Known-plaintext attack on AES, using the HHL algorithm and the Macaulay approach

Stolwijk, Y.M.

Award date:
2022

Known-plaintext attack on AES, using the HHL algorithm and the Macaulay approach

Graduation project

2021-2022

Yanelle Stolwijk 0899639

Supervisors TU/e - Mathematics and Computer Science:

Benne de Weger — Coding Theory and Cryptology
Boris Škorić — Security

July 15, 2022

Abstract

This report investigates the ins and outs of applying HHL to a known plaintext attack on AES, using the Macaulay matrix approach. The goal is on the one hand to raise awareness for the HHL algorithm and the fact that the applications might be further reaching than expected. Hopefully this will show the urgency for more research into HHL, before cryptosystems, that are considered quantum safe, are assumed not to be threatened by HHL. On the other hand, this report gives better insights in the complexity of the HHL algorithm in combination with the Macaulay approach and shows that certain bounds from the literature might be misleading. The report shows that it is doubtful whether HHL in combination with the Macaulay approach can be used to construct an attack on AES that is significantly more efficient than Grover search. Even though this report investigates a certain strategy (the Macaulay approach) and it can not be concluded that this leads to a significant speed-up, it does not mean that such a speed-up is not possible.

Contents

1 Introduction

The past decades have witnessed an immense development of digital connectivity. This goes hand in hand with a growing dependency on online communication (e.g. online banking) and connected devices in the IoT (e.g. medical devices and vehicles), which play a vital role in our daily life. Therefore, digital security is paramount. Cryptographic primitives are used to achieve security goals. Commonly used examples are: the Advanced Encryption Standard (AES) for confidentiality and sometimes integrity, elliptic curve cryptosystems (ECC) for integrity and authentication, and the Elliptic Curve Diffie-Hellman (ECDH) algorithm for exchanging keys used in, for example, AES.

However, cryptographic schemes are threatened by the recent advances in quantum computing. Post-quantum cryptography is cryptography under the assumption that the attacker has a large quantum computer. In this scenario, many commonly used cryptosystems will be completely broken; post-quantum cryptosystems strive to remain secure.

Even though it is not clear when large scale quantum computers will come into existence, it is important to start working on post-quantum security now. It is believed that only significant engineering obstacles need to be overcome, which might happen in the near future. An increase is expected in research in post-quantum cryptography, stimulated by NIST's competition, resulting in more designs, more optimizations and implementations, and also more attacks. Besides, attackers can record long-term confidential documents and state secrets, that have to stay secure for many years, and break the encryption when quantum computers become available. Therefore, it is a race against time to deploy post-quantum cryptography.

To determine if a cryptographic primitive, and therefore the cryptographic schemes based on this primitive, is quantum secure, it needs to be investigated how an adversary could attack the primitive by quantum cryptanalysis. This research should consist of finding new quantum algorithms and of investigating how known and new algorithms can be used in an attack. Even though quantum computers are really good at solving some problems, they can only handle a limited selection of problems. For instance, efficient algorithms for factoring are known, meaning the end of the security of RSA. However, at this moment, no efficient (meaning more efficient than for example Grover's search or lattice sieving) attacks are known on multivariate cryptography (like UOV), lattice-based crypto (like NTRU) or AES. Be that as it may, there is an algorithm that can solve a system of linear equations efficiently and AES can be written as a system of linear equations.

Rijndael is a cryptographic algorithm created by Joan Daemen and Vincent Rijmen. In November 2001, the NIST accepted it as the new standard, replacing DES and renaming it AES (advanced encryption standard). Now it is the most popular and widely used block cipher. It is expected that the rise of quantum computers does not yield a serious problem for the security of AES(-256), in contrast to many other cryptosystems. However, in 2008, Harrow et al. published a paper on the HHL algorithm; an algorithm for solving linear systems of equations. Since AES can be written as such a system, it is very surprising that the research on HHL is still very scarce.

The normal definition of 'broken' is that a better attack than the generic ones (like for instance, exhaustive key search) exists. In post-quantum, attacks should be compared to its best generic quantum attack (i.e. Grover's search). Therefore, this report compares an attack using HHL to an attack using Grover. This report will extend this research to reinforce the trust in AES as a post-quantum cryptosystem, yet shows that security proofs are not within reach, by answering the following research question:

'Does HHL facilitate an attack on AES,
that is more efficient than an attack using Grover?'

1.1 Related work and contributions

The basic notions of quantum computing are collected from [9]. More details about phase estimation are added from [16]. The AES and the Mini-AES algorithm in this report are copied from [1] and [7] respectively. For the S-box equations, [5], [4], [8], and [14] are consulted.

The analysis of the HHL algorithm in this report is based on [15] and [10]. However, at some points this report concludes the papers to be erroneous. This means that the first contribution of this report is highlighting some mistakes in the details of these papers. Besides, this report improves some results that can be found in the literature.

Reference [2] gives a system of equations that needs to be solved to conduct a known plaintext attack on AES. However, some details were missing. Therefore, the second contribution of this report is to fill in those gaps.

The same paper also states methods to translate equations from GF(2) to $\mathbb{C}$ and to make the equations sparser. This report contributes by discussing another method for both, of which the method to make the equations sparser is way more efficient for the attack in this report.

Furthermore, this paper explains how to contruct a Macaulay matrix. However, [2] makes two decisions in this process, without explaining why these decisions are made, without even specifying that decisions were made. This report investigates all options and shows that the Macaulay matrix based on [2] is probably not the most efficient one. Besides, this report shows that there are three strategies to go from a system of equations over GF(2) to a Macaulay matrix, whereas, [2] makes it seem like there is only one strategy, which is not the most efficient one.

Finally, [11] tries to lower bound the complexity of the HHL algorithm, using an upper bound on D (where D is the upper bound on the degree of the monomials in the Macaulay matrix), given by [2]. This report discusses two problems with that. A lower bound on D is used to construct an upper-bound on the complexity, meaning that it assumes that the system of equations derived from AES is a worst-case non-linear system, even though the literature does not make claims about this. Besides, this report shows that [2] does not use the most efficient Macaulay matrix nor the most efficient strategy, as described above. This means that the upper-bound on D could be way too loose.

1.2 Structure of the report

The goal of this report is to describe a known plaintext attack on AES, using HHL, and to investigae the complexity of such an attack. In order to do so, Section 3 explains how the HHL algorithm works, what input is needed, what output is produced, and what the complexity is. To make this section understandable, also for readers with limited knowledge of quantum computing, Section 2 gives all the required background.

As the goal of this report is to describe a known plaintext attack on AES, the AES algorithm is explained in detail in Section 4. This results in a system of equations. Since the number of variables and equations are quite large for AES, Mini-AES is explained in the same section. Section 5 gives the system of equations for Mini-AES, after which Section 6 investigates the size of the problem. From this section it can be concluded that three steps need to be taken to conduct a known plaintext attack. Section 7 elaborates two of these steps, using some simple examples, and Section 8 explains the remaining step: creating a Macaulay matrix. All of this is needed for Section 9, to construct the input for the HHL algorithm. After evaluating this known plaintext attack on Mini-AES, the report uses the same structure for the attack on AES.

Section 10 describes the differences between the system of equations for Mini-AES, and the one for AES. Then Section 11 evaluates the size of this problem. Finally, Section 12 explains what it

means to conduct a known-plaintext attack on AES using HHL and compares this with an attack using Grover search. Based on this, the report ends with a discussion and a conclusion in Section 13 and 14 respectively.

2 Basic notions of quantum computing

Section 3 assumes knowledge about quantum computing, that is summarized in this section. First the main idea of quantum computing is explained, followed by the two operations on a state (measurements and unitary operations). Then the concept of entangled states is described and some important gates are defined. The section ends with an explanation of the phase estimation algorithm. The HHL algorithm uses a subalgorithm that is similar to the well-known phase estimation algorithm. Therefore, this section explains the phase estimation algorithm and Section 3 explains how HHL uses phase estimation as a subroutine.

The main difference between classical computing and quantum computing is the notion of qubits. Whereas classical computing works with bits, quantum computing works with qubits. A qubit 'lives' in Hilbert space $\mathbb{C}^2$. A Hilbert space is a vector space in which an inner product is defined. Note that $\mathbb{C}^2$ can be spanned by two orthogonal vectors, i.e. basis states. There are multiple options for these basis states. When the two most convenient basis states are used, the basis is referred to as the computational basis.

For the concept of a computational basis might cause confusion, this report always uses the z-basis states: $|0\rangle$ and $|1\rangle$, corresponding to $(0,1),(1,0) \in \mathbb{C}^2$ respectively. Then the computational basis is the span by $|0\rangle$ and $|1\rangle$ and in bra-ket notation a qubit is denoted $|\varphi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle$, where $\alpha_0, \alpha_1 \in \mathbb{C}$ are the amplitudes of $|0\rangle$ and $|1\rangle$ respectively and $|\alpha_0|^2 + |\alpha_1|^2 = 1$. This is commonly explained as 'φ is a little bit zero and a little bit one at the same time', which contrasts classical bits which can only be either zero or one. If $\alpha_0 = 0 \wedge \alpha_1 = 1$, or $\alpha_0 = 1 \wedge \alpha_1 = 0$, the qubit is said to be in a classical state (note the correspondence with classical bits). The term 'classical state' is used for any state $|\varphi\rangle \in \mathbb{C}^N, N = 2^n, n \in \mathbb{N}$, which has amplitude one for some basis state and amplitude zero for all other basis states.

Qubits can be combined in two ways; tensor products of classical states or creating a superposition of classical states. A tensor product of multiple classical states results in a classical state again, as shown by Example 2.1.

Example 2.1. *Let* $|\alpha\rangle = \begin{pmatrix}\alpha_0\\ \alpha_1\end{pmatrix}$ *and* $|\beta\rangle = \begin{pmatrix}\beta_0\\ \beta_1\end{pmatrix}$ *be classical states. Then the tensor product of* $|\alpha\rangle$ *and* $|\beta\rangle$ *is again a classical state:*

$$|\alpha\rangle \otimes |\beta\rangle = |\alpha\rangle\,|\beta\rangle = |\alpha\beta\rangle = \begin{pmatrix}\alpha_0\\ \alpha_1\end{pmatrix} \otimes \begin{pmatrix}\beta_0\\ \beta_1\end{pmatrix} = \begin{pmatrix}\alpha_0\begin{pmatrix}\beta_0\\ \beta_1\end{pmatrix}\\ \alpha_1\begin{pmatrix}\beta_0\\ \beta_1\end{pmatrix}\end{pmatrix} = \begin{pmatrix}\alpha_0\beta_0\\ \alpha_0\beta_1\\ \alpha_1\beta_0\\ \alpha_1\beta_1\end{pmatrix}. \tag{1}$$

By combining more classical states, the resulting ket can get extended to a ket of a bit string of any length, resulting in consecutive tensor products. Any bit-string can be represented by a natural number. Sometimes, it is more convenient to write a natural number inside of a ket than a bit-string (for instance, if $n = a_{k-1}a_{k-2}...a_1a_0$ and $a_{k-1}, a_{k-2}, ..., a_1, a_0 \in \{0,1\}$ then $|n\rangle = |a_{k-1}a_{k-2}...a_1a_0\rangle = |a_{k-1}\rangle \otimes |a_{k-2}\rangle \otimes ... \otimes |a_1\rangle \otimes |a_0\rangle$) This notation is especially convenient when the numbers get bigger or when the ket is a term of a summation.

Tensor products can not only be used to combine classical states, but also to combine superpositions.

