As blockchains grow to support more users and more frequent transactions, so too grows the amount of information (the "state") that validators store to verify transactions. For example, in Bitcoin the state consists of a set of unspent transaction outputs (UTXOs). In Ethereum, the state consists of an account balance for each account as well as code and storage for each smart contract.

This storage burden will become unwieldy for blockchains with enough accounts or UTXOs to support a significant portion of the population's truly everyday transactions, making it difficult to become a validator and posing a threat to decentralization. It's tempting to turn to cryptography as a solution, where tools such as Merkle trees and zero knowledge proofs have helped us achieve the implausible before.

This is exactly what "stateless blockchains" aim to do. But despite substantial work on them, they remain far from practical. But this lag in progress, it turns out, is inherent – this gap between these constructions and practicality will never be bridged. Our recent work shows that no stateless blockchain scheme, no matter how clever, will ever be feasible without additional measures to manage the state . As we show at the end of this post, though, this impossibility result should not be discouraging.

The state of the stateless

Today, the state is large but manageable. For example, Bitcoin nodes store about 7 GB of data and Ethereum nodes store about 650 GB . But this storage burden for full nodes increases roughly linearly with the throughput (transactions per second or TPS) of the chain, which is unacceptably low today. With current designs, the state required to truly support everyday transactions (tens to hundreds of thousands of TPS) would be unwieldy, requiring terabytes or even petabytes.

This has motivated a search for technical approaches to dramatically reduce the amount of state required for validators. The holy grail is a stateless blockchain , which would require validators to store only a constant-size state regardless of the transaction throughput. (The term is actually a misnomer: there still is a state, it's just small enough to be practical at any future throughput – typically it's constant-sized.) Such a light storage requirement would make running a validator node much easier; optimistically everyone could run a node on their mobile phone. Since increasing the number of validators increases the security of the chain, lowering the barrier to entry for validators is important.

Despite substantial research on stateless blockchains (by, e.g., Todd , Buterin , Boneh et al. , Srinivasan et al.), they're far from practical, and none, to our knowledge, are deployed. The fundamental problem with all known stateless blockchains is that they require users to store additional data called witnesses to help validators verify transactions involving their accounts. For example, this witness might be a Merkle inclusion proof showing that a user's account and its balance are included in a global state commitment. When a user makes a transaction, they submit this witness to validators, showing that their account has sufficient balance.

Unlike storing a private key, which never needs to change, these witnesses change frequently, even for users who aren't actively transacting, placing an impractical burden on users. Similarly, imagine if you had to continually monitor all other credit card transactions worldwide and update some local data accordingly to use your own credit card. For the blockchain to be practical, users must be able to stay offline and interact with the blockchain only when they submit a transaction. In many cases, like hardware wallets, updating witnesses is not just inconvenient but impossible.

This leads us to a natural research question: Can we build a stateless blockchain that doesn't require witness updates (or requires them only rarely)? To answer this question we developed a novel theoretical framework (the revocable proof system) that generalizes stateless blockchains. Using this framework, we prove a conclusive impossibility result: this tradeoff between succinct global state and frequent witness updates is fundamental. Our proof techniques are information-theoretic, meaning no future computers will be powerful enough to solve this problem: the gap between stateless blockchain constructions and practicality will never be bridged.

Background on our research

To help build intuition for our impossibility result, we'll start by describing a natural though inefficient construction of a stateless blockchain using a Merkle tree . Our goal is for validators to determine whether transactions submitted by users are valid – for example, that the user has a large enough account balance to make the transaction. In a stateless blockchain scheme, validators store a constant-sized state . When a user makes a transaction, they must include with their transaction a witness. The validator can verify using the current state and a user-submitted (transaction, witness) pair that that user has sufficient account balance to make their transaction.

We first construct a Merkle tree where each (account ID, balance) pair (a , b) is included as a leaf. The constant-sized state V that validators store is the root of this tree, which acts as a commitment to the set of account-balance pairs. Each user maintains as its witness a Merkle inclusion proof of its (account ID, balance) pair. A Merkle inclusion proof of a leaf (a , b) consists of the partner nodes $(v 1 , …, vk)$ along its path to the root of the tree. Given a transaction made by a user with account a and claimed balance b , a validator can check that b is indeed the balance of account a by checking the proof $(v 1 , …, vk)$ for (a , b) against its current state V . If so, the validator executes the transaction and must update the account's balance accordingly. A convenient property of Merkle trees is that given a Merkle inclusion proof for a leaf, it's easy to compute the resulting root when that leaf is changed. In other words, validators can easily compute an updated state V' which captures the new balance of account a after the transaction has been executed.

