

# Verifying Post-Quantum Signatures in 8 kB of RAM

Ruben Gonzalez<sup>1</sup>, Andreas Hülsing<sup>2</sup>, Matthias J. Kannwischer<sup>3</sup>, Juliane Krämer<sup>4</sup>, Tanja Lange<sup>2</sup>, Marc Stöttinger<sup>5</sup>, Elisabeth Waitz<sup>6</sup>, Thom Wiggers<sup>7</sup>,  
and Bo-Yin Yang<sup>8</sup>

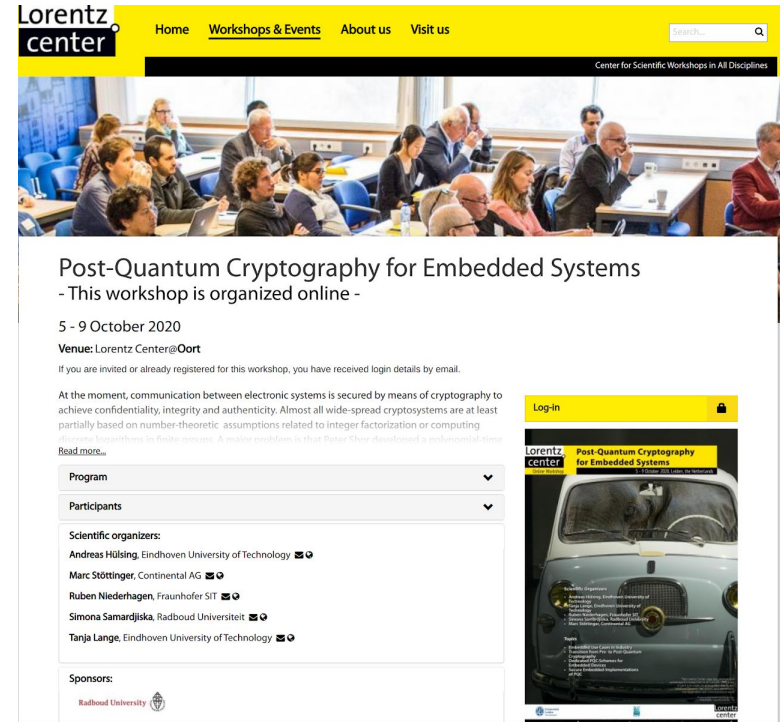
---

PQCrypto 2021, 20. - 22. July 2021

Presenter: Ruben Anthony 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. The header is yellow with the Lorentz Center logo and navigation links: Home, Workshops & Events, About us, and Visit us. A search bar is on the right. Below the header is a banner image of a group of people in a meeting. The main content area features the title "Post-Quantum Cryptography for Embedded Systems" with the subtitle "- This workshop is organized online -". It lists the dates "5 - 9 October 2020" and the venue "Lorentz Center@Oort". A paragraph explains that the workshop is for those invited or already registered, with login details sent by email. It also mentions that the workshop is based on number-theoretic assumptions related to integer factorization or computing discrete logarithms to finite groups. A "Log-in" button is visible. Below this, there are sections for "Program", "Participants", "Scientific organizers", and "Sponsors". The scientific organizers listed are Andreas Hülsing, Marc Stöttinger, Ruben Niederhagen, Simona Samardjiska, and Tanja Lange. The sponsors include Radboud University and others. A small image of a car is visible on the right side of the page.

