FACULTY OF INFORMATION TECHNOLOGY CTU IN PRAGUE

Assignment of master's thesis

Title:	Authentication Effectiveness in Vehicle Unified Diagnostic Services
Student:	Bc. Jakub Weisl
Supervisor:	Ing. Jiří Dostál, Ph.D.
Study program:	Informatics
Branch / specialization:	Computer Security
Department:	Department of Information Security
Validity:	until the end of summer semester 2023/2024

Instructions

Modern car design typically includes various Electronic Computing Units (ECUs) connected through communication buses and computer networks. ECUs also oversee and report the technical status of different car units, ensuring proper communication between them. They process the incoming signals and deploy actions based on gathered information, e.g., powertrain control or brake assist.

There is also a diagnostic unit present. It is responsible for reporting the health status of the car and communicating with service devices. It employs a server-client communication model using the CAN bus. As an overlay layer over the CAN bus, there is a Unified Diagnostic System (UDS) protocol, defined by ISO 14229-2020. All the functions of the UDS protocol are grouped into services. The main focus of this thesis is the authentication service defined in chapter 10.6 of the previously motioned ISO standard.

The authentication service offers two authentication schemes. The first is authentication using PKI and the second is challenge-response authentication using symmetric cryptography. The goals of this diploma thesis are:

1. Design a system for measuring the effectiveness of different authentication schemes.
2. Create a testing environment emulating communication between the diagnostic unit and client station based on ISO 14229-2020 standard.
3. Implement different authentication mechanisms. Implementations must cover both

---

Electronically approved by prof. Ing. Róbert Lórencz, CSc. on 4 January 2023 in Prague.

symmetric and asymmetric cryptography.
4. Assess the effectiveness of the implemented authentication mechanisms using the designed system and consider further strengths and weaknesses of implemented options.

Electronically approved by prof. Ing. Róbert Lórencz, CSc. on 4 January 2023 in Prague.

Master's thesis

AUTHENTICATION EFFECTIVENESS IN VEHICLE UNIFIED DIAGNOSTIC SERVICES

Bc. Jakub Weisl

Faculty of Information Technology
Department of Information Security
Supervisor: Ing. Jiří Dostál, Ph.D.
May 9, 2024

Contents

List of Figures

List of Tables

List of code listings

I would like to thank to mister Dostál for his patients during creation of this thesis. I would like to also thank to all my teachers who thought me during the time spent at this faculty and who showed me direction, which I am going to go in upcoming years.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis. I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular the fact that the Czech Technical University in Prague has the right to conclude a licence agreement on the utilization of this thesis as a school work pursuant of Section 60 (1) of the Act.

In Prague on May 9, 2024 .....

Abstract

This thesis tries to compare authentication effectiveness in form of volume of transferred data and duration of two kinds of authentication schemes, the challenge response and PKI authentication as defined in Unified Diagnostic Service. Part of this thesis is developed software simulating the authentication with different crypto algorithms (RSA, ECC, HMAC) and different throughput, which is achieved by using both, different payload sizes and frequency of sending the frames. The measured data shows that with smaller payloads the computing demands of crypto algorithms are not significant and the volume of transferred data is more important. It also shows the authentication using PKI and ECC crypto algorithm can keep with speed of challenge response authentication and carry the advantages of the PKI.

Keywords UDS, CAN, ISO-TP, automotive, athentication, PKI, effectivity

Abstrakt

Tato práce se snaží porovnat efektivitu ověřování v podobě objemu přenesených dat a doby trvání dvou druhů autentizačních schémat, challenge-response a PKI, jak jsou definována v rámci Unified Diagnostic Services. Součástí této práce je vyvinutý software simulující ověřování s různými kryptografickými algoritmy (RSA, ECC, HMAC) a různou propustností, které je dosaženo jednak použitím různého množství přenášených dat v jednom rámci, tak frekvencí odesílání rámců. Z naměřených dat vyplývá, že při menších objemech dat v rámci nejsou výpočetní nároky kryptografických algoritmů významné a důležitější je objem přenášených dat. Ukazuje se také, že ověřování pomocí PKI a kryptografického algoritmu ECC dokáže udržet rychlost s challenge-response autentizací a zároveň přenáší výhody PKI.

Klíčová slova UDS, CAN, automotive, ISO-TP, autentizace, PKI, efektivita

List of abbreviations

AES Advanced Encryption Standard. 33

BRS Bit Rate Switch. 6

CA Certification Authority. 13, 31, 33

CAN Control Area Network. 1–6, 18–20, 24, 25, 28, 35

CR Challenge response. 29, 31, 33

CRC Cyclic Redundancy Check. 5, 6

CRL Certificate Revocation List. 13

DLC Data length code. 5, 6

ECC Elliptic curve cryptography. 2, 18, 29–31, 33, 35

ECU Electronic computing unit. 1, 3, 7, 18, 21, 33

EOF End of Frame. 5

ESI Error Status Indicator. 6

HMAC Hash-based message authentication code. 14, 18, 23, 27, 29–31, 33

IDM Identity Management. 32

LAN Local area network. 24

NAT Network address translation. 24

OBD On Board diagnostic. 3

PKI Public key infrastructure. 1, 2, 8, 10, 13, 17, 18, 23, 27–29, 31–33, 35

POW Proof of ownership. 8, 18, 23

RRS Remote Request Substitution. 6

RSA Rivest, Shamir, Adleman. 18, 27, 29–31, 33, 35

RTR Remote Transmission Request. 4

SBC Stuff Bit Count. 6

SOF Start of Frame. 4, 5

UDS Unified diagnostic service. 1–3, 7, 8, 18, 23, 24

Chapter 1
Introduction

Modern vehicels are filled with electronics and can comunicate on many protocols. There are electronic devices for user (driver) interaction and then there are de,mi m mvices whose goal is to ensure the smooth and secure driving experience. These devices can have various tasks such running the breaking assistant, managing an automatic gearbox, timing the fuel injection into the engine, and many others. These devices are called the Electronic computing unit (ECU) and to comunicate to each other they use Control Area Network (CAN) protocol. CAN protocol is defined in set of standards ISO 11898. To protocol is in the 2nd (data link) layer of ISO/OSI model. The norm also defines the physical parameters and signaling, so it is possible t say that CAN protocol repsents both physical and data link layer. For the diagnostic and maintenance purposes, there is ECU capable of communicating with external devices connected to the vehicle.

The communication between the vehicle and external unit is in form of client-server comunication, where ECU acts as s server and external devices as a client. To be able to engage in more complex communication they use Unified diagnostic service (UDS) protocol. This protocol is defined by norms ISO 14229, which define the application layer session, layer and integration wit particular lower level protocols. The client interacts with the server through defined functionalities called services. Service is just an envelope to put together sub-functions which represents the real interaction with the server. UDS allows downloading various data, adjusting the vehicle settings, or even uploading new firmware. One of the available services, which have designation 29_{16} is the authentication service, which is the main focus of this thesis.

The today situation is that car manufactures use the authentication service, but they are using the challenge-response option. This thesis tries to compare effectiveness a evaluate use of challenge-response and Public key infrastructure (PKI) based authentication as defined in ISO 14229-1 norm. In order to do that this thesis includes developed simulation of ECU's authentication service. With this particular simulation, the whole experiment is conducted. The main goal was to evaluate if PKI authentication can keep up with challenge-response authentication in terms of effectiveness and whether it is a usable option of authentication because it carries significantly more advantages in terms of identity management and possibility to carry other beneficial information.

There are already works that are looking on current specified authentication protocols, but their main goal is to suggest improvements of the authentication schemes either by suggesting a new scheme or editing the current one. This thesis does not tries to improve the authentication, but instead it tries to compare a evaluate the two current authentication schemes against each other.

The outcome of this work will show if PKI authentication with all its benefits compare to classical challenge-response authentication is a viable option or batter option. This thesis could be the reason why to start implementing PKI authentication instead of challenge-response but

with no necessary change in industry standards.

In first chapter this thesis provides insight into all necessary technologies, protocols and algorithms. Namely the Control Area Network (CAN) protocol which is the data link layer protocol responsible for transfer of all the information in car. Than the ISO-TP protocol which provides possibility to transfer data larger than one CAN frame and ensures functionalities of network and transport layer on ISO/OSI model. The most important part brings the informatoin about Unified diagnostic service (UDS) and allowed authentication schemes as defined in ISO 14229-1 norm. In the last part of the chapter the reader can get insight into used cryptographic algorithms a Public key infrastructure (PKI) model.

The second chapter describes the implementation of authentication service its functions, main structure of the source code and the set up of the testing environment. It also describes the use of the program itself and some of the obstacles of which had to be overcome, particularly regarding lack of documentation for ISO-TP Linux kernel module. In the last section, it describes the chosen parameters of conducted experiment.

The purpose of the third chapter is to present the measured data and to give the their interpretation together with the last chapter.

It can be expected that PKI authentication will not have batter results in terms amount transferred data or speed compare to symmetric crypto algorithm but with use of effective asymmetric crypto algorithm such as Elliptic curve cryptography (ECC) could the PKI authentication achieve similar and comparable results?

Chapter 2
Therotical background

The focus of this chapter is mainly to introduce the technology and bring up the necessary context to them. Namely, this chapter will show, how the CAN protocol and its overlay ISO-TP protocol work and how these two are connected. The way they are used in modern cars and what the UDS is. Regarding UDS the main focus is to explain details of *service* 29_{16} which is the core topic of this thesis. Finally, the choice of particular cryptographic algorithms is reasoned.

Every modern car or other vehicle is equipped with many digital systems which have various purposes from timing fuel injection into the engine across breaking assistants to keeping the the chosen temperature inside a cockpit. These various systems are run by individual ECUs in the vehicle. The modern car can have more than 150 of ECUs.[1] In order to properly work, the ECUs must gather a lot of information from various sensors and exchange information between each other. Information exchange can be done through various proprietary protocols, however in case of communication with the outside word (e.g. diagnostic station, emission control), there are standardized protocols (e.g. OBD 2, UDS). These protocols are application level protocols on ISO/OSI model and are running on top of other protocols. The most widely used protocol for the physical and data link layer is currently the CAN protocol but other such as Ethernet (e.g. Tesla) are starting to emerge.

2.1 CAN bus

CAN protocol is a real time communication protocol, communicating over a central bus. Every node in a network is broadcasting frames to all the other nodes. To be able to receive important message with the smallest delay possible, the messages are priorities base on their ID. 2.1.2 The lower the ID, the higher priority of the given frame.

All the CAN frames are separated by 3 zero bits.[2]

2.1.1 History

CAN bus is a link layer protocol first developed by Bosch company in 1986. Its later version was introduced in 1993 under name CAN2.0. Since then CAN protocol became the standard communication protocol in automotive industry. In 1993 CAN was adopted as international standard ISO 11898. Later, the standard was broaden into series where standard ISO 11898-1 defines the data link layer and ISO 11898-2 defines the parameters for the physical layer.

In 2012 Bosch released CAN-FD, where FD stands for flexible data rate. This upgrade of CAN protocol was standardized in 2015 into previously mentioned ISO norm.[3]

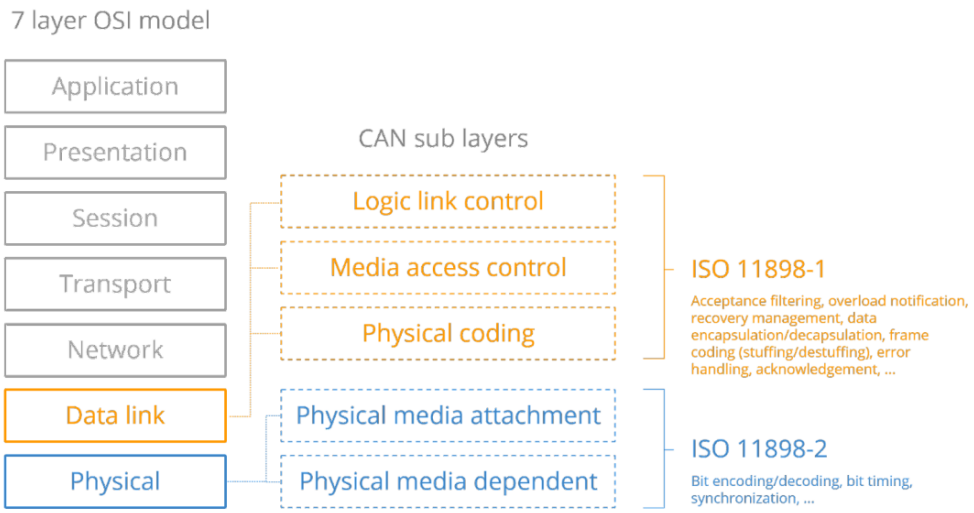

Figure 2.1 CAN bus protocol on ISO-OSI model and its ties to ISO 11989 series [3]

2.1.2 CAN 2.0

High speed CAN2.0 is currently the most used version of CAN protocol. It is used in various types of vehicles for internal communication. There are 2 types of CAN protocols CAN2.0 A and CAN2.0 B, which differ in the length of CAN ID (CAN2.0 A 11 bits, CAN2.0 B accepts both the 11 bit and the 29 bit IDs).[2] The CAN2.0 B is usually used in heavy-duty vehicles, which use the J1939 protocol at higher levels of communication on ISO/OSI model.[3]

Standard CAN communication has 4 types of frames:

- Data Frame - Standard CAN frame carrying data.
- Remote Frame - CAN frame requesting data from another node.
- Error Frame - Error frame is transmitted by any node which encounters an error on a common bus.
- Overload Frame - This frame indicates additional delays in communication due to node overload.

