Overcoming Cryptographic Impossibility Results Using Blockchains

Rishab Goyal[1(✉)] and Vipul Goyal[2]

[1] University of Texas at Austin, Austin, USA
goyal@utexas.edu
[2] Carnegie Mellon University, Pittsburgh, USA
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 [5,23,46] 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 [33,36].
3. *One-time programs and pay-per use programs:* Goldwasser et al. [29] introduced the notion of one time program and presented a construction using tamper-proof hardware. As noted by Goldwasser et al. [29], clearly a one-time program cannot be solely software

Y. Kalai and L. Reyzin (Eds.): TCC 2017, Part I, LNCS 10677, pp. 529–561, 2017.
https://doi.org/10.1007/978-3-319-70500-2_18

based, as software can always be copied and run again. While there have been a number of follow up works [4,6,30], 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.

1 Introduction

The last few years have seen a dramatic rise of cryptocurrencies such as Bitcoin [42] and Ethereum [49]. 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 [5,23,46] 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 [2,3,11,34,39,40] 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 [13,38]. Previously, various works [12,13,19–21,37,38,43–45] 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 [10,14]. 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 [32] propose NIZKs with n authorities where a majority of them must be honest. Goyal and Katz [31] 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 [33,36].

– *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. [29] 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. [29], 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. [29], 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 [4,6,30], there are indeed *no known constructions of one-time programs which do not rely on self destructing*

[1] We would like to point out that (unlike other works like [2,3,11,39]) *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.

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) [25,28] (which in turn requires strong knowledge assumptions related over multi-linear maps [15,22], see also [24]). 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 [34,40], we, for example, already know efficient WE schemes for hash proof system compatible relations [1,9,16,35,48] 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.

[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.

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 [19,37,43] 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 [41,42]. The most well-known properties analyzed are chain consistency and chain 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.

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

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

- (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 [13,38] 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. More details about our proposed properties as well as their reductions to other desideratum is provided later in Sects. 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

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

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.

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 [8], 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 [17] 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

[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 [47].

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 $\beta(<1)$ 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.

The 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 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 Sect. 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.

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 Sect. 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 the full version.

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 is discussed in detail in the full version.

Comparison with related work. Recently it was shown that in [34,40] 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 [34,40] 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 [34,40] only gave an ad hoc analysis arguing that Bitcoin could be used implement such reference clocks.

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. [43], which in turn is an extension of [19].

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]

[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.

– $\mathsf{GetRecords}(1^\lambda, \mathsf{st})$: 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 [19,43], 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] Here we assume that the reader is familiar with the standard blockchain execution model. In the full version we will give a comprehensive overview.

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.

Due to space constraints, we do not provide formal definitions of witness encryption [24,25], garbled circuits [7,50], (non-interactive) witness indistinguishable (WI) proofs [18], and weighted threshold secret sharing.

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 $(\mathsf{mpk}, \mathsf{msk})$.

[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.

- $\mathsf{Enc}(\mathsf{mpk}, m \in \mathcal{M}_1)$: 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 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* $negl_1(\cdot), negl_2(\cdot)$*, such that for all* $\lambda \in \mathbb{N}$, *the following holds:*

$$\left| \Pr\left[\mathcal{A}_1(\mathsf{ct}, \mathsf{st}) = b \,\middle|\, \begin{array}{c} (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda); b \leftarrow \{0,1\} \\ (m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_0^{\mathsf{Sign}(\mathsf{msk}, \cdot)}(\mathsf{mpk}); \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, m_b) \end{array} \right] - \frac{1}{2} \right| \leq negl_1(\lambda),$$

and

$$\Pr\left[\mathsf{Verify}(\mathsf{msk}, m^*, \sigma^*) = 1 \,\middle|\, \begin{array}{c} (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ (m^*, \sigma^*) \leftarrow \mathcal{A}_2^{\mathsf{Sign}(\mathsf{msk}, \cdot)}(\mathsf{mpk}) \end{array} \right] \leq negl_2(\lambda),$$

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 [47]. However this will be a slightly stronger assumption.

3.2 Non-interactive Argument Systems

Non-interactive Zero Knowledge Arguments. The notion of Zero Knowledge for interactive protocols was introduced by Goldwasser, Micali and Rackoff [27]. 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 [26]. They are usually defined with trusted setup.

In this work, we construct NIZKs over blockchain protocols without any additional setup assumption. Below we provide the formal definition.

