Time-Traveling Simulators Using Blockchains and Their Applications

Vipul Goyal
Carnegie Mellon University, Pittsburgh, PA, USA
NTT Research, Sunnyvale, CA, USA

Justin Raizes
Carnegie Mellon University, Pittsburgh, PA, USA

Pratik Soni
Carnegie Mellon University, Pittsburgh, PA, USA

Abstract

Blockchain technology has the potential of transforming cryptography. We study the problem of round-complexity of zero-knowledge, and more broadly, of secure computation in the *blockchain-hybrid model*, where all parties can access the blockchain as an oracle.

We study zero-knowledge and secure computation through the lens of a new security notion where the simulator is given the ability to "time-travel" or more accurately, to look into the future states of the blockchain and use this information to perform simulation. Such a *time-traveling simulator* gives a novel security guarantee of the following form: *whatever the adversary could have learnt from an interaction, it could have computed on its own shortly into the future (e.g., a few hours from now).*

We exhibit the power of time-traveling simulators by constructing round-efficient protocols in the blockchain-hybrid model. In particular, we construct:

1. Three-round zero-knowledge (ZK) argument for NP with a polynomial-time black-box time-traveling simulator.
2. Three-round secure two-party computation (2PC) for any functionality with a polynomial-time black-box time-traveling simulator for both parties.

In addition to standard cryptographic assumptions, we rely on natural hardness assumptions for Proof-of-Work based blockchains. In comparison, in the plain model, three-round protocols with black-box simulation are impossible, and constructions with non-black-box simulation for ZK require novel cryptographic assumptions while no construction for three-round 2PC is known. Our three-round 2PC result relies on a new, two-round extractable commitment that admits a time-traveling extractor.

2012 ACM Subject Classification Security and privacy → Mathematical foundations of cryptography

Keywords and phrases Cryptography, Zero Knowledge, Secure Two-Party Computation, Blockchain

Digital Object Identifier 10.4230/LIPIcs.ITCS.2022.81

Related Version *Full Version*: https://eprint.iacr.org/2022/035.pdf

Funding Research supported by the NSF award 1916939, DARPA SIEVE program, a gift from Ripple, a DoE NETL award, a JP Morgan Faculty Fellowship, a PNC center for financial services innovation award, and a Cylab seed funding award.

1 Introduction

We are seeing a surging interest in blockchain as a vibrant technology: it powers crypto-currencies like Bitcoin [30], Ethereum [37] and ZCash [10] which are valued in hundreds of billions of dollars, and it enables removing the central point of trust in existing banking infrastructure.

In a blockchain protocol, the goal of parties is to maintain a globally ordered sequence of records that are referred to as *blocks*. New blocks can only be added via a special mining procedure that simulates a puzzle solving race between participants and can be run by any participant or *miner*. For example, in Bitcoin, the puzzle is selected so that a block is mined (on average) every 10 minutes. The two widely used puzzles are Proof-of-Work (PoW) and Proof-of-Stake (PoS). A sequence of works [18, 27, 16, 33, 31, 34, 35, 19, 20] have analysed the security of current PoW- and PoS-based blockchains, and also proposed new ones. Understanding the security properties offered by these blockchains, and applications that such properties enable is a rapidly progressing direction.

Since its inception, blockchain has proven a successful testbed for a number of beautiful concepts in cryptography such as zero-knowledge proofs, digital signatures and hash functions. But, the blockchain itself offers several attractive properties which are conducive to build cryptography, e.g., provides a decentralized alternative to a trusted setup, which traditionally is performed by a central trusted authority and is inherent for constructing important cryptographic primitives like non-interactive zero-knowledge. In fact, several works view blockchains as an "enabler" for cryptography, much like one-way functions, trapdoor permutations and indistinguishability obfuscation, leading to constructions of concurrent multi-party computation [14], one-time programs [24], time-lock encryption [29] and fair secure multi-party computation [15, 3, 11]. In this work, we take this direction and focus on understanding the round-complexity of two fundamental cryptographic objects: zero-knowledge proofs and secure computation. To study zero-knowledge and secure computation in the world of blockchains, we study the *blockchain-hybrid model*.

The Blockchain-Hybrid Model. Here, the blockchain is modelled as a ledger functionality, and all participants of the cryptographic protocol can access it as an oracle. In particular, the parties (including the adversary) can access the blockchain by posting and reading content, but no single party has any control over the blockchain. Our modeling follows the work of [14] who introduce the blockchain-hybrid model, who in turn adopt the blockchain ledger model from [5].

Compared to the plain model, the presence of blockchain as an oracle is, as we will shortly see, quite empowering. But we emphasize that this is distinct from trusted-setup models explored in cryptography where some trusted party samples a reference string from some good distribution, and then all parties are given access to the string. In particular, in the security proofs, the reduction/simulator is given the power to choose the reference string (e.g., sample along with a trapdoor to aid simulation). *This is in sharp contrast with the blockchain-hybrid model where the reduction/simulator has no control over the blockchain.*

Simulation-security in the Blockchain-Hybrid Model. Zero-knowledge proofs are a cornerstone of modern cryptography. Informally, they are interactive protocols between two parties *prover* and a *verifier*, where the prover is trying to convince the verifier that some string x (also known as an instance) belongs to a language. We want two security properties: (a) Soundness - no cheating prover can convince the verifier on false statements, and (b) Zero-knowledge. The philosophy behind the notion of Zero-Knowledge [23] is that: *whatever an adversary can learn from the proof of a true statement, it could have learnt by itself in polynomial-time.* This counter-intuitive notion of security is formalized by asking the existence of an efficient *simulator* that can simulate the view of any malicious verifier.

While some works [24, 14] have considered zero-knowledge with blockchains, we take a different philosophical approach. We refer the reader to Section 3 for more details on these works.

Time-Traveling Simulation. We present a novel idea to quantify the knowledge an adversary learns. Following is our philosophy: *whatever the adversary could have learnt from this interaction, it could have computed on its own after some T units of time have passed.* Passage of T units of time results in some new blocks being added to the blockchain. E.g., for ZK, our guarantee would mean that whatever the adversarial verifier could have learnt from the given interaction today, it could have computed on its own tomorrow (once the new blocks have been added to the blockchain). More precisely, to prove the ZK property, we require the existence of a simulator who would be "capable of doing time travel", get information about the future blocks, and come up with an indistinguishable transcript. To model this, our simulator will be a given as auxiliary input any valid continuation of the current state of the blockchain for time T. We refer to such a simulator as T-time-traveling simulator, and the resulting notion as T-time-traveling zero-knowledge. We remark that T-time-traveling zero-knowledge coincides with the standard notion of zero-knowledge when $T = 0$, and becomes weaker as T increases.

The above time-traveling zero-knowledge guarantee is almost as meaningful as the original zero-knowledge guarantee as far as the "long-term knowledge" is concerned. As an example, consider the GMW compiler [22] which uses ZK proofs of "honest behavior" to compile a semi-honest secure computation protocol into one that achieves malicious security. Here the witness consists of the private input and the randomness of a party. Hence, passage of T units of time will not give the adversary any further advantage in gaining any knowledge related to the witness. However if a statement is such that for some reason, its witness will inherently be leaked tomorrow (e.g, a time-locked puzzle [36]), our ZK proofs could cause this witness to be leaked to the adversary today (at least as far as the security proofs are concerned). Our ideas can also be seen as being inspired by the construction of Dwork and Naor [17] who used the opening of a timed commitment as the trapdoor witness in a Zap. However, in contrast to their work which focuses on realizing the standard ZK definition, we propose a new notion of security and justify it in the blockchain model. In Section 1.2, we provide a detailed comparison of time-traveling simulation with other weakenings of standard simulation including super-polynomial simulation and majority simulation.

