

Threshold Signatures for Embedded Platforms

D. Gallay *, A. Duc *, D. Vizár

Embedded devices deployed in the so-called Internet of Things applications often face a tension between their generally weak security posture and a heightened exposure due to their connectivity (remote software exploits) and physical exposure (side channel attacks). The possibility of an ensuing leakage of a remote device's long-term keys may decrease trust in the application to an unacceptable level. This may be alleviated with help of threshold signatures, where the signing key is split into shares. Here, the usability of threshold signatures is evaluated for embedded devices.

The Internet of Things (IoT) application architecture enables new business models and unlocks new value in the existing markets thanks to a massive number of low-cost connected devices, which may be attached to objects (a.k.a. Things) to collect data, track assets, actuate and so on. However, the same features that act as enablers also create a tension in terms of security. The low cost means the IoT devices often use embedded HW platforms with medium to low security and are exposed to attacks such as remote software exploits (making use of the connectivity) and physical attack ranging from connected debugger, through PCB tampering, to side channel attacks (making use of the physical access available to the attacker). It is thus hard to prevent a motivated attacker from extracting key material from a device, including long-term keys. An attacker may impersonate the compromised device and push malicious data upstream.

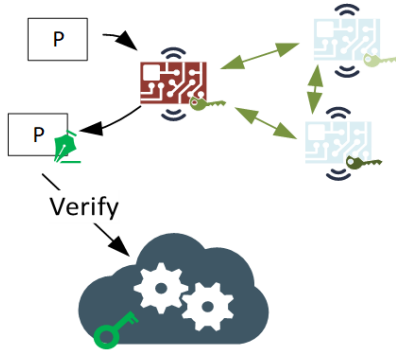


Figure 1: A distributed signature protocol with threshold $t=3$. The signing device (red) produces a signature with help of two companion devices, each of them possessing a private signing key share.

The risk related to these scenarios can be managed using threshold cryptography, where secret keys are cryptographically split into shares, and conventional cryptographic primitives, such as secret-key authenticated encryption or digital signatures are transformed into interactive protocols, such that each encryption, resp. signing requires a certain number of devices possessing the key shares to interact. In such a setup, whenever a device needs to sign a payload, it will run an interactive protocol with peer devices, such that their total number needs to be equal to a threshold t . The threshold t is configurable and allows for a trade-off between security and overhead; the higher t , the more devices must be compromised in a successful attack, but also more devices must participate in a distributed signature. The

procedure is illustrated in Figure 1. CSEM has evaluated [1,2] and confirmed a practical use of distributed encryption to harden the firmware update over the air [3], for example. For other use cases, however, the symmetric-key nature of distributed encryption is not ideal. For example, when data originating from an IoT fleet must be authenticated by other parties than the fleet owner, digital signatures are much better suited, as the public verification may be freely disseminated, while the private signing shares are distributed among the devices.

To assess the usability of threshold signatures in embedded platforms, two constructions have been evaluated. The first is a threshold variant of the BLS signature scheme [4]. Internally, BLS uses the so-called bilinear pairing, a special algebraic structure over certain elliptic curve groups yielding a construction that is easily transformed into a threshold version, albeit with a complex signature verification, which is not a major setback for the envisaged use case. The embedded implementation is based on the blst library [5]. The other scheme is a dedicated threshold signature design FROST [6], implemented over the elliptic curve secp256k1. The embedded implementation is based on the secp256k1-frost library [7]. Both libraries were adapted for integration with CSEM's real-time operating system $\mu 111$ and benchmarked on the Nordic nRF52840 devkit running at 64 MHz. The results are summarized in Figure 2.

	BLS	FROST
Pub. k./priv. k. /sig. [B]	48/96/32	33/64/32
Signing time [s]	$0.94 + 0.4 \cdot t$	$0.036 + 0.11 \cdot t$
Communication [B]	196	168
Comm. rounds	A broadcast by each device	2 (or 1 with pre-processing)

Figure 2: Performance of threshold signatures on nRF52840 @ 64 MHz.

CSEM benchmarked the proof-of-concept implementations of BLS and FROST threshold signatures and confirms that they are usable in embedded systems, although the computational and communication overhead requires optimizations.

* HEIG-VD, Switzerland

[1] D. Vizár, "Distributed encryption for robust data confidentiality and integrity for IoT", CSEM Scientific and Technical Report (2021) 48.

[2] R. Müller, "Robust IoT security with threshold cryptography", bachelor's thesis, HEIG-VD (2021).

[3] D. Vizár, M. I. Ben Salah, Q. Liang, "Securing embedded firmware update over the air with distributed encryption", CSEM Scientific and Technical Report (2023).

[4] D. Boneh, B. Lynn, H. Shacham, "Short signatures from the Weil pairing" ASIACRYPT (2001).

[5] <https://github.com/supranational/blst>

[6] C. Komlo, I. Goldberg, "FROST: flexible round-optimized Schnorr threshold signatures", SAC (2020)

[7] <https://github.com/bancaditalia/secp256k1-frost>