{\ell_2}, 1^n, \beta, i, b, \mathsf{id}, q), \quad \mathsf{ct}_{i,b} \leftarrow \mathsf{Enc}(1^\lambda, x_{i,b}, \mathbf{0}),$$
$$x_0 = (1^\lambda, \mathsf{st}, 1^{\ell_1}, 1^{\ell_2}, 1^n, \beta, 0, 0, \mathsf{id}, q), \quad \mathsf{ct}_0 \leftarrow \mathsf{Enc}(1^\lambda, x_0, \mathbf{0}).$$

Note that the last game is identical to the simulated execution, therefore establishing non-negligible gap between Game 1 and 2 is sufficient. Let $\mathcal{D}$ be any successful PPT adversary with α-bounded stake ratio that tries to distinguish between Game 1 and 2, and $\mathsf{Adv}_{\mathcal{D}}$ denote its advantage. We claim the following.

Claim 8.1. If WE is extractable secure witness encryption scheme for language $\widetilde{\mathcal{L}}_{\Gamma^V}$, and $(\alpha, \beta, \ell_1, \ell_2)$-distinguishable forking property holds for blockchain protocol Γ^V, then for any PPT adversary $\mathcal{D}$ with at most α stake ratio, $\mathsf{Adv}_{\mathcal{D}}$ is negligible in the security parameter λ.

Proof. The proof of this claim is identical to the proof of Claim 7.1. For proving indistinguishability, we need to sketch $2n+1$ intermediate hybrid games between these two, where n is the input length in circuit C. Observe that in Game 1, ciphertexts ct_0 and $\mathsf{ct}_{i,b}$ are encryptions of garbled circuit G and its input wire keys $w_{i,b}$ (respectively)); however, in Game 2, all these ciphertexts are encryptions of zeros.

The idea is to use a standard hybrid argument over all these ciphertexts and switch these to encryptions of $\mathbf{0}$ one-at-a-time by using extractable security of the witness encryption scheme. Intuitively, this could be done because the adversary $\mathcal{A}$ is not allowed to broadcast any currency transaction message of the form $(\widetilde{\mathsf{id}}, \mathsf{id}, q, y)$ for any n-bit string y. Thus, the only way in which $\mathcal{A}$ can generate a valid witness corresponding to an instance $x_{i,b}$ or x_0 is by generating a fork in the blockchain of sufficiently large length ($\geq \ell$). However, this would violate the $(\alpha, \beta, \ell_1, \ell_2)$-distinguishable forking property. Since we assume that WE is a secure witness encryption scheme for language $\widetilde{\mathcal{L}}_{\Gamma^V}$ and the distinguishable forking property holds for blockchain protocol Γ^V, thus the claim follows. ∎

Acknowledgements. We thank Krzysztof Pietrzak and anonymous TCC reviewers for their useful feedback.

References

[ABP15] Michel Abdalla, Fabrice Benhamouda, and David Pointcheval. Disjunctions for hash proof systems: New constructions and applications. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 2015.

[ADMM14a] Marcin Andrychowicz, Stefan Dziembowski, Daniel Malinowski, and Lukasz Mazurek. Fair two-party computations via bitcoin deposits. In *Financial Cryptography and Data Security - FC 2014 Workshops, BITCOIN and WAHC 2014*, 2014.

[ADMM14b] Marcin Andrychowicz, Stefan Dziembowski, Daniel Malinowski, and Lukasz Mazurek. Secure multiparty computations on bitcoin. In *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*, 2014.

[AIKW15] Benny Applebaum, Yuval Ishai, Eyal Kushilevitz, and Brent Waters. Encoding functions with constant online rate, or how to compress garbled circuit keys. *SIAM Journal on Computing*, 2015.

[BBC+13] Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. New techniques for sphfs and efficient one-round pake protocols. In *Advances in Cryptology–CRYPTO 2013*, 2013.

[BCG15] Joseph Bonneau, Jeremy Clark, and Steven Goldfeder. On bitcoin as a public randomness source. Cryptology ePrint Archive, Report 2015/1015, 2015.

[BGI+12] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):6, 2012.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, 1988.

[BGZ16] Iddo Bentov, Ariel Gabizon, and David Zuckerman. Bitcoin beacon. *arXiv preprint arXiv:1605.04559*, 2016.

[BHR12a] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Adaptively secure garbling with applications to one-time programs and secure outsourcing. In *Advances in Cryptology - ASIACRYPT 2012*, 2012.

[BHR12b] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In *CCS '12*, 2012.

[BK14] Iddo Bentov and Ranjit Kumaresan. How to use bitcoin to design fair protocols. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II*, 2014.

[BOV07] Boaz Barak, Shien Jin Ong, and Salil Vadhan. Derandomization in cryptography. *SIAM J. Comput.*, 37(2):380–400, May 2007.

[BP15] Nir Bitansky and Omer Paneth. Zaps and non-interactive witness indistinguishability from indistinguishability obfuscation. In *Theory of Cryptography*, pages 401–427. Springer Berlin Heidelberg, 2015.

[BPS16a] Iddo Bentov, Rafael Pass, and Elaine Shi. The sleepy model of consensus. Cryptology ePrint Archive, Report 2016/918, 2016. `http://eprint.iacr.org/2016/918`.

