Horst Meets *Fluid*-SPN: Griffin for Zero-Knowledge Applications

Lorenzo Grassi, Yonglin Hao, Christian Rechberger, Markus Schofnegger, Roman Walch, Qingju Wang

Santa Barbara, August 2023

Computational Integrity Proof Systems

- Prove that something has been computed correctly
  - Can be any program in theory
  - Permutation call, Merkle tree, ...
- Verification cost sublinear in program size
- Potentially also with zero knowledge
- SNARKs/STARKs
- Classical primitives (AES, Keccak, ...) often inefficient in this setting
  - Use more specialized ones

Computational Integrity Proof Systems

- Prove that something has been computed correctly
  - Can be any program in theory
  - Permutation call, Merkle tree, . . .
- Verification cost sublinear in program size
- Potentially also with zero knowledge
- SNARKs/STARKs
- Classical primitives (AES, KECCAK, . . .) often inefficient in this setting
  - Use more specialized ones

Computational Integrity Proof Systems

HORIZEN LABS

- Prove that something has been computed correctly
  - Can be any program in theory
  - Permutation call, Merkle tree, ...
- Verification cost sublinear in program size
- Potentially also with zero knowledge
- SNARKs/STARKs
- Classical primitives (AES, KECCAK, ...) often inefficient in this setting
  - Use more specialized ones

Computational Integrity Proof Systems

- Prove that something has been computed correctly
  - Can be any program in theory
  - Permutation call, Merkle tree, ...
- Verification cost sublinear in program size
- Potentially also with zero knowledge
- SNARKs/STARKs
- Classical primitives (AES, KECCAK, ...) often inefficient in this setting
  - Use more specialized ones

Computational Integrity Proof Systems

- Prove that something has been computed correctly
  - Can be any program in theory
  - Permutation call, Merkle tree, ...
- Verification cost sublinear in program size
- Potentially also with zero knowledge
- SNARKs/STARKs
- Classical primitives (AES, KECCAK, ...) often inefficient in this setting
  - Use more specialized ones

Arithmetization and Commitment

- Proofs are split into two steps
  - Arithmetization → convert program into polynomials
  - Polynomial commitment → prove validity of polynomials
- Mostly, any arithmetization approach can be combined with any commitment technique
- Focus on arithmetization of hash function
  - Includes set of constraints
  - Fewer constraints in general better

Arithmetization and Commitment

- Proofs are split into two steps
  - Arithmetization → convert program into polynomials
  - Polynomial commitment → prove validity of polynomials
- Mostly, any arithmetization approach can be combined with any commitment technique
- Focus on arithmetization of hash function
  - Includes set of constraints
  - Fewer constraints in general better

Symmetric Function Concepts

HORIZEN LABS

Type 1	Type 2	Type 3
"*low-degree only*"	"*non-procedural*", "*fluid*"	"lookups"
■ Low-degree	■ Equivalent low-degree	■ Lookup tables
$y = x^3$	$y = x^{1/3} \implies x = y^3$	$y = T[x]$
■ Fast	■ Slow	■ Very fast
■ Many rounds	■ Fewer rounds	■ Even fewer rounds
■ Often more constraints	■ Fewer constraints	■ Constraints depend on scheme
■ POSEIDON, POSEIDON2, NEPTUNE, GMiMC	■ FRIDAY, *Rescue*, GRIFFIN, Anemoi	■ Reinforced Concrete, Tip5, Monolith

Symmetric Function Concepts

Type 1

"low-degree only"

- Low-degree

$$y = x^3$$

- Fast
- Many rounds
- Often more constraints
- POSEIDON, POSEIDON2, NEPTUNE, GMiMC

Type 2

"non-procedural", *"fluid"*

- Equivalent low-degree

$$y = x^{1/3} \implies x = y^3$$

- Slow
- Fewer rounds
- Fewer constraints
- FRIDAY, *Rescue*, GRIFFIN, Anemoi

Type 3

"lookups"

- Lookup tables

$$y = T[x]$$

- Very fast
- Even fewer rounds
- Constraints depend on scheme
- Reinforced Concrete, Tip5, Monolith

Symmetric Function Concepts

Type 1

"low-degree only"