Note that tensor products essentially combine different Hilbert spaces: if $|0\rangle, ..., |N-1\rangle$ are an orthonormal basis of space $\mathcal{H}_A$ and $|0\rangle, ..., |M-1\rangle$ are an orthonormal basis of space $\mathcal{H}_B$, then the tensor product space $\mathcal{H} = \mathcal{H}_A \otimes \mathcal{H}_B$ is an NM-dimensional space spanned by the set of states $\{|i\rangle \otimes |j\rangle \,|\, i \in \{0, ..., N-1\}, j \in \{0, ..., M-1\}\}$. An arbitrary state in $\mathcal{H}$ is of the form

$\sum_{i=0}^{N-1}\sum_{j=0}^{M-1}\alpha_{ij}\left|i\right\rangle\otimes\left|j\right\rangle$. Such a state is called bipartite and sometimes denoted $\left|i_A j_B\right\rangle$ to indicate the space that the qubits belong to. Similarly, tripartite states that 'live' in a Hilbert space that is the tensor product of three smaller Hilbert spaces can be defined, etc.

Besides combining classical states using tensor products, a pure quantum state can be created. A pure quantum state is a superposition (corresponding to a linear combination in classical computing) of classical states, denoted:

$$\left|\phi\right\rangle = \alpha_0\left|0\right\rangle + \alpha_1\left|1\right\rangle + ... + \alpha_{2^n-1}\left|2^n-1\right\rangle = \begin{pmatrix}\alpha_0\\ \alpha_1\\ \vdots\\ \alpha_{2^n-1}\end{pmatrix}, \tag{2}$$

where $\alpha_i \in \mathbb{C}$ is the amplitude of $\left|i\right\rangle$. Note that a multi-qubit system of n qubits has $N = 2^n$ computational basis states denoted as $\left|00...00\right\rangle, \left|00...01\right\rangle, ..., \left|11...11\right\rangle$ and these states form an orthonormal basis of an N-dimensional Hilbert space. The states $\left|00...00\right\rangle$ and $\left|11...11\right\rangle$ of n qubits are sometimes abbreviated to $\left|0\right\rangle^{\otimes n}$ and $\left|1\right\rangle^{\otimes n}$ respectively.

Now that quantum states are defined, the difference between two states can be computed according to Definition 2.1.

Definition 2.1. *(2-Norm for quantum states) The difference between states $\left|\tilde{\varphi}\right\rangle$ and $\left|\varphi\right\rangle$ is defined by:*

$$\left|\left|\left|\tilde{\varphi}\right\rangle - \left|\varphi\right\rangle\right|\right|_2 = \sqrt{2(1 - Re\left\langle\tilde{\varphi}|\varphi\right\rangle)}. \tag{3}$$

Note that the above norm is ℓ_2-norm:

$$\begin{aligned}\left|\left|\left|\tilde{\varphi}\right\rangle - \left|\varphi\right\rangle\right|\right|_2 =& \sqrt{\left\langle\tilde{\varphi}-\varphi|\tilde{\varphi}-\varphi\right\rangle} \\ =& \sqrt{\left\langle\tilde{\varphi}|\tilde{\varphi}\right\rangle + \left\langle\varphi|\varphi\right\rangle - \left\langle\tilde{\varphi}|\varphi\right\rangle - \left\langle\varphi|\tilde{\varphi}\right\rangle} \\ =& \sqrt{1+1-2Re\left\langle\tilde{\varphi}|\varphi\right\rangle}.\end{aligned} \tag{4}$$

2.1 Measurements

When measuring $\left|\phi\right\rangle$ in the computational basis (the only kind of measurement that this report considers), the state collapses to one of the basis states of the Hilbert space. The probability that $\left|\phi\right\rangle$ collapses to $\left|i\right\rangle$ is $\left|\alpha_i\right|^2$. Therefore, $\left|\phi\right\rangle$ is only well-defined if $\sum_{i=0}^{2^n-1}\left|\alpha_i\right|^2 = 1$. It is important to note that all qubits in a superposition can influence the measurement outcome, as demonstrated by Example 2.2. This example uses the notions defined in Definitions 2.2 and 2.3.

Definition 2.2. *(Auxiliary qubit) Quantum operations can use qubits that do not store the inputs or outputs of the operation. Since these extra qubits do not contain useful information before or after the operation, their role is auxiliary, for which they are called auxiliary qubits. Some quantum circuits require auxiliary qubits, some quantum circuits can be made more efficient using auxiliary qubits.*

Definition 2.3. *(Phase query) Let $f : \{0,1\}^m \mapsto \{0,1\}^n$. A qubit query is a transformation O_f, acting on input register $\left|x\right\rangle$ of m qubits, writing $f(x)$ into the phase of the resulting state $\left|a\right\rangle$ of n qubits, as follows:*

$$O_f : \left|x,c,w\right\rangle = (-1)^{f(x)c}\left|x,c,w\right\rangle, \tag{5}$$

where $\left|c\right\rangle$ is a control register that controls whether the query happens and $\left|w\right\rangle$ are auxiliary qubits.

Example 2.2. *Suppose we have a 2-bit input $x = x_0x_1$ and two phase queries:*

$$\begin{cases} O_{x,\pm} : |b\rangle |0\rangle \mapsto (-1)^{x_b} |b\rangle |0\rangle & \text{for } b \in \{0,1\} \\ O'_{x,\pm} : |b\rangle |0\rangle \mapsto (-1)^{x_b} |b\rangle |b\rangle & \text{for } b \in \{0,1\}. \end{cases} \tag{6}$$

In other words, both queries map $|b\rangle$ to $(-1)^{x_b} |b\rangle$. However, in contrast to $O'_{x,\pm}$, $O_{x,\pm}$ sets the auxiliary qubit back to zero. This is an important difference, since applying $(H \otimes I)O_{x,\pm}(H \otimes I)$ or $(H \otimes I)O'_{x,\pm}(H \otimes I)$ to $|00\rangle$ lead to a different probability distribution when measuring the first qubit.

First for $O_{x,\pm}$:

$$\begin{aligned} (H \otimes I) |00\rangle &= \frac{1}{\sqrt{2}}\Big(|0\rangle + |1\rangle\Big) |0\rangle \Rightarrow \\ O_{x,\pm}(H \otimes I) |00\rangle &= \frac{1}{\sqrt{2}}\Big((-1)^{x_0} |0\rangle + (-1)^{x_1} |1\rangle\Big) |0\rangle \\ &= \frac{(-1)^{x_0}}{\sqrt{2}}\Big(|0\rangle + (-1)^{x_0 \oplus x_1} |1\rangle\Big) |0\rangle \Rightarrow \\ (H \otimes I)O_{x,\pm}(H \otimes I) |0\rangle &= \frac{(-1)^{x_0}}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}\Big(|0\rangle + |1\rangle\Big) + \frac{1}{\sqrt{2}}(-1)^{x_0 \oplus x_1}\Big(|0\rangle - |1\rangle\Big)\right) |0\rangle . \end{aligned} \tag{7}$$

Consider 2 cases:

$$\begin{cases} x_0 \oplus x_1 = 0 \Rightarrow (H \otimes I)O_{x,\pm}(H \otimes I) |00\rangle = \frac{(-1)^{x_0}}{\sqrt{2}} \frac{2}{\sqrt{2}} |00\rangle = (-1)^{x_0} |00\rangle \\ x_0 \oplus x_1 = 1 \Rightarrow (H \otimes I)O_{x,\pm}(H \otimes I) |00\rangle = \frac{(-1)^{x_0}}{\sqrt{2}} \frac{2}{\sqrt{2}} |00\rangle = (-1)^{x_0} |10\rangle . \end{cases} \tag{8}$$

So $(H \otimes I)O_{x,\pm}(H \otimes I) |00\rangle = (-1)^{x_0} |x_0 \oplus x_1\rangle |0\rangle$. Hence, measuring the first qubit in the computational basis results in $|x_0 \oplus x_1\rangle$ with probability 1.

Now for $O'_{x,\pm}$:

$$\begin{aligned} (H \otimes I) |00\rangle &= \frac{1}{\sqrt{2}}\Big(|0\rangle + |1\rangle\Big) |0\rangle = \frac{1}{\sqrt{2}}\Big(|00\rangle + |1\rangle\Big) \Rightarrow \\ O'_{x,\pm}(H \otimes I) |00\rangle &= \frac{1}{\sqrt{2}}\Big((-1)^{x_0} |00\rangle + (-1)^{x_1} |11\rangle\Big) \Rightarrow \\ (H \otimes I)O'_{x,\pm}(H \otimes I) |00\rangle &= \frac{1}{2}\Big((-1)^{x_0} |00\rangle + (-1)^{x_0} |10\rangle + (-1)^{x_1} |01\rangle - (-1)^{x_0} |11\rangle\Big). \end{aligned} \tag{9}$$

Measuring the first qubit in the computational basis results in $|0\rangle$ with probability $1/2$ and $|1\rangle$ with probability $1/2$.

This means that whether or not auxiliary qubits are put back to their initial value clearly influences the measurement outcomes. Therefore, it is important to make sure an algorithm always ends with all auxiliary qubits equal to their initial value.

The fact that measurements of superpositions are stochastic, highly influences the success probability of quantum algorithms. Even though an algorithm might lead to a superposition of the desired output, there is no guarantee that this output will be measured. To increase the amplitude of the desired output, i.e. increase the success probability of a quantum algorithm, amplitude amplification is commonly used. Even though this report uses the notion of amplitude amplification to bring down the complexity of the HHL algorithm, the amplitude amplification algorithm will not be discussed in detail. The only knowledge about the amplitude amplification algorithm that is needed in this report is given in Theorem 2.1.

Theorem 2.1. *(Amplitude amplification [10]) If an algorithm $\mathcal{A}$ has success probability p, then amplitude amplification needs $\mathcal{O}(1/\sqrt{p})$ repetitions of $\mathcal{A}$ to increase the success probability arbitrarily close to one.*

2.2 Unitary operations

A unitary operation changes a state $|\phi\rangle$ to $|\psi\rangle = \beta_0 |0\rangle + \beta_1 |1\rangle + ... + \beta_{N-1} |N-1\rangle$. Since quantum mechanics only allows linear operations, such an operation can be represented by an $N \times N$ complex matrix U:

$$U \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{N-1} \end{pmatrix} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_{N-1} \end{pmatrix} \tag{10}$$

Note that subscripts are sometimes used to denote the number of qubits that a operation is acting on.

Since the $U |\phi\rangle = |\psi\rangle$ and $|\psi\rangle$ should be a well-defined quantum state again, i.e. $\sum |\beta_i|^2 = 1$, Lemma 2.2 requires operation U to be unitary, as defined in Definition 2.4.

Definition 2.4. *(Unitary) A matrix U is unitary if it is invertable and its inverse U^{-1} equals its conjugate transpose U^*.*

Lemma 2.2 uses the notion of bra's whereas before only kets were used (as Lemma 2.2 is commonly known and easy to prove, a proof is omitted in this report). A ket $|\phi\rangle$ can be denoted by a column vector and a bra is simply the transpose of a ket, denoted by $\langle\phi|$. This also allows to denote the inner product of two kets $|\phi\rangle$ and $|\phi'\rangle$ by $\langle\phi|\phi'\rangle$ or $\langle\phi, \phi'\rangle$ (note the correspondence with the inner product of vectors).

Lemma 2.2. *Let V be a finite dimensional inner product space and U an operation working on V. Then the following are equivalent:*

1. *Matrix U is unitary, i.e. $U^*U = UU^* = I$.*
2. *$||Ux|| = ||x||$ for all $x \in V$.*
3. *$\langle Ux, Uy\rangle = \langle x, y\rangle$ for all $x, y \in V$.*

2.3 Entangled states

It is important to understand what entangled states, as defined in Definition 2.5, are and how they impact measurements. This impact is shown in Example 2.3.

