Efficient and Secure Compression Functions for Arithmetization-Oriented Hashing

CSF 2024, July 8–13

E. Andreeva[1] R. Bhattacharyya[2] A. Roy[3] **S. Trevisani**[1]

[1]TU Wien, [2]University of Birmingham, [3]University of Innsbruck

Verifiable Computation, Blockchains, and ZK-SNARKs

Verifiable Computation and ZK-SNARKs

Verifiable Computation for Trusted Cloud/P2P:

- Server: computes some function F, makes use of secret data.
- Clients: verify the correctness of the results.
- Use ZK-SNARKs:
  - Server $\iff$ Prover, Clients $\iff$ Verifiers
- Virtual Machines, Blockchains, Recursive SNARKs...

Hash functions and ZK-SNARKs

Hash functions play a central role:

- Blockchain roll-ups involve Merkle Tree (MT) hashing...
- ...And so does verification of recursive proofs.
- MT as commitment scheme $\Rightarrow$ **opening proof**.

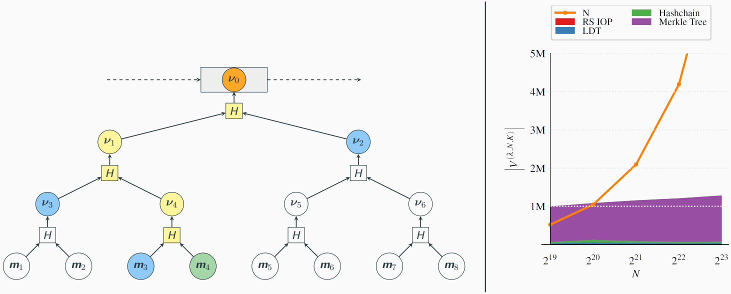


Figure 1: Left: binary Merkle Tree. Right: Fractal [6] verifier.

Arithmetization-Oriented Hash Functions

Computation complexity of a ZK-SNARK protocol:

- Proof Verification is fast (often constant time).
- Generation depends on the hash **multiplicative complexity**:
  - ◇ arithmetic circuit over a large (64/256-bits) prime field $\mathbb{F}_p$.
- Bit-oriented hash functions have high mult. complexity.
  - ◇ Bitwise operations in terms of field addition/multiplication.
- Arithmetization-oriented hash functions: defined over $\mathbb{F}_p$.
  - ◇ We will consider Poseidon [8] as an example.

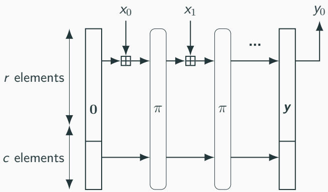
Primitive	Underlying Field	Native evaluation time	Proof generation time
SHA-256	$\mathbb{F}_2$	≈ 1	≈ 1000
Poseidon	$\mathbb{F}_p$	≈ 10	≈ 1

The PGV-LC and PGV-ELC Modes of Compression

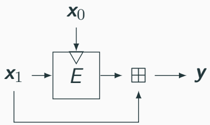
Blockcipher/Permutation-based hash functions

Compositional paradigms to obtain provable security guarantees:

- Permutation-based, like Sponge, used in SHA-3, POSEIDON.
    - ◇ Permutation is often a fixed-key blockcipher.
    - ◇ Provably secure over $\mathbb{F}_p$ (SAFE [12]).
    - ◇ Cannot use the key input to compress data.
- Blockcipher-based, like Davies-Meyer, used in SHA-2:
    - ◇ Exploit both key and plaintext inputs for compression.
    - ◇ Provably secure over $\mathbb{F}_2$, (PGV [14, 4]).



Sponge



Davies-Meyer

The PGV-LC mode

Inspired by the PGV modes, we introduce the PGV-LC mode:

- Underlying Blockcipher $E\colon \mathbb{F}_p^\kappa \times \mathbb{F}_p^n \to \mathbb{F}_p^n$.
- Matrix $\boldsymbol{R} \in \mathbb{F}_p^{\ell \times n}$ parametrizes output size.
  - ◇ Compresses its input $\Rightarrow \ell \leq n$.
  - ◇ Algebraic generalization of e.g. truncation and chopping.

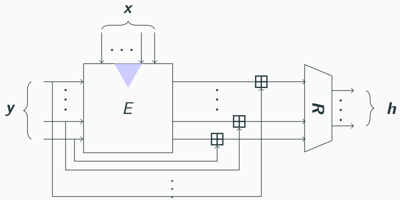


Figure 2: A depiction of the PGV-LC mode: $\boldsymbol{x} \in \mathbb{F}_p^\kappa$, $\boldsymbol{y} \in \mathbb{F}_p^n$, $\boldsymbol{h} \in \mathbb{F}_p^\ell$.