- Low-degree

$$y = x^3$$

- Fast
- Many rounds
- Often more constraints
- Poseidon, Poseidon2, Neptune, GMiMC

Type 2

"non-procedural", *"fluid"*

- Equivalent low-degree

$$y = x^{1/3} \implies x = y^3$$

- Slow
- Fewer rounds
- Fewer constraints
- Friday, *Rescue*, Griffin, Anemoi

Type 3

"lookups"

- Lookup tables

$$y = T[x]$$

- Very fast
- Even fewer rounds
- Constraints depend on scheme
- Reinforced Concrete, Tip5, Monolith

Symmetric Function Concepts

Type 1

"low-degree only"

- Low-degree

$$y = x^3$$

- Fast
- Many rounds
- Often more constraints
- Poseidon, Poseidon2, Neptune, GMiMC

Type 2

"non-procedural", *"fluid"*

- Equivalent low-degree

$$y = x^{1/3} \implies x = y^3$$

- Slow
- Fewer rounds
- Fewer constraints
- Friday, *Rescue*, Griffin, Anemoi

Type 3

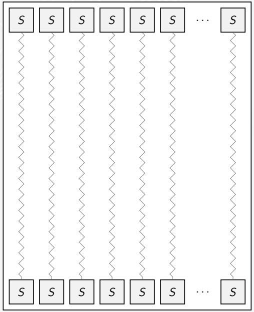
"lookups"

- Lookup tables

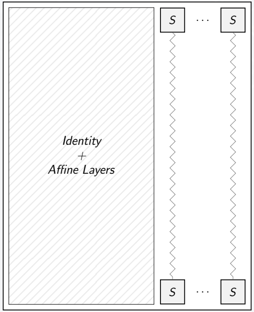
$$y = T[x]$$

- Very fast
- Even fewer rounds
- Constraints depend on scheme
- Reinforced Concrete, Tip5, Monolith

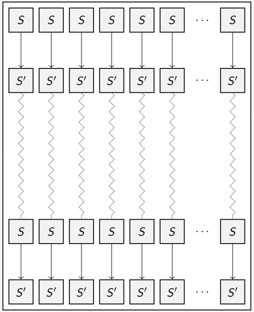
Constraints: The Nonlinear Layer



Classical SPN
(e.g., SHARK in 1996)



Partial SPN
(e.g., `Zorro` in 2013 and LowMC in 2015)



Different rounds/steps
(e.g., Rescue in 2019)

⟷ Constraints: The Nonlinear Layer cont.

- Focus on SPN instantiated with power maps
  - We need degree $\geqslant 3$ for the S-boxes (invertibility)
  - Most practically used primes even need degree $\in \{5, 7\}$
- For degree 7: $4t$ multiplications for t words
- Performance with low-degree functions?
  - Large number of rounds to reach maximum degree

  $\longrightarrow$ Many linear layers, high latency
- Mix rounds with x^d and rounds with $x^{1/d}$ like *Rescue*?
  - Multiple $x^{1/d}$ per round quite expensive ...

⟷ Constraints: The Nonlinear Layer cont.

- Focus on SPN instantiated with power maps
    - We need degree $\geqslant 3$ for the S-boxes (invertibility)
    - Most practically used primes even need degree $\in \{5, 7\}$
- For degree 7: $4t$ multiplications for t words
- Performance with low-degree functions?
    - Large number of rounds to reach maximum degree

    $\longrightarrow$ Many linear layers, high latency
- Mix rounds with x^d and rounds with $x^{1/d}$ like *Rescue*?
    - Multiple $x^{1/d}$ per round quite expensive . . .

