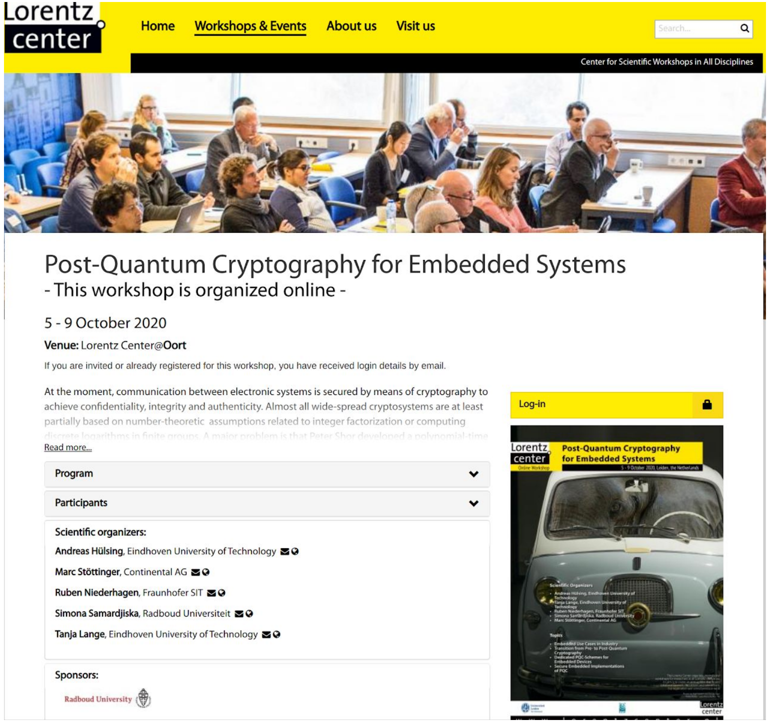
Verifying Post-Quantum Signatures in 8 kB of RAM