[BPS16b] Iddo Bentov, Rafael Pass, and Elaine Shi. Snow white: Provably secure proofs of stake. Cryptology ePrint Archive, Report 2016/919, 2016. `http://eprint.iacr.org/2016/919`.

[Can01] Ran Canetti. Universally Composable Security: A new paradigm for cryptographic protocols. In *FOCS '01*, page 136. IEEE Computer Society, 2001. `http://eprint.iacr.org/2000/067`.

[CLT13] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, pages 476–493, 2013.

[CS02] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *Proceedings of Eurocrypt '02*, 2002.

[FLS99] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs under general assumptions. *SIAM Journal of Computing*, 29(1):1–28, 1999.

[FS90] Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *STOC*, pages 416–426, 1990.

[GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In *EUROCRYPT*, 2013.

[GGH+13b] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, 2013.

[GGHW14] Sanjam Garg, Craig Gentry, Shai Halevi, and Daniel Wichs. On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. In *Advances in Cryptology - CRYPTO 2014*, 2014.

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In *STOC*, 2013.

[GIS+10] Vipul Goyal, Yuval Ishai, Amit Sahai, Ramarathnam Venkatesan, and Akshay Wadia. Founding cryptography on tamper-proof hardware tokens. In *TCC*, pages 308–326, 2010.

[GK08] Vipul Goyal and Jonathan Katz. Universally composable multi-party computation with an unreliable common reference string. In *Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008.*, 2008.

[GKL15] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In *Advances in Cryptology - EUROCRYPT 2015*, 2015.

[GKL16] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol with chains of variable difficulty. Cryptology ePrint Archive, Report 2016/1048, 2016.

[GKLP16] Juan A. Garay, Aggelos Kiayias, Nikos Leonardos, and Giorgos Panagiotakos. Bootstrapping the blockchain — directly. Cryptology ePrint Archive, Report 2016/991, 2016. `http://eprint.iacr.org/2016/991`.

[GKP+13] Shafi Goldwasser, Yael Kalai, Raluca Ada Popa, Vinod Vaikuntanathan, , and Nickolai Zeldovich. How to run turing machines on encrypted data. In *CRYPTO*, 2013.

[GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. One-time programs. In *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, 2008.

[GMR89] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, February 1989.

[GO94] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *J. Cryptology*, 1994.

[GO07] Jens Groth and Rafail Ostrovsky. Cryptography in the multi-string model. In *Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings*, 2007.

[GOS12] Jens Groth, Rafail Ostrovsky, and Amit Sahai. New techniques for noninteractive zero-knowledge. *J. ACM*, 59(3):11, 2012.

[HK07] Omer Horvitz and Jonathan Katz. Universally-composable two-party computation in two rounds. In *Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings*, 2007.

[Jag15] Tibor Jager. How to build time-lock encryption. Cryptology ePrint Archive, Report 2015/478, 2015. `http://eprint.iacr.org/2015/478`.

[JR13] Charanjit S Jutla and Arnab Roy. Shorter quasi-adaptive nizk proofs for linear subspaces. In *International Conference on the Theory and Application of Cryptology and Information Security*, 2013.

[KB14] Ranjit Kumaresan and Iddo Bentov. How to use bitcoin to incentivize correct computations. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*, 2014.

[KO04] Jonathan Katz and Rafail Ostrovsky. Round-optimal secure two-party computation. In *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, 2004.

[KP15] Aggelos Kiayias and Giorgos Panagiotakos. Speed-security tradeoffs in blockchain protocols. Cryptology ePrint Archive, Report 2015/1019, 2015. `http://eprint.iacr.org/2015/1019`.

[KRDO16] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. Cryptology ePrint Archive, Report 2016/889, 2016. `http://eprint.iacr.org/2016/889`.

[LKW15] Jia Liu, Saqib A. Kakvi, and Bogdan Warinschi. Extractable witness encryption and timed-release encryption from bitcoin. Cryptology ePrint Archive, Report 2015/482, 2015. `http://eprint.iacr.org/2015/482`.

[mtg10] mtgox, 2010. `https://bitcointalk.org/index.php?topic=2227.msg29606#msg29606`.

[Nak08] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

[NW94] Noam Nisan and Avi Wigderson. Hardness vs randomness. *J. Comput. Syst. Sci.*, 1994.

[PS16a] Rafael Pass and Elaine Shi. Fruitchains: A fair blockchain. Cryptology ePrint Archive, Report 2016/916, 2016. `http://eprint.iacr.org/2016/916`.

[PS16b] Rafael Pass and Elaine Shi. Hybrid consensus: Efficient consensus in the permissionless model. Cryptology ePrint Archive, Report 2016/917, 2016. `http://eprint.iacr.org/2016/917`.

[PSS16] Rafael Pass, Lior Seeman, and Abhi Shelat. Analysis of the blockchain protocol in asynchronous networks. *IACR Cryptology ePrint Archive*, 2016.

[Sho01] Victor Shoup. A proposal for an iso standard for public key encryption (version 2.1), 2001.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 475–484, 2014.

[Wee10] Hoeteck Wee. Efficient chosen-ciphertext security via extractable hash proofs. In *Annual Cryptology Conference*, 2010.

[Woo14] Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger, 2014.

[Yao86] Andrew Yao. How to generate and exchange secrets. In *FOCS*, pages 162–167, 1986.