Definition 2 *(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 $negl_1(\cdot), negl_2(\cdot)$ such that*

$$\Pr\left[\mathcal{V}(\widetilde{\mathbf{B}}, x, \pi) = 1 : \begin{array}{c} \mathsf{view} \leftarrow \mathsf{EXEC}^{\Gamma^V}\left(\mathcal{A}, \mathcal{Z}, 1^\lambda\right) \\ \mathbf{B} = \mathsf{GetRecords}(\mathsf{view}_i);\ \widetilde{\mathbf{B}} = \mathsf{GetRecords}(\mathsf{view}_j) \\ \pi \leftarrow \mathcal{P}(\mathbf{B}, x, w) \end{array}\right] \geq 1 - negl_1(|x|) - negl_2(\lambda),$$

  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 $negl_1(\cdot), negl_2(\cdot)$ such that*

$$\Pr\left[\mathcal{V}(\mathbf{B}, x, \pi) = 1 : \begin{array}{c} \mathsf{view} \leftarrow \mathsf{EXEC}^{\Gamma^V}\left(\mathcal{A}(x), \mathcal{Z}, 1^\lambda\right) \\ \mathbf{B} = \mathsf{GetRecords}(\mathsf{view}_i);\ \pi \leftarrow \mathcal{A} \end{array}\right] \leq negl_1(|x|) + negl_2(\lambda),$$

  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 $negl_1(\cdot), negl_2(\cdot)$ 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

$$\Pr\left[\begin{array}{l} \mathcal{V}(\mathbf{B}, x, \pi) = 0 \\ \vee\ \mathcal{R}(x, w) = 1 \end{array} : \begin{array}{c} \mathsf{view} \leftarrow \mathsf{EXEC}^{\Gamma^V}\left(\mathcal{A}, \mathcal{E}, \mathcal{Z}, 1^\lambda\right);\ (x, \pi) \leftarrow \mathcal{A} \\ \mathbf{B} = \mathsf{GetRecords}(\mathsf{view}_i);\ w \leftarrow \mathcal{E}(x, \pi) \end{array}\right] \geq 1 - negl_1(|x|) - negl_2(\lambda),$$

  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 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.*

3.3 One-Time Programs and Compilers

The notion of one-time programs was introduced by Goldwasser et al. [29]. 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 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.

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. *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 [12,13,19–21,37,38,43–45] 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 [13,38] 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 $\mathsf{stake} : \{0,1\}^* \times \{0,1\}^* \rightarrow \mathbb{Q}^+$ 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 [43] which is an extension of the common prefix property by Garay et al. [19]. 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 *(Chain Consistency). A blockchain protocol Γ satisfies $\ell_0(\cdot)$-consistency with adversary $\mathcal{A}$ in environment $\mathcal{Z}$, if there exists negligible function $negl(\cdot)$ such that for every $\lambda \in \mathbb{N}$, $\ell > \ell_0(\lambda)$ the following holds:*

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

[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.

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

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

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 6 *(Proof-of-*Honest*-Stake Fraction).* *The proof-of-honest-stake fraction $\textit{u-honest-stakefrac}(\mathbf{B}, \ell)$ w.r.t. blockchain* **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*

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

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* $\textit{suf-stake-contr}$ *be the predicate such that* $\textit{suf-stake-contr}^{\ell}(\mathsf{view}, \beta) = 1$ *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.* $\textit{u-stakefrac}(\mathbf{B}, \ell) > \beta$.

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

Definition 7 *(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* $\mathit{negl}(\cdot)$ *such that for every* $\lambda \in \mathbb{N}$*,* $\ell \geq \ell_0(\lambda)$ *the following holds:*

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

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* $\textit{honest-suf-stake-contr}$ *be the predicate such that* $\textit{honest-suf-stake-contr}^{\ell}(\mathsf{view}, \beta) = 1$ *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.* $\textit{u-honest-stakefrac}(\mathbf{B}, \ell) > \beta$.

Definition 8 *(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 negl$(\cdot)$ such that for every $\lambda \in \mathbb{N}$, $\ell \geq \ell_0(\lambda)$ the following holds:*

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

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 $\textit{u-stakefrac}(\widetilde{\mathbf{B}}, \ell' - \ell_1) \leq \alpha$.*

Definition 9 *(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 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:*

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

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 10 *(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 $negl(\cdot), \delta(\cdot)$ such that for every $\lambda \in \mathbb{N}$, $\ell \geq \ell_1(\lambda)$, $\widetilde{\ell} \geq \ell_2(\lambda)$ the following holds:*