1.1 Our Results

Zero-Knowledge with Time-traveling Simulators. First, we address the round-complexity of zero-knowledge with time-traveling simulation. We show that there exists a simple three-round Zero-knowledge Argument with a strict polynomial time, black-box time-traveling simulator. In the plain model, all of these properties are known to be impossible to achieve using the standard zero-knowledge definition w.r.t. black-box simulation [21]. Even w.r.t. non-black-box simulation, known three round zero-knowledge argument protocols require novel cryptographic assumptions [12].

▶ **Informal Theorem 1** (Zero-Knowledge (See Theorem 6)). *Assuming existence of injective one-way functions, there exists a three-round ZK argument for NP with a polynomial-time black-box time-traveling simulator.*

Looking ahead, we remark that zero-knowledge of our protocol is based on the assumption that injective one-way functions exist. For soundness, we need to rely on a natural assumption on the mining procedure of the underlying blockchain. Informally, we require that an adversary controlling minority of the computational power cannot be too far ahead of the honest miners. Specifically, such an adversary cannot compute k^2 blocks in time the honest miners compute k blocks. We emphasize that if this assumption is violated, then the adversary controls majority of the computing power over the network. This means that the adversary can effectively take over, e.g., the Bitcoin network. Our exact assumption is a variant of this natural assumption for PoW-based blockchains and is described in Assumption 1.

Secure Two-Party Computation with Time-traveling Simulators. We also study the task of secure two-party computation in the blockchain hybrid model. Intuitively, it allows two mutually distrustful parties to compute a joint function over their secret inputs such that neither party learns anything about other party's input beyond what can be learnt already by the function output. Secure computation is foundational task, and round-complexity is an important measure of efficiency. We focus on building two-party protocols between a *sender* and a *receiver* where only the receiver gets the output, but guarantee security against both a malicious sender and a malicious receiver.

Our main result is a three-round protocol for any two-party functionality where we achieve time-traveling simulation security for both parties.

▶ **Informal Theorem 2** (2PC (See Theorem 9)). *Assuming the existence non-interactive statistically binding commitments, IND-CPA secure public-key encryption, EUF-CMA secure public-key signatures, and two-round oblivious transfer, there exists a three-round two-party protocol for any efficiently computable functionality that exhibits a time-traveling strict polynomial-time black-box simulator against malicious adversaries.*

As before, the security is based on standard cryptographic primitives and a natural assumption on the underlying mining procedure of the blockchain as described above (see Section 4.1 for a formal description). We note that three-round 2PC in the plain model was constructed by [2] based on sub-exponential hardness of indistinguishability obfuscation but they achieve standard simulation security only for adversaries with bounded non-uniformity. Further, in the blockchain-hybrid model, [14] rule out even constant-round 2PC w.r.t. standard simulation.

The core technical challenge in achieving simulation-security in three rounds is extracting the input from an adversary. We develop a new two-round commitment scheme that satisfies a weak form of extraction. Specifically, we show that for every cheating committer, there exists a time-traveling extractor that can extract the value committed by the committer. We show that such weak time-traveling extraction guarantees are sufficient to achieve time-traveling simulation.

1.2 Relation of Time-traveling Simulation to Other Security Notions

Super-Polynomial Time Simulation. In super-polynomial simulation or angel-based security models [32, 8], the simulator is given access to a specific form of super-polynomial computing power. For example in [13], the simulator is given access to an "angel" which can break the security of certain commitment schemes. While such power allows for obtaining strong results by bypassing various impossibility results, a key downside is that it can be challenging to argue that such a computing power does not break the security of honest parties (since their security is computational and can potentially be broken using any super-polynomial computation). Arguing the security of other protocols or even that of the ideal functionality (in case the functionality is cryptographic in nature) is even more tricky.

Our simulator can be seen as having access to a very specific super polynomial time angel which can instantaneously compute and provide future blocks on the blockchain. However the key advantage is the guarantee that *any security the angel breaks is limited to whatever would anyway automatically be broken shortly in the future.* This arguably makes our security guarantee stronger, and easier to understand and work with. For example, regardless of which commitment scheme the parties use, the commitment to their input can never be broken by our simulator.

Simulation with a Common Reference String. In the Common Reference String (CRS) model, a trusted party publishes a CRS before the start of the protocol. In the blockchain-hybrid model, the blockchain serves much the same purpose, by allowing parties to agree on the blockchain state. Basing security on blockchains instead of a CRS makes use of existing real-world infrastructure which is already trusted with securing hundreds of billions of dollars. This pre-existing infrastructure does not need to be modified for every secure protocol we wish to run using it, whereas the CRS format may change significantly from protocol to protocol.

A second point of similarity is the extra power provided to the simulator. When formalizing security using the real/ideal world paradigm, often this trusted party also provides the simulator with some trapdoor information related to the CRS, or the simulator itself is able to control the trusted party. This is similar to how a time-traveling simulator receives a future state to aid in simulation. However, a time-traveling simulator does not have power over the blockchain and is not able to "program" the common state. More compellingly, whatever security our simulator breaks is limited to what would have anyway been broken shortly into the future. During the natural course of the blockchain execution, every party will eventually see a state which could have been used to simulate in the past.

Majority Simulation. Goyal and Goyal [24] proposed the notion of majority simulation for protocols based on the blockchain. In it, the simulator controls all honest parties on the blockchain, which is assumed to be the majority of parties. This notion is philosophically very similar to honest majority simulation except that it borrows a pre-existing honest majority setup (the blockchain). In contrast, time-traveling simulators control only a single party. In fact, majority simulation seems strictly stronger than time travel simulation, because control over all honest parties in the blockchain may allow computation of future states.

The ledger functionality in the majority simulation model must be *local* (or "private") to the protocol. This is because otherwise, a majority simulator for protocol Π_1 who controls the bulletin board can break the security of another protocol Π_2 that uses the same bulletin board. In contrast, we identify an achievable set of properties which allow multiple different protocols to use a global ledger functionality at the same time under time-travel simulation (see Section 2.2). Furthermore, the extent of "broken" security in time-travel simulation is very limited: anything which is leaked the adversary would anyway have become automatically public shortly into the future.

Time-Based Primitives. Dwork and Naor [17] proposed using the opening of a timed commitment as the trapdoor witness in a Zap. Since time-based primitives open only with significant computation effort (or with the help of the committer), the simulator needs to put in quite a bit of computational resources. Equivalently, the proofs could reveal information which can only be obtained by using high computation. This information may not automatically be available in the future. On the other hand, in our model, any information which the adversary learns is *automatically* available in the future (without the *adversary* putting in large computational resources).

2 Technical Overview

In Section 2.1, we introduce time-traveling simulators through the lens of zero-knowledge and sketch our three round zero-knowledge argument construction. Then, in Section 2.2, we discuss time-traveling simulators in the context of two-party computation and sketch our three-round 2PC construction. Along the way, we highlight several technical issues specific to simulating in the blockchain hybrid model and constructing time-traveling simulators specifically. For simplicity of exposition, we assume all parties see the most recent $\mathcal{G}_{\mathsf{ledger}}$ state here and address ephemeral consensus issues in the full version.

2.1 Time-Traveling Simulators for Zero-Knowledge

A zero-knowledge argument aims to prove the validity of an NP statement without revealing anything about the witness other than what the verifier could have computed on its own. Informally, this is realized by requiring a simulator which can simulate the view of the adversary without having access to the witness. Slightly more formally, the adversary's view in the following two experiments should be computationally indistinguishable. In the real experiment, the adversary interacts with the honest prover and any honest parties in the blockchain. In the simulated experiment, the adversary interacts only with the simulator, which acts as an interface between the adversary and the rest of the world.

