slug: block-building-inside-sgx title: Block Building inside SGX authors: [metachris, frieder] tags: [privacy, sgx] image: /img/posts/sgx-og.jpg hide_table_of_contents: false description: We are excited to announce that Flashbots is running a block builder inside an SGX enclave, and open-sourcing all the components. forum_link: https://collective.flashbots.net/t/block-building-inside-sgx/1373

We are excited to announce that Flashbots is successfully running a block builder inside an SGX enclave, a trusted execution environment (TEE) developed by Intel. The SGX block builder is now live on the Ethereum Sepolia testnet, and soon on mainnet!

Our previous work on running Geth inside SGX demonstrated the technical feasibility of this approach. Now, we've made the next step towards block building inside encrypted enclaves, and want to share our key learnings and challenges, as well as the all the code and tooling for running a block builder inside SGX.

Implementing block building inside encrypted enclaves brings us one step closer toward transaction confidentiality and decentralization of the block building role.

Running a block building algorithm inside an enclave ensures that block builders, as well other infrastructure providers, can no longer see the contents of user transactions, and run verifiable block construction algorithms on them, ensuring economically efficient blocks without compromising on user privacy.

Looking to the future, builders inside SGX can make blocks that are provably valid and truthfully report their bid size, possibly obviating the need for mev-boost relays.

Furthermore, it is an important step toward curbing the risks of exclusive orderflow by allowing transactions to be private yet accessible to all block builders operating inside enclaves.

Our Sepolia SGX Builder

The Flashbots SGX builder on Sepolia uses the public key `0x9817fc7d....`, and the block extra_data `Illuminate Dmocrtz Dstrib Prtct`.

- Delivered blocks can be found through the relay data API
- Example block here: sepolia.etherscan.io/block/3013275

Code & Tooling

The code and tooling required to run an SGX builder is available in this repository: github.com/flashbots/geth-sgx-gramine.

- See also the builder quickstart in the README
- And an overview of all the changes in the pull-request

Hardware

We're using Azure DCsv3 Series hardware, specifically:

- For Sepolia: Standard_DC2s_v3 instances, with 2 cores and 16GB memory + 64GB SSD Swapfile.
- For Mainnet: Standard_DC48s_v3 instances, which have 48 Cores, 384GB Memory + 2TB Premium SSD Disk for Swap.

Code Details

The SGX builder release includes several improvements and additions to the original geth-inside-SGX codebase:

- Fully working Flashbots block builder (branch `capella-test`)
- Remote attestation support, allowing anyone to verify the SGX proof:
    - A remote-attestation TLS (RA-TLS) certificate is created on startup, which embeds the SGX attestation report.
    - See here and here for more information about RA-TLS, as well as the patch for adding it to the builder codebase.
    - RA-TLS is used by several SGX projects including Gramine, Open Enclave and Inclavare Containers.
- Support for EDMM (enclave dynamic memory management) via an update to Gramine 1.4. EDMM obviates the need to reserve and write all memory on startup, which has the potential to significantly speed up the startup process.
- Guide for attesting the certificate of an SGX builder
- Reproducible builder Docker image, arriving at a reproducible MRENCLAVE value when rebuilding the builder Docker image.

Challenges

Geth state size

At the time of writing, the geth database requires about 1TB of storage.

- The database needs to be encrypted and copied into the enclave on every startup. Given the full mainnet geth database with about one terabyte, the startup process takes about 4.5 hours.
- This also means it takes a long time to reproduce bugs.
- Gramine is crashing at 1TB of state in the enclave (we've opened a bug report)
- A promising direction is this geth PBSS patch (auto-pruning storage) which might be able to reduce the state size to about 150GB (see also this post for more details)
- **Edit:** We made it work by keeping the ancient database outside the enclave, halving the state required inside (to ~500GB).

Merging performance and additional latencies

- The SGX builder currently performs at around ~150 mgasps, which is about half the performance of a non-SGX builder.
  - mgasps means "Million Gas Per Second", measured whenever a new block is processed by geth, providing a good indicator for performance.
  - Practically, this means that the SGX builder can only do half the simulations of a non-SGX builder running the same block construction algorithm.
- A likely culprit for this degradation is the way Golang handles syscalls. Instead of using the libc syscall interface, Golang calls directly into the kernel. This produces additional execution overhead when running inside an SGX enclave. Possible improvements could be made by building geth with gccgo

Allowing public access to the builder & preventing spam

- Public users want to connect directly to the code inside the enclave (via TLS), to be able to verify the system is running inside SGX and the code is exactly what they expect.
- This also means the request content cannot be inspected by any part of the outside infrastructure, which would simplify TLS termination, rate-limiting, spam prevention, load-balancing and reputation management.
- In the current single-SGX-instance setup, the single builder instance would receive all the user connections, terminate the SSL connections, and process the request. However, the builder wants to focus on building, not on terminating TLS connections or dealing with spam.
- A stopgap solution that's still permissionless could be applying heavy network-based rate-limiting.
- In order to achieve a scalable SGX infrastructure, we'll publicly explore architectures for end-to-end SGX designs that can deal with outside requests in a scalable fashion, and prevent spam through pre-simulation.

Reproducible builds

- We achieved reproducible SGX-builds for the builder application itself, but the Gramine base as well as certain system libraries aren't yet reproducible and the upstream releases need to be trusted. We are working with the teams towards for end-to-end reproducible builds.
- More information on reproducible Gramine builds are here and here).

Future Work

- Running a SGX builder on Mainnet (soon™).
- Allowing users to submit bundles directly to the SGX builder.
- Exploring designs for a scalable SGX infrastructure in public, potentially paving the way for distributed building designs.
- End-to-end reproducible builds.
- Performance improvements, in particular around system startup and block merging.
- Explore impact and mitigation of covert-channel attacks (leaking private information through resource usage patterns) as well as side-channel attacks and other vulnerabilities.
- Mutual TLS & mutual attestation, allowing multiple SGX enclaves to establish a trusted connection by mutually verifying their SGX proofs.
- SGX'izing further parts of the stack for e2e transaction confidentiality (i.e. mev-boost relay, searcher stack, bundle relay, ...).
- Exploring other TEE systems besides SGX (i.e. AMD SEV, ARM, Nvidia H100+ coprocessors, etc.)

Thanks

Thanks to konVera for the great collaboration on this project, as well as to Mateusz Morusiewicz and Tomasz K. Stańczak for the work towards making this happen. Also thanks to Kailin Rutherford, Robert Annessi, Alexo, Bert and Tina Z for reviewing drafts of this post.

References

- You can find our code and tooling for running a builder inside SGX atgithub.com/flashbots/geth-sgx-gramine.
- See also the first post in the SGX series:Running Geth inside SGX.
- On distributed block building (and exploring the builder design space, broadly) by Alex Stokes.
- **Join us in the forum to discuss and collaborate!**collective.flashbots.net/t/block-building-inside-sgx/1373

Updates

2023-03-14: We successfully run a mainnet SGX builder by separating the ancient database from the SGX state database.

- More details in thepull-request: geth-sgx-gramine#10
- The SGX builder landed the first block on mainnet: https://etherscan.io/block/16813125