

**SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA
FACULTY OF ELECTRICAL ENGINEERING AND
INFORMATION TECHNOLOGY**

Registration number: FEI-5384-8844

**SECURE IMPLEMENTATION OF MCELIECE
CRYPTOSYSTEM
MASTER THESIS**

2016

Martin Orem

**SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA
FACULTY OF ELECTRICAL ENGINEERING AND
INFORMATION TECHNOLOGY**

Registration number: FEI-5384-8844

**SECURE IMPLEMENTATION OF MCELIECE
CRYPTOSYSTEM
MASTER THESIS**

Study Programme:	Applied Informatics
Field Number:	2511
Study Field:	9.2.9 Applied Informatics
Training Workplace:	Institute of Computer Science and Mathematics
Supervisor:	doc. Ing. Pavol Zajac, PhD.
Consultant:	Ing. František Uhrecký

Bratislava 2016

Martin Orem



ZADANIE DIPLOMOVEJ PRÁCE

Študent: **Bc. Martin Orem**
ID študenta: **8844**
Študijný program: **Aplikovaná informatika**
Študijný odbor: **9.2.9. aplikovaná informatika**
Vedúci práce: **doc. Ing. Pavol Zajac, PhD.**
Miesto vypracovania: **Ústav informatiky a matematiky**

Názov práce: **Bezpečná implementácia McEliece kryptosystému**

Špecifikácia zadania:

Cieľom práce je pokračovať vo vývoji knižnice BitPunch, ktorá implementuje McEliece kryptosystém. Práca sa zameriava na použitie McEliece kryptosystému v hybridnej schéme a v schéme na výmenu kľúča.

Úlohy:

1. Naštudujte problematiku praktickej bezpečnosti McEliece kryptosystému.
2. Naštudujte aktuálny stav knižnice Bitpunch, analyzujte potrebné vylepšenia a zmeny.
3. Navrhnite a implementujte novú verziu knižnice.
4. Otestujte riešenie a vyhodnotte výsledky.

Zoznam odbornej literatúry:


1. F. Uhrecký: Implementácia kryptografickej knižnice s McEliece kryptosystémom. Diplomová práca, 2015.
2. M. Klein: Postranné kanály v SW implementácii McElieceovho kryptosystému. Diplomová práca, 2015.
3. M. Repka, P. Zajac: "Overview of the McEliece Cryptosystem and its Security." Tatra Mountains Mathematical Publications 60.1 (2014): 57-83.

Riešenie zadania práce od: **21. 09. 2015**

Dátum odovzdania práce: **20. 05. 2016**




Bc. Martin Orem
študent


prof. RNDr. Otokar Grošek, PhD.
vedúci pracoviska


prof. RNDr. Gabriel Juhás, PhD.
garant študijného programu

SÚHRN

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Študijný program:	Aplikovaná informatika
Autor:	Martin Orem
Diplomová práca:	Bezpečná implementácia McEliece kryptosys- tému
Vedúci záverečnej práce:	doc. Ing. Pavol Zajac, PhD.
Konzultant:	Ing. František Uhrecký
Miesto a rok predloženia práce:	Bratislava 2016

Post-quantová kryptografia vznikla ako odpoveď na významné pokroky vo vývoji kvantového počítača. Niekoľko kryptosystémov zostalo odolných aj voči útočníkom s prístupom ku takémuto počítaču. McElieceov kryptosystém, ktorý bol preskúmaný mnohými odborníkmi na kryptografiu vypadá byť jedným z najlepších kandidátov pre post-quantovú kryptografiu. Avšak pôvodná verzia nie je odolná voči rôznym pokročilým útokom. Tieto problémy boli riešené aj pomocou CCA bezpečných konverzií, ktoré v práci spomíname. Sústreďime sa na využitie McElieceovho kryptosystému vo viacerých schémach. Prezentujeme nový spôsob v navrhovaní hybridnej schémy, ktorý využíva špecifické vlastnosti kryptosystému. Ďalším prínosom je schéma na výmenu kľúčov podporujúca takzvanú "forward secrecy". Hlavným cieľom tejto práce je implementovať spomenuté schémy a overiť ich praktickú využiteľnosť.

Kľúčové slová: kryptografia, KEM, McEliece kryptosystém

ABSTRACT

SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA
FACULTY OF ELECTRICAL ENGINEERING AND INFORMATION TECHNOLOGY

Study Programme:	Applied Informatics
Author:	Martin Orem
Master Thesis:	Secure implementation of McEliece Cryptosystem
Supervisor:	doc. Ing. Pavol Zajac, PhD.
Consultant:	Ing. František Uhrecký
Place and year of submission:	Bratislava 2016

Post-quantum cryptosystems have been developed in the last decade in response to the rise of quantum computing research. McEliece cryptosystem based on the general decoding problem seems to be one of the best candidates for post-quantum cryptography. However, the original version of the cryptosystem is prone to various advanced attacks. Most of the problems were solved by introducing CCA-secure schemes overviewed in the thesis. We focus on a McEliece cryptosystem application within various schemes. We introduce a new way of designing a hybrid scheme featuring McEliece cryptosystem. Furthermore, we present a new scheme for exchanging keys supporting perfect forward secrecy. The main goal of the thesis is to implement mentioned schemes and to verify their practical use.

Keywords: crypto, McEliece, KEM, CCA

Declaration

I declare that I developed this diploma thesis myself with listed references.

Acknowledgments

I would like to express my deepest gratitude to my supervisor, doc. Ing. Pavol Zajac PhD., for his support and guidance. Also, I would like to say thank you to my consultant.

Contents

Introduction	12
1 Preliminaries	14
1.1 Notation	14
1.2 McEliece cryptosystem	14
1.3 IND-CCA security	16
1.4 Hybrid Encryption framework	17
1.5 Needham-Schroeder Protocol	18
1.6 Forward secrecy	18
1.6.1 Ephemeral key	19
1.7 Other cryptographic tools	19
1.7.1 Cryptographic hash function	19
1.7.2 Key derivation functions	20
1.7.3 Message authentication code	20
1.8 Initialization vector	21
2 Analysis	22
2.1 Bitpunch library	22
2.1.1 Modular architecture	22
2.1.2 ASN.1	23
2.2 mbed TLS	24
2.3 CCA conversions	26
2.3.1 Pointcheval conversion	26
2.3.2 Kobara and Imai conversion	26
2.4 Hybrid scheme based on Niederreiter PKC	29
3 Design of cryptographic schemes	31
3.1 Hybrid scheme based on McEliece PKC	31
3.2 Key exchange scheme supporting forward secrecy	33
4 Implementation of proposals	36
4.1 Used cryptographic algorithms	37
4.1.1 AES-GCM	37
4.1.2 PBKDF2	39
4.2 Hybrid scheme based on McEliece PKC	40

4.2.1	Messages and padding	40
4.3	Key exchange scheme supporting forward secrecy	41
5	Tests	43
	Conclusion	46
	Resumé	47
	Bibliography	51
6	Structure of attached medium	55

List of Figures and Tables

Figure 1	[33] Modular architecture of Bitpunch	23
Figure 2	[28] Use case of mbed TLS	25
Figure 3	[28] Kobara-Imai CCA-2 secure conversion of MECS	27
Figure 4	Hybrid scheme	32
Figure 5	Key exchange scheme	34
Figure 6	The class diagram	36
Figure 7	[28] AES encryption	38
Figure 8	The padding scheme, in which a message is shorter than MECS block	41
Figure 9	The padding scheme, when a message is longer than MECS block . .	41
Figure 10	Time complexity of operations	45
Figure 11	Time complexity of operations except initializations and a key generation	45
Table 1	Comparison between schemes and data redundancy	33
Table 2	Total data transmitted	33
Table 3	Performance test	43
Table 4	Performance test of encryption	43
Table 5	Memory footprint	44

List of Abbreviations and Symbols

CCA2 - Adaptive Chosen-Ciphertext Attack

MECS - McEliece Cryptosystem

OID - Object Identifier

PKCS - Public Key Cryptosystem

w/o - without

w/ - with

IND-CCA - indistinguishability under chosen ciphertext attack

KEM - key-encapsulation mechanism

DEM - data-encapsulation mechanism

MitM - Man in the middle

ASN.1 - Abstract syntax notation one

List of Algorithm

1	[33] Key generation	15
2	[33] McEliece encryption	15
3	[33] McEliece decryption	15
4	[17] The CCA experiment	16
5	A single query attack	17
6	[22] Needham-Schroeder public key protocol	19
7	[33] The Pointcheval CCA2 conversion	26
8	[18] Kobara-Imai CCA2 gamma conversion - encryption	27
9	[18] Kobara-Imai CCA2 gamma conversion - decryption	28
10	Hybrid scheme based on MECS	40
11	Key exchange scheme	42

Introduction

The majority of current cryptography relies on problems, which could be solved by the probabilistic algorithm using the quantum computer [30]. Therefore, it is necessary to have sufficient alternative algorithms to replace broken ones. The term post-quantum cryptography refers to a set of cryptographic algorithms that should be secure against an attack by a quantum computer.

We can divide these algorithms to the following groups with the respective representatives.

- Cryptography based on hash functions - Merkle's hash tree representing a signature scheme with a public key.
- Cryptography based on coding theory - McEliece and Niederreiter cryptosystems.
- Lattice based crypto systems - Hoffstein Pipher Silverman cryptosystem with a NTRU public key.
- Multivariate cryptography - Patarin's signature scheme utilizing hidden field equations.
- Symmetric cryptography - AES designed by Daemen and Rijmen.

The aforementioned McEliece cryptosystem is one of the unbroken candidates for the post-quantum cryptography. R.J. McEliece has proposed a cryptosystem based on error code problem, which was proven to be NP-hard [23].

The system in its original version is neither cryptographically nor semantically secure. The original version suffers from various kinds of attacks including susceptibility to an adaptive chosen cipher text attack [9, 2]. Many of these problems can be avoided by using CCA2-secure conversion of the McEliece cryptosystem.

Moreover, it is important to implement sufficient post-quantum algorithms to popular open protocols. These protocols use a hybrid encryption scheme, which features both asymmetric and symmetric cryptography to increase efficiency. Decryption using symmetric cryptography is much faster to execute on a computer than public key decryption. In practice, they are often used together, so that a public-key is used to encrypt a randomly generated secret key, and the secret key is used to encrypt the actual message. The only known scheme based on post-quantum cryptography was presented by Eduardo Perischetti [27]. The scheme features Niederreiter cryptosystem, which is modified version

of McEliece using dual codes.

As a goal of this diploma thesis, we aim to address problems regarding to practical usage of McEliece cryptosystem. Since symmetric cryptography is still considered to be resistant to quantum computers, we utilize a symmetric cipher to solve problems with a chosen-cipher text attack.

This first chapter introduces the reader to the background and preliminaries needed throughout this thesis. Section 2 gives an overview of already existing implementations of code-based cryptosystem within the Bitpunch project. The respective ways of storing and transferring keys are discussed. The section also outlines encryption and decryption process in CCA-secure schemes. Basic information about mbed TLS, a cryptographic library used for symmetric cryptography are provided. Section 3 specifies the goals of this thesis in more detail and points out the contributions of this work to the research process. Section 4 discusses implementation details. Section 5 presents tests results and compares our solution with another cryptographic library implementing a classical hybrid scheme. In the final part, a brief summary is available with a discussion of some problems covered by this thesis.

1 Preliminaries

In this section, we introduce selected important definitions, which are required throughout the thesis. We also present notations, and a short overview of McEliece PKCS (MECS) and its resistance against a chosen-cipher text attack (CCA). In the following subsections, hybrid framework and Needham-Schroeder protocol are discussed. We conclude the section with cryptographic algorithms and primitives used in further sections.

1.1 Notation

If x is a string, then $|x|$ denotes its length, while if S is a set then $|S|$ denotes its size. If S is a set then $s \leftarrow S$ denotes the operation of picking an element s of S uniformly at random. If $k \in \mathbb{N}$ then 1^k denotes the string of k ones. If P is a matrix, then P^{-1} denotes an inverse matrix to the matrix P .