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

5 Instantiating Our Framework

In this section, we show that the proposed proof-of-stake based blockchain protocols of [13,38] satisfy all the properties described in Sect. 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 [41], and formally defined by [19]. 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* quality *be the predicate such that* $\textsf{quality}_{\mathcal{A}}^{\ell}(\textsf{view}, \mu) = 1$ *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 $\mathbf{B}^{\ell\rceil}$ 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 11 *(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 $negl(\cdot)$ such that for every $\lambda \in \mathbb{N}$, $\ell \geq \ell_0(\lambda)$ the following holds:*

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

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* $\widetilde{\ell} < \ell'$*,* $\widetilde{\mathbf{B}}^{\lceil \widetilde{\ell}} \npreceq \mathbf{B}$*, and* the last block in chain vvB is honest *(i.e., not mined by the adversary).*

Definition 12 *(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 negl*$(\cdot)$ *such that for every* $\lambda \in \mathbb{N}$*,* $\ell \geq \ell_0(\lambda)$ *the following holds:*

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

In the full version, we prove the following theorem.

Theorem 1. *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.[13] 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 **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.

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

[13] 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 [47]. Thus, our NIZKs are instantiable over existing blockchain protocols.

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 valid-fork predicate which will be used later while defining the trapdoor part of the statement.

Predicate 7. *Let* valid-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 **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 **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.[14]
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
$$\exists \{\mathsf{sh}_i, r_i\}_{i \in \mathcal{M}}, s \text{ such that } \begin{pmatrix} \{\mathsf{sh}_i\}_i = \mathsf{Share}(w, \{\mathsf{stake}_{\mathsf{id}}\}_{\mathsf{id}}, \beta \cdot \mathsf{stake}_{\mathsf{total}}; s) \wedge \\ \forall\ i,\ \mathsf{ct}_{i,1} = \mathsf{Enc}(\mathsf{pk}_i, \mathsf{sh}_i; r_i) \ \wedge\ \mathcal{R}(x, w) = 1 \end{pmatrix}$$
$$\bigvee \begin{pmatrix} \{\mathsf{sh}_i\}_i = \mathsf{Share}(\widetilde{\mathbf{B}}, \{\mathsf{stake}_{\mathsf{id}}\}_{\mathsf{id}}, \beta \cdot \mathsf{stake}_{\mathsf{total}}; s) \wedge \\ \forall\ i,\ \mathsf{ct}_{i,2} = \mathsf{Enc}(\mathsf{pk}_i, \mathsf{sh}_i; r_i) \ \wedge\ \mathsf{valid\text{-}fork}^V(\mathbf{B}', \widetilde{\mathbf{B}}, \ell_3, \ell_4, 1 - \alpha) \end{pmatrix}$$

[14] 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.

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

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

– $\mathcal{V}(\mathbf{B}, x, \pi)$: 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 Sect. 6.1 is NIZK argument of knowledge as per Definition 2. More formally, we prove the following theorem where all the parameters are polynomials in the security parameter λ.

Theorem 2. *If* $(\mathcal{P}_{\mathsf{NIWI}}, \mathcal{V}_{\mathsf{NIWI}})$ *is a NIWI argument system for* **NP***,* SS *is a weighted threshold secret sharing scheme,* HS *is a secure integrated public key encryption-signature scheme (Definition 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.*

We provide the proofs of completeness, soundness, zero-knowledge and argument of knowledge in the full version.

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 V. Consider the following relation.

Definition 13. *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*[15]
- *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* $\textsf{u-stakefrac}(\widetilde{\mathbf{B}}, \ell' - \ell_1) > \beta$

Remark 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 13 could also be verified simultaneously.

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 $(G, \{w_{i,b}\}_{i \leq n, b \in \{0,1\}}) \leftarrow \mathsf{GC.Garble}(1^\lambda, C)$. Next, it encrypts each of the wire keys $w_{i,b}$ separately under instances $x_{i,b}$ as follows:

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

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

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

  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 $\perp$.

[15] Formally, the consistency should be checked as $\mathbf{B}^{\lceil \kappa} \preceq \widetilde{\mathbf{B}}$ for an appropriate value of parameter κ (Definition 4), however for ease of exposition we avoid it.

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), then with all but negligible probability for all $i \leq n$, $\widetilde{\mathsf{st}}$ 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 7) 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, $\mathsf{GC.Eval}(G, \{w_{i,y_i}\}_{i \leq n}) = C(y)$ as it follows from correctness of the garbling scheme. Therefore, OTC satisfies the one-time compiler correctness condition.

Remark 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.

The security proof will be provided in the full version.

Acknowledgements. We thank Krzysztof Pietrzak and anonymous TCC reviewers for their useful feedback. The second author is supported in part by NSF CNS-1228599 and CNS-1414082, DARPA SafeWare.

