PRØST

A round-1 CAESAR submission

Elif Bilge Kavun [1] Martin M. Lauridsen [2] Gregor Leander [1]
Christian Rechberger [2] Peter Schwabe [3] Tolga Yalçın [4]

mmeh @ dtu.dk

[1]Horst Görtz Institute for IT-Security, Ruhr University Bochum, Germany

[2]DTU Compute, Technical University of Denmark, Denmark

[3]Digital Security Group, Radboud University Nijmegen, The Netherlands

[4]University of Information Science and Technology, Ohrid, Republic of Macedonia

DIAC 2014
Santa Barbara, August 23, 2014

Motivation + Features

Motivation + Features

As opposed to **mode designs** we wanted to focus on designing a **solid primitive**.

We chose a **permutation** due to its

- Simplicity
- Not requiring a key schedule

We plug the PRØST permutation into three excellent existing modes

- Upshot: Any analysis on those modes applies to our submissions

Features of PRØST which are not in AES (and thus AES-GCM)

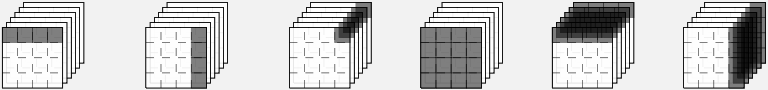
- Easy bit-sliced implementation
- Straightforward constant-time implementation
- Cheaper countermeasures due to 4-bit Sbox

Excellent bounds against many attack vectors despite relatively small state

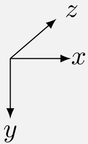
Specification + design rationale

Notation and state representation

- We use PRØST-n for the permutation on $2n$ bits
- Permutation size is **256 bits** ($n = 128$) or **512 bits** ($n = 256$)
- State is three-dimensional block of size $4 \times 4 \times d$, so $d \in \{16, 32\}$



(a) Row (b) Column (c) Lane (d) Slice (e) Plane (f) Sheet



(g) Axes

(We use Keccak notation for state parts)

The PRØST permutation

PRØST-n **iteratively** applies **round permutations** R_i T times, so

$$\text{PRØST-}n = R_{T-1} \circ \cdots \circ R_0.$$

- For PRØST-128 we have $T = 16$ rounds
- For PRØST-256 we have $T = 18$ rounds

Each round R_i, $0 \leq i < T$, is composed of smaller permutations:

$$R_i = \mathtt{AddConstants}_i \circ \mathtt{ShiftPlanes}_i \circ \mathtt{MixSlices} \circ \mathtt{SubRows}$$

(Subscript i denotes round-number dependency)

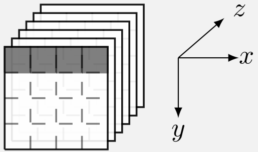
The PRØST round permutation

SubRows

MixSlices

ShiftPlanes$_i$

AddConstants$_i$



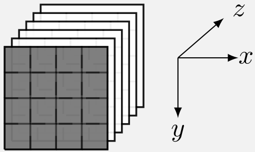
4-bit Sbox is applied to each **row** of the state. Why 4-bit?

- Well understood
- **Compact** implementation
- **Cheap masking** countermeasure

	Involution?	Algebraic degree	Instr. (AVR)	Max DP (#)	Max $\lvert\epsilon\rvert$ (#)
PRESENT	no	$(3, 3, 3, 2)$	20	2^{-2} (24)	2^{-2} (36)
PRINCE	no	$(3, 3, 3, 3)$	32	2^{-2} (15)	2^{-2} (30)
PRØST	yes	$(2, 2, 3, 3)$	10	2^{-2} (24)	2^{-2} (36)

The PRØST round permutation

SubRows
MixSlices
ShiftPlanes$_i$
AddConstants$_i$



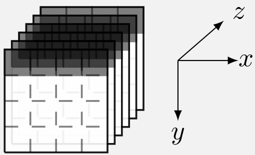
Each **slice** (seen over $\mathbb{F}_2^{16}$) is multiplied by a 16×16 matrix M over $\mathbb{F}_2$.

