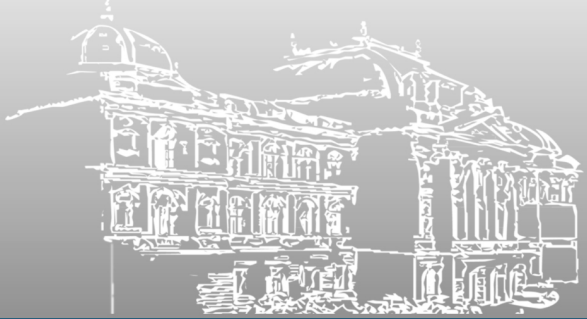


Skyscraper: Fast Hashing on Big Primes

Clémence Bouvier Lorenzo Grassi Dmitry Khovratovich Katharina Koschatko Christian Rechberger **Fabian Schmid** Markus Schofnegger

Motivation

- Why a new hash function?
  - Cost of proving related to algebraic description
  - Traditional hashes have complex representations
  - Tailored designs improve many applications
- Focus on large primes
  - Used in pairing-based ZK protocols
  - Small proofs, fast verification
  - Efficiently used *on-chain*

Motivation

- Why a new hash function?
  - Cost of proving related to algebraic description
  - Traditional hashes have complex representations
  - Tailored designs improve many applications
- Focus on large primes
  - Used in pairing-based ZK protocols
  - Small proofs, fast verification
  - Efficiently used *on-chain*

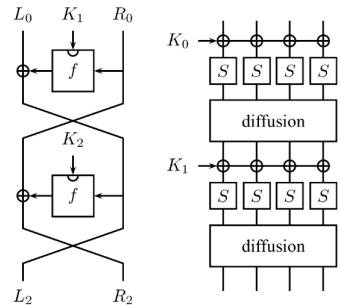
Design Strategy

What do we want:

- Improve performance for large prime fields.
- Make design as simple as possible to minimize attack vectors.
- Optimize chosen power map.
- Use lookups to increase the algebraic degree.

How we achieve it:

- Feistel structure instead of SPN.



Feistel cipher versus SP network [CBP06, Fig. 4]

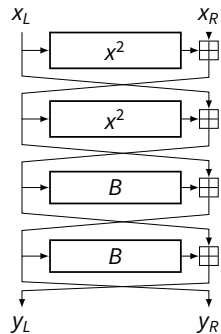
Design Strategy

What do we want:

- Improve performance for large prime fields.
- Make design as simple as possible to minimize attack vectors.
- Optimize chosen power map.
- Use lookups to increase the algebraic degree.

How we achieve it:

- Non-invertible Squarings and S-Boxes.



Design Strategy

What do we want:

- Improve performance for large prime fields.
- Make design as simple as possible to minimize attack vectors.
- Optimize chosen power map.
- Use lookups to increase the algebraic degree.

How we achieve it:

- Account for Montgomery reduction.

Algorithm 1: Montgomery Reduction

Input: $c \in \{0, \ldots, p\sigma - 1\}$

$$\tilde{c} \leftarrow \left(c \cdot \underbrace{\left((-p)^{-1} \bmod \sigma\right)}_{\text{precomputed}}\right) \bmod \sigma$$

$$r \leftarrow \frac{c + \tilde{c}\,p}{\sigma}$$

$$r \leftarrow \begin{cases} r - p & \text{if } r \geq p, \\ r & \text{else.} \end{cases}$$

Return $r = c\,\sigma^{-1} \bmod p$

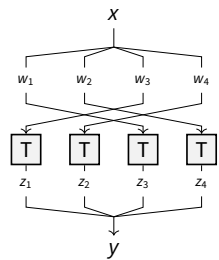
Design Strategy

What do we want:

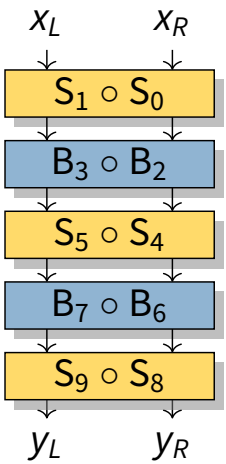
- Improve performance for large prime fields.
- Make design as simple as possible to minimize attack vectors.
- Optimize chosen power map.
- Use lookups to increase the algebraic degree.