This Merkle tree scheme has two main drawbacks. First, users' witnesses are relatively large, growing logarithmically in the total number of accounts in the system. Ideally, they should be constant-sized, which we can achieve using schemes such as RSA accumulators (studied in the context of stateless blockchains by Boneh et al).

The second drawback is harder to avoid: The proof of an account-balance pair changes whenever any other user transacts. Recall that the proof of a leaf consists of partner nodes along the path from that leaf to the root of the tree. If any other leaf changes, one of these nodes changes, presenting a problem in practice. Most blockchain users want to keep their coins in a wallet passively and come online only when they want to make a transaction. In practice in this stateless blockchain, however, users would have to constantly monitor everyone else's transactions to keep their witnesses up to date. (Although a third party could do this monitoring on behalf of the user, this departs from the standard stateless blockchain model. We discuss this at the end of this post.) Practically, this is an insurmountable challenge for the stateless blockchain, placing a heavy burden on users.

Our result: Statelessness is impossible

This phenomenon is not specific to this Merkle tree construction – all known stateless blockchain schemes require users to frequently update their witnesses, which we demonstrate here. More precisely, we show that the number of users who must update their witness grows roughly linearly with the total number of transactions made by all users.

This means that even if a user Alice makes no transactions, her witness might need to change as other users transact. As long as the succinct state stored by validators is too small to capture the full state (i.e., the set of all account balances), increasing the succinct state size helps little. We plot this relationship as implied by our theorem below, along with the number of witness changes required per day for blockchains of various throughput. These plots show the number of times witnesses will need to change for the best possible stateless blockchain. Here, the data universe refers to the total number of accounts (in the account model) or UTXOs (in the UTXO model).

At the heart of our proof is an information theoretic argument. A core principle of information theory, formalized by Claude Shannon , is that if Alice randomly chooses an object from a set of size 2n and wishes to tell Bob which object she chose, she must send him at least n bits. If there exists a stateless blockchain scheme where users update their witnesses infrequently, Alice can tell Bob which object she chose using fewer than n bits – which Shannon showed is impossible. Therefore, no such stateless blockchain can exist.

For simplicity, we'll describe here the proof of a slightly weaker statement: that there can be no stateless blockchain where users never need to update their witnesses. The key idea is that Alice uses the stateless blockchain scheme to encode her message to Bob. Initially, Alice and Bob both know the full set of account-balance pairs for all n users. Assume that each account has at least one coin. Alice and Bob also both know the stateless blockchain's succinct state V and witnesses wi for all account-balance pairs (ai , bi) . Alice and Bob have also agreed upon a mapping between messages and sets of accounts. Alice will choose a set A of accounts corresponding to her message, then she'll spend coins from these accounts. She'll use the stateless blockchain to communicate to Bob which set she chose, and he can learn from this set what her message was.

Encoding: Alice spends one coin from each account in A . Using the stateless blockchain scheme, Alice computes an updated state V' and sends V' to Bob.

Decoding: For each i, Bob checks whether Verify(wi, (ai , bi)) . Bob outputs the set B of accounts such that Verify(wi , (ai , bi)) = false.

Bob succeeds in outputting the same set that Alice chose: B = A. First, observe that if Alice spent a coin from an account ai , the witness for its old balance should no longer be accepted – otherwise, Alice would be able to double spend. Therefore, for every account ai in A , Verify(wi , (ai , bi)) = false, and Bob will include this account in B. On the other hand, Bob will never include in B an account that Alice did not spend a coin from because these accounts' balances remain the same, and (recalling the relaxed statement we set out to prove) their witnesses never change. Thus, B is exactly equal to A .

Finally, we reach our contradiction by calculating the number of bits that Alice should have had to send to Bob. There were 2 n possible subsets of accounts that she could have chosen, so by Shannon's law, she should have had to send at least n bits to Bob. However, she only sent the constant-sized state V' , which is much shorter than n bits.

(Readers familiar with cryptography might notice that we swept a few details under the rug here; for example, Bob's decoding may fail with negligible probability. Our paper includes the full proof.)

While we described our proof in terms of a stateless blockchain, Alice and Bob can execute a similar too-efficient communication using a variety of other authenticated data structures, including accumulators, vector commitments, and authenticated dictionaries. We formalize this class of data structures using a new abstraction we call a revocable proof system.