This matrix

- Has linear/differential **branch number 5** (MDS)
- Is **involutive**
- Has low density: **Hamming weight 88**
  (lowest we could find with given conditions w/ hardware assisted search)

The PRØST round permutation

`SubRows`
`MixSlices`
`ShiftPlanes`$_i$
`AddConstants`$_i$



Rotates each of the 4 **planes** in the positive z direction (front towards back).

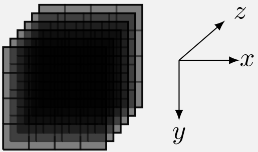
Like AES `ShiftRows`, but using different offsets every second round, from a **rotation matrix** $\pi \in \mathbb{Z}_d^{2\times 4}$.

Rotation constants chosen to

- Maximize **diffusion**
- Maximize differential/linear **trail weights**
- Use as many **multiples of** 8 as possible, otherwise minimize value

The PRØST round permutation

SubRows
MixSlices
$\texttt{ShiftPlanes}_i$
$\texttt{AddConstants}_i$



In each round, a constant is XORed to **each register** of the state to make rounds R_i different.

The constant added to the j**th** lane in **round** i is

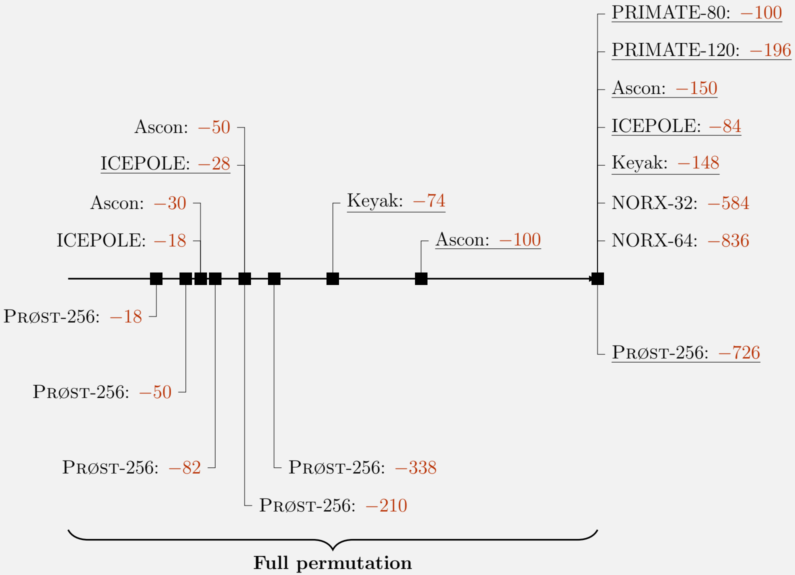
$$\begin{cases} c_1 \lll (i+j) & \text{when } j \text{ is even} \\ c_2 \lll (i+j) & \text{when } j \text{ is odd} \end{cases}$$

Constants c_1, c_2 are derived from Pi.

Security analysis

Analysis: Differential/Linear trail probabilities

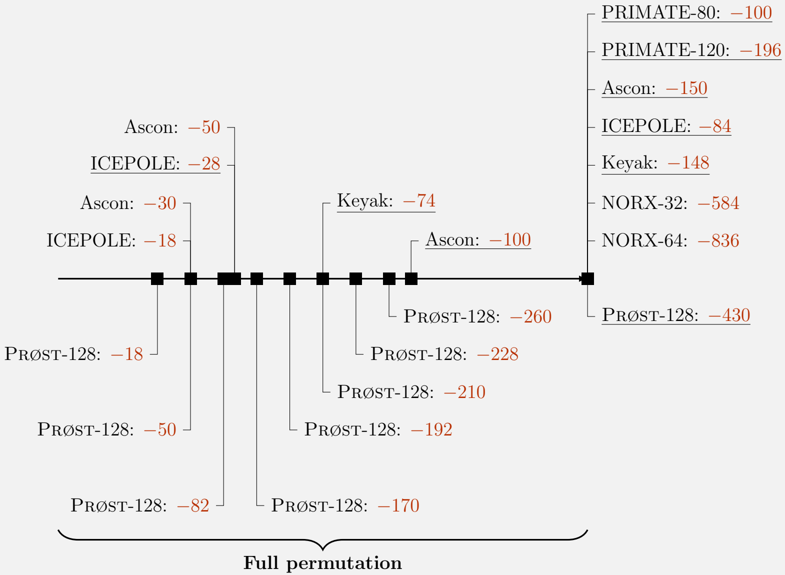
Numbers are $\log_2$ of upper bound, underlined are non-tight



