Hochschule Offenburg
offenburg.university

ivESK Institut für verlässliche Embedded Systems und Kommunikationselektronik

E+I Elektrotechnik und Informationstechnik

Extension and Integration of an Abstract Interface to Cryptography Providers

Report

Steve Wagner
EI-3nat

February 14, 2016

Prof. Dr. Axel Sikora
Dipl.-Phys. Andreas Walz
Laboratory for Embedded Systems and Communication Electronics
Hochschule Offenburg

Statutory declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Offenburg, _____ _____

Date Signature

Abstract

Contents

1 Introduction

1.1 Cryptography

Cryptography uses mathematical techniques for the security of datas transmitted over an insecure network.
Cryptanalysis is the complementary of the cryptography with the focus on the defeat of the cryptographic mathematical techniques.

The security of the information is definied into:

- Confidentiality or privacy
  No one, except whom is intended, can understand the transmitted datas
- Integrity
  No one can alter the transmitted message without the alteration is being detected
- Authentication
  The sender and the receiver can identify the destination of the datas and identify themself
- Non-repudiation
  The sender cannot deny at a later stage the tranmission of the datas

The mathematical cryptographic techniques are grouped into several algorithms:

- Hash algorithm
- Signature algorithm
- Symmetric cipher algorithm
- Asymmetric cipher algorithm

1.1.1 Hash algorithm

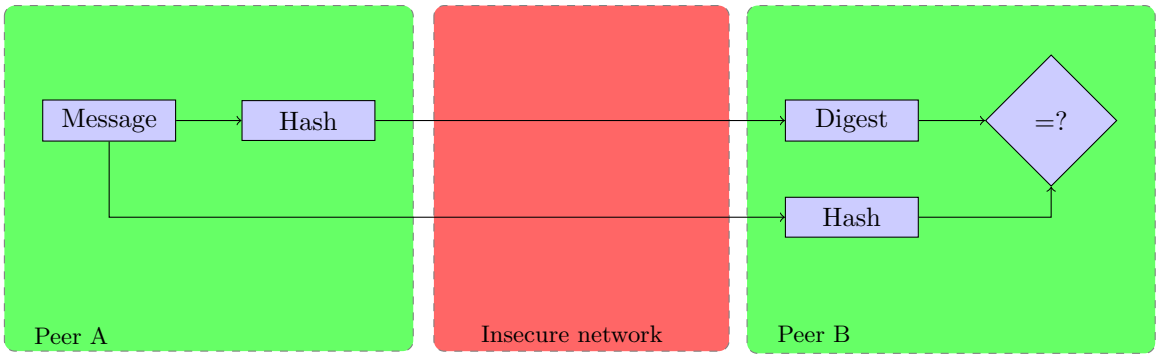

Figure 1.1: Scheme of an hash operation

A cryptographic algorithm is considered pratically impossible to invert, meaning that impossible to recreate the input data (message) with the digest (output of the hash).

Tha main properties of a hash function are:

- it's quick to compute the digest for any message
- it's infeasible to generate a message from its digest
- it is infeasible to modify a message without changing the digest
- it is infeasible to find two different messages with the same digest.

On Figure 1.1, the message is hashed and the result (digest) is sent to the other peer with the original message.
Then the other peer hashs the message too (with the same algorithm) and compares the two digests to know if the message has been changed during the transmission.
This is the princip of integrity.

1.1.2 Signature algorithm

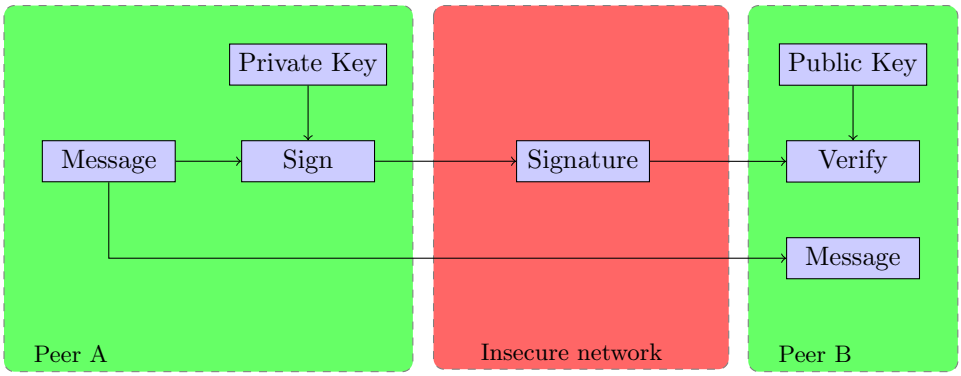

Figure 1.2: Scheme of a signature operation

A digital signature is a mathematical princip to demonstrate the authenticity of a message.
It infeasible to generate the original message with the signature.