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

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

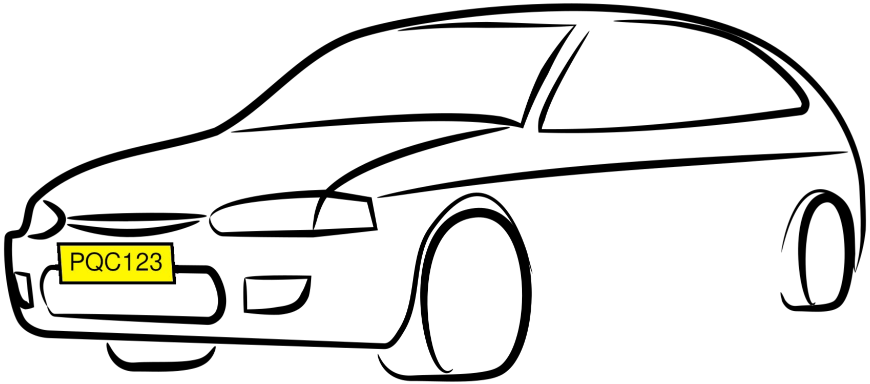
Background

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



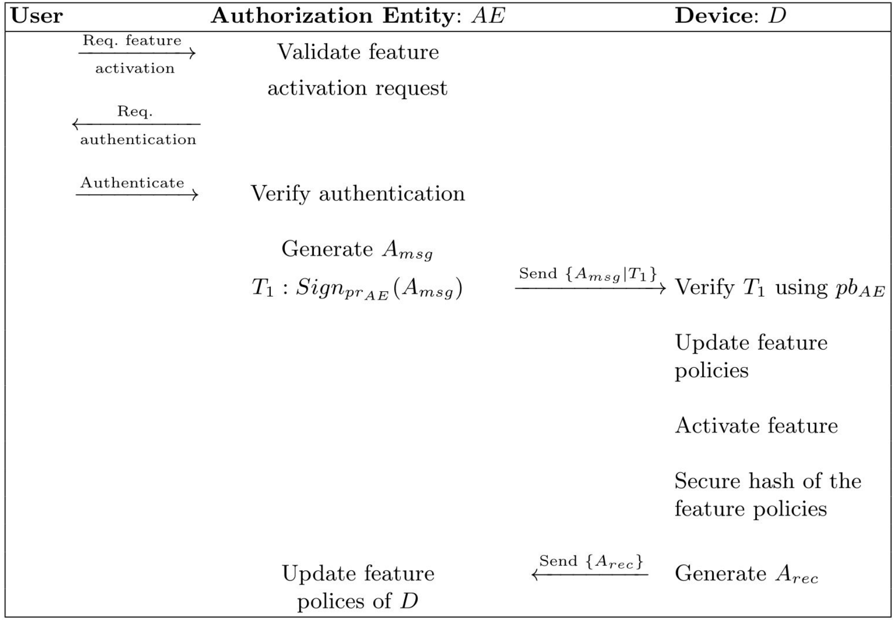
Use Case

- Feature Activation in Cars
  - Short signed messages



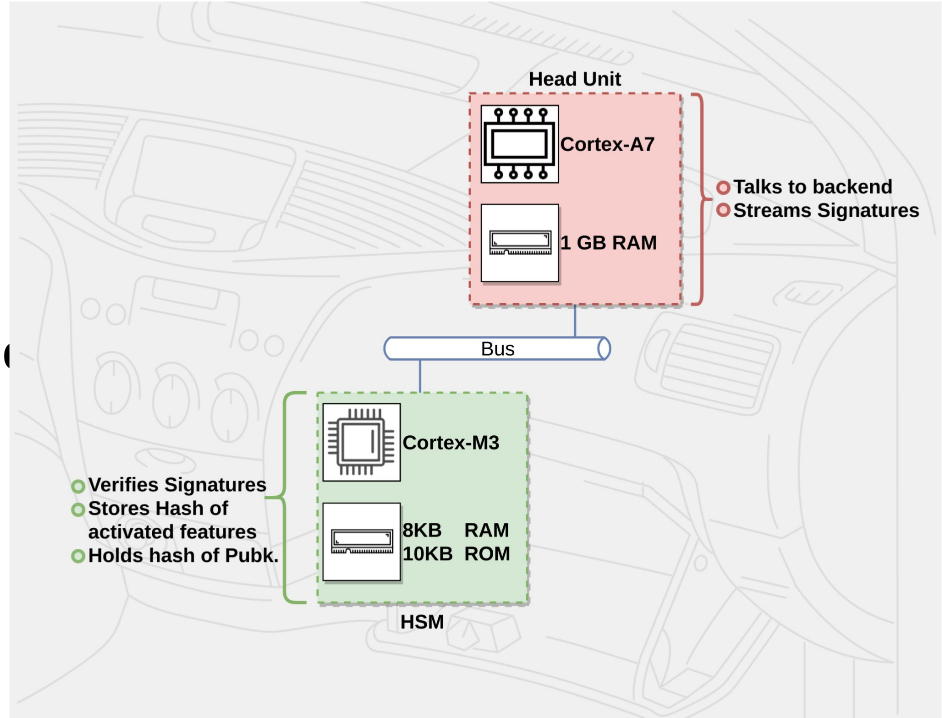
Use Case

- 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 an Pubkey
  - Is resource constrained
  - Holds hash of public key
  - Stores activated features in secure memory



