I missed something

In the last blog I rather carelessly stated that “Note that the Pico uses the ARM M0+ processor and I am unaware of any optimized assembly language implementation for this particular processor”. I should have looked harder.

The redoubtable Thomas Pornin has a lovely 3000 line pure ARM Cortex-M0+ constant time assembly language implementation of the x25519 function.

Thanks to our flexible architecture it was easy to integrate it into a composite SAL, which uses hardware for secp256r1 support, Pornin’s assembly language for x25519, and MIRACL for everything else. See **`tls_sal_mhp.xpp`**. In fact just 10 lines of code needed to be added to an existing SAL, and the Pornin **x25519.S** file dropped into the project.

The previous timings on the Arduino RP2040 (using MIRACL for x25519) were

```
10:53:27.994 -> Cryptography by MIRACL Core + ECC608A hardware
10:53:33.423 -> SAL supported Key Exchange groups
10:53:33.423 ->     X25519
10:53:34.548 ->         Key Generation (ms)= 139
10:53:35.626 ->         Shared Secret (ms)= 139
10:53:35.626 ->     SECP256R1
10:53:36.704 ->         Key Generation (ms)= 81
10:53:37.735 ->         Shared Secret (ms)= 50
10:53:37.735 ->     SECP384R1
10:53:43.447 ->         Key Generation (ms)= 1427
10:53:49.124 ->         Shared Secret (ms)= 1427
```

Using instead Pornin’s assembly language implementation of x25519, the timings are

```
19:44:20.913 -> Cryptography by MIRACL Core + ECC608A hardware + Pornin
19:44:20.913 -> SAL supported Key Exchange groups
19:44:20.913 ->     X25519
19:44:21.944 ->         Key Generation (ms)= 28
19:44:22.977 ->         Shared Secret (ms)= 28
19:44:22.977 ->     SECP256R1
19:44:23.999 ->         Key Generation (ms)= 86
19:44:25.046 ->         Shared Secret (ms)= 58
19:44:25.046 ->     SECP384R1
19:44:29.633 ->         Key Generation (ms)= 1146
19:44:34.221 ->         Shared Secret (ms)= 1145
```

X25519 is now 5 times faster! Not only that, the assembly language for x25519 is faster than the hardware implementation of secp256r1. That’s pretty impressive. This illustrates the power of a flexible SAL which can source its cryptography from a range of resources. The overall impact for TLS1.3 on the RP2040 is that it now runs at close to optimal speed.

There is a lot of very good high quality open source software out there. We must make use of it!

Interesting paper - **“**Requirements for Post-Quantum Cryptography on Embedded Devices in the IoT” - https://csrc.nist.gov/CSRC/media/Events/third-pqc-standardization-conference/documents/accepted-papers/atkins-requirements-pqc-iot-pqc2021.pdf

From the conclusions - “We encourage NIST to consider devices smaller than the ARM Cortex M4 when finalizing its PQC candidates. Specifically, they should consider the resource constraints of a Cortex M0, if not even smaller processors.” Couldn’t agree more!