Anemoi and Jive

New Arithmetization-Oriented tools for Plonk-based applications.

Clémence Bouvier [1,2] and Danny Willems [3,4]

joint work with Pierre Briaud[1,2], Pyrros Chaidos[5], Léo Perrin[2],
Robin Salen[6] and Vesselin Velichkov[7,8]

[1]Sorbonne Université, [2]Inria Paris,
[3]Nomadic Labs, Paris, [4]Inria and LIX, CNRS

[5]National & Kapodistrian University of Athens, [6]Toposware Inc., Boston,
[7]University of Edinburgh, [8]Clearmatics, London,

ZKProof5, November 16th, 2022

Some Motivation

Anemoi: Family of ZK-friendly Hash functions

Improve PlonK state-of-the-art

Up to 54%

over highly optimized Poseidon

AnemoiJive: 51 constraints

Poseidon: 110 constraints

Content

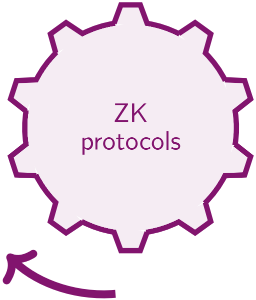
Anemoi and Jive
New Arithmetization-Oriented tools for Plonk-based applications.

1. Preliminaries
   - Emerging uses in symmetric cryptography
   - CCZ-equivalence

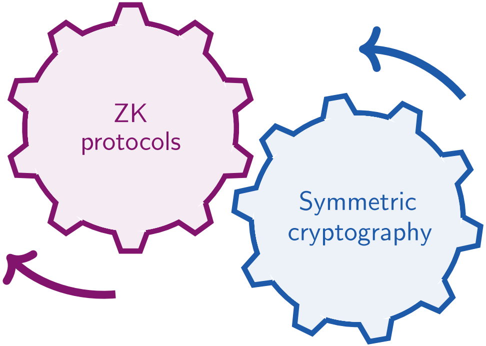
2. New tools for AO primitives
   - New permutation: `Anemoi`
   - New mode: `Jive`
   - Comparison to previous work

3. Conclusions

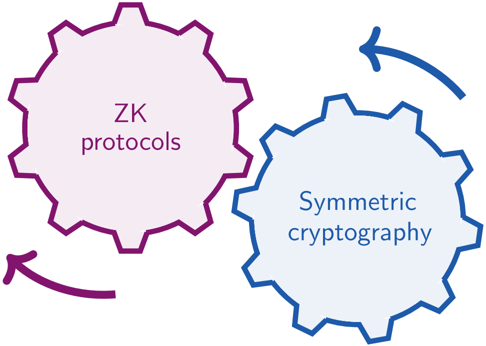
A need of new primitives



A need of new primitives



A need of new primitives



Arithmetization-oriented primitives

$\Rightarrow$ What differs from the "usual" case?

Comparison with "usual" case

A new environment

"Usual" case

- Field size:
  $\mathbb{F}_{2^n}$, with $n \simeq 4, 8$ (AES: $n = 8$).
- Operations:
  logical gates/CPU instructions

Arithmetization-friendly

- Field size:
  $\mathbb{F}_q$, with $q \in \{2^n, p\}, p \simeq 2^n$, $n \geq 64$.
- Operations:
  large finite-field arithmetic

Comparison with "usual" case

A new environment

"Usual" case

- Field size:
  $\mathbb{F}_{2^n}$, with $n \simeq 4, 8$ (AES: $n = 8$).
- Operations:
  logical gates/CPU instructions

Arithmetization-friendly

- Field size:
  $\mathbb{F}_q$, with $q \in \{2^n, p\}, p \simeq 2^n$, $n \geq 64$.
- Operations:
  large finite-field arithmetic

$\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$, with p given for instance by the order of commonly used pairing-friendly elliptic curves

Examples:

- Curve BLS12-381 $\log_2 p = 255$

  $p = 52435875175126190479447740508185965837690552500527637822603658699938581184513$

- Curve BLS12-377 $\log_2 p = 253$

  $p = 84446174942837042424882493878154653137589933515406382793523345591740923904 1$

Comparison with "usual" case

A new environment

"Usual" case

- Field size:
  $\mathbb{F}_{2^n}$, with $n \simeq 4, 8$ (AES: $n = 8$).
- Operations:
  logical gates/CPU instructions

Arithmetization-friendly

- Field size:
  $\mathbb{F}_q$, with $q \in \{2^n, p\}, p \simeq 2^n$, $n \geq 64$.
- Operations:
  large finite-field arithmetic