**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 to solve these problems.

**Log-in**

**Program**

**Participants**

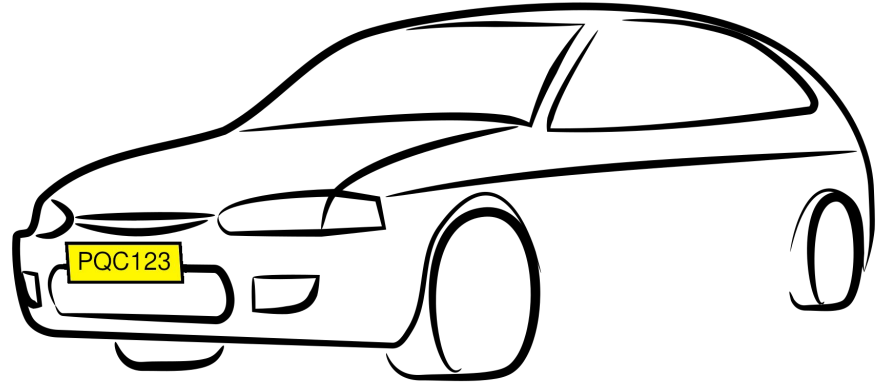
**Scientific organizers:**  
Andreas Hülsing, Eindhoven University of Technology  
Marc Stöttinger, Continental AG  
Ruben Niederhagen, Fraunhofer SIT  
Simona Samardjiska, Radboud University  
Tanja Lange, Eindhoven University of Technology