Time-traveling simulation for zero-knowledge aims to capture the notion that the verifier does not learn anything other than what it could have computed on its own after waiting for some time. We formalize this using the same experiments as the standard definition in the blockchain hybrid model, except that the simulator is additionally provided with a future state of the blockchain (e.g. 24 hours into the future). Specifically, if the state of the blockchain at the start of the protocol is st_S, the simulator receives a state st_F which is recognized by blockchain's chain validity predicate as a valid extension of st_S by F blocks. Since the blockchain will, if left alone, generate extensions of itself which are independent of x or its witnesses, st_F is effectively harmless and contains no information about the witness beyond what is naturally leaked with the passage of time.

Dependence on Blockchains. Though it may be extended to other contexts, the notion of time-traveling simulation pairs particularly well with blockchains. Blockchains give a natural notion of time, since blocks are continually added to the chain. Furthermore, since blockchains provide a method of checking whether new blocks are a valid extension of the chain, what it means to be a future state can be quantified as an NP language. Finally, blockchains provide the guarantee that an adversary cannot be too far ahead of the rest of the blockchain (e.g. 24 hours worth of blocks). In contrast, a time-traveling simulator gets a future state for free. This increased power difference allows us to overcome impossibility results for standard simulation.

Three Round Zero-Knowledge. A variety of impossibility results are known for three round zero-knowledge both in the plain model and the blockchain-hybrid model. Time-travel simulation bypasses these impossibility results and gives rise to a remarkably simple three round construction with a strict-polynomial-time blackbox time-traveling simulator. To prove $x \in \mathcal{L}$, the prover and verifier engage in a three round witness indistinguishable proof of knowledge (WIPoK)[1] for the statement "$x \in \mathcal{L}$ or I know a blockchain state F blocks ahead of the current state st_S".

[1] Here we refer to a WIPoK construction for the plain model. The proof of knowledge property is more complicated in the blockchain-hybrid model [14].

Given the future state st_F, the time-traveling simulator can easily compute the proof using a witness for the latter half of the statement. Witness indistinguishability guarantees that the adversarial verifier cannot distinguish between this and a proof using a real witness for x.

Soundness. Despite how much time-travel simulation simplifies the simulation itself, soundness becomes more subtle. First off, note that such proofs can only be sound if it finishes within an a-priori bounded time.[2] To ensure soundness, we need to ensure that the adversarial prover cannot produce an accepting proof using the trapdoor branch. We base this on the assumption that no adversary can get F blocks ahead of the global blockchain, even given access to an oracle which computes a small number of blocks (e.g. $\sqrt{F}$) identically distributed to those mined by honest miners. Intuitively, in order to break this assumption, the adversary would need to compute $k + F$ blocks in the time it takes the honest miners to compute k blocks. This requires significantly more computational power than the honest miners have, contradicting the PoW assumption that the adversary has less computational power than the honest miners [30, 18, 27, 19, 34]. Even given the ability to generate, say, $\sqrt{F}$ blocks for free, the adversary would still need to mine the remaining $F - \sqrt{F} + k$ blocks on its own.[3] This is formalized in Assumption 1.

Using this assumption, a natural approach to proving soundness is to rely on the oracle in conjunction with the proof of knowledge property of the WIPoK to extract a witness for the adversary's proof. Then an adversarial prover which breaks soundness must be able to produce a state F blocks ahead of the global blockchain, violating its security.

The WIPoK knowledge extraction works by rewinding the adversarial prover and replaying messages to obtain two different accepting proofs with the same first message. A witness can be efficiently computed from these. However, as observed by Choudhuri et. al. [14], an adversary may attempt to maintain state across rewinding by posting messages on the blockchain. Any extractor must therefore also rewind the blockchain in order to rewind the adversary. The extractor can do this by using the oracle to compute an indistinguishable fork of the blockchain (i.e. a private chain of blocks which differs from the global blockchain) which it presents to the rewound adversarial prover.[4] Using only a single rewinding, the probability of getting two accepting transcripts from an adversarial prover which breaks soundness is noticeable. Since two accepting transcripts define a witness, an adversarial prover which breaks soundness can also violate the security of the blockchain.

2.2 Time-Traveling Simulators for Secure Two-Party Computation

In two-party secure computation, P_1 and P_2 compute a function f of their inputs x_1, x_2 where P_2 learns the $f(x_1, x_2)$ but nothing else, and P_1 learns nothing. This is formalized using the real/ideal world paradigm. In the real experiment, the adversary interacts directly with the honest party and the blockchain in an execution of the protocol. In the ideal world,

[2] If the prover is allowed to wait until F blocks are added then it immediately gets access to a trapdoor, and can break soundness.

[3] In blockchains such as Bitcoin where the difficulty parameter may vary, the adversary might attempt to use the oracle to decrease the difficulty parameter. However, because the oracle's blocks are identically distributed to those mined by honest miners, the change in difficulty parameter will be identically distributed to an execution where the adversary attempts to directly modify the blockchain's difficulty parameter.

[4] Without the oracle, the adversary might attempt to use data in the blocks about who the miners are (e.g. signatures) is to determine if is being rewound.

the adversary interacts only with the simulator, which acts as an interface to the rest of the world. In turn, the simulator interacts with the blockchain and a trusted third party which reveals only the prescribed output. It is required that the joint distribution of the adversary's view and the honest party's output (as given by the trusted third party) are computationally indistinguishable across the two experiments. Time-traveling simulators additionally receive a future state st_F which is F blocks longer than the current blockchain state.

A unique property of the blockchain-hybrid model which will become relevant later on is that a distinguisher might attempt to use the number of blocks mined during an execution as side channel information to distinguish the two experiments. Since the simulator cannot rewind the global blockchain, it cannot hide how long the simulation takes. Thus a good simulator must be careful not to take more time than a real execution.

Extraction of Inputs. To guarantee security against malicious adversaries, which can behave arbitrarily, the simulator must extract the adversaries input so it can force the appropriate output. In the case of three round protocols, extracting an adversarial P_2's input is particularly challenging for standard simulators, since P_2 sends only a single message. Current solutions rely on non-black-box simulation and only hold for adversaries with bounded auxiliary input [2], or require a simulator which runs in super-polynomial time [6, 7]. Intriguingly, extracting an adversarial P_1's input also appears to be quite challenging in the blockchain-hybrid model. As discussed previously, rewinding based extraction approaches require computing an indistinguishable fork for each rewinding. Even assuming the simulator could do so on its own, the number of rewindings required would blow up the running time of the simulator. This makes the outputs of simulators based on such approaches quite easy to distinguish.

Time-Traveling Extractors (Proposition 7). To solve the issue of timely knowledge extraction, we introduce time-traveling extractors. We design a two round commitment scheme with a time-traveling knowledge extractor.[5] Looking ahead, this will allow parties to commit to their input without revealing any information about it (hiding), while also ensuring they cannot change it later on (binding). A time-traveling extractor for a commitment scheme intuitively shows that any committer will eventually "know" the value that they committed to. Informally, we require the existence of an efficient extractor that, upon receiving a state st_F which is F blocks ahead of the current blockchain, produces both a commitment transcript indistinguishable from a real execution as well as the value committed to in that transcript. This concept naturally synergizes with time-travel simulation, which is our main use case. A key point is that hiding must continue to hold even after the passage of any (polynomial) amount of time.

To construct a two round commitment scheme with a time-traveling extractor, we start from a two round Conditional Disclosure of Secrets (CDS) protocol. A CDS protocol is associated with an NP relation $R_{\mathcal{L}}$ for the language $\mathcal{L}$ and a statement x. In a CDS protocol for a language $\mathcal{L}$, one party P_1 inputs a witness w, and the other party P_2 inputs a secret. If $(x, w) \in R_{\mathcal{L}}$, then P_1 gets the secret as output. However, if $x \notin \mathcal{L}$, then P_1 learns nothing about the secret. Furthermore, P_2 learns nothing about the witness input by P_1.

The commitment scheme proceeds as follows. The receiver sends a non-interactive commitment com_{ni} to the committer and begins the 2 round CDS for the statement "com_{ni} is a commitment to a future state of the blockchain". To commit to a message m, the

[5] This construction may be of independent interest.

committer inputs m to the CDS as the secret. Additionally, to ensure binding, the committer non-interactively commits to m and sends this to the receiver as well.