Observations

- SPN
  - $x \mapsto x^d$ and $x \mapsto x^{1/d}$ can be included in a single round, e.g.

    $$x_0 \mapsto {x_0}^d, \quad x_1 \mapsto {x_1}^{1/d}$$

    for state with two elements
  - Only needed for 2 elements (instead of entire state)
- Feistel
  - Allows for lower degrees (e.g., non-invertible $x \mapsto x^2$)
  - Instead of addition in original Feistel, consider multiplication

  $\longrightarrow$ Nonlinear diffusion, better protection against attacks

Observations

- SPN
  - $x \mapsto x^d$ and $x \mapsto x^{1/d}$ can be included in a single round, e.g.

    $$x_0 \mapsto {x_0}^d, \quad x_1 \mapsto {x_1}^{1/d}$$

    for state with two elements
  - Only needed for 2 elements (instead of entire state)
- Feistel
  - Allows for lower degrees (e.g., non-invertible $x \mapsto x^2$)
  - Instead of addition in original Feistel, consider multiplication

  $\longrightarrow$ Nonlinear diffusion, better protection against attacks

Horst – Multiplicative Feistel

- Representation over $\mathbb{F}^2$

$$(x, y) \mapsto (x, \underbrace{y \cdot G(x)}_{\text{Multiplication}} + F(x))$$

  for $G(x) \neq 0$

- Generalization over $\mathbb{F}^t$

$$(x_0, \ldots, x_{t-1}) \mapsto (x_0, x_1 \cdot G_1(x_0) + F_1(x_0), x_2 \cdot G_2(x_0, x_1) + F_2(x_0, x_1), \ldots)$$

  - Setting $G_i(\cdot) = 1$ results in classical Feistel

- How to choose G for invertibility?

Horst – Multiplicative Feistel

- Representation over $\mathbb{F}^2$

$$(x, y) \mapsto (x, \underbrace{y \cdot G(x)}_{\text{Multiplication}} + F(x))$$

  for $G(x) \neq 0$

- Generalization over $\mathbb{F}^t$

$$(x_0, \ldots, x_{t-1}) \mapsto (x_0, x_1 \cdot G_1(x_0) + F_1(x_0), x_2 \cdot G_2(x_0, x_1) + F_2(x_0, x_1), \ldots)$$

  - Setting $G_i(\cdot) = 1$ results in classical Feistel

- How to choose G for invertibility?

Horst – Multiplicative Feistel cont.

- Low-degree monomial does not work (since then $G(0) = 0$)
- Exploit fact that $x \mapsto x^2$ is not a permutation in $\mathbb{F}_p$
- Choose α, β such that

$$\alpha^2 - 4\beta \neq w^2 \quad \forall w \in \mathbb{F}_p$$

- Then $G(x) = x^2 + \alpha x + \beta = 0$ has no solutions, hence $G(x) \neq 0$ for each x

$\longrightarrow$ Degree-2 function for G

Merging SPN and Horst: GRIFFIN-π

SPN: $\begin{cases} y_0 = {x_0}^{1/d} \\ y_1 = {x_1}^{d} \end{cases}$

Horst: $\begin{cases} y_2 & = x_2 \cdot \left(L_2(y_0, y_1, 0)^2 + \alpha_2 \cdot L_2(y_0, y_1, 0) + \beta_2\right) \\ y_3 & = x_3 \cdot \left(L_3(y_0, y_1, x_2)^2 + \alpha_3 \cdot L_3(y_0, y_1, x_2) + \beta_3\right) \\ & \vdots \\ y_{t-1} & = x_{t-1} \cdot \left(L_{t-1}(y_0, y_1, x_{t-2})^2 + \alpha_{t-1} \cdot L_{t-1}(y_0, y_1, x_{t-2}) + \beta_{t-1}\right) \end{cases}$

- L_i is linear in the inputs

Merging SPN and Horst: Griffin-π

$$\text{SPN:}\quad \begin{cases} y_0 = {x_0}^{1/d} \\ y_1 = {x_1}^{d} \end{cases}$$

$$\texttt{Horst:}\quad \begin{cases} y_2 & = x_2 \cdot \left(L_2(y_0, y_1, 0)^2 + \alpha_2 \cdot L_2(y_0, y_1, 0) + \beta_2\right) \\ y_3 & = x_3 \cdot \left(L_3(y_0, y_1, x_2)^2 + \alpha_3 \cdot L_3(y_0, y_1, x_2) + \beta_3\right) \\ & \vdots \\ y_{t-1} & = x_{t-1} \cdot \left(L_{t-1}(y_0, y_1, x_{t-2})^2 + \alpha_{t-1} \cdot L_{t-1}(y_0, y_1, x_{t-2}) + \beta_{t-1}\right) \end{cases}$$

- L_i is linear in the inputs

What do we achieve?