**Sponsors:**  
Radboud University

# Use Case

---

- ➔ Feature Activation in Cars
  - ➔ Short signed messages



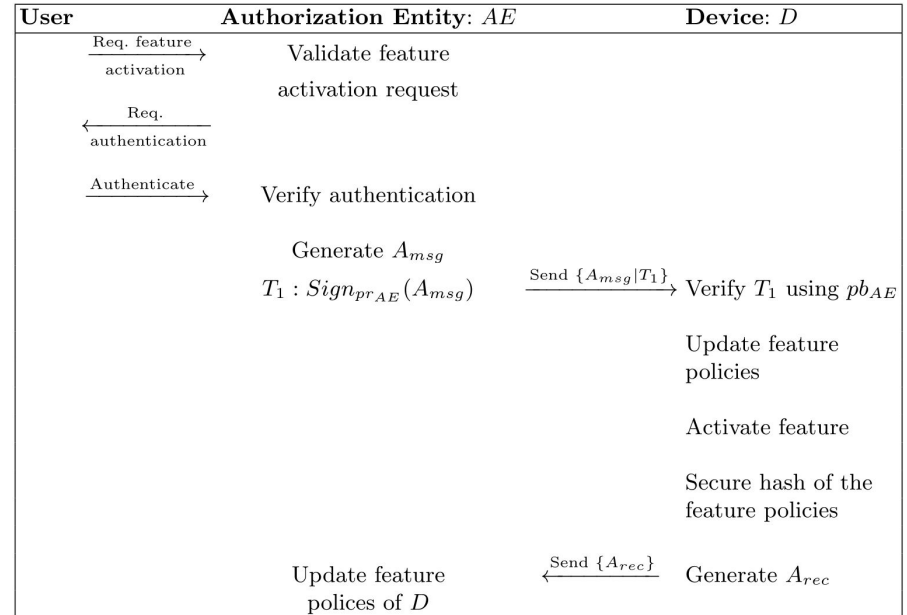
# Use Case

## ➡ Feature Activation in Cars

➡ Short signed messages

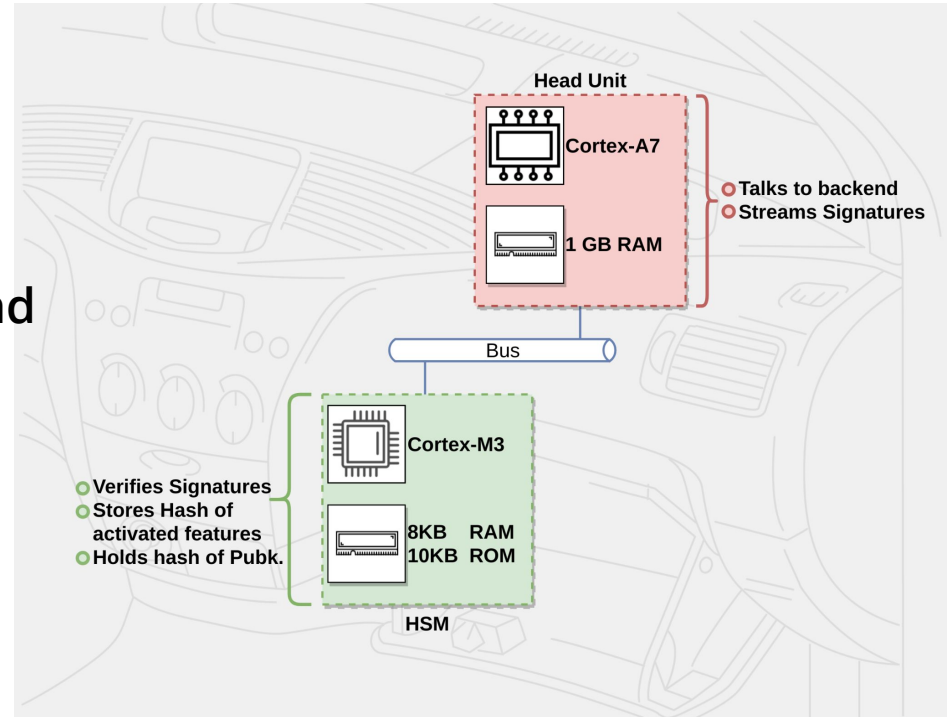
## ➡ Protocol already exists

➡ 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
