MQOM: MQ on my Mind — Version 2 —

Ryad Benadjila, Charles Bouillaguet
Thibauld Feneuil, Matthieu Rivain

Sixth PQC Standardization Conference

September 25, 2025, *NIST*

CryptoExperts
WE INNOVATE TO SECURE YOUR BUSINESS
SORBONNE UNIVERSITÉ

Table of Contents

Round-2 Design Updates

MQOM v2 - Design Rationale

- **Conservative security**: signature scheme for which the security relies on the hardness of solving ***fully random unstructured*** instances of the multivariate quadratic (MQ) problem.

    MQ Problem: Given a random multivariate quadratic system $(\mathcal{F}, y)$, find x such that $\mathcal{F}(x) = y$.

- **Design Choice**: Signature scheme built upon the ***MPC-in-the-Head*** paradigm, which provides a generic way to build a secure scheme from a hard problem.

MQOM v2 - Design Rationale

- **MQ Parameters**:

  - The hardest instances:

    the number of variables = the number of equations

  - choice of the MQ field

    - MQOM v1: $GF(31)$ and $F(251)$
    - MQOM v2: $GF(2)$, $G(16)$ and $GF(256)$

      The field $GF(16)$ has been added in the version 2.1

🔎 *Motivation to choose binary fields*:

Avoid rejection sampling and arithmetic-Boolean conversation.

MQOM v2 - Design Rationale

- **MPC-in-the-Head paradigm**: possible existing frameworks
  - MQOM v1: based on the framework using linear broadcast-based MPC
  - MQOM v2: ***two new MPCitH frameworks*** since the previous NIST deadline:

VOLE-in-the-Head (summer 2023) and **TC-in-the-Head** (fall 2023)

[BBD+23] Baum, Braun, Delpech, Klooß, Orsini, Roy, Scholl. Publicly Verifiable Zero-Knowledge and Post-Quantum Signatures From VOLE-in-the-Head. Crypto 2023.

[FR25] Feneuil, Rivain. *Threshold Computation in the Head: Improved Framework for Post-Quantum Signatures and Zero-Knowledge Arguments*. Journal of Cryptology, 2025.

MQOM v2 - Design Rationale

Instance	*Trade-off*	*Batching*	*TCitH*	*VOLEitH*
L1 - MQ over GF(2)	Short	✔	2 820 B	2 790 B
		✘	2 868 B	2 966 B
L1 - MQ over GF(16)	Short	✔	2 916 B	2 878 B
		✘	3 060 B	3 054 B
L5 - MQ over GF(2)	Short	✔	11 564 B	11 434 B
		✘	11 764 B	12 170 B
L5 - MQ over GF(16)	Short	✔	12 014 B	11 848 B
		✘	12 664 B	12 584 B

When targeting small signature sizes:

TCitH vs VOLEitH, for MQ: Less than 3% of difference!

Trade-off definition:
- « Short » uses trees of 2048 leaves,
- « Fast » uses trees of 256 leaves.

Tree Optimisation: Correlated trees

See MQOM's specifications for sizes with the one-tree optimization (using TCitH and VOLEitH)

MQOM v2 - Design Rationale

Instance	*Trade-off*	*Batching*	*TCitH*	*VOLEitH*
L1 - MQ over GF(2)	Fast	✔	3 144 B	3 054 B
		✘	3 212 B	3 294 B
L1 - MQ over GF(16)	Fast	✔	3 280 B	3 174 B
		✘	3 484 B	3 414 B
L5 - MQ over GF(2)	Fast	✔	13 124 B	12 378 B
		✘	13 412 B	13 370 B
L5 - MQ over GF(16)	Fast	✔	13 772 B	12 936 B
		✘	14 708 B	13 928 B

When targeting fast schemes:

TCitH vs VOLEitH, for MQ: Less than 6% of difference!

Trade-off definition:
- « Short » uses trees of 2048 leaves,
- « Fast » uses trees of 256 leaves.