1.2 McEliece cryptosystem

MECS was proposed by R. J. McEliece in 1978 [23]. The cryptosystem is a typical public key cryptography system. These systems are characterized by using two different keys for encryption and decryption. MECS is considered to be one of the best candidates for post-quantum cryptography, which causes a growing interest in implementation and further cryptanalysis.

The original construction of MECS is based on Goppa codes. This version of PKCS seems to be secure with the right choice of parameters. Goppa codes are well suited for cryptographic application due to its generator matrix, which is hard to be distinguished from a random binary matrix and also for their high error-correcting capabilities. Many other variants with various linear codes of the cryptosystem have been proposed over the years, but the most of them were subsequently proven to be insecure [20].

Security of MECS relies on a decoding problem, which is known to be NP-hard [7] and should also resist attackers with access to a quantum computer. Although a general decoding problem is NP-hard, a careful choice of system parameters (n, k, t) is required to ensure that the estimated security level of MECS is sufficient. Even with carefully chosen system parameters, MECS is not cryptographically secure, with various kinds of attacks exploiting implementation weaknesses rather than a decoding problem.

MECS takes advantage from randomness. A public key is "permuted" and "varied" form of a chosen-linear code (secret key), which should be hard to distinguish from a completely random linear code. We can formally define McEliece cryptosystem by describing algorithms $\Pi = (Gen, Enc, Dec)$ as follows:

Key generation algorithm In order to generate keys, we have to define a Goppa code constructed over irreducible polynomial. The output of this algorithm is:

- Private key - S, G, P (random singular matrix, generation matrix, permutation matrix).
- Public key - \hat{G}, t (masked generation matrix and the number of errors).

Algorithm 1 [33] Key generation

1. Pick a random irreducible polynomial g over $GF(2^m)$ of a degree t ,
 2. Compute a $k \times n$ generation matrix G , of goppa code $\Gamma = (\alpha_1, \dots, \alpha_n, g)$, with a dimension $k = n - td$,
 3. Generate a random $k \times k$ singular matrix S ,
 4. Generate a random $n \times n$ permutation matrix P ,
 5. Compute a $k \times n$ matrix $\hat{G} = SGP$,
 6. Public key is the pair of (\hat{G}, t) , where the t is maximum of corrected errors, private key consists of S, G, P .
-

Algorithm 2 [33] McEliece encryption

1. Let m be a k -bit message, and let e be a random n -bit vector with $wH(e) \leq t$. Then $c = m \cdot \hat{G} + e$ is a ciphertext.
-

Algorithm 3 [33] McEliece decryption

1. Calculate S^{-1} and P^{-1} ,
 2. Calculate $\mathbf{c}' = \mathbf{c}P^{-1}$,
 3. Apply the proper decoding algorithm Dec of the code Γ to decode \mathbf{c}' to $\hat{\mathbf{m}}$,
 4. Obtain m by $\mathbf{m} = \hat{\mathbf{m}}S^{-1}$.
-

The most important part of a secret key is the description of the structured linear code, created by an irreducible polynomial in a key-generation process. An efficient

decoding algorithm for the chosen linear code is required for successful decryption of messages. The original construction uses irreducible binary Goppa codes, for which an efficient decoding algorithm was presented by Patterson [26]. In order to apply Patterson's algorithm, the polynomial generating the Goppa code must be known.

1.3 IND-CCA security

In order to define IND-CCA, we need to consider the following experiment with a private-key encryption scheme $\Pi = (Gen, Enc, Dec)$, adversary A , and a value n for the security parameter.

Algorithm 4 [17] The CCA experiment

1. A random key k is generated by running $Gen(1^n)$.
 2. The adversary A is given the input 1^n and the oracle access to encryption and decryption. It outputs a pair of messages m_0, m_1 s.t $|m_0| = |m_1|$.
 3. A random bit $b \leftarrow \{0, 1\}$ is chosen, and then a cipher text is computed and subsequently given to A . We denote the challenge as c .
 4. Adversary continues to have an access to the oracle encryption and decryption, however it is not allowed to query challenge c later.
 5. The output of the experiment is 1 if $b' = b$, otherwise it is 0.
-

A private-key encryption scheme Π is CCA-secure, if for all probabilistic polynomial-time adversaries A , there exists a negligible function $negl$, such that:

$$Pr[PrivK_{A,\Pi}(n) = 1] \leq \frac{1}{2} + negl(n)$$

where the resulted probability is taken over by all random coins used in the experiment.

MECS is not CCA-secure since a single-query attack can be modelled as follows:

Algorithm 5 A single query attack

1. $m' := \text{decrypt}(c + \delta \hat{G})$
 2. $m := m' + \delta$
 3. if $(m = m_0)$ the output is $b' = 0$, else $b' = 1$
-

1.4 Hybrid Encryption framework

The following subsection is adapted from [27]. A Hybrid Encryption scheme is a cryptographic protocol that features both a symmetric encryption scheme and a public-key encryption scheme. An asymmetric encryption scheme is used for a purpose of transmission of a symmetric key between participants. The participants can exchange the symmetric key and subsequently use a symmetric scheme, which is suitable and faster for encrypting bulk data. We have to consider the security of this scheme complexly.

The hybrid encryption framework was first introduced in a seminal work by Cramer and Shoup [10] with an example based on the decisional Diffie-Hellman assumptions. The framework consists of two components. The first component using the asymmetric scheme is known as Key Encapsulation Mechanism (*KEM*), while the second using the symmetric scheme is referred to as a Data Encapsulation Mechanism (*DEM*). *KEM* can be defined as $KEM = (Gen, Enc, Dec)$, where *Gen*, *Enc*, *Dec* are polynomial-time algorithms described below.

KeyGen A probabilistic key generation algorithm that takes as an input a security parameter $(1)^\lambda$ and outputs a public key pk and a private key sk .

Enc A probabilistic encryption algorithm that receives as an input a public key pk and returns a key/ciphertext pair (K, ψ_0) .

Dec A deterministic decryption algorithm that receives as input a private key sk and a ciphertext ψ_0 and outputs either a key K or the failure symbol \perp .

DEM comprises two algorithm $DEM = (Enc, Dec)$, which are utilized as an encapsulation layer for a message.

Enc A deterministic encryption algorithm that receives as an input a key K and a plaintext ϕ and returns a ciphertext ψ_1 .

Dec A deterministic decryption algorithm that receives as an input a key K and a ciphertext ψ_1 and outputs either a plaintext ϕ or the failure symbol \perp .

As presented in Section 3, we will use a modified hybrid encryption that can not be properly split into a KEM/DEM part. In this case, we can compare our model to a general public key encryption with arbitrary long messages.

1.5 Needham-Schroeder Protocol

Needham-Schroeder Public-Key Protocol refers to a key protocol based on public-key cryptography. The protocol was supposed to provide mutual authentication between two communicating parties, however, in its original, it is considered insecure [22]. An agent A possesses a public key PKA , which can be obtained from a trusted key servers. The agent also possess a respective secret key SKA . Every agent can encrypt a message m for the agent A using PKA , denoted as $\{m\}_{PKA}$. We assume that only A can decrypt the message properly, which ensures confidentiality.

The protocol utilizes nonces. The nonces are pseudo -random numbers that should be used just once. The nonces denoted as NA and NB represent numbers generated by both parties A and B . The original full form of the protocol is prone to a replay attack [22], where an attacker can compromise operations of both parties by using old public keys. The flaw is caused by the absence of the proof of freshness. The design flaw has been addressed by introducing a modification of a message to include the responder's identity. The aforementioned protocol is presented as Algorithm 6. The scheme is often referred to as the Needham-Schroeder-Lowe protocol [22]. However, this protocol does not support forward secrecy.

In Section 3, we present a key exchange protocol, which is mainly inspired by Needham-Schroeder-Lowe, but includes additional nonces that are encrypted with an ephemeral key to achieve forward secrecy.

1.6 Forward secrecy

If an attacker is able to learn a party's private key or keys, a potential vulnerability of recovering derived keys exists. The longer (the period) these keys are used, the more of derived keys are endangered. In order to limit a private key's potential vulnerability, the private keys should be removed after their period expires. This prevents even a passive attacker who successfully recorded past communications encrypted with the shared secret keys from decrypting them some time in the future by compromising the parties. One-party forward secrecy means that attackers's knowledge of that party's cryptographic

Algorithm 6 [22] Needham-Schroeder public key protocol

1. A queries the server S for B 's public key
 2. S returns to A an encrypted message $m = \{PK_B, B\}_{SK_S}$, where B denotes the identity of the agent B
 3. A generates a random nonce and sends an encrypted message to B , $m1 = \{NA, A\}_{PK_B}$
 4. B also wants to communicate, therefore B queries the server S for A 's public key
 5. S returns to B an encrypted message $m2 = \{PK_A, A\}_{SK_S}$, where A denotes the identity of the agent A
 6. B generates a random nonce and sends it to A along with the identity, $m4 = \{NA, NB, B\}_{PK_A}$
 7. A decrypts the message to confirm that A is able to decrypt NB to B
-

state after a key-agreement operation does not enable him to recover the derived keys anymore. The two-party forward secrecy means that an opponent's knowledge of both parties' cryptographic state does not enable recovery of the previously derived keys [15].

1.6.1 Ephemeral key

In order to achieve forward secrecy, NIST Special Publication 800-57 Part 1 [6] defines an ephemeral key as a cryptographic key that meets requirements of the key type and is usually generated for each execution of a key establishment process (for instance, unique to each session). "Public ephemeral key-agreement keys are used for a single key agreement transaction. The cryptoperiod of the public ephemeral key-agreement key ends immediately after it is used to generate the shared secret. Note that in some cases, the cryptoperiod of the public ephemeral key-agreement key may be different for the participants in the key-agreement transaction.[6]"

1.7 Other cryptographic tools

In this section we survey other cryptographic tools that we need for our constructions.

1.7.1 Cryptographic hash function

A cryptographic hash function [35] is a function that takes an arbitrary block of data and returns a fixed-size bit string, denoted as a cryptographic hash value. Any change in

a block of data means a highly probable change of the hash value.

”The ideal cryptographic hash function has four main properties:

- it is easy to compute the hash value for any given message
- it is infeasible to generate a message from its hash
- it is infeasible to modify a message without changing the hash
- it is infeasible to find two different messages with the same hash

” [35] We typically denote a hash function by $HASH(seed)$. In concrete instances, we can choose a hash function from a set of recommended algorithms, such as SHA-2 family [11] or a newer SHA-3 standard [25].

1.7.2 Key derivation functions

A Key Derivation Function (KDF) [27] is a function that takes a string x of an arbitrary length (greater than 0) and outputs a randomly looking bit string of an arbitrary length. The KDF in the best case is modeled as a random oracle, and it satisfies the entropy smoothing property: ”if x is chosen at random from a high entropy distribution, the output of the KDF should be computationally indistinguishable from a random bit string” [27]. We use a KDF in our hybrid scheme to derive secondary keys and other parameters from the entropy encoded in an error vector. A suitable alternative for future workings can be an extensible output function such as SHAKE [25].

1.7.3 Message authentication code

Standard ISO/IEC 9797-2:2011 [1] defines a MAC algorithm for computing a function which maps arbitrary lengths strings of bits and a secret key to fixed-length strings of bits, satisfying the following three properties:

- the function should be efficient for any arbitrary input string and any key,
- with given no prior knowledge of the fixed key the computation of the output value from the function on any new input should be infeasible,
- even with knowledge of input strings and relating function outputs, it is computationally infeasible to predict the output of a further input string

MACs are used to provide a cryptographic integrity in symmetric encryption schemes. We use it in a hybrid scheme to detect that a ciphertext was not tampered with by an attacker during the transfer.