References

1. Abdalla, M., Benhamouda, F., Pointcheval, D.: Disjunctions for hash proof systems: new constructions and applications. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 69–100. Springer, Heidelberg (2015). doi:10.1007/978-3-662-46803-6_3
2. Andrychowicz, M., Dziembowski, S., Malinowski, D., Mazurek, Ł.: Fair two-party computations via bitcoin deposits. In: Böhme, R., Brenner, M., Moore, T., Smith, M. (eds.) FC 2014. LNCS, vol. 8438, pp. 105–121. Springer, Heidelberg (2014). doi:10.1007/978-3-662-44774-1_8
3. Andrychowicz, M., Dziembowski, S., Malinowski, D., Mazurek, L.: Secure multiparty computations on bitcoin. In: 2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, 18–21 May 2014

4. Applebaum, B., Ishai, Y., Kushilevitz, E., Waters, B.: Encoding functions with constant online rate, or how to compress garbled circuit keys. SIAM J. Comput. **44**(2), 433–466 (2015)
5. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S.P., Yang, K.: On the (im)possibility of obfuscating programs. J. ACM **59**(2), 6 (2012)
6. Bellare, M., Hoang, V.T., Rogaway, P.: Adaptively secure garbling with applications to one-time programs and secure outsourcing. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 134–153. Springer, Heidelberg (2012). doi:10.1007/978-3-642-34961-4_10
7. Bellare, M., Hoang, V.T., Rogaway, P.: Foundations of garbled circuits. In: CCS 2012 (2012)
8. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: Proceedings of the 20th Annual ACM Symposium on Theory of Computing, Chicago, Illinois, USA, 2–4 May 1988
9. Benhamouda, F., Blazy, O., Chevalier, C., Pointcheval, D., Vergnaud, D.: New techniques for SPHFs and efficient one-round PAKE protocols. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 449–475. Springer, Heidelberg (2013). doi:10.1007/978-3-642-40041-4_25
10. Bentov, I., Gabizon, A., Zuckerman, D.: Bitcoin beacon. arXiv preprint arxiv:1605.04559 (2016)
11. Bentov, I., Kumaresan, R.: How to use bitcoin to design fair protocols. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8617, pp. 421–439. Springer, Heidelberg (2014). doi:10.1007/978-3-662-44381-1_24
12. Bentov, I., Pass, R., Shi, E.: The sleepy model of consensus. Cryptology ePrint Archive, Report 2016/918 (2016). http://eprint.iacr.org/2016/918
13. Bentov, I., Pass, R., Shi, E.: Snow white: provably secure proofs of stake. Cryptology ePrint Archive, Report 2016/919 (2016). http://eprint.iacr.org/2016/919
14. Bonneau, J., Clark, J., Goldfeder, S.: On bitcoin as a public randomness source. Cryptology ePrint Archive, Report 2015/1015 (2015)
15. Coron, J.-S., Lepoint, T., Tibouchi, M.: Practical multilinear maps over the integers. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 476–493. Springer, Heidelberg (2013). doi:10.1007/978-3-642-40041-4_26
16. Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 45–64. Springer, Heidelberg (2002). doi:10.1007/3-540-46035-7_4
17. Feige, U., Lapidot, D., Shamir, A.: Multiple non-interactive zero knowledge proofs under general assumptions. SIAM J. Comput. **29**(1), 1–28 (1999)
18. Feige, U., Shamir, A.: Witness indistinguishable and witness hiding protocols. In: STOC, pp. 416–426 (1990)
19. Garay, J., Kiayias, A., Leonardos, N.: The bitcoin backbone protocol: analysis and applications. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 281–310. Springer, Heidelberg (2015). doi:10.1007/978-3-662-46803-6_10
20. Garay, J.A., Kiayias, A., Leonardos, N.: The bitcoin backbone protocol with chains of variable difficulty. Cryptology ePrint Archive, Report 2016/1048 (2016)
21. Garay, J.A., Kiayias, A., Leonardos, N., Panagiotakos, G.: Bootstrapping the blockchain – directly. Cryptology ePrint Archive, Report 2016/991 (2016). http://eprint.iacr.org/2016/991