New properties

"Usual" case

- Operations:
  $$y \leftarrow E(x)$$
- Efficiency:
  implementation in software/hardware

Arithmetization-friendly

- Operations:
  $$y == E(x)$$
- Efficiency:
  integration within advanced protocols

Comparison with "usual" case

A new environment

"Usual" case

- Field size:
  $\mathbb{F}_{2^n}$, with $n \simeq 4, 8$ (AES: $n = 8$).
- Operations:
  logical gates/CPU instructions

Arithmetization-friendly

- Field size:
  $\mathbb{F}_q$, with $q \in \{2^n, p\}, p \simeq 2^n$, $n \geq 64$.
- Operations:
  large finite-field arithmetic

New properties

"Usual" case

- Operations:
  $$y \leftarrow E(x)$$
- Efficiency:
  implementation in software/hardware

Arithmetization-friendly

- Operations:
  $$y == E(x)$$
- Efficiency:
  integration within advanced protocols

Our approach

Need: verification using few multiplications.

Our approach

Need: verification using few multiplications.

First approach: evaluation also using few multiplications.

Our approach

Need: verification using few multiplications.

First approach: evaluation also using few multiplications.

$\boxed{y \leftarrow E(x)}$ $\rightsquigarrow E$: low degree

$\boxed{y == E(x)}$ $\rightsquigarrow E$: low degree

Our approach

Need: verification using few multiplications.

First approach: evaluation also using few multiplications.

$\boxed{y \leftarrow E(x)}$ $\rightsquigarrow$ E: low degree

$\boxed{y == E(x)}$ $\rightsquigarrow$ E: low degree

$\Rightarrow$ potential vulnerability to some attacks...

Our approach

Need: verification using few multiplications.

First approach: evaluation also using few multiplications.

$y \leftarrow E(x)$ $\rightsquigarrow E$: low degree

$y == E(x)$ $\rightsquigarrow E$: low degree

$\Rightarrow$ potential vulnerability to some attacks...

New approach:

CCZ-equivalence

Our vision

A function is arithmetization-oriented if it is **CCZ-equivalent** to a function that can be verified efficiently.

Our approach

Need: verification using few multiplications.

First approach: evaluation also using few multiplications.

$y \leftarrow E(x)$ $\rightsquigarrow$ E: low degree

$y == E(x)$ $\rightsquigarrow$ E: low degree

$\Rightarrow$ potential vulnerability to some attacks...

New approach:

CCZ-equivalence

Our vision

A function is arithmetization-oriented if it is **CCZ-equivalent** to a function that can be verified efficiently.

$y \leftarrow F(x)$ $\rightsquigarrow$ F: high degree

$v == G(u)$ $\rightsquigarrow$ G: low degree

CCZ-equivalence

Definition [Carlet, Charpin, Zinoviev, DCC98]

$F : \mathbb{F}_q \rightarrow \mathbb{F}_q$ and $G : \mathbb{F}_q \rightarrow \mathbb{F}_q$ are **CCZ-equivalent** if

$$\Gamma_F = \left\{ (x, F(x)) \mid x \in \mathbb{F}_q \right\} = \mathcal{A}(\Gamma_G) = \left\{ \mathcal{A}(x, G(x)) \mid x \in \mathbb{F}_q \right\} ,$$

where $\mathcal{A}$ is an affine permutation, $\mathcal{A}(x) = \mathcal{L}(x) + c$.

CCZ-equivalence

Definition [Carlet, Charpin, Zinoviev, DCC98]

$F : \mathbb{F}_q \to \mathbb{F}_q$ and $G : \mathbb{F}_q \to \mathbb{F}_q$ are **CCZ-equivalent** if

$$\Gamma_F = \left\{ (x, F(x)) \mid x \in \mathbb{F}_q \right\} = \mathcal{A}(\Gamma_G) = \left\{ \mathcal{A}(x, G(x)) \mid x \in \mathbb{F}_q \right\} ,$$

where $\mathcal{A}$ is an affine permutation, $\mathcal{A}(x) = \mathcal{L}(x) + c$.

Important things to remember!

