

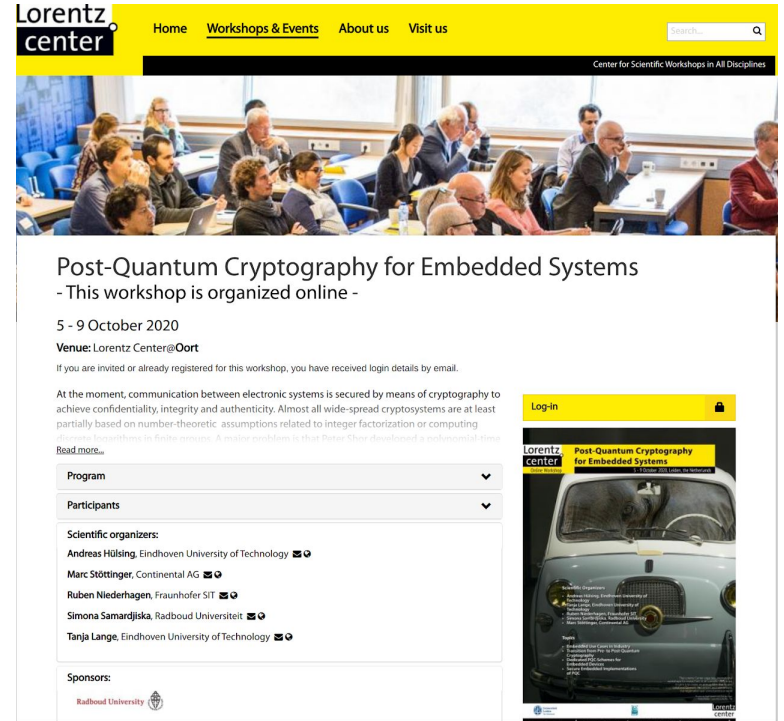
Verifying Post-Quantum Signatures in 8 kB of RAM

Ruben Gonzalez¹, Andreas Hülsing², Matthias J. Kannwischer³, Juliane Krämer⁴, Tanja Lange², Marc Stöttinger⁵, Elisabeth Waitz⁶, Thom Wiggers⁷,
and Bo-Yin Yang⁸

Third PQC Standardization Conference, 07. - 09. June 2021
Presenter: Ruben Gonzalez

Background

- We're at Round 3
 - Let's look at the real world
- PQC for Embedded Systems Workshop
 - Bringing together industry and academia



The screenshot shows the Lorentz Center website for the "Post-Quantum Cryptography for Embedded Systems" workshop. The page features a yellow header with the Lorentz Center logo and navigation links: Home, Workshops & Events, About us, and Visit us. A search bar is located in the top right corner. Below the header is a large photograph of a group of people in a meeting room. The main content area includes the workshop title, dates (5-9 October 2020), and venue (Lorentz Center@Oort). A paragraph of text describes the workshop's focus on post-quantum cryptography. To the right, there is a "Log-in" button and a thumbnail image of a classic car. Below the main text, there are dropdown menus for "Program" and "Participants". The "Scientific organizers" section lists: Andreas Hülsing (Eindhoven University of Technology), Marc Stöttinger (Continental AG), Ruben Niederhagen (Fraunhofer SIT), Simona Samardjiska (Radboud Universiteit), and Tanja Lange (Eindhoven University of Technology). The "Sponsors" section includes Radboud University.

Lorentz center Home Workshops & Events About us Visit us Search

Center for Scientific Workshops in All Disciplines

Post-Quantum Cryptography for Embedded Systems
- This workshop is organized online -

5 - 9 October 2020
Venue: Lorentz Center@Oort

If you are invited or already registered for this workshop, you have received login details by email.

At the moment, communication between electronic systems is secured by means of cryptography to achieve confidentiality, integrity and authenticity. Almost all wide-spread cryptosystems are at least partially based on number-theoretic assumptions related to integer factorization or computing discrete logarithms to finite groups. A major problem is that Peter Shor developed a polynomial time algorithm for integer factorization on a quantum computer.

Read more...