A valid digital signature allows to be sure that the incoming message comes from the peer we are communicated with and not from someone else (authenticity).
With digital signature the sender cannot deny having sended the message (non-repudiation).
It allows too to be sure that the message has not been corrupted during the transmission (integrity).

The princip of a digital signature is shown figure 1.2.
The message is signed with the private key (no one has this key too) and the result (signature) is sent to the other peer (peer B).
The public key of the peer A has already be sent previously. With this key, peer B can verify the signature (only with the public key of peer A) and be sure that the message sends with it comes from peer A and not from someone else.

1.1.3 Symmetric cipher algorithm

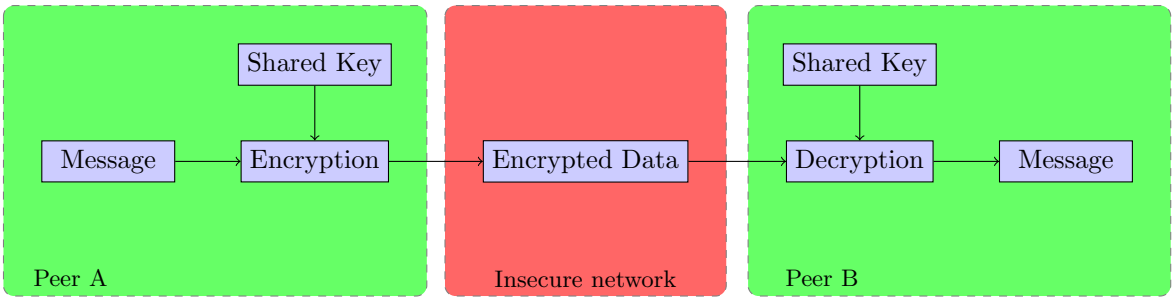

Figure 1.3: Scheme of a symmetric cipher operation

Symmetric-key algorithms are algorithms for cryptography that use the same cryptographic keys for both encryption of plaintext and decryption of ciphertext in a communication.
The key is often named shared secret key.

There is two kind of symmetric encryption:

1. Stream ciphers, which the encryption is done only for one digit (typically bytes) of a message at a time.
2. Block ciphers, which take a number of bits and encrypt them as a single unit, padding the plaintext so that it is a multiple of the block size.

Figure 1.3 represents the process for encryption and decryption with a symmetric key.

The shared key has already been exchanged between the two peers.
Peer A encrypts the message (plaintext) with this key and sends the result (the encrypted data or ciphertext) to peer B.
Than peer B uses the same key but to decrypt the encrypted data (ciphertext) and then reads the plaintext sent by the peer A.

No one who doesn't have this key can understand this message over the insecure network represents in red figure 1.3.

1.1.4 Asymmetric cipher algorithm

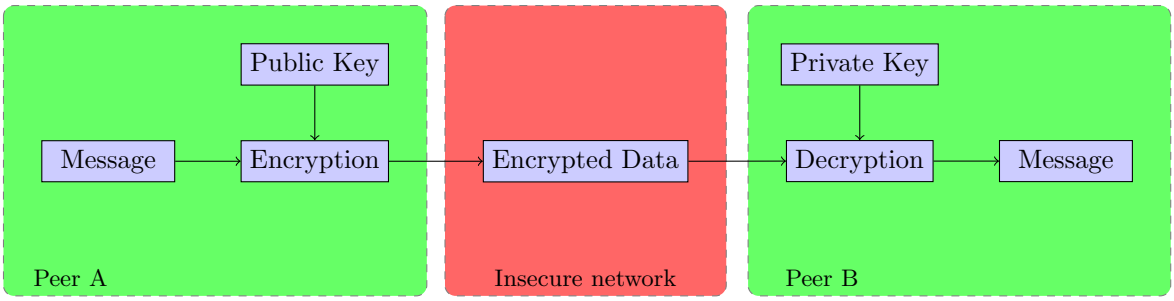

Figure 1.4: Scheme of an asymmetric cipher operation

Asymmetric algorithm is a method for encryption and decryption of messages with public and private keys.
One peer generates the private and public key together, keep the private key (meaning that he is the only one to have it) and sends the public key to every one he wants to communicate with.
With the public key everyone can encrypt a message, which no one can understand through the insecure network and no one can decrypt it, except this one who has the private key.

Figure 1.4 shows the princip of encryption and decryption with asymmetric algorithms.
Peer B is this one who has generated the public and private keys and has already sent the public key to peer A.
Peer A encrypts the message with the public key of peer B. This message is than sends to peer B over the insecure network.
Thanks to the private key can peer B decrypts the message of peer A.