- ⋆ Verification is the same: if $(x, y) = \mathcal{A}((u, v))$ with $y \leftarrow F(x)$, $v \leftarrow G(u)$

$$y == F(x)? \iff v == G(u)?$$

CCZ-equivalence

Definition [Carlet, Charpin, Zinoviev, DCC98]

$F : \mathbb{F}_q \to \mathbb{F}_q$ and $G : \mathbb{F}_q \to \mathbb{F}_q$ are **CCZ-equivalent** if

$$\Gamma_F = \left\{ (x, F(x)) \mid x \in \mathbb{F}_q \right\} = \mathcal{A}(\Gamma_G) = \left\{ \mathcal{A}(x, G(x)) \mid x \in \mathbb{F}_q \right\} ,$$

where $\mathcal{A}$ is an affine permutation, $\mathcal{A}(x) = \mathcal{L}(x) + c$.

Important things to remember!

- ⋆ Verification is the same: if $(x, y) = \mathcal{A}((u, v))$ with $y \leftarrow F(x)$, $v \leftarrow G(u)$

$$y == F(x)? \iff v == G(u)?$$

- ⋆ The degree is not preserved.

CCZ-equivalence

Definition [Carlet, Charpin, Zinoviev, DCC98]

$F : \mathbb{F}_q \rightarrow \mathbb{F}_q$ and $G : \mathbb{F}_q \rightarrow \mathbb{F}_q$ are **CCZ-equivalent** if

$$\Gamma_F = \left\{ (x, F(x)) \mid x \in \mathbb{F}_q \right\} = \mathcal{A}(\Gamma_G) = \left\{ \mathcal{A}(x, G(x)) \mid x \in \mathbb{F}_q \right\} ,$$

where $\mathcal{A}$ is an affine permutation, $\mathcal{A}(x) = \mathcal{L}(x) + c$.

Important things to remember!

- Verification is the same: if $(x, y) = \mathcal{A}((u, v))$ with $y \leftarrow F(x)$, $v \leftarrow G(u)$

$$y == F(x)? \iff v == G(u)?$$

- The degree is not preserved.

Why Anemoi?

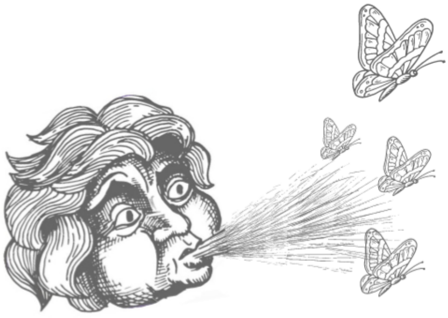
- **Anemoi**
  Family of ZK-friendly Hash functions

Why Anemoi?

- ⋆ **Anemoi**
  Family of ZK-friendly Hash functions

  ⇓

- ⋆ **Anemoi**
  Greek gods of winds

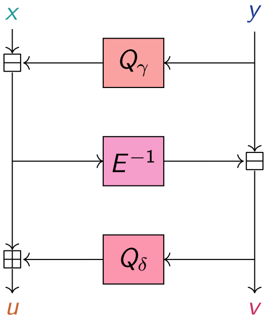


The Flystel

Butterfly + Feistel ⇒ Flystel

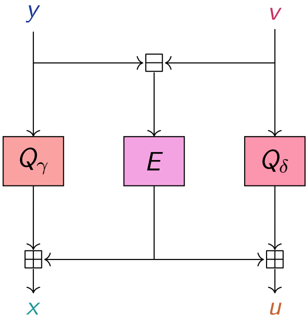
A 3-round Feistel-network with
$Q_\gamma : \mathbb{F}_q \to \mathbb{F}_q$ and $Q_\delta : \mathbb{F}_q \to \mathbb{F}_q$ two quadratic functions, and $E : \mathbb{F}_q \to \mathbb{F}_q$ a permutation

High-degree permutation



Open Flystel $\mathcal{H}$.

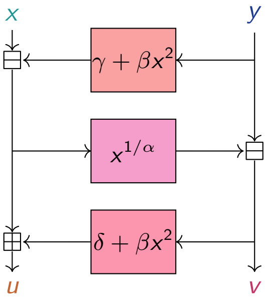
Low-degree function



Closed Flystel $\mathcal{V}$.

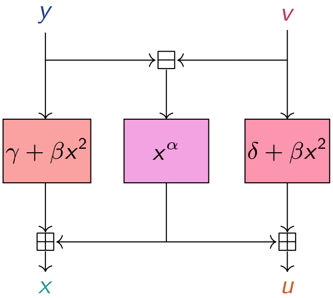
Flystel in $\mathbb{F}_p$

$$Q_\gamma : \mathbb{F}_p \to \mathbb{F}_p, x \mapsto \gamma + \beta x^2 \qquad Q_\delta : \mathbb{F}_p \to \mathbb{F}_p, x \mapsto \delta + \beta x^2 \qquad E : \mathbb{F}_p \to \mathbb{F}_p, x \mapsto x^\alpha$$



Open Flystel$_p$.

usually
$\alpha = 3$ or 5.