How we achieve it:

- Use simple 8-bit lookup tables. (cf. `Monolith` [GKL+24])
- S-box combined with circular shift.

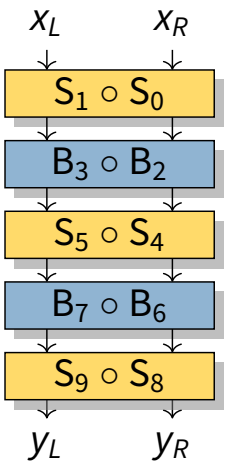


Skyscraper Design: Overview

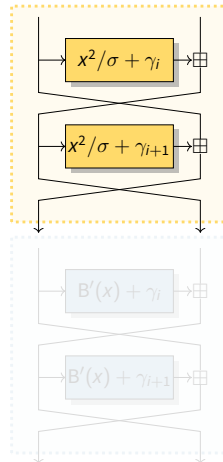


- Square operation S_i
  - Non-invertible x^2
  - Good statistical properties
  - Speed-up via Montgomery
- Bars operation B_i
  - Non-invertible S-Box B'
  - Applicable to any prime
  - High algebraic degree
  - Speed-up via efficient bit operations

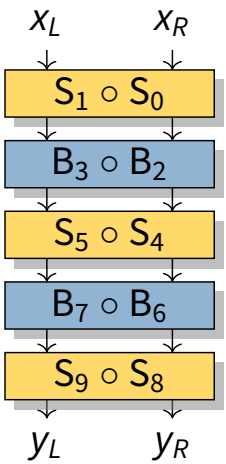
Skyscraper Design: Overview



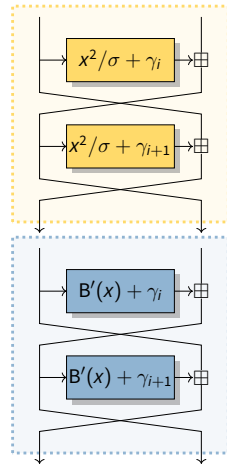
- Square operation S_i
  - Non-invertible x^2
  - Good statistical properties
  - Speed-up via Montgomery
- Bars operation B_i
  - Non-invertible S-Box B′
  - Applicable to any prime
  - High algebraic degree
  - Speed-up via efficient bit operations



Skyscraper Design: Overview



- Square operation S_i
  - Non-invertible x^2
  - Good statistical properties
  - Speed-up via Montgomery
- Bars operation B_i
  - Non-invertible S-Box B'
  - Applicable to any prime
  - High algebraic degree
  - Speed-up via efficient bit operations



Skyscraper Design: Wide-State Dilemma

Question: Feistel takes only two inputs. How to "increase statesize"?

Skyscraper Design: Wide-State Dilemma

Question: Feistel takes only two inputs. How to "increase statesize"?

Switch to extension field $\mathbb{F}_{p^n}$

Skyscraper Design: Wide-State Dilemma

Question: Feistel takes only two inputs. How to "increase statesize"?

Switch to extension field $\mathbb{F}_{p^n}$

- $\mathbb{F}_{p^n} \equiv \mathbb{Z}_p[X]/\mathsf{G}(x)$, where $deg(\mathsf{G}) = n$
- Square: Evaluate in $Z_p[X]$ and reduce by G
- Bar: Interpret $x \in \mathbb{F}_{p^n}$ as vector $\mathbb{F}_p^n$

$$x_0 + x_1 \cdot X + \cdots + x_{n-1} \cdot X^{n-1} \equiv (x_0, x_1, \ldots, x_{n-1})$$

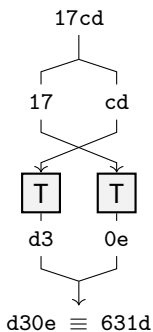
	Bar	Sq
$\mathbb{F}_p$	8	14
$\mathbb{F}_{p^2}$	12	52
$\mathbb{F}_{p^3}$	20	128

Table: Runtime [ns]

Skyscraper Design: S-Box component B′

Examples: $B' : \mathbb{F}_{p^n} \to \mathbb{F}_{p^n}$ for $p = 28657$ (15-bit prime)

