SoK: Efficient Design and Implementation of Polynomial Hash Functions over Prime Fields

Jean Paul Degabriele[1], Jan Gilcher[2], Jérôme Govinden[3] & Kenneth G. Paterson[2]
[1] Technology Innovation Institute Cryptography Research Center, Abu Dhabi, United Arab Emirates
[2] Applied Cryptography Group, ETH Zurich, Switzerland
[3] CNS, Technische Universität Darmstadt, Germany

Abstract

Poly1305 is a widely-deployed polynomial hash function. The rationale behind its design was laid out in a series of papers by Bernstein, the last of which dates back to 2005. As computer architectures evolved, some of its design features became less relevant, but implementers found new ways of exploiting these features to boost its performance. However, would we still converge to this same design if we started afresh with today's computer architectures and applications? To answer this question, we gather and systematize a body of knowledge concerning polynomial hash design and implementation that is spread across research papers, cryptographic libraries, and developers' blogs. We develop a framework to automate the validation and benchmarking of the ideas that we collect. This approach leads us to five new candidate designs for polynomial hash functions. Using our framework, we generate and evaluate different implementations and optimization strategies for each candidate. We obtain substantial improvements over Poly1305 in terms of security and performance. Besides laying out the rationale behind our new designs, our paper serves as a reference for efficiently implementing polynomial hash functions, including Poly1305.

Motivation: Improve Poly1305

For $M = M_1 \| \cdots \| M_n$,

$$\mathsf{Poly1305}(r, M) = (c_1 x^n + c_2 x^{n-1} + \cdots + c_n x^1 \mod 2^{130} - 5) \mod 2^{128},$$

where $c_i = M_i \| 1$ and $x = \mathsf{clamp}(r, 22)$.

Key Points of Poly1305:

- Widely deployed, default choice in OpenSSH and WireGuard
- Good performance across all architectures without specific hardware support
- Clamping weakens security without adequate payback in performance
- Tailored for 32-bit architectures and waste limb space on 64-bit ones
- Limited security of Chacha20-Poly1305 in the multi-user setting due to Poly1305