Definition 2.5. *(Entanglement) Let $\mathcal{H}_A$ and $\mathcal{H}_B$ be two Hilbert spaces. A bipartite state $|\phi\rangle \in \mathcal{H} = \mathcal{H}_A \otimes \mathcal{H}_B$ is called entangled if it cannot be written as a tensor product $|\phi_A\rangle \otimes |\phi_B\rangle$, where $|\phi_A\rangle$ lives in the first space and $|\phi_B\rangle$ lives in the second space.*

Example 2.3. *Say Alice and Bob each have one qubit and these qubits together form an EPR-pair (note there are 4 maximally entangled states of two qubits, called EPR-pairs or Bell states):*

$$|\phi\rangle = \frac{1}{\sqrt{2}} |0_A 0_B\rangle + \frac{1}{\sqrt{2}} |1_A 1_B\rangle \tag{11}$$

It is easy to verify that $|\phi\rangle$ can not be written as a tensor product of some $|\phi_A\rangle$ and $|\phi_B\rangle$, where $|\phi_A\rangle$ and $|\phi_B\rangle$ are the qubits of Alice and Bob respectively. Furthermore, neither of the two qubits is in a classical state ($|0\rangle$ or $|1\rangle$). However, if one qubit is measured, this determines the value of the other qubit. Say that the first qubits gets measured and $|0\rangle$ is observed, then the EPR-pair has collapsed to $|00\rangle$. This means that the second qubit has been collapsed to $|0\rangle$ as well.

2.4 Important gates

First the Hadamard gate is defined in Definition 2.6. Theorem 2.3 gives a variation of this gate, after which Lemma 2.4 shows that this gates also results in a well defined quantum state.

Definition 2.6. *(Hadamard gate) The Hadamard gate is defined as:*

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}. \tag{12}$$

This means that applying the Hadamard gate to a state $|i\rangle$ results in:

$$\Rightarrow H^{\otimes t}|i\rangle = \frac{1}{\sqrt{2^t}}\sum_{j\in\{0,1\}^t}(-1)^{i\cdot j}|j\rangle\,, \tag{13}$$

where $i \cdot j$ denotes the inner product of two bit-strings.

Theorem 2.3. *The Hadamard-like gate (denoted by H'_t), applied to t qubits, results in:*

$$H'_t : |0\rangle^{\otimes t} \mapsto |\Psi_0\rangle = \sqrt{\frac{2}{T}}\sum_{\tau=0}^{T-1}\sin\left(\frac{\pi(\tau+1/2)}{T}\right)|\tau\rangle\,. \tag{14}$$

Lemma 2.4. *Quantum gate H'_t applied to $|0\rangle^{\otimes t}$ results in a well defined quantum state.*

The proof of this lemma can be found in Appendix E.

The second important gate is the controlled rotation, defined in Definition 2.7.

Definition 2.7. *(Controlled rotation) Let $\theta \in \mathbb{R}$ and let $\tilde{\theta} \in \{0,1\}^t$ be the representation of θ using t bits. Define the unitary matrix:*

$$R = \sum_{\tilde{\theta}\in\{0,1\}^t}\begin{pmatrix}\cos(\theta) & -\sin(\theta)\\ \sin(\theta) & \cos(\theta)\end{pmatrix}\otimes|\tilde{\theta}\rangle\langle\tilde{\theta}|. \tag{15}$$

Then R acts as:

$$R : |0\rangle|\tilde{\theta}\rangle \mapsto \big(\cos(\theta)|0\rangle + \sin(\theta)|1\rangle\big)|\tilde{\theta}\rangle\,. \tag{16}$$

Note that $f(\theta)$, where $f(\cdot)$ can be any function, can also be used instead of θ. Then R_f is defined as:

$$R_f = \sum_{\tilde{\theta}\in\{0,1\}^t}\begin{pmatrix}\cos(f(\theta)) & -\sin(f(\theta))\\ \sin(f(\theta)) & \cos(f(\theta))\end{pmatrix}\otimes|\tilde{\theta}\rangle\langle\tilde{\theta}|, \tag{17}$$

and acts as:

$$R_f : |0\rangle|\tilde{\theta}\rangle \mapsto \big(\cos(f(\theta))|0\rangle + \sin(f(\theta))|1\rangle\big)|\tilde{\theta}\rangle\,. \tag{18}$$

The next important gate is the controlled U-gate, defined in Definition 2.8.

Definition 2.8. *(Controlled U-gate) Let U be a unitary with eigenvector u and eigenvalue $\lambda = e^{2\pi i\phi}$. Let it be possible to write the phase $\phi \in [0,1)$ using t bits, where $T = 2^t$ for some $T \in \mathbb{N}$ (an estimation of t bits results in a similar analysis of the algorithm [16]). The controlled U-gate is defined as:*

$$U|b\rangle|u\rangle = e^{2\pi ib\phi}|b\rangle|u\rangle, \tag{19}$$

where b is a binary vector.

Note that this definition holds for any eigenvector u of U, but does not depend on u being an eigenvector. If U is applied to a state that is not an eigenstate of U, then this state can be written as a superposition of eigenvectors of U. One can easily verify that the same reasoning as above holds for a superposition of eigenvectors.

These controlled-U gates can be summarized by $U'_{t,n} = \sum_{k=0}^{T-1} |k\rangle\langle k| \otimes e^{iUkt_0/T}$, called Hamiltonian simulation. Section 3 uses this notation.

The last important gate is the quantum Fourier transform, defined in Definition 2.9. After the definition, this operation is rewritten to the form that is needed in the next subsection.

Definition 2.9. *(Quantum Fourier transform) Let F_T be the unitary matrix (for row j and column k):*

$$F_T = \frac{1}{\sqrt{T}} \sum_{j,k\in\{0,\dots,T-1\}} (e^{2\pi i/T})^{jk} |j\rangle\langle k| = \frac{1}{\sqrt{T}} \begin{pmatrix} & \vdots & \\ \dots & (e^{2\pi i/T})^{jk} & \dots \\ & \vdots & \end{pmatrix} \in \mathbb{C}^{T\times T}. \tag{20}$$

Matrix F_T implements the quantum Fourier transform (QFT) and its inverse ($QFT^\dagger$):

$$|k\rangle \xrightarrow{QFT} \frac{1}{\sqrt{T}} \sum_{j=0}^{T-1} e^{2\pi ijk/T} |j\rangle \quad \textit{and} \tag{21a}$$

$$|j\rangle \xrightarrow{QFT^\dagger} \frac{1}{\sqrt{T}} \sum_{k=0}^{T-1} e^{-2\pi ijk/T} |k\rangle. \tag{21b}$$

The quantum Fourier transform can be rewritten using $k = \phi$, $T = 2^t$ and $j = j_0 ... j_{t-1}$:

$$\begin{aligned} F_T|\phi\rangle &= \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} e^{2\pi i\phi\cdot(j_0\dots j_{t-1})2^{-t}} |j_0\dots j_{t-1}\rangle \\ &= \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} e^{2\pi i\phi\cdot(\sum_{l=0}^{t} j_l 2^{-l})} |j_0\dots j_{t-1}\rangle \\ &= \frac{1}{\sqrt{2^t}} \bigotimes_{l=1}^{t} \big(|0\rangle + e^{2\pi i\phi\cdot 2^{-l}}|1\rangle\big). \end{aligned} \tag{22}$$

Consider the following example to see why the last equality holds.

Example 2.4. *For $t = 2$:*

$$\begin{aligned} \frac{1}{\sqrt{2^2}} \sum_{j=0}^{3} e^{2\pi i\phi(2^{-1}j_1+2^{-2}j_2)} |j_1 j_2\rangle = \frac{1}{\sqrt{2^2}} \Big[& e^{2\pi i\phi(2^{-1}0+2^{-2}0)} |00\rangle + e^{2\pi i\phi(2^{-1}0+2^{-2}1)} |01\rangle + \\ & e^{2\pi i\phi(2^{-1}1+2^{-2}0)} |10\rangle + e^{2\pi i\phi(2^{-1}1+2^{-2}1)} |11\rangle \Big] \end{aligned} \tag{23}$$

$$
\begin{aligned}
&= \frac{1}{\sqrt{2^2}} \begin{pmatrix} 1 \\ e^{2\pi i\phi 2^{-2}} \\ e^{2\pi i\phi 2^{-1}} \\ e^{2\pi i\phi(2^{-1}+2^{-2})} \end{pmatrix} \\
&= \frac{1}{\sqrt{2^2}} \begin{pmatrix} 1 \\ e^{2\pi i\phi 2^{-1}} \end{pmatrix} \otimes \begin{pmatrix} 1 \\ e^{2\pi i\phi 2^{-2}} \end{pmatrix} \\
&= \frac{1}{\sqrt{2^2}} \big(|0\rangle + e^{2\pi i\phi 2^{-1}} |1\rangle\big) \otimes \big(|0\rangle + e^{2\pi i\phi 2^{-2}} |1\rangle\big) \\
&= \frac{1}{\sqrt{2^2}} \bigotimes_{l=1}^{2} \big(|0\rangle + e^{2\pi i\phi 2^{-l}} |1\rangle\big).
\end{aligned}
$$

2.5 Phase estimation

This section is based on [16]. The goal of the phase estimation algorithm is to determine the eigenvalues of a unitary operation. For an eigenstate u of unitary matrix U, phase estimation finds a numerical estimation for ψ in $e^{i\psi}$, where $e^{i\psi}$ is an eigenvalue of U. Figure 1 is an overview of the phase estimation algorithm. The algorithm assumes that $|u\rangle$ can be prepared and that operation U^{2^j} can be applied and sees this as black boxes (i.e. oracles). Since phase estimation makes use of these black boxes, it is not a stand-alone procedure; it can only be used as subroutine, but it can do very interesting and important computations needed for other modules. For instance, phase estimation is a very important subroutine in the HHL algorithm.

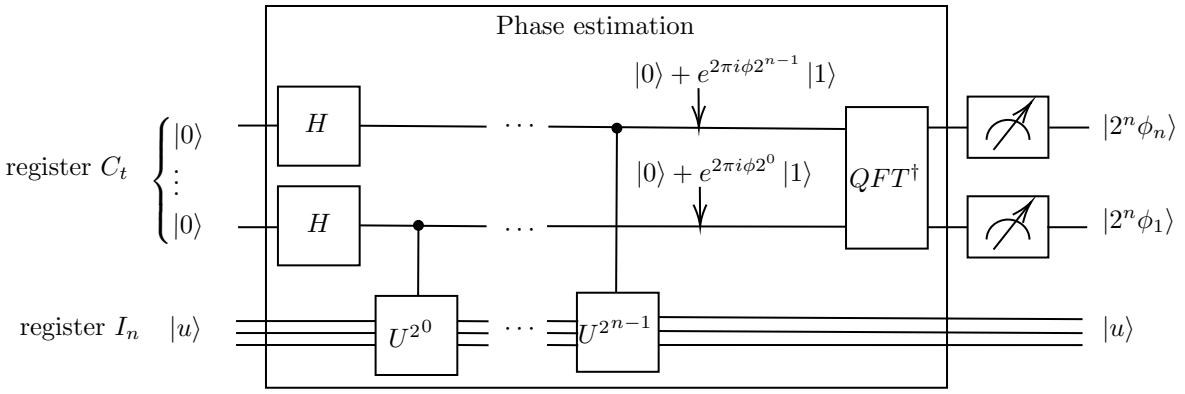


Figure 1: Phase estimation algorithm overview

Phase estimation works on two registers: register C_t containing t qubits, and register I_n containing n qubits. For notation purposes define $N = 2^n$ and $T = 2^t$. To choose the value of t the number of digits of accuracy of the estimation and the probability that the phase estimation procedure is successful, should be taken into account. These dependencies will become clear in the following analysis. The value of n is determined by the number of qubits that is needed to store $|u\rangle$.