The PGV-ELC mode

We further generalize the design with the PGV-ELC mode:

- Matrices $\boldsymbol{K} \in \mathbb{F}_p^{\kappa \times \kappa'}$ and $\boldsymbol{P} \in \mathbb{F}_p^{n \times n'}$ parametrize input size.
  - ◇ Expand their inputs $\Rightarrow \kappa' \leq \kappa$ and $n' \leq n$.
  - ◇ Algebraic generalization of e.g. zero-padding.
- Matrix $\boldsymbol{F} \in \mathbb{F}_p^{\ell \times n'}$ adapts input to output size.
  - ◇ Expands its input $\Rightarrow \ell \leq n'$.

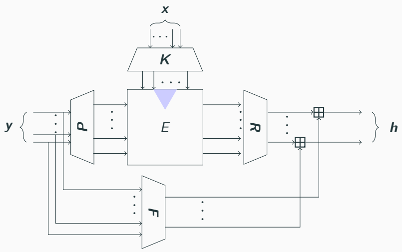


Figure 3: Visualization of PGV-ELC: $\boldsymbol{x} \in \mathbb{F}_p^{\kappa'}$, $\boldsymbol{y} \in \mathbb{F}_p^{n'}$, $\boldsymbol{h} \in \mathbb{F}_p^{\ell}$

How to Prove Your Security

In order to prove that our modes are secure, we need:

- A formal model: the ideal cipher $E \xleftarrow{\$} \mathsf{Block}\big(\mathbb{F}_p^\kappa, \mathbb{F}_p^n\big)$.
    - Standard security assumption in classic cryptography.
    - $\approx$ For permutations, ideal permutation $\pi \xleftarrow{\$} \mathsf{Perm}\big(\mathbb{F}_p^n\big)$.
- An adversary $\mathcal{A}$:
    - Unbounded randomized algorithm.
    - Can query E forward and backward via the oracle $\mathcal{E}$.
- A security notion (e.g. collision resistance).
- An advantage function $\mathbf{Adv}_{\mathrm{scheme}}^{\mathrm{NOTION}}(\mathcal{A}, q)$:
    - Must be negligible in the number q of oracle queries.
    - $\mathbf{Adv}_{\mathrm{scheme}}^{\mathrm{NOTION}}(q) = \max_{\mathcal{A}}\big\{\mathbf{Adv}_{\mathrm{scheme}}^{\mathrm{NOTION}}(\mathcal{A}, q)\big\}$

PGV-LC Security

Collision resistance:

$$\mathbf{Adv}_C^{\mathrm{COL}}(\mathcal{A}, q) = \Pr\Big[\big(\boldsymbol{x}, \boldsymbol{x}'\big) \xleftarrow{\$} \mathcal{A}^{\mathcal{E}}() : \boldsymbol{x} \neq \boldsymbol{x}' \wedge C_E(\boldsymbol{x}) = C_E\big(\boldsymbol{x}'\big)\Big]$$

For PGV-LC $C_E(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{R} \cdot (E_{\boldsymbol{y}}(\boldsymbol{x}) + \boldsymbol{x})$:

1. Consider $\boldsymbol{R}$ right-invertible (full row rank).
2. $\mathbb{F}_p^n$ is partitioned into p^ℓ equivalence classes.
3. $\mathcal{A}$ can exploit partition unbalances from oracle replies.
4. Still, $\mathbf{Adv}_C^{\mathrm{COL}}(q) \leq \frac{q^2+q}{p^\ell - q}$ (≈ birthday attack).
5. Similarly, for preimage resistance: $\mathbf{Adv}_C^{\mathrm{PRE}}(q) \leq \frac{q}{p^\ell - q}$.

