



Mystrium: Wide Block Encryption Efficient on Entry-Level Processors

Parisa Amiri Eliasi, Koustabh Ghosh, Joan Daemen

Radboud University, Nijmegen, the Netherlands

End-of-ESCADA Workshop presentation

August 27, 2024

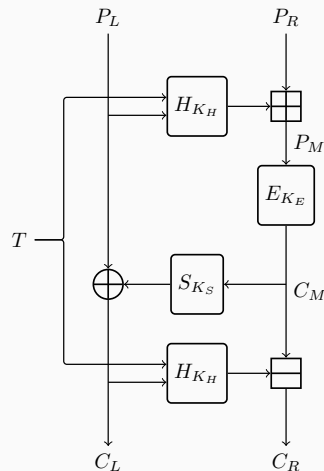
- Block ciphers have been the main building block for symmetric cryptography
- They operate on data of **fixed** and pre-determined length
- They are used in a **mode** of operation for encrypting data of **variable-length**
 - Galois/Counter Mode (GCM), OFB, or CBC
- Newer processors have cryptography extensions including **AES instructions**
- Therefore, it is **favorable** to use **AES** e.g. in **GCM mode**
- The security of these modes depends on the **proper** use of a **nonce/IV**

- In some applications we want to have security
 - **absence** of a nonce, or
 - in case of a nonce-**uniqueness** violation
- E.g. in disk encryption on entry-level processors:
 - **nonce reuse** is inevitable as there is nowhere to store a varying nonce
 - performance is critical (user experience, battery life)
 - the ciphertext can be no larger than the plaintext (sector size)
 - **no** cryptography extensions, including **AES instructions**, are available
 - standard constructions, like AES-GCM, are **not** an option
- The **best** that can be done is a *Wide Block Cipher (WBC)*

- Encrypt **arbitrarily** large strings such that:
each bit of the ciphertext depends on **each** bit of the plaintext and vice versa
- *Tweakable* WBCs have **one more input**: **Tweak**
 - with the same key but **a different tweak** we have **a different permutation**
- They are indistinguishable from a family of independent random permutations
- When the adversary:
 - has access to both encryption and decryption functions, and
 - can choose tweak and plaintext/ciphertext freely

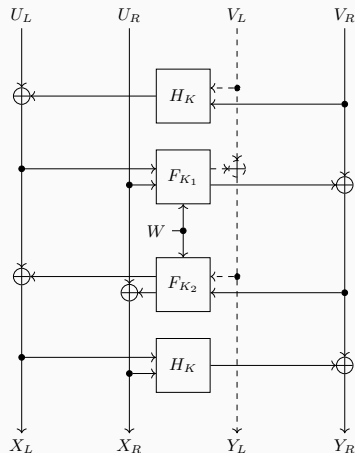
- They can be built using (unbalanced) Feistel network
- Many proposals require (per 128 bits of input) [CS08; MM07; Nan08]
 - either one or two AES invocations and
 - two $\text{GF}(2^{128})$ multiplications
- On ARM architectures **older** than ARMv8 there are **no cryptographic extensions**
 - e.g. ARMv7
- **Adiantum** [CB18] is the first Tweakable **WBC** that
 - has an acceptable performance on these processors
 - it was developed for encryption on Android devices with entry-level processors

- The HBSH mode is an unbalanced Feistel with:
 - an arbitrary and a fixed length branches
- Adiantum uses a combination of
 - $NH^T_{[Kro00]}$, Poly-1305 [Ber05] for **universal hashing**,
 - XChaCha12 [Ber+08; NL18] for the **stream cipher**, and
 - AES-256 [DR99] for the **single block cipher** application
- Using a **block cipher** is undesirable
 - no cryptographic instructions are available
 - decryption requires calculating the **inverse** of the block cipher.



our Design based on Double-Decker mode

- Double-decker is an unbalanced Feistel network
 - but **without a block cipher**
 - it uses **deck** (doubly extendable cryptographic keyed) function [Dae+18]
 - it is based on the **Farfalle-WBC** construction [Ber+16]
- The double-decker mode has
 - two branches with a size of *at least* **256 bits** for **128 bits** of security, U_L, V_R
 - two arbitrary length branches, U_R, V_L
- It consists of
 - two Feistel rounds with the deck function and
 - two Feistel rounds of a keyed hash function
- Our Tweakable WBC is called **Mystrium**



What is a deck function?