1.8 Initialization vector

An initialization vector (IV) is a fixed-size input that needs to be random or pseudorandom. In order to achieve semantic security a randomization of the vector is a key factor. NIST Special Publication 800-38A [14] recommends that an IV must be properly generated for each encryption operation, and the same IV is necessary for the respective execution of the decryption. Thus, the information that is sufficient to determine the IV or the IV itself must be available to each party involved in the communication. An IV does not necessarily need be secret, thus, the IV, or suitable information to calculate the IV, can be transferred together with the ciphertext. In our proposal, we generate IVs from the provided entropy and static salt using a KDF and, thus, the IV is not transmitted within the message directly. Therefore, the IV is unknown to the attacker.

2 Analysis

In this section, we analyse the concrete details required to implement a code-based hybrid encryption scheme and a key-exchange scheme, respectively.

We base our implementation on existing libraries:

- Bitpunch library [33] - a lightweight implementation of MECS developed at the Slovak University of Technology
- mbed TLS [4] - an open source library that implements many cryptographic primitives including KDF, MAC and symmetric encryption algorithms.

A cryptographic library called Bitpunch [33] implements MECS operations. However, since the Bitpunch is still in a development phase, various features still need to be implemented and included to the library. A modified generic Pointcheval conversion [31] has already been implemented, but an attack reducing complexity of a brute force attack exists [36]. Therefore, we analyse other CCA2 conversions, but subsequently show that we can use a hybrid scheme to address the problems. We want to test the scheme within a scheme for exchanging keys, which is another usage scenario for public key cryptography.

2.1 Bitpunch library

In this section, we present the current status of Bitpunch library, which is a very lightweight cryptographic library written in C language. The library provides the following features:

- several mathematic operations on vector space over $GF(2^m)$
- generation of a keypair for MECS with $m = 11$, $t = 50$,
- an adapted version of Pointcheval conversion,
- encryption/decryption of one block

2.1.1 Modular architecture

Modular architecture of the library provides an easy extensibility. The modules are interconnected via relationships as depicted on Figure 1.

- asn1 - provides an import/export of keys using libtasn1, code - provides implementation of used codes (Goppa, QC-MDPC),

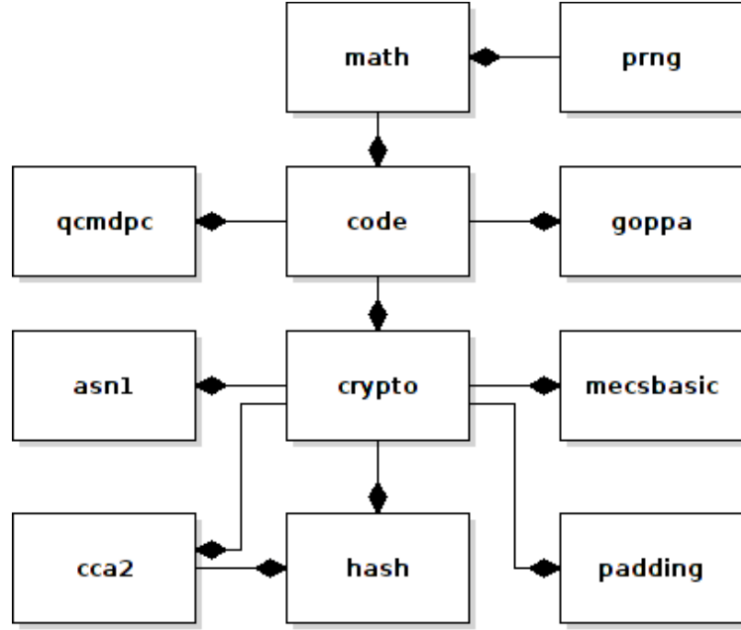


Figure 1: [33] Modular architecture of Bitpunch

- crypto:
 - cca2 - implementation of CCA2 conversions,
 - hash - implemented hash from PolarSSL
 - mecsbasic - implementation of the basic McEliece PKC,
 - padding - implementation of padding,
- math - provides arithmetic functions over $Gf(2^m)$, etc.
- prng - API for a pseudo-random generator

We consider the modular architecture and the design of the architecture taking into account future extensibility.

2.1.2 ASN.1

Abstract Syntax Notation One (ASN.1) is a telecommunication standard, which defines an interoperability between various systems [32]. ASN.1 belongs to open standards and it is widely implemented among open source solutions. The standard is dedicated to encoding and representation of basic data types such as string, integers, structures, etc. Therefore ASN.1 defines the way of encoding data, so another party can treat them in the proper manner.

Cryptography utilizes ASN.1 for encoding of various data types, that are not strictly defined in protocols.

Object Identifier (OID) is a mechanism developed by International Telegraph and Telephone Consultative Committee, International Engineering Consortium and International Organization for Standardization. OID is very often used in an asymmetric cryptography for identification of used hash algorithms, encryption algorithms with various modes.

Bitpunch allows us to serialize and deserialize keys for encryption and decryption, respectively. This is ensured by a standard library called libtasn1, which supports ASN.1 notation. The library also provides encoding to DER format. McEliece PKCS can be identified according to the above mention OID. Bitpunch applies a similar notation as the one used in BouncyCastle [21].

- 1.3.6.1.4.1.8301.3.1.3.4.2.2 - McEliece Pointcheval PKCS,
- 1.3.6.1.4.1.8301.3.1.3.4.1 - McEliece PKCS

The Bitpunch library [33] defines own structure for the notation, that includes only essential data for recovering keys.

Listing 1: Private key ASN.1

```
MecsPriKey ::= SEQUENCE {
oid OBJECT IDENTIFIER, — oid
t INTEGER, — error capability of code
mod INTEGER, — field polynomial GF(2m)
m INTEGER, — degree of field polynomial
g OCTET STRING, — goppa polynomial
p OCTET STRING, — permutation
h_mat OCTET STRING — control matrix H over GF2[x]
}
```

2.2 mbed TLS

In order to implement our proposals, we need to either implement or include cryptographic functions and algorithms. Considering the fact that one of our goals is to keep the library as small as possible, we decided to use aforementioned cryptographic algorithms from a well-known library mbed TLS [4], previously known as PolarSSL.

The mbed TLS library has been reviewed by many researchers, therefore a lot of bugs

have already been fixed with respective CVEs [5]. A CVE identifier is a unique number that can be used over different security advisories by different vendors to refer to the same issue.

The library implements the TLS and SSL protocols and the respective cryptographic algorithms. It is dual-licensed with the Apache License version 2.0. GPL2 allows programmers to use the software for non-commercial projects, thus the library perfectly suits for this project.

The library is written in the C programming language and implements the TLS/SSL stack with basic cryptographic functions providing various utility functions. Due to the fact that the library is implemented in low level C without external dependencies, it works on most operating systems and architectures. Slightly different versions are also available for Linux, Windows, OS X, Android, iOS. Supported chipsets include ARM, MIPS, x86, PowerPC. One of the beneficial features is that design allows the library to operate on small embedded devices, with the minimum complete TLS stack requiring very low resources [4].

The library is highly modular, since each component, such as a cryptographic function or algorithm can be used independently from the rest of the framework. mbed TLS also provides high-level design, API documentation and a source code documentation.

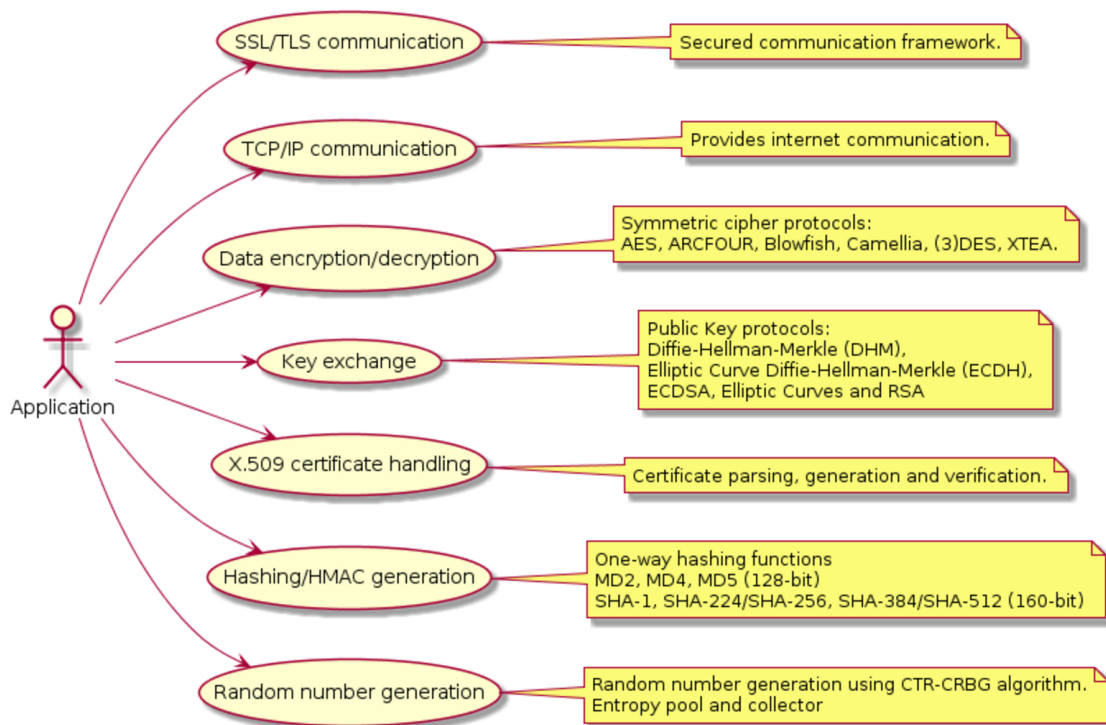


Figure 2: [28] Use case of mbed TLS

2.3 CCA conversions

The original version of MECS is neither secure in theoretical nor in cryptographical way. In practice, a PKCS can be considered secure if it provides IND-CCA security. The original version of MECS is also susceptible to partially-known plaintext, related plaintext, reaction attacks and also does not provide security against malleability [18]. These advanced attacks can be mitigated by using CCA2 secure variants. The current version of Bitpunch includes a modified version of Pointcheval conversion.

2.3.1 Pointcheval conversion

The modified version of Pointcheval conversion is already implemented in the library. However, this version of Pointcheval conversion is susceptible to the attack, where the complexity of a brute force attack is reduced [36].

Algorithm 7 [33] The Pointcheval CCA2 conversion

1. An input of MECS is $\widehat{m} = r_1 || \text{HASH}(m || r_2)$, r_1 is a random bit vector of length $k - l$ and output of the hash function is indistinguishable from random l -bit vector,
 2. A ciphertext consists of $(c_1, c_2, c_3) = (\widehat{m}\widehat{G} + e, \text{HASH}(r_1) + m, \text{HASH}(e) + r_2)$.
-

The attack is applicable, when incorrect parameters for MECS are chosen. If \widehat{G} is in a systematic form and a message m is distinguishable from a random bit string, an attacker is able to extract r_1 . The bit string is influenced by $t' = t(k - l)/n$ errors from a vector e . The attacker computes $r'_1 = r_1 + e$. The attacker tries to guess t' errors on positions from 0 up to $k - l - 1$. If $m' = c_2 + \text{hash}(r'_1 + e)$ for the given error vector is distinguishable from a random bit string, the attacker succeeded and can decrypt the message.

2.3.2 Kobara and Imai conversion

Kobara and Imai [18] proposed two generic schemes that can be used to mitigate above mentioned attacks. These schemes require the use of random oracles (hash functions) to randomize the input and break relations of the plaintext and ciphertext. Unfortunately, conversions cause an overhead since some redundant data are necessary. As depicted in the Figure 3, an algorithm for encryption is straightforward and can be described as follows: The input of this process is a public key (G', t) , a message m and a constant $Const$. The output of encryption is a ciphertext. The encryption and decryption processes are described respectively by algorithms 8 and 9.