Tree Optimisation: Correlated trees
See MQOM's specifications for sizes with the one-tree optimization (using TCitH and VOLEitH)

MQOM v2 - Design Rationale

- **<u>MPC-in-the-Head paradigm</u>**: possible existing frameworks
  - MQOM v1: based on the framework using linear broadcast-based MPC
  - MQOM v2: ***two new MPCitH frameworks*** since the previous NIST deadline:

VOLE-in-the-Head (summer 2023) and **TC-in-the-Head** (fall 2023)

- **Exponentially large fields:** e.g. $GF(2^{128})$ for L1
- **Only one protocol execution** (over the large field)
- **Based on 7-round protocol (or 5-round protocol)**
  - Rely on a **consistency check**
- Better signature sizes**: not true for MQ**

- **Small fields:** typically $GF(256), GF(256^2), \ldots$
- **Several parallel repetitions** (over the small field)
- **Based on 5-round protocol (or 3-round protocol)**

MQOM v2 - Design Rationale

- **MPC-in-the-Head paradigm**: possible existing frameworks
  - MQOM v1: based on the framework using linear broadcast-based MPC
  - MQOM v2: ***two new MPCitH frameworks*** since the previous NIST deadline:

VOLE-in-the-Head (summer 2023) and **TC-in-the-Head** (fall 2023)

VOLE-in-the-Head

- **Exponentially large fields:** e.g. $GF(2^{128})$ for L1
- **Only one protocol execution** (over the large field)
- **Based on 7-round protocol (or 5-round protocol)**
  - Rely on a **consistency check**
- Better signature sizes**: not true for MQ**

TC-in-the-Head

- **Small fields:** typically $GF(256), GF(256^2), \ldots$
- **Several parallel repetitions** (over the small field)
- **Based on 5-round protocol (or 3-round protocol)**

MQOM v2

MQOM v2 - Design Rationale

- **<u>Formalism</u>**:
  - TCitH: sharing-based formalism
  - VOLEitH: VOLE-based formalism
  - MQOM v2: PIOP-based formalism (polynomial-based formalism)

🔎 *<u>Motivation to choose the PIOP formalism</u>:*

Simpler description of the scheme, that does not depend on MPC technology. Easier-to-understand scheme for those who do not already know those two frameworks.

[Fen24] Feneuil. *The Polynomial-IOP Vision of the Latest MPCitH Framework for Signature Schemes*. PQ Algebraic Cryptography Workshop, IHP 2024.

MQOM v2 - Underlying 3-Round Identification Scheme

Public Key: $(A_j, b_j, y_j)_{j=1..m}$

Secret Key: x such that $x^\top A_j x + b_j^\top x = y_j$ for all j

① Sample n random degree-1 polynomials $P_1, \ldots, P_n$ such that the degree-1 terms are $x_1, \ldots, x_n$. Sample m random degree-1 polynomials $M_1, \ldots, M_m$.

② Commit to those polynomials.

MQOM v2 - Underlying 3-Round Identification Scheme

Public Key: $(A_j, b_j, y_j)_{j=1..m}$

Secret Key: x such that $x^\top A_j x + b_j^\top x = y_j$ for all j

① Sample n random degree-1 polynomials $P_1, \ldots, P_n$ such that the degree-1 terms are $x_1, \ldots, x_n$.
Sample m random degree-1 polynomials $M_1, \ldots, M_m$.

② Commit to those polynomials.

③ Send the polynomials $Q_1, \ldots, Q_m$ (of degree at most 1) defined as

$$\begin{aligned} Q_1(X) &:= M_1(X) + \overrightarrow{P}(X)^\top \cdot A_1 \cdot \overrightarrow{P}(X) + b_1^\top \cdot \overrightarrow{P}(X) \cdot X - y_1 \cdot X^2 \\ &\vdots \\ Q_m(X) &:= M_m(X) + \overrightarrow{P}(X)^\top \cdot A_m \cdot \overrightarrow{P}(X) + b_m^\top \cdot \overrightarrow{P}(X) \cdot X - y_m \cdot X^2 \end{aligned}$$

where $\overrightarrow{P} = (P_1, \ldots, P_n)$.

MQOM v2 - Underlying 3-Round Identification Scheme

Public Key: $(A_j, b_j, y_j)_{j=1..m}$

Secret Key: x such that $x^\top A_j x + b_j^\top x = y_j$ for all j

① Sample n random degree-1 polynomials $P_1, \ldots, P_n$ such that the degree-1 terms are $x_1, \ldots, x_n$.
Sample m random degree-1 polynomials $M_1, \ldots, M_m$.

② Commit to those polynomials.

③ Send the polynomials $Q_1, \ldots, Q_m$ (of degree at most 1) defined as

$$Q_1(X) := M_1(X) + \overrightarrow{P}(X)^\top \cdot A_1 \cdot \overrightarrow{P}(X) + b_1^\top \cdot \overrightarrow{P}(X) \cdot X - y_1 \cdot X^2$$
$$\vdots$$
$$Q_m(X) := M_m(X) + \overrightarrow{P}(X)^\top \cdot A_m \cdot \overrightarrow{P}(X) + b_m^\top \cdot \overrightarrow{P}(X) \cdot X - y_m \cdot X^2$$

where $\overrightarrow{P} = (P_1, \ldots, P_n)$.