The algorithm uses Hadamard gates, controlled U-gates, and the quantum Fourier transform, defined in Definition 2.6, 2.8, and 2.9 respectively.

The algorithm starts by applying Hadamard gates, as defined in Definition 2.6, to all qubits in register C_t. After the Hadamard gates, register C_t contains:

$$\frac{1}{\sqrt{2^t}}\Big[\big(|0\rangle + |1\rangle\big) \otimes ... \otimes \big(|0\rangle + |1\rangle\big)\Big], \tag{24}$$

i.e. the t qubits in register C_t are all in the superposition $\frac{1}{\sqrt{2}}\big(|0\rangle + |1\rangle\big)$. Next, controlled U-gates are applied to $|u\rangle$ in register I_n, for which the qubits in register C_t act as control bits. This means that for $j = 0, ..., t$, the j-th qubit in register C_t determines whether or not U^{2^j} is applied to $|u\rangle$ in register I_n. Since all the qubits in register C_t are in the same superposition, the effect of each power of the controlled U gate can be described generally using Definition 2.8:

$$\begin{aligned} U^{2^j}\frac{1}{\sqrt{2}}\big(|0\rangle + |1\rangle\big)|u\rangle &= \frac{1}{\sqrt{2}}\big(|0\rangle (e^0)^{2^j}|u\rangle + (e^{2\pi i\phi})^{2^j}|1\rangle|u\rangle\big) \\ &= \frac{1}{\sqrt{2}}\big(|0\rangle|u\rangle + (e^{2\pi i\phi})^{2^j}|1\rangle|u\rangle\big). \end{aligned} \tag{25}$$

Note that this did only alter the qubits in register C_t (the control qubits) and not the qubits in register I_n (the target qubits), which means that register C_t contains:

$$\frac{1}{\sqrt{2^t}}\Big[\big(|0\rangle + (e^{2\pi i\phi})^{2^{t-1}}|1\rangle\big) \otimes ... \otimes \big(|0\rangle + (e^{2\pi i\phi})^{2^0}|1\rangle\big)\Big] = \frac{1}{\sqrt{2^t}}\bigotimes_{l=1}^{t}\big(|0\rangle + e^{2\pi i\phi 2^{t-l}}|1\rangle\big). \tag{26}$$

Remark. The phase can be written as $\phi = 0.\phi_1...\phi_t = \sum_{k=1}^{t} 2^{-k}\phi_k$, where $\phi_0, ..., \phi_t \in \{0, 1\}$. The bits of ϕ are now in the exponents:

$$\begin{aligned} (e^{2\pi i\phi})^{2^{t-l}} &= (e^{2\pi i})^{\sum_{k=1}^{t} 2^{t-l-k}\phi_k} \\ &= (e^{2\pi i})^{2^{t-l-1}\phi_1} \cdot ... \cdot (e^{2\pi i})^{2^0\phi_{t-l}} \cdot (e^{2\pi i})^{2^{-1}\phi_{t-l+1}} \cdot ... \cdot (e^{2\pi i})^{2^{-l}\phi_t} \\ &= 1 \cdot ... \cdot 1 \cdot (e^{2\pi i})^{2^{-1}\phi_{t-l+1}} \cdot ... \cdot (e^{2\pi i})^{2^{-l}\phi_t} \\ &= (e^{2\pi i})^{\phi_{t-l+1}...\phi_t}. \end{aligned} \tag{27}$$

So, $l = 1$ leads to the last bit of ϕ and $l = 2$ leads to the last 2 bits of ϕ, and so on.

Since the different bits of ϕ are now in the frequency domain, the inverse of the Fourier transform is applied to register C_t to move this information to the time domain. Using Equation (22), it is clear that Equation (26) equals the Fourier transform of $|\phi\rangle$. Note that the quantum Fourier transform swaps around the order of the qubits. So, applying $QFT^\dagger$ on the qubits in register C_t results in $|\phi_t\rangle \otimes ... \otimes |\phi_1\rangle$. Define $\phi' = \phi_t...\phi_1$ to summarize phase estimation:

$$|0\rangle^{\otimes t}|u\rangle \mapsto |\phi'\rangle|u\rangle. \tag{28}$$

The last step of the phase estimation is to measure register C_t. Measuring the first register of the above quantum state, results in $|\phi'\rangle$, i.e. the t most important bits of the phase, with certainty (if the quantum computations occurred without error, which can happen on paper but rarely happens in practice). At the beginning of this section is mentioned that t should be chosen with both the preferred precision of the estimation and the success probability of the algorithm in mind. So, this boils down to choosing t equal to the preferred precision of the estimation. Therefore, t can be called the precision of the phase estimation algorithm.

As phase estimation is used as subroutine in HHL, no measurements will be done at this point. Measuring now would lead to one eigenvalue, but that is not enough to find x such that $Ax = b$. Instead of doing a measurement at this point, controlled rotation is used to 'store' the information about the eigenvalue in an auxiliary qubit.

3 HHL algorithm

Let $Ax = b$ be a system of linear equations that needs to be solved, where $A \in \mathbb{C}^{N\times N}$ and $x, b \in \mathbb{C}^N$ for some $N = 2^n$ with $N, n \in \mathbb{N}$. The goal of the HHL algorithm is to find $A^{-1}b$. This can also be done classically. The best general purpose classical matrix-inversion algorithm, the conjugate-gradient method, runs in $\mathcal{O}(Ns\kappa \log(1/\epsilon))$, where s is the sparsity of A, as defined in Definition 3.1, and κ the condition number of A [10]. Since N is very big for AES, because AES is not linear and linearization results in an enormous system of linear equations, the complexity is very high. Up until now, there is no practically executable attack know, using this approach. The HHL algorithm might change this. The complexity of HHL is $\mathcal{O}\big(\log(N)s^2\kappa^2\epsilon\big)$. So a quadratic speed-up in N, however, the dependency on s, κ, and ϵ is worse. Since N is known to be big for AES and s is known to be small for AES, it is worth investigating whether or not the HHL algorithm can be used instead of the conjugate-gradient method.

Definition 3.1. *(Sparsity) A matrix is called s-sparse, if every row of this matrix has at most s non-zero terms. Since every term of an equation corresponds to a non-zero term in the matrix, for the Macaulay approach, this report considers an equation s-sparse if it has at most s terms.*

The rest of this section is based on [15] and [10] which both provide an analysis of the HHL algorithm, overlapping in some parts; complementing each other in other parts. However, at some points this report considers the papers to be erroneous. Another reason that this section deviates from the papers is that this section only considers well-conditioned matrices. It is indicated where the report deviates from the literature.

Remark. The reason to only invert well-conditioned matrices (or only the well-conditioned part of the matrix) is that the algorithm extracts the eigenvalues of A^{-1} at some point. Therefore, the algorithm essentially works with $1/\lambda$ where λ is an eigenvalue of A, which means it is important to take into account the numerical stability of the algorithm. If the algorithm would be used to invert an eigenvalue λ that is small with respect to κ, for instance $\lambda = \epsilon_\kappa/\kappa$ for $0 < \epsilon_\kappa << 1$, then $1/\lambda = \kappa/\epsilon_\kappa$. A small relative error in $1/\epsilon_\kappa$, will give a result deviating from the true value by many times κ. Since κ is considered the characteristic scale of the matrix, this would mean that the relative error is big. HHL uses filter functions to make sure only the eigenvalues that are at least $1/\kappa$ are inverted. These filter functions make the analysis of the algorithm harder and less intuitive. Therefore, this project only considers well-conditioned matrices, which makes the filter functions unnecessary.

The input of the HHL algorithm consists of an Hermitian matrix $A \in \mathbb{C}^{N\times N}$, according to Definition 3.2, with condition number $\kappa = |\lambda_{\max}/\lambda_{\min}|$, where $\lambda_{\max}$ and $\lambda_{\min}$ are the eigenvalues of A with respectively the maximum and minimum ℓ_2-norm, error ϵ, and vector $b \in \mathbb{C}^N$. The goal of the HHL algorithm is to find a vector $x \in \mathbb{C}^N$, satisfying $Ax = b$. However, instead of computing $x = A^{-1}b$ the HHL algorithm writes:

$$|b\rangle = \sum_{i=1}^{N} b_i |i\rangle = \sum_{j=1}^{N} \beta_j |u_j\rangle , \tag{29}$$

where $u_1, ..., u_N$ are the eigenvectors of A, and tries to find an $\tilde{x}$, using quantum computing, such that :

$$||A|\tilde{x}\rangle - |b\rangle||_2 \leq \epsilon, \tag{30}$$

with probability at least $1/2$. Note that the goal is to get this probability close to one, however, if the probability is at least $1/2$, amplitude amplification can be used to increase this probability.

Definition 3.2. *(Hermitian matrix) Matrix A is Hermitian iff $A = A^\dagger$ where $A^\dagger$ is the conjugate transpose of A.*

Let $\lambda_1, ..., \lambda_N$ be the eigenvalues of A, then with out loss of generality $|\lambda_1| \geq ... \geq |\lambda_N|$. Rescale A such that $|\lambda_1| = 1$ and $\kappa = 1/|\lambda_N|$, where κ is the condition number of A. Scale $|b\rangle$ such that it is a normalized state, i.e. $\langle b|b\rangle = 1$.

Since the eigenvectors of a Hermitian matrix are orthogonal and A is a Hermitian matrix, it is diagonalizable, i.e. it can be written as:

$$A = \sum_{j=1}^{N} \lambda_j |u_j\rangle\langle u_j| \Rightarrow A^{-1} = \sum_{j=1}^{N} \frac{1}{\lambda_j} |u_j\rangle\langle u_j|, \tag{31}$$

where $\lambda_1, ..., \lambda_N$ and $u_1, ..., u_N$ are the eigenvalues and eigenvectors of A respectively. Combining this with Equation 29 results in:

$$A\,|b\rangle = \sum_{j=1}^{N} \lambda_j \beta_j\,|u_j\rangle \Rightarrow A^{-1}\,|b\rangle = \sum_{j=1}^{N} \frac{\beta_j}{\lambda_j}\,|u_j\rangle\,. \tag{32}$$

Even though this can be written theoretically, the values λ_j and the vectors u_j are unknown at this point. However, this expression is used to recognize the solution later on.

3.1 Algorithm overview

The main idea of the HHL algorithm is that phase estimation can estimate the eigenvalues of a matrix. These can be inversed transposed to the amplitudes of an auxiliary qubit, meaning that the amplitudes of the basis states $|0\rangle$ and $|1\rangle$ in the qubit depend on the eigenvalues. By uncomputing the garbage that ended up in the other registers, the input vector b is multiplied by these amplitudes (up to normalization). This essentially computes $A^{-1}b = x$.

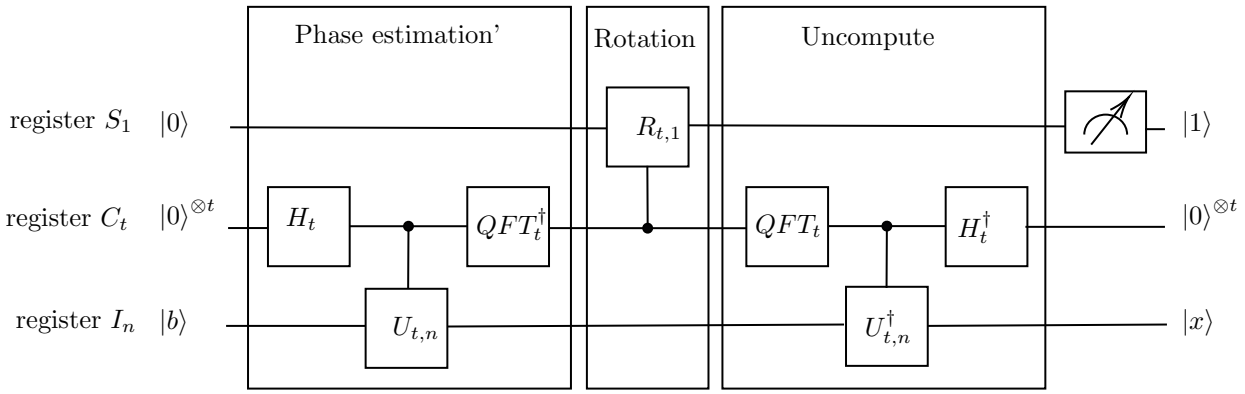


Figure 2: HHL′$(|b\rangle, A, t_0, T)$

Figure 2 gives an overview of the HHL algorithm. The algorithm uses:

- register S_1 (consisting of one qubit) used as auxiliary qubit,
- register C_t (consisting of t qubits, with $2^t = T$, where t is the precision of the phase estimation as explained in Section 2.5) used to store an estimate of the phases of matrix e^{iA}, and

- register I_n (consisting of n qubits, where $2^n = N \in \mathbb{N}$ and $b \in \mathbb{C}^N$) used to store a representation of vector b.

