

## TLDR

: We suggest using SGX as a pragmatic hedge against zk-rollup SNARK vulnerabilities.

Thanks you for the feedback, some anonymous, to an early draft. Special thanks to the Flashbots and Puffer teams for their insights.

## Construction

Require two state transition proofs to advance the on-chain zk-rollup state root:

1. cryptographic proof

: a SNARK

1. 2FA

: an additional SGX proof

SGX proofs are generated in two steps:

1. one-time registration

: An SGX program first generates an Ethereum (pubkey, privkey)

pair. This keypair is tied to a specific SGX enclave, and only signs valid state transitions (pre\_state\_root, post\_state\_root, block\_root)

. The pubkey

is registered on-chain by verifying an SGX remote attestation which attests to the pubkey

being correctly generated.

1. proving

: After registration the SGX program runs the rollup state transition function to sign messages of the form (pre\_state\_root, post\_state\_root, block\_root)

. These SGX proofs are verified on-chain by checking signatures against the registered pubkey

.

## Context

Early zk-rollups are prone to SNARK soundness vulnerabilities from circuit or proof system bugs. This is concerning because:

1. complexity

: The engineering of zk-rollups is particularly complex. Even bridges, an order of magnitude less complex than rollups, are [routinely hacked](#).

1. value secured

: The value secured by leading zk-rollups is expected to become significantly higher than that of today's bridges. These large bounties may be a systemic risk for Ethereum.

1. competition

: The zk-rollup landscape is competitive, with first-mover advantages. This encourages zk-rollups to launch early, e.g. without multi-proofs. (See Vitalik's [presentation](#) and [slides](#) on multi-proofs.)

## Discussion

SGX 2FA is particularly attractive:

- safety

: There is no loss of safety to the zk-rollup—the additional requirement for SGX proofs is a strict safety improvement. Notice that the SGX enclaves do not handle application secrets (unlike, say, the [Secret Network](#)).

- liveness

: There is almost no loss of liveness. The registration step does require Intel to sign an SGX remote attestation but: \* a) The specific SGX application Intel is providing a remote attestation for can be hidden from Intel. Intel would have to stop providing remote attestations for multiple customers to deny remote attestations for a targeted zk-rollup.

- b) Hundreds of SGX enclaves can register a pubkey prior to the rollup launch. The currently registered pubkeys can generate SGX proofs even if Intel completely stops producing remote attestations for new registrations.
- c) If required, rollup governance can remove the SGX 2FA.
- a) The specific SGX application Intel is providing a remote attestation for can be hidden from Intel. Intel would have to stop providing remote attestations for multiple customers to deny remote attestations for a targeted zk-rollup.
- b) Hundreds of SGX enclaves can register a pubkey prior to the rollup launch. The currently registered pubkeys can generate SGX proofs even if Intel completely stops producing remote attestations for new registrations.
- c) If required, rollup governance can remove the SGX 2FA.
- gas efficiency

: The gas overhead of verifying an SGX proof is minimal since only an Ethereum ECDSA signature is being verified on-chain (other than the one-time cost of verifying remote attestations).

- latency

: There is no additional proof latency—producing SGX proofs is faster than producing SNARKs. Notice that SGX 2FA provides little value to optimistic rollups which have multi-day settlement and can use governance to fix fraud proof vulnerabilities.

- throughput

: There is no loss of throughput. The Flashbots team has shown Geth can run at over 100M gas per second on a single SGX enclave. If necessary multiple SGX enclaves can work in parallel, with their proofs aggregated.

- computational resources

: The SGX computational resources can be minimal. When the state transition is run statelessly (e.g. see [minigeth](#)) there is no need for an encrypted disk and minimal encrypted RAM is required.

- simplicity

: The engineering of SGX is easy relative to SNARK engineering. Geth can be compiled for Gramine with [an 11-line diff](#). The [Puffer](#) team is working on a Solidity verifier for SGX remote attestations, supported by an Ethereum Foundation grant.

- auditability

: Auditing the 2FA should be relatively straightforward. The SGX proof verification logic is contained and the incremental smart contract risk from introducing SGX 2FA is minimal.

- flexibility

: Enclaves from non-Intel vendors (e.g. [AMD SEV](#)) can replace or be used in parallel to SGX enclaves.

- bootstrapping

: 2FA can be used alone—without SNARK verification—to bootstrap an incremental rollup deployment. (This would be similar to Optimism launching without fraud proofs.)

- upgradability

: The SGX proof verification logic is upgraded similarly to the SNARK verification logic. Previously registered pubkeys are invalidated and the definition of what constitutes a valid pubkey is changed by upgrading the remote attestation verification logic.

- deactivation

: SGX 2FA is removable even without governance. For example, the 2FA could automatically deactivate after 1559 days.

There are also downsides to SGX 2FA:

- memetics

: SGX has a bad reputation, especially within the blockchain space. Association with the technology may be memetically suboptimal.

- false sense of security

: Easily-broken SGX 2FA (e.g. if the privkey

is easily extractable) may provide a false sense of security.

- novelty

: No Ethereum application that verifies SGX remote attestations on-chain could be found.