To extract the message m, time-traveling extractor can commit to the received st_F in com_{ni} and provide the witness to the CDS that com_{ni} is a commitment to a future state of the blockchain. Then the CDS will output m to the extractor.

Hiding is more difficult to ensure. To do so, we need to guarantee that an adversarial receiver cannot commit to a future state. If this is the case, then the CDS statement is false, so an adversarial receiver can learn nothing about the CDS secret m. Similarly to our zero-knowledge protocol's soundness, making this guarantee seems to require some form of extraction from com_{ni}. Goyal and Goyal [24] encounter a similar problem in their construction of non-interactive zero-knowledge using majority simulation. We are able to leverage their insights to extract from com_{ni} in a way which should not break the blockchain - therefore, if com_{ni} contains a future state, the extractor would have produced a future state in a short period of time, violating the security of the blockchain.

They construct such commitments by secret-sharing the message and encrypting the shares under the public keys of parties which have recently mined blocks. This provides hiding against adversaries with static corruptions. Then, given the secret keys of parties which have recently mined blocks, it is straightforward to decrypt and recombine the shares to recover the message. Since access to only a few extra secret keys should not provide the computational power required to get significantly ahead of the global blockchain (see Assumption 2), we are able to show that an adversary which breaks hiding also violates the security of the blockchain.

Note that this extractor requires power which is denied to time-travel simulators. Specifically, time-travel simulators must make do with control over only a single party that has minimal computational power compared to the other miners. As such they cannot gain access to even a small number of other parties' secret keys.[6] However, this method of extraction is sufficient to show hiding for our two round commitment.

Three Round Two-Party Secure Computation (Theorem 9). Given the tools we have constructed so far, a natural approach to three round two-party secure computation is to compile a two round semi-malicious two-party secure computation protocol (which can be implemented using a garbled circuit and oblivious transfer) into the malicious setting. In the first two rounds, P_2 commits to its input using the two round commitment with a time-traveling extractor. In the last two rounds, P_1 also commits to its input using the two round commitment with a time-traveling extractor. Additionally, the parties run the semi-malicious protocol in the last two rounds. To ensure that P_2 behaves honestly, in the last two rounds the parties participate in a CDS protocol where P_2 receives the third message if and only if it inputs a witness to its honesty in the second round. To ensure that P_1 behaves honestly, it proves its honesty using the three round zero-knowledge argument with a time-traveling simulator. The zero-knowledge argument is run in parallel with everything.

The ideas behind simulation are simple, though we will discuss some subtleties later. For time-travel simulation against a corrupted P_2, the simulator can use its future state to extract P_2's input in round two. Then in round three, it uses the extracted input to simulate the semi-malicious protocol and the future state to simulate the zero-knowledge argument. The case for time-travel simulation against P_1 is similar. Using the future state,

[6] We assume that honest parties continue to maintain the semantic security of encryption under their secret keys indefinitely.

the simulator extracts P_1's input in round 3, then provides it to the trusted third party. Otherwise, it behaves honestly using a fixed input (since P_1 does not receive output, there is no need to simulate the semi-malicious protocol).

Malleability. Unfortunately, relying on time-traveling simulators for security both against malicious P_1 and against malicious P_2 introduces malleability issues. In round two, P_2 commits to its input in com_2, which has a time-traveling extractor. In round three, P_1 commits to its input in com_3, which *also* has a time-traveling extractor. When simulating against a malicious P_2^*, we need to rely on the hiding of com_3 while simultaneously extracting P_2^*'s input from com_2. This suggests that com_3's hiding needs to be "stronger" than com_2's hiding ($\mathsf{com}_3 >> \mathsf{com}_2$).

To achieve this, we give an analogue of complexity leveraging for time-traveling simulators. We assume that for times $F_1 << F_2$ (e.g. k and k^2), knowledge of a state st_{F_1} which is F_1 blocks ahead of the current blockchain does not allow the adversary to get F_2 blocks ahead of the blockchain (see Assumption 2). As in our previous assumption, the adversary would still need to compute $F_2 - F_1 + k$ blocks in the time the honest miners compute k blocks, requiring the adversary to have much more computational power than the honest miners.

Complexity leveraging for time-traveling simulators can be applied to ensure $\mathsf{com}_3 >> \mathsf{com}_2$ in much the same way that ordinary complexity leveraging can. We set the parameters of com_3 so that it has an F_2-time-traveling extractor but remains hiding against F_1-time-traveling receivers[7], and set the parameters of com_2 so that it has an F_1-time-traveling extractor. With this setting, a simulator against a malicious P_2^* can use st_{F_1} to break com_2 without disturbing com_3's hiding. Looking ahead, in our secure two-party computation construction we will set up a hierarchy $F_1 << F_2 << F_3$, where messages in round one use the parameter F_1, messages in round two use the parameter F_2, and messages in round three use the parameter F_3.

When simulating against a malicious P_1^*, we have the opposite problem from before - we need to rely on the hiding of com_2 while simultaneously extracting from com_3. Combining this with our previous requirement, we need both $\mathsf{com}_3 >> \mathsf{com}_2$ and $\mathsf{com}_3 << \mathsf{com}_2$. However, complexity leveraging for time-traveling simulators only allows hardness in one direction. To break the circularity, we observe that the extractor for com_2 requires knowledge of the future state *in round one*, but com_3 doesn't start until *round two*. In other words, com_2's period of vulnerability is over before com_3 even begins! We formalize the idea of a limited period of vulnerability using the following security game:

1. The adversary interacts with a challenger who commits in com_2 to either m_0 or m_1.
2. The distinguisher receives com_2 and a future state st_F which is F blocks ahead of the blockchain. It wins if it can determine which message the challenger committed to.

If no efficient adversary/distinguisher pair can win this game with noticeable advantage, we say com_2 has hiding against a time-traveling distinguisher. To show hiding of com_2 while simultaneously extracting from com_3 during simulation against a malicious P_1^*, we can reduce to this property. At a high level, the reduction participates in the above security game, then in step two uses st_F to extract from a locally computed com_3 before deciding which message com_2 contains. To show that com_2 has this property, we rely on the security of the underlying components in the plain model as well as the observation that the ledger (and therefore knowledge of a future state) can be efficiently emulated in the plain model.

[7] F_1 time-traveling receivers which break hiding of com_2 can be used to obtain an F_2 future state (breaking Assumption 2) using the same ideas we developed for showing hiding of the commitments.

Two-Party Computation Construction. We summarize the final construction here. It makes use of the following tools, where $F_1 << F_2 << F_3$.

- A two round commitment com_2 with a F_1-time-traveling extractor and hiding against time-traveling distinguishers.
- A two round commitment com_3 with a F_2-time-traveling extractor and hiding against time-traveling distinguishers.
- A three round zero-knowledge argument with a F_3-time-traveling simulator.
- A two round semi-malicious secure two-party computation protocol.
- A two round conditional disclosure of secrets protocol.

In the first two rounds, P_2 commits to its input in com_2. It also begins the semi-malicious two-party computation protocol and the com_3, as well as provides a witness to its honest behavior into the CDS. P_1 completes the semi-malicious protocol and inputs the resulting message into the CDS as the secret. Additionally it commits to its input in com_3 during the third round and proves its honesty using the zero-knowledge argument, which is run in parallel with everything.

3 Related Work

Zero-knowledge. Round-complexity of zero-knowledge protocols is a fundamental measure of efficiency, and a heavily research topic in the plain model. In fact, in the plain model, we can construct four-round zero-knowledge protocols with black-box expected polynomial-time simulators assuming only one-way functions [9].[8] This result is tight w.r.t. black-box simulation [21], and known three-round zero-knowledge protocols with non-black-box simulation require novel cryptographic assumptions [12].