Program

Participants

Scientific organizers:

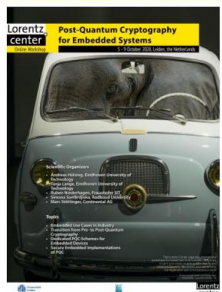
- Andreas Hülsing, Eindhoven University of Technology
- Marc Stöttinger, Continental AG
- Ruben Niederhagen, Fraunhofer SIT
- Simona Samardjiska, Radboud Universiteit
- Tanja Lange, Eindhoven University of Technology

Sponsors:

Radboud University

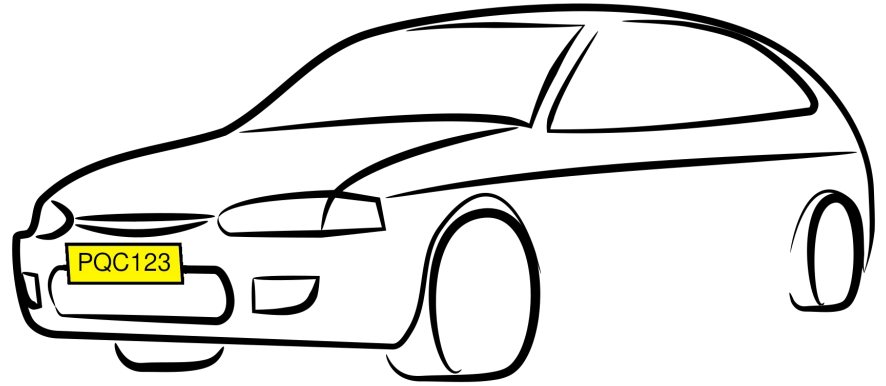
Log-in

Lorentz center Post-Quantum Cryptography for Embedded Systems



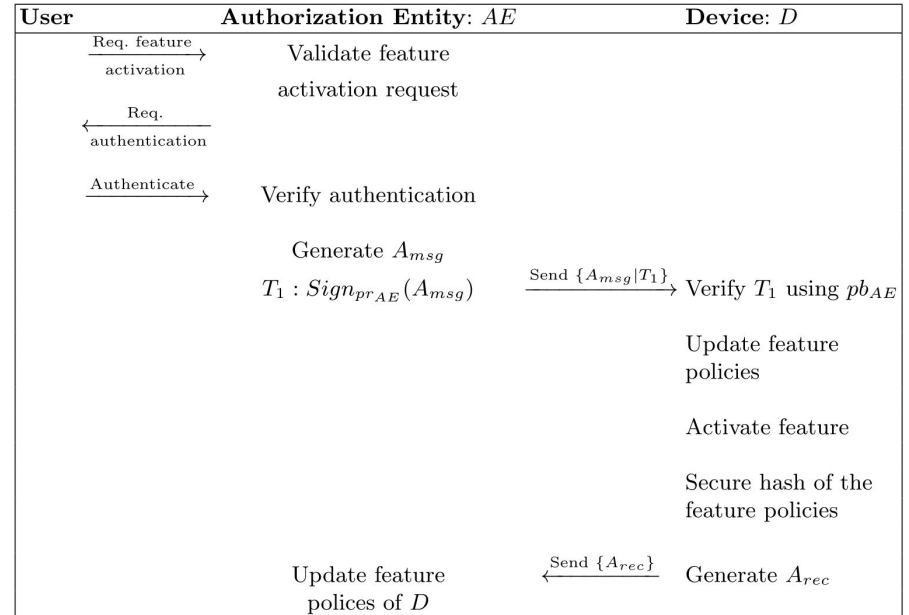
Use Case

- Feature Activation in Cars
 - Short signed messages



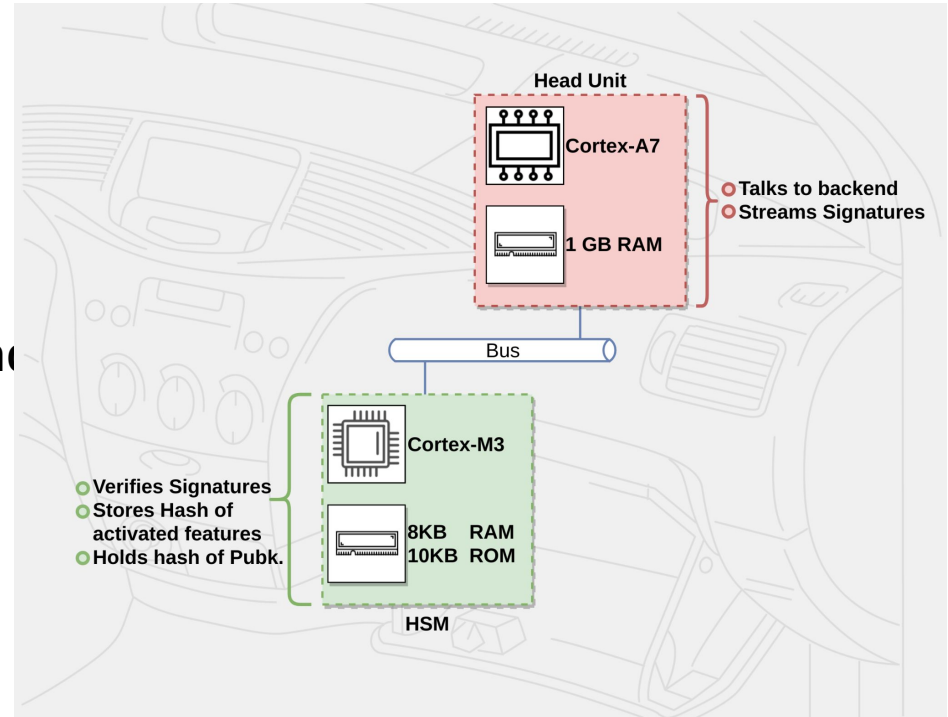