PGV-ELC Security

Collision resistance:

$$\mathbf{Adv}_C^{\mathrm{COL}}(\mathcal{A}, q) = \Pr\Big[\big(\boldsymbol{x}, \boldsymbol{x}'\big) \xleftarrow{\$} \mathcal{A}^{\mathcal{E}}() : \boldsymbol{x} \neq \boldsymbol{x}' \wedge C_E(\boldsymbol{x}) = C_E\big(\boldsymbol{x}'\big)\Big]$$

For PGV-ELC $C_E(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{R} \cdot E_{\boldsymbol{Ky}}(\boldsymbol{Px}) + \boldsymbol{Fx}$:

1. Consider $\boldsymbol{K}$ and $\boldsymbol{P}$ left-invertible, $\boldsymbol{F}$ right-invertible.
2. Linear transformations induce partitions.
3. 'Meaningless' queries, cannot be used to form a collision:
    - However, can be exploited to guide further queries.
4. Nevertheless, we again obtain $\mathbf{Adv}_C^{\mathrm{COL}}(q) \leq \frac{q^2+q}{p^\ell - q}$.
5. Similarly, for preimage resistance: $\mathbf{Adv}_C^{\mathrm{PRE}}(q) \leq \frac{q}{p^\ell - q}$.

Merkle Tree Opening Security

Security notion for openings over a t-ary Merkle Tree:

- Merkle Tree intended as a hash function H.
- Generator $\mathcal{G}$ creates an opening π.
- Verifier $\mathcal{V}$ checks validity of π.
- Adversary $\mathcal{A}$ attempts to forge $\tilde{\pi}$.

Formally:

$$\mathbf{Adv}^{\mathrm{OPEN}}_{H,\mathcal{G},\mathcal{V}}(\mathcal{A}, q) =$$
$$\Pr\Big[M \xleftarrow{\$} \left(\mathbb{F}_p^m\right)^*, \tilde{\pi} \xleftarrow{\$} \mathcal{A}^{\mathcal{E}}(M) : \forall i \in \mathbb{N}\colon \tilde{\pi} \neq \mathcal{G}(M, i) \wedge \mathcal{V}(\tilde{\pi}, H_C(M)) = \top\Big]$$

Merkle Tree Opening Security (cont.)

For a t-ary Merkle Tree:

- $\mathbf{Adv}^{\mathrm{OPEN}}_{H,\mathcal{G},\mathcal{V}}(q) \leq \mathbf{Adv}^{\mathrm{COL}}_{C}(q)$
- Additionally, $\mathbf{Adv}^{\mathrm{COL}}_{H}(q) \leq \mathbf{Adv}^{\mathrm{COL}}_{C}(q) + \mathbf{Adv}^{\mathrm{PRE}}_{C}(q)$.
- Proof is standard, generalizes reasoning for binary trees.

$\Longrightarrow$ Our modes can be securely used for MT commitments.

Implementations and Experiments

Consider the POSEIDON hash function:

- Sponge mode over the fixed-key HADES block cipher.
- Affine key scheduler, which we instantiated with:

$$\boldsymbol{M}_{\mathcal{K},2} = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} \qquad \boldsymbol{M}_{\mathcal{K},4} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 3 & 1 & 2 & 1 \\ 1 & 1 & 1 & 2 \end{bmatrix}$$

- Instantied in PGV-LC: POSEIDON-DM.
- All other parameters kept the same as in POSEIDON.
  - ◇ Focus on improvement due to compression mode.

R1CS **arithmetization**

We considred R1CS arithmetization:

- Used by many ZK-SNARKs (Groth16, Aurora, Spartan, ...):
  - R1CS System: $\boldsymbol{Ax} \odot \boldsymbol{Bx} = \boldsymbol{Cx}$
- Concrete performance tends to follow theoretical numbers.

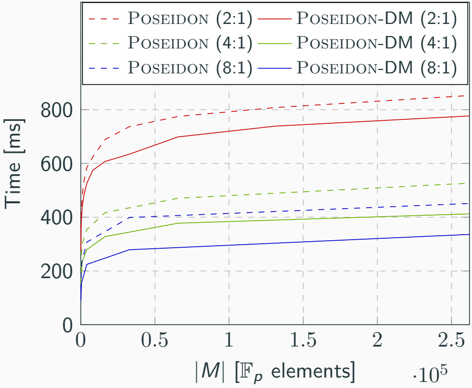
Hash	Compression Rate 2:1	4:1	8:1
POSEIDON	237	288	384
POSEIDON-DM	213	213	261
Constraint Reduction			
POSEIDON-DM w.r.t. POSEIDON	−11%	−35%	−47%

Table 1: Number of R1CS constraints for target primitives.

Benchmarks: Proof Generation

Time to generate a MT opening proof:

- Scalar field of the BLS12-381 elliptic curve: $\log_2(p) \approx 255$.
- ZK-SNARK framework: Groth16 (`libsnark`).



Benchmarks: Native execution

Native evaluation time speedup :

- Averaged over the scalar field of various curves:
  - BLS12-381, BN254, Ed-180.