CAN frames can carry up to 8 B of payload and can achieve speed up to 1 Mb/s. Maximal speed can be achieved only with cables length less then 40 meters, then the speed decreases. The structure of a CAN frame is shown in picture 2.2.

The frame carries another 44 bits of information. Their meaning is the following:

- SOF: The Start of Frame is set to 0, to indicate start of a message
- ID: The frame identifier, as mentioned earlier, in CAN2.0 B ID can be either 11 bits or 29 bits. Lower IDs have higher priority.
- RTR: The Remote Transmission Request this flag indicates if a frame requests data (remote frame set to 1) or carries data (data frame set to 0).

Figure 2.2 Structure of CAN2.0 A frame

[3]

- Control: Control has length of 6 bits where first 2 bits are reserved and must be set to 0 and next 4 bits contains DLC indicating the length of a payload.
- Data: Payload.
- CRC: The Cyclic Redundancy Check ensures the integrity of the payload.
- ACK: The ACK slot indicates if the node has acknowledged and received the data correctly. If everything is correct, the receiving node sends back same message, but with ACK bit set to 0.
- EOF: The End of Frame.

As mentioned before communication on common bus is broadcasted from transmitter to all other nodes, which means that only one node can transmit at the time. In the case that bus is idle, any node can transmit its data as long as node with message with higher priority needs to start transmitting. In case more nodes need to transmit information at the same time, the priority of the message is considered. The arbitration process between nodes is called *carrier sense multiple access with collision detection*. The arbitration field is put together from the SOF bit and ID bits. Since the SOF of the same for all messages, the ID field is what decides. The arbitration scheme is the following:

1. All nodes start broadcasting at once.
2. The bus acts as bitwise AND, so if any node is transmitting 0 bit and some other nodes are trying to transmit 1 at the same position. The bit is zeroed.
3. A node reads back written bit from a bus if it is the same as last transmitted bit it keeps transmitting if not, the node stops transmitting and waits for the bus to be idle, or for the the start of new message.

[2]

2.1.3 CAN-FD

CAN-FD is a upgrade of classical CAN2.0 in several ways. It allows variable payload size between 1 and 64 bytes. Because of the limited number of bits in DLC CAN-FD payload can have sizes only 1-8, 12, 16, 20, 24, 32, 48 or 64 bytes. In case when payload does not exactly fit into defined payload size, the padding is used to achieve the closest bigger payload size. The CAN-FD header is bigger than the classic CAN header, so in case of 8 byte payload the overhead using the CAN-FD is greater than with standard CAN.

Another improvement is that CAN-FD is capable of higher speed, theoretically up to 12 Mb/s when sending data. However, the technical specification in 11898-2:2016 sets standards for up to 5 Mb/s. CAN-FD frame has the following structure:

Figure 2.3 Start of a CAN-FD frame

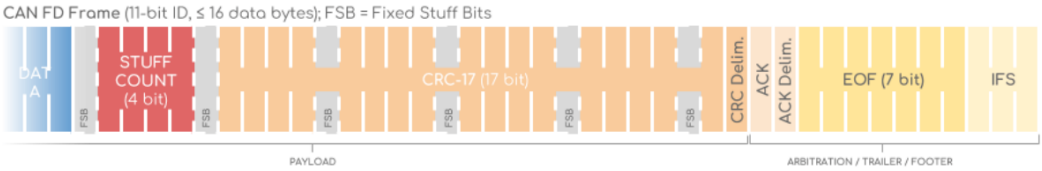

Figure 2.4 End of a CAN-FD frame

As can be seen in picture 2.3 the start of the frame is the same as with the standard CAN protocol frame, so the arbitration process does not change for the CAN-FD. However, after CAN ID field there are some major changes. The frame after ID is composed of following parts:

- RRS replaces 2.1.2 in classic CAN. CAN-FD si not supporting remote requests and Remote Request Substitution (RRS) is always set to 0.
- IDE: IDE is reserved bit as in CAN frames and is set 0.
- FDF: FDF is also reserved bit but compare to classic CAN is set to 1.
- 3 bits newly introduced in CAN-FD:
  - RES: RES reserved bit in CAN-FD protocol and it is set to 0.
  - BRS: The Bit Rate Switch (BRS) indicates the mode in which the payload is sent. If set to 0 the data section of the frame is sent in the same rate as the header of the frame. In case BRS is set to 1, then the payload is sent at higher bitrates.
  - ESI: The Error Status Indicator (ESI) is set to 0 and indicates that the transmitter is in error active mode, otherwise it is in error passive mode.
- DLC: 4 bits of DLC has the same meaning as in classical CAN (2.1.2)
- Data: The payload carried by a CAN frame. Payload can be 1-8 and then 12, 16, 20, 24, 32, 48 or 64 bytes long.
- SBC: The Stuff Bit Count (SBC) is 3 bits in gray's code and 1 parity bit. The following parity bit can be used as the second parity bit.
- CRC: Cyclic Redundancy Check (CRC) ensures the integrity of the frame. For 20-64 bytes payloads the CRC is prolonged to 21 bits.
- ACK: Marks the end of a payload.

[4]

2.2 ISO-TP protocol

ISO-TP protocol is transport layer protocol which represents the 3rd and 4th layer of the ISO/OSI model. In the language of ISO norms it is referred to as DoCAN especially in ISO 14229-3 [5], which connects together, ISO-TP protocol and UDS as all the higher layers in ISO/OSI model. ISO-TP protocol as defined by ISO 15765-2 is closely tied to CAN networks and is designed to server as network and transport layer on them.[6]

ISO-TP protocols enables transferring data greater than payload size of 1 CAN frame (8 bytes for CAN-A and up to 64 bytes for CAN-FD) up to 4 294 967295 bytes [7]. ISO-TP protocol is defined in norm ISO 15765-2 this norm defines data segmentation and all the control flow frames.

ISO-TP as higher layer protocol is not encapsulated into CAN frames as in normalt TCP/IP networks instead properties of ISO-TP are mapped on to properties on CAN. This causes that ISO-TP protocol and CAN protocol to be inseparable. The implementation of encoding information from ISO-TP into CAN frame properties is left to developer and is not neccesary for puprose of this thesis. [8]

ISO-TP protocol uses 4 types of CAN frames in order to provide required services:

- SF: Single Frame. SF is used when ISO-TP PDU can fit into single CAN frame.
- FF: First Frame. FF is frame used, when ISO-TP PDU must segmented into multiple can frames. This frame indicates the start of a message and carries the length of the entire ISO-TP PDU.
- CF: Consecutive Frame. CF are frames following a FF. They carry rest of ISO-TP PDU. CFs contain sequence numbers to more easily serialize and concatenate incoming PDUs. The acceptable values of sequence numbers are from 0 to 15. When first CF following the FF has a sequence number set to 1 and when value 15 is reached the value overflows to 0 and sequence is started over.
- FC: Flow Control. FC frame is set by receiving side of communication and it is send immediately after receiving a FF and then after every n messages defined in FC frame or after FC frame. FC can signal three status messages:
  - CTS: continue to send
  - WAIT: request to temporarily stop sending CFs
  - OVFLW: buffer overflow, size of PDU specified in FF exceeds the size of buffer on receiving size

  FC carries two properties block size, which defines after how many CFs other FC should be send, and separation time minimum, which indicates minimal time separation between individual CFs.

[9]

2.3 Unified Diagnostic Service

Unified diagnostic service (UDS) is application interface a communication protocol defined by ISO 14229 norm and represents the session and application layer in the ISO / OSI model. The communication works as client server. Clients is an external diagnostic application and server is internal ECU. Norm in its first part specifies internal communication data structures, the architecture of internal communication and structures of data exchange and individual data packets. Norm in its second part specifies requirements for session behavior and in the next parts specifies integration on particular data link protocol such as CAN.

UDS is universal application and communication protocol enabling extended diagnostic, routine testing and firmware updates. Its functionalities are divided into so called services which can be further divided into sub-functions. Each services has assigned 1 byte identification code. Services can be divided into privileged and unprivileged ones. Privileged service can be accessed only from privileged sessions other from default sessions which is always available.

The norm specifies list of diagnostic functions which require privileged access. The means of establishing privileged access is left to a manufacturer. The norm offers suggestions to to allow establish session based on client identifier. Restriction to access other services are left to manufacturer. The norm specifies two services for this purpose. The Security Access service offers establishing sessions differentiated by security level based on challenge response with established key. Another service is the Authentication service with an assigned code 29_{16}, which is the main focus of this thesis.[10]

2.3.1 Authentication Service (29_{16})

Authentication service offers means to verify either client's and server's identity based on knowledge of secrete key. The norm also allows of unidirectional authentication, where only client is authenticated toward a server. Norm further specifies all other support sub-functions for this service to work properly such as deauthentication sub-functions or authentication configuration. The specification also includes desired behavior in case of authentication failure and offers possibilities for creation of shared secrete key in case it is derived from certificates of authentication keys.

The authentication is divided into independent authentication schemes. The first is challenge-response authentication using symmetric or asymmetric cryptographic protocols. The second authentication scheme is based on PKI and asymmetric crypto protocols. The norm does not favor any of these schemes and leaves the choice of implemented scheme and cyrpto protocol to the manufacturer.

Data exchanged during the authentication process must containe all the data specified as required in application layer. Any other configuration data is left to manufacturer to specify. It may include a negotiation of cryptgraphic protocol or structure of the Proof of ownership (POW) etc.

The flowchart graphs used to describe the authentication process are missing the parts used to establish shared session key hence it is not necessary for purpose of this thesis. They also missing data required by the application layer, which are specified in the ISO norm, which this thesis does not tried to replace. The graphs also include processes called calculate POW and check aPOW. The POW is signed authentication token build from at least part of send challenge and should be build according to ISO 9798-3 (mutual, three pass authentication) or as a security-related equivalent authentication token. The authentication token should be able to check the integrity of transferred data. [11]

2.3.1.1 Authentication with PKI

Authentication with PKI specifies the whole authentication process plus additional sub-function. The sub-function's purpose is to enbale send eithr client or server certificate without going through whole authentication workflow in case certificate is required to check an authenticity of signed data. The norm specifies the allowed form of a certificate to be CVC and X509 in compliance with norms ISO 7816-8, ISO/IEC 9594-8,RFC 5280 and RFC 5755 or IEEE 1609.2.

The unidirectional authentication flow is described in the picture 2.5 and bidirectional authentication is described in the picture 2.6.