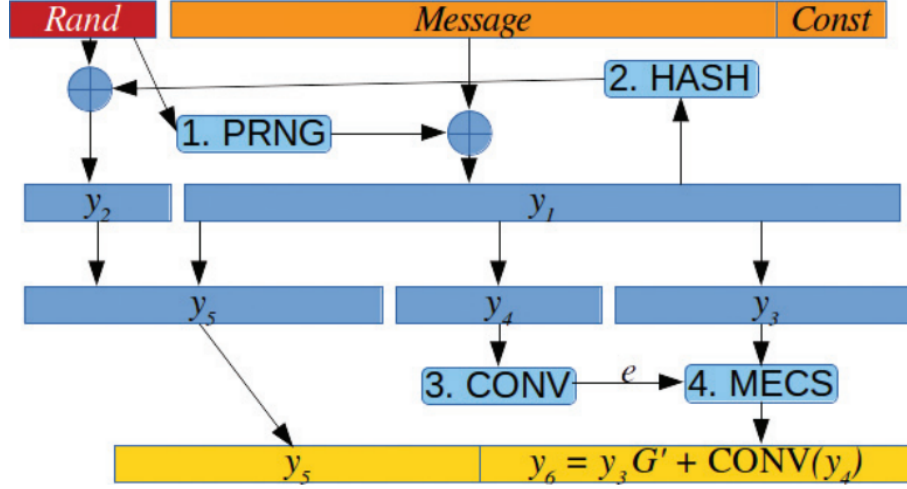


Figure 3: [28] Kobara-Imai CCA-2 secure conversion of MECS

Algorithm 8 [18] Kobara-Imai CCA2 gamma conversion - encryption

1. Let m be a message, $Const$ a constant and $Rand$ a random r -bit number,
 2. Calculate $y_1 = Prng(Rand) + (m || Const)$,
 3. Calculate $y_2 = HASH(y_1) + Rand$,
 4. Let $y_1 = y_1' || y_4 || y_3$,
 5. Let $y_5 = y_2 || y_1'$,
 6. Encode error vector $e = CONV(y_4)$,
 7. Encrypt y_3 using MECS such that $y_6 = MECS_{enc}(e, y_3) = y_3 G' + CONV(y_4)$,
 8. A ciphertext is $c = y_5 || y_6$.
-

Algorithm 9 [18] Kobara-Imai CCA2 gamma conversion - decryption

1. Let c be the received message, $c = y_5 || y_6$ and the constant $Const$,
 2. Decrypt y_6 using MECS $(e, y_3) = MECS_{dec}(y_6)$, where e is the error vector,
 3. Decode $y_4 = CONV^{-1}(e)$,
 4. Let $y_5 = y_2 || y'_1$, we reconstruct y_2 and y_1 such that $y_1 = y'_1 || y_4 || y_3$,
 5. Calculate $Rand = y_2 + HASH(y_1)$,
 6. Calculate $(m || Const') = PRNG(Rand) + y_1$,
 7. If $Const'$ is equal to $Const$, the integrity of the message is not corrupted,
 8. The plaintext is m .
-

In the figure 3 the following notation is used:

- $PRNG$ is a cryptographically secure pseudo-random generator (can be for instance a symmetric encryption algorithm).
- $Rand$ is a random session key (with the size given by the demanded security level for symmetric encryption).
- $Const$ is a public constant (with the size given by the demanded level of security for authentication).
- $HASH$ is a cryptographically secure hash function.
- $CONV$ is a bijection function, which converts an integer into a respective error vector.
- $MECS$ is a standard MECS routine.

The conversion scheme requires a function $CONV$ to be used in order to encode a number between 0 and $(n|t)$ (binomial coefficient) into a binary vector of length n and weight t . The $CONV$ function can be implemented using arithmetic coding, which seems to be suitable for this case. Arithmetic coding [19] is a form of encoding utilizing entropy and probability. In majority of codes, a string of characters are represented using a fixed number of bits per character. A string encoded with arithmetic encoding stores frequently

used characters within fewer bits and occasionally occurring characters are stored with more bits, leading to fewer bits used. Unlike other forms of entropy encoding, such as Huffman coding, the arithmetic coding does not replace a character with code words. On the contrary, arithmetic coding encodes the entire message into a single number. The main problem with *CONV* is that it is relatively slow especially in comparison with generating a random error vector directly. Note that Kobara-Imai scheme can be extended to a hybrid encryption scheme by allowing an arbitrary long message m .

2.4 Hybrid scheme based on Niederreiter PKC

Edoardo Persichetti [27] presents the KEM model based on Niederreiter cryptosystem [24], which is often considered a dual version of the McEliece cryptosystem. Niederreiter cryptosystem is also based on the hardness of syndrome decoding problem. This model provides IND-CCA security in the random oracle model. The following scheme [27] is simply a composition of components:

Setup Fix public system parameters $q, n, k, w \in N$, then choose a family F of w -error-correcting $[n, k]$ linear coders over Fq .

K_{publ} The set of $(n - k) \times k$ over Fq .

K_{priv} The set of code descriptions for F .

P The set of binary strings $\{0, 1\}^m$.

C The set of triples formed by a vector of $Fq^{(n-k)}$, a bit string of length l and a tag.

KeyGen Generate at random a code $C \in F$ given by its code description ∇ and compute its parity-check matrix in systematic form $H = (M|I_{n-k})$. Publish the public key $M \in K_{priv}$ and store the private key in $\nabla \in K_{publ}$.

Enc As an input takes a public key M and a plaintext $\phi \in P$, choose a random $e \in W_{q,n,w}$, set $H = (M|I_{n-k})$, then compute $K = KDF(e, m + l_{MAC})$ and $\psi_0 = He^T$. Parse K as $(K_1||K_2)$ then compute $\psi' = K_1 \oplus \phi$, set $T = \psi'$ and evaluate $\tau = Ev(K_2, T)$. Return the ciphertext $\psi = (\psi_0||\psi'||\tau)$.

Dec As an input takes a private key ∇ and a ciphertext ψ , first parse ψ as $(\psi_0||\psi_1)$, then compute $Decode_{\nabla}(\psi_0)$. If the decoding succeeds, use its output e to compute $K = KDF(e, m + l_{mac})$. Otherwise, determine K as a pseudorandom function of ψ_0 . Parse ψ_1 as $(\psi'||\tau)$ then parse K as $(K_1||K_2)$, set $T = \psi'$ and apply the MAC

algorithm to obtain $\tau' = Ev(K_2, T)$. If $\tau' \neq \tau$ the verification fails, hence return \perp . Otherwise, compute $\phi = K_1 \oplus \psi'$ and return the plaintext ϕ .

This is the first KEM based on a coding theory problem with a simple construction. Similarly to our proposal, a random generated error vector with sufficient security properties is utilized. It was also shown that the Hybrid Niederreiter encryption scheme, that makes use of KEM, satisfies the most important notion of anonymity IK-CCA [27]. This means that an anonymous PKE scheme does not disclose information about which key, out of the set of valid keys, has been used for encryption.

3 Design of cryptographic schemes

It is important to employ post-quantum algorithms to popular open protocols and open-source libraries that implement them. These protocols include a hybrid encryption scheme, which features both asymmetric and symmetric cryptography to increase the efficiency. Our goal is to design a hybrid scheme that exploits special properties of MECS, in order to decrease data redundancy. Afterward, this hybrid scheme is utilized to design a scheme intended to secure key exchange supporting forward secrecy.

3.1 Hybrid scheme based on McEliece PKC

The hybrid scheme depicted in Figure 4 utilizes error vectors to transfer key material. This was inspired by the hybrid scheme presented in Section 2.4. We have also adapted the approach used in CCA-secure paddings: the scheme allows the recipient to verify the integrity of messages.

We do not provide a security proof for this scheme. However, we point out the following security properties:

1. An error vector e is randomly generated from a huge space of possible error vectors. This space should be larger than the domain of $HASH(.)$. Under this condition, $SK = HASH(e)$ can also be considered a random value. SK is then used to derive encryption and authentication key ke, ka .
2. After authenticated encryption, $(c_dem|s)$ should be indistinguishable from a random bitstring for an attacker. Also, it should be infeasible for the attacker to supplant $(c_dem|s)$ by a related bitstring that can be verified as authentic by the recipient if the attacker does not know the original message.
3. The message $(c_dem|s)$ can be sent even unencrypted, the McEliece cryptosystem is only used to transmit the session key along with the message.
4. To obtain the session key, the attacker needs to find the original error vector e . If the attacker were able to do this, he could break the MECS.
5. Any manipulation with ciphertext is detected by checking authentication tags. There is a potential security hole here: if the attacker changes the MECS encrypted part in such way that it cannot be decoded, the legitimate recipient must continue the algorithm to the end in the same way, as if no problem had been detected in

order to prevent reaction attacks. This means that we cannot abort the encryption, if more than t errors are detected. Furthermore, e must be encoded as a constant-length bitstring (not a vector of t positions) to avoid timing attacks (computation time of HASH might depend on the length of representation of e).

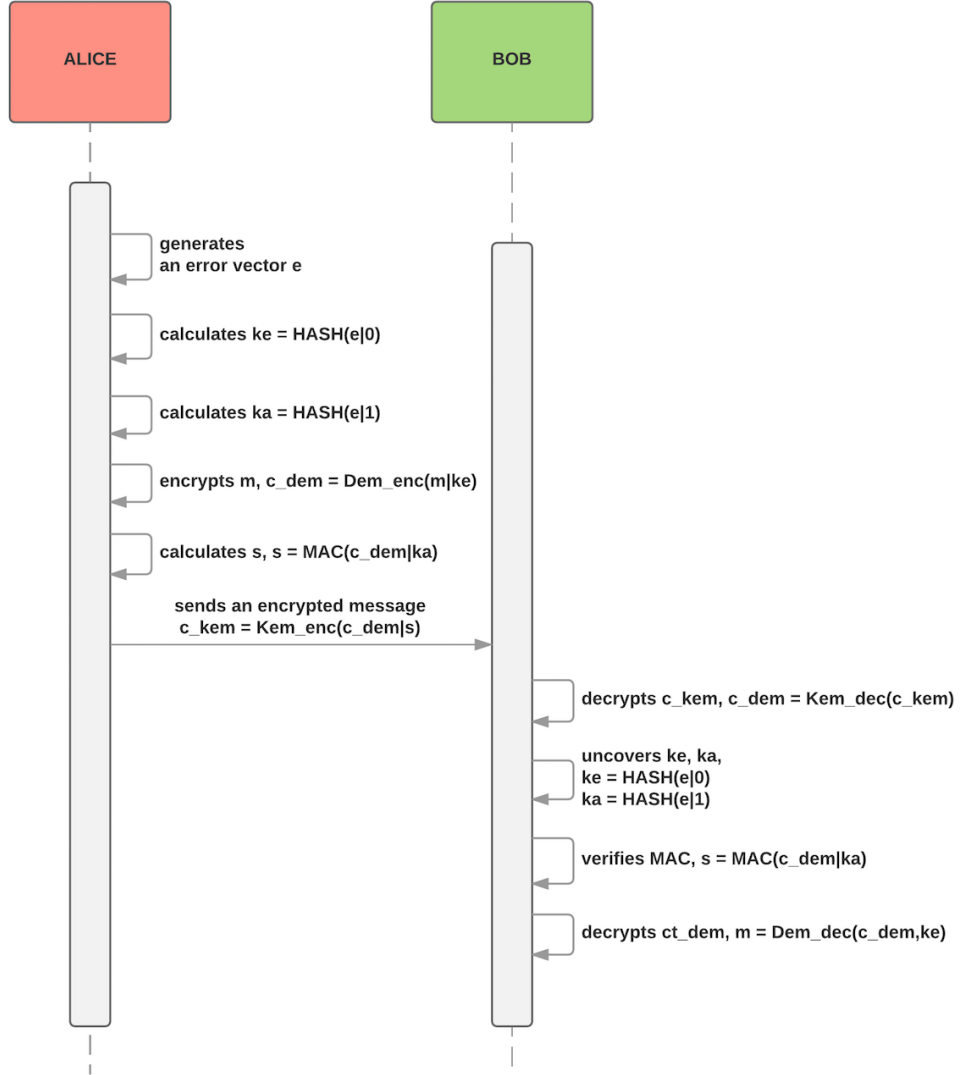


Figure 4: Hybrid scheme

In Table 1, we compare various related MECS conversions and their data redundancy. As for the redundancy, our proposal shows exactly same results as the proposal from Persichetti. The advantage of our hybrid scheme is that the key is transmitted along with the ciphertext. Therefore, our scheme reduces a number of required bits for transferring messages. For $n = 1024$ bits, $k = 644$ bits, $tag = 128$ bits, Table 2 shows the respective amount of bits for transferring messages with given lengths. In case of transferring a

Table 1: Comparison between schemes and data redundancy

Data redundancy = Ciphertext size - Plaintext size				
Scheme	(n,k, t)	(1024, 644, 38)	(2048,1289, 69)	(4096,2560, 128)
Original	n-k	380	759	1536
Pointcheval's	n + Len(r)	1184	2208	4256
Kobara-Imai's	n-k+Len(r) +Len(Const) -log ₂ C(n,t)	470	648	1040
Our propopsal	n-k+Len(tag)	508	887	1664
Persichetti's propopsal	n-k+Len(tag)	508	887	1664

message of 128 bits, both schemes shows a data overhead. In order to meet security requirements, our proposal must fulfill an entire MECS block, thus, the overhead is caused by padding. The significant part of messages transferred by Persichetti proposal represents an encrypted error vector.