Taken from [Băc+22]

- A deck function is a **primitive type**, the same concept as a **block cipher**
- It is hard to be distinguished from a **random oracle**

A deck function F_K

$$Z = 0^n + F_K(X^{(1)}, \dots, X^{(m)}) \ll q$$

doubly extendable cryptographic keyed function

- input: K , and sequence of strings: $X^{(1)}, \dots, X^{(m)}$
- output: Potentially infinite output
 - pseudo-random function of the input
 - taking n bits starting from offset q

A deck function F_K

$$Z = 0^n + F_K(X^{(1)}, \dots, X^{(m)}) \ll q$$

doubly extendable cryptographic keyed function

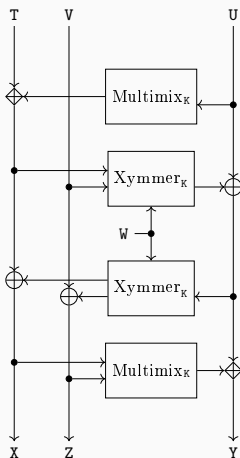
Efficient Incrementality, extendable input:

- ① Compute $F_k(X)$
- ② Compute $F_k(X; Y)$: cost independent of X

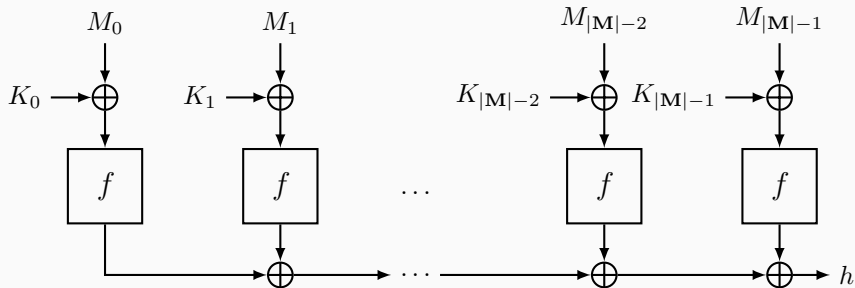
Efficient Incrementality, extendable output:

- ① Request n_1 bits from offset 0
- ② Request n_2 bits from offset n_1 : cost independent of n_1

- Its **non-linearity** comes exclusively from:
 - **32-bit integer multiplications**
- Mystrium uses
 - a keyed hash function called Multimix,
 - and a newly proposed deck function named Xymmer
- Multimix is defined on top of Multimixer-128 [GED23]
 - it takes a short key (256 bits) and can process:
any sequence of arbitrary length byte-strings
- It is **highly efficient** on software platforms
 - w **SIMD instructions**

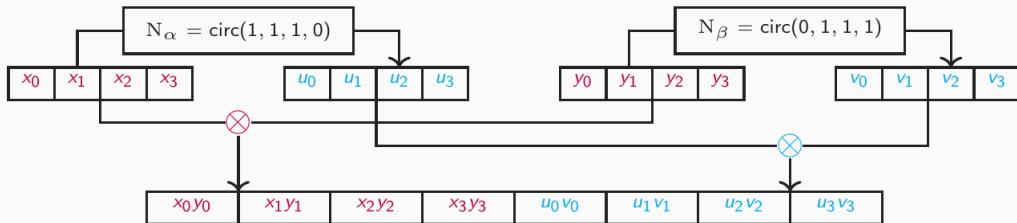


- Multimix is defined on top of Multimixer-128 [GED23]
- **Multimixer-128** is a parallel keyed hash function [FRD23; Gho+23]
 - it is ϵ - Δ universal with $\epsilon = 2^{-127}$



Multimixer-128

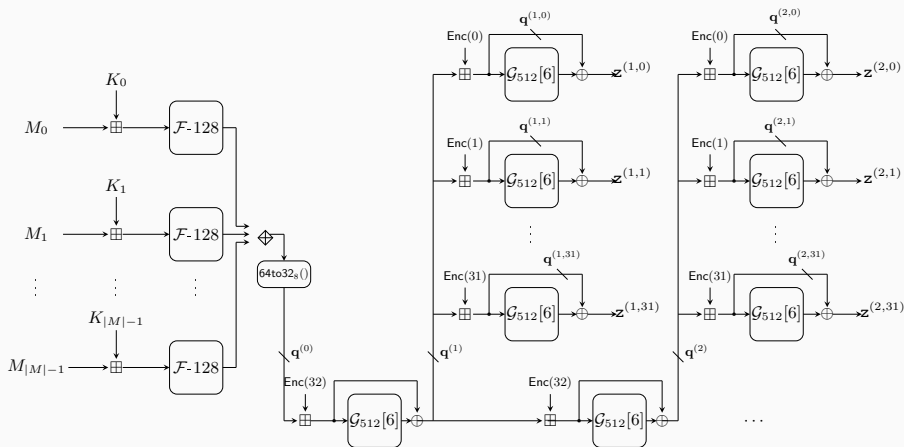
- The public function of Multimixer-128 is called $\mathcal{F}$ -128