Keyak: lake, sea and ocean

Analysis: Differential/Linear trail probabilities

Numbers are $\log_2$ of upper bound, underlined are non-tight



Keyak: lake, sea and ocean

Security analysis: Higher-order attacks

The number of rounds T chosen allow zero-sum distinguishers when Sbox degree is 2

The $\textsc{PrØst}$ Sbox yields algebraic degrees $(2, 2, 3, 3)$, so we believe our choice is conservative

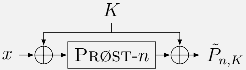
Interesting problem:

- Upper bounding algebraic degree when Sbox has mixed degrees

Proposals

The proposals: We propose the use of PRØST in...

- Block cipher-based **COPA** and **OTR**
  - by using the Single-key Even-Mansour construction



- Permutation-based **APE** “as is”
  - Using rate/capacity 128/128 for PRØST-128
  - Using rate/capacity 256/256 for PRØST-256

Elena Andreeva, Andrey Bogdanov, Atul Luykx, Bart Mennink, Elmar Tischhauser and Kan Yasuda
Parallelizable and Authenticated Online Ciphers
In Asiacrypt 2013, pages 424–443.

Kazuhiko Minematsu
Parallelizable Rate-1 Authenticated Encryption from Pseudorandom Functions
In Eurocrypt 2014, pages 275–292.

Elena Andreeva, Begül Bilgin, Andrey Bogdanov, Atul Luykx, Bart Mennink, Nicky Mouha and Kan Yasuda
APE: Authenticated Permutation-Based Encryption for Lightweight Cryptography
In FSE 2014

Fractional data

What we observed:

- Many **elegant** designs are crippled by **inelegant** handling of fractional data to avoid ciphertext expansion
- Begging for implementation errors

For simplicity

- **Always** 10^***-pad the message**

What do we gain?

- No special cases for fractional message blocks
- Avoids extra code/circuit size in software/hardware
- Less prone to implementation errors (quite frequent in practice!)
- Implementations are easier to optimize

Security goals

- PRØST-COPA/PRØST-OTR:
  - Mode proof: Birthday-bound in block size assuming underlying PRP
  - SK Even-Mansour: Birthday-bound attacks on $\tilde{P}_{n,K}$
  - Thus, we **conservatively** claim $2n/4$ bits of security
- PRØST-APE:
  - $c/2$ bits security assuming ideal permutation

Rank	Proposal	PT_{CONF}	PT_{INT}	AD_{INT}
1	PRØST-COPA-128	64	64	64
2	PRØST-COPA-256	128	128	128
3	PRØST-OTR-128	64	64	64
4	PRØST-OTR-256	128	128	128
5	PRØST-APE-256[256, 256]	128	128	128
6	PRØST-APE-128[128, 128]	64	64	64

Performance

Performance

Preliminary figures from vectorized implementations

- Intel(R) Core(TM) i5-3210M CPU @ 2.50 GHz

The PRØST permutation

- **4.24 cpb** with **8-way parallelization**

For PRØST-COPA

- Roughly **10.6 cpb** for **long messages**

More coming in near future...

Conclusion

Features of PRØST

- Easy bit-sliced implementation
- Straightforward constant-time implementation
- Cheaper countermeasures due to 4-bit Sbox
- No fractional data cases
- Excellent bounds against many attack vectors despite relatively small state
  - Sufficient security margin to reduce # of rounds
- Permutation cheap to inverse

Slides will be available at

`http://proest.compute.dtu.dk`

Thank you.