④ Get a random evaluation point $r \in \mathscr{C}$ from the verifier.

⑤ Reveal the evaluations $P_1(r), \ldots, P_n(r)$ and $M_1(r), \ldots, M_m(r)$.

MQOM v2 - Design Rationale

- **Miscellaneous**:
  - Tree optimisation: correlated-tree technique
  - PRG: based on AES-128 and Rijndael-256, instead of SHAKE
  - Seed Commitment: based on AES-128 and Rijndael-256, instead of SHAKE
  - Folding: using Gray code (version 2.1)
  - Sigma variant (*i.e.* FS transform of a 3-round protocol), with very small communication penalty
  - Computation Speed-Up using Matrix Packing (version 2.1)

👉 *See MQOM's specifications for details.*

Round-2 Performance Updates

The Field Arithmetic in MQOM

- **MQ Field**:
  - $GF(2)$ - Shortest signature
  - $GF(16)$ - Short signature and efficient signing (v2.1)
  - $GF(256)$ - Efficient signing

- **TCitH Field**:
  - $GF(256)$ - Trade-off « Fast »
  - $GF(256^2)$ - Trade-off « Short »

(For all security levels)

The main arithmetic in MQOM v2 : $GF(256)$

The Field Arithmetic in MQOM

- **MQ Field**:
  - $GF(2)$ - Shortest signature
  - $GF(16)$ - Short signature and efficient signing (v2.1)
  - $GF(256)$ - Efficient signing

- **TCitH Field**:
  - $GF(256)$ - Trade-off « Fast »
  - $GF(256^2)$ - Trade-off « Short »

(For all security levels)

The main arithmetic in MQOM v2 : $GF(256)$

Several possible optimized implementation strategies:

- Recent AVX-based CPU: can be highly sped up with **GFNI**
- **Vectorized** (e.g. with SIMD techniques)
- **Bitsliced** (*e.g.* across the parallel repetitions)
- **Look-up Table of 65 kB**
- Small **Look-up Tables Log/Exp**

Performance on AVX-based CPU

We propose three optimized implementations for AVX-based CPU (v2.1):

- Using only AVX2 and AES-NI
- Using AVX2, AES-NI and GFNI
- Using AVX-512, AES-NI and GFNI

Performance on AVX-based CPU

We propose three optimized implementations for AVX-based CPU (v2.1):

- Using only AVX2 and AES-NI
- Using AVX2, AES-NI and GFNI
- Using AVX-512, AES-NI and GFNI

About GFNI and AVX-512 support across x86 platforms:

		GFNI	***AVX512***
Intel	Laptop	Most, since Q1 2022	None
	Server	Most, since Q1 2022	Most
AMD	Laptop	All, since Q2 2024	All, since Q2 2024
	Server	All, since Q2 2024	All, since Q2 2024

See MQOM's specifications for details.

Homogeneous GFNI support across x86 is a reality in 2025

Performance on AVX-based CPU

We propose three optimized implementations for AVX-based CPU (v2.1):

- Using only AVX2 and AES-NI
- **Using AVX2, AES-NI and GFNI**
- Using AVX-512, AES-NI and GFNI

MQOMv2 Instance			*PK Size*	*Sizes (R3)*	*Sizes (R5)*	*Sig. / Verif. Running times*
NIST I	GF(2)	Short	52 B	2 868 B	2 820 B	≈ 6.3 Mcycles
		Fast		3 212 B	3 144 B	≈ 3.5 Mcycles
	GF(16)	Short	80 B	3 060 B	2 916 B	≈ 5.3 Mcycles
		Fast		3 484 B	3 280 B	≈ 2.0 Mcycles
NIST V	GF(2)	Short	104 B	11 764 B	11 564 B	≈ 51 Mcycles
		Fast		13 412 B	13 124 B	≈ 28 Mcycles
	GF(16)	Short	160 B	12 664 B	12 014 B	≈ 38 Mcycles
		Fast		14 708 B	13 772 B	≈ 13 Mcycles

GF(256) has similar running times than GF(16), but leads to larger signatures.
See MQOM's specifications for complete benchmarks, along with the experimental setup.

