Engineering Cryptographic Software

Elwin Tamminga
s1013846

Lucas van der Laan
s1047485

1 ChaCha20

We followed the following steps to optimize ChaCha20:

- Implement Quarterround in Assembly.
- Implement two full rounds (Basically one instance of the for loop).
- Implement all the rounds after each other.
- Assign all the ‘x’ variables at the end.

We ended up not optimizing much else, except for the fact that we implemented it in assembly. We used all available registers to optimize the assembly code.

Our final speed was 24.920 cycles.

2 Poly1305

The main optimization we did for Poly1305 was changing all functions to radix 2^{26}, except freeze. We did not succeed in changing freeze to radix 2^{26}, as we did not have enough time left to figure it out. This meant that we converted everything back at the end to radix 2^8, just for the freeze function. We also unwrapped the loops for some more optimizations.

Our final speed was 21.345 cycles.

3 ECDH

This was a tough one. The first thing we did was remove the timing leak in “crypto_scalarmult” where an if statement was depending on a secret. We made it so that we now add the neutral group instead of doing nothing, initially this increased the cycle count by ~20 million. We also replaced the if statement in “fe25519_cmov”.

We then figured out that the “fe25519_mul” function was used basically everywhere so we set to optimize this next. We started by changing the radix from 2^8 to 2^{16}, which halved the array in size. We then changed “reduce_mul” as well, as this not had to apply to the new radix. At the end of the “fe25519_mul” function we then change back to radix 2^8, as to not have to modify the entire functionality of ECDH to radix 2^{16}, due to time constraints.

We also optimized “fe25519_add” in the same way we did to “fe25519_mul” and optimized the “fe25519_double” function to now use “fe25519_add”. The last optimization was optimizing the “fe25519_sub” function by removing the unnecessary ‘t’ array.

Our final speeds were:

1. crypto_scalarmult: 28.362.971 cycles.
2. crypto_scalarmult_base: 29.850.494 cycles.