MAX PLANCK INSTITUTE
FOR SECURITY AND PRIVACY

Rainbow on Cortex-M4

Tung Chou[1], Matthias J. Kannwischer[2], and Bo-Yin Yang[1]
blueprint@crypto.tw, matthias@kannwischer.eu, by@crypto.tw

07 June, 2021, 3rd NIST PQC Standardization Conference

[1]Academia Sinica, Taipei, Taiwan
[2]Max Planck Institute for Security and Privacy, Bochum, Germany ⟶ Academia Sinica, Taipei, Taiwan

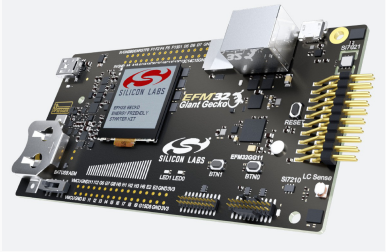
Rainbow

- Rainbow is a NISTPQC signature finalist using multivariate quadratic equations

security	scheme	public key	signature
128 bits	`I-classic`	161 600 bytes	66 bytes
	`I-circumzenithal`	60 192 bytes	66 bytes
	`I-compressed`	60 192 bytes	66 bytes
192 bits	`III-classic`	882 080 bytes	164 bytes
	`III-circumzenithal`	264 608 bytes	164 bytes
	`III-compressed`	264 608 bytes	164 bytes
256 bits	`V-classic`	1 930 600 bytes	212 bytes
	`V-circumzenithal`	536 136 bytes	212 bytes
	`V-compressed`	536 136 bytes	212 bytes

- In this talk: New speed records on the Cortex-M4
  - New paper `https://eprint.iacr.org/2021/532`
  - Code available at `https://github.com/rainbowm4/rainbowm4`

Our target platform: The Giant Gecko



- Keys are too big for the standard `STM32F407` (128 KB RAM)
- We instead target the Giant Gecko `EFM32GG11B` Starter Kit
  - `EFM32GG11B820F2048GL192` Cortex-M4 Cortex
  - 2 MB of flash memory, 512 kB of RAM, up to 72 MHz
- Still: Only `rainbowI` feasible
  - `rainbowIII` and `rainbowV` don't fit

Rainbow summary

- $n > m$,
  e.g., `rainbowI` uses $n = 100, m = 64$
- $\mathbb{F}_q$, $\mathbb{F}_{16}$ for `rainbowI` $\mathbb{F}_{256}$ for `III` and `V`
- Key Generation
  - Sample linear invertible transformations $T(\mathbb{F}_q^m \to \mathbb{F}_q^m)$ and $S(\mathbb{F}_q^n \to \mathbb{F}_q^n)$
  - Sample quadratic invertible central map $\mathcal{Q}(\mathbb{F}_q^n \to \mathbb{F}_q^m)$
  - Compute public key: $\mathcal{P} = T \circ \mathcal{Q} \circ S$ $(\mathbb{F}_q^m \to \mathbb{F}_q^n)$
  - Private key: $T^{-1}, \mathcal{Q}, S^{-1}$

- Signing
  - Compute digest $\mathbf{w} \in \mathbb{F}_q^m$ from message
  - Then compute signature $\mathbf{z}$

$$\mathbf{w} \in \mathbb{F}_q^m \overset{T^{-1}}{\longmapsto} \mathbf{x} \overset{\mathcal{Q}^{-1}}{\longmapsto} \mathbf{y} \overset{S^{-1}}{\longmapsto} \mathbf{z} \in \mathbb{F}_q^n$$

- Verification
  - Compute digest $\mathbf{w} \in \mathbb{F}_q^m$ from message
  - Compute $\mathbf{w}' = \mathcal{P}(\mathbf{z})$
  - Check $\mathbf{w} \overset{?}{=} \mathbf{w}'$

Rainbow summary: The central map $\mathcal{Q}$

- Most interesting part in any MQ scheme: $\mathbf{x} \overset{\mathcal{Q}^{-1}}{\longmapsto} \mathbf{y}$
- In Rainbow $\mathcal{Q}$ is defined in two layers
  - $n = 100 = v_1 + o_1 + o_2 = 36 + 32 + 32$

$$x_k = q_k(\mathbf{y}) = \sum_{i=1}^{v_1} \sum_{j=i}^{v_1+o_1} \alpha_{ij}^{(k)} y_i y_j, \text{for } 0 \leq k < o_1;$$

$$x_k = q_k(\mathbf{y}) = \sum_{i=1}^{v_1+o_1} \sum_{j=i}^{n} \alpha_{ij}^{(k)} y_i y_j, \text{for } o_1 \leq k < o_1 + o_2.$$

  - Pick first v_1 variables y_i at random; Solve first set of equation for o_1 variables
  - Plug into second set of equations; Solve for remaining o_2 variables

Rainbow: Implementation checklist

- $\mathbb{F}_{16}$ Multiplication
  - Used throughout key generation, signing, and verification of `rainbow-I`
  - Need constant-time and non-constant time versions
- Efficient linear equation solving
  - Used in signing for inverting the central map
  - Needs to be constant-time!
- Evaluating the public map $\mathcal{P}$
  - Run-time may depend on signature/pk