Table 2: Total data transmitted

Scheme	128 bits [bits]	512 bits [bits]	1024 bits [bits]
1. Our proposal	1276	1276	1152
2. Persichetti's proposal	1280	1664	2176

3.2 Key exchange scheme supporting forward secrecy

The hybrid schemes offers the opportunity to implement various protocols utilizing MECS with arbitrary long messages. For instance, the original form of Needham-Schroeder-Lowe [22] protocol with trusted servers can be efficiently implemented, although the protocol requires certificates in order to ensure the trust. These certificates are usually distributed online and the security of the protocol strongly relies on trusted servers.

We present a generic scheme for exchanging session keys, which supports perfect forward secrecy. The scheme is inspired by already mentioned Needham-Schroeder-Lowe protocol, however we omit the trusted servers. The price for omitting server is data redundancy discussed in more detail at the end of this section. In the scheme, participants exchange a session key in order to subsequently create a symmetric secure channel, for instance by using TLS with an option called pre-shared key. The both sides of communi-

cation must actively participate.

If we omit *PKE* and *SKE*, the protocol can be very similar to the NSL protocol.

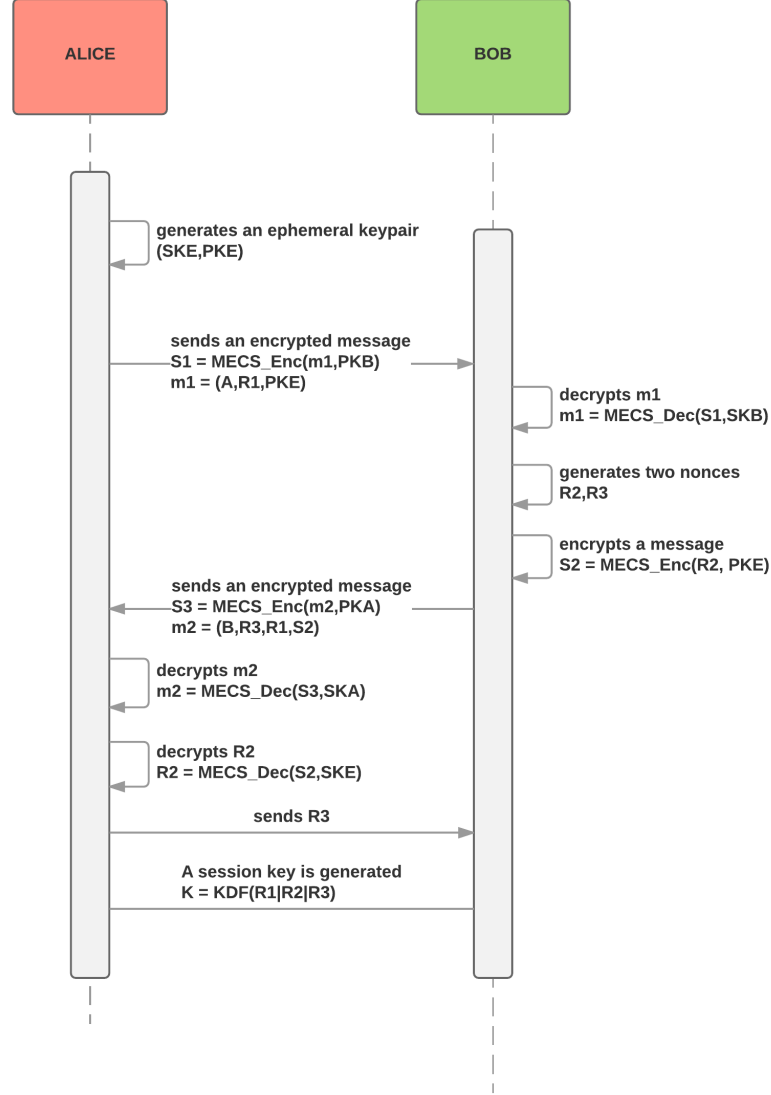


Figure 5: Key exchange scheme

We use $R1$ and $R3$ to confirm that the participants can decrypt messages. The random numbers $R2$, $R3$ are nonces tied to PKB and PKE respectively. The $R2$ is randomly looking and should be computationally infeasible to decrypt without knowledge of the SKE . Thus, after discarding SKE , our protocol ensures perfect forward secrecy.

In order to ensure perfect forward secrecy, the process of exchanging must be done in the beginning of every session. This requires additional overhead of data. For example, using our hybrid scheme for McEliece parameters [2048,1498,101] we can transfer up to

1370 bits per encryption. The public key size is 67,072 bytes, and we need to send $S1$, $S3$ and $R3$. Considering random seeds and identity strings of length 512 bits together, roughly 10 KB needs to be transferred in order to start a session.

4 Implementation of proposals

In Figure 6, we present an architecture overview of the system. Our implementation requires no major changes in the library. We developed the proposed schemes a standalone extension utilizing features from both mbedtls and Bitpunch. The central point is the CryptoBox, which provides an API for encryption and decryption, respectively. The cryptographic primitives are utilized from Utils. Both AES and PBKDF2 are wrappers around functions provided from mbed TLS. These wrappers initialize appropriate contexts and perform actions, e.g. encryption, key derivation, etc. The implemented interface is inspired by NaCl cryptographic software library written by Daniel J. Bernstein, Tanja Lange and Peter Schwabe [8].

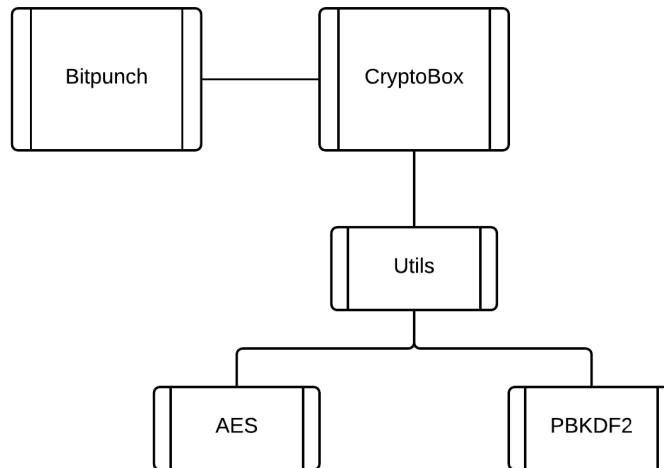


Figure 6: The class diagram

The lines of the code show basic encryption and decryption processes that can be performed using the CryptoBox:

Listing 2: Cryptobox generation of keypairs

```
#include "crypto_box.h"

BPU_T_GF2_Vector *pt_dem_a, *ct_kem;
char *pk = NULL;
int size;

BPU_cryptobox_send(ct_kem, pt_dem_a, pk, size);
```

...

`BPU_cryptobox_recieve(pt_dem, ct_kem, ctx)`

The *pk* is a buffer, which needs to be filled with a public key intended for encryption with the respective *size*. The *ctx* represents a context in which a secret key is stored.

The *BPU_cryptobox_recieve* function verifies and decrypts a ciphertext *ct_kem* using the receiver's secret key *sk*. The function puts the plaintext into *pt_dem* and subsequently returns 0, in case of a successful decryption.

4.1 Used cryptographic algorithms

This subsection covers implementations details of used cryptographic algorithms.

4.1.1 AES-GCM

Symmetric encryption is implemented using cipher module from mbed TLS. The module provides symmetric encryption and decryption in a generic way. Since there are no other connection with different modules from the library, the module can be considered standalone. The high level API [3] describes following provided functions from the Cipher module interface:

- Initialization and cleanup functions - A generic cipher context initialization and cleaning up.
- Key handling functions - Keys for encryption and decryption can be set. The mode of operation, encryption or decryption, is set with the respective key.
- Encrypting/Decrypting a message - The encryption and decryption processes can be set with a new initialization vector (IV), subsequently updated with new input data. The data are then finalized to write awaiting data to the output buffer. The calling function must provide pre-allocated input and output buffers.

AES Galois Counter Mode (GCM) Cipher Suites for TLS is defined in RFC5288 [29]. GCM must be constructed from an approved symmetric key block cipher, for instance the Advanced Encryption Standard. GCM is considered a mode of operation of the AES algorithm. Specific recommendation for Block Cipher Modes of galois counter mode can be found in NIST [13]. The following paragraphs remark the essentials from the documents.

AES-GCM provides authenticated encryption with associated data providing both confidentiality and authenticity. AES-GCM is efficient, secure and can be effectively employed for speeds of 10 gigabits per second [13]. The authenticity of the confidential