Use Case

- Feature Activation in Cars
 - Short signed messages
- Protocol already exists
 - Uses ECC



Use Case

- Feature Activation in Cars
 - Short signed messages
- Protocol already exists
 - Uses ECC
- HSM has to verify signatures and Pubkey
 - Is resource constrained
 - Holds hash of public key
 - Stores activated features in secure memory



Investigated Schemes

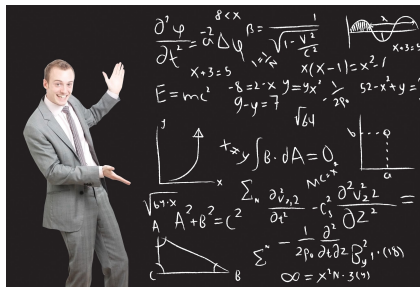
SPHINCS+



Hash Based

GeMSS

Rainbow



Multivariate

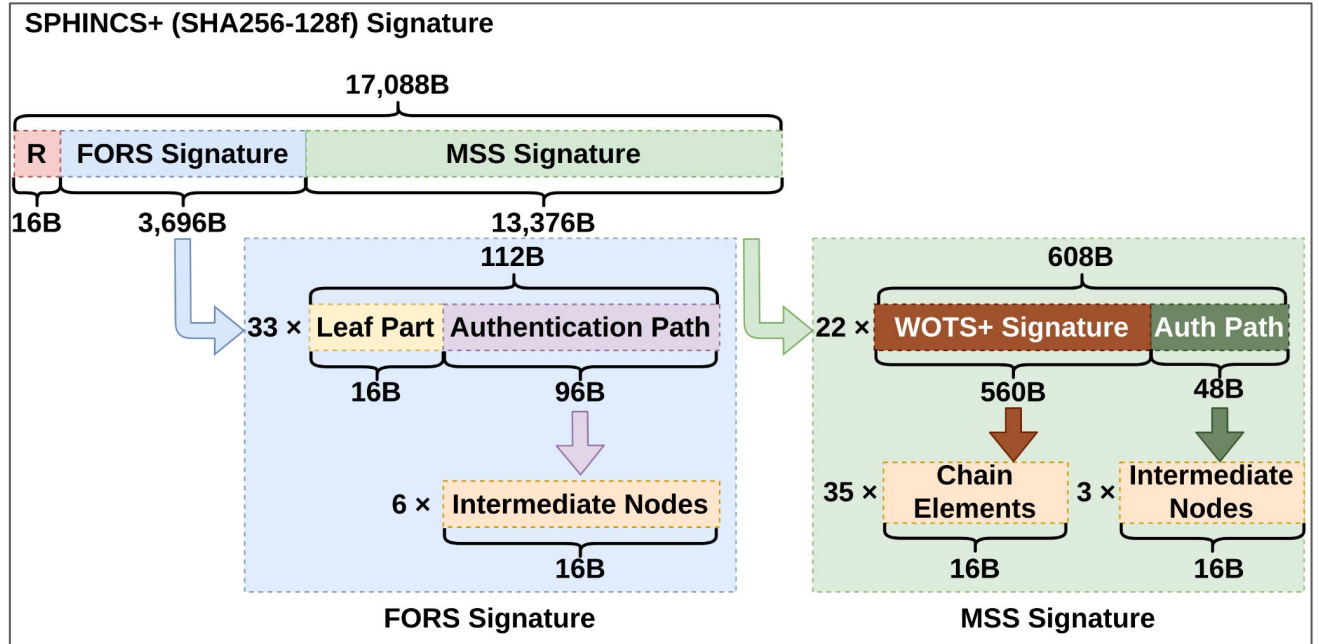
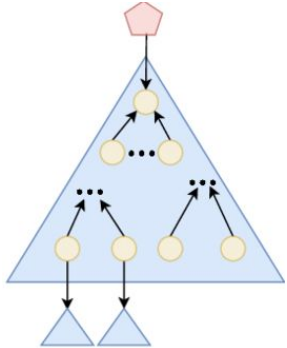
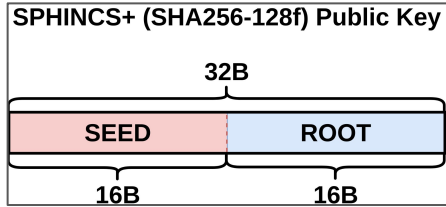
Dilithium