As can be seen in pictures 2.5 and 2.6, both types of authentication send the same number of messages but the volume of exchanged data is different. The three exchange messages ensure the possibility to establish session key even for unidirectional authentication. [11]

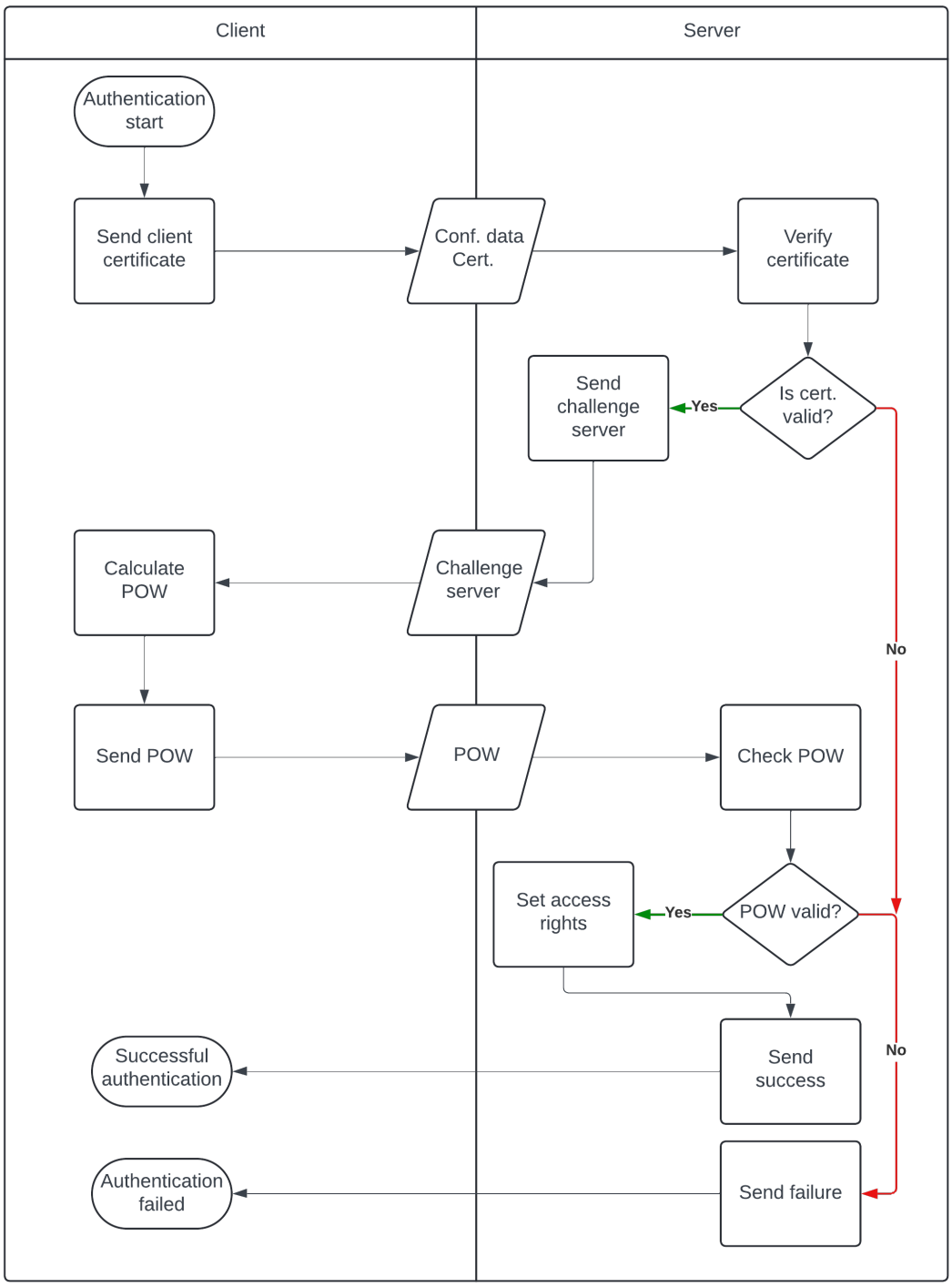

Figure 2.5 PKI unidirectional authentication workflow

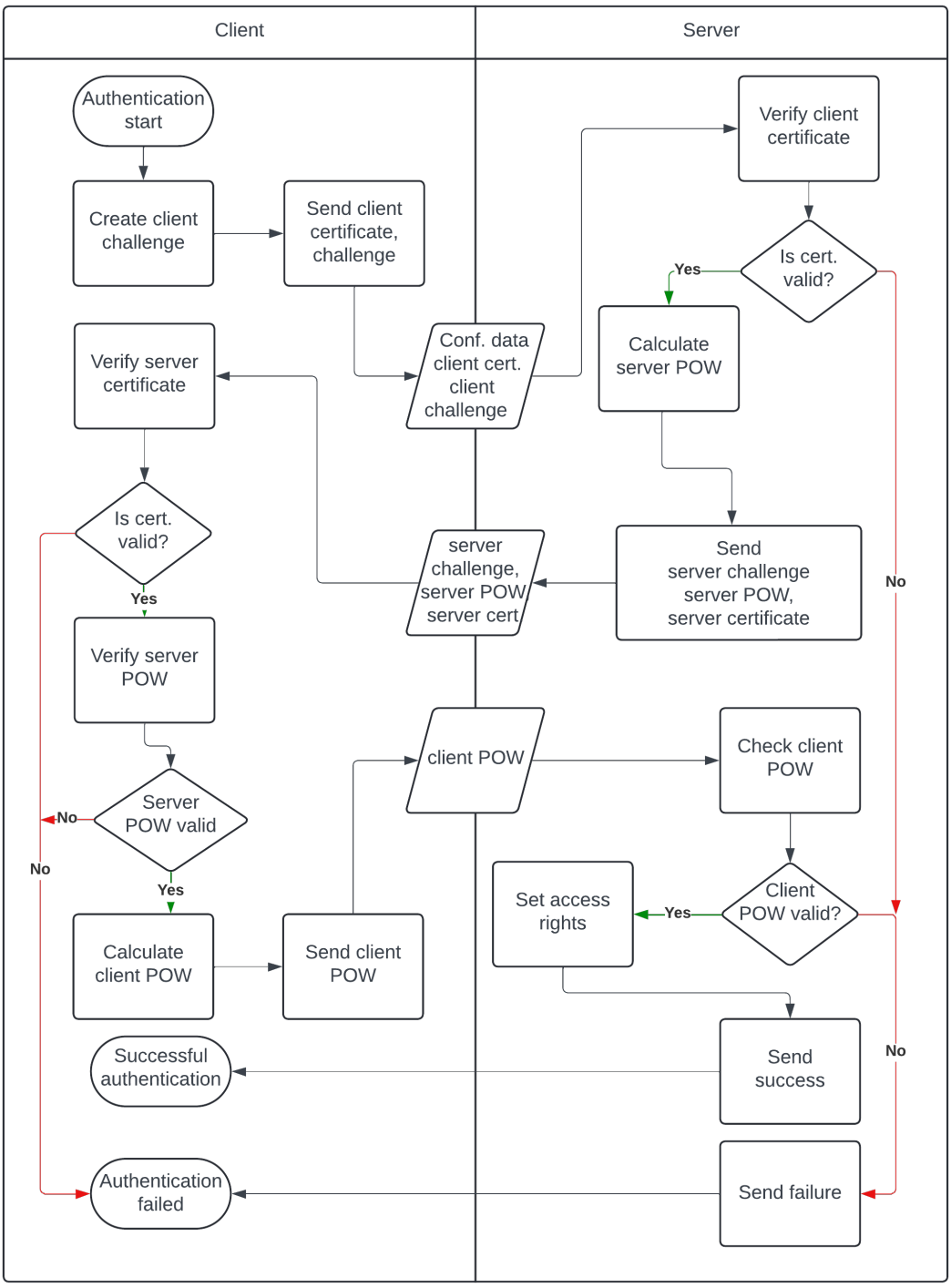

Figure 2.6 PKI bidirectional authentication workflow.

2.3.1.2 Challenge-response authentication

Compare to challenge PKI authentication the challenge-response authentication requires previous knowledge either public key or pre-shared symmetric key (of client or both sides). It also requires

algorithm negotiation because server is missing the information present in a certificate. The process of challenge-response authentication is depicted at diagrams 2.7 and 2.8.