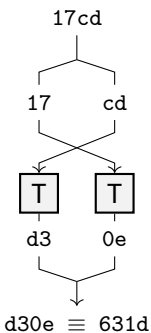
$n = 1$



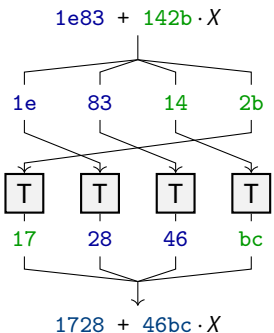
Skyscraper Design: S-Box component B′

Examples: $B' : \mathbb{F}_{p^n} \to \mathbb{F}_{p^n}$ for $p = 28657$ (15-bit prime)

$n = 1$



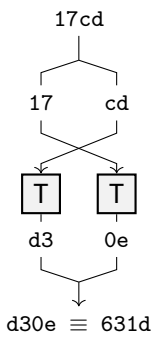
$n = 2$



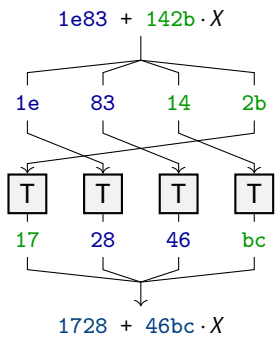
Skyscraper Design: S-Box component B′

Examples: $B' : \mathbb{F}_{p^n} \to \mathbb{F}_{p^n}$ for $p = 28657$ (15-bit prime)

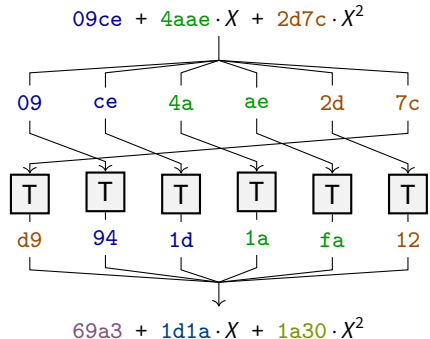
$n = 1$



$n = 2$

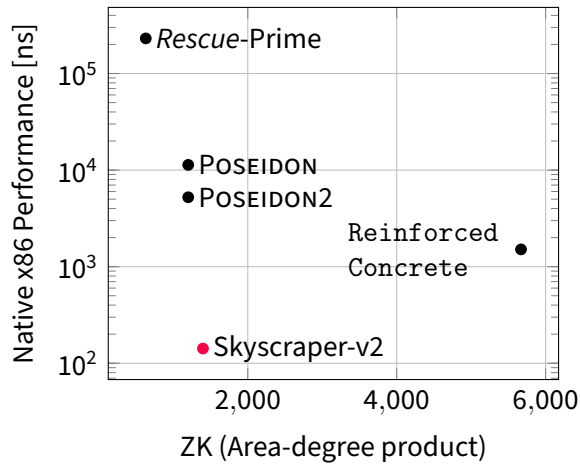


$n = 3$



Performance Comparison for BN254

Hash Function	x86	ZK
Skyscraper	142	1 398
RC	1 510	5 670
POSEIDON	11 324	1 200
POSEIDON2	5 233	1 200
Rescue-Prime	230 950	630

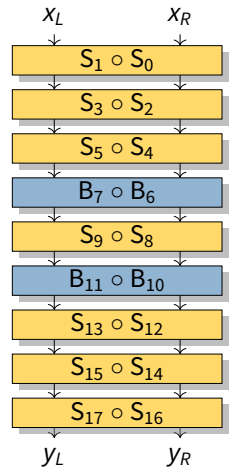


Area-degree product = size of witness matrix $\times$ max. degree of polynomial that encodes a gate

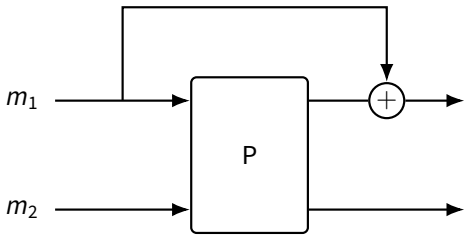
Security Issues and Update

What happened:

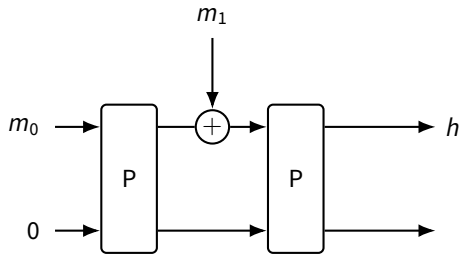
- Rebound attack by Antoine Bak [Bak25]
- Distinguisher on 9-round version
- No security margin
- Release of Skyscraper-v2



Recall: Mode of Operation



Compression Mode



Sponge Mode

Native performance for hashing over $\mathbb{F}_p$

Hash Function		Message 508 bit [ns]	Message 1 Mbit [μs]
Rescue-Prime		235 630	495 460
POSEIDON		8 276	26 375
POSEIDON2		4 907	13 043
`Reinforced Concrete`		1 511	3 411
Skyscraper	$\mathbb{F}_p$	135	711
	$\mathbb{F}_{p^2}$	320	657

Table: Native performance for hashing with $\log_2 p \approx 254$.

Native performance for hashing over $\mathbb{F}_p$

Hash Function		Message 508 bit [ns]	Message 1 Mbit [μs]
Rescue-Prime		235 630	495 460
POSEIDON		8 276	26 375
POSEIDON2		4 907	13 043
`Reinforced Concrete`		1 511	3 411
Skyscraper(v2)	$\mathbb{F}_p$	135 *(256)*	711 *(1482)*
	$\mathbb{F}_{p^2}$	320 *(731)*	657 *(1423)*

Table: Native performance for hashing with $\log_2 p \approx 254$.

Conclusion

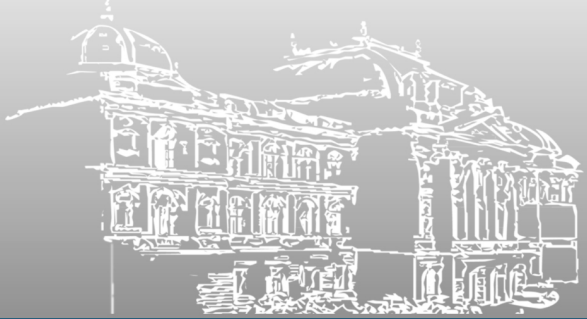
- New hash function Skyscraper-v2
  - Efficient in plain and in proof systems for large primes
  - Plain performance comparable to SHA-3
- Feistel design strategy
  - Allows for non-invertible components
  - Remove affine layer (costly for large state or large round number)
  - Simple design (no growing state) due to extension field usage
- Generic Bar construction
  - Not fixed to specific prime
  - High algebraic degree

Conclusion

- New hash function Skyscraper-v2
  - Efficient in plain and in proof systems for large primes
  - Plain performance comparable to SHA-3
- Feistel design strategy
  - Allows for non-invertible components
  - Remove affine layer (costly for large state or large round number)
  - Simple design (no growing state) due to extension field usage
- Generic Bar construction
  - Not fixed to specific prime
  - High algebraic degree

Conclusion

- New hash function Skyscraper-v2
  - Efficient in plain and in proof systems for large primes
  - Plain performance comparable to SHA-3
- Feistel design strategy
  - Allows for non-invertible components
  - Remove affine layer (costly for large state or large round number)
  - Simple design (no growing state) due to extension field usage
- Generic Bar construction
  - Not fixed to specific prime
  - High algebraic degree



Skyscraper: Fast Hashing on Big Primes

Clémence Bouvier Lorenzo Grassi Dmitry Khovratovich Katharina Koschatko Christian Rechberger **Fabian Schmid** Markus Schofnegger

Bibliography I

[AAE+20] A. Aly, T. Ashur, Eli Ben-Sasson, S. Dhooghe, and A. Szepieniec. **Design of Symmetric-Key Primitives for Advanced Cryptographic Protocols.** *IACR Trans. Symmetric Cryptol.* 2020.3 (2020), pp. 1–45.

[Bak25] A. Bak. **A practical distinguisher on the full Skyscraper permutation.** Cryptology ePrint Archive, Paper 2025/102. 2025 (cit. on p. 18).

