ECOLE
POLYTECHNIQUE
DE BRUXELLES

ULB

UNIVERSITÉ LIBRE DE BRUXELLES

Implementation of High-Level Cryptographic Protocols using a SoC Platform

Mémoire présenté en vue de l'obtention du diplôme
d'Ingénieur Civil en informatique à finalité spécialisée

Quentin Delhaye

Directeur
Professeur Frédéric Robert

Superviseur
Sébastien Rabou (Barco Silex)

Service
BEAMS

Année académique
2014 - 2015

Abstract

by Quentin Delhaye, Master in Computer Science and Engineering, Professional Focus, Université Libre de Bruxelles, 2014–2015.

Implementation of high level cryptographic protocol using a SoC platform

This is an abstract.

Résumé

par Quentin Delhaye, Master en ingénieur civil en informatique, à finalité spécialisée, Université Libre de Bruxelles, 2014–2015.

Aknowledgements

Thanks a bunch of people here : - Frédéric Robert for his support - Sébastien Rabou for his insight - Batien Heneffe for his extensive help

Contents

Abstract **i**

Résumé **ii**

Aknowledgements **iii**

Contents **iv**

1 Introduction **1**
1.1 Challenge 1
1.2 Network security 1

2 Technical background **2**
2.1 Operating system 2
2.2 Cryptography 2
2.3 Network and VPN implementation 2

3 Presentation **3**
3.1 Experimental setup 3

4 Implementation **4**

5 Conclusion **5**
5.1 Future work 5

A Toolchain **8**

B Cross-compilation **9**
B.1 OpenSSL 9
B.2 OpenVPN 9
B.3 nginx 9
B.4 Strongswan 9

B.5 kernel 9

C Stuff **10**

Bibliography **12**

Chapter 1

Introduction

1.1 Challenge

1.2 Network security

Chapter 2

Technical background

This chapter will address the technical ground inherent to this work: first the operating system, followed by the cryptography and networking.

2.1 Operating system

2.2 Cryptography

2.2.1 Asymetric cryptography

2.2.2 Symetric cryptography

2.3 Network and VPN implementation

There exist several major implementations of VPN: SSL, IPSec and PPTP. The later was developped by a vendor consortium and proposed in the RFC 2637 and will not be discussed further.

2.3.1 SSL/TLS

2.3.2 IPSec

Chapter 3

Presentation

Here we talk about the protocols, the platform. Show The OS stack (kernel/user)

3.1 Experimental setup

The experimental environment is build around a standard x86 host and an ARM Cortex-A9 alongside an Altera Cyclone V FPGA as the target.

3.1.1 x86 host

OS Ubuntu 12.04 LTS, kernel 3.16

CPU Intel Core-i3 ... (two logical core out of four)

RAM 1.5GB DDR3

3.1.2 Altera Socrates SoCFPGA

OS Yocto project, kernel 3.14

CPU Dual core ARM Cortex-A9, 800MHz

RAM ...GB DDR3

FPGA Altera Cyclone V

Chapter 4

Implementation

Chapter 5

Conclusion

5.1 Future work

Although the present work presents some promissing results, the implementation can certainly be improved in several ways and some further experiments should be conducted.

Driver improvement The driver of the BA411E can be made less ressources hungry by improving the initialisation of the descriptors and their linking, but the gain would not be significant enough to justify the time investment at this point. A better alternative would be to avoid descriptors altogether by modifiying the interface with the IP so that it can use the scatterlist directly. We would then spare a lot of DMA mapping instruction and thus some precious cycles on the software side.

As we already remarked in **??**, the use cases involving IPSec were conducted using a previous revision of the driver still activelly polling the IP for its results. A better and cleaner way to proceed is to use interruption routines, as shown in **??**. However, the kernel does not support their current implementation and panics upon usage. If one were to be willing to spend the time replacing the active polling by clean asynchronous interruptions, he should be aware of the overhead imposed by an interruption. In some cases, when the operation is just a few clock cycle long for the IP, an active polling could still be the better way to go. A more thorough comparison of the mutual trade-off deserves some investigation, and as a starting point, the packet size could be treated as a branching point between the two solutions.

Registering public key verification with the crypto API As we saw in **??**, the driver is already capable of offloading a large portion of public key operations to the IP, but only with very specific libraries at the time being – openSSL in our case. The next step is to register the very same operations with the crypto API so it can be used without having to rely on a custom openSSL engine.