  - ➔ Simulated on a STM32-F207ZG



# Experimental Setup



# Investigated Schemes

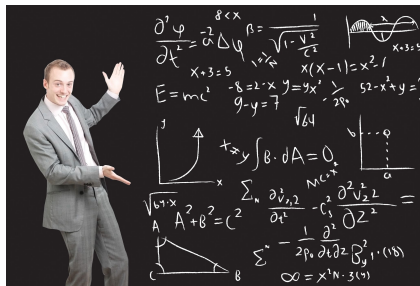
**SPHINCS+**



*Hash Based*

**GeMSS**

**Rainbow**



*Multivariate*

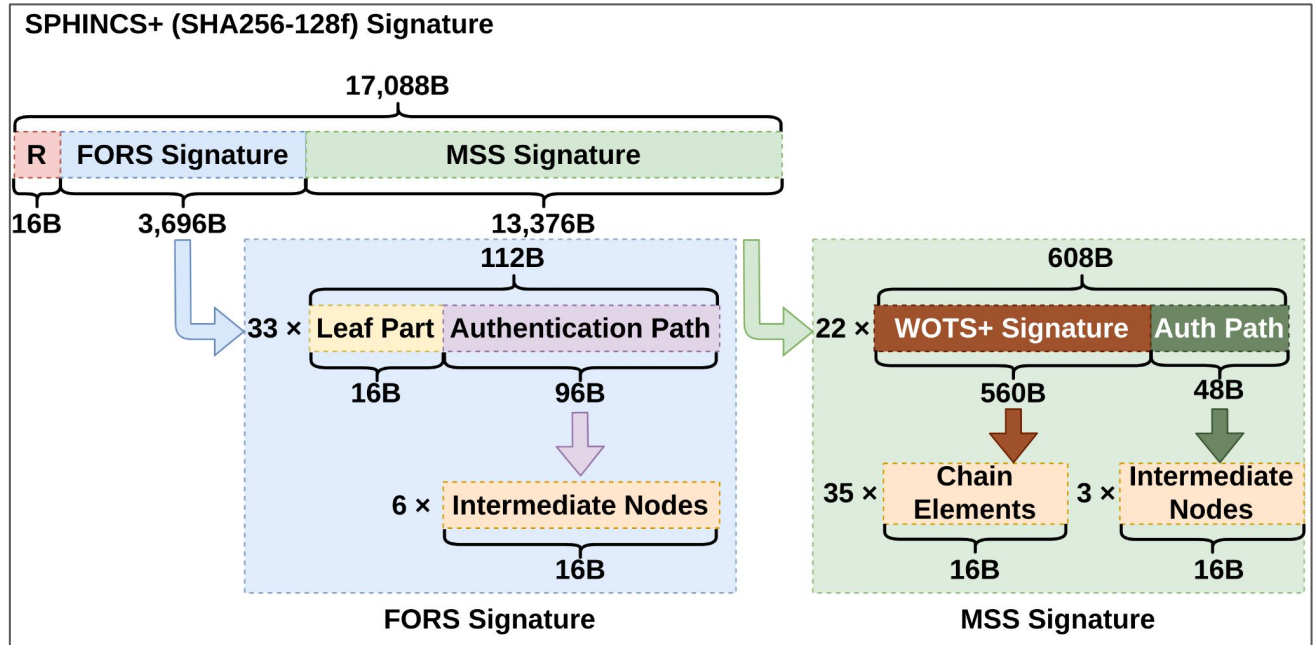
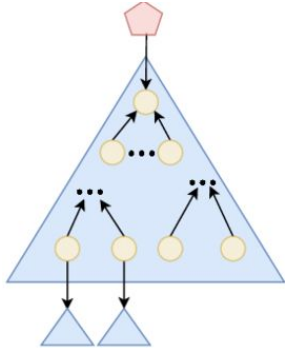
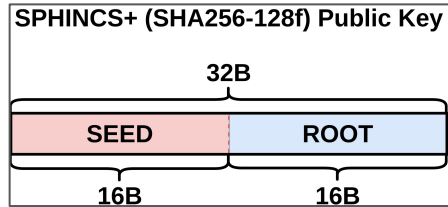
**Dilithium**