Falcon

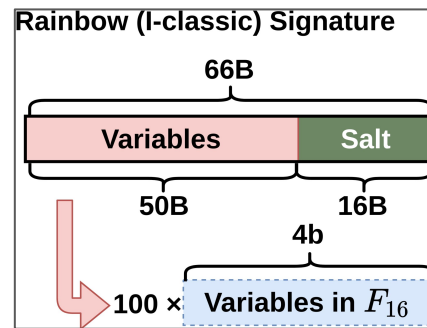
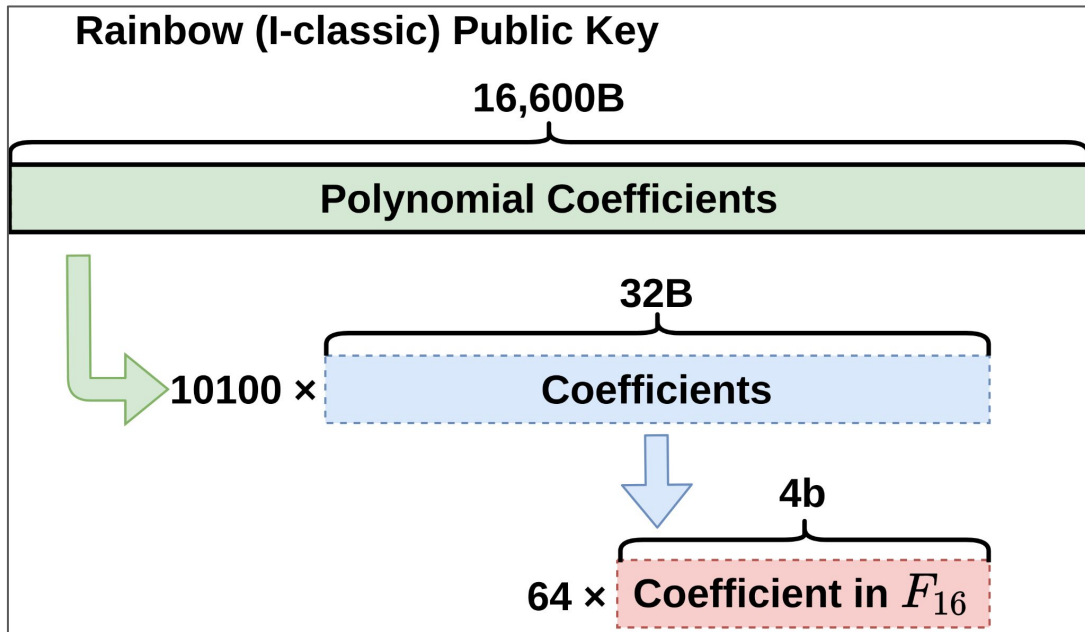


Lattice Based

SPHINCS+ (SHA256-128)

