Quantum-Resistant Security for Software Updates on Low-power Networked embedded Devices

Gustavo Banegas, Koen Zandberg, Emmanuel Baccelli, Adrian Herrmann, **Benjamin Smith**

April 15, 2022

Inria + FU Berlin + Laboratoire d'Informatique de l'École polytechnique (LIX)

You can’t secure what you can’t update

You can’t secure what you can’t update, *securely*

You can’t secure what you can’t update, *securely*

What are we trying to (securely) update today: **low-end IoT**.

- Low computing power
- Low memory
- Low battery
- Low price

Full results: `https://ia.cr/2021/781`

Target platform: RIOT OS

RIOT is a free, community-drive open-source OS for **low-end** IoT devices.

- Supports $\geq$ 68 **CPUs** (8-, 16-, and 32-bit)
- Supports $\geq$ 240 **different boards**
- Application development: **C, C++, Rust**
- Modular microkernel design
- Find out more: `https://riot-os.org`

Target platform: RIOT OS

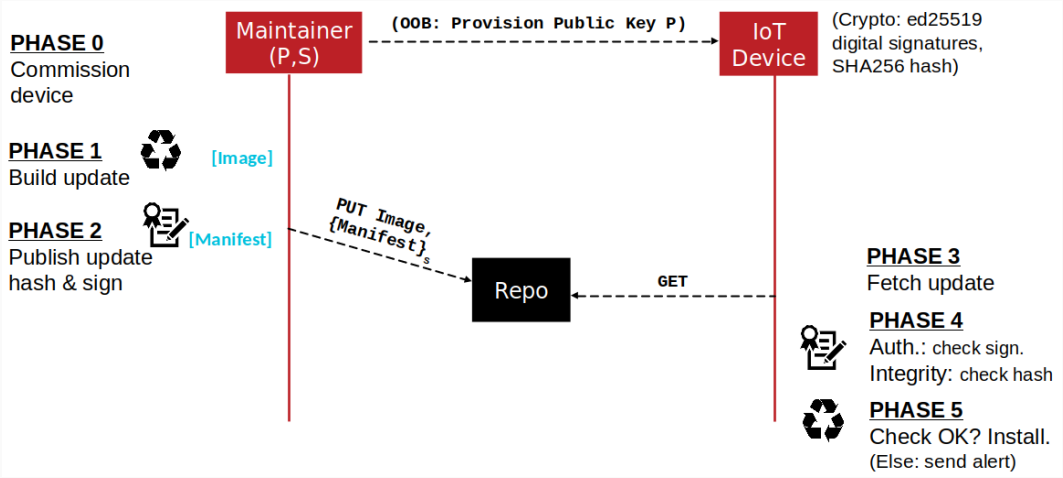
RIOT is a free, community-drive open-source OS for **low-end** IoT devices.

- Supports $\geq$ 68 **CPUs** (8-, 16-, and 32-bit)
- Supports $\geq$ 240 **different boards**
- Application development: **C, C++, Rust**
- Modular microkernel design
- Find out more: `https://riot-os.org`

RIOT supports **SUIT** (RFC 9019): **S**ecure **U**pdates for the **I**nternet of **T**hings.

Question: what is the practical cost of transitioning from pre-quantum SUIT (based on ECDSA/P256 or Ed25519) to **post-quantum** alternatives?

SUIT: Software Updates for the Internet of Things



What we measured: pre- and post-quantum signature schemes

Pre-quantum baseline (SUIT standard) and **Post-quantum alternatives**[1]

Algorithm	Private key		Public key		Signature		SUIT Manifest	
	Bytes	*Ratio*	Bytes	*Ratio*	Bytes	*Ratio*	Bytes	*Ratio*
Ed25519 *or* **ECDSA**	32	*1×*	32	*1×*	64	*1×*	483	*1×*
Dynamic[2] **Dilithium**	2528	*79×*	1312	*41×*	2420	*37.8×*	2839	*5.88×*
Static[3] **Dilithium**	18912	*591×*	17696	*553×*				
Falcon	1281	*40.0×*	897	*28.0×*	666	*10.4×*	1085	*2.24×*
LMS[4] (RFC8554)	64	*2×*	60	*0.94×*	4756	*74.3×*	5175	*10.7×*

[1] No, we (probably) *didn't* measure your favourite PQ signature scheme

[2] *Dynamic Dilithium* = "standard".