**Falcon**



*Lattice Based*

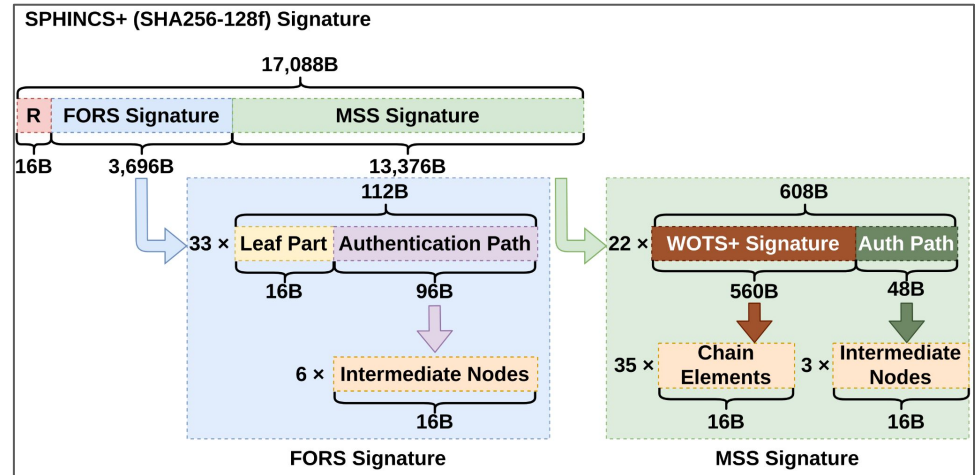
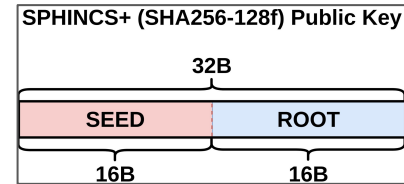
# SPHINCS+ (SHA256-128f)