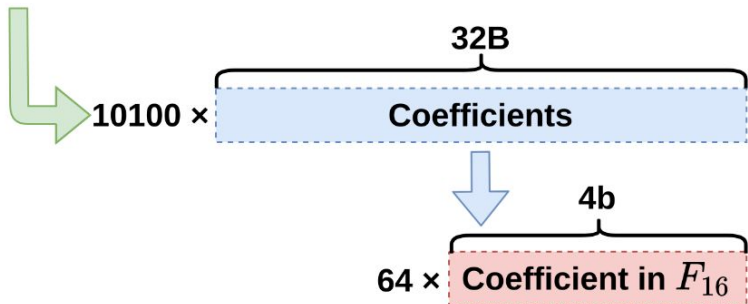
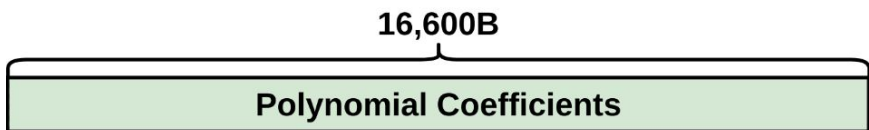


Rainbow (I-classic)

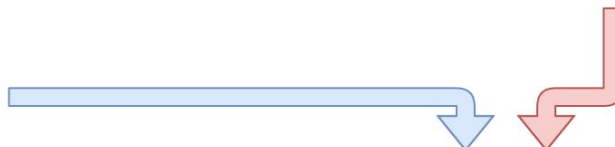
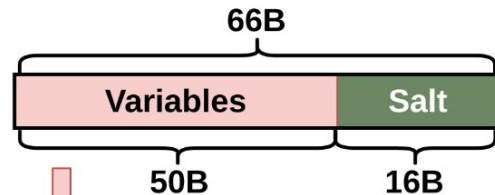


Rainbow (I-classic)

Rainbow (I-classic) Public Key



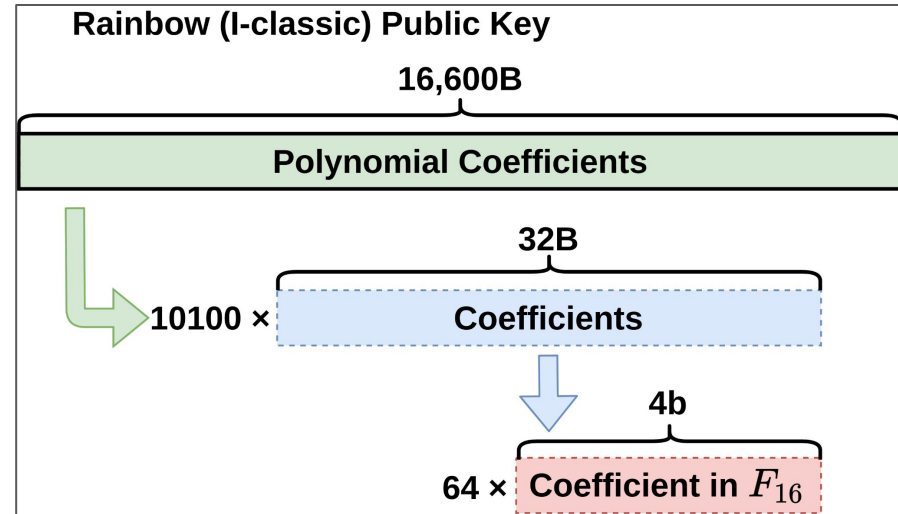
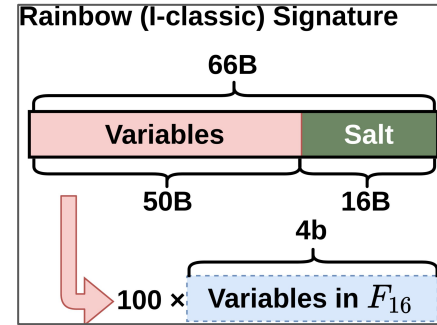
Rainbow (I-classic) Signature



$$\begin{aligned}
 p^{(1)}(x_1, \dots, x_n) &= \sum_{i=1}^n \sum_{j=i}^n p_{ij}^{(1)} \cdot x_i x_j + \sum_{i=1}^n p_i^{(1)} \cdot x_i + p_0^{(1)} \\
 p^{(2)}(x_1, \dots, x_n) &= \sum_{i=1}^n \sum_{j=i}^n p_{ij}^{(2)} \cdot x_i x_j + \sum_{i=1}^n p_i^{(2)} \cdot x_i + p_0^{(2)} \\
 &\vdots \\
 p^{(m)}(x_1, \dots, x_n) &= \sum_{i=1}^n \sum_{j=i}^n p_{ij}^{(m)} \cdot x_i x_j + \sum_{i=1}^n p_i^{(m)} \cdot x_i + p_0^{(m)}
 \end{aligned}$$