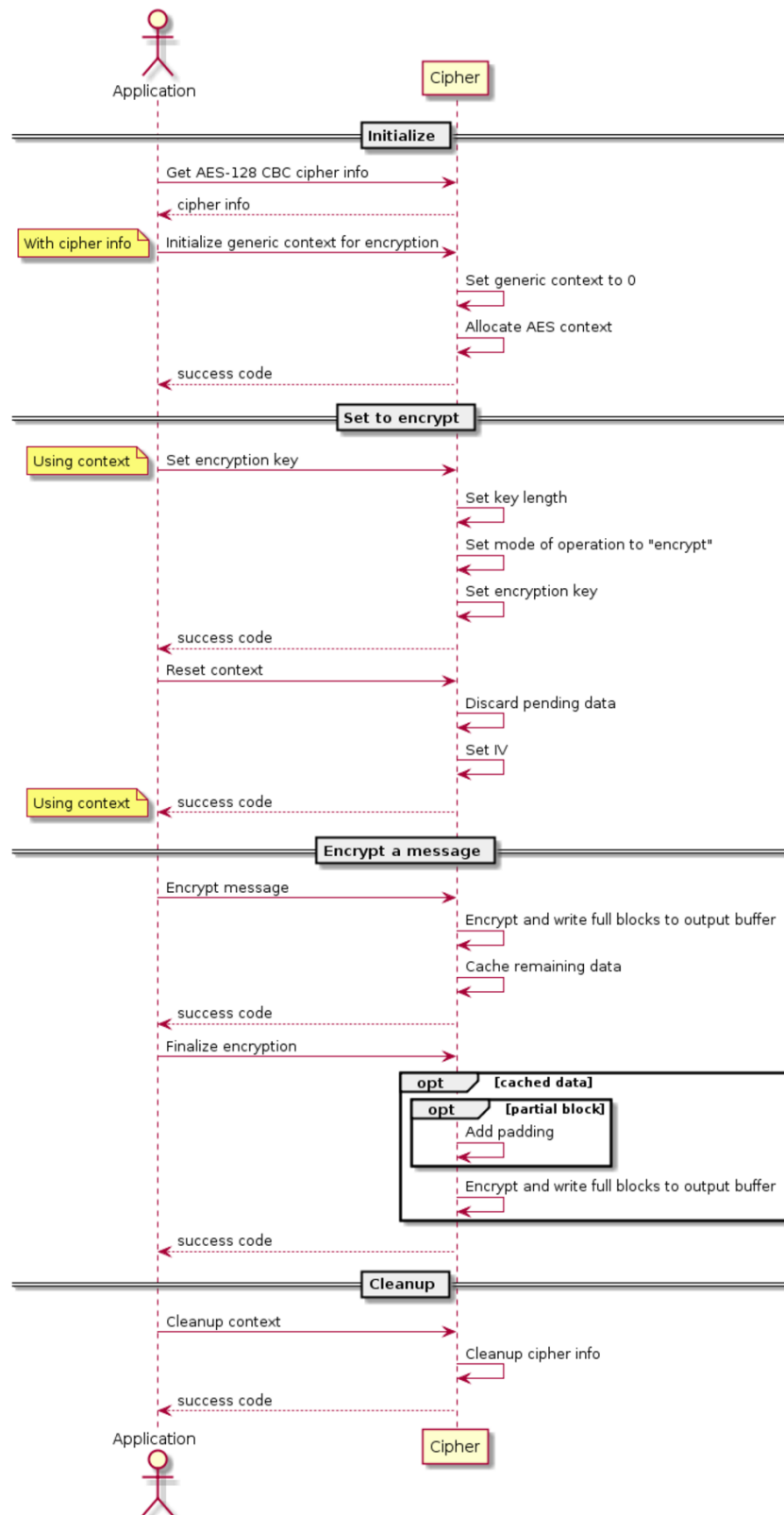


Figure 7: [28] AES encryption

data is assured using a universal hash function that is defined over a binary Galois field. GCM can also provide authentication assurance for additional non-encrypted data. For example, within a network protocol, the additional data might include addresses, protocol version numbers, and other fields that indicate how the plaintext should be treated. However, this feature is not included in our protocol.

AES-GCM requires an initialization vector. This raises a problem, since the vector should have the properties already mentioned in preliminaries. Among them, uniqueness in stream cipher encryption is extremely important. Otherwise, the first block of plaintext can be easily recovered.

NIST SP800-38a [12] recommends two methods for generating unpredictable IVs. The first method applies a cipher function, under the same key that is used for encryption of the plaintext, to a nonce. The nonce refers to a number used once, thus must be unique per each encryption. For instance, the nonce can be either a message number or a counter. The second method is to generate a random block of data using a random number generator that is approved by FIPS.

For generation of a key for encryption and an IV, we use PBKDF2, which is seeded with sufficient entropy that comes from an error vector with static salts. These salts must be declared as protocol constants.

4.1.2 PBKDF2

PBKDF2 is a commonly used key derivation function that applies a pseudorandom function. The length of the derived key is essentially arbitrary, which perfectly fits our requirements. However, the effective search space for the derived key can be limited by the underlying pseudorandom function [16].

The function *BPU_gf2VecKDF* wraps functions from mbed TLS. The part of source code in mbed TLS containing PBKDF2 is called PKCS5 v2.0 and is defined in [16]. We implemented a version with underlying SHA512 defined as *MBEDTLS_MD_SHA512*.

As for salts used in PBKDF2, we use static strings stated in the scheme above. The salts are usually incorporated in order to discourage an attacker, who wants to brute force or use rainbow tables against password hashes. Standard salt should be random, generated per each password. There is no need to keep it secret. However, in our case they act as a part of password, which is publicly known and does not provide any security improvements. The most important entropy comes from the error vector and each salt being different creates independent random oracles from the theoretical point of view.

An iteration count is used for the purpose of increasing the cost of producing keys from a password, thus increasing the difficulty of an attack. The iteration count indicates

how many times to iterate underlying function. However, in our case iteration count greater than one slows down encryption and decryption processes. For this specific case, it does not add any security improvements, too.

4.2 Hybrid scheme based on McEliece PKC

The following algorithm shows specific cryptographic algorithms used in the scheme. The requirements that need to be met are that A knows the public key of a participant B .

Algorithm 10 Hybrid scheme based on MECS

1. A generates an error vector e
 2. A generates $ke = PBKDF2(e, "PBKDF2 - SALT - ENC")$
 3. A generates $iv = PBKDF2(e, "AES - IV - SALT")$
 4. A encrypts $(c_dem, tag1) = AUTH_enc(m, ke, iv)$
 5. A encrypts the first part of a message, $c_kem = MECS_enc(c_dem|tag1, PKB, e)$
 6. B decrypts the part and uncovers e , $(c_dem, e) = MECS_dec(c_kem, SKB)$
 7. A uncovers $iv = PBKDF2(e, "AES - IV - SALT")$
 8. B uncovers $ke = PBKDF2(e, "PBKDF2 - SALT - ENC")$
 9. B decrypts $(m, tag2) = AUTH_dec(c_dem, ke, iv)$
 10. B verifies if $tag1 \neq tag2$ then return $CHECK_INTEGRITY_FAIL$, else return $SUCCESS$
-

The participant B must continue in decryption process, even in case when he can not decrypt the message using MECS properly, in order to prevent reaction attacks. Should it be the case, the system must react consistently and return an error message only after authenticated decryption.

4.2.1 Messages and padding

Several cryptographic primitives require a various length plaintext to be padded in order to fit to block size. Both cryptographic algorithms AES and MECS used in our hybrid scheme require expansion of messages with various length.

We are aware of various attacks misusing padding schemes from the history of cryp-

tography [34]. These problems must be addressed carefully. Our padding scheme is designed to give an attacker no advantages for breaking the hybrid scheme.

In the hybrid scheme, there are two possible cases when we need to extend data. The first case occurs when the length of data is less than a block size of MECS minus length of an authentication tag. In this case, we need to append padding to data in order to fulfill the entire MECS block. The data are subsequently encrypted by AES. Therefore, the text passed to MECS block contains randomly looking data. The data are also padded to fit 128 bit AES blocks.

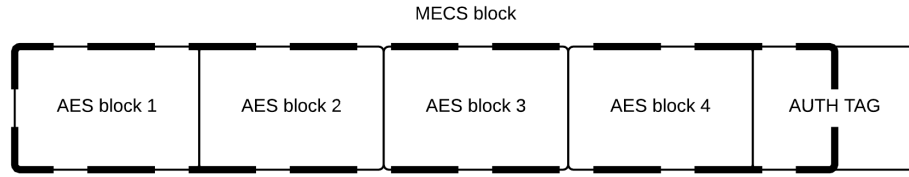


Figure 8: The padding scheme, in which a message is shorter than MECS block

It is inherently clear, that this padding scheme requires adding and encrypting a significant amount of data, when a message is very short. For example, with given parameter of Goppa code $k = 1498$ and a message that contains 128 bits, we need to transfer 1664 bits.

In the second case, when the message is longer than MECS block, the data must be padded to fit AES blocks.

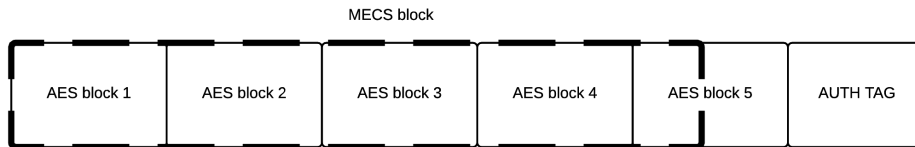


Figure 9: The padding scheme, when a message is longer than MECS block

4.3 Key exchange scheme supporting forward secrecy

The following scheme represents a concrete implementation of the key exchange scheme supporting forward secrecy inspired by the NSL protocol. In order to exchange an ephemeral key, we utilize the previous hybrid scheme. We also assume that A owns the keypair (SKA, PKA) , the pre-shared PKB and vice versa. For sake of simplicity,

the details of identities of agents A and B are omitted. In a real protocol, they must be connected to the respective public keys.

The scheme is implemented as a proof-of-concept in one of test functions with the following parameters and the respective length of messages:

- MECS: Goppa codes, $n = 2048$ bits, $k = 1498$ bits
- $R1, R2, R3 = 256$ bits
- $S1 = 104486$ bits, $m1 = 103686$ bits
- $S2 = 2086$ bits, $m2 = 2598$ bits
- $S3 = 3366$ bits

Algorithm 11 Key exchange scheme

1. A generates an ephemeral keypair SKE, PKE
 2. A sends $S1 := Hyb_Enc(m1, PKB)$ to B , where $m1 := (A, R1, PKE)$
 3. B decrypts $m1 := Hyb_Dec(S1, SKB)$
 4. B generates two nonces $R2, R3$
 5. B computes $S2 = Hyb_Enc(R2, PKE)$
 6. B sends $S3 := Hyb_Enc(m2, PKA)$ to A , where $m2 := (B, R3, R1, S2)$
 7. A decrypts $m2 := Hyb_Dec(S3, SKA)$
 8. A decrypts $R2 := Hyb_Dec(S2, SKE)$
 9. A sends $R3$ to B
 10. A session key is derived, $K = PBKDF2(R1|R2|R3, "SALT")$
-

5 Tests

In order to measure speed, size and memory requirements of our changed fork of the library, we use the following configuration.

CPU 2,2 GHz Intel Core i7

RAM 16 GB 1600 MHz DDR3

OS Mac OS X El Capitan, Darwin Kernel Version 15.3.0

Parameters of the library $m = 11$, $t = 50$, Goppa code

The following table 3 shows a performance test, where the proposed hybrid scheme is compared with NaCl [8]. NaCl provides a hybrid scheme utilizing elliptic curves and Salsa20. We transferred statically initialized 52 bytes as a message. The table also contains performance results of the basic McEliece from the Bitpunch project.

Table 3: Performance test

Scheme	KeyGen [ms]	Enc [μ s]	Dec [ms]
1. Our proposal	780	547	11114
2. Bitpunch	760	177	10759
3. NaCl	0.175	16	10

It is inherently clear that our proposal shows slightly worse results in comparison with the Bitpunch due to the involvement of AES-GCM. However, highly optimized NaCl shows much better results. It is caused by its using different underlying cryptographic primitives. Table 4 shows times that both solutions need to encrypt messages with selected lengths. Considering the aforementioned reason, it could be anticipated that an encryption with NaCl takes less time in comparison with our solution. The time complexity grows in linear fashion with the length of messages.

Table 4: Performance test of encryption

Scheme	1MB [ms]	10MB [ms]	100MB [ms]
1. Our proposal	417	4174	42663
2. NaCL	4	47	518

Table 5: Memory footprint

Scheme	Memory usage [MB]
1. Our proposal	7.3
2. NaCl	9.5
3. Bitpunch v0.0.3 w. H	0.3

We measured memory footprint of our proposal as shown in Table 5. Our proposal and NaCl allocate approximately 10 MB of memory. There is a significant gap between the full hybrid scheme application, and the basic version of Bitpunch. This is natural, as we need extra memory for algorithms and data structures on top of the core McEliece routines. Due to this fact our solution is not very well suited for embedded devices. Since the decryption of our proposal takes much more time than the encryption, we recommend using our proposal in environments, where sessions are initialized by a server.

Figure 10 and Figure 11 show a time-complexity during operations performed within the key exchange scheme. As can be seen from Figure 10, the most time-consuming operations are initializations of contexts and a generation of an ephemeral key.

Figure 11 shows a time-complexity of operations except initializations of contexts and key generation. It is noticeable that decryption processes of messages together with the first encryption constitute a major proportion of the chart. It could be easily anticipated from the previous results of our hybrid scheme, where decryption processes are significantly slower than encryption operations.