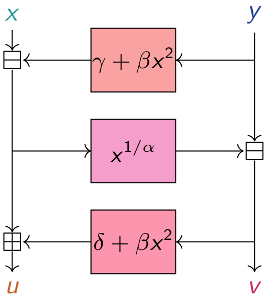
Closed Flystel$_p$.

Flystel in $\mathbb{F}_p$

$$Q_\gamma : \mathbb{F}_p \to \mathbb{F}_p, x \mapsto \gamma + \beta x^2 \qquad Q_\delta : \mathbb{F}_p \to \mathbb{F}_p, x \mapsto \delta + \beta x^2 \qquad E : \mathbb{F}_p \to \mathbb{F}_p, x \mapsto x^\alpha$$

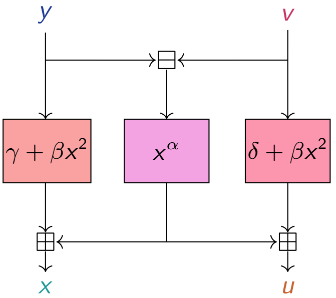
Example: Curve BLS12-381:

$$\begin{cases} p & = 52435875175126190479447740508185965837690552500527637822603658699938581184513 \\ \alpha & = 5 \\ \alpha^{-1} & = 20974350070050476191779096203274386335076221000211055129041463479975432473805 \end{cases}$$



Open Flystel$_p$.

usually $\alpha = 3$ or 5.



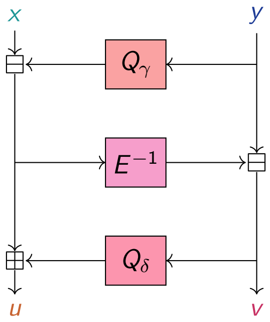
Closed Flystel$_p$.

Flystel and CCZ-equivalence

$\mathcal{H}$ and $\mathcal{V}$
are CCZ-equivalent

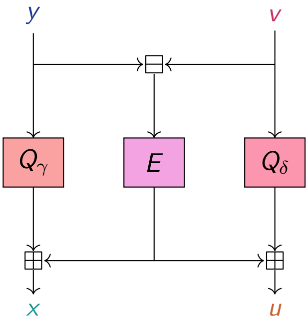
$$\Gamma_{\mathcal{H}} = \{(\ (x,y),\ \mathcal{H}((x,y))\ )\mid (x,y)\in \mathbb{F}_q^2\}$$
$$= \mathcal{A}\left(\{(\ (v,y),\ \mathcal{V}((v,y))\ )\mid (v,y)\in \mathbb{F}_q^2\}\right) = \mathcal{A}(\Gamma_{\mathcal{V}})$$

High-degree permutation



Open Flystel $\mathcal{H}$.

Low-degree function

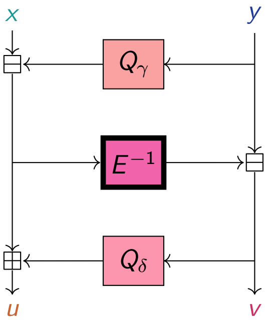


Closed Flystel $\mathcal{V}$.

Advantage of CCZ-equivalence

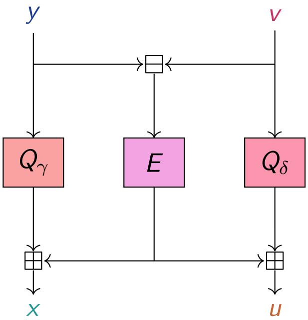
⋆ High Degree Evaluation.

High-degree permutation



Open Flystel $\mathcal{H}$.

Low-degree function



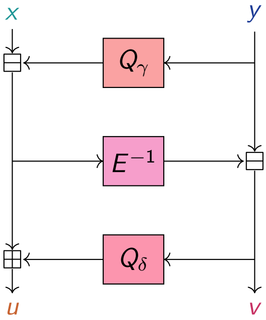
Closed Flystel $\mathcal{V}$.

Advantage of CCZ-equivalence

- ⋆ High Degree Evaluation.
- ⋆ Low Cost Verification.