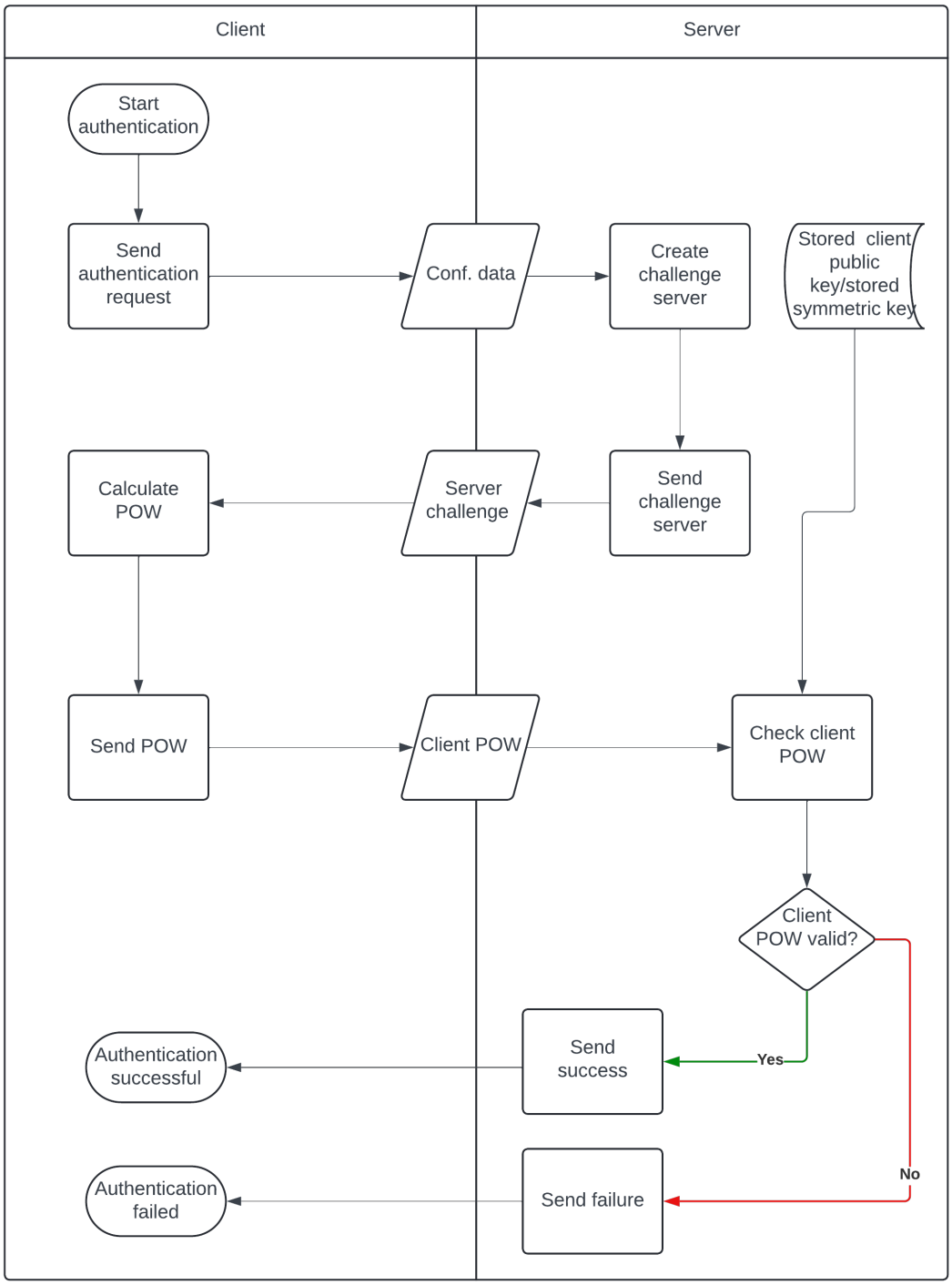

Figure 2.7 Challenge-response unidirectional authentication workflow

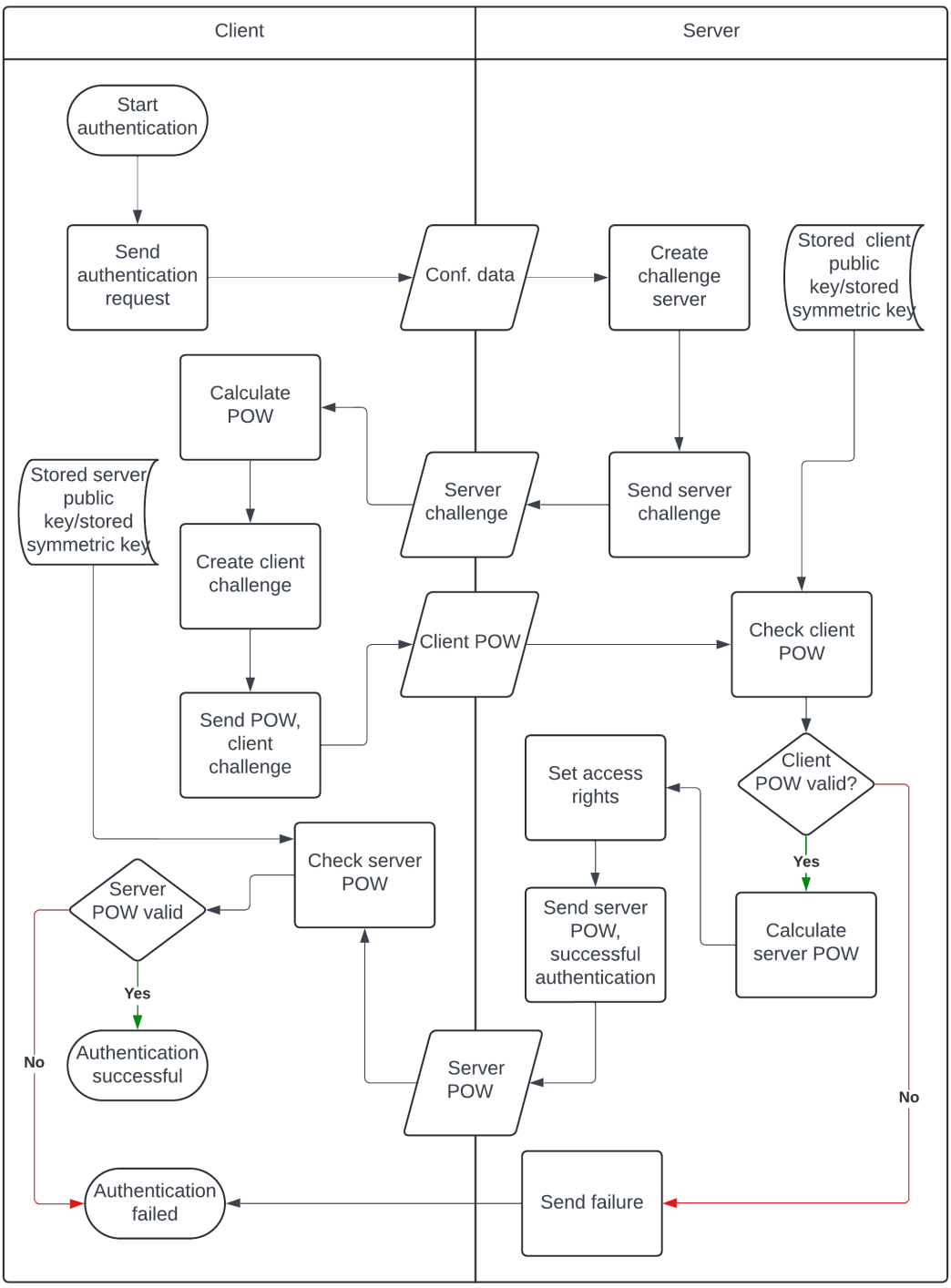

Figure 2.8 Challenge-response bidirectional authentication workflow.

[11]

2.4 Public Key infrastructure (PKI)

2.4.1 Digital signature

Digital signature is a process that ensures the integrity of information. In order to represent the actual state of information, all the information is processed by cryptographic hashing function, which returns stream of byte of constant length called hash, representing the actual data. The three main properties of crypto hashing function is:

1. Digest data of any length
2. Return constant length stream
3. Any change in input data must result in change of out-coming byte stream

As a last step the hash is encrypted by private encryption key.

PKI is framework of procedures and technologies having common goal to provide secure authentication. It relays on asymmetric cryptography and generally trusted third party called Certification Authority (CA).

CA provides necessary service to ensure whole life cycle of signed certificates. First provided service is called Registration Authority, which goal is to accept incoming certificate request validate data in request and validate an applicant's identity. If all these steps are successful the Registration authority passes the request to the CA for signing. When the request is signed applicant receives a certificate with specified validity. The certificate must contain identification of CA which signed it, identification of certificate (serial number), applicant identity, expiration time and public key. It may contain other additional data. All the data is then signed by CA.

There are two kinds of CAs intermediate and top level. Intermediate CAs have their certificate signed by other CA. The top level CA has either self signed certificate and provider of this top level CA must be generally trusted or two top level CAs can sign top level certificates to each other. In case applicant's certificate is signed by intermediate CA the party verifying the validity of the certificate must also check the validity of all the CAs which are the part of signing chain for the certificate (This step is often omitted by admins a can create additional vulnerability in a system). In case any certificate from the chain is not valid, the whole chain becomes invalid.

In addition, CA must provide Certificate Revocation List (CRL). CRL is the tool to announce the compromise of a certificate and invalidate it before its expiration. The CRL must be regularly checked for changes in order to find out newly distrusted end certificate or distrusted intermediate certificate authorities.[12]

2.5 Cryptography

2.5.1 RSA

RSA (Rivest–Shamir–Adleman) is one of the first public-key cryptosystems and is widely used for secure data transmission. It is based on the practical difficulty of factoring the product of two large prime numbers, the factoring problem.

1. Key Generation:
   - Choose two distinct large random prime numbers p and q.
   - Compute $n = p \times q$. n is used as the modulus for both the public and private keys. Its length, usually expressed in bits, is the key length.
   - Compute $\phi(n) = (p-1)(q-1)$, where ϕ is Euler's totient function.
   - Choose an integer e such that $1 < e < \phi(n)$ and $gcd(e, \phi(n)) = 1$; e is the public exponent.

- Compute d to satisfy the congruence relation $d \cdot e \equiv 1 \pmod{\phi(n)}$; d is the private exponent.

2. Encryption:

- For a plaintext message m, where $0 \leq m < n$, the ciphertext c is computed as $c = m^e \mod n$.

3. Decryption:

- Using the private key d, the plaintext m is recovered by computing $m = c^d \mod n$. RSA's security relies on the difficulty of factoring large composite numbers. The public key consists of the modulus n and the public exponent e. The private key consists of the modulus n and the private exponent d. The keys for the RSA algorithm are generated in such a way that while it is easy to compute the ciphertext from the plaintext, it is infeasible to compute the plaintext from the ciphertext without the private key.

2.5.2 ECC

Elliptic Curve Cryptography (ECC) is a form of public-key cryptography based on the algebraic structure of elliptic curves over finite fields. ECC allows for smaller keys compared to non-ECC cryptography (like RSA) while providing equivalent security.

An elliptic curve is defined by an equation of the form:

$y^2 = x^3 + ax + b$ where a and b are coefficients that define the curve, and the equation is satisfied over a finite field

$mathbbF_p$ where p is a prime number. The set of points (x, y) that satisfy this equation, along with a special point called 'point at infinity', form a group under addition.

Key generation in ECC involves selecting a private key, which is a random number, and computing the public key by multiplying this private key with a predefined point on the curve called the generator point G. The public key is then the resulting point on the curve.

ECC is widely used in secure communications protocols such as SSL/TLS and can be found in many modern applications like mobile devices, smart cards, and government IDs due to its efficiency and strong security foundation.

2.5.3 HMAC

Hash-based message authentication code (HMAC) is a specific type of message authentication code (MAC) involving a cryptographic hash function and a secret pre-shared cryptographic key. It is used to verify both the data integrity and the authenticity of a message. The HMAC process involves the following the steps:

1. Normalize the key:

$$K' = \begin{cases} H(K) & \text{if } |K| > B \\ K & \text{if } |K| \leq B \end{cases}$$

where H is the hash function, K is the secret key, and B is the block size of the hash function in bytes.

2. Adjust the key to the right length:

$K'' = K' \oplus (0^{B-|K'|})$

where $\oplus$ denotes the XOR operation, and $0^{B-|K'|}$ is padding of zeros to make K' equal to the block size B.

3. Create inner and outer padded keys:

$K_i = K'' \oplus ipad$

$K_o = K'' \oplus opad$

where $ipad$ and $opad$ are specific padding constants defined in the HMAC specification.

4. Apply the hash function:
$HMAC(K, m) = H(K_o \| H(K_i \| m))$
where m is the message, $\|$ denotes concatenation, and H is the hash function applied twice, first to the concatenation of the inner key and the message, and then to the concatenation of the outer key and the result of the first hash.

Chapter 3
Analysis

The goal of this thesis is to design the system for measuring effectiveness of two authentication schemes offered by the norm ISO 14229-1. The norm specifies exactly to flow of authentication and the interface of public facing functions. In order to that there are some steps which have to be don first.

3.1 Effectiveness

In order to measure effectiveness first it has to be told what symbolizes the effectiveness in context of this thesis. Effectiveness can be measured by many indicators, computational demands, memory demands, time complexity etc. some of the indicators can cover more of the others, for example time complexity also includes the indicators of computational demands. For the purpose of this thesis it was decided, that effectiveness is measured through two indicators:

- Time complexity of the whole authentication flow
- Volume of data exchange during the authentication process

The time complexity of authentication process is great indicator, giving the information about more variables at once it includes computational demands of different cryptgraphic algorithms it also includes dimension of data transfer demands which can be proportionally changed based on available bandwidth. Another advantage of this indicator of effectiveness is that it can be easily a precisely measured in any environment compared to other indicators. Finally, it is parameter which directly influence the user experience and can propagate any change in experiment set up.

The volume of exchanged data is chosen reason because it partly correlates to time complexity parameter and it stays constant throughout the experiment. This two attributes makes good to show the level in which particular parts of authentication process influence the time complexity.

3.2 Design of System for Measuring Effectiveness

The system to measure effectiveness of authentication scheme must put the authentication scheme into number of situation, to see how they perform under different circumstances. First of all the choice of cryptography algorithm is left for choice. It makes sense that various algorithms have different demands computing resources and amount of transferred data. In order to have representative results and to see what kind of impact different algorithms has on both authentication schemes the asymmetric cryptograhic algorithm must be chosen (asymmetric because of the PKI). Further to demonstrate the impact of the crypto algorithms on the authentication process, the

algorithms should differ as much as possible. With this in mind there are two good candidates RSA and ECC cyrpto algorithm. They are the most widely used asymmetric crypto algorithm of todys and they greatly differ in length of encryption keys and computational complexity.

With choice of asymmetric crypto algorithms which can compare the two schemes. There is one advantage of challenge-response scheme, which is now not used. Compare to PKI in can use symmetric cryptography, which in general is computationally less demanding than asymmetric. So in orther to have representative results there must member of symmetric cryptography group of algorithms present in the experiment. For this experiment choice was made to use HMAC algorithm which was designed exactly to ensure the integrity and authenticity of the message. It does not provide the best performence in means of time efficiency. This would probebly the ChaCha20 stream cipher however due incorporated hashing function in the HMAC algorithm it is very effective in transferring POW during bidirectional authentication because authentication token which POW is made of is put together from all the information in contained in client message plus from server challenge and hashing function in HMAC compreses this token to constant 32 bytes instead of just xoring the tokken with expanded encryption key.[13]

The other dimension which influences the chosen effectiveness indicator the time complexity is the data throughput. The data throughput is explicitly used here because it can be influenced by a manufacturer by using different versions of CAN protocol, various payload sizes, and changing the frequency in which CAN frames are sent. But on real HW the bandwidth of CAN interfaces and the massage timing are given by its properties. So in order to simulate particular data throughput instead of changing the frequencyof CAN frames, which is limited by external factors, it is batter to change the size of the payload and use the frequency for fine tuning, to achieve desired throughput. This way the resulting data better simulate testing on the same hardware.

With previous paragraphs in mind the testing system and environment must be designed. The choice was made to make the experiment in the virtual environment and because no real ECU firmware is publicly available it has to be simulated. To create the simulation of the authentication service there is necessary some kind of software layer which would interact with virtual hardware. Because it is not in scope of this thesis to develop simulation of the ECU or develop its firmware choice was made to run the simulation on top standard Unix like operating system. This choice provides some advatages it already includes developed CAN and ISO-TP kernel modules provides desired communication interface. However problem is that standard virtual environment can not emulate hardware of CAN connection. The virtual environment provides three options of interconnecting nodes:

1. Standrad TCP/IP netowrk
2. Serial connection
3. USB connection

There are tools available that establish the CAN connection over all three of these possibilities, however only the option over the TCP/IP network provides the CAN-FD format of communication. This made the choice of tunneling the CAN network through the TCP/IP the only possible choice, because as mentioned earlier this thesis uses the different (bigger than 8 bytes) payload sizes of CAN-FD to test authentication with different data throughput.

3.2.1 Developed Testing Application

With virtual nodes being able to establish connection with the capability of desired protocols. The simulation of authentication itself must be dealt with. There are no tools available simulating the behavior of the ECU with UDS protocol. This brings the neccesity to develop rest of the testing environment. The environment was developed according to the specifications of the ISO 14229-1 norm. It follows the same authentication flow as depicted in diagrams 2.8, 2.7 and 2.6

and 2.5. Compare to specifications in the ISO norm the simulation lacks some of the features and requirements, which are described there. Individual functionalities of services are its sub-functions. The simulation only implements the sub-functions necessary for the authentication process as depicted in the diagrams and additional serving functions. The missing sub-functions are:

- Authentication configuration
- Transmig certificate
- Deauthenticate

These sub-functions do not play a role in authentication flow and they are not necessary for the purpose of the experiment.