Performance on embedded devices

We propose three optimized implementations for Cortex-M4 (v2.1):

- Using tables log/exp and T-table AES-Rijndael
- Using table 65 kB and T-table AES-Rijndael

*Constant -time only when there is no cache system for SRAM (**true** for many classical boards)*

- Using vectorized field multiplication and bitsliced AES (only for L1)

The **used memory optimizations** are mainly

- On-the-fly GGM trees
- Incremental hash operations
- On-the-fly matrix expansion-multiplication

Performance on embedded devices

We propose three optimized implementations for Cortex-M4 (v2.1):

- **Using tables log/exp and T-table AES-Rijndael**
- Using table 65 kB and T-table AES-Rijndael
- Using vectorized field multiplication and bitsliced AES (only for L1)

*Constant -time only when there is no cache system for SRAM (**true** for many classical boards)*

MQOMv2 Instance			***Size R5***	***Sig. / Verif. Running times***	***Sig. / Verif. Mem. Usage***
NIST I	GF(2)	Short	2 820 B	*≈ 312 Mcycles*	**≈ 18 KB**
		Fast	3 144 B	*≈ 150 Mcycles*	**≈ 16 KB**
	GF(16)	Short	2 916 B	**≈ 221 Mcycles**	**≈ 14 KB**
		Fast	3 280 B	**≈ 73 Mcycles**	**≈ 14 KB**
NIST V	GF(2)	Short	11 564 B	*≈ 3550 Mcycles*	**≈ 48 KB**
		Fast	13 124 B	*≈ 2122 Mcycles*	**≈ 42 KB**
	GF(16)	Short	12 014 B	**≈ 1971 Mcycles**	**≈ 28 KB**
		Fast	13 772 B	**≈ 684 Mcycles**	**≈ 29 KB**

GF(256) has similar running times than GF(16), but leads to larger signatures.
See MQOM's specifications for complete benchmarks, along with the experimental setup.
Benchmark performed on STM Nucleo-L4R5ZI board with a STM32 Cortex-M4 MCU

Performance on embedded devices

We propose three optimized implementations for Cortex-M4 (v2.1):

- Using tables log/exp and T-table AES-Rijndael
- Using table 65 kB and T-table AES-Rijndael

*Constant -time only when there is no cache system for SRAM (**true** for many classical boards)*

- **Using vectorized field multiplication and bitsliced AES (only for L1)**

MQOMv2 Instance			***Size R5***	***Sig. / Verif. Running times***	***Sig. / Verif. Mem. Usage***
NIST I	GF(2)	Short	2 820 B	*≈ 860 Mcycles*	**≈ 13 KB**
		Fast	3 144 B	*≈ 318 Mcycles*	**≈ 11 KB**
	GF(16)	Short	2 916 B	≈ 656 Mcycles	**≈ 9 KB**
		Fast	3 280 B	≈ 168 Mcycles	**≈ 9 KB**

See MQOM's specifications for complete benchmarks, along with the experimental setup.

Performance on embedded devices

We propose three optimized implementations for Cortex-M4 (v2.1):

- Using tables log/exp and T-table AES-Rijndael
- Using table 65 kB and T-table AES-Rijndael

*Constant -time only when there is no cache system for SRAM (**true** for many classical boards)*

- **Using vectorized field multiplication and bitsliced AES (only for L1)**

MQOMv2 Instance			***Size R5***	***Sig. / Verif. Running times***	***Sig. / Verif. Mem. Usage***
NIST I	GF(2)	Short	2 820 B	*≈ 860 Mcycles*	**≈ 13 KB**
		Fast	3 144 B	*≈ 318 Mcycles*	**≈ 11 KB**
	GF(16)	Short	2 916 B	≈ 656 Mcycles	**≈ 9 KB**
		Fast	3 280 B	≈ 168 Mcycles	**≈ 9 KB**

See MQOM's specifications for complete benchmarks, along with the experimental setup.

Additional optimizations are still possible:

- Bitsliced field multiplication
- Refining the memory usage
- …

Moreover, **alternative trade-offs between speed and memory usage** are possible.