In the blockchain-hybrid model, the complexity of zero-knowledge was only recently considered by Choudhari et al. [14]. Quite surprisingly, they show that constant-round zero-knowledge protocols are impossible in the blockchain-hybrid model even for expected polynomial-time black-box simulators. Although they give a super-constant round protocol assuming one-way functions, their lower-bound hints that low-interaction protocols are quite hard to come by in the blockchain hybrid model unless simulator is given more power. In another work, Goyal and Goyal [24] present a significant strengthening of simulator's power that allows to achieve non-interactive zero-knowledge. In particular, in their notion of simulation the simulator is given control of all honest miners in the blockchain protocol. Although non-trivial, we believe such a notion of honest-majority simulation as described in [24] to be quite weak.

Secure Two-Party Computation. The round complexity of secure two-party computation is another vibrant area of research. In the plain model, we can construct four-round secure two-party computation protocols (where a single party receives output) with black-box expected-polynomial-time simulators, though three-round protocols with respect to black-box simulation do not exist [26]. Known three-round protocols with non-black-box simulation hold only against adversaries with bounded non-uniformity [2]. Furthermore, in the blockchain-hybrid model, [14] rule out even constant-round protocols w.r.t. standard black-box simulation.

[8] A simulator is *black-box* if it only makes uses of the underlying malicious verifier as a black-box.

A variety of works have achieved protocols with less than four rounds by considering relaxed security notions such as the common reference string model [25, 1] and super-polynomial time simulation [32, 7, 6].

4 Blockchain Modeling and Assumptions

In a blockchain protocol, mutually distrustful parties attempt to maintain and agree upon a global append-only ledger. The ledger consists of an ordered set of blocks which each contain some data. New blocks can only be added by using a special mining procedure which any party (called a miner) can run. Currently, two broad categories of mining procedures exist: Proof of Work (PoW) and Proof of Stake (PoS). In this work, we primarily consider PoW instantiations, though it may be possible to extend our ideas to PoS instantiations.

As in previous work [14, 28, 4], we model the blockchain as a global ledger $\mathcal{G}_{\mathsf{ledger}}$ which internally keeps track of the agreed-upon sequence of blocks. Parties may interact with the ledger via one of the queries specified by the ledger functionality.

We present a shorter, more informal version of our blockchain model here. For the formal version, see the full version.

- $\mathcal{G}_{\mathsf{ledger}}$ maintains an internal state st consisting of blocks submitted by miners (not all submitted blocks are incorporated into st).
- Every party sees some prefix of st. Honest parties' local views are at most $\mathsf{windowSize}$ blocks behind st at any time.
- Blocks consist of messages submitted to the blockchain by any party, along with some information chosen by the miner submitting the block. The miner chooses which messages to incorporate into the block (though this block is not guaranteed to be incorporated into st).
- The blockchain implementing $\mathcal{G}_{\mathsf{ledger}}$ is associated with a predicate $\mathsf{isvalidchain}(\mathsf{st}_1, \mathsf{st}_2)$ which decides whether st_2 is a valid extension of st_1. For example, in Bitcoin this consists of verifying that st_1 is a prefix of st_2 and that all later blocks have an accepting proof of work with respect to the previous blocks. We overload this notation as $\mathsf{isvalidchain}(\mathsf{st}_1, \mathsf{st}_2, k)$ to also check that st_2 is at least k blocks longer than st_1 (i.e. $|\mathsf{st}_2| - |\mathsf{st}_1| \geq k$).
- Define $D_{\mathsf{fut}}(T, F)$ to be the distribution over the future states of the blockchain with length T+F, where the current state is $\mathcal{G}_{\mathsf{ledger}}$'s state at time T. Concretely, the distribution is over the random coins of the miners and parties submitting messages to the blockchain. Every state st_F drawn from $D_{\mathsf{fut}}(T, F)$ satisfies $\mathsf{isvalidchain}(\mathcal{G}_{\mathsf{ledger}}.\mathsf{st}, \mathsf{st}_F, F) = 1$.

Simulation in the $\mathcal{G}_{\mathsf{ledger}}$-Hybrid Model. We consider simulators with the same power as other parties while accessing $\mathcal{G}_{\mathsf{ledger}}$, with the exception of being given a future state. Unlike the setting considered in [24, 15], the simulator does not have full control over the blockchain and cannot "rewind" it by discarding and re-creating blocks. It also cannot quickly compute future states beyond what it was given (see Assumption 2).

The simulator acts as an interface between any parties (e.g. the adversary) it runs internal to itself and $\mathcal{G}_{\mathsf{ledger}}$. It can therefore choose which messages to deliver. Note that the adversary may attempt to post messages to $\mathcal{G}_{\mathsf{ledger}}$ as well as base its behavior on its view of $\mathcal{G}_{\mathsf{ledger}}$.

Since the protocols we construct begin at a specified time (in terms of the number of blocks in $\mathcal{G}_{\mathsf{ledger}}$'s state), the simulator begins at the same time. Without loss of generality, it knows the pointers of all parties at this time. Observe that for every adversary which exists before the protocol, there is another adversary which produces an identical ledger state and tracks the pointers of all parties, by forwarding and tracking messages between the original and $\mathcal{G}_{\mathsf{ledger}}$. Since the simulator inherits the state of the adversary upon starting, it also inherits the pointers of all parties at this time.

4.1 Complexity of Blockchain Mining

First, let us introduce some notation. We use $\mathcal{G}_{\mathsf{ledger}}.\mathsf{st}$ denotes the current state of the $\mathcal{G}_{\mathsf{ledger}}$, and it is a sequence $(B_1, B_2, \ldots,)$ of blocks where each B_i contains some m, a proof-of-work π and we also assume the block contains the public-key of the miner who mined the block.[9] We denote by $|\mathcal{G}_{\mathsf{ledger}}.\mathsf{st}|$ the number of blocks in the $\mathcal{G}_{\mathsf{ledger}}$, and use the shorthand $\mathcal{G}_{\mathsf{ledger}}.\mathsf{size}$ to refer to $|\mathcal{G}_{\mathsf{ledger}}.\mathsf{st}|$. For any natural number $i \leq \mathcal{G}_{\mathsf{ledger}}.\mathsf{size}$, we denote by $\mathcal{G}_{\mathsf{ledger}}.\mathsf{st}[:i]$ as the state of the $\mathcal{G}_{\mathsf{ledger}}$ including only the first i blocks. Finally, we have a predicate $\mathsf{isledgerstate}$ that takes a candidate state st as input, and outputs 1 iff there exists $i \leq \mathcal{G}_{\mathsf{ledger}}.\mathsf{size}$ such that $\mathcal{G}_{\mathsf{ledger}}.\mathsf{st}[:i] = \mathsf{st}$. By $\mathsf{EXEC}^{\mathcal{G}_{\mathsf{ledger}}}(Z, A, 1^\lambda)$ be the random variable denoting an execution of the $\mathcal{G}_{\mathsf{ledger}}$ with the environment Z and some adversary A on security parameter λ. We let $(\mathsf{st}, z, y) \leftarrow\$ \mathsf{EXEC}^{\mathcal{G}_{\mathsf{ledger}}}(Z, A, 1^\lambda)$ denote the random process of running an execution of $\mathcal{G}_{\mathsf{ledger}}$ with (Z, A) until A outputs y where st is the current state of $\mathcal{G}_{\mathsf{ledger}}$ and z is output of Z.