[3] *Static Dilithium* = matrices expanded from seed and stored.

[4] LMS = Leighton–Micali, stateful hash-based signatures. State is not a problem for this application.

Three boards representing the 32-bit microcontroller landscape

RIOT supports $\geq$ 240 platforms. We took **three** representative 32-bit boards:

Architecture	Board	Speed	RAM	Flash
ARM Cortex-M4	Nordic nRF52480	64MHz	256kB	1MB
Espressif ESP32	WROOM-32	80MHz	520kB	448kB
RISC V	Sipeed Longan Nano	72MHz	32kB	128kB

Three boards representing the 32-bit microcontroller landscape

RIOT supports ≥ 240 platforms. We took **three** representative 32-bit boards:

Architecture	Board	Speed	RAM	Flash
ARM Cortex-M4	Nordic nRF52480	64MHz	256kB	1MB
Espressif ESP32	WROOM-32	80MHz	520kB	448kB
RISC V	Sipeed Longan Nano	72MHz	32kB	128kB

For RIOT, emphasis on **portability**.

- No assembly, no platform-specific tricks.
- Open implementations (notably `PQClean`)
- Minimal modifications for RIOT compatibility: removing `malloc`, etc.

This talk: mostly `nRF52480`*. (Similar figures for ESP32 and RISC-V in the paper.)*

Signature benchmarks: Verification on ARM Cortex-M4

Algorithm	Base library	Flash (B)	Stack (B)	Time (ms)	kTicks
Ed25519	**C25519**	5106	1300	1953	125012
Ed25519	**Monocypher**	13852	1936	40	2599
ECDSA	**Tinycrypt**	6498	1024	313	20037
Dynamic **Dilithium**	**PQClean**	11664	**36058**	53	3407
Static **Dilithium**	**PQClean**	**26672**	**19504**	23	1510
Falcon	**PQClean**	**57613**	4744	15	1004
LMS (RFC8554)	Cisco	12864	1580	**123**	7908

Note: Dynamic Dilithium requires too much stack to run in the 32kB RAM available on the Sipeed Nano (RISC-V) board.

Impact on data transfer

Example: suppose we want to update RIOT firmware for the `nRF52480` board. The firmware itself is a ≈ 46kB binary, and the (pre-quantum) crypto is ≈ 6kB.

How much data do we need to transmit?

SUIT				Data Transfer	
Signature	Hash	Flash	Stack	no crypto	crypto incl.
Ed25519	SHA256	52.4kB	16.3kB	47kB	53kB
Dilithium	SHA3-256	+30%	+210%	+4.3%	+34%
Falcon	SHA3-256	+120%	+18%	+1.1%	+120%
LMS	SHA3-256	+34%	+1.2%	+9%	+43%

Recommendations for four typical software updates

First two use cases:

1. **Small software module update**: $\approx$ 5kB.
2. **Small firmware update** $\approx$ 50kB *without* crypto libs

In both of these applications, **speed** and **signature size** are critical.

Here, **Falcon has a clear advantage** over Dilithium and LMS.

Recommendations for four typical software updates

Updating the crypto complicates things:

3. **Small firmware update** $\approx$ 50kB *including* crypto libs

Larger crypto lib transfer $\implies$ higher energy cost on low-power networks.

*It takes 30-60s to transfer 50*kB *on a low-power IEEE802.15.4 radio link, but signature verification only varies by 2 seconds between all candidates...*

In this case, **LMS gives the best tradeoff** between flash size, network transfer costs, verification time, and stack size.

Recommendations for four typical software updates

Finally,

4. **Large firmware update** $\approx$ 250kB

For larger updates, the network transfer costs overwhelm the other factors.

This reduces any relative advantage between PQ signatures.

Conclusions

Post-quantum IoT software updates with SUIT are **feasible now**.

- **Falcon** is best for smaller module and firmware updates;
- **LMS** is better when the crypto lib is transferred;
- but there is no clear winner for much larger updates.

Consider using RIOT for easy, portable, open IoT crypto development.

`https://riot-os.org/`
`https://ia.cr/2021/781`