The Journal of Open Source Software

Komodo: Cryptographically-proven Erasure Coding

Antoine Stevan [1*], **Jonathan Detchart** [1*], **Tanguy Pérennou** [1], and **Jérôme Lacan** [1]

1 ISAE-SUPAERO, France ***** These authors contributed equally.

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Daniel S. Katz

Reviewers:

- @cassiersg
- @ankitabanerjee015

Submitted: 04 December 2025
Published: unpublished

Summary

We present **Komodo**, a library that allows to encode data with erasure-code techniques such as Reed-Solomon encoding, prove the resulting shards with cryptographic protocols, verify their integrity on the other end of any distributed network and decode the original data from a subset of said shards (A. Stevan et al., 2024) and (Antoine Stevan et al., 2023). The library is implemented in the *Rust* programming language and available on the ISAE-SUPAERO GitLab instance [1] with a mirror on GitHub [2], both released under the MIT license. **Komodo** should be of interest for people willing to explore the field of cryptographically-proven shards of data in distributed systems or data availability sampling settings.

Komodo provides a *Rust* API to achieve the following on any input data in a distributed network or setup:

- `encode`: encodes data into *shards* with a (k, n) code. This adds redundancy to the data, making the network more resilient to failure, fragmentation, partitioning, loss or corruption.
- `commit` and `prove`: generate cryptographic commitments and proofs for all n encoded shards with one of three available cryptographic protocols (see below for more information). This step consists in attaching extra information to them and sharing augmented *blocks* of data onto the network. This extra information should guarantee with a very high probability that a given shard has been generated indeed through an expected encoding process, namely a polynomial evaluation or vector inner-product encoding such as Reed-Solomon.
- `verify`: verifies any shard individually for its validity. This allows to discriminate invalid or corrupted shards without any decoding attempt. Without this shard-level verification step, it is impossible to know if a shard is valid until the decoding step. Then, when decoding fails, it is not possible to know which shards were invalid, leading to a *try-and-error* process that is not scalable.
- `decode`: decodes the original data using any subset of k valid shards.

Komodo provides the three following cryptographic protocols:

- **KZG+**: based on (Kate et al., 2010) and its multi-polynomial extension (Boneh et al., 2020)
- **aPlonK**: based on **PlonK** (Gabizon et al., 2019) and **aPlonK** (Ambrona et al., 2023)
- **Semi-AVID**: based on **Semi-AVID-PR** (Nazirkhanova et al., 2022)

Komodo is based on the Arkworks library (contributors, 2022) which provides implementations of elliptic curves, fields and polynomial algebra.

[1] GitLab source code: https://gitlab.isae-supaero.fr/dragoon/komodo
[2] GitHub mirror for issues and pull requests: https://github.com/dragoon-rs/komodo

Keywords

Cryptography; Erasure codes; Distributed systems; Verifiable information dispersal; Data availability;

Statement of need

Komodo provides mechanisms that satisfy various distributed systems' needs such as verifiable information dispersal or data availability. Such systems range from private drone swarms to public blockchains.

For instance, in a distributed storage system, nodes can encode data into shards, prove their integrity, and distribute them across the network. Other nodes can then verify the shards' validity before storing or retrieving them, ensuring data robustness and trustworthiness.

In blockchain systems, **Komodo** can be used as the key enabling mechanism for checking data availability, similar to how 2D Reed-Solomon codes and Danksharding (*What Is Proto-Danksharding?*, 2024) are used within Ethereum 2.0, or similar mechanisms in the Celestia or Avail blockchains, among many others.

A few libraries provide similar functionalities, with a few gaps filled by **Komodo**.

The `arkworks` ecosystem (contributors, 2022) is probably the closest library, providing many of the necessary building blocks involved in Data Availability Sampling: prime fields, possibly paired with elliptic curves like BLS12-381 or BN254 among many others; linear algebra operations like polynomial operations and polynomial commitment. On top of those features, **Komodo** adds Reed-Solomon encoding, tightly integrated with proof generation.