Thanks to the princip of private and public keys, asymmetric algorithm allows privacy (like symmetric algorithm) but integrity of data too, because only this one who creates the keys has the private key for decryption.

1.1.5 Diffie-Hellman

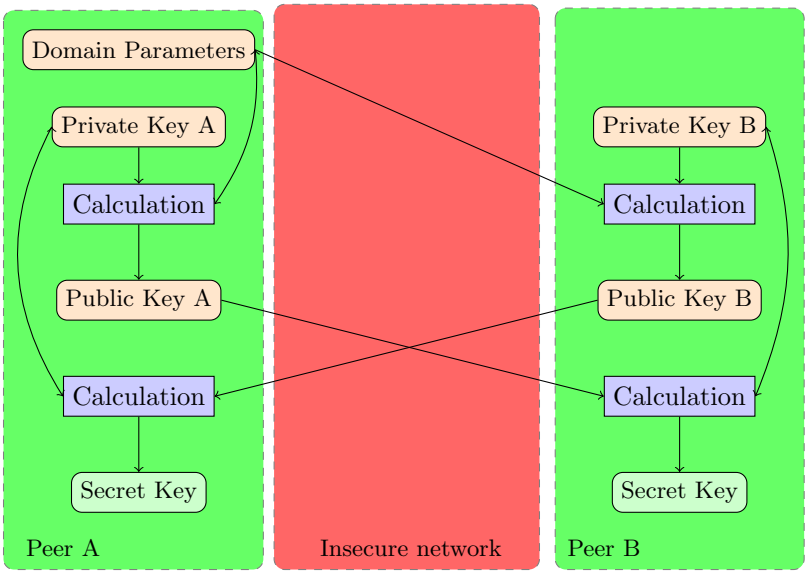

Figure 1.5: Scheme of a Diffie-Hellman operation

Diffie-Hellman Key Exchange etablishes a shared secret key between two peers that can be used for secret communication for exchanging data over an insecure network.
The princip of Diffie-Hellman Key Exchange is to begin with asymmetric keys and to finish with symmetric key.
It makes sure that the both parties participate in the generation of the symmetric key.

Figure 1.5 shows the princip of the Diffie-Hellman key exchanges.
Peer A creates the domain parameters and a private key only (not the public).
With the domain parameters and the private key can peer A calculates his public key.
He sends than the domain parameters and his public key to peer B.
Peer B generates a private key too and calculates his public key with the domain parameters from peer A and its own private key.
Peer B sends than his public key to peer A.
To finish, each one calculates the shared secret (which is the same for the twice) with the public key of the other peer and it's own private key.

The shared secret is the same for the two peers and can be used for encryption and decryption.

1.2 SSL/TLS protocol

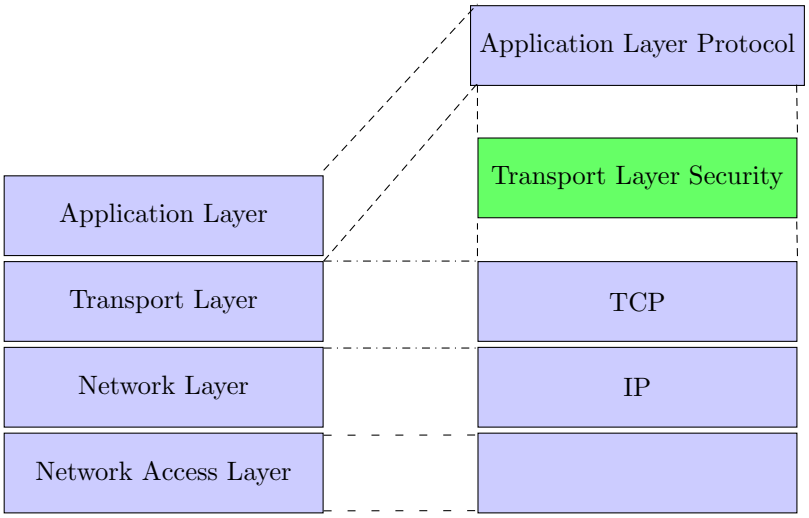

Figure 1.6: Placement of TLS in OSI model

Transport Layer Security (TLS) is a client/server protocol that provides different basic security services for the communication between peers:

- Authentication (both peer and data origin authentication) services
- Connection confidentiality services
- Connection integrity services (without recovery)

This security layer is situated between the transport and the application layer on the OSI model (see figure 1.6)

This security protocol is often uses in:

- E-commerce website for secured transaction and client authentication acces
- Remote access
- Web browsers to browse the Internet
- Simple Mail Transfer Protocol (SMTP)
- Virtual Private Network (VPN)

2 Motivation

In the institut of reliable Embedded Systems and Communication Electronics (ivESK) is a project named emb::TLS which has the goal to use the TLS protocol (see chapter 1.2) in embedded systems.