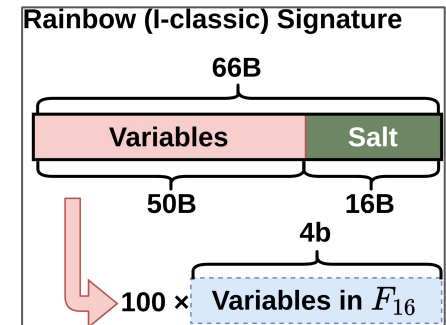
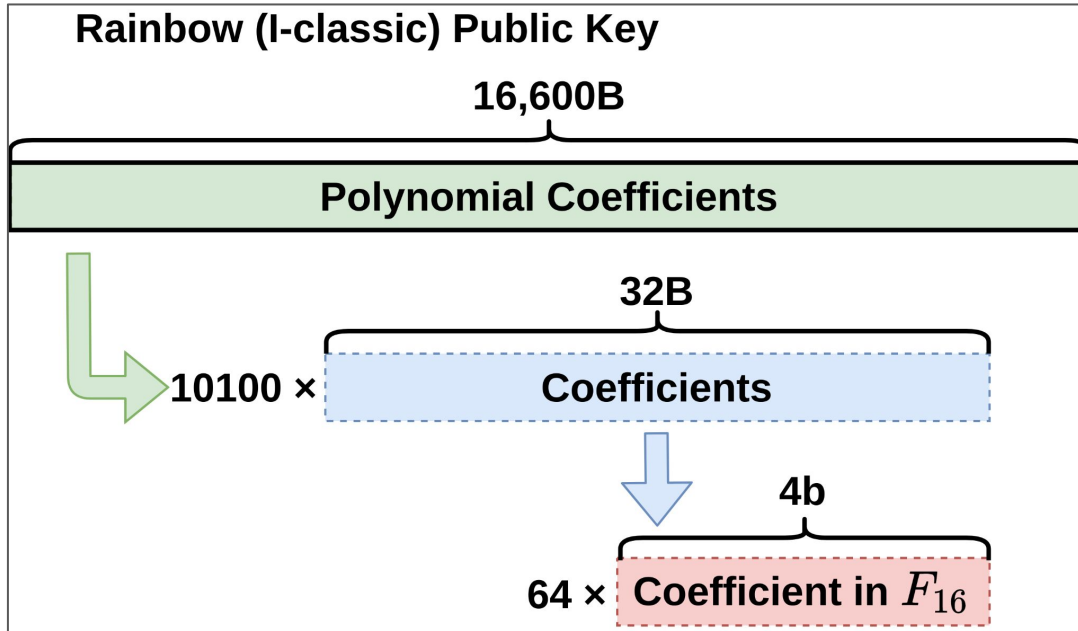


# SPHINCS+ (SHA256-128)

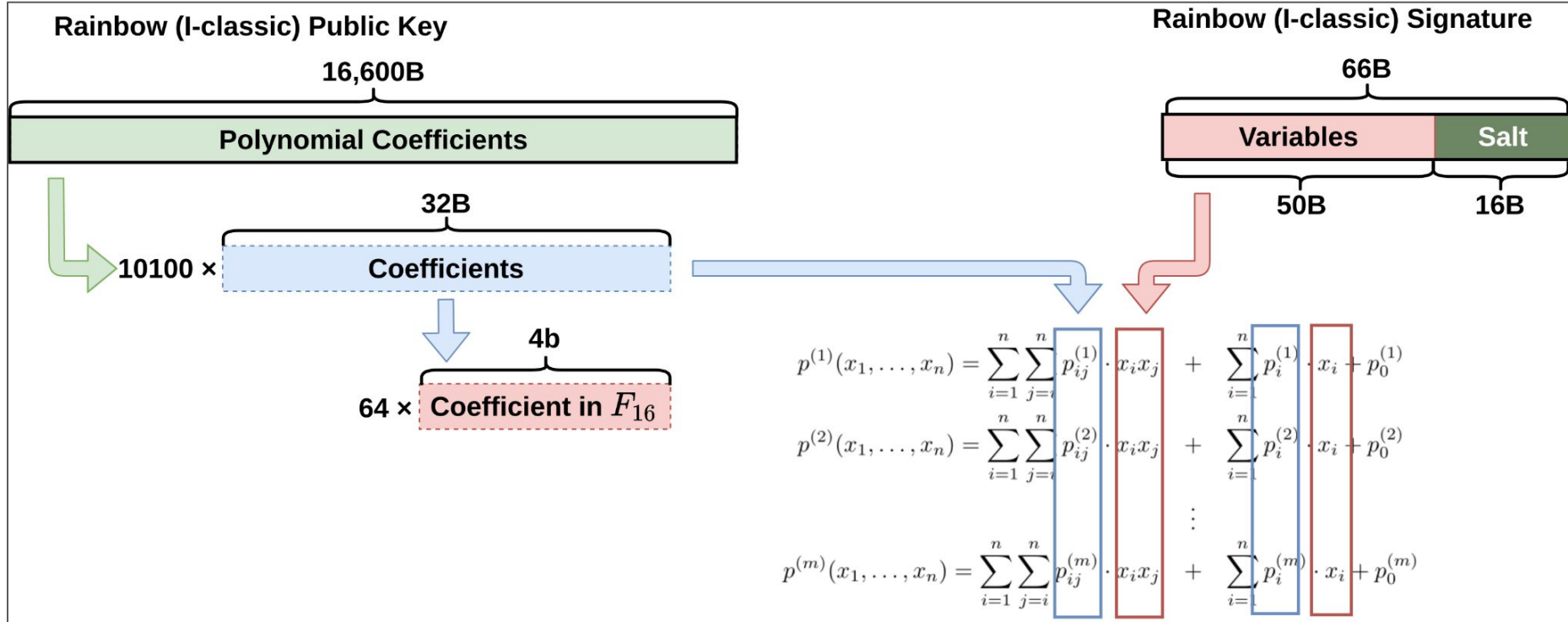
- ➔ Hold pubkey in memory
- ➔ Stream signature
  - in chunks of  $n \times 16\text{B}$
- ➔ Used reference code
  - Majority of time is spend in hash function