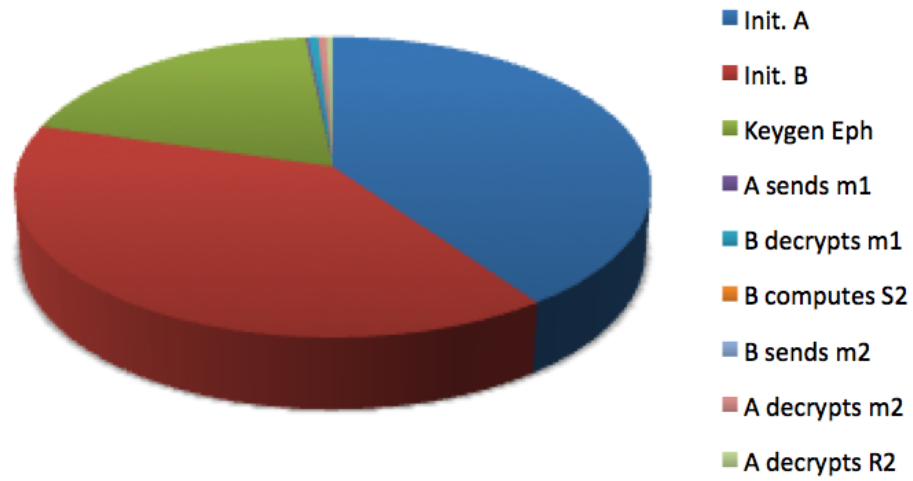


Figure 10: Time complexity of operations

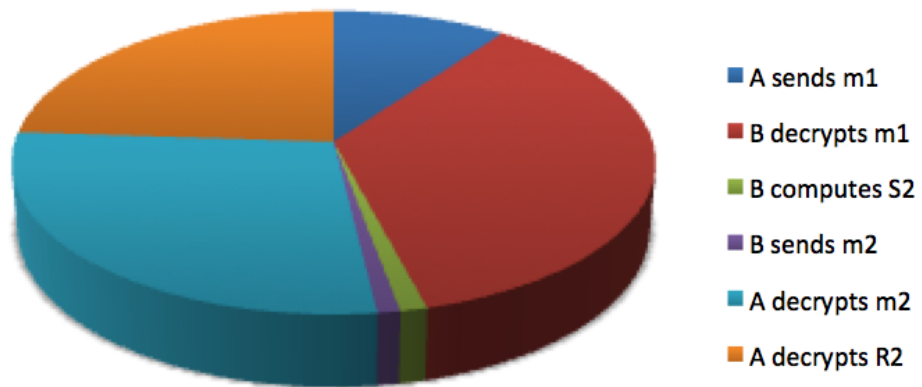


Figure 11: Time complexity of operations except initializations and a key generation

Conclusion

The thesis focuses on implementation problems regarding post-quantum cryptography. We designed and implemented a hybrid encryption scheme utilizing Bitpunch and mbed TLS libraries. The scheme should be resistant against various advanced attacks. We proposed and implemented a proof-of-concept of a key exchange scheme supporting perfect forward secrecy.

The hybrid encryption scheme can be 'competitive' even in comparison with cryptobox functions implemented in NaCl. However, decryption is more resource demanding in terms of computation. Therefore, we recommend using our proposal in environments, where sessions are initialized by a server. This would delegate the more difficult task to client machines and improve server throughput. Nevertheless, if a server possesses better computational resources, sessions can be initialized by clients (e.g. in a scenario with mobile clients).

The key exchange scheme suffers from a significant data redundancy. The problem might be solved by using QC-MDPC codes instead of Goppa codes.

Beside the aforementioned proposals, we fixed two memory corruptions within the Bitpunch library. We discovered an integer overflow that had lead to a buffer overflow and improper writing to a memory for a function that provides padding. Our fixes are integrated into the core library.

Our tests have shown that the implemented solution can be used in practice. For hybrid scheme, a further optimization can be obtained by adapting a faster stream cipher (such as Salsa used in crypto-box). For the key exchange protocol, we recommend using QC-MDPC codes, as the Goppa-code based version has unacceptably large data redundancy for most of the practical applications.

Resumé

Diplomová práca sa zaoberá problémami, ktoré vyplývajú z rýchlo napredujúceho výskumu v oblasti kvantových počítačov. Predpokladá sa, že väčšina z dnešných populárnych algoritmov na šifrovanie nebude v budúcnosti poskytovať dostatočnú ochranu. Algoritmus, ktorý by umožnil rýchlu faktorizáciu bol predstavený už v roku 1997. Preto je nevyhnutné nájsť a implementovať alternatívne algoritmy ako náhradu za tie nedostačujúce. Post-quantová kryptografia je pojem, ktorým označujeme množinu kryptografických algoritmov (zvyčajne asymetrické), ktoré by mali byť odolné voči útočníkom s prístupom ku kvantovému počítaču. Medzi algoritmy pri ktorých sa stále verí, že dokážu odolať spomínanému útoku patria aj zástupcovia zo súčasnej symetrickej kryptografie (napríklad AES). Hlavným cieľom diplomovej práce je navrhnúť, implementovať a otestovať hybridnú schému a schému na výmenu kľúča, využívajúc post-quantové algoritmy.

Jedným z najlepších kandidátov pre post-quantovú kryptografiu je systém, založený na dekódovaní problému, ktorý bol predstavený R.J. McEliece v roku 1978. McEliecov systém patrí k zástupcom asymetrickej kryptografie založenej na verejnom a súkromnom kľúč. Viacero výskumníkov predstavilo kódy, na ktorých môže byť tento kryptosystém založený avšak časom sa ukázalo, že nie všetky z nich poskytujú dostatočnú úroveň bezpečnosti.

Dekódovací problém, na ktorom sa zakladá bezpečnosť kryptosystému je NP úplný. Avšak pôvodná verzia McEliecovho systému je zraniteľná voči pokročilým útokom (napríklad vhodnou zmenou šifrovaného textu dokážeme zmysluplne zmeniť príslušný nešifrovaný text), čiže systém nie je kryptograficky ani sémanticky bezpečný. Medzi pokročilé útoky patrí aj prípad, pri ktorom má útočník prístup k dešifrovaciemu orákulu a sleduje závislosti medzi nezašifrovaným a zašifrovaným textom. Tento útok môže byť eliminovaný použitím CCA-bezpečnej schémy respektíve hybridnej schémy.

Hybridné schémy sú kryptografické protokoly umožňujúce využívať výhody asymetrickej aj symetrickej kryptografie. Asymetrická kryptografia je použitá na výmenu kľúča, ktorý sa neskôr používa ako vstup pre algoritmus zo symetrickej kryptografie. Hybridné schémy sa skladajú z dvoch modulov KEM a DEM. KEM modul, reprezentuje asymetrický algoritmus a poskytuje operácie na vytvorenie kľúčov, šifrovanie a dešifrovanie. Pričom DEM modul reprezentuje symetrický algoritmus a poskytuje len operácie na šifrovanie a dešifrovanie správ. Naš návrh hybridnej schémy však nie je možné takto striktne rozdeliť a tak by sa dal označiť ako všeobecná asymetrická šifrovacia schéma s ľubovoľne dlhými správami.

V práci sa zaoberáme aj schémou, ktorá slúži na výmenu kľúča. Needham a Schroeder predstavili protokol, v ktorom si dve strany navzájom vymenia kľúč. V protokole sa využíva dôveryhodný server, ktorý overuje identity oboch strán. O niekoľko rokov Lowe objavil chybu, ktorá umožňuje takzvaný MitM útok a následne navrhol protokol, v ktorom sa táto chyba už nevyskytuje. Tento protokol nemá predpoklady na použitie špecifickej asymetrickej šifry, rozhodli sme sa ho teda adaptovať v kombinácii s McEliecovým kryptosystémom.

V úvodnej časti práce uvádzame potrebné kryptografické primitívy ako napríklad hašovacia funkcia, funkcia na odvádzanie hesla, inicializačný vektor, atď. Analytická časť práce popisuje implementačné detaily využívaných knižníc Bitpunch a mbed TLS. Bitpunch je knižnica vytvorená v rámci rovnomenného projektu zo Slovenskej Technickej Univerzity. Knižnica implementuje McEliecov kryptosystém v originálnej podobe s Goppovými kódmi, ktoré boli použité aj v našej práci. Knižnica je modulárna a umožňuje jednoduché rozšírenie avšak v záujme zachovania veľkosti knižnice, je naše riešenie realizované ako samostatný projekt, ktorý Bitpunch používa ako zdieľanú knižnicu.

Bitpunch pracuje s informáciami (napríklad kľúče), ktoré by mali byť v štandardnom formáte zachovávajúc interoperabilitu medzi komunikujúcimi stranami. Knižnica využíva štandard ASN.1. Tento štandard okrem iného popisuje ako by mali byť kľúče serializované tak aby sa dali na druhej strane správne deserializovať. Štandard využívame aj v našom riešení napríklad pri šifrovaní, kde odosielateľ posiela do šifrovacej funkcie verejný kľúč v špecifikovanom formáte.

Knižnica nazvaná mbed TLS (v minulosti PolarSSL) poskytuje nášmu riešeniu potrebné algoritmy ako napríklad AES-GCM a PBKDF2. Vzhľadom na to, že knižnica je distribuovaná pod GPL2 licenciou, je možné ju voľne použiť pre nekomerčné projekty. Okrem toho je táto knižnica voľne prenositeľná na takmer všetky operačné systémy, vzhľadom na to, že je implementovaná v nízko-úrovňovom jazyku C.

V práci analyzujeme špecifickú CCA-bezpečnú schému, ktorú navrhli Kobara a Imai. Táto schéma môže byť taktiež braná ako hybridná schéma nakoľko využíva hašovacie funkcie na odstránenie závislostí medzi šifrovaným a nešifrovaným textom. Schéma využíva konverznú funkciu, ktorá sa skladá z viacerých výpočtových krokov. Je preto prirodzené, že funkcia je pomalšia ako priamočiare generovanie náhodného vektora, ktoré je využité v našej schéme.

Podľa našich aktuálnych znalostí, jediná hybridná schéma založená na dekodovaní problému, pri ktorej sa podarilo dokázať, že je CCA-bezpečná bola navrhnutá Edvardom Persichettim. Tento model je založený na Niederreiterovom kryptosystéme, ktorý

je variantou McElievovho kryptosystému. Model využíva chybový vektor ako vstup do funkcie na derivovanie kľúčov, ktorý je následne použitý pri šifrovaní. Podobný trik využijeme aj v nami navrhnuťej schéme.

Hlavným prínosom nami navrhnuťej hybridnej schémy je redukcia dátovej redundancie, využívajúc špecifické vlastnosti McElievovho kryptosystému. V práci neposkytujeme dôkaz o bezpečnosti schémy avšak uvádzame bezpečnostné vlastnosti, z ktorých vychádzame. Chybový vektor je náhodne generovaný z veľkého priestoru možných vektorov, ktorý je väčší ako priestor hašovacej funkcie. S ohľadom na tento fakt výstup funkcie na derivovanie kľúčov, ktorý používa hašovaciu funkciu môžeme taktiež považovať za náhodný.

Po autentizovanom šifrovaní by vstup do McElievovho kryptosystému mal vyzeráť neodlíšiteľný od náhodného reťazca. Autentizované šifrovanie je zabezpečené pomocou AES-GCM, čo je špecifický mód štandardu na šifrovanie, ktorý okrem dôvernosti zabezpečuje aj autentickosť údajov. Obvyklé zabezpečenie oboch požiadaviek je riešené samostatným šifrovaním a samostatným výpočtom hodnoty autentizačného kódu, napríklad pomocou HMAC. Rozhodli sme sa pre AES-GCM z dôvodu, že autentizované šifrovanie spája obe operácie do jednej, čím obvykle dosahuje vyššiu rýchlosť.