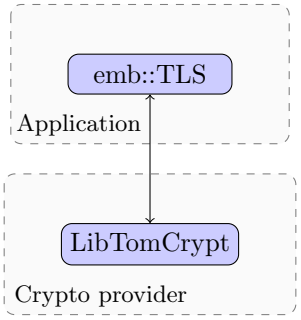

Figure 2.1: Scheme of emb::TLS's project

For this, a cryptographic provider (LibTomCrypt) is used for the part of cryptographic calculation needed in the application.
This cryptographic provider is an open-source cryptography software library.

Problems with this implementation is that only LibTomCrypt is supported as cryptographic provider, meaning that no other libraries can be used without changing the complet implementation for emb::TLS.

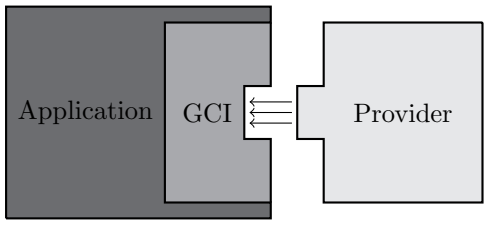

Figure 2.2: Goal of the new implementation

The goal of the project is therefore to create an interface, a Generic Cryptographic Interface (GCI), to have a base of the existing cryptography services and to have the possibility to easily add other providers only by changing some lines in the interface, instead of the complet application.
Through to this new interface new other cryptographic algorithms may be easier to add in the interface and to use in the application.

As shown in figure 2.2, the interface is implemented in the application and nothing has to be changed, except if other mathematical calculation has to be changed in a cryptography algorithm, like another kind of hash.

3 Design

The requirements for this Generic Cryptographic Interface (GCI) are listed below:

1. No hidden states shall be used in the interface, meaning that the behavior of functions should only be affected by the input parameters.
   All parameters written in input of the function will be used for the cryptographic algorithm and nothing else.
2. Different cryptographic providers may be used for the cryptographic calculation.
   That could be open-source cryptographic software libraries or hardware-based cryptographic modules.
3. Interaction between the provider and the application shall be enable for the key by a key management services, meaning that the key generated by the provider should be stored to have the possibility to use it by the application (like sending it to another peer) and keys coming from another peer shall be stored too, meaning that this key shall be used by the provder.
   to interact between the application and the provider.

4 Generic Cryptographic Interface

4.1 Cryptographic services

4.1.1 Hash

4.1.2 Generate key pair

4.1.3 Signature

4.1.4 Symmetric Cipher

4.1.5 Asymmetric Cipher

4.1.6 Diffie-Hellman

4.1.7 Random number generator

4.2 Context

4.3 Key management

5 Cryptography in the TLS protocol

5.1 Handshake Protocol

5.2 Cryptographic parts in the Handshake Protocol

6 Implementation

6.1 Interface for an application

6.2 Provider for the interface

7 Results

7.1 embetterTLS as client

7.2 embetterTLS as server

8 Conclusion

8.1 Achieved work

8.2 Future work

Bibliography

Books

[1] Christof Paar and Jan Pelzl. *Understanding cryptography: a textbook for students and practitioners*. Springer Berlin Heidelberg, Berlin; Heidelberg [u.a.], 2. corr. printing edition, 2010.

[2] Ph.D. Rolf, Oppliger. *SSL and TLS: Theory and Practice*. Artech House, eSECURITY Technologies; Beethovenstrasse 10; CH-3073; Gümligen; Switzerland, 2009.

Internet

[3] Wikipedia. Symmetric-key algorithm, December 2015. URL: `https://en.wikipedia.org/wiki/Symmetric-key_algorithm`.

[4] Wikipedia. Cryptographic hash function, January 2016. URL: `https://en.wikipedia.org/wiki/Cryptographic_hash_function`.

[5] Wikipedia. Diffiehellman key exchange, February 2016. URL: `https://en.wikipedia.org/wiki/Diffie-Hellman_key_exchange`.

[6] Wikipedia. Digital signature, January 2016. URL: `https://en.wikipedia.org/wiki/Digital_signature#How_they_work`.

[7] Wikipedia. Public-key cryptography, February 2016. URL: `https://en.wikipedia.org/wiki/Public-key_cryptography`.

Appendices

Appendix A

Documentation Interface

This documentation lists all functions use in the Generic Cryptographic Interface (GCI) and explains step by step the working of each cryptographic services definied in the interface.

List of Figures