# Rainbow (I-classic)

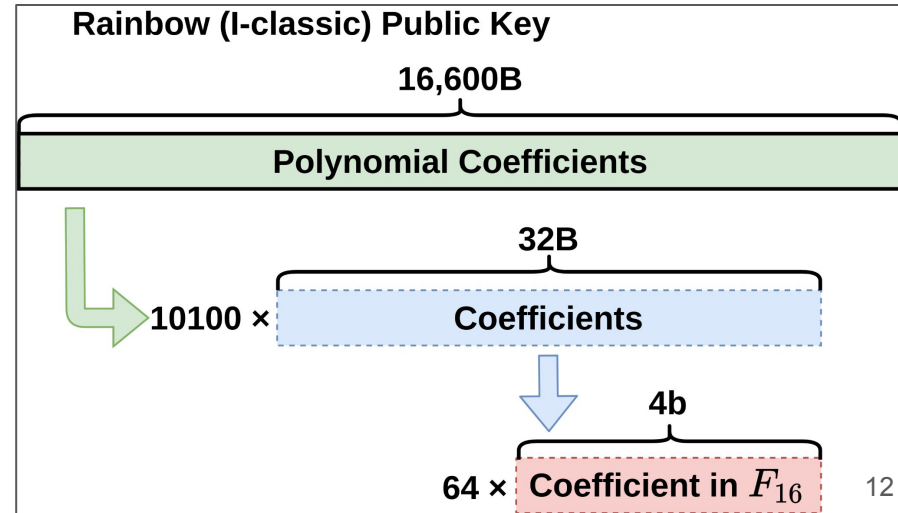
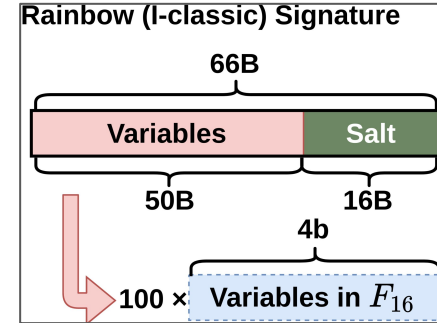


# Rainbow (I-classic)

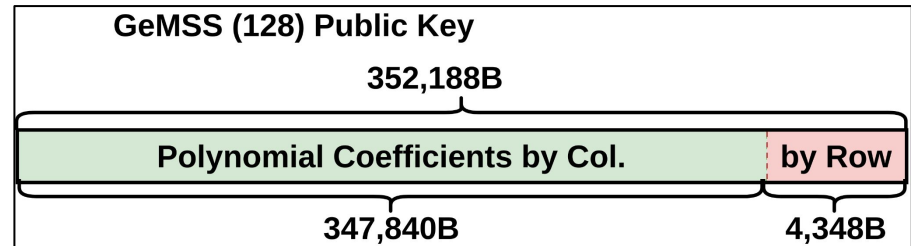
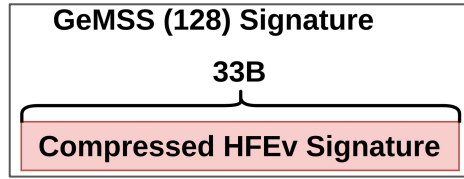