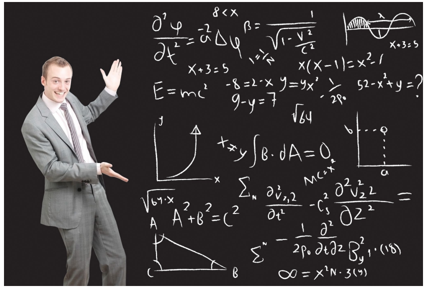
Investigated Schemes

SPHINCS+

Hash Based

GeMSS

Rainbow



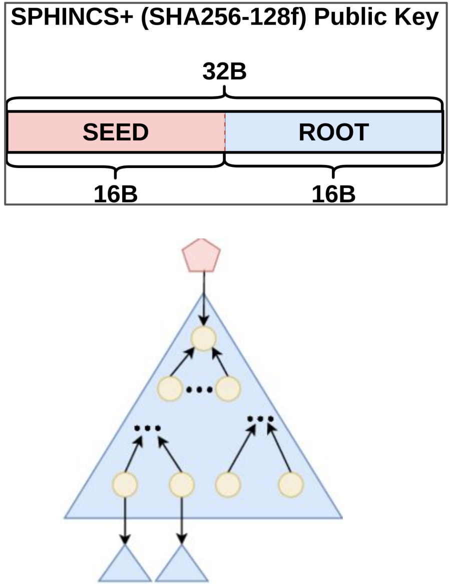
Multivariate

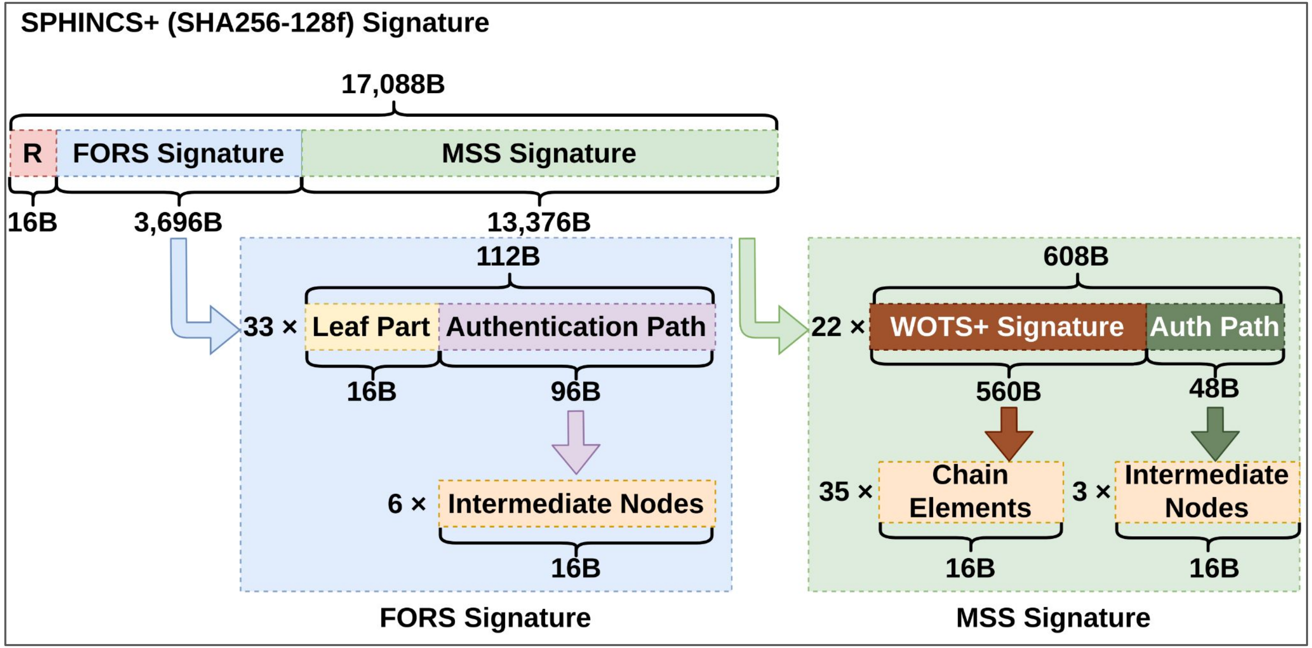
Dilithium

Falcon

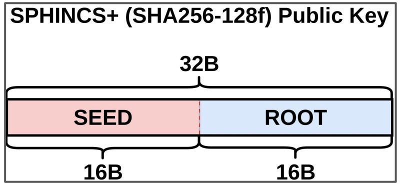
Lattice Based

SPHINCS+ (SHA256-128)