The developed simulation also lacks some other required features which are not used in authentication process such as session handling or error handling, as specified in the ISO norm. The simulation does handle error states and notifies the user when error occurs, but it is not according the norm.

The simulation is developed in C++ 2017 standard using the current version of OpneSSL library and functionalities provided by Linux kernel. OpenSSL library is chosen because it long time developed library containing many crypto algorithms and as one of the most used crypto libraries it has well-tuned performance on all main algorithms, which one of the best currently available. This minimizes the chance to have outgoing data influenced by inefficient implementation of crypto algorithm.[13]

The C++ programming language is used because it allows portability to various system and together with C language is widely used for programs running on less sophisticated HW.

The structure of class that forms the whole simulation with its main features is described in following sections 3.3.1 and 3.3.2. In this section, the main focus will be more on methods forming the authentication itself than implemented features and ways to interact with the simulator.

The simulation implements the transferred data into structures according to ISO 14229-1 norm. To ensure accurate realistic measurement of the transferred data volume. The implemented sub-fucntions defined in the ISO norm have a different naming convention than in the source code of the simulation but can be easily identified, more about implemented methods can be found in the section 3.3.1. The sub-sunctions and their responses defined the ISO norm are distributed across 2 implemented classes the *client* and the *server* based on who sends and receives the response in authentication workflow 2.3.1.

3.2.2 Preparing the Testing application for the Experiment

The program, simulating the authentication service testing, provided one disappointing fact. It revealed a bug in ISO-TP kernel module which in random times disrupts the ISO-TP session, when using standrad CAN and causes authentication to fail. Extensive debugging proved that the bug is not in the developed application, however, the author was not able to find the bug in the kernel module during the testing phase. Fortunately the bug manifests itself not too often so does not statistically influence the experiment.

For the proper testing of authentication mechanisms, they have to be tested using different speeds and payload sizes. The choice was made to test the authentication with data throughput 50, 100, 175, 250 and 325 KB/s.

The original plan was to use virtual serial line between the client and the server. However, slcand [1] tool which is used to handle CAN bus protocol over serial line does not support CAN-FD version of the protocol. This raised a problem, when variable size of CAN FD paylaod must

[1] `https://github.com/linux-can/can-utils/tree/master`

be used to achieve desired throughput of the channel. This is because of Linux kernel module, which handles CAN bus communication is not able send CAN frames in shorter intervals than 100 μs, which is not enough for needs of this experiment. This restriction has its reason because maximal speed which CAN2.0 bus protocol can achieve in reality is 1 Mb/s.

In order to use the CAN-FD the tunneling through TCP/IP must be used. To achieve desired throughput the frequency of sending the frames was calculated for individual packet sizes and is in table 3.1.

Data rate (KB/s)	CAN frame size (bits)	Pyaload size (bits)	Time rate (ns)
50	108	64	270 000
100	190	128	237 500
175	318	256	227 143
250	446	384	223 000
325	574	512	191 333

Table 3.1 Chosen data rates for ISO-TP protocol and corresponding time rates.

To achieve the exact timing in sending the frames the standard communication of ISO-TP protocol must be altered. The setting of the socket created for ISO-TP communication was altered to ignore minimal separation time incoming flow control frame which can be set with precision to hundreds of μs. How it is done is described in section 3.3.2.

3.2.3 Data Measurement

This experiment needs to extract to kinds of data. First volume of transferred data this volume is not effected by externalities such as CPU context switching, or memeory use. This means that one run of authentication for each type of authentication and for all cyrpto algorithms and for all payload sizes suffices. To measure transferred data thre are no special functionalities in developed application instead the tool Candump from CanUtils packege can be used. When run and attached to set up virtual can interface it created dump of entire communication. From this dump it is then easy to extract the volume of data transferred during the communication.

To measure time duration of an authentication the developed application uses the chrono utility from standard C++ library. The stop watch starts just before the call of client's *auth* method is called and stopped immediately after return from the method. The measured time is then, in case of successful authentication, written into a file. The file with results are named according to following scheme: *cypherName_authSchemeName* in case of unidirectional authenticaton the file name ends with *_uni*. The file are sorted into directories based on payload size.

In order to change the payload size/data throughput the source code in *connect* class must be edited and whole application must be recompiled. For this purpose the makefile is attached to the source code.

The experiment was carried on the same virtual infrastructure, it was developed. In order to minimize influence of guest operating system load and the load of host operating system as well, the client is newly started and waited to properly finished in each run. The server on the other virtual machine runs all the time and waits for all incoming connections.

In order to get reliable data of the time complexity of the individual authentication schemes wit various set ups, the client was run 2000 times for bidirectional and also unidirectional authentication in each configuration. All the runs were done with the same arguments. In order to get closer to real environment all the encryption keys were generated using OpenSSL command line tool with the standard recommended length of the keys. For the RSA the key length was chosen to 2048 bits with default padding scheme, ECC used key of length 256 bits and for HMAC crypto algorithm was used 256 bits of key. All the runs were made with the same cryptographic

keys. All the authentication runs where made with 20B randomly generated challenge, however the challenges were not checked for duplicity during the run.

The duration of the authentication is measured through the all acceptable combination of authentication scheme, payload sizes and crypto protocols. However the main focus is directed towards the data from the bidirectional authentication. This is because of the authentication workflow. Both workflows for unidirectional authentications are just parts of the bidirectional. This indicates, that all the corelations should be the same, but for bidirectional authentication batter visible because of the longer duration and more transferred data.

3.3 Implementation of Testing Environment

The goal of this section is to show and explain the structure of the program which was used to carry out the experiment and simulate the authentication process between client endpoint and server ECU. It also names other software tools which were used on the experiment.

The whole simulation of authentication process to ECU is implemented in one piece of software which can simulate either a client side or a server side of the authentication process, based on chosen parameters. The server must be run with configuration for all supported authentication mechanisms and cryptography protocols to be able to answer all the supported requests.

When the software is run in client mode it is necessary to supply only encryption keys required for the chosen authentication mechanism and cryptography protocol.

3.3.1 Class structure

Whole software is build from set of classes which most of them is common for both modes the server and the client. Development of the software went through several stages, and during them was almost completely redesigned. First designs worked implemented the client and the server as two independent pieces software. This design proved inefficient since it duplicated approximately 60% of classes. Later stages worked with one piece of software which incorporated both the client class and the server class which each provides role specific methods, and both inherit grate part of their functionalities from common parent. The final stage of impelemented classes and theire realtions can be seen in diagram 3.1

The fundamental class is class named *connection*. Its main purpose is to setup CAN socket and provide methods for sending and receiving messages through a socket. The *connection* class mainly uses uses ISO-TP kernel module [2] and SocketCAN kernel module, which both are now included in standard kernel since version 5.10. Work with SocketCAN kernel module was same as with standard C++ network sockets this module is also well documented in kernel documentation [14]. The work with sockets is not very comfortable in C++, but this kernel module can work only with individual CAN packets.

In order to simplify work with incoming and outgoing traffic ISO-TP kernel module was used. This module implements ISO-TP protocol, which manages CAN packets into complete massages and creates some sort of sessions. Work with this module is bit more trickier than with the SocketCAN kernel module, because of missing documentation. Only clues how to use the module lies in the source code comments and slide presentation from author of the module [3].

The class which holds most of the functions a makes base for both the client and the server is class called *service_29*. This class holds message structures as defined in ISO 14229-1:2020. This class holds also methods for cryptography operations done during the communication and some other support functions like loading encryption keys, creating random challenges etc.

[2] https://github.com/hartkopp/can-isotp

[3] https://s3.eu-central-1.amazonaws.com/cancia-de/documents/proceedings/slides/hartkopp_slides_15icc.pdf

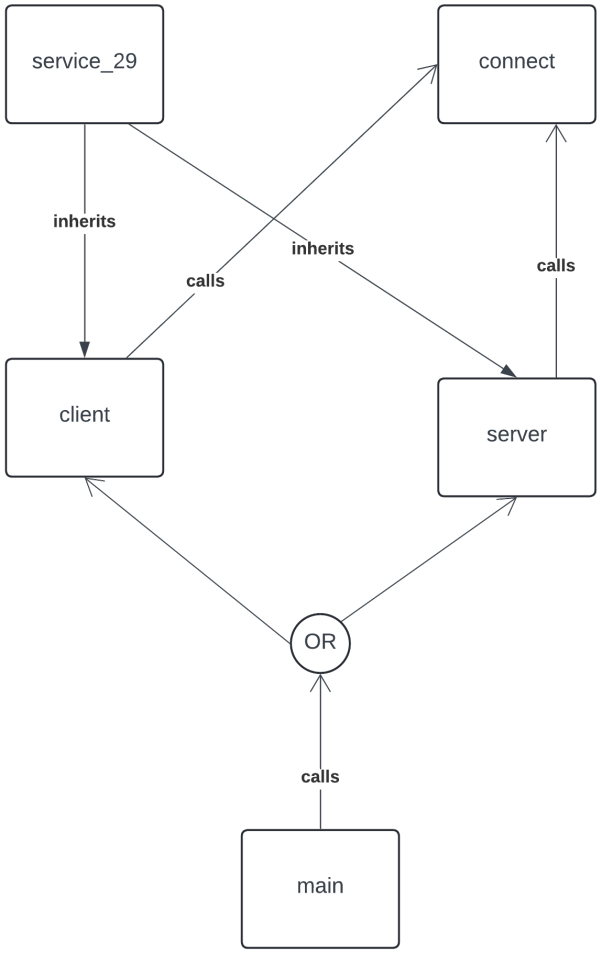

Figure 3.1 Diagram depicts implemented classes and their relations

Classes representing the client and the server both inherit structures and methods from previously mentioned *service_29* class. The *client* class overloads *auth*, which starts the whole authentication process, and based passed parameters, it is determined which kind of authentication is used.

The *server* class represents the server and its main purpose is to listen for incoming authentication communication. All the necessary data are passed to *server* class constructor method, and the *auth* method only starts listening for incoming connection. The class is constructed in a way, that it can handle all kinds of authentication mechanism as defined in ISO 14229-1:2020 and all currently implemented encryption algorithms based on information in incoming packets.

The classes *client* and *server* hold implementation of methods, implementing the communication scheme as defined in the ISO 14229-1 norm mentioned above. The *client* and *server* classes hold the methods implementing sub-functions and the class *service_29* holds the definitions of data structures for request and reponses. The naming convention in the source code slightly differ from the one used in the norm. The sub-functions from the norms have the prefix *send_* or *recv_* in client class in the source code. In the *server* class sub-functions have the names without the prefixes, but it keeps the abbreviated sub-fucntions names like the *client*.The names of individual sub-functions defined in the norm are changed not use camel notaion and the individual

words are abbreviated. The sub-functions which have the same purpose and handles same data are marged together into one method for both uni-directional and bi-directional authentication. The table **??** shows the mapping of the methods implemented to request in ISO 14229-1.

3.3.2 Implemented features

This implementation of UDS service 29 communication holds few options and set ups which are worth mentioning.

Implementation of cryptographic function uses openSSL crypto library. There are two types of cryptographic operations implemented first one calculating POW and second checking incoming POW. Method calculating POW must have interface in following:

```
1    static const std::vector<uint8_t> calculate_POW (const std::vector<uint8_t>
     & key,
2                                                     const std::vector<uint8_t>
     & pow_base);
3
```

Code listing 3.1 Interface for calculating proof of ownership

Method for checking received POW must have interface in this form:

```
1    static const std::vector<uint8_t> check_POW (const std::vector<uint8_t> &
     key,
2                                                 const std::vector<uint8_t> &
     rcv_pow,
3                                                 const std::vector<uint8_t> &
     pow_base);
4
```

Code listing 3.2 Interface for checking proof of ownership

The static keyword is only necessary if implemented functions are also methods of *service_29* class. They can be also implemented as stand alone functions outside of the class.

Use of these interfaces provides possibility adding various encryption algorithms and their easy incorporation into this project. When adding new crypthographic algorithm into project, it must be add into switch statement, which is located in the method *service_29::_match_alg*, where addresses of new methods are assigned to the class variables. New methods must be also added into main function which handles input and parameters from command line.

The current implementation allows the use of HMAC cryptographic protocol and all asymmetric ciphers supported by OpenSSL library in both scenarios challenge response or with use of PKI. The broad spectrum of asymmetric ciphers available for use is achieved by using OpenSSL d2i functions which can determine correct cipher based on provided DER encoded public key or from certificate which directly holds desired information.