- x_i and y_i are 32-bit integers
- It is highly efficient on software platforms with SIMD instructions
 - which compute 32-bit integer multiplications efficiently

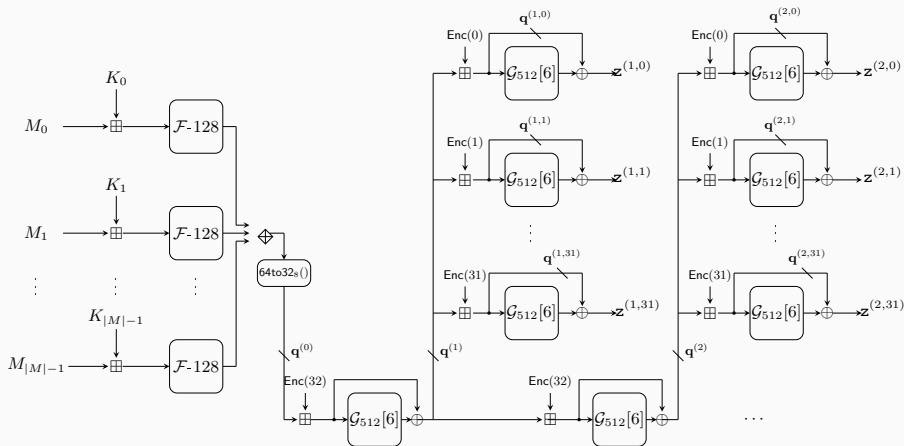
Our Deck function: Xymmer

- Xymmer structure is inspired by [Farfalle](#) [Ber+16]



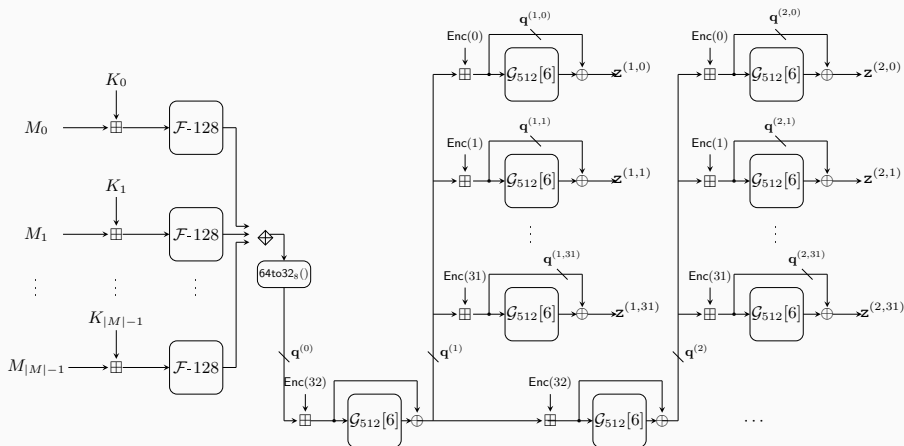
Our Deck function: Xymmer

- Compression uses **Multimix**, arbitrary length input to a 512-bit state
- The state is then expanded to an output bit-string of arbitrary length



Our Deck function: Xymmer

- Expansion makes use of the **Kirby** construction [LBD23]
- It allows efficient **incremental computing** [Dae+18]
- As the compression phase, the expansion phase also supports **parallelism**.