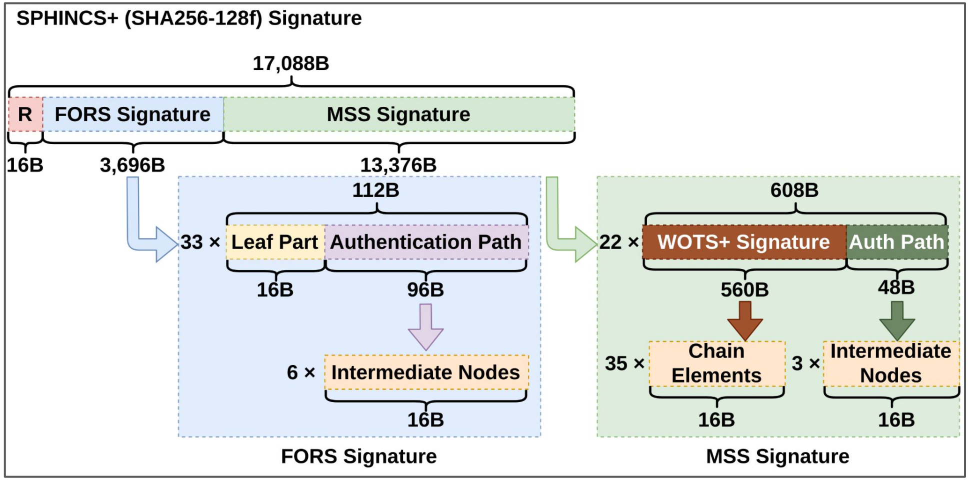




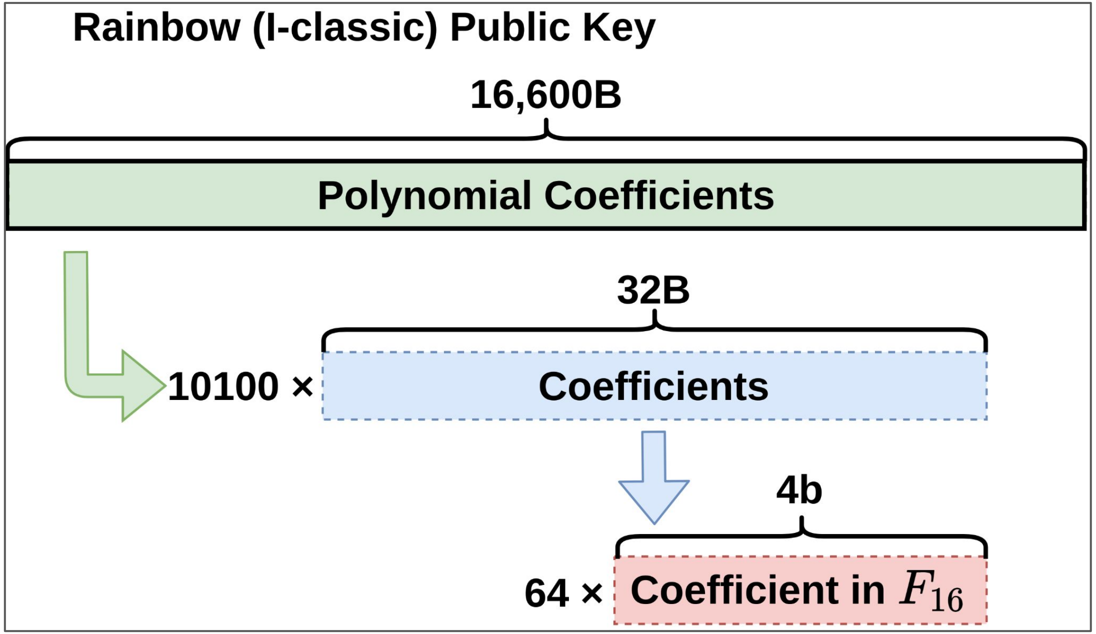
SPHINCS+ (SHA256-128)