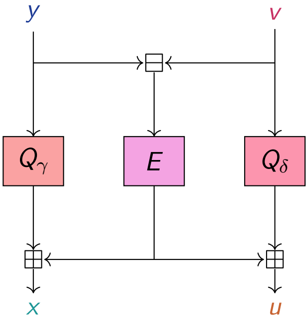
$$(u, v) == \mathcal{H}(x, y) \Leftrightarrow (x, u) == \mathcal{V}(y, v)$$

High-degree permutation



Open `Flystel` $\mathcal{H}$.

Low-degree function

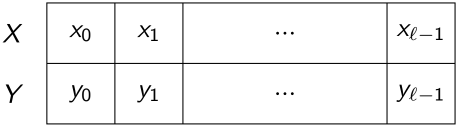


Closed `Flystel` $\mathcal{V}$.

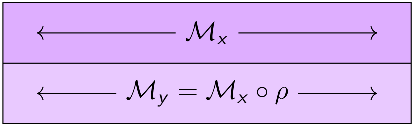
The SPN Structure

SPN: Substitution-Permutation Network

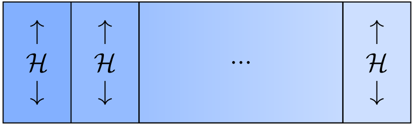
The internal state of Anemoi and its basic operations:



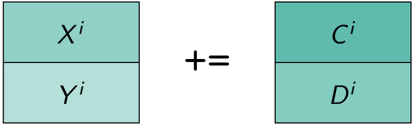
(a) *Internal state*



(b) *The diffusion layer (matrix multiplication).*

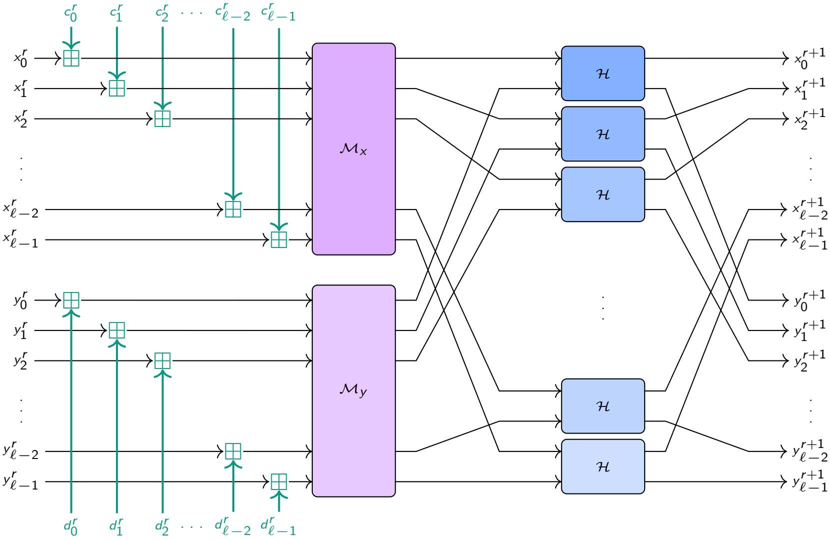


(c) *The confusion or S-box layer $\mathcal{H}$ (the Flystel).*



(d) *The constant addition.*

The SPN Structure



Overview of Anemoi.

Number of rounds

$$\texttt{Anemoi}_{q,\alpha,\ell} = \mathcal{M} \circ \mathrm{R}_{n_r-1} \circ \ldots \circ \mathrm{R}_0$$

⇒ Choosing the number of rounds:

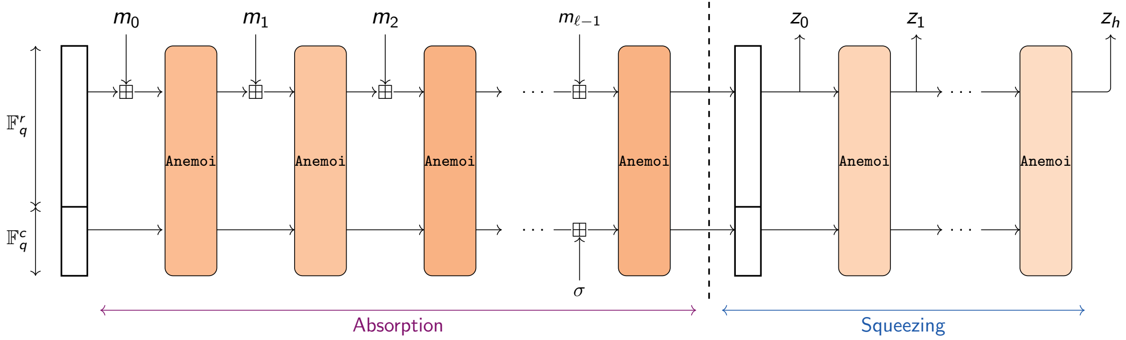
$$n_r \geq \max \left\{ 10 \; , \; \underbrace{1+\ell}_{\text{security margin}} + \underbrace{\min \left\{ r \in \mathbb{N} \;\middle|\; \binom{2\ell r + \alpha + 1 + 2 \cdot (\ell r - 2)}{2\ell r}^2 \geq 2^s \right\}}_{\text{to prevent algebraic attacks}} \right\} .$$