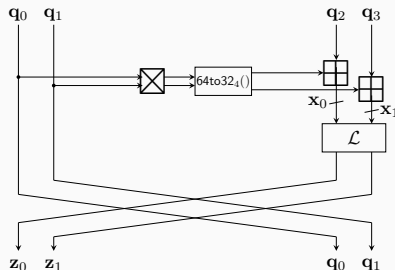
Xymmer, Expansion Phase

- We designed a public permutation:
 - $\mathcal{G}_{512}[6]: (\mathbb{Z}/2^{32}\mathbb{Z})^{16} \rightarrow (\mathbb{Z}/2^{32}\mathbb{Z})^{16}$
with 6 Feistel rounds

- q_i are 4 32-bit words
- non-linearity comes from:
 - 32-bit integer multiplication

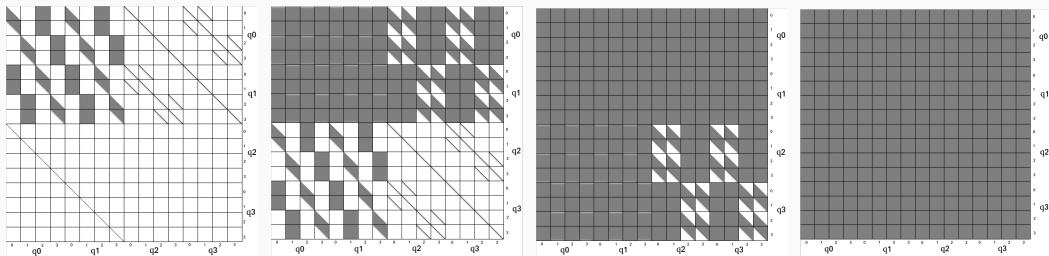
- For $\mathcal{L}$ we apply a 4×4 circulant matrix

$$\begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ x_1 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} z_0 \\ z_1 \\ z_4 \\ z_5 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_2 \\ x_3 \\ x_6 \\ x_7 \end{pmatrix} = \begin{pmatrix} z_2 \\ z_3 \\ z_6 \\ z_7 \end{pmatrix}.$$



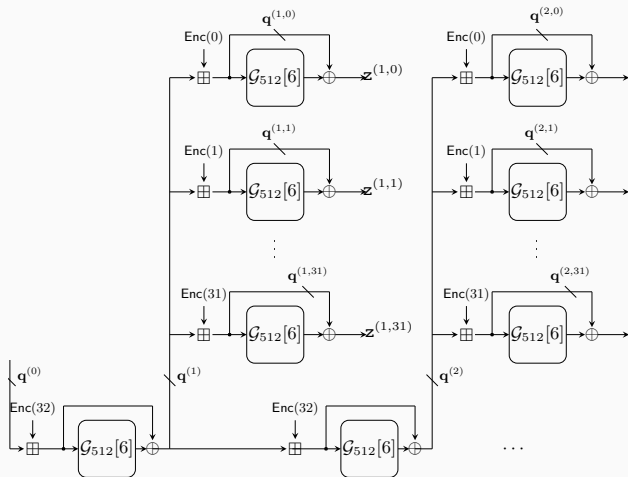
- similar to circulant matrix used in $\mathcal{F}$ -128 [GED23] with branch number 4

- How $\mathcal{G}_{512}[r]$ realizes the avalanche criteria through the rounds?
 - we use avalanche probability vector defined in [Dae+18]
- For a cryptographic function F , we calculate $P_{i,j}$
 - probability that output bit of F in position i flips due to flip in input bit j
- We depict $P_{i,j}$ matrix as a 2-D figure:



- More analysis are done

- The expansion phase of Xymmer is
 - uses Kirby construction [LBD23]
 - with the 512-bit $P = \mathcal{G}_{512}[6]$
- It achieves 256 bits of security
 - in the ideal permutation model,
 - its inputs form a prefix-free set
- Efficient on platforms with limited resources



Distinguishing advantage bound for the deck-based wide block cipher modes:

$$2\text{Adv}^{\text{Xymmer}}(\sigma) + \sum_W 2\text{Adv}^{\text{Multimix}}(q_W, \sigma_W)$$

- q_W : number of queries with tweak W
- σ : total data complexity, σ_W : data complexity with tweak

Mystrium vs. Adiantum: Security Claim

Distinguishing advantage bound for the deck-based wide block cipher modes:

$$2\text{Adv}^{\text{Xymmer}}(\sigma) + \sum_W 2\text{Adv}^{\text{Multimix}}(q_W, \sigma_W)$$

$$\text{Adv}^{\text{Xymmer}} = \underbrace{\frac{q^2}{2^{128}}}_{\text{Collision at digest of Multimix}} + \underbrace{\frac{N}{2^{256}}}_{\text{Exhaustive key search}}$$

- short key of 256 bits is used.