McElievov kryptosystém je použitý na prenesenie prvej časti správy a materiálu potrebného na generovanie kľúča pre AES-GCM. Ostatná časť správy, ktorá je väčšia ako McElievov blok a bola zašifrovaná pomocou AES-GCM môže byť priamo poslaná prijímateľovi. Aby útočník dokázal kompromitovať komunikáciu musí získať pôvodný chybový vektor, čo by znamenalo, že sa mu podarilo narušiť bezpečnosť McElievovho kryptosystému. Manipulácia so zašifrovaným textom je detegovateľná pomocou AUTH tagu. Schéma konzistentne reaguje aj na reakčné útoky. V prípade, že útočník podhodí systému vstup, ktorý nedokáže dešifrovať pomocou McElievovho kryptosystému, proces pokračuje ďalej s náhodným chybovým vektorom. Chybový vektor musí byť zakódovaný ako reťazec konštantnej dĺžky aby sme predišli časovacím útokom. V práci porovnávame dátovú redundanciu voči ostatným analyzovaným schémam.

Ďalšia navrhovaná schéma určená na výmenu kľúča podporuje takzvanú 'forward secrecy'. Takáto schéma znemožňuje útočníkovi odvodenie kľúčov použitých v komunikácii aj v tom prípade ak hlavný kľúč prípadne kľúče boli kompromitované. V tomto protokole využívame predchádzajúcu schému na prenos potrebných informácií. Vzhľadom na to, že v našom koncepte nepoužívame dôveryhodný server, musíme medzi účastníkmi komunikácie preniesť dočasný verejný kľúč, čo spôsobuje značné nároky na množstvo prenesených dát. Tak isto počítame s tým, že verejné kľúče pre oboch účastníkov sú už dopredu známe.

Po vytvorení komunikačného kanálu a odhodení dočasného kľúča schéma podporuje 'forward secrecy'. Avšak spomínaný dočasný kľúč sa musí prenášať pred začiatkom každej komunikácie a teda len na začiatok komunikácie musíme preniesť zhruba 10KB dát.

V práci poukazujeme aj na konkrétne implementačné detaily riešenia, akým sú napríklad základné volania pre šifrovanie a dešifrovanie. Rozhrania boli inšpirované knižnicou NaCl vyvinutou D. Bernsteinom, T. Lange a P. Schwabom. NaCl implementuje hybridnú schému využívajúc kryptografické algoritmy ECC a Salsa20. Ďalej popisujeme detaily použitia AES-GCM a PBKDF2.

Štandardná implementácia PBKDF2 si vyžaduje soľ a počet iterácií. Ako výstup tejto funkcie očakávame inicializačný vektor a kľúč, ktorý je použitý na autentizované šifrovanie a dešifrovanie. Vzhľadom na to, že vstupom do PBKDF2 je v oboch prípadoch chybový vektor, rovnaká soľ pre inicializačný vektor a kľúč nemohla byť použitá. Dve rôzne orákulá dosahujeme pomocou statických solí, ktoré sú uvedené ako protokolové konštanty. Taktiež sme nastavili minimálny možný počet iterácií, keďže by v našom prípade len spomaľovali operácie a nepridali žiadne bezpečnostné vylepšenia.

V nasledujúcej časti sme zabráňovali útokom, ktoré zneužívajú slabiny padding schém. Padding schémy sa používajú na expandovanie správ tak aby veľkosťou sedeli do blokov šifry. Aby sme sa vyhli problémom, pri ktorých by mohol byť odhalený chybový vektor, rozhodli sme sa expandovať správu ešte predtým ako je šifrovaná pomocou AES-GCM. V prípade, že je správa kratšia ako dĺžka jedného bloku McEliecevho kryptosystému expandujeme správu tak aby vyplnila celý blok a zároveň dĺžka správy musí byť násobkom veľkosti bloku AES-GCM. Takéto riešenie v praxi znamená, že pri veľmi krátkych správach vzniká dátová redundancia. V prípade, že je správa dlhšia expandujeme len na dĺžku násobku AES-GCM. Nakoniec do správy pripojíme autentizačný odtlačok.

Počas testovania sme preverovali pamäťovú a výpočtovú zložitosť. Výsledky testovania poukazujú na to, že naše riešenie je o trochu pomalšie ako originálny Bitpunch a tak sa použitie mbed TLS ukázalo ako efektívne. V porovnaní s NaCl, rýchlosť dešifrovania nie je našou devízou a teda riešenie je výpočtovo náročnejšie na strane prijímateľa. Odporúčame preto, ak je možné používať toto riešenie v prostredí, kde sú spojenia inicializované serverom. Výpočtová náročnosť rastie lineárne s nárastom dĺžky správ.

Okrem spomenutých návrhov sa nám podarilo identifikovať a opraviť dve závažné zraniteľnosti v Bitpunchi, pri ktorých bol útočník schopný ľubovoľne prepisovať miesto v pamäti, čo by mohlo spôsobiť úplnú kompromitáciu. Opravy už boli pridané do hlavnej vetvy projektu Bitpunch. Riešenie má potenciál využitia v praxi a môže byť nasadené v reálnych systémoch.

Bibliography

- [1] BS ISO/IEC 9797 2:2011. <https://www.iso.org/obp/ui/#iso:std:51618:en>. Accessed: 2016-05-01.
- [2] ADAMS, C., AND MEIJER, H. Security related comments regarding mceliece public key cryptosystem. *Advances in Cryptology CRYPTO 87* (2000), 224–228.
- [3] BAKKER, P. Symmetric cipher (cipher) module level design. <https://tls.mbed.org/module-level-design-cipher>. Accessed: 2016-05-01.
- [4] BAKKER, P. Github repository for mbedtls, 2016. <https://github.com/ARMmbed/mbedtls>, (01.05.2016).
- [5] BAKKER, P. Security center for mbedtls, 2016. <https://tls.mbed.org/security>, (01.05.2016).
- [6] BARKER, E. B., BARKER, W. C., BURR, W. E., POLK, W. T., AND SMID, M. E. Sp 800-57. recommendation for key management, part 1: General (revised). Tech. rep., Gaithersburg, MD, United States, 2007.
- [7] BERLEKAMP, E., MCELIECE, R., AND VAN TILBORG, H. On the inherent intractability of certain coding problems (corresp.). *Information Theory, IEEE Transactions on* 24, 3 (May 1978), 384–386.
- [8] BERNSTEIN, D. J., LANGE, T., AND SCHWABE, P. Nacl: Networking and cryptography library. <https://nacl.cr.yp.to/>. Accessed: 2016-05-01.
- [9] BERSON, T. Failure of the mceliece public key cryptosystem under message resend and related message attack. *Advances in Cryptology CRYPTO 97* (2006), 213–220.
- [10] CRAMER, R., AND SHOUP, V. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM J. Comput.* 33, 1 (Jan. 2004), 167–226.
- [11] DANG, Q. Changes in federal information processing standard fips 180-4, secure hash standard. *Cryptologia* 37, 1 (Jan. 2013), 69–73.
- [12] DWORKIN, M. J. Sp 800-38a 2001 edition. Recommendation for block cipher modes of operation: Methods and techniques. Tech. rep., Gaithersburg, MD, United States, 2001.

- [13] DWORKIN, M. J. Sp 800-38d. Recommendation for block cipher modes of operation: Galois/counter mode (gcm) and gmac. Tech. rep., Gaithersburg, MD, United States, 2007.
- [14] DWORKIN, M. J. Sp 800-38a addendum. Recommendation for block cipher modes of operation: Three variants of ciphertext stealing for cbc mode. Tech. rep., Gaithersburg, MD, United States, 2010.
- [15] IEEE. Standard specifications for public-key cryptography. *IEEE Std 1363-2000* (Aug 2000), 1–228.
- [16] KALISKI, B. Password-based cryptography specification version 2.0. <http://www.rfc-editor.org/info/rfc2898>. Accessed: 2016-05-01.
- [17] Katz, Jonathan and Lindell, Yehuda. *Introduction to Modern Cryptography (Chapman & Hall/Crc Cryptography and Network Security Series)*. Chapman & Hall/CRC. 100-102 (2007).
- [18] KOBARA, K., AND IMAI, H. Semantically secure mceliece public-key cryptosystems -conversions for mceliece pkc -. In *Public Key Cryptography*, K. Kim, Ed., vol. 1992 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2001, pp. 19–35.
- [19] Koller, Daphne and Friedman, Nir. *Probabilistic Graphical Models: Principles and Techniques, Adaptive Computation and Machine Learning*. The MIT Press.
- [20] LANDAIS, G., AND TILlich, J.-P. An efficient attack of a mceliece cryptosystem variant based on convolutional codes. In *Post-Quantum Cryptography*, P. Gaborit, Ed., vol. 7932 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2013, pp. 102–117.
- [21] LEGION OF THE BOUNCY CASTLE INC. About the legion of the bouncy castle, 2013.
<https://www.bouncycastle.org/>, (01.05.2016).
- [22] LOWE, G. An attack on the needham-schroeder public-key authentication protocol. *Information Processing Letters* 56, 3 (1995), 131 – 133.
- [23] MCELIECE, R. J. A public-key cryptosystem based on algebraic coding theory. *DSN progress report* 42, 44 (1978), 114–116.
- [24] NIEDERREITER, H. Knapsack-type cryptosystems and algebraic coding theory.

- [25] NIST. SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. FIPS Publication 202, National Institute of Standards and Technology, U.S. Department of Commerce, May 2014.
- [26] PATTERSON, N. The algebraic decoding of goppa codes. *IEEE Trans. Inf. Theor.* 21, 2 (Sept. 2006), 203–207.
- [27] PERSICHETTI, E. Secure and anonymous hybrid encryption from coding theory. In *PQCrypto* (2013), P. Gaborit, Ed., vol. 7932 of *Lecture Notes in Computer Science*, Springer, pp. 174–187.
- [28] REPKA, M., AND ZAJAC, P. Overview of the McEliece cryptosystem and its security. *Tatra Mountains Mathematical Publications* 60, 1 (2014), 57–83.
- [29] SALOWEY, A., J. C., AND MCGREW, D. Aes galois counter mode (gcm) cipher suites for tls. <http://www.rfc-editor.org/info/rfc5288>. Accessed: 2016-05-01.
- [30] SHOR, P. W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *Society for Industrial and Applied Mathematics* (1997), 1484–1509.
- [31] SHOUFAN, A., WINK, T., MOLTER, G., HUSS, S., AND STRENTZKE, F. A novel processor architecture for McEliece cryptosystem and FPGA platforms. In *Application-specific Systems, Architectures and Processors, 2009. ASAP 2009. 20th IEEE International Conference on* (2009), IEEE, pp. 98–105.
- [32] STRENTZKE, F. Efficiency and implementation security of code based cryptosystems. Fachbereich Informatik der Technischen Universität Darmstadt (2013).
- [33] UHRECKÝ, F. Implementácia kryptografickej knižnice s McEliece kryptosystémom. Master’s thesis, 2015.
- [34] VAUDENAY, S. *Advances in Cryptology — EUROCRYPT 2002: International Conference on the Theory and Applications of Cryptographic Techniques Amsterdam, The Netherlands, April 28 – May 2, 2002 Proceedings*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002, ch. Security Flaws Induced by CBC Padding — Applications to SSL, IPSEC, WTLS..., pp. 534–545.
- [35] WANG, Y. New Way to Construct Cryptographic Hash Function. <https://eprint.iacr.org/2014/122.pdf> (Accessed: 2016-05-01).

- [36] ZAJAC, P. A note on CCA2-protected McEliece Cryptosystem with a systematic public key. <https://eprint.iacr.org/2014/651.pdf> (Accessed: 2016-05-01).

6 Structure of attached medium

\

\Diplomova_praca.pdf

\mbed_TLS

\Bitpunch

\CryptoBox