Universidade do Minho

Mestrado em Engenharia Informática

Tecnologia Criptográfica

Auditing a Poly1305 MAC implementation in Jasmin for x86

Miguel Miranda Quaresma
A77049

December 28, 2018

Abstract

Poly1305 is a one time authenticator that generates a message authentication code for a given input and secret key using, for that purpose, a similar mechanism to universal hashing. Jasmin is a framework for developing high performance and high assurance cryptographic software. The current works aims to audit an implementation of the Poly1305 MAC using the Jasmin framework. Firstly the Poly1305 MAC is described from a high(abstraction) level, followed by an in-depth analysis of the Jasmin implementation of the algorithm. The work concludes with an auditing/verifcation of the premisses/assumptions that were made for the implementation in a mathematical manner.

1 Introduction

Message authentication codes play a major role in guaranteeing the authenticity and integrity of data being sent across an untrusted channel. There are many crypto-primitives that work as MAC's and, for a long time, HMAC(Hash-MACs) were preferred over other MACs due to their performance, with other primitives such as the ones based on universal hashing being discarded. This was further reinforced by the introduction of assembly instructions that performed hash functions directly in hardware([2]), such as Intel's instruction for SHA-256: `sha256rnds2`. The development of cryptoprimitives such as Poly1305 MAC using Jasmin, a framework for developing high performance and high assurance cryptographic software [1], allowed this tendency to be reversed by obtaining highly performant implementations with relative ease.

2 Poly1305 explained

Poly1305 is a message authentication code(MAC) that guarantees integrity and authenticity of messages. It achieves so similarly to a universal hash function, evaluating a given message over a polynomial and using the result as a MAC. Additionaly, Poly1305 evaluates this polynomial over a prime field, from 0 to $2^{130} - 5(=\text{p})$, where the name (Poly**1305**) stems from. Thus, Poly1305 can be expressed by the following expression:

$$mac = (m1 * r^4 + m2 * r^3 + m3 * r^2 + m4 * r + k) mod p$$

where mi is the ith block of the message. Being an authenticator, Poly1305 uses a 256 bit secret, derived via a Password-Based Key Derivation Function(PBDKF), that is then split up into two blocks of 128 bits each, the first block being used for the parameter k, and the second for the parameter r. Poly1305 works by breaking the input in 16 byte blocks, appending each one with a 1 byte(00000001) to prevent forgery. It then proceeds to apply the following algorithm/formula:

```
h = 0
for block in blocks:
    h += block
    h *= r
h+=k mod 2^130-5
```

where:

- `block` is a 17 byte block from the message
- `r` and `k` are the 128 bit values derived from the 256 bit key used by Poly1305

One of the consequences of the presented algorithm is the fact that, after a certain number of blocks, the variable h will overflow due to the successive multiplications therefore, it is necessary to perform the calculation via modular arithmetic. To perform this in an efficient way the prime $2^{130} - 5$ was chosen to perform school book multiplication futhermore carry propagation is delayed to allow for fast(er) modulo reduction.

2.1 (Schoolbook) Multiplication

After adding the message block mi to the current h value, h is multiplied with r using schoolbook multiplication, as such r and h are divided in five 4 byte (32 bit) blocks and multiplied with eachother:

	m5	m4	m3	m2	m1
*	r5	r4	r3	r2	r1

3 Conclusion

References

[1] José Bacelar Almeida et al. "Jasmin: High-Assurance and High-Speed Cryptography". In: Oct. 2017, pp. 1807–1823. DOI: `10.1145/3133956.3134078`.

[2] Sean Gulley et al. *Intel ® SHA Extensions New Instructions Supporting the Secure Hash Algorithm on Intel ® Architecture Processors.* 2013. URL: `https://software.intel.com/sites/default/files/article/402097/intel-sha-extensions-white-paper.pdf`.