Because HHL uses an N-dimensional Hilbert space to store the result x, one can only learn $\log N = n$ bits of x from one run of the algorithm.

Algorithm 1: HHL

Input: Vector $b \in \mathbb{C}^N$, matrix $A \in \mathbb{C}^{N \times N}$ (s-sparse matrix with oracle access to its elements), and parameters $t_0 = \mathcal{O}\left(\frac{\kappa}{\epsilon}\right)$, $T = \mathcal{O}\left(\log(N) s^2 t_0\right)$, and ϵ (the desired precision).

1 Prepare quantum state vector $|b\rangle$.

2 Apply $\mathcal{O}(\kappa)$ round of amplitude amplification on $\text{HHL}'(|b\rangle, A, t_0, T)$.

HHL′$(|b\rangle, A, t_0, T)$

Phase estimation ($\tilde{P}$)

3 Prepare state $|\Psi_0\rangle$, i.e. apply H_t to register C_t.

4 Create in register C_t a superposition of quantum states that correspond to the eigenvalues of A, i.e. apply $U_{t,n}$ to register I_n, controlled by register C_t.

5 Transpose the information on the eigenvalues of A from the kets to the amplitudes, i.e. apply $\text{QFT}_t^\dagger$ to register C_t.

6 Relabel $|k\rangle$ with $|\tilde{\lambda_k}\rangle$, where $\tilde{\lambda_k} = 2\pi k / t_0$.

Rotation

7 Transform qubit S_1 to $|h(\tilde{\lambda_k})\rangle$, with $|h(\lambda)\rangle$ as defined in Equation (47), i.e. apply $R_{t,1}$ to qubit S_1, controlled by register C_t .

Uncompute

8 Undo the relabeling of $|k\rangle$.

9 Undo $\text{QFT}_t^\dagger$, i.e. apply QFT_t to register C_t.

10 Undo $U_{t,n}$, i.e. apply $U_{t,n}^\dagger$ to register I_n, controlled by register C_t.

11 Undo H_t, i.e. apply $H_t^\dagger$ to register C_t.

Post-selection

12 Measure qubit S_1.

13 **if** *Measurement results in 0* **then**
Go back to step 1.
else if *Measurement results in 1* **then**
Return register I_n.

Output: Quantum state vector $|\tilde{x}\rangle$.

In order to apply the HHL algorithm, vector b needs to be prepared as a quantum state $|b\rangle$ in register I, and $|0\rangle^{\otimes t}$ needs to be prepared in register C_t.

As shown in Figure 2, the HHL algorithm starts with a subroutine that is similar to phase estimation. Section 2 explains how phase estimation works. The subroutine that the HHL algorithm uses, is not exactly the same as the phase estimation algorithm: it does not use Hadamard gates, it uses a gate that is similar to Hadamard gates. The rest of the subroutine is the same as the phase estimation algorithm: a Hamiltonian evolution on register I_n controlled by register C_t, followed by

an inverse quantum Fourier transform on register C_t. Section 2.5 explains that phase estimation results in the bits of an eigenvalue being contained in register C_t. However, in order to solve $A|x\rangle = |b\rangle$, knowing one eigenvalue of A is not sufficient. Therefore, instead of a measurement, a controlled rotation is done to move the value from the qubits in register C_t to the coefficients of the superposition of the qubit in register I_n. Controlled rotation consists of only one gate: a rotation gate on auxiliary qubit S_1 controlled by register C_t. The HHL algorithm ends with uncomputing the phase estimation which consists of the inverse of the gates that are used in the phase estimation, in reversed order: first a quantum Fourier transform on register C_t, then a Hamiltonian evolution on register I_n controlled by register C_t, and finally the inverse of the Hadamard-like gates that the algorithm started with.

The details of the HHL algorithm can be found in Algorithm 1. The next section analyses how the states change with every step of the algorithm.

3.2 Analysis of the basic HHL algorithm

This section will show every step of the HHL algorithm in a bit more detail. However, to increase readability, the technical details are left out and can be found in Section 3.3.

3.2.1 Preparation

For HHL, the input register I_n is prepared as Equation (29).

3.2.2 Phase estimation

The first step of the HHL algorithm is to decompose $|b\rangle$ in the eigenvector basis, using phase estimation as explained in Section 2.5.

Phase estimation uses Hamiltonian simulation $U_{t,n}$, to apply the unitary matrix e^{-iA}, τ times. This corresponds to τ time-steps of evolution induced by Hamiltonian A. Therefore, the state that $U_{t,n}$ is applied to can be seen as a clock register. Clocks can be based on different states. However, quantum clocks based on $|\psi_0\rangle$, as defined in Equation (33), seem to be particularly desirable according to [18].

Remark. Note that [18] denotes the optimal state by $|\psi_{\text{opt}}\rangle$ and the state create by Hadamard gates by $|\psi_0\rangle$, whereas this report denotes the optimal state by $|\psi_0\rangle$.

To create this optimal state, the HHL algorithm uses Hadamard-like gates, instead of Hadamard gates. According to Theorem 2.3, this results in the quantum state:

$$|\Psi_0\rangle = \sqrt{\frac{2}{T}} \sum_{\tau=0}^{T-1} \sin\left(\frac{\pi(\tau + 1/2)}{T}\right) |\tau\rangle\,, \tag{33}$$

which is a well-defined quantum state according to Lemma 2.4. Note that using Hadamard gates instead would result in an algorithm which should give the same output, however, it will be less efficient.

After the H_t' gate, controlled $U_{t,n}$ gates, as defined in Definition 2.8, need to be applied on $|\Psi_0\rangle \otimes |b\rangle$. The input matrix A is used as unitary matrix to define $U_{t,n}$, hence $U_{t,n} = \sum_{k=0}^{T-1} |k\rangle\langle k| \otimes e^{iAkt_0/T}$, where $t_0 = O(\kappa/\epsilon)$. This operation acts on registers I_n and C_t:

$$\left[\sum_{k=0}^{T-1}|k\rangle\langle k|\otimes e^{iAkt_0/T}\right]\cdot\left[\sqrt{\frac{2}{T}}\sum_{\tau=0}^{T-1}\sin\left(\frac{\pi(\tau+1/2)}{T}\right)|\tau\rangle\otimes\sum_{j=1}^{N}\beta_j|u_j\rangle\right] \tag{34}$$

$$=\left[\sum_{k=0}^{T-1}|k\rangle\langle k|\otimes e^{iAkt_0/T}\right]\cdot\left[\sqrt{\frac{2}{T}}\sum_{j=1}^{N}\beta_j\sum_{\tau=0}^{T-1}\sin\left(\frac{\pi(\tau+1/2)}{T}\right)|\tau\rangle\otimes|u_j\rangle\right]$$

$$=\sqrt{\frac{2}{T}}\sum_{j=1}^{N}\beta_j\sum_{k=0}^{T-1}\sum_{\tau=0}^{T-1}\left[\sin\left(\frac{\pi(\tau+1/2)}{T}\right)|k\rangle\langle k|\cdot|\tau\rangle\right]\otimes\left[e^{iAkt_0/T}\cdot|u_j\rangle\right]$$

note that $\langle k|\cdot|\tau\rangle=\delta_{k\tau}$

$$=\sqrt{\frac{2}{T}}\sum_{j=1}^{N}\beta_j\sum_{k=0}^{T-1}\sum_{\tau=0}^{T-1}\left[\sin\left(\frac{\pi(\tau+1/2)}{T}\right)|k\rangle\,\delta_{k\tau}\right]\otimes\left[e^{iAkt_0/T}\cdot|u_j\rangle\right]$$

note that $\sum_k\delta_{k\tau}f(k)=f(\tau)$

$$=\sqrt{\frac{2}{T}}\sum_{j=1}^{N}\beta_j\sum_{\tau=0}^{T-1}\left[\sin\left(\frac{\pi(\tau+1/2)}{T}\right)|\tau\rangle\right]\otimes\left[e^{iA\tau t_0/T}\cdot|u_j\rangle\right]$$

note that $f(A)\,|u_j\rangle=f(\lambda_j)\,|u_j\rangle$

$$=\sqrt{\frac{2}{T}}\sum_{j=1}^{N}\beta_j\sum_{\tau=0}^{T-1}\left[\sin\left(\frac{\pi(\tau+1/2)}{T}\right)|\tau\rangle\right]\otimes\left[e^{i\lambda_j\tau t_0/T}\cdot|u_j\rangle\right]$$

$$=\sqrt{\frac{2}{T}}\sum_{j=1}^{N}\beta_j\left[\sum_{\tau=0}^{T-1}e^{i\lambda_j\tau t_0/T}\sin\left(\frac{\pi(\tau+1/2)}{T}\right)|\tau\rangle\right]\otimes|u_j\rangle\,.$$

The summation $\sum_{\tau=0}^{T-1}\sin\left(\frac{\pi(\tau+1/2)}{T}\right)e^{i\lambda_j\tau t_0/T}|\tau\rangle$ contains $e^{i\lambda_j\tau t_0/T}$, which is a function in τ with frequency λ_j. Since this is essentially the value that the HHL algorithm is looking for, the next step is to extract this frequency. So, an inverse Fourier transform, as defined in Definition 2.9, is applied:

$$\sum_{\tau=0}^{T-1}e^{i\lambda_j\tau t_0/T}\sin\left(\frac{\pi(\tau+1/2)}{T}\right)|\tau\rangle\xrightarrow{QFT_t^\dagger} \tag{35}$$

$$\sum_{\tau=0}^{T-1}e^{i\lambda_j\tau t_0/T}\sin\left(\frac{\pi(\tau+1/2)}{T}\right)\frac{1}{\sqrt{T}}\sum_{k=0}^{T-1}e^{-2\pi ik\tau/T}|k\rangle$$

$$=\frac{1}{\sqrt{T}}\sum_{k=0}^{T-1}\sum_{\tau=0}^{T-1}e^{i\lambda_j\tau t_0/T}e^{-2\pi ik\tau/T}\sin\left(\frac{\pi(\tau+1/2)}{T}\right)|k\rangle\,.$$

Substituting this back leads to:

$$\sqrt{\frac{2}{T}}\sum_{j=1}^{N}\beta_j\left[\sum_{\tau=0}^{T-1}e^{i\lambda_j\tau t_0/T}\sin\left(\frac{\pi(\tau+1/2)}{T}\right)|\tau\rangle\right]\otimes|u_j\rangle\xrightarrow{I_1\otimes QFT_t^\dagger\otimes I_n} \tag{36}$$

$$\sqrt{\frac{2}{T}}\sum_{j=1}^{N}\beta_j\left[\frac{1}{\sqrt{T}}\sum_{k=0}^{T-1}\sum_{\tau=0}^{T-1}e^{i\lambda_j\tau t_0/T}e^{-2\pi ik\tau/T}\sin\left(\frac{\pi(\tau+1/2)}{T}\right)|k\rangle\right]\otimes|u_j\rangle\,.$$