# Rainbow (I-classic)

- Hold signature in memory
- Stream public key
  - In chunks of  $n \times 32\text{B}$
- Based on bitslice implementation

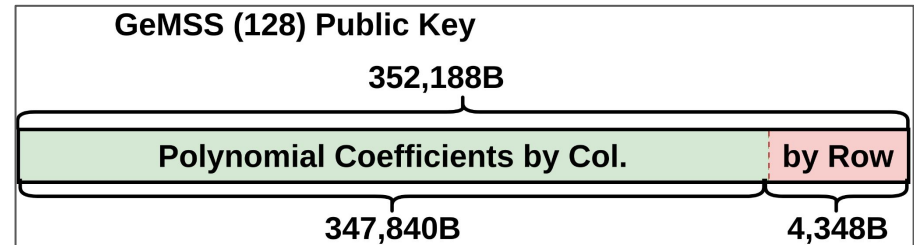
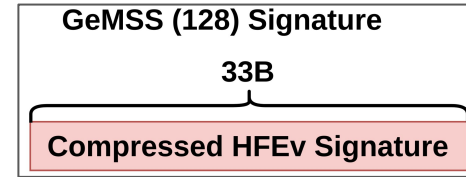


# GeMSS (128)

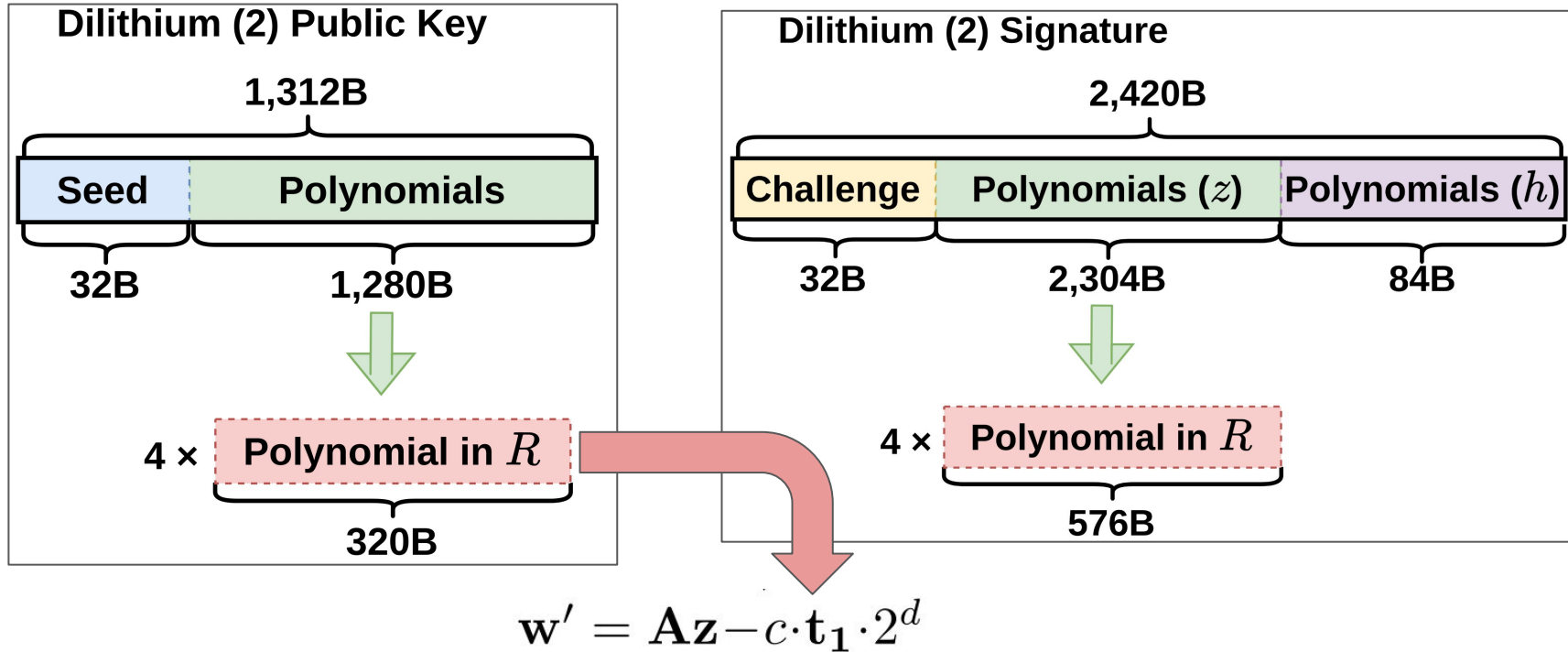


# GeMSS (128)

- **Hold signature in memory**
- **Stream public key 4 times**
  - Verification has 4 iterations
  - in chunks of  $n \times 2174\text{B}$
- **Based on reference code**
  - No embedded implementation available

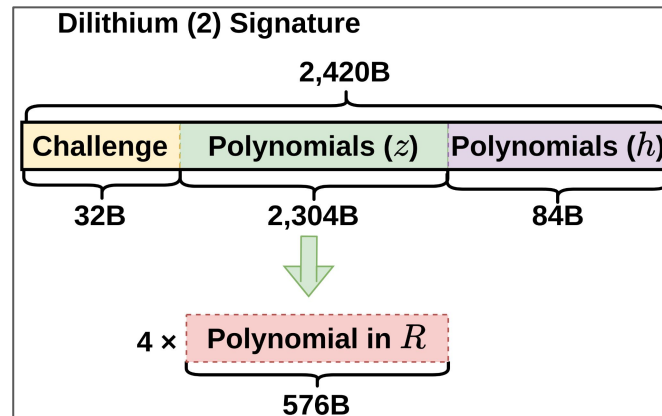
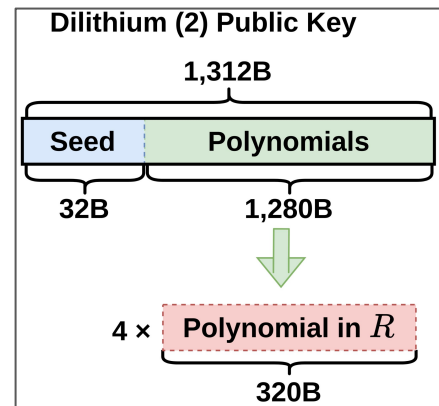


# Dilithium (2)



# Dilithium (2)

- Hold signature in memory
- Stream public key
  - One polynomials at a time
  - in chunks of 320B
- Based on optimized round 2 implementation for Cortex-M3/4
  - Updated for round 3 parameters





# Falcon

- **Everything fits in memory**
- **No streaming required**
- **Cortex-M3 compatible**  
optimized implementation  
available
  - **No FPU on M3, has to be emulated**
  - **Only tiny modification needed for  
memory constraints**



# Results<sup>1</sup>

	streaming data			streaming time	
	pk	sig	total	500 kbit/s	20 Mbit/s
sphincs-s <sup>a</sup>	32	7 856	7 888	126.2 ms	3.2 ms
sphincs-f <sup>b</sup>	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 <sup>c</sup>	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

<sup>a</sup> -sha256-128s-simple   <sup>b</sup> -sha256-128f-simple   <sup>c</sup>  $4 \cdot |pk| + |sig|$

## Data Volume

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

<sup>a</sup> -sha256-128s-simple   <sup>b</sup> -sha256-128f-simple   <sup>c</sup> SHA-3/SHAKE

<sup>d</sup> SHA-256   <sup>e</sup> At 100 MHz (no wait states)

## Cycle Counts

	memory				code
	total	buffer	.bss	stack	.text
sphincs-s <sup>a</sup>	6 904	4 928	780	1 196	2 724
sphincs-f <sup>b</sup>	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

<sup>a</sup> -sha256-128s-simple   <sup>b</sup> -sha256-128f-simple

## Memory Usage

<sup>1</sup>for 1000 iterations

# Resources

---



Paper: <https://ia.cr/2021/662>

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

*Hypertree visualization by Abid Khan.*