Rainbow (I-classic)

- Public key processed in order
- Signature fits in memory
- Chunk size of 32B possible



GeMSS (128)

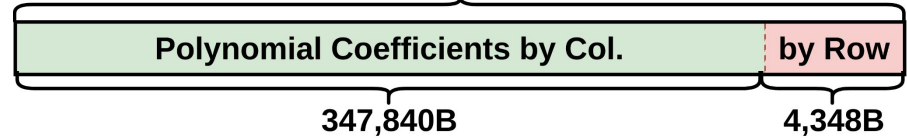
GeMSS (128) Signature

33B



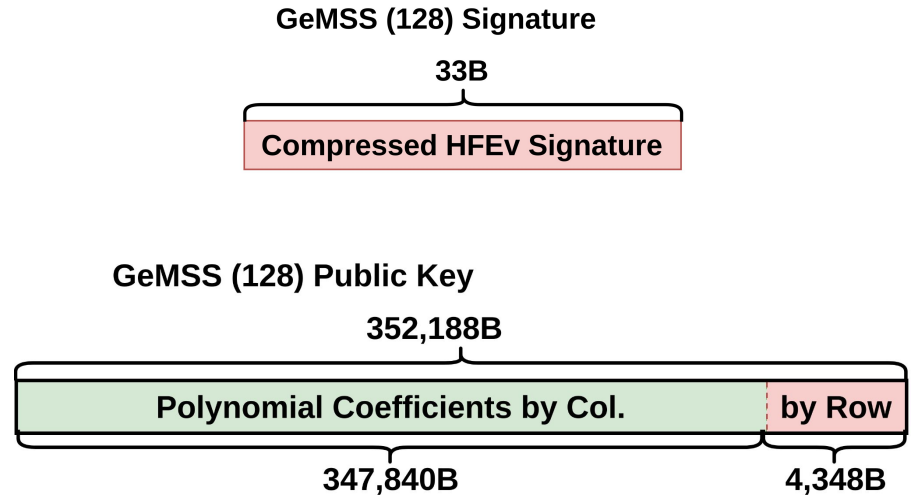
GeMSS (128) Public Key

352,188B



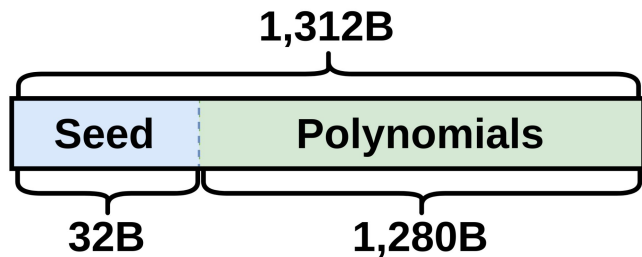
GeMSS (128)

- Verification has 4 iterations
 - Pubkey has to be streamed 4 times
- Signature fits in memory
- Chunk size of 2174B possible
 - Due to row wise storage

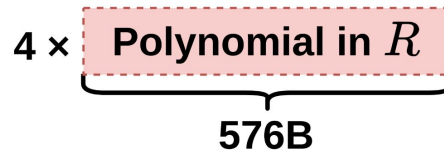
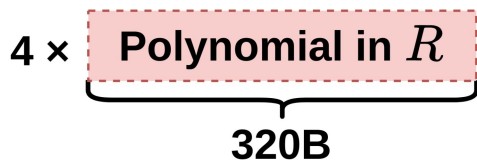
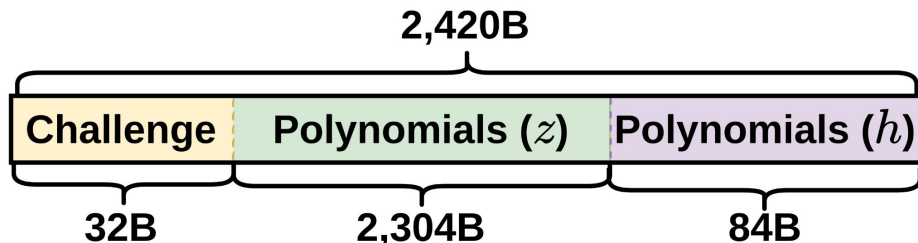


Dilithium (2)