Library	2:1	4:1	8:1
`NTL`	$1.17\times$	$2.80\times$	$2.51\times$
`libff`	$1.17\times$	$2.87\times$	$2.57\times$
`libarith`	$1.15\times$	$2.27\times$	$2.27\times$

Table 2: Poseidon-DM speed-up for a single compression call.

Benchmarks: Arity Matters

Choosing an optimal arity of the Merkle Tree matters:

- Binary trees are the most common choice.
- For generating an opening proof:
  - $8{:}1$ Poseidon-DM $\approx 2.5\times$ faster than $2{:}1$ Poseidon.
- For building the tree:
  - $4{:}1$ Poseidon-DM $\approx 4\times$ faster than $2{:}1$ Poseidon.
- 💡 Improve existing *t*-ary Merkle Tree opening proof circuits:
  - $\approx 10\%$ improvement to known strategies.

Roll credits

$\mathcal{T}he\ \mathcal{E}nd$

Thank you for your attention!

Martin Albrecht, Lorenzo Grassi, Christian Rechberger, Arnab Roy, and Tyge Tiessen.
Mimc: Efficient encryption and cryptographic hashing with minimal multiplicative complexity.
In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016*, pages 191–219, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

Elena Andreeva, Rishiraj Bhattacharyya, Arnab Roy, and Stefano Trevisani.
On efficient and secure compression modes for arithmetization-oriented hashing.
Cryptology ePrint Archive, Paper 2024/047, 2024.

https://eprint.iacr.org/2024/047.

Amit Singh Bhati, Erik Pohle, Aysajan Abidin, Elena Andreeva, and Bart Preneel.
Let's go eevee! a friendly and suitable family of aead modes for iot-to-cloud secure computation.
In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, CCS '23, pages 2546–2560, New York, NY, USA, 2023. Association for Computing Machinery.

John Black, Phillip Rogaway, and Thomas Shrimpton.
Black-box analysis of the block-cipher-based hash-function constructions from pgv.
Cryptology ePrint Archive, Paper 2002/066, 2002.
https://eprint.iacr.org/2002/066.

Joppe W. Bos and Peter L. Montgomery.
Montgomery arithmetic from a software perspective.
Cryptology ePrint Archive, Paper 2017/1057, 2017.
https://eprint.iacr.org/2017/1057.

Alessandro Chiesa, Dev Ojha, and Nicholas Spooner.
Fractal: Post-quantum and transparent recursive proofs from holography.
Cryptology ePrint Archive, Paper 2019/1076, 2019.
https://eprint.iacr.org/2019/1076.

Shafi Goldwasser, Silvio Micali, and Charles Rackoff.
The knowledge complexity of interactive proof systems.
SIAM Journal on Computing, 18(1):186–208, 1989.

Bibliography v

Lorenzo Grassi, Dmitry Khovratovich, Christian Rechberger, Arnab Roy, and Markus Schofnegger.
Poseidon: A new hash function for zero-knowledge proof systems.
Cryptology ePrint Archive, Paper 2019/458, 2019.
https://eprint.iacr.org/2019/458.

Lorenzo Grassi, Dmitry Khovratovich, and Markus Schofnegger.
Poseidon2: A faster version of the poseidon hash function.
Cryptology ePrint Archive, Paper 2023/323, 2023.
https://eprint.iacr.org/2023/323.

Lorenzo Grassi, Reinhard Lüftenegger, Christian Rechberger, Dragos Rotaru, and Markus Schofnegger.
On a generalization of substitution-permutation networks: The hades design strategy.
Cryptology ePrint Archive, Paper 2019/1107, 2019.
https://eprint.iacr.org/2019/1107.

Jens Groth.
On the size of pairing-based non-interactive arguments.
Cryptology ePrint Archive, Paper 2016/260, 2016.
https://eprint.iacr.org/2016/260.

Dmitry Khovratovich, Mario Marhuenda Beltrán, and Bart Mennink.
Generic security of the safe api and its applications.
In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology – ASIACRYPT 2023*, pages 301–327, Singapore, 2023. Springer Nature Singapore.

Ralph Charles Merkle.
Secrecy, Authentication, and Public Key Systems.
PhD thesis, Stanford University, Stanford, CA, USA, 1979.
AAI8001972.

Bart Preneel, René Govaerts, and Joos Vandewalle.
Hash functions based on block ciphers: A synthetic approach.
In *Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22-26, 1993, Proceedings*, volume 773 of *Lecture Notes in Computer Science*, pages 368–378. Springer, 1993.