Private Computing on Public Blockchains

Hugo Krawczyk

Algorand Foundation

Is Algorand…

- Scalable?
  - Yes!
- Decentralized?
  - Yes (Decentral*izable*)
- Secure?
  - Yes!
- Private?
  - Nope! Except for some pseudonymity
  - Do not panic – neither are other L1 chains (with limited privacy in special cases, e.g., Zcash)

Blockchains are non-private…

- … by definition
- They are first and foremost *transparent* (and immutably so)
- All data smart contracts carry is in the clear (it can be encrypted data, but the smart contract itself cannot decrypt it)
- **Smart contracts cannot say:** Hey, here is a secret signature key, use it to sign these messages if the SC logic is satisfied.

But we know how to make blockchain private, don't we?

- Hmm… We know, partially, with Zero-Knowledge (ZK) Proofs, in some cases
  - For example, can prove my balance is about some limit (w/o disclosing the balance)
  - I own the output of a previous bitcoin transaction (w/o pointing to the precise transaction)
  - I carry a certificate vowing for being 21 or older (w/o disclosing my exact age)
  - The output of a given private computation is y (w/o disclosing private information)

(Note: "ZK" often used, wrongly, when referring to compact *verification* SNARKs)

But can a smart contract carry a secret it can use?

For example:

- A SC that carries an encrypted will and only discloses it to the heirs and only upon death of the subject

- A SC equipped with a bitcoin wallet key, that upon some logic, generates a bitcoin transaction (*the SC runs on Algorand*!)

- A SC that with an authorizing multi-signature can move funds from account A to account B (for pre-established accounts A and B, in the same or different chain)

Wouldn’t it be wonderful if we could…

- A SC that carries an encrypted will and only discloses it to the heirs and only upon death of the subject
  - **Running general purpose private contracts**
- A SC equipped with a bitcoin wallet key, that upon some logic, generates a bitcoin transaction (the SC runs on Algorand!)
  - **SC’s in Bitcoin, running on… Algorand! Or, Ethereum SCs with Algorand speed/cost!**
- A SC that with an authorizing multi-signature can move funds from account A to account B (for pre-established accounts A and B)
  - **On-chain social recovery, for any chain!**

YES, WE ~~CAN~~ COULD

- We know how. Implementation is feasible but requires a non-trivial engineering effort
- Let me tell you more…

Major example: State proofs

- Every 250 blocks a *compact* proof is published allowing to verify these blocks
- Enables smart contracts running in other chains to verify transactions in Algorand
- Much better than a *trusted* oracle that ingests and checks all Algorand blocks
- Still non-trivial computation on the non-Algorand SC and verification depends on validity of all previous SP's (also needs to wait to next SP to validate current blocks)

Major example: State proofs (cont.)

- **We can do better!** A service that answers *any query* to the Algorand blockchain with *full consensus authority* – namely, *as trustful as the consensus itself*!
  - Anyone can authenticate the answer with *a single standard signature* against a known Algorand public key (no other work on the verifying chain needed)
- Queries can range from individual transaction to a block hash to any statement about the state of the blockchain

➔ Simplified and secure (“trustless”) bridges

And many more applications (let your imagination fly…)

- Carrying private medical data within a SC (controlling the data, using it selectively, and keeping it private!)
  - Note: policy/logic needs transparency, not of the data
- Enabling “tunable privacy” (for auditing and regulations, managed by SC)
- Cryptography for the end user. Key management for:
  - Decentralized social networks, messaging systems
  - User-protected storage, including backed-up wallets!
- More: trustless randomness beacons, auctions, voting, defenses against MEV, …

Threshold Cryptography to the Rescue

- It is easy to do all the above with a trusted third party, say a server, that keeps signature and encryption keys and follows faithfully the logic of a SC
- So “all we need” is to decentralize that trusted server
- Enter: Threshold cryptography (a branch of *secure multi-party computation*)
  - Cryptographic keys shared among a set of servers
  - Needs a threshold of them to produce a signature (or decryption in encryption applications)
  - Breaking to less than a threshold achieves nothing for the attacker
- Traditionally, a small-to-moderate number of servers, say 2 to 50
- But here we want security in a network of many thousands to millions parties

Scalability

- 100,000 servers, 1M eventually? That is a lot of work, and communication
- We want that total work/communication be *independent* of total # participants
  - Sounds familiar?

That's exactly what Algorand consensus does!

- Algorand: Many participants but only a *small committee* active at any given time
  - Work is proportional to committee size, *independent of the total number of participants*
  - Committees are *unpredictable and short lived* to prevent targeted corruption
  - They are chosen *in proportion to their stake*!

Committee-based computation (à la Algorand)

- Many participants but only a *small committee* is active at any given time (work is proportional to committee size, *independent of the total number of participants*)
- We can do that too!
- At each epoch, a committee runs the threshold signature for multiple apps (SCs)
  - Committee chosen via a *Threshold VRF* so that <n/3 stake is adversarial
  - **Stake-weighted trust in the committee = trust for consensus**
- When their work is done (can last for a few rounds depending on network conditions) *they re-share their keys to the next committee*

Compromise of participants in one committee is useless for attacking shares from another committee

Recap

- Blockchains are natively non-private
- Smart contracts cannot act on private data
- Ability to perform secret-key operations (by SCs) on the blockchain opens a world of otherwise impossible (decentralized) applications
- We developed an architecture and solution to enable such applications
- Is this practical? Yes, though details vary by application.
- Is it implemented? Not yet. A serious but feasible engineering effort
  - Best suited for Algorand since it works in tandem with consensus hence providing same security level as the consensus itself

Thank you!

- Paper with the design of threshold signatures in the above setting/approach

  SPRINT: High-Throughput Robust Distributed Schnorr Signatures,

  F. Benhamouda, S. Halevi, H. Krawczyk, Y. Ma, T. Rabin, https://ia.cr/2023/427

- Inspired in prior work on the "YOSO model"
  - *Can a Public Blockchain Keep a Secret?* F. Benhamouda, C. Gentry, S. Gorbunov, S. Halevi, H. Krawczyk, C. Lin, T. Rabin, L. Reyzin, TCC 2020, https://ia.cr/2020/464
  - *You Only Speak Once: Secure MPC with Stateless Ephemeral Roles*, C. Gentry, S. Halevi, H. Krawczyk, B. Magri, J. B. Nielsen, T. Rabin, S. Yakoubov, Crypto 2021, https://ia.cr/2021/210
  - *Threshold Cryptography as a Service (in the multi-server and YOSO models),* F. Benhamouda, S. Halevi, H. Krawczyk, A. Miao, T. Rabin, CCS 2022