α	3	5	7	11	13	17
$\ell = 1$	19	19	18	18	17	16
$\ell = 2$	12	12	11	11	11	10
$\ell = 3$	10	10	10	10	10	10
$\ell = 4$	10	10	10	10	10	10

Number of Rounds of `Anemoi` *($s = 128$).*

New Mode: Jive

- Hash function (random oracle):
  - input: arbitrary length
  - ouput: fixed length

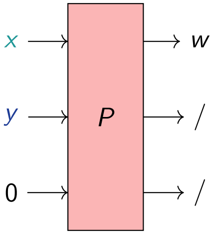


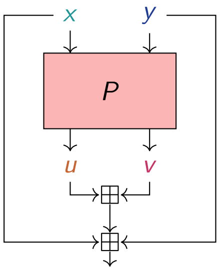
New Mode: Jive

- ⋆ Hash function (random oracle):
  - ⋆ input: arbitrary length
  - ⋆ ouput: fixed length
- ⋆ Compression function (Merkle-tree):
  - ⋆ input: fixed length
  - ⋆ output: (input length) /2

Dedicated mode ⇒ 2 words in 1

$$(x, y) \mapsto x + y + u + v \ .$$





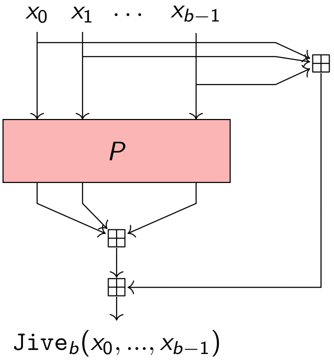
$\texttt{Jive}_2(x, y)$

New Mode: Jive

- Hash function (random oracle):
  - input: arbitrary length
  - ouput: fixed length
- Compression function (Merkle-tree):
  - input: fixed length
  - output: (input length) /b

Dedicated mode ⇒ b words in 1

$$\text{Jive}_b(P) : \begin{cases} (\mathbb{F}_q^m)^b & \to \mathbb{F}_q^m \\ (x_0, \ldots, x_{b-1}) & \mapsto \sum_{i=0}^{b-1} (x_i + P_i(x_0, \ldots, x_{b-1})) \end{cases} .$$



Some Motivation

Anemoi: Family of ZK-friendly Hash functions

Improve PlonK state-of-the-art

Up to 54%

over highly optimized Poseidon

AnemoiJive: 51 constraints

Poseidon: 110 constraints

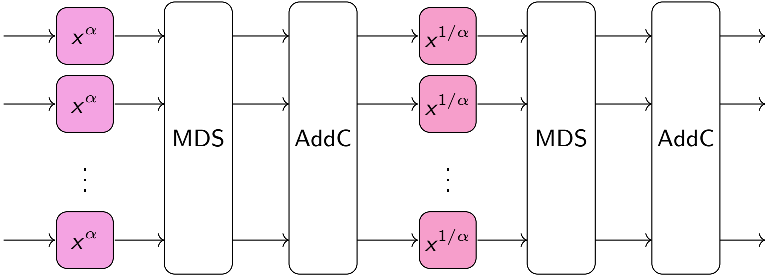
Rescue–Prime

[Aly et al., ToSC20]

- ⋆ S-Box layer
- ⋆ Linear layer: MDS
- ⋆ Round constants addition: AddC

$S : x \mapsto x^{\alpha}$, and $S^{-1} : x \mapsto x^{1/\alpha}$

$R \approx 10$



Overview of Rescue–Prime.