Implications of the result

Our result shows that you can't "cryptography the state away" – there's no silver bullet commitment scheme allowing us to build a stateless blockchain where users never have to update their witnesses. The state doesn't disappear but rather shifts

away from the validators and gets pushed onto users in the form of frequent witness updates.

Several other promising solutions that depart from the strict stateless blockchain model do exist. One natural relaxation of this model is to allow a third party – that's neither a user nor validator – to be responsible for storing the full state. This party, called a proof-serving node (and examined most rigorously by Srinivasan et al.), uses the full state to generate up-to-date witnesses on behalf of users. Users can then transact using these witnesses as in a regular stateless blockchain, where the validators still store only a succinct state. The incentives of this system, especially how users compensate the proof-serving nodes, are an interesting open research direction.

While our discussion so far has focused on L1 blockchains, our results also have consequences for L2 systems like rollup servers. Rollups (whether optimistic or ZK) typically take a large state and commit to it using a small value stored on L1. This state includes accounts for each user at the L2. We'd like these users to be able to withdraw funds directly on L1 (without cooperation by the L2 server) by posting a witness to their current account balance. This setup is also an instance of a revocable proof system in our model. In fact, one could argue that stateless blockchains have been implemented in practice, in the form of L2 rollups.

Unfortunately though, this means that our impossibility results directly apply. A user's rollup withdrawal witness must change frequently, or else almost the entire L2 state would have to be written to L1. As a result, today's rollups commonly assume a data availability committee (sometimes called a "validium"), which functions akin to "proof-serving nodes" in helping users compute new witnesses when they are ready to withdraw. Our results show that the warning to users in Ethereum's documentation – "Without access to transaction data, users cannot compute the Merkle proof required to prove ownership of funds and execute withdrawals." – will always apply.

Developing more efficient ways to manage blockchain state will become more critical as blockchain systems grow. Although our result ruling out a stateless blockchain may seem negative, impossibility results are useful to blockchain designers because they tell us to focus our research elsewhere, ideally helping us find a viable solution more quickly.

For more details, you can read the complete paper and watch Miranda's presentation on the paper from the a16z crypto lab seminar series.

Miranda Christ is a PhD student in Computer Science at Columbia University, where she is a member of the Theory Group and a Presidential Fellow. She was a Summer '22 research intern with a16z crypto.

Joseph Bonneau is a Research Partner at a16z crypto. His research focuses on applied cryptography and blockchain security. He has taught cryptocurrency courses at the University of Melbourne, NYU, Stanford, and Princeton, and received a PhD in computer science from the University of Cambridge and BS/MS degrees from Stanford.

The views expressed here are those of the individual AH Capital Management, L.L.C. ("a16z") personnel quoted and are not the views of a16z or its affiliates. Certain information contained in here has been obtained from third-party sources, including from portfolio companies of funds managed by a16z. While taken from sources believed to be reliable, a16z has not independently verified such information and makes no representations about the current or enduring accuracy of the information or its appropriateness for a given situation. In addition, this content may include third-party advertisements; a16z has not reviewed such advertisements and does not endorse any advertising content contained therein.

This content is provided for informational purposes only, and should not be relied upon as legal, business, investment, or tax advice. You should consult your own advisers as to those matters. References to any securities or digital assets are for illustrative purposes only, and do not constitute an investment recommendation or offer to provide investment advisory services. Furthermore, this content is not directed at nor intended for use by any investors or prospective investors, and may not under any circumstances be relied upon when making a decision to invest in any fund managed by a16z. (An offering to invest in an a16z fund will be made only by the private placement memorandum, subscription agreement, and other relevant documentation of any such fund and should be read in their entirety.) Any investments or portfolio companies mentioned, referred to, or described are not representative of all investments in vehicles managed by a16z, and there can be no assurance that the investments will be profitable or that other investments made in the future will have similar characteristics or results. A list of investments made by funds managed by Andreessen Horowitz (excluding investments for which the issuer has not provided permission for a16z to disclose publicly as well as unannounced investments in publicly traded digital assets) is available at https://a16z.com/investments/.

Charts and graphs provided within are for informational purposes solely and should not be relied upon when making any investment decision. Past performance is not indicative of future results. The content speaks only as of the date indicated. Any projections, estimates, forecasts, targets, prospects, and/or opinions expressed in these materials are subject to change without notice and may differ or be contrary to opinions expressed by others. Please see https://a16z.com/disclosures for additional important information.