[CBP06] C. Cannière, A. Biryukov, and B. Preneel. **An introduction to Block Cipher Cryptanalysis.** *Proceedings of the IEEE* 94 (Mar. 2006), pp. 346–356. DOI: 10.1109/JPROC.2005.862300 (cit. on p. 4).

[GKL+22] L. Grassi, D. Khovratovich, R. Lüftenegger, C. Rechberger, M. Schofnegger, and R. Walch. **Reinforced Concrete: A Fast Hash Function for Verifiable Computation.** CCS 2022. ACM, 2022, pp. 1323–1335. DOI: 10.1145/3548606.3560686.

Bibliography II

[GKL+24] L. Grassi, D. Khovratovich, R. Lüftenegger, C. Rechberger, M. Schofnegger, and R. Walch. **Monolith: Circuit-Friendly Hash Functions with New Nonlinear Layers for Fast and Constant-Time Implementations.** *IACR Trans. Symmetric Cryptol.* 2024.3 (2024), pp. 44–83. DOI: 10.46586/TOSC.V2024.I3.44-83 (cit. on pp. 7, 28).

[GKR+21] L. Grassi, D. Khovratovich, C. Rechberger, A. Roy, and M. Schofnegger. **Poseidon: A New Hash Function for Zero-Knowledge Proof Systems.** USENIX 2021. USENIX Association, 2021, pp. 519–535.

[GKS23] L. Grassi, D. Khovratovich, and M. Schofnegger. **Poseidon2: A Faster Version of the Poseidon Hash Function.** AFRICACRYPT 2023. Vol. 14064. LNCS. Springer, 2023, pp. 177–203. DOI: 10.1007/978-3-031-37679-5_8.

[SAD20] A. Szepieniec, T. Ashur, and S. Dhooghe. **Rescue-Prime: a Standard Specification (SoK).** Cryptology ePrint Archive, Report 2020/1143. https://ia.cr/2020/1143. 2020.

Plain Performance

		BN254		Goldilocks [GKL+24]		
Hashing algorithm	(t,deg.)	Rounds	Time [ns]	(t,deg.)	Rounds	Time [ns]
Poseidon	(3,5)	64 (4+56+4)	11 324	(8,7)	30 (4+22+4)	1 898
Poseidon2	(3,5)	64 (4+56+4)	5 233	(8,7)	30 (4+22+4)	1 292
Rescue-Prime	(3,5)	14	230 950	(8,7)	8	12 128
`Reinforced Concrete`	(3,5)	7 (3+1+3)	1 510			
Skyscraper-v2	(2,2)	18 (14 Sq., 4 B)	256			
`Monolith`				(8,2)	6	130

Table: Native performance of compressing 512 bits (2-1 compression) over $\mathbb{F}_p^t$, where t denotes the statesize. For Poseidon,Poseidon2, *Rescue*-Prime, and `Reinforced Concrete`, "deg." denotes the non-linear permutation degree, that is, the smallest positive integer d such that $\gcd(d, p-1) = 1$ (e.g., $d = 5$ for BN254 and $d = 7$ for Goldilocks). In `Monolith` and Skyscraper-v2, (non-invertible) squaring is employed as part of a Feistel construction.

Plonkish Performance

Component	Witnesses	Constraints (deg.)	Lookups	Area-degree product
		$n = 1$		
Double squaring	3	3 (2)	0	–
All squarings	9	9 (2)	0	–
`Bars`	56	10 (1)	32	–
All `Bars`	224	40 (1)	128	–
Skyscraper-v2	233	49 (1, 2)	128	1398
`Reinforced Concrete`	378	24 (1, 3, d)	267	5670
POSEIDON, POSEIDON2	80	80 (d)	0	1200
Rescue-Prime	42	42 (d)	0	630

Table: Circuit performance of all components of Skyscraper-v2, where we assume the BN254 setting. This also includes a comparison with `Reinforced Concrete`, POSEIDON, POSEIDON2, and *Rescue*-Prime, where d is the smallest positive integer such that $\gcd(d, p - 1) = 1$ (e.g., $d = 5$ for BN254).