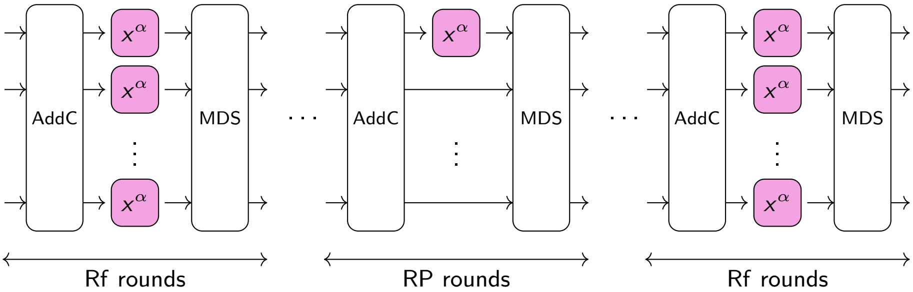
Poseidon

[Grassi et al., USENIX21]

- S-Box layer
- Linear layer: MDS
- Round constants addition: AddC

$S : x \mapsto x^\alpha$

$R = \mathrm{RF} + \mathrm{RP} \approx 50$



Overview of Poseidon.

Griffin

[Grassi et al. 2022]

- S-Box layer
- Linear layer: MDS
- Round constants addition: AddC

S: new design

$R \approx 12$

$$S(x_0, \ldots, x_{t-1}) = y_0 \mid\mid \ldots \mid\mid y_{t-1}$$

$$\begin{aligned} y_0 &= x_0^{\frac{1}{\alpha}} \\ y_1 &= x_1^{\alpha} \\ y_2 &= x_2(L_2(y_0, y_1, 0)^2 + \alpha_2 \cdot L_2(y_0, y_1, 0) + \beta_2) \\ y_i &= x_i(L_i(y_0, y_1, x_{i-1})^2 + \alpha_i \cdot L_i(y_0, y_1, x_{i-1}) + \beta_i) \end{aligned}$$

where $L_i(y_0, y_1, x_{i-1}) = (i-1)y_0 + y_1 + x_{i-1}$

Some Benchmarks

	m	*Rescue′*	Poseidon	Griffin	Anemoi
R1CS	2	208	198	-	**76**
	4	224	232	112	**96**
	6	216	264	-	**120**
	8	256	296	176	**160**
PlonK	2	312	380	-	**173**
	4	560	1336	291	**220**
	6	756	3024	-	**320**
	8	1152	5448	635	**456**
AIR	2	156	300	-	**114**
	4	168	348	168	**144**
	6	**162**	396	-	180
	8	**192**	480	264	240

(a) *when* $\alpha = 3$

	m	*Rescue′*	Poseidon	Griffin	Anemoi
R1CS	2	240	216	-	**95**
	4	264	264	**110**	120
	6	288	315	-	**150**
	8	384	363	**162**	200
PlonK	2	320	344	-	**192**
	4	528	1032	253	**244**
	6	768	2265	-	**350**
	8	1280	4003	543	**496**
AIR	2	200	360	-	**190**
	4	**220**	440	**220**	240
	6	**240**	540	-	300
	8	**320**	640	360	400

(b) *when* $\alpha = 5$

Constraint comparison for Rescue–Prime, Poseidon, Griffin *and Anemoi* ($s = 128$)
for standard arithmetization, without optimization.

Some Benchmarks

	m	*Rescue'*	Poseidon	Griffin	Anemoi
R1CS	2	208	198	-	**76**
	4	224	232	112	**96**
	6	216	264	-	**120**
	8	256	296	176	**160**
PlonK	2	312	380	-	**173**
	4	560	1336	291	**220**
	6	756	3024	-	**320**
	8	1152	5448	635	**456**
AIR	2	156	300	-	**114**
	4	168	348	168	**144**
	6	**162**	396	-	180
	8	**192**	480	264	240

(a) *when* $\alpha = 3$

	m	*Rescue'*	Poseidon	Griffin	Anemoi
R1CS	2	240	216	-	**95**
	4	264	264	**110**	120
	6	288	315	-	**150**
	8	384	363	**162**	200
PlonK	2	320	344	-	**192**
	4	528	1032	253	**244**
	6	768	2265	-	**350**
	8	1280	4003	543	**496**
AIR	2	200	360	-	**190**
	4	**220**	440	**220**	240
	6	**240**	540	-	300
	8	**320**	640	360	400

(b) *when* $\alpha = 5$

Constraint comparison for Rescue–Prime, Poseidon, Griffin *and Anemoi* ($s = 128$) for standard arithmetization, without optimization.

Comparison for PlonK (with optimizations)

	m	Constraints
Poseidon	3	110
	2	88
Reinforced Concrete	3	378
	2	236
Rescue–Prime	3	252
Griffin	3	125
AnemoiJive	2	**79**

(a) *With 3 wires.*

	m	Constraints
Poseidon	3	98
	2	82
Reinforced Concrete	3	267
	2	174
Rescue–Prime	3	168
Griffin	3	111
AnemoiJive	2	**58**

(b) *With 4 wires.*

Constraints comparison with an additional custom gate for x^α and 'next' wires ($s = 128$).