Performance on embedded devices

Bonus: when using a *AES hardware accelerator*

MQOMv2 Instance			*Size R5*	*Sig. / Verif. Running times*	*Sig. / Verif. Mem. Usage*
NIST I	GF(2)	Short	2 820 B	≈ *320 Mcycles*	≈ 12 KB
		Fast	3 144 B	≈ *155 Mcycles*	≈ 10 KB
	GF(16)	Short	2 916 B	≈ 180 Mcycles	≈ 8 KB
		Fast	3 280 B	≈ 70 Mcycles	≈ 8 KB

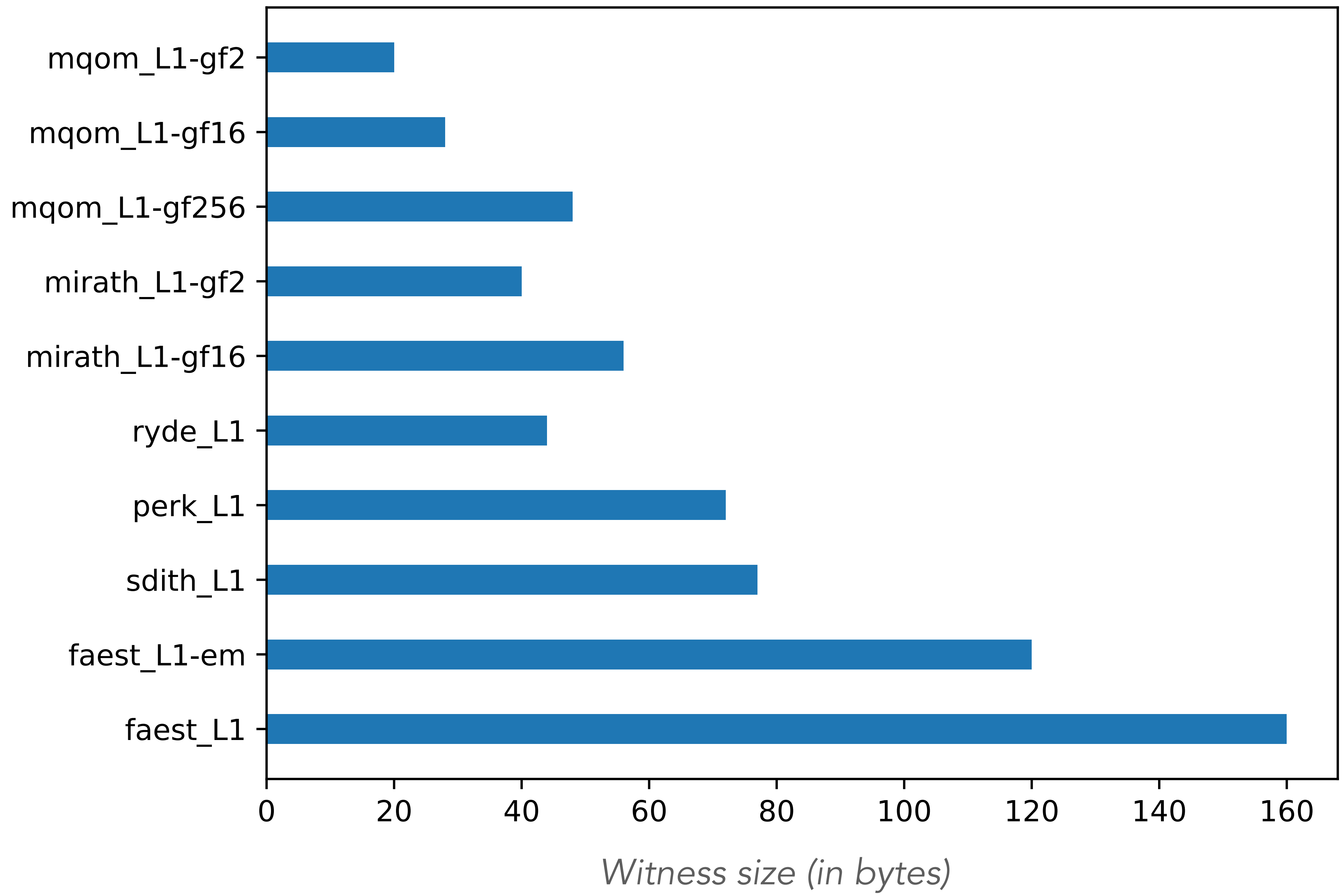
See MQOM's specifications for complete benchmarks, along with the experimental setup.