Simplify this to:

$$\begin{aligned}\sqrt{\frac{2}{T}}\sum_{j=1}^{N}\beta_j\left[\frac{1}{\sqrt{T}}\sum_{k=0}^{T-1}\sum_{\tau=0}^{T-1}e^{i\lambda_j\tau t_0/T}e^{-2\pi ik\tau/T}\sin\left(\frac{\pi(\tau+1/2)}{T}\right)|k\rangle\right]\otimes|u_j\rangle \\ =\sum_{j=1}^{N}\beta_j\sum_{k=0}^{T-1}\alpha_{k|j}\,|k\rangle\otimes|u_j\rangle\,,\end{aligned} \tag{37}$$

by defining:

$$\alpha_{k|j}=\frac{\sqrt{2}}{T}\sum_{\tau=0}^{T-1}e^{i\lambda_j\tau t_0/T}e^{-2\pi ik\tau/T}\sin\left(\frac{\pi(\tau+1/2)}{T}\right). \tag{38}$$

According to Lemma 3.4, $|\alpha_{k|j}|$ is only large when $k\approx\frac{\lambda_j t_0}{2\pi}$. This means that the state can be approximated by only taking into account the terms in which $k\approx\frac{\lambda_j t_0}{2\pi}$. The terms that are neglected will lead to some error. The proof of Lemma 3.4 gives an upper bound on the omitted terms. Relabeling basis states $|k\rangle$ by defining $\tilde{\lambda_k}=\frac{2\pi k}{t_0}$ leads to:

$$\sum_{j=1}^{N}\beta_j\sum_{k=0}^{T-1}\alpha_{k|j}\,|k\rangle\otimes|u_j\rangle=\sum_{j=1}^{N}\beta_j\sum_{k=0}^{T-1}\alpha_{k|j}\,|\tilde{\lambda_k}\rangle\otimes|u_j\rangle\,. \tag{39}$$

To summarize the phase estimation:

$$\begin{aligned}|0\rangle_S|0^{\otimes t}\rangle_C|b\rangle_I &\xrightarrow{I_1\otimes H_t'\otimes I_n}|0\rangle\otimes\sqrt{\frac{2}{T}}\sum_{\tau=0}^{T-1}\sin\left(\frac{\pi(\tau+1/2)}{T}\right)|\tau\rangle\otimes|b\rangle \\ &\xrightarrow{I_1\otimes U_{t,n}'}|0\rangle\otimes\left[\sum_{k=0}^{T-1}|k\rangle\langle k|\otimes e^{iAkt_0/T}\right]\cdot \\ &\qquad\left[\sqrt{\frac{2}{T}}\sum_{\tau=0}^{T-1}\sin\left(\frac{\pi(\tau+1/2)}{T}\right)|\tau\rangle\otimes\sum_{j=1}^{N}\beta_j|u_j\rangle\right] \\ &=|0\rangle\otimes\sqrt{\frac{2}{T}}\sum_{j=1}^{N}\beta_j\left[\sum_{\tau=0}^{T-1}e^{i\lambda_j\tau t_0/T}\sin\left(\frac{\pi(\tau+1/2)}{T}\right)|\tau\rangle\right]\otimes|u_j\rangle \\ &\xrightarrow{I_1\otimes QFT_t^\dagger\otimes I_n}|0\rangle\otimes\sum_{j=1}^{N}\beta_j\left[\sum_{k=0}^{T-1}\frac{\sqrt{2}}{T}\sum_{\tau=0}^{T-1}\right. \\ &\qquad\left.e^{i\lambda_j\tau t_0/T}e^{-2\pi ik\tau/T}\sin\left(\frac{\pi(\tau+1/2)}{T}\right)|k\rangle\right]\otimes|u_j\rangle \\ &=|0\rangle\otimes\sum_{j=1}^{N}\beta_j\sum_{k=0}^{T-1}\alpha_{k|j}\,|\tilde{\lambda_k}\rangle\otimes|u_j\rangle\,.\end{aligned} \tag{40}$$

3.2.3 Controlled rotation

After phase estimation, information about λ is stored in the quantum state in register C. However, this information needs to be stored in the coefficients, not in the ket. To achieve this, controlled

rotation, as defined in Definition 2.7, is used with $f(\theta) = \arcsin(c/\tilde{\lambda_k})$ and $c = O(1/\kappa)$. The reason behind the choice for this f and θ is explained in Section 3.3.2. The controlled rotation results in:

$$\begin{aligned}|0\rangle \otimes \sum_{j=1}^{N} \beta_j \sum_{k=0}^{T-1} \alpha_{k|j} |\tilde{\lambda_k}\rangle \otimes |u_j\rangle &= \sum_{j=1}^{N} \beta_j \sum_{k=0}^{T-1} \alpha_{k|j} |0\rangle \otimes |\tilde{\lambda_k}\rangle \otimes |u_j\rangle \\ &\xrightarrow{R_{t,1} \otimes I_n} \sum_{j=1}^{N} \beta_j \sum_{k=0}^{T-1} \alpha_{k|j} \big(\cos(\tilde{\lambda_k}) |0\rangle + \sin(\tilde{\lambda_k}) |1\rangle\big) \otimes |\tilde{\lambda_k}\rangle \otimes |u_j\rangle \\ &= \sum_{j=1}^{N} \beta_j \sum_{k=0}^{T-1} \alpha_{k|j} \left(\sqrt{1 - \frac{c^2}{\tilde{\lambda_k}^2}} |0\rangle + \frac{c}{\tilde{\lambda_k}} |1\rangle\right) \otimes |\tilde{\lambda_k}\rangle \otimes |u_j\rangle \,.\end{aligned} \tag{41}$$

3.2.4 Uncompute

Even though the controlled rotation results in the desired states for qubit S and the qubits in register I, Example 2.2 illustrates how important it is that the qubits in register C are transformed back to the initial zero qubits. Therefore, the phase estimation needs to be uncomputed. The derivation of the uncompute step can be found in Appendix F. The uncompute step results in:

$$\begin{aligned}&\sum_{j=1}^{N} \left(\sqrt{1 - \frac{c^2}{\lambda_j^2}} |0\rangle + \frac{c}{\lambda_j} |1\rangle\right) \otimes |\Psi_0\rangle \otimes \beta_j |u_j\rangle \xrightarrow{I_1 \otimes H_t'^{\dagger} \otimes I_n} \\ &\sum_{j=1}^{N} \left(\sqrt{1 - \frac{c^2}{\lambda_j^2}} |0\rangle + \frac{c}{\lambda_j} |1\rangle\right) \otimes |0\rangle \otimes \beta_j |u_j\rangle \,.\end{aligned} \tag{42}$$

3.2.5 Measurement

The result of uncompute, Equation (152), can be written as:

$$\begin{aligned}&\sum_{j=1}^{N} \left(\sqrt{1 - \frac{c^2}{\lambda_j^2}} |0\rangle + \frac{c}{\lambda_j} |1\rangle\right) \otimes |0\rangle \otimes \beta_j |u_j\rangle = \\ &\left(\sum_{j=1}^{N} \beta_j \sqrt{1 - \frac{c^2}{\lambda_j^2}} |0\rangle |0\rangle |u_j\rangle\right) + \left(\sum_{j=1}^{N} \beta_j \frac{c}{\lambda_j} |1\rangle |0\rangle |u_j\rangle\right).\end{aligned} \tag{43}$$

Measurement of the first qubit would result in $|0\rangle$ with probability $\sum_{j=1}^{N} |\beta_j|^2 \left(1 - \frac{c^2}{\lambda_j^2}\right)$ and would result in $|1\rangle$ with probability $\sum_{j=1}^{N} |\beta_j|^2 \frac{c^2}{\lambda_j^2}$. More importantly, if the register that is measured has collapsed to $|1\rangle$, this results in state (up to normalization):

$$\sum_{j=1}^{N} \beta_j \frac{c}{\lambda_j} |1\rangle |0\rangle |u_j\rangle = c |1\rangle |0\rangle \sum_{j=1}^{N} \frac{\beta_j}{\lambda_j} |u_j\rangle \,. \tag{44}$$

Considering normalization, this would be:

$$c \sqrt{\frac{1}{c^2 \sum_{j=1}^{N} \beta_j^2 / \lambda_j^2}} |1\rangle |0\rangle \sum_{j=1}^{N} \frac{\beta_j}{\lambda_j} |u_j\rangle = \sqrt{\frac{1}{\sum_{j=1}^{N} \beta_j^2 / \lambda_j^2}} |1\rangle |0\rangle \sum_{j=1}^{N} \frac{\beta_j}{\lambda_j} |u_j\rangle \,. \tag{45}$$

3.3 Error analysis and complexity

This section analysis the error and derives the complexity of the HHL algorithm. The error in the HHL algorithm is induced by: error induced by the preparation of $|b\rangle$, imperfect gates (Hadamard-like gate, Hamiltonian simulation, QFT, and controlled rotation), phase estimation, and by post selecting.

This section follows the main ideas of [15], however, where the paper considers matrices with a well-conditioned part as well as an ill-conditioned part, this report only considers well-conditioned matrices.

3.3.1 Error analysis and complexity of preparation

Since the error of the preparation of state $|b\rangle$ highly depends on the way that state $|b\rangle$ is produced (as part of a larger algorithm, or as a standard state preparation procedure etc.), this error will be neglected for now. Section 9 gives more detail about this error when applying HHL to Mini-AES.

Furthermore, it is assumed that $|b\rangle$ can be prepared in logarithmic time, which is possible according to Lemma 3.1. This assumption is made to make sure that, also in the context of time, the preparation of $|b\rangle$ can be neglected in comparison with the rest of the HHL algorithm.

Lemma 3.1. *(qRAM loading [10]) If $b \in \mathbb{R}^N$, $||b||_2 = 1$, and b is stored in the quantum RAM, then $|b\rangle = \sum_{i=1}^{N} b_i |i\rangle$ can be prepared in time $\mathcal{O}(\log(N))$.*

3.3.2 Error analysis and complexity of quantum gates

As Section 3.2.2 shows, phase estimation consists of a Hadamard-like gate, Hamiltonian simulation and a Quantum Fourier Transform.

Phase estimation starts with the Hadamard-like gate to prepare state $|\Psi_0\rangle$. According to [15], this state state can be prepared up to error ϵ_Ψ in time $T_\Psi = \mathcal{O}(\text{poly}(\log(T/\epsilon_\Psi)))$.

The second step of phase estimation is Hamiltonian simulation. The error and complexity of Hamiltonian simulation is dominated by the simulation of e^{iAt}. According to the literature [15], this can be performed up to error ϵ_H in time $T_H = \tilde{\mathcal{O}}(\log(N)s^2t_0)$, if A is s-sparse and $t \leq t_0$. Note that this is the only part of the algorithm that requires A to be sparse. Since there are other types of Hamiltonian, this requirement can be relaxed. Note furthermore that this running time determines what number of qubits ($t = \log(T)$) should be used in register C_t.

Phase estimation ends with reversed Quantum Fourier Transform, which can be done efficiently (the complexity is negligible to the complexity of phase estimation).

To conclude, phase estimation can be performed up to error $\mathcal{O}(\epsilon_\psi + \epsilon_H)$ and the running time is dominated by the running time of Hamiltonian simulation, which is $\mathcal{O}(\log(N)s^2t_0)$.

After the phase estimation, controlled rotation is applied. This can be performed efficiently (the complexity is negligible to the complexity of phase estimation) up to t bits of precision. However, since phase estimation also works with a precision of t bits, controlled rotation does not induce any extra error.