The Rust implementation of Reed-Solomon erasure coding (rust-rse contributors, 2021) provides mechanisms to encode and decode data into raw shards, using elements of finite fields $\mathbb{F}_{2^8}$ or $\mathbb{F}_{2^{16}}$, containing respectively 2^8 and 2^{16} elements. **Komodo** adds the proving mechanisms, and makes it possible to use elements from `arkworks`' prime fields.

Komodo also adds a unified high-level API, allowing to benchmark and compare different combinations of prime fields, elliptic curves and polynomial commitment schemes, as we did in two publications (Antoine Stevan et al., 2023; A. Stevan et al., 2024). Finally, a modular design allows to extend **Komodo** with new polynomial commitment schemes or new encoding methods, which performance can be evaluated in the same benchmarking conditions.

Some measurements

Building on the work from (A. Stevan et al., 2024), we have conducted some measurements of the performance of the three methods. All experiments were run on a laptop with x86-64 Intel Core i7-12800H (14 cores / 20 threads, 0.4–4.8 GHz) system with a 3-level cache hierarchy (L1d 544 KiB, L1i 704 KiB, L2 11.5 MiB, L3 24 MiB) and a single NUMA node. Only one thread was used for all experiments.

The time to run `commit`, `prove` and `verify` has been measured for $k = 8$ and a code rate $\rho = \frac{1}{2}$, i.e. $n = 16$, on the BN-254 elliptic curve, and for small and large input data.

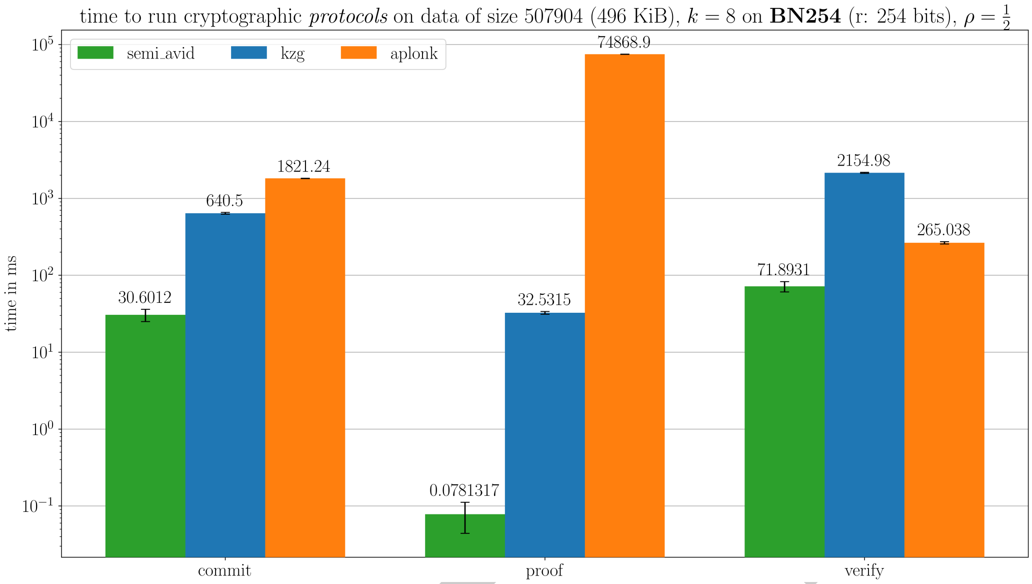

Figure 1: Performance for small files. Average over 10 runs.