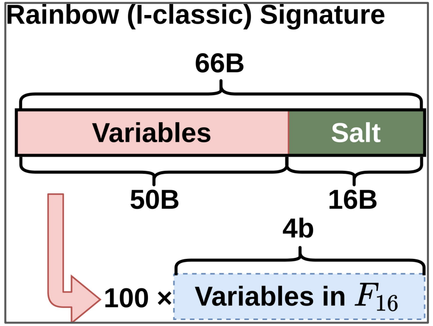


- Signature processed in order
- Pubkey fits in memory
- Chunk size of 16B possible

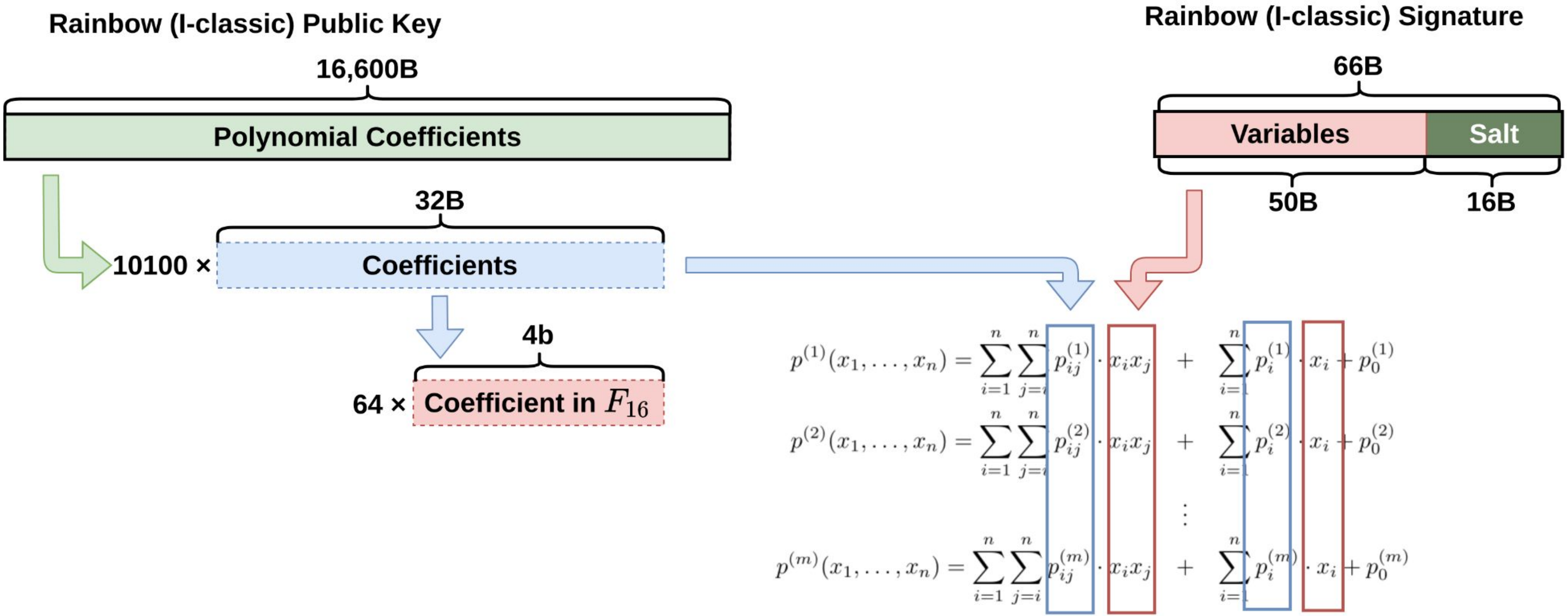


Rainbow (I-classic)