- N : amount of computations, number of evaluations of $\mathcal{G}_{512}[6]$

$$\text{Adv}^{\text{Multimix}} = \underbrace{\frac{1}{2^{127}}}_{\text{Adv. of Multimixer}} \binom{q_W}{2}$$

Mystrium vs. Adiantum: Security Claim

Distinguishing advantage bound for the deck-based wide block cipher modes:

$$2\text{Adv}^{\text{Xymmer}}(\sigma) + \sum_W 2\text{Adv}^{\text{Multimix}}(q_W, \sigma_W)$$

- q_W : number of queries with tweak W
- σ : total data complexity, σ_W : data complexity with tweak

$$\begin{aligned}\text{Adv}^{\text{Mystrium}} &= 2 \left(\frac{q^2}{2^{128}} + \frac{N}{2^{256}} \right) + \sum_W 2 \left(\frac{1}{2^{127}} \binom{q_w}{2} \right) \\ &= 2^{-127} (q^2 + \sum_W q_w^2) + \frac{N}{2^{255}}\end{aligned}$$

Mystrium vs. Adiantum: Security Claim – Cont'd

Distinguishing advantage bound quadratic in queries, linear in message/tweak length:

$$\begin{aligned} &= \left(3(2^{-128}) + \underbrace{2^{-103} \cdot \max \left(1 + \left\lfloor \frac{l_T}{128} \right\rfloor, 2 \left\lfloor \frac{(l_M - 128)}{8192} \right\rfloor \right)}_{\epsilon \text{ in } \epsilon\text{-}\Delta\text{universality of Adiantum hashing}} \right) \binom{q}{2} \\ &\quad + \underbrace{\text{Adv}_{S_{K_S}}^{\text{sc}} (q + 1, 256 + 8832 + q(l_M - 128), t')}_{\text{distinguishing advantage of XChaCha12}} + \underbrace{\text{Adv}_{E_{K_E}}^{\pm\text{prp}} (q, t')}_{\text{distinguishing advantage of AES-256}} \end{aligned}$$

- q : number of queries, l_M and l_T maximum lengths of message and tweak

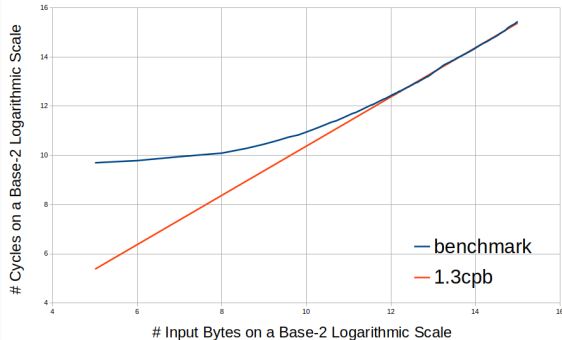
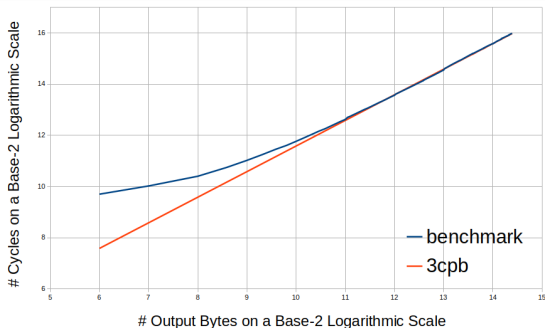
Implementation and Benchmarking

- We wrote optimized code on the 32-bit ARMv7 Cortex-A processors.
 - for Multimixer-128, $\mathcal{G}_{512}$ [6], Xymmer, and Mystrium,
- These processors lack cryptography extensions, including AES instructions
- We used Advanced SIMD instruction set in implementation and design.
- Benchmarking is done on a Raspberry Pi 2 model B single-board computer in cycles per byte (cpbs)

Algorithm	Input length in bytes	
	512	4096
Mystrium including mask derivation	13.121	9.766
Mystrium	9.969	6.765
Adiantum-XChaCha8-AES (ENC) ^a	17.032	9.652
Adiantum-XChaCha12-AES (ENC) ^a	19.401	11.559
Adiantum-XChaCha20-AES (ENC) ^a	24.862	15.871