Since in the controlled rotation $f(\theta) = \arcsin(c/\tilde{\lambda_k})$ and $c = \mathcal{O}(1/\kappa)$ were chosen, it is important to zoom in on the reason for these choices. Since [15] did not give any argumentation, one can only guess. However, it does state that it considers all eigenvalues $\lambda \geq 1/\kappa'$ as the well-conditioned part

of the matrix and all other eigenvalues as the ill-conditioned part, for some $\kappa' > \kappa$. This κ' can be chosen by the reader to obtain the preferred ratio between error complexity and running time on one hand and the number of eigenvalues that will be computed on the other hand, for example $\kappa' = 2\kappa$. Since this section considers a well-conditioned matrix, all eigenvalues must fall in the well-conditioned part. Therefore, κ is used instead of κ' (note that for all eigenvalues $\lambda \geq 1/\kappa$).

Besides, controlled rotation results in a pure quantum state with amplitudes $\cos(f(\theta))$ and $\sin(f(\theta))$, it must hold that $\cos^2(f(\theta)) + \sin^2(f(\theta)) = 1$, $|\cos(f(\theta))| \leq 1$, and $|\sin(f(\theta))| \leq 1$. This obviously holds, regardless of f and θ. So, f and θ must be chosen such that $|\sin(f(\theta))| \leq 1$ and that the eigenvalues can be extracted from this value. Therefore, it makes sense to choose f and θ such that $\sin(f(\theta))$ scales the eigenvalues to a value between zero and one. However, the eigenvalues as well as the range in which the eigenvalues fall are not known, which makes it hard to scale them to the desired range. Here κ' comes in handy, since this value can be chosen freely. Since $\kappa' > \kappa$ and $\lambda \geq 1/\kappa$ for the well-conditioned part of the matrix:

$$\left|\frac{1}{\lambda} \cdot \frac{1}{\kappa'}\right| \leq \left|\frac{1}{\lambda} \cdot \frac{1}{\kappa}\right| = \left|\frac{1}{\lambda} \cdot \frac{\lambda_{\min}}{\lambda_{\max}}\right| \leq \left|\frac{1}{\lambda_{\min}} \cdot \frac{\lambda_{\min}}{\lambda_{\max}}\right| \leq 1, \tag{46}$$

since without loss of generality is assumed that $|\lambda_1| = 1$. This means that choosing $f(\theta) = \arcsin(c/\tilde{\lambda_k})$ and $c = \mathcal{O}(1/\kappa)$ is a smart way of making sure that the controlled rotation results in a well-defined pure quantum state with amplitudes from which the eigenvalues can be determined, making use of the desired κ' instead of the unknown κ. Note that [15] considers a hypothetical case. Therefore, it is possible to write κ instead of κ', just as it is possible to write $|b\rangle = \sum_{i=1}^{N} b_i |i\rangle = \sum_{j=1}^{N} \beta_j |u_j\rangle$, even though the eigenvectors are not known.

The last part of the HHL algorithm is the uncompute. Since the uncompute part of the algorithm does not introduce any new transformations, it does not influence the error of the complexity or the analysis.

3.3.3 Error analysis and complexity of phase estimation

Let HHL be the desired output of the algorithm (without any error), HHL' be the real output algorithm, and $\widetilde{HHL}$ be the output of the algorithm in which everything is perfect except for phase estimation. The error induced by the phase estimation equals the difference between HHL and $\widetilde{HHL}$. This section is devoted to prove the upper bound for $||\widetilde{HHL} - HHL||_2$ that is given in Lemma 3.2.

Lemma 3.2. *The error of the variant of the HHL algorithm in which everything works perfect except for the phase estimation is bounded by* $||\widetilde{HHL} - HHL||_2 \leq \mathcal{O}(\kappa/t_0)$.

In order to prove Lemma 3.2, Lemmas 3.3 and 3.4 are needed.

Lemma 3.3. *For well-conditioned matrices, the auxiliary qubit ends up in the state:*

$$|h(\lambda)\rangle = \sqrt{1 - \frac{c^2}{\lambda^2}} |0\rangle + \frac{c}{\lambda} |1\rangle . \tag{47}$$

Then the map $\lambda \mapsto |h(\lambda)\rangle$ is $\mathcal{O}(\kappa)$-Lipschitz continuous, i.e.:

$$|| \, |h(\lambda_1)\rangle - |h(\lambda_2)\rangle \, ||_2 \leq C\kappa|\lambda_1 - \lambda_2|, \tag{48}$$

for some constant C.

Proof. It is sufficient to prove $|\frac{\partial h}{\partial \lambda}|^2 \leq (C\kappa)^2$ for all eigenvalues λ:

$$\begin{aligned}
\left|\frac{\partial h}{\partial \lambda}\right|^2 &= \left| -\frac{c^2}{\lambda^3\sqrt{1-\frac{c^2}{\lambda^2}}} - \frac{c}{\lambda^2}\right| \\
&= \left|\frac{2}{\lambda}\right|^2 \left|\frac{c^2}{\lambda\sqrt{1-\frac{c^2}{\lambda^2}}} + c\right| \\
&\leq \kappa^2 \left|\frac{c^2}{\lambda\sqrt{1-\frac{c^2}{\lambda^2}}} + c\right| \\
&= \kappa^2 \left(\frac{c^2}{\sqrt{\lambda^2 - c^2}} + c\right) \\
&\leq \kappa^2 \left(\frac{c^2}{\sqrt{\lambda_{\min}^2 - c^2}} + c\right) = (C\kappa)^2.
\end{aligned} \tag{49}$$

□

Remark. Note that [15] and [10] use filter functions to invert only the well-conditioned part of the matrix, which leads to a different $h(\lambda)$. This results in a slightly different version of Lemma 3.3 and the corresponding proof.

Lemma 3.4. *$|\alpha_{k|j}|$ is only large when $k \approx \frac{\lambda_j t_0}{2\pi}$, assuming $2\pi \leq \delta \leq T/10$, where $\delta = \lambda_j t_0 - 2\pi k$.*

The proof can be found in Appendix G.

Now Lemma 3.2 can be proved.

Proof. Define $\tilde{P}$ the phase estimation part of the HHL algorithm. Using Equation (39), adding the auxiliary qubit S_1, and undoing the relabeling of $|k\rangle$ it can be seen that $\tilde{P}$ maps:

$$|in\rangle |0\rangle^{\otimes t} |b\rangle \mapsto \sum_{j=1}^{N} \sum_{k=0}^{T-1} \beta_j \alpha_{k|j} |in\rangle |k\rangle |u_j\rangle , \tag{50}$$

where *in* denotes the initial value of the auxiliary qubit.

Using Equation (29), this defines:

$$\tilde{P} = \sum_{j=1}^{N} \sum_{k=0}^{T-1} \alpha_{k|j} |in\rangle \langle in| \otimes |k\rangle \langle k| \otimes |u_j\rangle \langle u_j| . \tag{51}$$

Remark. Note that this definition of $\tilde{P}$ differs from the definition given in [15], but it is believed that the definition given there is not correct. The first mistake with this definition is the dimension, the computations do not work for this dimension. This problem can be fixed by replacing $|k\rangle$ by $|k\rangle\langle k|$. Furthermore, their definition uses $|\text{garbage}(j,k)\rangle\langle in|$ instead of $|in\rangle\langle in|$. This report uses $\sum_{j=1}^{N}\sum_{k=0}^{T-1} |in\rangle\langle in|\, |in\rangle = \sum_{j=1}^{N}\sum_{k=0}^{T-1} |in\rangle\, I = \sum_{j=1}^{N}\sum_{k=0}^{T-1} |in\rangle$. Using the definition of [15], this would claim $\sum_{j=1}^{N}\sum_{k=0}^{T-1} |\text{garbage}(j,k)\rangle\langle in|\, |in\rangle = \sum_{j=1}^{N}\sum_{k=0}^{T-1} |\text{garbage}(j,k)\rangle\langle I| = \sum_{j=1}^{N}\sum_{k=0}^{T-1} |in\rangle$.

Since the HHL algorithm results in uncomputing the phase estimation of Equation (41), the output can be written as:

$$|\tilde{\varphi}\rangle = \tilde{P}^{\dagger} \sum_{j=1}^{N} \sum_{k=0}^{T-1} \beta_j \alpha_{k|j} \, |h(\tilde{\lambda_k})\rangle \, |k\rangle \, |u_j\rangle \,, \tag{52}$$

where the relabeling of $|k\rangle$ is again undone and $|h(\lambda_k)\rangle$ is defined in Equation (47). However, HHL would ideally result in:

$$|\varphi\rangle = \sum_{j=1}^{N} |h(\lambda_j)\rangle \otimes |0\rangle \otimes \beta_j \, |u_j\rangle \,. \tag{53}$$

The difference between $\widetilde{HHL}$ and HHL now boils down to the difference between $|\tilde{\varphi}\rangle$ and $|\varphi\rangle$. Hence, the proof consists of showing that $|| \, |\tilde{\varphi}\rangle - |\varphi\rangle \, ||_2 = \mathcal{O}\Big(\frac{\kappa}{t_0}\Big)$, for the norm as defined in Definition 2.1. For $\langle\tilde{\varphi}|\varphi\rangle = \langle\tilde{P}\tilde{\varphi}|\tilde{P}\varphi\rangle$, the next step is to compute the real part of the inner product $\langle\tilde{\varphi}|\varphi\rangle$. Use Equations (51), (52), and, (53) to see:

$$\tilde{P}\tilde{\varphi} = \sum_{j=1}^{N} \sum_{k=0}^{T-1} \beta_j \alpha_{k|j} \, |h(\tilde{\lambda_k})\rangle \, |k\rangle \, |u_j\rangle \tag{54a}$$

$$\begin{aligned} \tilde{P}\varphi = & \left[\sum_{j=1}^{N} \sum_{k=0}^{T-1} \alpha_{k|j} \, |in\rangle \langle in| \otimes |k\rangle \langle k| \otimes |u_j\rangle \langle u_j| \right] \cdot \\ & \left[\sum_{j=1}^{N} |h(\lambda_j)\rangle \otimes |0\rangle \otimes \beta_j \, |u_j\rangle \right] \\ = & \sum_{j=1}^{N} \sum_{k=0}^{T-1} \beta_j \alpha_{k|j} \, |h(\lambda_j)\rangle \otimes |k\rangle \otimes |u_j\rangle \,. \end{aligned} \tag{54b}$$

So, the inner product equals:

$$\begin{aligned} \langle \tilde{P}\tilde{\varphi} | \tilde{P}\varphi \rangle = & \sum_{j=1}^{N} \sum_{k=0}^{T-1} |\beta_j|^2 \cdot |\alpha_{k|j}|^2 \, \langle h(\tilde{\lambda_k}) | h(\lambda_j) \rangle \cdot \langle k|k\rangle \cdot \langle u_j|u_j\rangle \\ = & \sum_{j=1}^{N} \sum_{k=0}^{T-1} |\beta_j|^2 \cdot |\alpha_{k|j}|^2 \, \langle h(\tilde{\lambda_k}) | h(\lambda_j) \rangle \,. \end{aligned} \tag{55}$$

Remember that only the real part of the inner product is important:

$$\begin{aligned} Re \, \langle \tilde{P}\tilde{\varphi} | \tilde{P}\varphi \rangle = & Re \left(\sum_{j=1}^{N} \sum_{k=0}^{T-1} |\beta_j|^2 \cdot |\alpha_{k|j}|^2 \cdot \langle h(\tilde{\lambda_k}) | h(\lambda_j) \rangle \right) \\ = & \sum_{j=1}^{N} \sum_{k=0}^{T-1} |\beta_j|^2 \cdot |\alpha_{k|j}|^2 \cdot Re \, \langle h(\tilde{\lambda_k}) | h(\lambda_j) \rangle \,. \end{aligned} \tag{56}$$