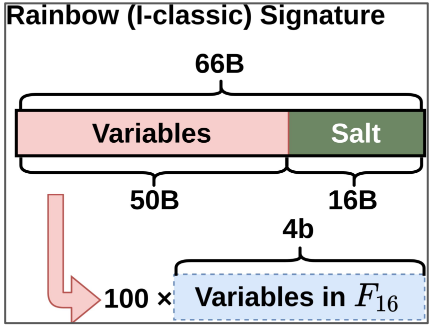


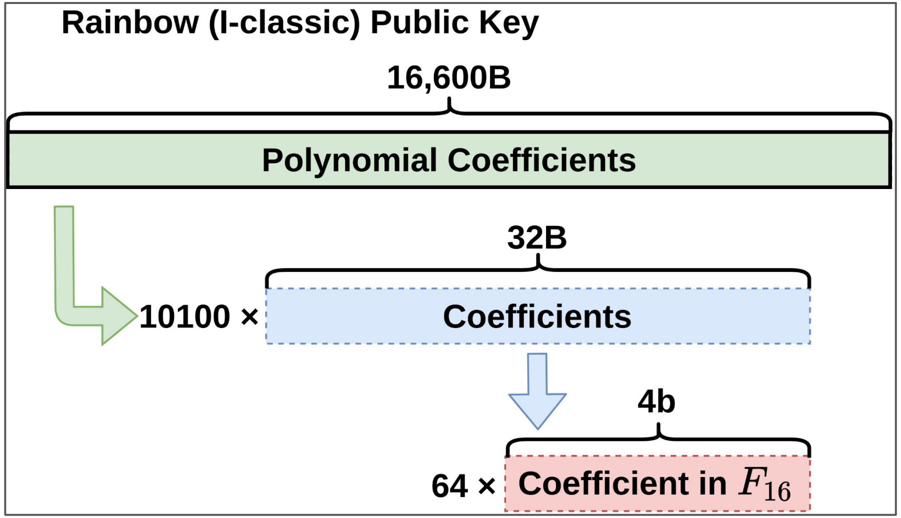
Rainbow (I-classic)



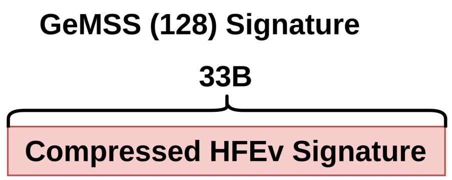
Rainbow (I-classic)

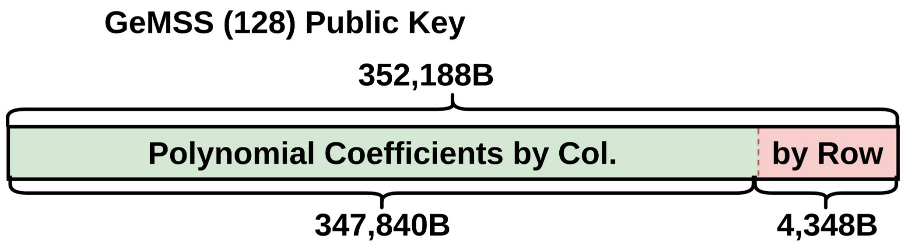
- Public key processed in order
- Signature fits in memory
- Chunk size of 32B possible