- Fast degree growth in both directions due to y_0, y_1
  - Constraints of degree d
- `Horst` part: no degree 2, but...
  - `Horst` leads to degree 3 (independent of d and p)

$\longrightarrow$ Seems algebraically stronger than classical Feistel

What do we achieve?

- Fast degree growth in both directions due to y_0, y_1
  - Constraints of degree d
- `Horst` part: no degree 2, but...
  - `Horst` leads to degree 3 (independent of d and p)

$\longrightarrow$ Seems algebraically stronger than classical Feistel

Algebraic Security of GRIFFIN-π with Feistel

- Two Gröbner basis strategies
- Intermediate variables
  - Practical degree of regularity d_{reg} constant for any number of rounds
  - Does not mean it is insecure, but potentially harder to analyze
- No intermediate variables (only for $x \mapsto x^{1/d}$ in SPN part)
  - Reduced degree of regularity due to missing multiplication
  - Faster Gröbner basis computation than with `Horst` when using same degrees

$\rightarrow$ Suggests `Horst` is algebraically stronger and more efficient than Feistel

- Formal analysis left as open problem

Algebraic Security of GRIFFIN-π with Feistel

- Two Gröbner basis strategies
- Intermediate variables
  - Practical degree of regularity d_{reg} constant for any number of rounds
  - Does not mean it is insecure, but potentially harder to analyze
- No intermediate variables (only for $x \mapsto x^{1/d}$ in SPN part)
  - Reduced degree of regularity due to missing multiplication
  - Faster Gröbner basis computation than with `Horst` when using same degrees

$\rightarrow$ Suggests `Horst` is algebraically stronger and more efficient than Feistel

- Formal analysis left as open problem

Griffin-π Specification

- Affine layer
  - Multiplication by efficient matrix M with small values
  - Round constant addition
  - Good for plain performance, full diffusion
- Nonlinear layer
  - Defined by

$$y_i = \begin{cases} {x_0}^{1/d} & \text{if } i = 0, \\ {x_1}^{d} & \text{if } i = 1, \\ x_2 \cdot \left((L_i(y_0, y_1, 0))^2 + \alpha_2 \cdot L_i(y_0, y_1, 0) + \beta_2 \right) & \text{if } i = 2, \\ x_i \cdot \left((L_i(y_0, y_1, x_{i-1}))^2 + \alpha_i \cdot L_i(y_0, y_1, x_{i-1}) + \beta_i \right) & \text{otherwise.} \end{cases}$$

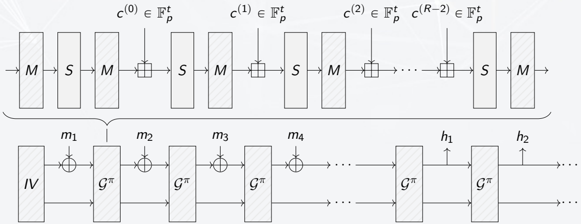
Griffin-π Specification

- Affine layer
  - Multiplication by efficient matrix M with small values
  - Round constant addition
  - Good for plain performance, full diffusion
- Nonlinear layer
  - Defined by

$$y_i = \begin{cases} {x_0}^{1/d} & \text{if } i = 0, \\ {x_1}^{d} & \text{if } i = 1, \\ x_2 \cdot \left((L_i(y_0, y_1, 0))^2 + \alpha_2 \cdot L_i(y_0, y_1, 0) + \beta_2 \right) & \text{if } i = 2, \\ x_i \cdot \left((L_i(y_0, y_1, x_{i-1}))^2 + \alpha_i \cdot L_i(y_0, y_1, x_{i-1}) + \beta_i \right) & \text{otherwise.} \end{cases}$$

Griffin Hash Function

- Sponge function



- Compression function

$$x \in \mathbb{F}_p^t \mapsto \mathcal{C}(x) := \mathrm{Tr}_n(\mathcal{G}^\pi(x) + x) \in \mathbb{F}_p^n$$