Conditional offloading in cryptodev The figure **??** clearly shows a threshold on the packet size from which the hardware has a clear advantage, and below which the user mode software implementation is to go for. Using this tipping point, one could set a conditional branch as shown in listing 5.1

```
int some_function() {
  if(packet_size < 1024*1024) {
    callback_function();
  }
}
```

Listing 5.1: cryptodev conditional offloading

He should however be aware that as the tipping point is around 1024kB, the performance for a network application should very close to those of a full software implementation, knowing that the ethernet frame size, the MTU, is set by default at 1500kB.

Disk encryption As the hardware is better used with larger blocks of data, disk encryption could be an interesting application to look into.

Cryptographic libraries OpenSSL is not the only cryptographic library available; GnuTLS is also a very popular alternative and supports cryptodev engines too.

However, one library definitely worth to keep an eye on is mbed TLS, formally known as PolarSSL, recently bought by ARM [1]. We can expect the future releases of this library to be more optimized for ARM platforms, and maybe the software footprint and overhead to be reduced.

Cryptodev If patches adding the GCM support to cryptodev have already been released, those are not compatible with Barco Silex' driver. Adapting the interface would open the GCM hardware offload to the whole user space applications park.

MAC offloading While the symetric and asymetric encryption ciphers IPs are usable from the operating system, the IP computing MACs does not have a usable driver yet. Wherever there is encryption, authentication is also needed. As such, any real day-to-day use case can not be fully offloaded to hardware yet, even if some tricks and patches allowed us to bypass this requirement. The implementation of the GCM mode, combining encryption and authentication, showed us that stopping relying on the software implementation of MACs would be a huge step forward.

Appendix A

Toolchain

Talk here about linux linaro and shit. This does not need to be long, just explain which version was used, what environment variable to export.

See if this does not enter in conflict with the "cross-compilation" appendix.

Appendix B

Cross-compilation

B.1 OpenSSL

B.2 OpenVPN

B.3 nginx

B.4 Strongswan

B.5 kernel

Force kernel version
Change list of modules (show the final version)

Appendix C

Stuff

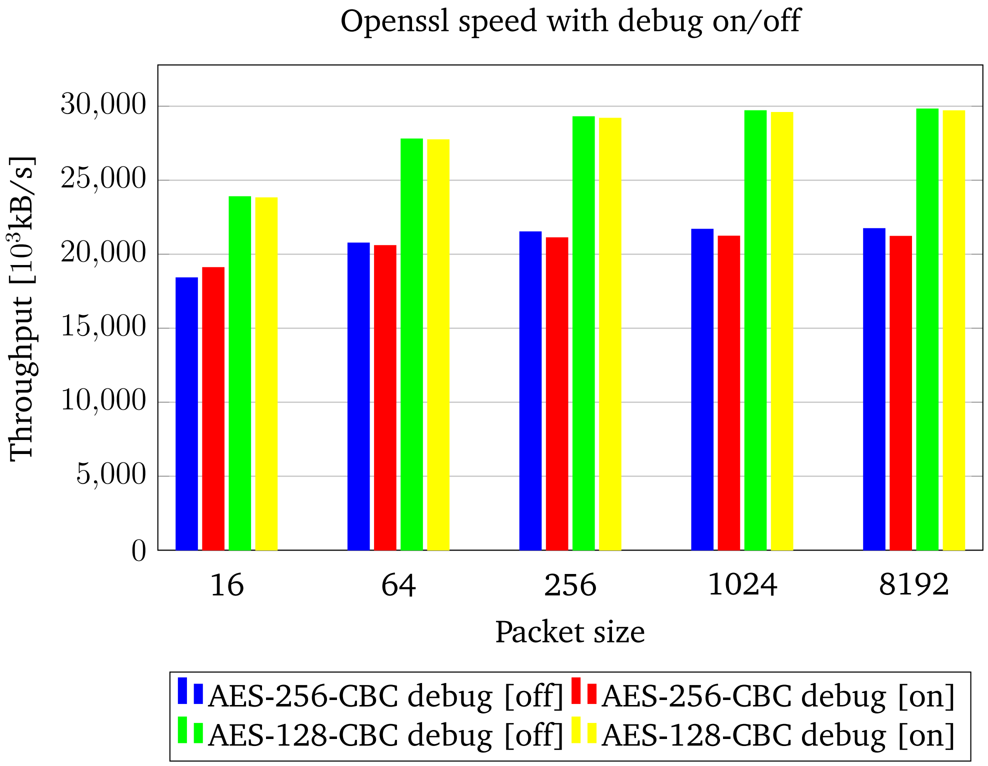

Figure C.1: Software benchmark of Openssl speed for AES mode CBC, with 128- and 256-bit keys, debugging flags (de)activated at compilation (`-fno-inline -g -marm`).