Another noteworthy thing is the setup of CAN socket when using ISO-TP kernel module. The only available documentation is in the source code of the module. In this project the set up is done in the constructor of *connection* class. The set up is done through function *setsocketopt* and structures defined in the kernel module. The function has the following declaration:

```
1    int setsockopt(int socket,
2                   int level,
3                   int option_name,
4                   const void *option_value,
5                   socklen_t option_len);
6
```

Code listing 3.3 Declaration of setsocketopt function

Except the *socket* parameter, all the other parameters are defined in the kernel module. Parameters *level* and *option_name* are defined as macros at the start of header of file of the module. According to option name proper structure must be passed as the *option_value* and also its length as *option_len* parameter, which are defined in the same file. There are three structures defined which each is responsible for setting up different part of ISOTP protocol. There is *can_isotp_opotions* structer, which handles all local setting of ISO-TP communication. There are 2 parts of this structer wort mentioning the first one is *can_isotp_options.flags*. This is unsigned 32 bits data type which holds values of all setup flags defined in the header file. These flags are key to set up correct behavior of the communication. The second one is *can_isotp_options.frame_txtime*, which holds time interval between individual frames. However this time interval is used only when *CAN_ISOTP_FORCE_TXSTMIN* flag is set up, or there is no Flow Control Frame received. In other cases the value in *can_isotp_options.frame_txtime* is overwritten by value received in Flow Control Frame. The structure *can_isotp_fc_options* sets up the values received in already mentioned Flow Control Frame specially the time interval between frames, block size and maximum number of wait frames. The last structure named *can_isotp_ll_options* enables use of CAN-FD packets and also sets up a size of a payload.

The last thing to point out from the implementation of the UDS service 29 is the way encryption keys and certificates are handled in the client and the server. Both instances must handle those differently because server must be able to handle any incoming implemented cipher suit and so needs list of all crpytographic keys available. But the client only needs a chosen encryption and correct decryption key. This need resulted in different requirements for passed arguments to some parameters.

The difference is with parameters *-k* and *-p*. With parameter *-k* the the client expects path to client's private key, similarly with parameter *-p* the client expects path to server's public key. However the server with these parameters expects list of all keys available. The server's *-p* parameter expects path to file which holds structured information about private keys available to the server, this includes keys to symmetric ciphers. The file delimiter is *:* character and the structure of the file is following:

$$server\_ID : cipher\_ID : path\_to\_private\_key$$

Server_ID is defined with parameter *-i* and must match with at least one server_ID in the configuration file. Cipher_ID is defined in program and shown in help of the program. The *-k* parameter requires also path to the file but this file holds structured information about public keys of the clients. This file is used only for authentication with asymmetric encryption but for simplification and possible implementation of crypthographic algorithms not supported by OpenSSL library it hods same structure as file for storing paths to private keys, expcet instead of holding server_IDs it holds client_IDs. If in an incoming connection the asymmetric cryptography is identified a path to particular public key is matched base on client_ID in the incoming message.

3.4 Infrastructure

The whole experiment was developed and run in virtual environment using Virtual box hypervisor and Cenneloni tool [4], which provides the CAN protocol connection between virtual client and virtual server. Although CAN bus is a layer 2 protocol on ISO/OSI model Canneloni channels CAN bus protocol over TCP/IP. Both virtual machines were equipped with two virtual network cards, one connected to the private LAN to establish connection between client and server and to ensure minimal interference in the communication. Second virtual card was connected to the internet through NAT to be able to communicate with git repository which held the latest version of the code. In order to avoid any unforeseen problems with portability between development environment a testing virtual machines the code was always compiled on target machines.

[4] `https://github.com/mguentner/cannelloni`

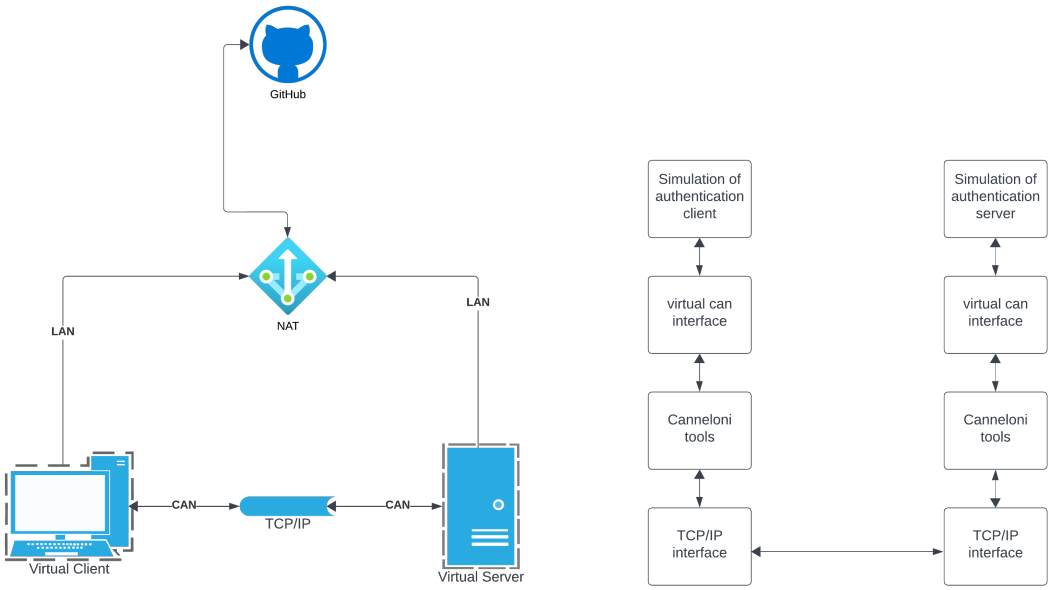

Figure 3.2 Virtual environment

Virtual machines used as client as server both run publicly available Linux distribution Ubuntu server version 22.04.02 LTS with can-utils [5] package installed. For the compilation of source code it is also necessary to download Linux header files and compiler for C++ source code with C++ standard library and OpenSSL library.

To successfully establish the CAN connection with Canneloni, one side must always act as a server, that side must be started first to the client side to connect to it. The connection can be done through the TCP or the UDP session. In order to have reliable connection and based on chosen speeds which are not that high, the choice was made to use the TCP session.

[5] `https://packages.ubuntu.com/jammy/net/can-utils`

Chapter 4

Data Evaluation

Measured data gives expected trends where authentication schemes using the PKI with RSA encryption (which is the most computational and data demanding) to be slowest and the challenge response with HMAC encryption to be the fastest.

The bidirectional authentication with different parameters have the following average times.

payload size (B)	cypher	auth. scheme	avg. time (μs)
8	ECC	PKI	543 513
8	ECC	CR	402 663
8	HMAC	CR	369 391
8	RSA	PKI	891 133
8	RSA	CR	436 456
16	ECC	PKI	294 294
16	ECC	CR	192 480
16	HMAC	CR	184 292
16	RSA	PKI	532 617
16	RSA	CR	211 563
32	ECC	PKI	233 757
32	ECC	CR	135 160
32	HMAC	CR	85 602
32	RSA	PKI	279 322
32	RSA	CR	143 220
48	ECC	PKI	162 410
48	ECC	CR	113 151
48	HMAC	CR	19 901
48	RSA	PKI	243 918
48	RSA	CR	126 015
64	ECC	PKI	119 098
64	ECC	CR	19 137
64	HMAC	CR	13 497
64	RSA	PKI	207 028
64	RSA	CR	124 098

Table 4.1 Average bidirectional authentication time.

The unidirectional authentication data follows the same patterns as the bidirectional authentication.

payload size (B)	cypher	auth. scheme	avg. time (μs)
8	ECC	PKI	348 712
8	ECC	CR	392 608
8	HMAC	CR	379 332
8	RSA	PKI	514 062
8	RSA	CR	397 086
16	ECC	PKI	158 920
16	ECC	CR	152 399
16	HMAC	CR	145 195
16	RSA	PKI	329 145
16	RSA	CR	160 110
32	ECC	PKI	141 871
32	ECC	CR	87 705
32	HMAC	CR	24 550
32	RSA	PKI	146 589
32	RSA	CR	96 413
48	ECC	PKI	87 305
48	ECC	CR	35 556
48	HMAC	CR	19 066
48	RSA	PKI	143 379
48	RSA	CR	79 802
64	ECC	PKI	84 840
64	ECC	CR	20 814
64	HMAC	CR	8 451
64	RSA	PKI	138 481
64	RSA	CR	78 825

Table 4.2 Average unidirectional authentication times.

As can be seen in above tables there is an unexpected phenomenon. There is noticeable speed up in challenge response authentication schemes where CAN payload is greater than 32 bytes. This speed up should not be there because generated challenges are of size 20 bytes and overhead of ISO-TP protocol is less then 10 bytes, though the whole message should be sent in singel frame. This is caused by setup of different time rates, for sanding packets with different can payload sizes. The time rates were calculated in order to have the CAN protocol particular throughput. Selected data rates are shown in the table 3.1.

As can bee seen in order to achieve desired throughput the time gap between sanding individual packets is shortening with increasing payload size.

The following table demonstrates data demands with for different combinations of payload size, authentication scheme and encryption algorithm. Because the trends of increasing effectiveness with increasing payload size are the same in unidirectional and bidirectional authentication. The difference is that with bidirectional authentication the trends are clearer, because of the greater overall data volume that must be transferred and so the transferred data volumes are measured only for bidirectional authentication.

In the table 4.3 there is expected decrease of transferred data volumes specially at the PKI authentication schemes. However at smaller data volumes, later at the table there are fluctuating data volumes. This is caused by the situation where the impact of the increase in effectiveness is suppressed by limitations of allowed CAN-FD payload sizes (section CAN-FD) causing padding

Payload size (B)	Authentication scheme	Encryption alg.	Transferred data volume (B)
8	CR	ECC	330
8	PKI	ECC	1576
8	CR	HMAC	238
8	CR	RSA	750
8	PKI	RSA	2897
16	CR	ECC	308
16	PKI	ECC	1466
16	CR	HMAC	225
16	CR	RSA	704
16	PKI	RSA	2692
32	CR	ECC	299
32	PKI	ECC	1427
32	CR	HMAC	225
32	CR	RSA	683
32	PKI	RSA	2603
48	CR	ECC	297
48	PKI	ECC	1411
48	CR	HMAC	218
48	CR	RSA	674
48	PKI	RSA	2581
64	CR	ECC	318
64	PKI	ECC	1411
64	CR	HMAC	219
64	CR	RSA	686
64	PKI	RSA	2571

Table 4.3 Measured data flow volumes for various bidirectional authentication schemes

to be added to the messages.

Further, there are no unexpected events in the above results. The PKI authentication scheme is slower in general than challenge response due to its greater data demands. Use of elliptic curve cryptography proves more effective than use of RSA. This is mainly caused by size of keys and computing complexity.

4.1 Measurement Interpretation

Because patterns a behavior of the data is the same for the unidirectional and bidirectional authentication following sections are working only with data for bidirectional authentication. However everything said can be applied on unidirectional authentication as well.

The measured data does not give any unexpected results. All unexpected phenomenons in the data were explained in previous chapter. The goal of this chapter is to give interpretation the raw data presented.

4.1.1 Challenge Response Authentication

The challenge response authentication scheme is using 3 encryption algorithms HMAC, RSA and ECC as shown at the graph 4.1

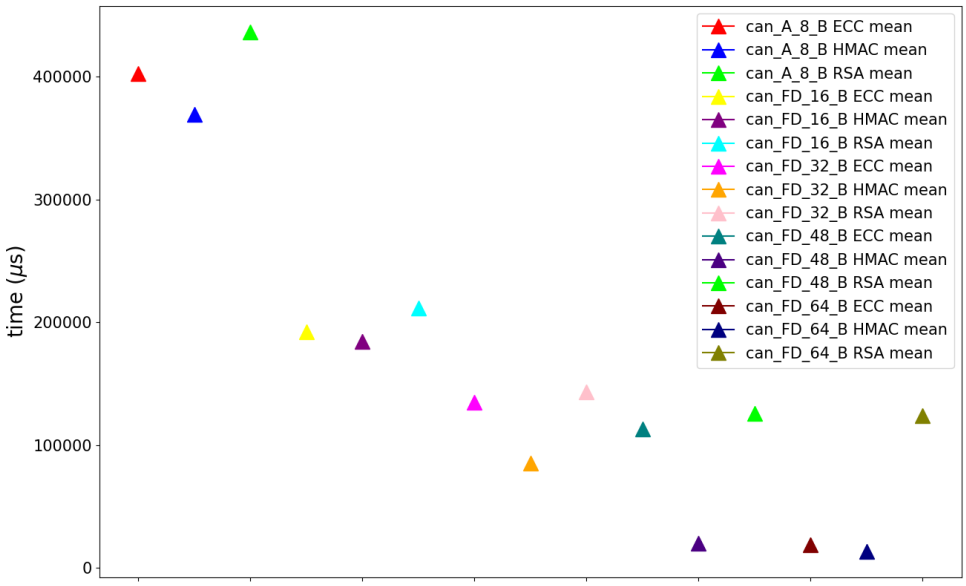

Figure 4.1 Challenge response average authentication duration with different payload sizes.