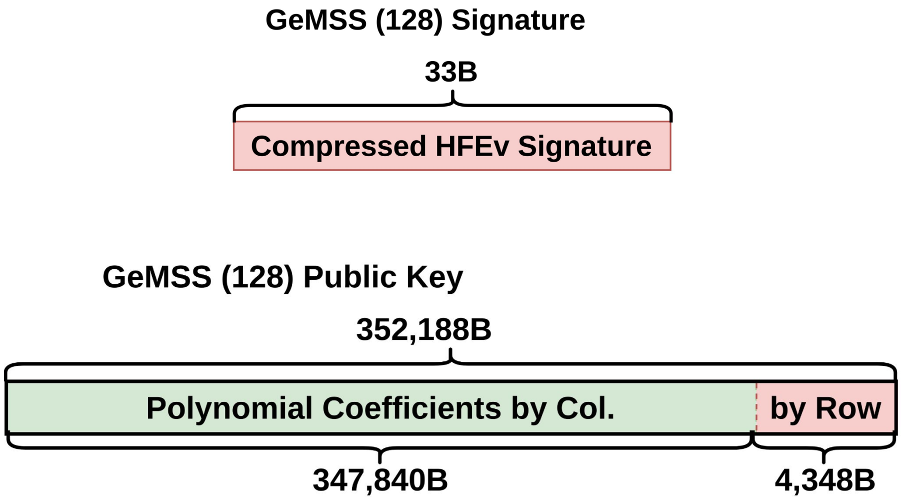
GeMSS (128)


