Private Data Availability on Celestia

Motivation

Want to provide a credibly neutral place where data availability (DA) for the *general public* has strong assurances, but also *selective disclosure* of the data's contents. As anyone *may* be able to retrieve the data, we want a way to define the conditions that enable *reading the content* of the data.

With a decentralized DA network like Celestia, protocols requiring access to this data have robust assurance that, once published, data cannot be withheld from any party (retrievability) for a period of time. With Data Availability Sampling (DAS), anyone can succinctly verify that DA has occurred without retrieving the original data.

Research Considerations

Requirements

DA *must* be tightly coupled with protocols to *selectively disclose* [parts of] the some otherwise private data contents to defined parties with specific conditions being fulfilled.

Properties of the protocol that *should* apply to provide value in using Private DA

- *Require* publishing data to DA for a protocol to progress
  - Integrate *verification* that DA has occurred
- *Require* data published *contains* the *correct* information
  - *Verifiable* encryption methods
- Provide methods to reveal data contents
  - Decryption methods with key generation/derivation assurances

Tools & Techniques

We want *verifiable computation of encryption* . A "Verifiable Virtual Machine" abstractly is likely what we want, although custom purpose-built mechanisms should be considered based on performance needs.

zkVM / VVM

Run encryption program inside of Verifiable Virtual Machine (VVM aka zkVM minus formal "ZK" guarantees) that produces output sealed by some verifiable attestation.

- RISC Zero / SP1 / Others?
- **Example: https://dorahacks.io/buidl/14098 - Stock0** - protocol to sell (photo) data using ZKPs & smart contract escrow & DA. Uses zkVM to encrypt data in a verifiable way.

TEEs

Trusted Execution Environments enable confidential and verifiably correct computation:

Process based TEE direct use

Directly accessing trusted hardware (like Intel's SGX) via tools is possible today, here are a few options:

- https://www.fortanix.com/
  - https://github.com/fortanix/rust-sgx rust based tooling to develop applications - essentially just a compilation target for `rustc` for any application.
  - https://www.fortanix.com/resources/solution-briefs/secure-key-management-for-blockchain-applications
- https://gramineproject.io/
  - TODO: research & summary
- **https://enarx.dev/docs/start/tee**
  - **WASM (WASI) runtime that occurs in TEE** (likely needs VM based hardware, not process based?)
  - TODO: research & summary

VM based TEE

Indirectly accessing trusted hardware via running an application in a special container/VM is an emerging tech, here are a few options:

- Extending base TEE functionality, running containers/VMs in a verifiable way is a recent phenomenon. Intel is leading here as well, but others are in the game:
- Intel Trust Domain Extensions (TDX)
- AMD Secure Encrypted Virtualization (SEV)
- ARM Confidential Compute Architecture (CCA)

Tooling to access this hardware has an emerging ecosystem of options:

- Low level working example: https://docs.trustauthority.intel.com/main/articles/tutorial-tdx-workload.html - TDX workload example using a key management service to send confidential data exposed only within an attestable VM to perform some operations on. (We replace ML with any task, {another layer of } encryption in our case.)
- Kata Containers -
  - Confidential Containers (CoCo) - Framework building on Kata Containers and integrates with Kubernetes.

Provider options for TEEs

Vendor	Tech	TEE Type	Supports Confidential VMs/Containers?
Intel	TDX, SGX	Memory & VM	Yes (Azure, GCP, Enarx, Kata)

Vendor	Tech	TEE Type	Supports Confidential VMs/Containers?
AMD	SEV-SNP	Memory & VM	Yes (Azure, CoCo, Kata)
ARM	CCA	Memory & VM	Early stage
IBM	Secure Execution	VM Isolation	Yes (used in IBM Cloud)
AWS	Nitro Enclaves	Enclave (not V…	Yes (via EC2)

zkVMs & TEEs

Hybrid architectures could be of great interest to provide confidential operations in the context of a zkVM prover - just hiding specific inputs and/or a subset of operations, up to fully confidential wrapper on the zkVM.

- Create an "accelerator" / co-processor for a zkVM that verifies a TEE attestation & extracts output from some program (i.e. signing with a hidden-to-the-operator key, perhaps on a transaction used in the zkVM) .
- Wrap the entire zkVM prover inside a VM based TEE, enabling fully encrypted inputs and prover process memory from the host/operator. TEE provides attestation of confidentiality and correctness, and zkVM provides another layer of correctness & integrity constraints.
    - Note: GPU based TEEs are available now, and add overhead on the order of single-digets in cost, so likely this wrapping would be practically efficient.
    - See https://phala.network/posts/performance-benchmark-running-sp1-zkvm-in-tee-h200-with-low-overhead for a concrete example of this
    - **https://github.com/Phala-Network/zk-sgx-attester - convert a SGX TEE -> groth16 to post on EVM networks**
        - https://github.com/base/nitro-validator/ for AWS nitro (but all solidity, we would want zkvm conversion)
    - **https://github.com/automata-network/automata-dcap-attestation - This repo serves as a code base for the Intel Data Center Attestation Primitive (DCAP) Web3-based Quote Verification program for both EVM and Solana.**

FHE / MPC (?)

On it's own, the verifiable property is lacking, even with MAC protections, there is (probably) no way *without decrypting* to prove that an untrusted party returned correct data.

- Enable semi/untrusted re-encryption of encrypted data. Coupled with some MAC scheme, a verifiable encryption may be possible to achieve. Today, MPC is required for FHE properties, eventually single party confidential computation is perhaps possible.

coSNARKs

Collaborative SNARKs (coSNARKs) merge the strengths of MPC and zkSNARKs. They allow multiple parties to collaboratively create a proof verifying the correctness of a computation while keeping their individual inputs private.

- https://docs.taceo.io/docs/primer/collabSNARKs-primer/ is one promising implementation.
- **We now have a Noir implementation of verifiable encryption needed for PDA started here: https://github.com/nuke-web3/noir-verifiable-chacha**

Purpose-built Cryptography

There are a few verifiable encryption protocols proposed with some having a demo implementation.

- Verifiable Encryption from MPC-in-the-Head (2024) is one such promising one, as an example.
  Implementation (rust with ASM & c++ deps)
  https://github.com/akiratk0355/verenc-mpcith
- TODO: References in paper above are a great source for more resources & work to investigate for other novel crypto verifiable encryption protocols

Concrete Proposal

See -> Proposal - Private Data Availability

WIP private proxy

References

1. Encrypted DA - original meeting notes outlining requirements
    a. Copy of Proposal: Threshold Encryption System for Celestia blobs - followup: initial proposal
2. Celestia <> Hibachi Kick-off 20 Mar 2025 - call defining a few things around the first customer.

Alternatives to Private DA

Single Cloud

- One trusted owner & operator of service to retrieve data

Redundant Cloud

- Multiple [independent] trusted parties running separate services to retrieve [identical] data

Central Cloud Storage

-

Private/Permissioned DA Network

-