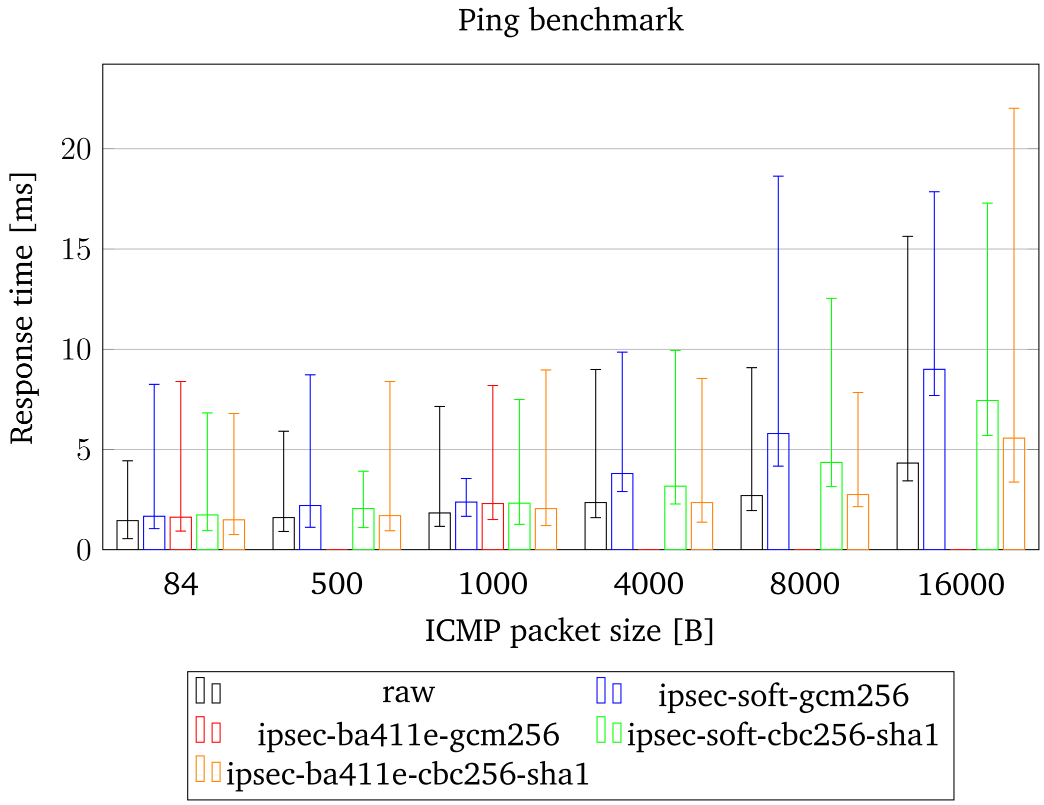

Figure C.2: Ping min/avg/max response time for different packet sizes using IPsec. For each packet size, 1000 requests were flooded to the board, that is "*outputs packets as fast as they come back or one hundred times per second, whichever is more*", according to the `ping` command manual.

Bibliography

[1] ARM. *ARM buys Leading IoT Security Company Offspark as it Expands its mbed Platform*. February 2015. `http://www.arm.com/about/newsroom/arm-buys-leading-iot-security-company-offspark-as-it-expands-its-mbed-platform.php`.

[2] Hugo Krawczyk. The order of encryption and authentication for protecting communications (or: how secure is ssl?). Cryptology ePrint Archive, Report 2001/045, 2001. `http://eprint.iacr.org/`.

[3] NAFSMA CPD Institute. Adidas-Reebok Merger. North America Financial Services and Management Association Case study, 2011.

[4] Kenneth G. Paterson. A cryptographic tour of the IPsec standards. *Information Security Technical Report*, 11(2):72 – 81, 2006. ISSN 1363-4127.

[5] J. Viega and D. McGrew. The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP). RFC 4106, RFC Editor, June 2005. URL `https://tools.ietf.org/rfc/rfc4106.txt`.

[6] Christos Xenakis, Nikolaos Laoutaris, Lazaros Merakos, and Ioannis Stavrakakis. A generic characterization of the overheads imposed by IPsec and associated cryptographic algorithms. *Computer Networks*, 50(17):3225 – 3241, 2006. ISSN 1389-1286.