Comparison to other MPCitH-based schemes

Comparison with other MPCitH-based schemes

Comparison with FAEST, Mirath, PERK, RYDE and SDitH

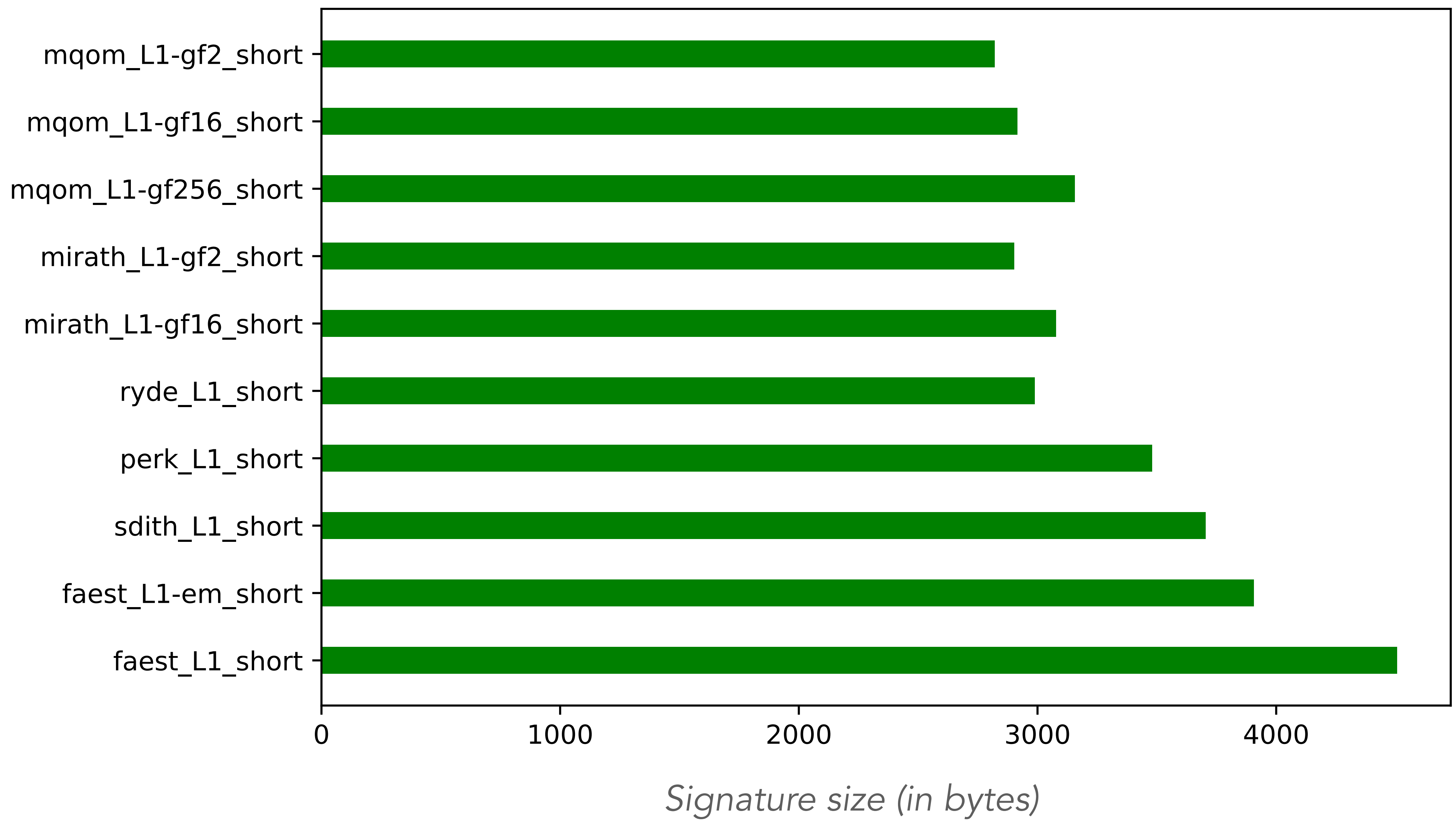
🧐 **Witness size: an important metric!**