The graph indicates that the time difference between particular encryption algorithms is not significant with use of CAN-A protocol. This can be confirmed by looking at the table 4.1, where is exact value of mean of measured times. But considered the difference in transferred data, where when using the RSA algorithm 750 bytes is transferred, with ECC and HMAC 330 and 238 bytes is transferred. The conclusion is that with this payload size the main component of the overall duration of the authentication is made by data exchange. With the greater payloads there is noticeable speed up in the authentication duration. With the 16 and 32 bytes payloads the average times of different encryption algorithms are still close together and effectiveness increases evenly for all. But with 48 and 64 bytes payloads there is noticeable higher speed up for HMAC and ECC than for RSA. This indicates change when data transfer is effective enough that computational demands of the particular algorithms manifest itself in over authentication duration. In order to demonstrate this, difference of authentication duration between RSA and HMAC when CAN-A is used is 67065 μs and when CAN-FD with 64 bytes payload size the difference is 110601 μs.

4.1.2 PKI Authentication

The PKI by its nature allows to use only the asymmetric cryptography and must transfer grater data volumes then when challenge response authentication is use. Amounts of transferred data can be seen in table 4.3. This table shows that with the 8 byte payload size the difference between amount of transferred data is 45.6%. When transferred data volumes with 64 bytes payloads are compared the difference 45.1%. This indicates that protocol overhead (both CAN-FD and ISO-TP) is negligible and so the main difference in transferred data volumes is caused by transferring the X.509 certificates.

As shown in the graph 4.2 authentication with ECC is more effective through all the payload

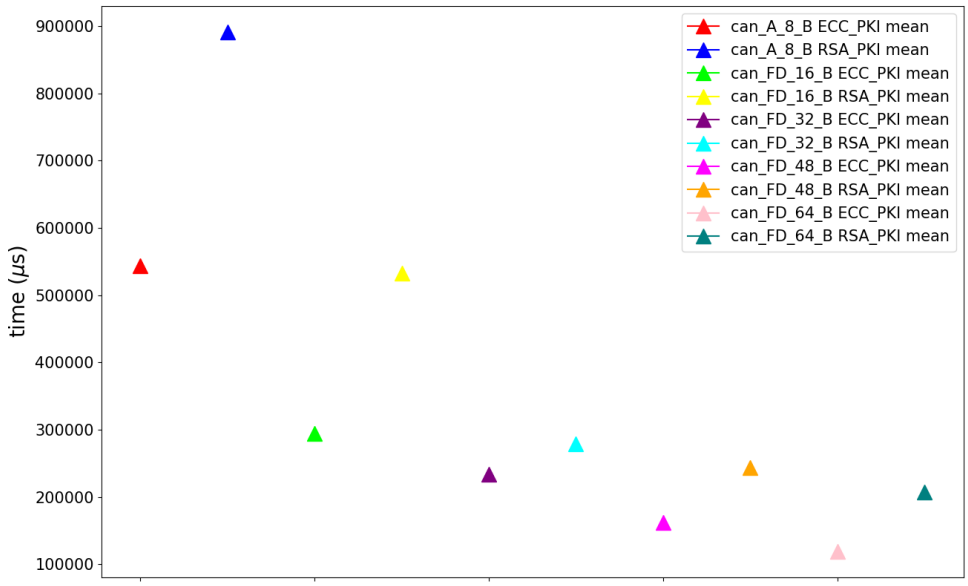

Figure 4.2 PKI average authentication duration with different payload sizes.

sizes. This is expected behavior because of smaller transferred data volume (ECC has significantly smaller encryption keys than RSA) and also lower computational complexity. The graph shows great increase effectiveness in data transfer, but still showing authentication with ECC algorithm to be faster than with RSA. The pattern makes by authentication times with different payload sizes indicates that increase in effectiveness caused by greater payload size is slowing down and remaining time difference is mainly the difference in time complexity of particular encryption algorithms. This difference can not be mitigated by any means.

4.1.3 Comparison of Challenge Response and PKI

To compare CR and PKI authentication only the ECC crypto algorithm and HMAC are chosen. RSA is omitted from this comparison because it performs the worst in both the PKI and CR authentication as shown in sections 4.1.2 and 4.1.1. The HMAC crypto algorithm is chosen, as it is the only tested symmetric cyrpto algorithm.

The graph shows the expected. Challenge response authentication if faster with both HMAC and ECC crypto algorithms than PKI with ECC crypto algorithm used for both the certificate and for CA. However whats worth noticing on this graph (4.3) is the time duration of all the displayed options a time difference between them. All the authentications either to PKI a CR took at maximum little bit over 0.5 second and the maximum difference between the slowest and fastest authentication mechanism for the same payload is only 0.17 of a second and in average it is 0.14 second. This shows the PKI authentication with ECC crypto algorithm to be able to compete with the challenge response authentication even when fast symmetric cryptography is used. The PKI authentication is getting even better when high data throughput is available.

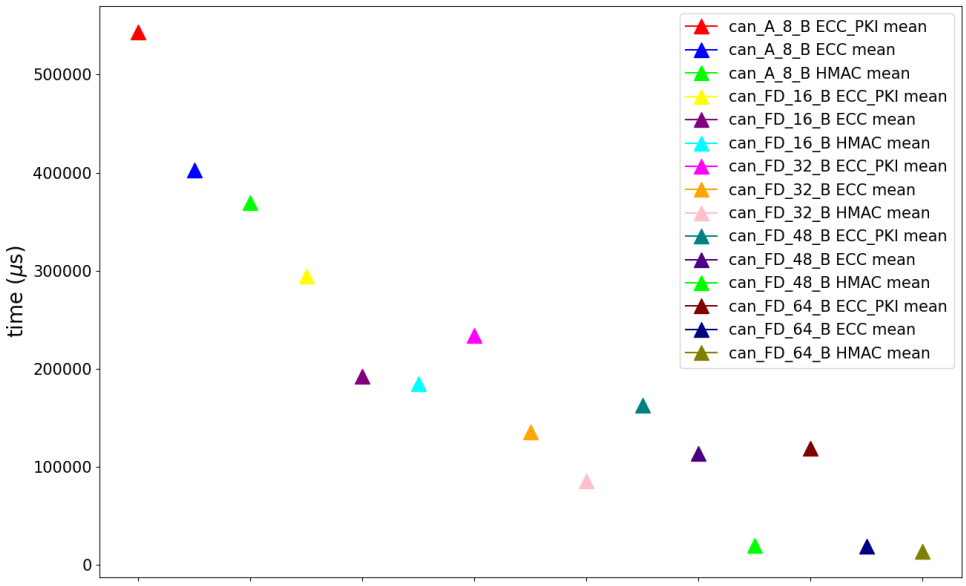

Figure 4.3 Comparison of PKI with ECC crypto algorithm authenticaation duration to callenge response authentication using the HMAC and ECC crypto algorithm.

4.2 Immeasurable variables

The preceding sections are focused solely on measured data not taking into account some other important properties of the authentication schemes, which are however relevant to the use case where the authentication occurs.

First of all challenge response authentication requires to keep and maintain some king of user database, which stores encryption keys of particular users or groups of users. Maintaining this kind of database onboard a vehicle such as a car can be challenging and requires direct or remote access of a user or some kind of automated tool something like IDM system. In order to overcome this obstacle set predefined encryption keys set by manufacturer can be loaded into memory and distributed to groups of clients which shall have access with given rights. This approach has the disadvantage that in case any encryption key is leaked the authentication with given rights is compromised and this part of system becomes publicly accessible, unless undergoing a great deal of struggle adding new encryption, deleting the old one and distributing the new key to all the authorized users. The new dimension of this problem rises, when it is considered that there are hundreds of thousands or even millions of vehicles. In ideal case, vehicle should have its own database and set of encryption keys. But this approach is impossible mainly for authorized users who would have to hold millions of encryption keys for each vehicle manufacturer, whose vehicles he should have access to, and so there is usually one or just a few encryption keys for each manufacturer, which works for each manufacturers vehicle. Given this situation, the replacement of the leaked encryption key becomes an even bigger problem.

On the other hand, the PKI authentication scheme does not require the holding of encryption keys that would be tied to the client's identity, because the required key is provided by the client during the authentication process (more details in 2.3.1). Disabling client access is an easy and

already automated task done by CA and all it is necessary is to import a certification revocation list into vehicle ECU. An other advantage can be the easy and even automated distribution of certificates for clients. This implies the possibility of starting using individual authentication certificates and the only restriction that must be considered is that an ECU must be able to digest the size of a certification revocation list. The last thing to take into account is that a certificate can hold many other information in addition to the identity of the clients, such as the rights of clients for the authorization of ECU or anything that comes to mind.

4.3 Discussion

As shown in previous sections, measured data confirm the authentication of the challenge response faster than the PKI authentication scheme. Further more it confirms that less computational demanding and bandwith demanding crypto protocols are slower than the more effective ones. The key thing this data is showing that the overall duration of authentication scheme is useble even for the least effective option which is PKI authentication scheme with RSA crypto algorithm. In configuration with CAN-A protocol (8 byte payload) the bidirectional authentication took little less than 1 second (4.1). Considering the use case where new sessions are started maximally once every 10 minutes in case the maintenance is done on vehicle the less than 1 second wait time for authentication seems to acceptable. This can become a problem, when authentication service would be used for automated data remote collection which could collect data for example every 30 seconds. But the data shows that with the proper choice of cryptographic algorithm the desired authentication effectivnes, can be achived even when using the PKI scheme not just the challnge response.

When immeasurable variables (Section 4.2) are taken into account the PKI scheme seems to have more advantages then the challenge response. The data also shows that PKI authentication scheme with lower bandwidth requirements and, for fast CAN buses, with lower computational demands such as measured ECC crypto algorithm in avrage 0.14 second slower than the fastest measured challenge response authentication algorithm. The ECC crypto algorithm with PKI authentication scheme with 8-byte payload took only 0.54 seconds and with 64-byte payload only 0.12 second.

The data further shows that with low bandwidth such as 8 and 16 byte payload the computing demands of various cyrpto algorithms have only small impact on over all authentication duration. This puts main pressure on choice of padding for the crypto algorithm and in case of PKI the length of public keys and hashing functions used. This gives great space to balance the security requirements of authentication with effectiveness, especially with the PKI schemes.

Measurement was made only with one symmetric crypto algorithm, which is not even proper encryption but only message authentication code. The HMAC algorithm was chosen for two main reasons. Algorithm designed exactly matches the use case here, ensuring integrity of message and authenticating the sender. The second one is that it is one of the recommended algorithms in ISO norm 14229-1. There are other suitable candidates such as AES representing the symmetric block ciphers or ChaCha20 as stream cipher. However other symmetric cypher would not bring any new information, because the main focus of this thesis is to compare CR and PKI authentication scheme.

Chapter 5
Conclusion

The goal of this thesis was to compare the effectiveness of two authentication schemes offered by ISO 14229-1. It compared these two schemes in combination with 3 most widely used cryptograhic algorithms in terms of time complexity and volume of transferred data.

As expected amount of transferred data is significantly higher with authentication using the PKI, how ever compared to challenge-response authentication there is much more space for optimisation in this part.

At first sight the data about time effectiveness are not unexpected in means that authentication with PKI is slower (it must carry more data) than authentication with challenge-response. What this experiment brings new is the exact comparison and when the data are considered from the perspective of expected use case it is not as simple as at first sight. Data showed that even in the slowest possible configuration (RSA crypto algorithm and CAN-A netowork protocol) the PKI is still usable for authentication. When added value of PKI authentication is considered, the PKI authentication scheme emerges as comparable and usable solution to challenge-response authentication scheme in terms of speed and much more effective from point of access and identity management.

Data also showed clearly the most effective solution to be the PKI authentication scheme with ECC cyrpto algorithm, which could hold up with the speed of fastest authentication scheme (challenge response with HMAC crypto algorithm) and also still carry advantages of PKI.

Data also showed that computational effectiveness of particular crypto algorithms takes noticeable effect at networks with bandwith over 250 KB/s otherwise the most significant impact has amount of transferred data. This information is very important for a future when considereing the use various cyrpto algorithms, with PKI, to pay close attention to used padding, length of public keys and consider amount of information contained on a certificate.