GeMSS (128) Signature
33B
Compressed HFEv Signature

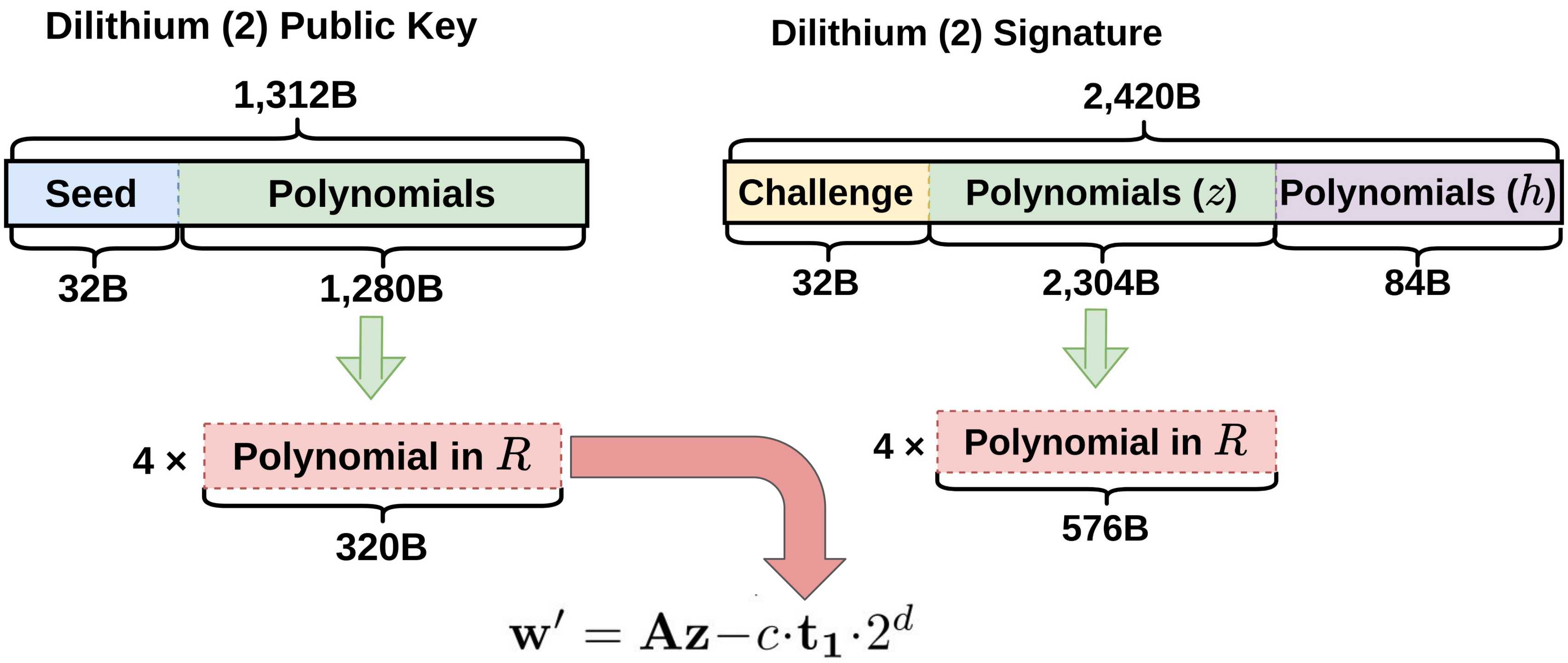

GeMSS (128) Public Key
352,188B
Polynomial Coefficients by Col.
by Row
347,840B
4,348B

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



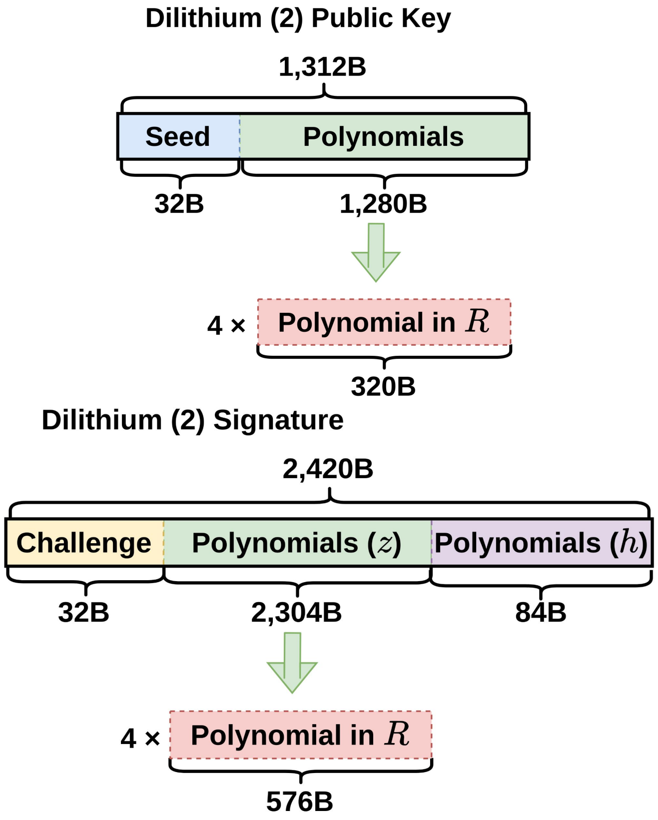
Dilithium (2)



Dilithium (2) Public Key
1,312B
Seed
Polynomials
32B
1,280B
4 × Polynomial in R
320B
Dilithium (2) Signature
2,420B
Challenge
Polynomials (z)
Polynomials (h)
32B
2,304B
84B
4 × Polynomial in R
576B
$\mathbf{w}' = \mathbf{A}\mathbf{z} - c \cdot \mathbf{t}_1 \cdot 2^d$

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` [c] $4 \cdot |pk| + |sig|$

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	90.6 ms
`sphincs-f`[b]	26 186k	0	26 186k	261.9 ms	268.7 ms
`rainbowI-classic`	333k	6 850k[d]	7 182k	71.8 ms	136.5 ms
`gemss-128`	1 619k	109 938k[c]	111 557k	1 115.6 ms	1 679.1 ms
`dilithium2`	1 990k	133k[c]	2 123k	21.2 ms	21.8 ms
`falcon-512`	581k	91k[c]	672k	6.7 ms	8.2 ms

[a] `-sha256-128s-simple` [b] `-sha256-128f-simple` [c] SHA-3/SHAKE
[d] SHA-256 [e] At 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. Hyptertree visualization by Abid Khan.