Comparison with other MPCitH-based schemes

Comparison with FAEST, Mirath, PERK, RYDE and SDitH

Witness size: scale the signature size

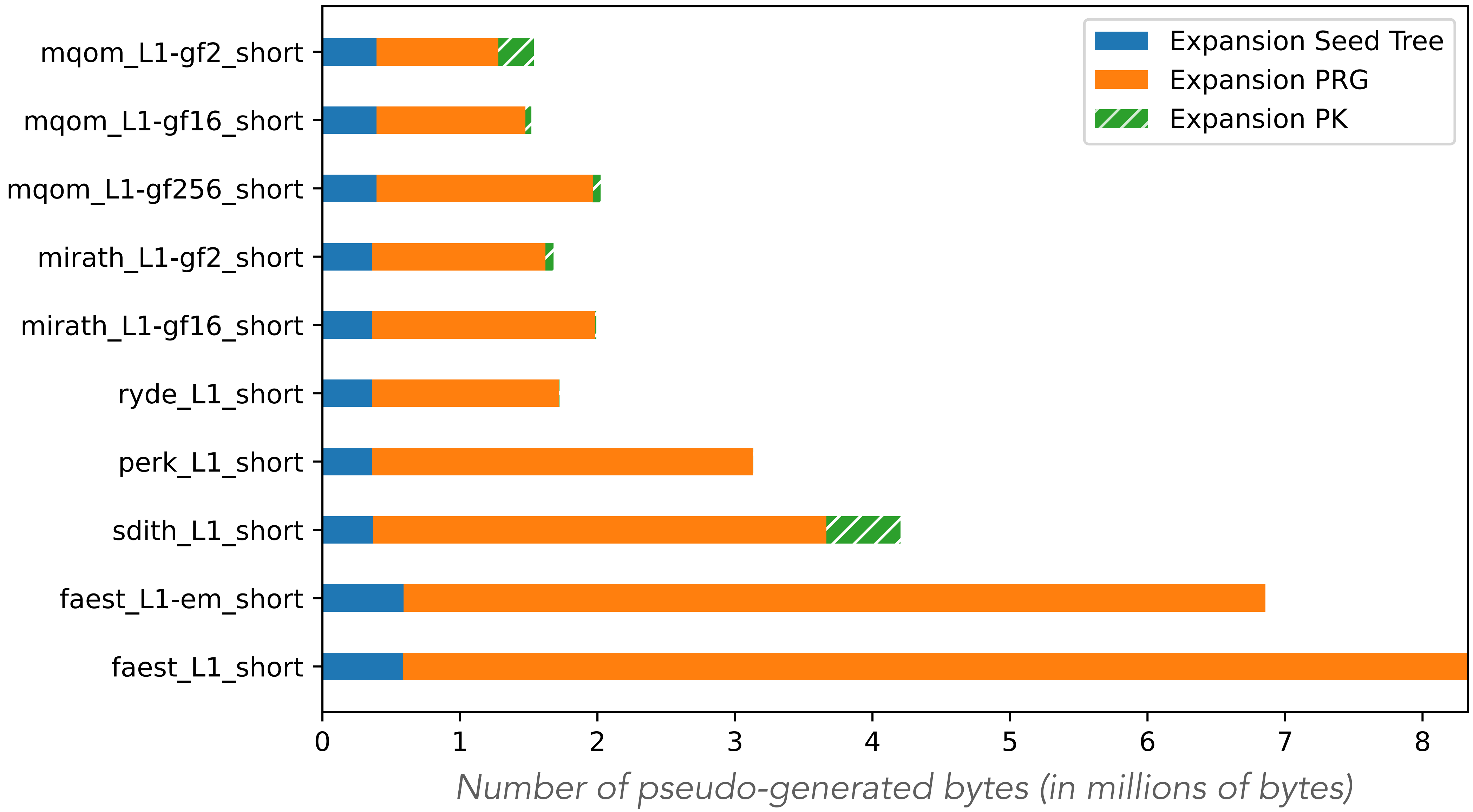


Note: the above sizes are produced using the statistical batching.