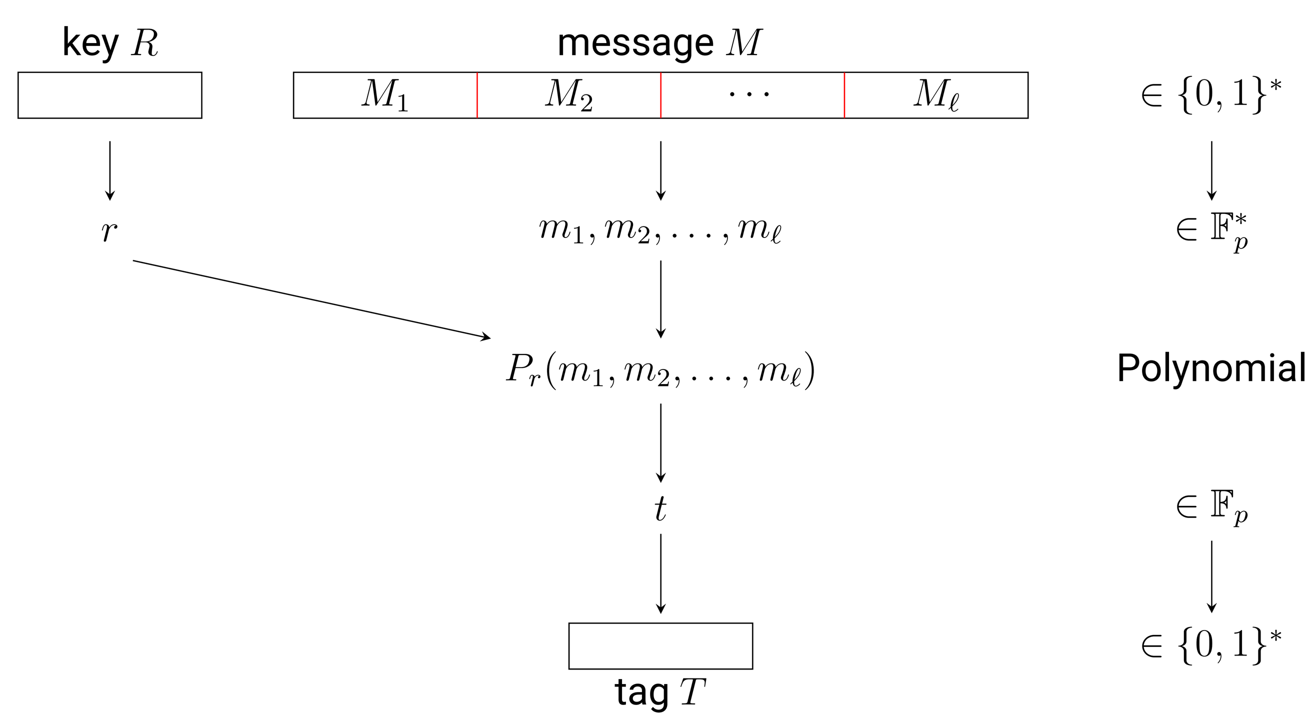
Security Provided: Δ-Universality

Given a tag $z \in \mathcal{T}$ and two distinct messages $M \neq M' \in \mathcal{M}$,

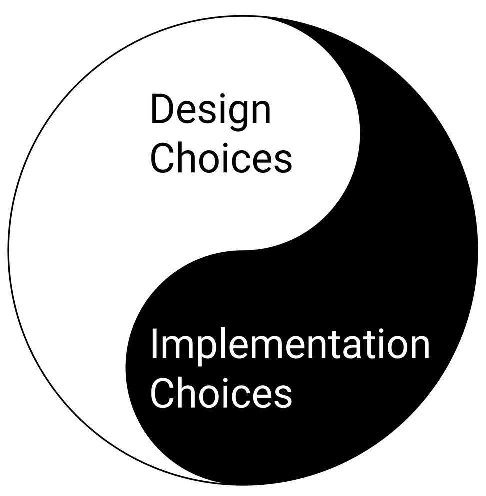
$$\Pr_{r \leftarrow_\$ \mathcal{R}}\left[H_r(M) - H_r(M') = z\right] \leq \epsilon(M, M').$$

Systematization of Knowledge (SoK)