Hardness Assumptions. In this section, we describe the specific hardness we require from the underlying mining procedure. For this, first consider the following relation $R^{\mathcal{G}_{\mathsf{ledger}}} = \{R_\lambda\}_\lambda$ containing pairs defined by $\mathcal{G}_{\mathsf{ledger}}$ functionality.

$$R_\lambda^{\mathcal{G}_{\mathsf{ledger}}} = \left\{ ((\mathsf{st}_s, T), \mathsf{st}_f) : \begin{array}{c} \mathsf{isledgerstate}(st_s) = 1, T \in \mathbb{N}, \\ \mathsf{isvalidchain}(\mathsf{st}_s, \mathsf{st}_f, T) \end{array} \right\} . \tag{1}$$

Intuitively, we want to capture that any adversary, performing bounded number of operations, cannot compute F blocks by the time $k << F$ blocks are added to $\mathcal{G}_{\mathsf{ledger}}$ by the honest miners. We refer to this as (k, F)-security of $\mathcal{G}_{\mathsf{ledger}}$, and in Definition 1 we formalize the notion of what it means for an adversary performing t operations to break (k, F)-security of $\mathcal{G}_{\mathsf{ledger}}$. To allow the adversary to choose the starting state, we consider a two-stage adversary $A = (A_0, A_1)$ where A_0 is arbitrary PPT and participates in an execution of $\mathcal{G}_{\mathsf{ledger}}$ (with the honest miners) to arrive at a starting state st_s, and A_1 is supposed to find an extension of st_s by F blocks, by the time k blocks are added to $\mathcal{G}_{\mathsf{ledger}}$. We overload notation of EXEC by giving it an fourth parameter k which it uses to terminate the execution if k additional blocks are added.

▶ **Definition 1.** *We say that $A = (A_0, A_1)$ t-breaks the (k, F)-security of $\mathcal{G}_{\mathsf{ledger}}$ if there exists some non-negligible function ϵ such that for all $\lambda \in \mathbb{N}$*

$$\Pr\left[((\mathsf{st}_s, F), \mathsf{st}_f) \in R : \begin{array}{c} (\mathsf{st}_s, z, a) \leftarrow\$ \mathsf{EXEC}^{\mathcal{G}_{\mathsf{ledger}}}(Z, A_0, 1^\lambda) \\ (\mathsf{st}_i, z', \mathsf{st}_f) \leftarrow\$ \mathsf{EXEC}^{\mathcal{G}_{\mathsf{ledger}}, \mathcal{O}_{\mathsf{blockify}}}(Z(z), A_1(\mathsf{st}_s), 1^\lambda, k(\lambda)) \end{array} \right] \geq \epsilon(\lambda) \tag{2}$$

where A_0 is arbitrary PPT, and A_1 performs at most $t(\lambda)$ operations and makes at most k queries to $\mathcal{O}_{\mathsf{blockify}}$, where $\mathcal{O}_{\mathsf{blockify}}$ takes as input a block B and a message m, and outputs a new block B' that contains message m such that $\mathsf{isvalidchain}(B, B') = 1$, and R is as defined in Equation (1).

[9] We believe our ideas can also be applied to proof-of-stake blockchains, though we only provide formalization for proof-of-work blockchains.

▶ **Assumption 1.** *For all polynomially bounded functions k, t, there exists some polynomially bounded function F such that no PPT adversary $A = (A_0, A_1)$ t-breaks the (k, F)-security of $\mathcal{G}_{\mathsf{ledger}}$ (as per Definition 1.*

▶ Remark 2. Intuitively, t represents the number of mining operations that an adversary can perform in the time it takes the honest miners to mine k blocks. For the adversary to be able to produce $F > k$ blocks (e.g. $F = k^2$ or even ck, for large enough constant c) in this time, t should be significantly larger than the honest mining power. This would violate the PoW assumption that the adversary controls less mining power than the honest miners. Even given access to an oracle which mines up to k blocks for free (either in the past or currently), the computational requirement on the adversary does not change too much. The adversary would still need to mine the remaining $F - k$ blocks on its own. Since the oracle creates blocks according to the same distribution that honest parties would have, the adversary also cannot use it to decrease the difficulty parameter from what it would have been in a real execution, meaning the work it needs to do for each block does not change.

In fact, for some of security proofs, we will require a stronger notion of security from $\mathcal{G}_{\mathsf{ledger}}$. Specifically, we want it to be computationally hard for A_1 to compute a state F blocks ahead even if its given as auxiliary information (a) some $F' << F$ future blocks, and (b) secret-keys of the most recent miners. As before, we first formalize what it means for such an A_1 to break the security of $\mathcal{G}_{\mathsf{ledger}}$, and next state the exact assumption which we conjecture to hold for PoW-based blockchains, e.g., Bitcoin.

Let $t, \ell, F', k, F : \mathbb{N} \to \mathbb{N}$ be polynomially bounded functions.

▶ **Definition 3.** *We say that $A = (A_0, A_1)$ (t, ℓ, F')-breaks the (k, F)-security of $\mathcal{G}_{\mathsf{ledger}}$ if there exists some non-negligible function ϵ such that for all $\lambda \in \mathbb{N}$*

$$\Pr\left[((\mathsf{st}_s, F), \mathsf{st}_f) \in R : \begin{array}{c} (\mathsf{st}_s, z, a) \leftarrow\$ \mathsf{EXEC}^{\mathcal{G}_{\mathsf{ledger}}}(Z, A_0, 1^\lambda) \\ \mathsf{st}_i \leftarrow\$ \mathcal{S}(\mathsf{st}_s, F'(\lambda)) \\ (\mathsf{st}_j, a', \mathsf{st}_f) \leftarrow\$ \mathsf{EXEC}^{\mathcal{G}_{\mathsf{ledger}}, \mathcal{O}_{\mathsf{blockify}}}(Z(z), A_1(\mathsf{st}_s, \mathsf{st}_i, \vec{sk}_\ell), 1^\lambda, k(\lambda)) \end{array}\right] \geq \epsilon(\lambda) \tag{3}$$

where A_0 is arbitrary PPT, A_1 performs at most $t(\lambda)$ operations and makes at most k queries to $\mathcal{O}_{\mathsf{blockify}}$, where $\mathcal{O}_{\mathsf{blockify}}$ takes as input a block B and a message m then outputs a new block B' which contains message m such that $\mathsf{isvalidchain}(B, B') = 1$, $\mathcal{S}$ on input a starting state st_s outputs an intermediate future state state st_i such that $\mathsf{isvalidchain}(\mathsf{st}_s, \mathsf{st}_i, F') = 1$ and $\vec{sk}_\ell$ are the secret-keys of the miners mining the last ℓ blocks in st_s.

▶ **Assumption 2.** *For all polynomially bounded functions k, t, ℓ, F', there exists polynomially bounded functions F such that no PPT adversary $A = (A_0, A_1)$ (t, ℓ, F')-breaks the (k, F)-security of $\mathcal{G}_{\mathsf{ledger}}$ (as per Definition 3).*

Although Assumption 2 is a strengthening of Assumption 1 we believe, especially in the context of bitcoin, that having access to the secret-keys of the miners and/or some of the future states doesn't make computing states much further in the future any easier. For PoW blockchains such as Bitcoin, having the private keys of recent miners does not make the computational puzzle (based on hashing) any easier to solve. Knowledge of the future state can be seen as a special case of the oracle provided in Assumption 1.

5 Time-Traveling Simulator Definition for Zero Knowledge

We include the formal definition of time-traveling simulators for zero knowledge here. Due to space constraints, we omit the definitions of time-traveling simulators for two-party computation and arguments of time-traveling knowledge. These can be found in the full version.

The security parameter is λ. We denote cryptographic indistinguishability by $\approx_c$.

Though our definitions specify the circuit class for sake of generality, in this work we only consider adversaries which are unable to break the security of the blockchain. Distinguishers, however, are allowed to be arbitrary PPT machines.

Blockchain Awareness. Unless otherwise specified, we consider adversaries which have access to the global functionality $\mathcal{G}_{\mathsf{ledger}}$. For indistinguishability of two distributions, this means the distinguisher also has access to $\mathcal{G}_{\mathsf{ledger}}$ as soon as the samples are generated. One consequence of this is that the distinguisher can check the time an interaction ends as well as wait for future states to become available before attempting to distinguish. Additionally, this means that the distinguisher can immediately distinguish any view generated by a simulator using a privately initialized $\mathcal{G}_{\mathsf{ledger}}$ from the real execution by checking the state of the global $\mathcal{G}_{\mathsf{ledger}}$.