Comparison with other MPCitH-based schemes

Comparison with FAEST, Mirath, PERK, RYDE and SDitH

Witness size: scale the usage of symmetric primitives

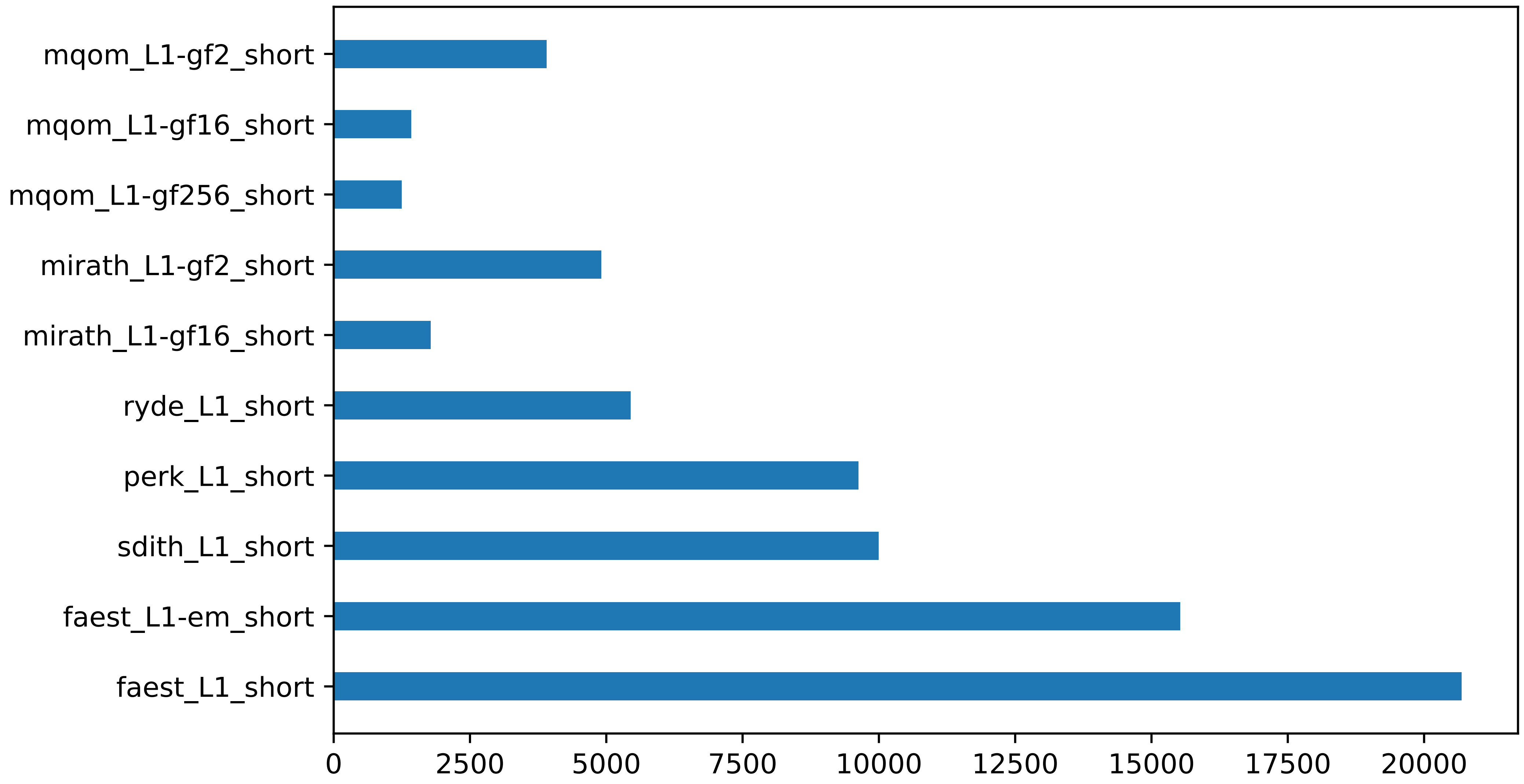


Note: the above figure does not take in account the saving due to half/correlated trees.

Comparison with other MPCitH-based schemes

Comparison with FAEST, Mirath, PERK, RYDE and SDitH

Witness size: scale the memory footprint



Memory Footprint (in bytes)

of the committed polynomials $P_1, \ldots, P_n$ *and* $M_1, \ldots, M_m$

Comparison with other MPCitH-based schemes

Comparison with FAEST, Mirath, PERK, RYDE and SDitH

- **Advantages**:
    - Unstructured MQ is a very old problem
    - MQOM is the MPCitH-based scheme with the smallest witness
        - Among the smallest signature sizes
        - Scheme that makes the lowest use of symmetric primitives
        - Scheme that has natively the smallest memory footprint

Comparison with other MPCitH-based schemes

Comparison with FAEST, Mirath, PERK, RYDE and SDitH

- **Advantages**:
  - Unstructured MQ is a very old problem
  - MQOM is the MPCitH-based scheme with the smallest witness
    - Among the smallest signature sizes
    - Scheme that makes the lowest use of symmetric primitives
    - Scheme that has natively the smallest memory footprint

MQOM may be considered one of the most embedded-friendly options among MPCitH candidates.

Comparison with other MPCitH-based schemes

Comparison with FAEST, Mirath, PERK, RYDE and SDitH

- **Advantages**:
  - Unstructured MQ is a very old problem
  - MQOM is the MPCitH-based scheme with the smallest witness
    - Among the smallest signature sizes
    - Scheme that makes the lowest use of symmetric primitives
    - Scheme that has natively the smallest memory footprint
  - Simplicity :
    - No need to arithmetize the hard problem
    - Rely on TCitH (no need to have some consistency check)
    - Have a sigma variant (R3), with very small penalty in signature size

- **Limitations**:
  - Large expanded public key - Can be mitigated by on-the-fly expansion without much computational penalty (*c.f.* MQOM's benchmarks in embedded devices)
  - Large numbers of multiplications over $GF(256)$: can be sped up using GFNI or generic SIMD, for example.