GRIFFIN-π Security

- Statistical attacks
  - No straightforward application for wide-trail strategy (alignment)
  - Simple argument thanks to large field size
- Algebraic attacks
  - Often the strongest attacks against these schemes
  - Higher-order diff., interpolation avoided by high degrees, density
  - Various strategies for Gröbner basis attacks
  - Non-aligned approach seems good here
- ≈ 10 rounds for practically relevant instances

GRIFFIN-π Security

- Statistical attacks
  - No straightforward application for wide-trail strategy (alignment)
  - Simple argument thanks to large field size
- Algebraic attacks
  - Often the strongest attacks against these schemes
  - Higher-order diff., interpolation avoided by high degrees, density
  - Various strategies for Gröbner basis attacks
  - Non-aligned approach seems good here
- ≈ 10 rounds for practically relevant instances

GRIFFIN Performance

- Much better SNARK performance than competitors
- Similar STARK performance as currently best constructions
  - Better plain performance than close competitors
- Scales well with larger state sizes
  - Only one expensive $x \mapsto x^{1/d}$ computation per round
  - Efficient linear layer

GRIFFIN Performance in SNARKs

- Security level of 128 bits
- `bellman_ce` library generating Groth16 [Gro16] proofs

Permutation	State size t							
	3		4		8		12	
	Prove	R1CS	Prove	R1CS	Prove	R1CS	Prove	R1CS
GRIFFIN	**39.08**	**96**	**42.46**	**110**	**60.54**	**162**	**82.29**	**234**
NEPTUNE [GOPS22]	–	–	71.41	228	95.99	264	121.04	306
POSEIDON [GKR+21]	74.98	240	87.99	264	108.22	363	131.89	459
Rescue-Prime [SAD20]	76.09	252	76.70	264	94.00	384	138.94	576
$\mathrm{GMiMC}_{\mathrm{erf}}$ [AGP+19]	172.78	678	179.11	684	189.07	708	252.36	942
Anemoi [BBC+22]	–	–	n/a	120	n/a	200	n/a	300

Conclusion

- Focus on proof performance in various frameworks
  - Round function built specifically for this purpose
- New design strategy for permutations
  - Merge advantages of SPN and `Horst`
- GRIFFIN used in various projects
  - Winterfell by Facebook[1]
- Future work
  - Non-aligned schemes against algebraic attacks?
  - `Horst` vs. Feistel

[1] https://github.com/facebook/winterfell/tree/main/crypto/src/hash

Questions?

References I

[AGP+19] Martin R. Albrecht, Lorenzo Grassi, Léo Perrin, Sebastian Ramacher, Christian Rechberger, Dragos Rotaru, Arnab Roy, and Markus Schofnegger. "Feistel Structures for MPC, and More". In: ESORICS 2019. Vol. 11736. LNCS. 2019, pp. 151–171.

[BBC+22] Clémence Bouvier, Pierre Briaud, Pyrros Chaidos, Léo Perrin, Robin Salen, Vesselin Velichkov, and Danny Willems. "New Design Techniques for Efficient Arithmetization-Oriented Hash Functions: Anemoi Permutations and Jive Compression Mode". In: IACR Cryptol. ePrint Arch. (2022), p. 840.

[GKR+21] Lorenzo Grassi, Dmitry Khovratovich, Christian Rechberger, Arnab Roy, and Markus Schofnegger. "Poseidon: A New Hash Function for Zero-Knowledge Proof Systems". In: USENIX Security Symposium. USENIX Association, 2021, pp. 519–535.

[GOPS22] Lorenzo Grassi, Silvia Onofri, Marco Pedicini, and Luca Sozzi. "Invertible Quadratic Non-Linear Layers for MPC-/FHE-/ZK-Friendly Schemes over Fnp: Application to Poseidon". In: IACR Trans. Symmetric Cryptol. 2022.3 (2022), pp. 20–72.

[Gro16] Jens Groth. "On the Size of Pairing-Based Non-interactive Arguments". In: EUROCRYPT (2). Vol. 9666. Lecture Notes in Computer Science. Springer, 2016, pp. 305–326.

[SAD20] Alan Szepieniec, Tomer Ashur, and Siemen Dhooghe. Rescue-Prime: a Standard Specification (SoK). Cryptology ePrint Archive, Report 2020/1143. 2020.