Many of our protocol definitions require some quantification of the time an execution is started, with respect to $\mathcal{G}_{\mathsf{ledger}}$. We denote that an execution of an algorithm or protocol Π is started when the internally held state of $\mathcal{G}_{\mathsf{ledger}}$ has T blocks ("time T") by Π_T. The algorithm may not be aware of the actual state or time of $\mathcal{G}_{\mathsf{ledger}}$.

Similar to [24], we only consider statically corrupting adversaries, though the blockchain may be secure against adaptive corruptions. We believe our assumptions and results also hold against adaptive corruptions with erasures. Achieving our results against adaptively corrupting adversaries without erasures is an interesting open question.

For interactive algorithms P_1, P_2, we denote by $\mathsf{out}_{P_1}(P_1(x), P_2(y))$ the output of P_1 in an interaction with P_2 where P_1 has input x and P_2 has input y. The corresponding view is denoted by $\mathsf{view}_{P_1}(P_1(x), P_2(y))$.

▶ **Definition 4** (Argument Systems). *Let $\mathcal{C}$ be a circuit class. An interactive protocol (P, V) is a $\mathcal{C}$-argument system for a language L with NP relation R_L if it satisfies the following properties:*

- ***Completeness:*** *For every $(x, w) \in R_L$,*

$$Pr\left[\mathsf{out}_V(P(1^\lambda, x, w), V(1^\lambda, x)) = 1\right] = 1$$

- ***$\mathcal{C}$-Soundness:*** *For every $x \notin L$ and every adversary $P^* \in \mathcal{C}$,*

$$Pr\left[\mathsf{out}_V(P^*(1^\lambda, x), V(1^\lambda, x)) = 1\right] = \mathsf{negl}(n)$$

This definition is in the plain model. If these properties hold in the $\mathcal{G}_{\mathsf{ledger}}$-hybrid model and (P, V) may require interaction with $\mathcal{G}_{\mathsf{ledger}}$, then we call (P, V) a blockchain-aware argument system. If soundness holds against unbounded provers, (P, V) is called a proof system.

A time-traveling simulator is given a randomly sampled future state of the blockchain and must produce an indistinguishable transcript from the real-world protocol.

▶ **Definition 5** (Zero Knowledge with Time-Travel Simulation). *Let $\mathcal{C}$ be a circuit class. An argument system (P, V) for a language L is **$\mathcal{C}$-Zero Knowledge with an F-Time-Traveling Simulator** if there is a PPT algorithm* Sim *such that for all adversaries $V^* \in \mathcal{C}$, every $(x, w) \in R_L$, and all times T, the following holds in the $\mathcal{G}_{\mathsf{ledger}}$-hybrid model:*

$$\{\mathsf{view}_{V^*}(P_T(1^\lambda, x, w), V_T^*(1^\lambda, x))\} \approx_c \{\mathsf{Sim}_T(1^\lambda, x, \mathsf{st}_F) : \mathsf{st}_F \leftarrow\$ D_{\mathsf{fut}}(T, F)\}$$

6 Theorems

▶ **Theorem 6.** *Let k be an upper-bound on the numbers of blocks added in an honest execution of $\langle P, V\rangle$. Let t, F be polynomially bounded function such that no PPT A $3t$-breaks the (k, F)-security of $\mathcal{G}_{\mathsf{ledger}}$ (as per Definition 1). Further, assuming the witness-indistinguishability of* WIPOK*, there exists a 3-round argument in the $\mathcal{G}_{\mathsf{ledger}}$ model where soundness holds against cheating prover P^* performing t operations and $(F + 2\mathsf{windowSize})$-time-traveling zero-knowledge holds against all PPT V^*.*

In order to transform this construction into an argument of time-traveling knowledge, we construct a two-round statistically binding commitment scheme with a time-traveling extractor.

▶ **Proposition 7.** *Assume the existence of a public key integrated encryption-signature scheme, a Conditional Disclosure of Secrets protocol, a non-interactive statistically-binding commitment scheme, a threshold secret-sharing scheme, a public-key integrated encryption-signature scheme, and that $\mathcal{G}_{\mathsf{ledger}}$ has (β, ℓ)-chain-quality. Let t, F' be polynomially bounded functions as in Assumption 2 and let $\mathcal{C}_{t,F'}$ be the class of adversaries which receives an F'-future state and performs at most $t(\lambda)$ operations.*

Assuming that no PPT adversary $(2t, \ell + 2\mathsf{windowSize}, F')$-breaks the $(1, F)$-security of $\mathcal{G}_{\mathsf{ledger}}$ (Assumption 2), there exists a statistically binding commitment scheme in the $\mathcal{G}_{\mathsf{ledger}}$-hybrid model with $\mathcal{C}_{t,F'}$-hiding against time-traveling distinguishers. Furthermore, it has an $(F + 2\mathsf{windowSize})$-time-traveling extractor against time-traveling distinguishers.

The commitment scheme has hiding against an interactive receiver with an a-priori bounded polynomial computational power. We emphasize that the restriction on computation power *only* applies to the generation of the transcript - once the transcript is complete, it remains hiding even against arbitrary computationally efficient distinguishers.

▶ **Proposition 8.** *Let k be an upper-bound on the numbers of blocks added in an honest execution of $\langle P, V\rangle$. Let t, ℓ, F', F be polynomially bounded functions such that no PPT A $(2t, \ell, F')$-breaks the (k, F)-security of $\mathcal{G}_{\mathsf{ledger}}$ (as per Definition 3). Assume the existence of a statistically-binding commitment scheme with hiding against all PPT receivers which perform at most $t(\lambda)$ operations and that it has an F'-time-traveling extractor.*

Then there exists a 3-round argument in the $\mathcal{G}_{\mathsf{ledger}}$ model such that $(F + 2\mathsf{windowSize})$-time-traveling zero-knowledge holds against all PPT verifiers which perform at most $t(\lambda)$ operations. Furthermore, it has an F'-time-traveling extractor against every cheating prover P^* performing at most $t(\lambda)$ operations.

For our secure two-party computation protocol, we use the following building blocks. See the full version for constructions and definitions of time-traveling knowledge extractors and hiding against time-traveling distinguishers. In the following, we have a hierarchy of polynomially bounded function $F_1 < F_2 < F_3$.

1. three round delayed-input ZKAoK $\mathcal{L}leP, V\rangle$ such that
   a. F_3-time-traveling simulator for ZK,
   b. F_2-time-traveling knowledge-extractor for cheating provers doing t operations.
2. two round commitment scheme $(\mathsf{com}, \mathsf{Rec}, \mathsf{Open})$ which has
   a. Hiding against (R^*, D^*) where R^* performs t operations and D^* is F_2-time-traveling distinguisher.
   b. An F_1-time-traveling extractor which produces transcripts indistinguishable to F_3-time-traveling distinguishers.
3. Standard cryptographic tools: a two round oblivious transfer protocol $\mathsf{OT} = (\mathsf{OT}_1, \mathsf{OT}_2)$, two round conditional disclosure of secret protocol $\mathsf{CDS} = (\mathsf{CDS}_1, \mathsf{CDS}_2)$, garbled circuit $\mathsf{GC} = (\mathsf{Garble}, \mathsf{Eval})$, a PRF family F, a statistically binding non-interactive commitment $mathsfSBCom$.

The time-traveling building blocks can be constructed using standard cryptographic tools and our blockchain assumptions.