Finite field multiplication

- Tower field: $\mathbb{F}_{16} := \mathbb{F}_4[y]/(y^2 + y + x)$ with $\mathbb{F}_4 := \mathbb{F}_2[x]/(x^2 + x + 1)$
- Each element is represented by 4 bits
- Almost always: multiplication of large vectors by a scalar
- Easiest implementation: **Look-up tables**
  - 256 bytes: 1 multiplication (**3+ cycles / multiplication**)
  - 4096 bytes: 2 multiplications in parallel (**1.5+ cycle / multiplication**)
  - BUT: Cortex-M4 cores may have a cache → only use this on public data
- Constant-time: **Bitslice vector into 4 registers**
  - Takes 32 cycles for 32 field elements (excluding bitslicing)
  - **1 cycles / multiplication**

- Verification: $\mathcal{P}(\mathbf{z}) \stackrel{?}{=} h(M)$

$$\mathcal{P} : \mathbf{w}_i = \sum_{j=0}^{n} \sum_{k=j}^{n} \mathbf{z}_j \cdot \mathbf{z}_k \cdot a_{i,j,k} \text{ with } a_i = \begin{bmatrix} a_{i,0,0} & a_{i,0,0} & \dots & a_{i,0,n-1} \\ 0 & a_{i,1,1} & \dots & a_{i,1,n-1} \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & a_{i,n-1,n-1} \end{bmatrix}$$

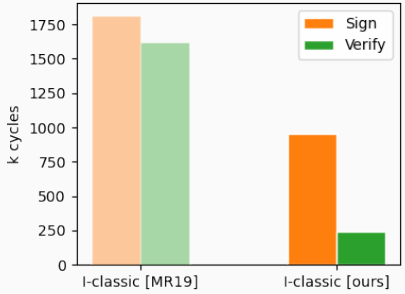
$$pk = \begin{bmatrix} a_{0,0,0} & a_{0,0,1} & \dots & a_{0,0,n-1} & a_{0,1,1} & \dots & a_{0,n-2,n-1} & a_{0,n-1,n-1} \\ a_{1,0,0} & a_{1,0,1} & \dots & a_{1,0,n-1} & a_{1,1,1} & \dots & a_{1,n-2,n-1} & a_{1,n-1,n-1} \\ \vdots & \vdots & & & & & & \\ a_{m-1,0,0} & a_{m-1,0,1} & \dots & a_{m-1,0,n-1} & a_{m-1,1,1} & \dots & a_{m-1,n-2,n-1} & a_{m-1,n-1,n-1} \end{bmatrix}$$

$$\begin{matrix} \mathbf{z}_0\mathbf{z}_0 & \mathbf{z}_0\mathbf{z}_1 & \dots & \mathbf{z}_0\mathbf{z}_{n-1} & \mathbf{z}_1\mathbf{z}_1 & \dots & \mathbf{z}_{n-2}\mathbf{z}_{n-1} & \mathbf{z}_{n-1}\mathbf{z}_{n-1} \end{matrix}$$

Verifying Rainbow signatures

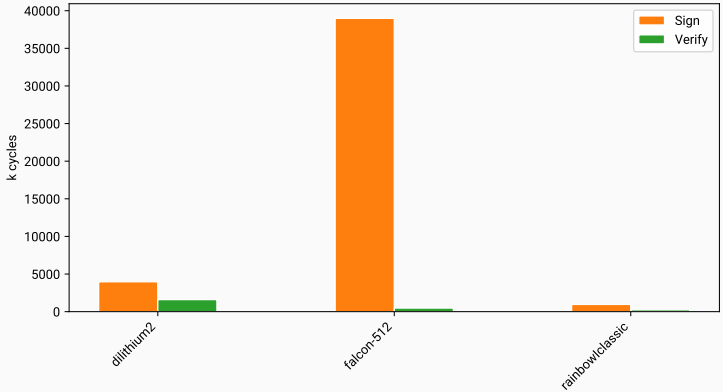
- Our approach
  - Instead of multiplying each column by $\mathbf{z}_i\mathbf{z}_j$ and accumulating $\mathbf{w}$
  - We have 15 accumulators $\mathbf{w}^{(i)}$ (for each possible value $\mathbb{F}_{16}\backslash\{0\}$)
  - Depending on $\mathbf{z}_i\mathbf{z}_j$, we add the column to the corresponding accumulator
  - Do multiplications in the end

Results



- Our implementation is much faster than current state-of-the-art
  - Previous implementation of round 2 Rainbow (smaller params, $n = 96$) by Moya Riera
  - `https://hdl.handle.net/2117/169145`
  - Using LUTs throughout
- We outperform this implementation by **2× for signing** and **7× for verification**

Results (2)



- Rainbow is **by far the fastest** NIST PQC signature finalist on the Cortex-M4
  - Signing is $4\times$ faster than Dilithium, $45\times$ faster than Falcon
  - Verification is $5\times$ faster than Dilithium, $2\times$ faster than Falcon

Thank you very much for your attention

matthias@kannwischer.eu

Paper: `https://eprint.iacr.org/2021/532`

Code: `https://github.com/rainbowm4/rainbowm4`