22. Garg, S., Gentry, C., Halevi, S.: Candidate multilinear maps from ideal lattices. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 1–17. Springer, Heidelberg (2013). doi:10.1007/978-3-642-38348-9_1
23. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: FOCS (2013)
24. Garg, S., Gentry, C., Halevi, S., Wichs, D.: On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8616, pp. 518–535. Springer, Heidelberg (2014). doi:10.1007/978-3-662-44371-2_29
25. Garg, S., Gentry, C., Sahai, A., Waters, B.: Witness encryption and its applications. In: STOC (2013)
26. Goldreich, O., Oren, Y.: Definitions and properties of zero-knowledge proof systems. J. Cryptology (1994)
27. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM J. Comput. **18**(1), 186–208 (1989). http://dx.doi.org/10.1137/0218012
28. Goldwasser, S., Kalai, Y.T., Popa, R.A., Vaikuntanathan, V., Zeldovich, N.: How to run turing machines on encrypted data. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8043, pp. 536–553. Springer, Heidelberg (2013). doi:10.1007/978-3-642-40084-1_30
29. Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: One-time programs. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 39–56. Springer, Heidelberg (2008). doi:10.1007/978-3-540-85174-5_3
30. Goyal, V., Ishai, Y., Sahai, A., Venkatesan, R., Wadia, A.: Founding cryptography on tamper-proof hardware tokens. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 308–326. Springer, Heidelberg (2010). doi:10.1007/978-3-642-11799-2_19
31. Goyal, V., Katz, J.: Universally composable multi-party computation with an unreliable common reference string. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 142–154. Springer, Heidelberg (2008). doi:10.1007/978-3-540-78524-8_9
32. Groth, J., Ostrovsky, R.: Cryptography in the multi-string model. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 323–341. Springer, Heidelberg (2007). doi:10.1007/978-3-540-74143-5_18
33. Horvitz, O., Katz, J.: Universally-composable two-party computation in two rounds. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 111–129. Springer, Heidelberg (2007). doi:10.1007/978-3-540-74143-5_7
34. Jager, T.: How to build time-lock encryption. Cryptology ePrint Archive, Report 2015/478 (2015). http://eprint.iacr.org/2015/478
35. Jutla, C.S., Roy, A.: Shorter quasi-adaptive NIZK proofs for linear subspaces. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013. LNCS, vol. 8269, pp. 1–20. Springer, Heidelberg (2013). doi:10.1007/978-3-642-42033-7_1
36. Katz, J., Ostrovsky, R.: Round-optimal secure two-party computation. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 335–354. Springer, Heidelberg (2004). doi:10.1007/978-3-540-28628-8_21
37. Kiayias, A., Panagiotakos, G.: Speed-security tradeoffs in blockchain protocols. Cryptology ePrint Archive, Report 2015/1019 (2015). http://eprint.iacr.org/2015/1019
38. Kiayias, A., Russell, A., David, B., Oliynykov, R.: Ouroboros: a provably secure proof-of-stake blockchain protocol. Cryptology ePrint Archive, Report 2016/889 (2016). http://eprint.iacr.org/2016/889

39. Kumaresan, R., Bentov, I.: How to use bitcoin to incentivize correct computations. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, 3–7 November 2014
40. Liu, J., Kakvi, S.A., Warinschi, B.: Extractable witness encryption and timed-release encryption from bitcoin. Cryptology ePrint Archive, Report 2015/482 (2015). http://eprint.iacr.org/2015/482
41. mtgox (2010). https://bitcointalk.org/index.php?topic=2227.msg29606#msg29606
42. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2008)
43. Pass, R., Seeman, L., Shelat, A.: Analysis of the blockchain protocol in asynchronous networks. IACR Cryptology ePrint Archive (2016). http://eprint.iacr.org/2016/454
44. Pass, R., Shi, E.: Fruitchains: a fair blockchain. Cryptology ePrint Archive, Report 2016/916 (2016). http://eprint.iacr.org/2016/916
45. Pass, R., Shi, E.: Hybrid consensus: efficient consensus in the permissionless model. Cryptology ePrint Archive, Report 2016/917 (2016). http://eprint.iacr.org/2016/917
46. Sahai, A., Waters, B.: How to use indistinguishability obfuscation: deniable encryption, and more. In: Symposium on Theory of Computing, STOC 2014, New York, NY, USA, 31 May–03 June 2014, pp. 475–484 (2014)
47. Shoup, V.: A proposal for an ISO standard for public key encryption (version 2.1) (2001)
48. Wee, H.: Efficient chosen-ciphertext security via extractable hash proofs. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 314–332. Springer, Heidelberg (2010). doi:10.1007/978-3-642-14623-7_17
49. Wood, G.: Ethereum: a secure decentralised generalised transaction ledger (2014)
50. Yao, A.: How to generate and exchange secrets. In: FOCS, pp. 162–167 (1986)