▶ **Theorem 9.** *Assuming the building blocks with appropriate security as described above, there exists a two-party protocol* Π *that securely computes any 2-party functionality* f *with* F_3*-time-traveling simulation.*

References

1 Arash Afshar, Payman Mohassel, Benny Pinkas, and Ben Riva. Non-interactive secure computation based on cut-and-choose. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 387–404. Springer, Heidelberg, 2014. doi:10.1007/978-3-642-55220-5_22.

2 Prabhanjan Ananth and Abhishek Jain. On secure two-party computation in three rounds. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 612–644. Springer, Heidelberg, 2017. doi:10.1007/978-3-319-70500-2_21.

3 Marcin Andrychowicz, Stefan Dziembowski, Daniel Malinowski, and Lukasz Mazurek. Secure multiparty computations on bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 443–458. IEEE Computer Society Press, 2014. doi:10.1109/SP.2014.35.

4 Christian Badertscher, Peter Gazi, Aggelos Kiayias, Alexander Russell, and Vassilis Zikas. Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 913–930. ACM Press, 2018. doi:10.1145/3243734.3243848.

5 Christian Badertscher, Ueli Maurer, Daniel Tschudi, and Vassilis Zikas. Bitcoin as a transaction ledger: A composable treatment. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 324–356. Springer, Heidelberg, 2017. doi:10.1007/978-3-319-63688-7_11.

6 Saikrishna Badrinarayanan, Sanjam Garg, Yuval Ishai, Amit Sahai, and Akshay Wadia. Two-message witness indistinguishability and secure computation in the plain model from new assumptions. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part III*, volume 10626 of *LNCS*, pages 275–303. Springer, Heidelberg, 2017. doi:10.1007/978-3-319-70700-6_10.

7 Saikrishna Badrinarayanan, Vipul Goyal, Abhishek Jain, Dakshita Khurana, and Amit Sahai. Round optimal concurrent MPC via strong simulation. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 743–775. Springer, Heidelberg, 2017. doi:10.1007/978-3-319-70500-2_25.

8 Boaz Barak and Amit Sahai. How to play almost any mental game over the net - concurrent composition via super-polynomial simulation. In *46th FOCS*, pages 543–552. IEEE Computer Society Press, 2005. doi:10.1109/SFCS.2005.43.

9 Mihir Bellare, Markus Jakobsson, and Moti Yung. Round-optimal zero-knowledge arguments based on any one-way function. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 280–305. Springer, Heidelberg, 1997. doi:10.1007/3-540-69053-0_20.

10 Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474, 2014. doi:10.1109/SP.2014.36.

11 Iddo Bentov and Ranjit Kumaresan. How to use bitcoin to design fair protocols. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 421–439. Springer, Heidelberg, 2014. doi:10.1007/978-3-662-44381-1_24.

12 Nir Bitansky, Yael Tauman Kalai, and Omer Paneth. Multi-collision resistance: a paradigm for keyless hash functions. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *50th ACM STOC*, pages 671–684. ACM Press, 2018. doi:10.1145/3188745.3188870.

13 Ran Canetti, Huijia Lin, and Rafael Pass. Adaptive hardness and composable security in the plain model from standard assumptions. In *51st FOCS*, pages 541–550. IEEE Computer Society Press, 2010. doi:10.1109/FOCS.2010.86.

14 Arka Rai Choudhuri, Vipul Goyal, and Abhishek Jain. Founding secure computation on blockchains. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 351–380. Springer, Heidelberg, 2019. doi:10.1007/978-3-030-17656-3_13.

15 Arka Rai Choudhuri, Matthew Green, Abhishek Jain, Gabriel Kaptchuk, and Ian Miers. Fairness in an unfair world: Fair multiparty computation from public bulletin boards. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 719–728. ACM Press, 2017. doi:10.1145/3133956.3134092.

16 Phil Daian, Rafael Pass, and Elaine Shi. Snow white: Robustly reconfigurable consensus and applications to provably secure proof of stake. In Ian Goldberg and Tyler Moore, editors, *Financial Cryptography and Data Security*, pages 23–41. Springer International Publishing, 2019.

17 Cynthia Dwork and Moni Naor. Zaps and their applications. In *41st FOCS*, pages 283–293. IEEE Computer Society Press, 2000. doi:10.1109/SFCS.2000.892117.

18 Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 281–310. Springer, Heidelberg, 2015. doi:10.1007/978-3-662-46803-6_10.

19 Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol with chains of variable difficulty. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 291–323. Springer, Heidelberg, 2017. doi:10.1007/978-3-319-63688-7_10.

20 Juan A. Garay, Aggelos Kiayias, Nikos Leonardos, and Giorgos Panagiotakos. Bootstrapping the blockchain, with applications to consensus and fast PKI setup. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part II*, volume 10770 of *LNCS*, pages 465–495. Springer, Heidelberg, 2018. doi:10.1007/978-3-319-76581-5_16.

21 Oded Goldreich and Hugo Krawczyk. On the composition of zero-knowledge proof systems. *SIAM J. Comput.*, 25(1):169–192, 1996. doi:10.1137/S0097539791220688.

22 Oded Goldreich, Silvio Micali, and Avi Wigderson. How to prove all NP-statements in zero-knowledge, and a methodology of cryptographic protocol design. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 171–185. Springer, Heidelberg, 1987. doi:10.1007/3-540-47721-7_11.

23 Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989. doi:10.1137/0218012.

24 Rishab Goyal and Vipul Goyal. Overcoming cryptographic impossibility results using blockchains. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 529–561. Springer, Heidelberg, 2017. doi:10.1007/978-3-319-70500-2_18.

25 Omer Horvitz and Jonathan Katz. Universally-composable two-party computation in two rounds. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 111–129. Springer, Heidelberg, 2007. `doi:10.1007/978-3-540-74143-5_7`.

26 Jonathan Katz and Rafail Ostrovsky. Round-optimal secure two-party computation. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 335–354. Springer, Heidelberg, 2004. `doi:10.1007/978-3-540-28628-8_21`.

27 Aggelos Kiayias and Giorgos Panagiotakos. Speed-security tradeoffs in blockchain protocols. Cryptology ePrint Archive, Report 2015/1019, 2015. URL: `https://eprint.iacr.org/2015/1019`.

28 Aggelos Kiayias, Hong-Sheng Zhou, and Vassilis Zikas. Fair and robust multi-party computation using a global transaction ledger. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 705–734. Springer, Heidelberg, 2016. `doi:10.1007/978-3-662-49896-5_25`.

29 Jia Liu, Tibor Jager, Saqib A. Kakvi, and Bogdan Warinschi. How to build time-lock encryption. *Des. Codes Cryptogr.*, 86(11):2549–2586, 2018. `doi:10.1007/s10623-018-0461-x`.

30 Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, page 21260, 2008.

31 R. Pass and E. Shi. Hybrid consensus: Efficient consensus in the permissionless model. In *DISC*, 2017.

32 Rafael Pass. Simulation in quasi-polynomial time, and its application to protocol composition. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 160–176. Springer, Heidelberg, 2003. `doi:10.1007/3-540-39200-9_10`.

33 Rafael Pass, Lior Seeman, and abhi shelat. Analysis of the blockchain protocol in asynchronous networks. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 643–673. Springer, Heidelberg, 2017. `doi:10.1007/978-3-319-56614-6_22`.

34 Rafael Pass and Elaine Shi. FruitChains: A fair blockchain. In Elad Michael Schiller and Alexander A. Schwarzmann, editors, *36th ACM PODC*, pages 315–324. ACM, 2017. `doi:10.1145/3087801.3087809`.

35 Rafael Pass and Elaine Shi. The sleepy model of consensus. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part II*, volume 10625 of *LNCS*, pages 380–409. Springer, Heidelberg, 2017. `doi:10.1007/978-3-319-70697-9_14`.

36 R. L. Rivest, A. Shamir, and D. A. Wagner. Time-lock puzzles and timed-release crypto. techreport, Massachusetts Institute of Technology, 1996.

37 Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger, 2014.