Description of the Design Space for a Polynomial Hash

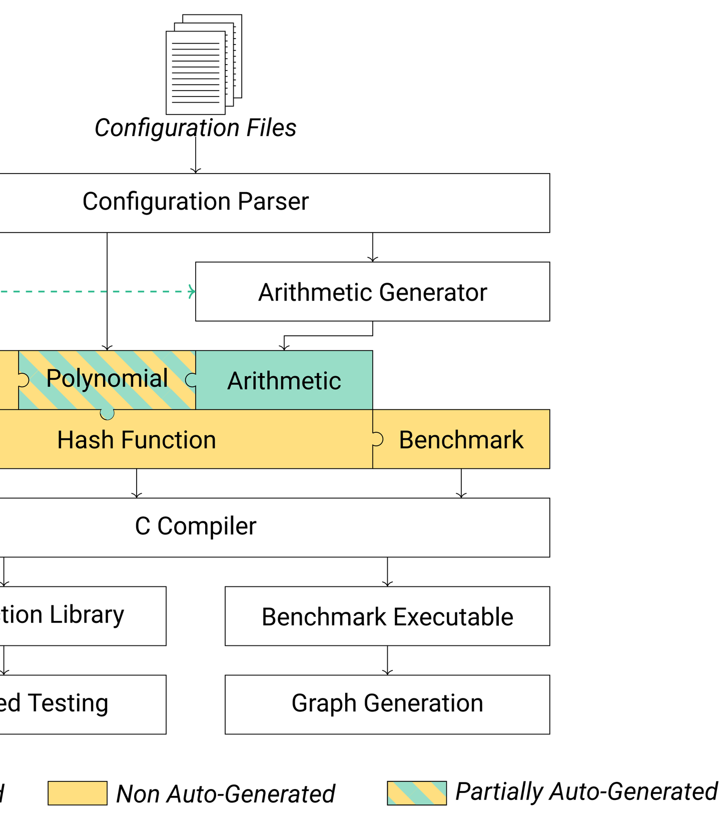


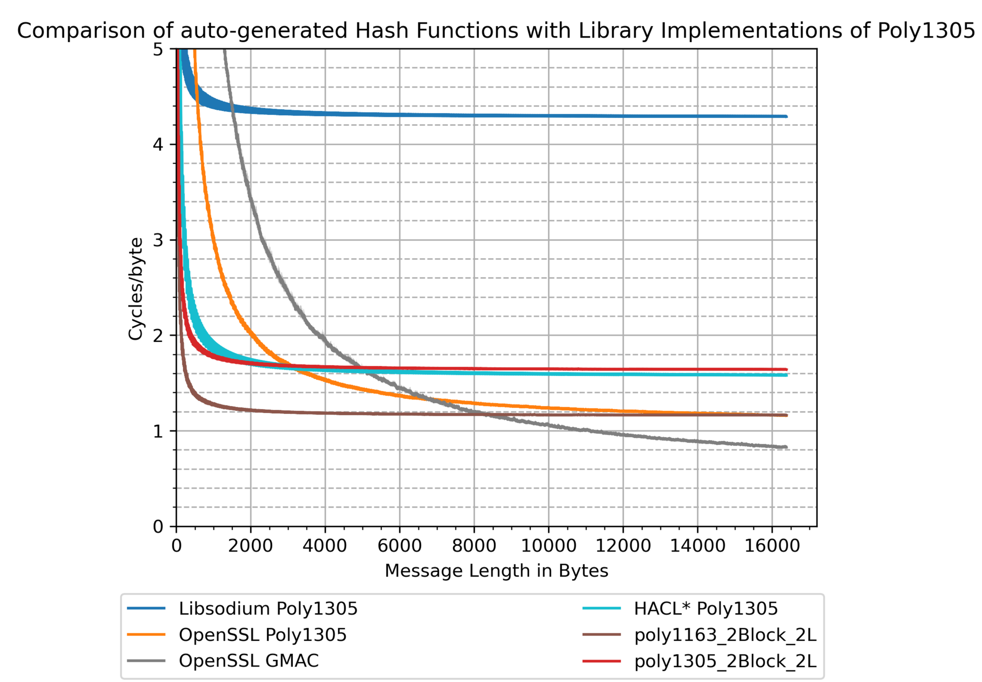
Our Exposition:



- **Choice of Polynomial Construction:**
  - Classical Polynomial
  - Polynomial Combinations
- **Choice of Encodings:**
  - Field-to-Tag Encoding
  - Key-to-Field Encoding
  - Data-to-Field Encoding
- **Choice of Finite Fields:**
  - Format of p: $p = 2^\pi - 1$ (Mersenne), $p = 2^\pi - \theta$ (Crandall), $p = 2^\pi + \theta$ or $p = 2^{m \cdot d} - \sum_{i=0}^{d-1} c_i 2^{i \cdot m}$ (Solinas)
  - Size of p

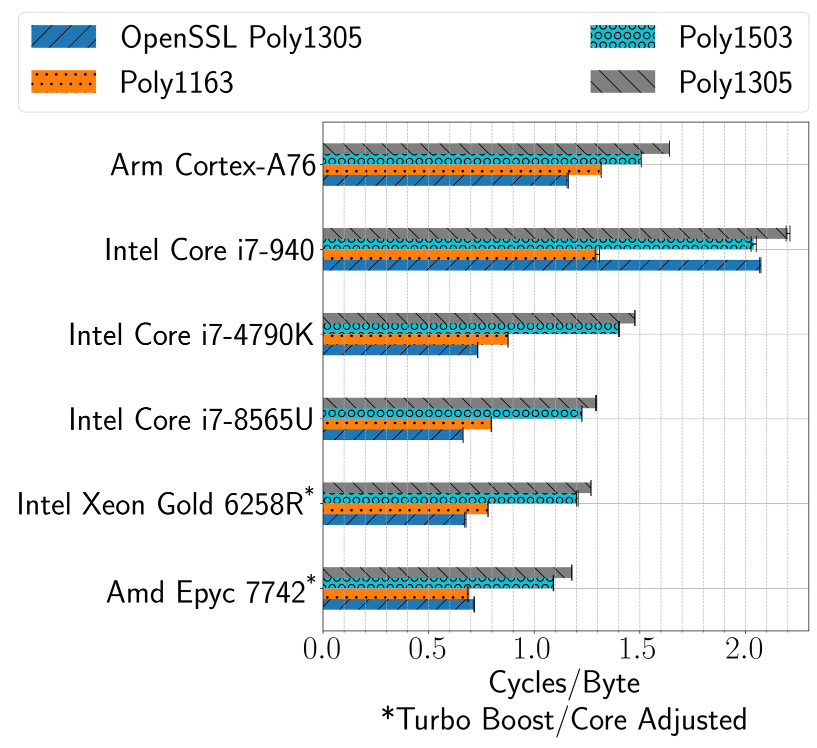
Modular Benchmarking Framework





Code of Framework:

New Designs



Results:

- Our modular implementations achieve **high performance without vectorization or hand-optimization**
- Poly1163 performance makes it **suitable as drop-in replacement for Poly1305**

Our Expectations for Vectorization:

- Poly1163: Significantly outperforms Poly1305 at the same security level
- Poly1503: Replacement for Poly1305 with 34 bits of extra security ($103 \rightarrow 137$) at similar performance