The next steps could lead to several ways. One is to broaden the choice of encryption algorithms by post qauntuam encryption algorithms defined by NIST. Other one is to enrich the implementation of service 29 which is attached to this thesis to fully emulate the ECU with error handling as defined by ISO norm and measure the memory demands. Other one and the most beneficial one would be to transfer the implementation to real device with HW constraints and measure the real life values to validate data from this model with implementation of cryptographic protocols and libraries which could be run such devices.

The last think to interest the reader would be to take a look into ISO-TP kernel module and find a bug which was encountered during the experiment and which was not solved until the end of experiment.

Appendix A

Attachments

Because of the amount of measured data. They are not attached here directly into the thesis, but they are saved at the attached media storage together with developed testing applicaton for the reader to examine.

Bibliography

1. CHARETTE, Robert N. *HOW SOFTWARE IS EATING THE CAR* [online]. IEEE, [n.d.]. Available also from: `https://spectrum.ieee.org/software-eating-car`.
2. J. A. COOK, J. S. Freudenberg. *Control Area Network (CAN)* [online]. 2008. Tech. rep. Michigan University. Available also from: `https://www.eecs.umich.edu/courses/eecs461/doc/CAN_notes.pdf`.
3. CAN bus the ultimate guide. In: [online]. CSS Electronics, 2023, chap. a simple intro to CAN.
4. CAN bus the ultimate guide. In: [online]. CSS Electronics, 2023, chap. a simple intro to CAN-FD.
5. Road vehicles — Unified diagnostic services (UDS) — Part 3: Unified diagnostic services on CAN implementation (UDSonCAN). In: [online]. ISO, 2013, chap. Overview.
6. Road vehicles — Diagnostic communication over Controller Area Network (DoCAN) — Part 2: Transport protocol and network layer services. In: [online]. ISO, 2016, chap. Introduction.
7. Road vehicles — Diagnostic communication over Controller Area Network (DoCAN) — Part 2: Transport protocol and network layer services. In: [online]. ISO, 2016, chap. Network layer overview.
8. Road vehicles — Diagnostic communication over Controller Area Network (DoCAN) — Part 2: Transport protocol and network layer services. In: [online]. ISO, 2016, chap. ISO 11898-1 CAN data link layer extension.
9. Road vehicles — Diagnostic communication over Controller Area Network (DoCAN) — Part 2: Transport protocol and network layer services. In: [online]. ISO, 2016, chap. Transport layer protocol data units.
10. Road vehicles — Unified diagnostic services (UDS) — Part 1: Application layer. In: [online]. ISO, 2020, pp. 1–40.
11. Road vehicles — Unified diagnostic services (UDS) — Part 1: Application layer. In: [online]. ISO, 2020, chap. 10.6 Authentication service.
12. TRCEK, Denis. *Managing Information Systems Security and Privacy*. [N.d.]. ISBN 9783540281047.
13. ALROWAITHY, Majed. *Performance-Efficient Cryptographic, Primitives in Constrained Devices*. 2021. PhD thesis. School of Computing, Newcastle University.
14. *The Linux Kernel* [online]. Available also from: `https://www.kernel.org/doc/html/latest/networking/can.html`.

Content of attached memory storage

- `client` ..... Samples of client clients encryption key and necessary file structures
- `command.txt` ... Sample commands to use the Canneloni tool and to run client and server of the developed application
- `dumps` ..... captured data dumps
  - `data_volume.py` ..... Python scrip which calculates volume of transferred data from the dumps
  - `8B` ..... Dumps of authentication with payload size 8 bytes
    - `8B_ECC_bidir` .. Dump of bidirectional CR authentication with ECC crypto algorithm
    - `8B_ECC_PKI_bidir` ..... Dump of bidirectional PKI authentication with ECC crypto algorithm
    - `8B_HMAC_bidir` ..... Dump of bidirectional CR authentication with HAMC crypto algorithm
    - `8B_RSA_bidir` .. Dump of bidirectional CR authentication with RSA crypto algorithm
    - `8B_RSA_PKI_bidir` ..... Dump of bidirectional PKI authentication with RSA crypto algorithm
  - `16B` ..... Dumps of authentication with payload size 16 bytes
    - `16B_ECC_bidir` . Dump of bidirectional CR authentication with ECC crypto algorithm
    - `16B_ECC_PKI_bidir` ..... Dump of bidirectional PKI authentication with ECC crypto algorithm
    - `16B_HMAC_bidir` ..... Dump of bidirectional CR authentication with HAMC crypto algorithm
    - `16B_RSA_bidir` . Dump of bidirectional CR authentication with RSA crypto algorithm
    - `16B_RSA_PKI_bidir` ..... Dump of bidirectional PKI authentication with RSA crypto algorithm
  - `32B` ..... Dumps of authentication with payload size 32 bytes
    - `32B_ECC_bidir` . Dump of bidirectional CR authentication with ECC crypto algorithm
    - `32B_ECC_PKI_bidir` ..... Dump of bidirectional PKI authentication with ECC crypto algorithm
    - `32B_HMAC_bidir` ..... Dump of bidirectional CR authentication with HAMC crypto algorithm
    - `32B_RSA_bidir` . Dump of bidirectional CR authentication with RSA crypto algorithm
    - `32B_RSA_PKI_bidir` ..... Dump of bidirectional PKI authentication with RSA crypto algorithm
  - `48B` ..... Dumps of authentication with payload size 48 bytes

- 48B_ECC_bidir. Dump of bidirectional CR authentication with ECC crypto algorithm
- 48B_ECC_PKI_bidir..... Dump of bidirectional PKI authentication with ECC crypto algorithm
- 48B_HMAC_bidir ..... Dump of bidirectional CR authentication with HAMC crypto algorithm
- 48B_RSA_bidir. Dump of bidirectional CR authentication with RSA crypto algorithm
- 48B_RSA_PKI_bidir..... Dump of bidirectional PKI authentication with RSA crypto algorithm

64B ..... Dumps of authentication with payload size 64 bytes

- 64B_ECC_bidir. Dump of bidirectional CR authentication with ECC crypto algorithm
- 64B_ECC_PKI_bidir..... Dump of bidirectional PKI authentication with ECC crypto algorithm
- 64B_HMAC_bidir ..... Dump of bidirectional CR authentication with HAMC crypto algorithm
- 64B_RSA_bidir. Dump of bidirectional CR authentication with RSA crypto algorithm
- 64B_RSA_PKI_bidir..... Dump of bidirectional PKI authentication with RSA crypto algorithm

Makefile. Makefile written to create necessary CAN infrastructure with Canneloni tool and to compile source code of the testing SW

measured_timesTimes measured on individual runs of testing application

- can_A_8_BTimes measured for authentication using the standard CAN with up to 8 bytes payload
  - ECC_PKI.resTimes measured for bidirectional authentication using the PKI and ECC crypto protocol
  - ECC_PKI_uni.resTimes measured for unidirectional authentication using the PKI and ECC crypto protocol
  - ECC.res.....Times measured for bidirectional authentication using the CR and ECC crypto protocol
  - ECC_uni.resTimes measured for unidirectional authentication using the CR and ECC crypto protocol
  - HMAC.res . Times measured for bidirectional authentication using the CR and HMAC crypto protocol
  - HMAC_uni.res ... Times measured for unidirectional authentication using the CR and HMAC crypto protocol
  - RSA_PKI.resTimes measured for bidirectional authentication using the PKI and RSA crypto protocol
  - RSA_PKI_uni.resTimes measured for unidirectional authentication using the PKI and RSA crypto protocol
  - RSA.res.....Times measured for bidirectional authentication using the CR and RSA crypto protocol
  - RSA_uni.resTimes measured for unidirectional authentication using the CR and RSA crypto protocol
- can_FD_16_BTimes measured for authentication using the CAN-FD with 16 bytes payload
  - ECC_PKI.resTimes measured for bidirectional authentication using the PKI and ECC crypto protocol
  - ECC_PKI_uni.resTimes measured for unidirectional authentication using the PKI and ECC crypto protocol
  - ECC.res.....Times measured for bidirectional authentication using the CR and ECC crypto protocol
  - ECC_uni.resTimes measured for unidirectional authentication using the CR and ECC crypto protocol

- HMAC.res Times measured for bidirectional authentication using the CR and HMAC crypto protocol
- HMAC_uni.res ... Times measured for unidirectional authentication using the CR and HMAC crypto protocol
- RSA_PKI.res Times measured for bidirectional authentication using the PKI and RSA crypto protocol
- RSA_PKI_uni.res Times measured for unidirectional authentication using the PKI and RSA crypto protocol
- RSA.res..... Times measured for bidirectional authentication using the CR and RSA crypto protocol
- RSA_uni.res Times measured for unidirectional authentication using the CR and RSA crypto protocol

can_FD_32_B Times measured for authentication using the CAN-FD with 32 bytes payload

- ECC_PKI.res Times measured for bidirectional authentication using the PKI and ECC crypto protocol
- ECC_PKI_uni.res Times measured for unidirectional authentication using the PKI and ECC crypto protocol
- ECC.res..... Times measured for bidirectional authentication using the CR and ECC crypto protocol
- ECC_uni.res Times measured for unidirectional authentication using the CR and ECC crypto protocol
- HMAC.res Times measured for bidirectional authentication using the CR and HMAC crypto protocol
- HMAC_uni.res ... Times measured for unidirectional authentication using the CR and HMAC crypto protocol
- RSA_PKI.res Times measured for bidirectional authentication using the PKI and RSA crypto protocol
- RSA_PKI_uni.res Times measured for unidirectional authentication using the PKI and RSA crypto protocol
- RSA.res..... Times measured for bidirectional authentication using the CR and RSA crypto protocol
- RSA_uni.res Times measured for unidirectional authentication using the CR and RSA crypto protocol

can_FD_48_B Times measured for authentication using the CAN-FD with 48 bytes payload

- ECC_PKI.res Times measured for bidirectional authentication using the PKI and ECC crypto protocol
- ECC_PKI_uni.res Times measured for unidirectional authentication using the PKI and ECC crypto protocol
- ECC.res..... Times measured for bidirectional authentication using the CR and ECC crypto protocol
- ECC_uni.res Times measured for unidirectional authentication using the CR and ECC crypto protocol
- HMAC.res Times measured for bidirectional authentication using the CR and HMAC crypto protocol
- HMAC_uni.res ... Times measured for unidirectional authentication using the CR and HMAC crypto protocol
- RSA_PKI.res Times measured for bidirectional authentication using the PKI and RSA crypto protocol
- RSA_PKI_uni.res Times measured for unidirectional authentication using the PKI and RSA crypto protocol

- `RSA.res` ..... Times measured for bidirectional authentication using the CR and RSA crypto protocol
- `RSA_uni.res` Times measured for unidirectional authentication using the CR and RSA crypto protocol
- `can_FD_64_B` Times measured for authentication using the CAN-FD with 64 bytes payload
  - `ECC_PKI.res` Times measured for bidirectional authentication using the PKI and ECC crypto protocol
  - `ECC_PKI_uni.res` Times measured for unidirectional authentication using the PKI and ECC crypto protocol
  - `ECC.res` ..... Times measured for bidirectional authentication using the CR and ECC crypto protocol
  - `ECC_uni.res` Times measured for unidirectional authentication using the CR and ECC crypto protocol
  - `HMAC.res` Times measured for bidirectional authentication using the CR and HMAC crypto protocol
  - `HMAC_uni.res` ... Times measured for unidirectional authentication using the CR and HMAC crypto protocol
  - `RSA_PKI.res` Times measured for bidirectional authentication using the PKI and RSA crypto protocol
  - `RSA_PKI_uni.res` Times measured for unidirectional authentication using the PKI and RSA crypto protocol
  - `RSA.res` ..... Times measured for bidirectional authentication using the CR and RSA crypto protocol
  - `RSA_uni.res` Times measured for unidirectional authentication using the CR and RSA crypto protocol
- `server` ..... Samples of server key databases and expected file structures
- `src` ..... Source code application simulating the authentication to the ECU
  - `client.cpp` ..... Source code of the client class
  - `client.hpp` ..... Header file for the client class
  - `connect.cpp` ..... Source code of the connect class
  - `connect.hpp` ..... Header file for the connect class
  - `main.cpp` .. Source code of the main function of the application, handles input parametrs and start either client or server
  - `server.cpp` ..... Source code of the server class
  - `server.hpp` ..... Header file for the server class
  - `service_29.cpp` ..... Source code of the parent class to the client and the server
  - `service_29.hpp` ..... Header file for the parent class to the client and the server
- `text` ..... Text of the thesis
  - `thesis.pdf` ..... Text of the thesis in form of PDF
- `thesis_src` ..... Source code the thesis in LaTeX