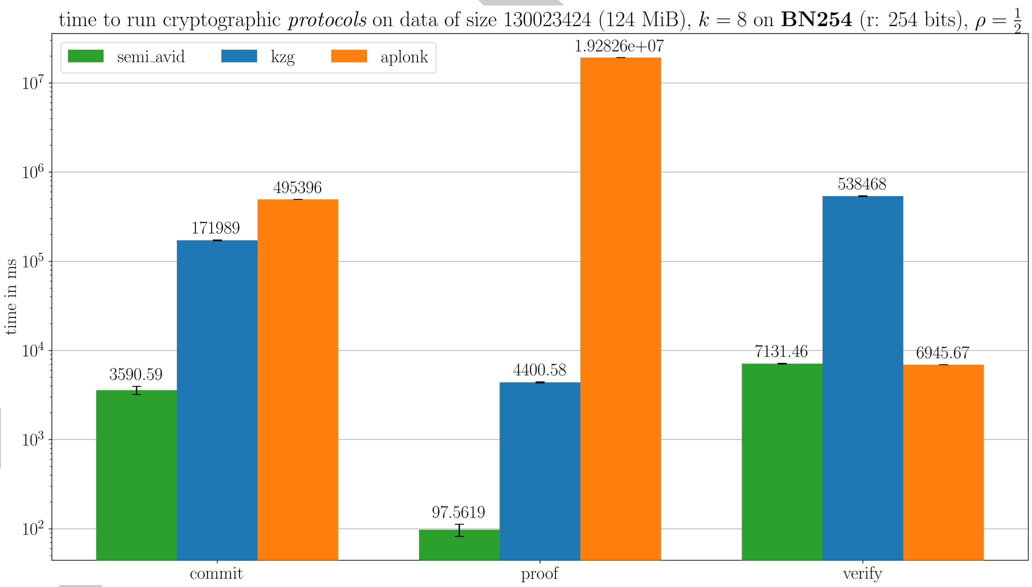

Figure 2: Performance for large files. Average over 10 runs for **KZG+** and **Semi-AVID**. Only 1 run for **aPlonK**.

Figure 1 shows that **Semi-AVID** is the best for committing, proving and verifying small files.

Figure 2 shows that **aPlonK** is slightly better for verifying large files but still suffers from performance orders of magnitude worst than **Semi-AVID** for committing and proving.

KZG+ is neither good nor too bad.

Additional information

Komodo is fully written in *Rust* and thus all dependencies are taken care of by *Cargo* and `Cargo.toml`.

Contact

- by email: `firstname.lastname@isae-supaero.fr`
- ticket tracker: https://github.com/dragoon-rs/komodo/issues
- contributions: https://github.com/dragoon-rs/komodo/pulls

Acknowledgements

This work was supported by the Defense Innovation Agency (AID) of the French Ministry of Defense through the Research Project DRAGOON: Dependable distRibuted storAGe fOr mObile Nodes (2022 65 0082).

References

Ambrona, M., Beunardeau, M., Schmitt, A.-L., & Toledo, R. R. (2023). *aPlonK: Aggregated PlonK from multi-polynomial commitment schemes*. 195–213. https://doi.org/10.1007/978-3-031-41326-1_11

Boneh, D., Drake, J., Fisch, B., & Gabizon, A. (2020). Efficient polynomial commitment schemes for multiple points and polynomials. *Cryptology ePrint Archive*.

contributors, arkworks. (2022). *arkworks zkSNARK ecosystem*. https://arkworks.rs

Gabizon, A., Williamson, Z. J., & Ciobotaru, O. (2019). Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. *Cryptology ePrint Archive*.

Kate, A., Zaverucha, G. M., & Goldberg, I. (2010). Constant-size commitments to polynomials and their applications. *International Conference on the Theory and Application of Cryptology and Information Security*, 177–194. https://doi.org/10.1007/978-3-642-17373-8_11

Nazirkhanova, K., Neu, J., & Tse, D. (2022). Information dispersal with provable retrievability for rollups. *Proceedings of the 4th ACM Conference on Advances in Financial Technologies*, 180–197. https://doi.org/10.1145/3558535.3559778

rust-rse contributors. (2021). Rust implementation of reed-solomon erasure coding. In *GitHub repository*. GitHub. https://github.com/rust-rse/reed-solomon-erasure

Stevan, Antoine, Lavaur, T., Lacan, J., Detchart, J., & Pérennou, T. (2023). Assessing the efficiency of polynomial commitment schemes in erasure code-based data distribution. *International Conference on Information Systems Security and Privacy*, 274–300. https://doi.org/10.1007/978-3-031-89518-0_13

Stevan, A., Lavaur, T., Lacan, J., Detchart, J., & Pérennou, T. (2024). Performance evaluation of polynomial commitments for erasure code based information dispersal. *10th International Conference on Information Systems Security and Privacy*. https://doi.org/10.5220/0012377900003648

What is proto-danksharding? (2024). https://ethereum.org/roadmap/danksharding/#what-is-protodanksharding