Dilithium (2) Public Key



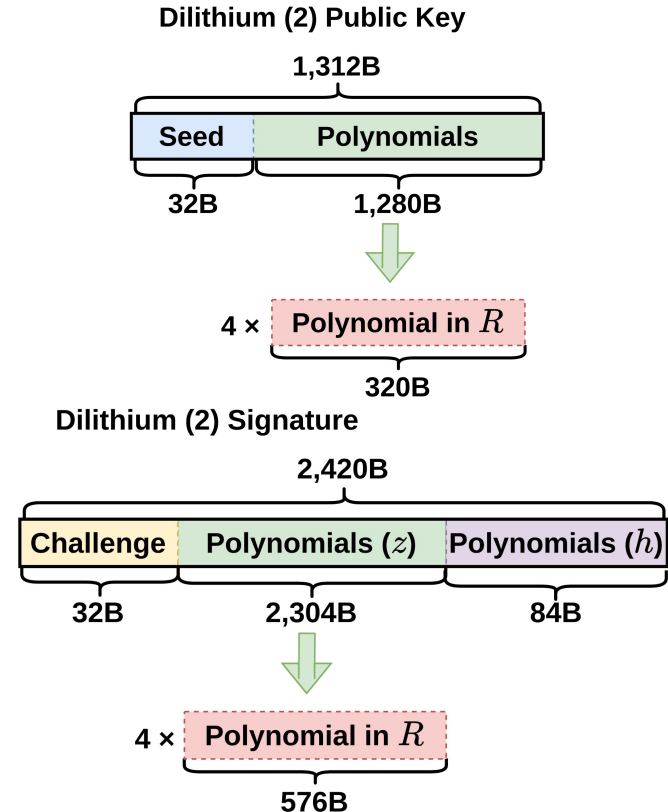
Dilithium (2) Signature



$$w' = Az - c \cdot t_1 \cdot 2^d$$

Dilithium (2)

- Signature fits in memory
- Public key is streamed one polynomial at a time
- Chunk size of 2420B and then 320B possible



Falcon

- No streaming required
- Everything fits in memory



Results

	streaming data			streaming time	
	$ pk $	$ sig $	total	500 kbit/s	20 Mbit/s
sphincs-s ^a	32	7 856	7 888	126.2 ms	3.2 ms
sphincs-f ^b	32	17 088	17 120	273.9 ms	6.9 ms
rainbowI-classic	161 600	66	161 666	2 586.7 ms	64.7 ms
gemss-128	352 188	33	1 408 785 ^c	22 540.6 ms	563.5 ms
dilithium2	1 312	2 420	3 732	59.7 ms	1.5 ms
falcon-512	897	690	1 587	25.4 ms	0.6 ms

^a-sha256-128s-simple ^b-sha256-128f-simple ^c4 · $|pk|$ + $|sig|$

Data Volume

	w/o pk vrf.	w/ pk verification		w/ streaming
		pk vrf.	total	time ^e 20 Mbit/s
sphincs-s ^a	8 741k	0	8 741k	87.4 ms
sphincs-f ^b	26 186k	0	26 186k	261.9 ms
rainbowI-classic	333k	6 850k ^d	7 182k	71.8 ms
gemss-128	1 619k	109 938k ^c	111 557k	1 115.6 ms
dilithium2	1 990k	133k ^e	2 123k	21.2 ms
falcon-512	581k	91k ^e	672k	6.7 ms

^a-sha256-128s-simple ^b-sha256-128f-simple ^cSHA-3/SHAKE
^dSHA-256 ^eAt 100 MHz (no wait states)

Cycle Counts

	memory				code
	total	buffer	.bss	stack	.text
sphincs-s ^a	6 904	4 928	780	1 196	2 724
sphincs-f ^b	7 536	4 864	780	1 892	2 586
rainbowI-classic	8 168	6 848	724	596	2 194
gemss-128	8 176	4 560	496	3 120	4 740
dilithium2	8 048	40	6 352	1 656	7 940
falcon-512	6 552	897	5 255	400	5 784

^a-sha256-128s-simple ^b-sha256-128f-simple

Memory Usage

Resources



Paper: <https://eprint.iacr.org/2021/662.pdf>

Code: <https://git.fslab.de/pqc/streaming-pq-sigs/>

Images used in this presentation were licensed from colourbox.de or freesvg.org. Hypertree visualization by Abid Khan.