Comparison for PlonK (with optimizations)

	m	Constraints
Poseidon	3	110
	2	88
Reinforced Concrete	3	378
	2	236
Rescue–Prime	3	252
Griffin	3	125
AnemoiJive	2	~~79~~ 51

(a) *With 3 wires.*

	m	Constraints
Poseidon	3	98
	2	82
Reinforced Concrete	3	267
	2	174
Rescue–Prime	3	168
Griffin	3	111
AnemoiJive	2	**58**

(b) *With 4 wires.*

Constraints comparison with an additional custom gate for x^α and 'next' wires ($s = 128$).

with an additional quadratic custom gate: 51 constraints

Native performance

Rescue–Prime-12-8	Poseidon-12-8	Griffin-12-8	Anemoi-8
11.39 μs	1.93 μs	3.13 μs	3.93 μs

2-to-1 compression functions for $\mathbb{F}_p$ with $p = 2^{64} - 2^{32} + 1$ ($s = 128$).

Rescue–Prime	Poseidon	Griffin	Anemoi
255.36 μs	14.43 μs	73.66 μs	115.82 μs

For $BLS12-381$, Anemoi is instantiated with state size of 2, others of 3 ($s = 128$)

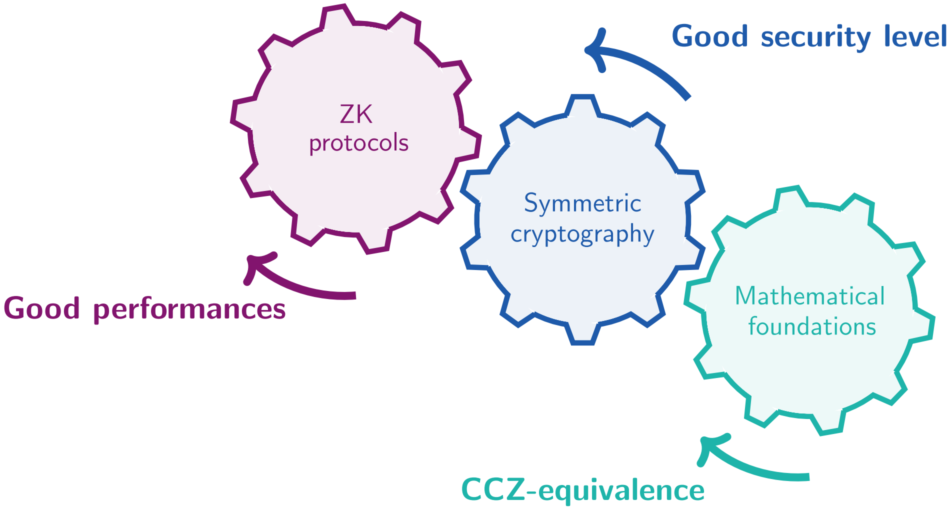
Conclusions

- A new family of ZK-friendly hash functions:

  ⇒ Anemoi efficient accross proof system, specially for PlonK

- New observations of fundamental interest:
  - Standalone components:
    - New S-box: Flystel
    - New mode: Jive
  - Identify a link between AO and CCZ-equivalence

Conclusions



Conclusions

- A new family of ZK-friendly hash functions:
  - ⇒ Anemoi efficient accross proof system, specially for PlonK
- New observations of fundamental interest:
  - Standalone components:
    - New S-box: Flystel
    - New mode: Jive
  - Identify a link between AO and CCZ-equivalence

☞ More details on https://ia.cr/2022/840

⇒ Another version of $\texttt{AnemoiJive}_3$ with TurboPlonK: 8.5 faster than Rescue–Prime.

☞ More details on https://ia.cr/2022/1487

Conclusions

- A new family of ZK-friendly hash functions:

  ⇒ Anemoi efficient accross proof system, specially for PlonK

- New observations of fundamental interest:
  - Standalone components:
    - New S-box: Flystel
    - New mode: Jive
  - Identify a link between AO and CCZ-equivalence

☞ More details on https://ia.cr/2022/840

⇒ Another version of $\texttt{AnemoiJive}_3$ with TurboPlonK: 8.5 faster than Rescue–Prime.

☞ More details on https://ia.cr/2022/1487

Thanks for your attention!