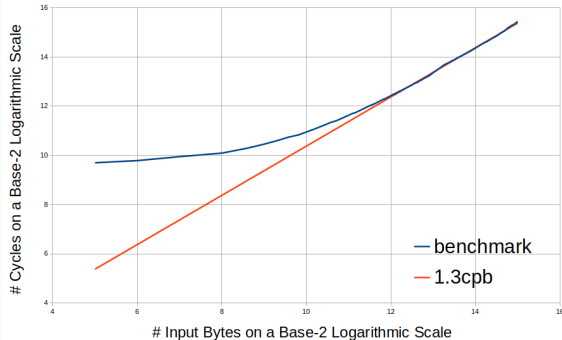
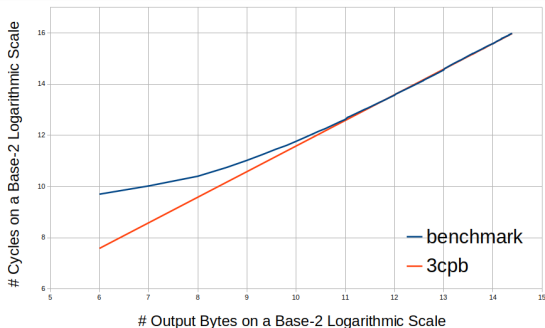
Xymmer Performance

- Deck functions can also be used in modes to build
 - encryption, authenticated encryption, and authentication schemes
- So, Xymmer is of independent interest.



Xymmer Performance

- Deck functions can also be used in modes to build
 - encryption, authenticated encryption, and authentication schemes
- So, Xymmer is of independent interest.



Thank you for your attention!

References



Joan Daemen and Vincent Rijmen. “AES proposal: Rijndael”. In: (1999).



T Krovetz. “Software-Optimized Universal Hashing and Message Authentication. UC Davis Ph. D”. PhD thesis. Dissertation, 2000.



Daniel J Bernstein. “The Poly1305-AES message-authentication code”. In: *International workshop on fast software encryption*. Springer. 2005, pp. 32–49.



Kazuhiko Minematsu and Toshiyasu Matsushima. “Tweakable enciphering schemes from hash-sum-expansion”. In: *Progress in Cryptology–INDOCRYPT 2007: 8th International Conference on Cryptology in India, Chennai, India, December 9-13, 2007. Proceedings 8*. Springer. 2007, pp. 252–267.



Daniel J Bernstein et al. “ChaCha, a variant of Salsa20”. In: *Workshop record of SASC*. Vol. 8. 1. Citeseer. 2008, pp. 3–5.



Debrup Chakraborty and Palash Sarkar. “HCH: A new tweakable enciphering scheme using the hash-counter-hash approach”. In: *IEEE Transactions on Information Theory* 54.4 (2008), pp. 1683–1699.



Mridul Nandi. “Improving upon HCTR and matching attacks for Hash-Counter-Hash approach”. In: *Cryptology ePrint Archive* (2008).



Guido Bertoni et al. “Farfalle: parallel permutation-based cryptography”. In: *Cryptology ePrint Archive* (2016).



Paul Crowley and Eric Biggers. “{Adiantum}: length-preserving encryption for entry-level processors”. In: *Cryptology ePrint Archive* (2018).



Joan Daemen et al. “The design of Xoodoo and Xoofff”. In: *IACR Transactions on Symmetric Cryptology* (2018), pp. 1–38.



Yoav Nir and Adam Langley. *ChaCha20 and Poly1305 for IETF Protocols*. Tech. rep. 2018.



Norica Băcuieți et al. *Jammin' on the deck*. Presentation at Asiacrypt 2022. International Conference on the Theory and Application of Cryptology and Information Security. 2022.



Jonathan Fuchs, Yann Rotella, and Joan Daemen. “On the security of keyed hashing based on public permutations”. In: *Annual International Cryptology Conference*. Springer. 2023, pp. 607–627.



Koustabh Ghosh, Parisa Amiri Eliasi, and Joan Daemen. “Multimixer-128: universal keyed hashing based on integer multiplication”. In: *IACR Transactions on Symmetric Cryptology* 2023.3 (2023), pp. 1–24.



Koustabh Ghosh et al. “Universal hashing based on field multiplication and (near-) MDS matrices”. In: *International Conference on Cryptology in Africa*. Springer. 2023, pp. 129–150.



Charlotte Lefevre, Yanis Belkheyar, and Joan Daemen. “Kirby: A Robust Permutation-Based PRF Construction”. In: *Cryptology ePrint Archive* (2023).