According to Lemma 3.3:

$$|| \, |h(\lambda_1)\rangle - |h(\lambda_2)\rangle \, ||_2 \leq C\kappa |\lambda_1 - \lambda_2|, \tag{57}$$

for some constant C. By using Definition 2.1:

$$\begin{aligned}\sqrt{2(1-Re\,\langle h(\tilde{\lambda_k})|h(\lambda_j)\rangle)} \leq & C\kappa|\tilde{\lambda_k}-\lambda_j| \Rightarrow \\ 2(1-Re\,\langle h(\tilde{\lambda_k})|h(\lambda_j)\rangle) \leq & C^2\kappa^2|\tilde{\lambda_k}-\lambda_j|^2 \Rightarrow \\ Re\,\langle h(\tilde{\lambda_k})|h(\lambda_j)\rangle \geq & 1-\frac{C^2\kappa^2|\tilde{\lambda_k}-\lambda_j|^2}{2}.\end{aligned} \tag{58}$$

In the proof of Lemma 3.4 is already defined $\delta = \lambda_j t_0 - 2\pi k$. Using the relabeling of k to $\tilde{\lambda_k}$ this can be written as $\delta = t_0(\lambda_j - \tilde{\lambda_k})$. Substituting this leads to:

$$\begin{aligned}Re\,\langle h(\tilde{\lambda_k})|h(\lambda_j)\rangle \geq & 1-\frac{C^2\kappa^2|\tilde{\lambda_k}-\lambda_j|^2}{2} \\ = & 1-\frac{C^2\kappa^2\delta^2}{2t_0^2}.\end{aligned} \tag{59}$$

Substituting this in Equation (56) results in:

$$\begin{aligned}Re\,\langle \tilde{P}\tilde{\varphi}|\tilde{P}\varphi\rangle = & \sum_{j=1}^{N}\sum_{k=0}^{T-1}|\beta_j|^2\cdot|\alpha_{k|j}|^2\cdot Re\Big(\langle h(\tilde{\lambda_k})|h(\lambda_j)\rangle\Big) \\ \geq & \sum_{j=1}^{N}\sum_{k=0}^{T-1}|\beta_j|^2\cdot|\alpha_{k|j}|^2\cdot 1-\frac{C^2\kappa^2\delta^2}{2t_0^2} \\ \geq & \left(\sum_{j=1}^{N}\sum_{k=0}^{T-1}|\beta_j|^2\cdot|\alpha_{k|j}|^2\cdot 1\right)-\left(\sum_{j=1}^{N}\sum_{k=0}^{T-1}|\beta_j|^2\cdot|\alpha_{k|j}|^2\cdot\frac{C^2\kappa^2\delta^2}{2t_0^2}\right)\end{aligned} \tag{60}$$

Note that Equation (52) (among many others) shows that $\beta_j\alpha_{k|j}$ are the amplitudes in a quantum state, meaning that $\sum_{j=1}^{N}\sum_{k=0}^{T-1}|\beta_j|^2\cdot|\alpha_{k|j}|^2\cdot 1 = 1$. The remaining sum will be computed in two parts, for $|\delta| \leq 2\pi$, i.e. for $\frac{\lambda_j t_0}{2\pi} - 1 \leq k \leq \frac{\lambda_j t_0}{2\pi} + 1$, and for $|\delta| > 2\pi$, i.e. for $k < \frac{\lambda_j t_0}{2\pi} - 1$ or $k > \frac{\lambda_j t_0}{2\pi} + 1$. In the first part, 2π will be used as an upper bound of δ, in the second part Lemma 3.4 will be used for an upper bound of $|\alpha_{k|j}|^2 \leq \frac{144}{\delta^2}$. Note that Lemma 3.4 can be applied, since in this case $2\pi \leq \delta$ and T is sufficiently large such that $\delta \leq T/10$, so all assumptions of the Lemma are met.

$$\begin{aligned}\sum_{j=1}^{N}\sum_{k=0}^{T-1}|\beta_j|^2|\alpha_{k|j}|^2\frac{C^2\kappa^2\delta^2}{2t_0^2} = & \left(\sum_{j=1}^{N}\sum_{k=\frac{\lambda_j t_0}{2\pi}-1}^{\frac{\lambda_j t_0}{2\pi}+1}|\beta_j|^2|\alpha_{k|j}|^2\frac{C^2\kappa^2\delta^2}{2t_0^2}\right)+ \\ & \left(\sum_{j=1}^{N}\sum_{k=0}^{\frac{\lambda_j t_0}{2\pi}-1}|\beta_j|^2|\alpha_{k|j}|^2\frac{C^2\kappa^2\delta^2}{2t_0^2}\right)+ \\ & \left(\sum_{j=1}^{N}\sum_{k=\frac{\lambda_j t_0}{2\pi}+1}^{T-1}|\beta_j|^2|\alpha_{k|j}|^2\frac{C^2\kappa^2\delta^2}{2t_0^2}\right) \\ \leq & \left(\sum_{j=1}^{N}\sum_{k=\frac{\lambda_j t_0}{2\pi}-1}^{\frac{\lambda_j t_0}{2\pi}+1}|\beta_j|^2|\alpha_{k|j}|^2\frac{C^2\kappa^2\delta^2}{2t_0^2}\right)+\end{aligned} \tag{61}$$

$$
\begin{aligned}
&2\left(\sum_{j=1}^{N}\sum_{k=\frac{\lambda_j t_0}{2\pi}+1}^{\infty}|\beta_j|^2|\alpha_{k|j}|^2\frac{C^2\kappa^2\delta^2}{2t_0^2}\right)\\
\leq&\left(\sum_{j=1}^{N}\sum_{k=\frac{\lambda_j t_0}{2\pi}-1}^{\frac{\lambda_j t_0}{2\pi}+1}|\beta_j|^2|\alpha_{k|j}|^2\frac{C^2\kappa^2(2\pi)^2}{2t_0^2}\right)+\\
&2\left(\sum_{j=1}^{N}\sum_{k=\frac{\lambda_j t_0}{2\pi}+1}^{\infty}|\beta_j|^2\frac{144}{\delta^4}\frac{C^2\kappa^2\delta^2}{2t_0^2}\right)\\
=&\frac{2C^2\kappa^2\pi^2}{t_0^2}\left(\sum_{j=1}^{N}\sum_{k=\frac{\lambda_j t_0}{2\pi}-1}^{\frac{\lambda_j t_0}{2\pi}+1}|\beta_j|^2|\alpha_{k|j}|^2\right)+\\
&\frac{144C^2\kappa^2}{t_0^2}\left(\sum_{j=1}^{N}|\beta_j|^2\sum_{k=\frac{\lambda_j t_0}{2\pi}+1}^{\infty}\frac{1}{\delta^2}\right)\\
\leq&\frac{2C^2\kappa^2\pi^2}{t_0^2}+\frac{144C^2\kappa^2}{t_0^2}\left(\sum_{j=1}^{N}\sum_{k=\frac{\lambda_j t_0}{2\pi}+1}^{\infty}|\beta_j|^2\frac{1}{\delta^2}\right)\\
=&\frac{2C^2\kappa^2\pi^2}{t_0^2}+\frac{144C^2\kappa^2}{t_0^2}\left(\sum_{j=1}^{N}|\beta_j|^2\sum_{k=\frac{\lambda_j t_0}{2\pi}+1}^{\infty}\frac{1}{\left(\lambda_j t_0-2\pi k\right)^2}\right)\\
=&\frac{2C^2\kappa^2\pi^2}{t_0^2}+\frac{144C^2\kappa^2}{t_0^2}\left(\sum_{j=1}^{N}|\beta_j|^2\sum_{k'=1}^{\infty}\frac{1}{\left(2\pi k\right)^2}\right)\\
=&\frac{2C^2\kappa^2\pi^2}{t_0^2}+\frac{144C^2\kappa^2}{4\cdot 6t_0^2}\left(\sum_{j=1}^{N}|\beta_j|^2\right)\\
\leq&\frac{2C^2\kappa^2\pi^2}{t_0^2}+\frac{144C^2\kappa^2}{4\cdot 6t_0^2}\leq\frac{2C^2\kappa^2\pi^2}{t_0^2}+\frac{15C^2\kappa^2\pi^2}{4\cdot 6t_0^2}\\
=&\frac{2C^2\kappa^2\pi^2}{t_0^2}+\frac{\frac{5}{8}C^2\kappa^2\pi^2}{t_0^2}=\frac{21C^2\kappa^2\pi^2}{8t_0^2}.
\end{aligned}
$$

Substituting this back results in:

$$
\begin{aligned}
Re\left\langle\tilde{P}\tilde{\varphi}|\tilde{P}\varphi\right\rangle\geq&\left(\sum_{j=1}^{N}\sum_{k=0}^{T-1}|\beta_j|^2\cdot|\alpha_{k|j}|^2\cdot 1\right)-\left(\sum_{j=1}^{N}\sum_{k=0}^{T-1}|\beta_j|^2\cdot|\alpha_{k|j}|^2\cdot\frac{C^2\kappa^2\delta^2}{2t_0^2}\right)\\
\geq&1-\frac{21C^2\kappa^2\pi^2}{8t_0^2}
\end{aligned}
\tag{62}
$$

This concludes the proof, since:

$$
\begin{aligned}
||\left|\tilde{\varphi}\right\rangle-\left|\varphi\right\rangle||_2=&\sqrt{2(1-Re\left\langle\tilde{\varphi}|\varphi\right\rangle)}\\
=&\sqrt{2\Big(1-Re\left\langle\tilde{P}\tilde{\varphi}|\tilde{P}\varphi\right\rangle\Big)}
\end{aligned}
\tag{63}
$$

$$\leq \sqrt{2\left(1 - 1 + \frac{21C^2\kappa^2\pi^2}{8t_0^2}\right)}$$
$$\leq \sqrt{\frac{21C^2\kappa^2\pi^2}{4t_0^2}} = \sqrt{\frac{21}{4}}C\pi\frac{\kappa}{t_0} = \mathcal{O}\left(\frac{\kappa}{t_0}\right)$$

Remark. Note that a better upper bound is derived than the upper bound of $|| \, |\tilde{\varphi}\rangle - |\varphi\rangle \, ||_2 \leq 4C\pi\frac{\kappa}{t_0}$ in [15].

□

3.3.4 Error analysis and complexity of post-selection

According to Section 3.2.5, measurement leads to the desired output with probability $\sum_{j=1}^{N} |\beta_j|^2 \frac{c^2}{\lambda_j^2}$, i.e. it induces an error of:

$$\begin{aligned}\mathcal{O}\left(\sum_{j=1}^{N} |\beta_j|^2 \left(1 - \frac{c^2}{\lambda_j^2}\right)\right) &= \mathcal{O}\left(\sum_{j=1}^{N} |\beta_j|^2 - \sum_{j=1}^{N} |\beta_j|^2 \frac{c^2}{\lambda_j^2}\right) \\ &= \mathcal{O}\big(1 - c^2\big).\end{aligned} \tag{64}$$

Since $c = O(1/\kappa)$, the error induced by the post-selection step is $O(1/\kappa^2)$. Using amplitude amplification, this error can be made negligibly small when running the algorithm $O(\kappa)$ times, according to Theorem 2.1.

3.3.5 Total error and complexity

To determine the total error, an upper bound for $||HHL - HHL'||_2$ needs to be determined. For all norms it holds that $||HHL - HHL'||_2 \leq ||HHL' - \widetilde{HHL}||_2 + ||\widetilde{HHL} - HHL||_2$. Sections 3.3.2 and 3.3.1 derive $||HHL' - \widetilde{HHL}||_2 \leq \mathcal{O}(\epsilon_\psi + \epsilon_H)$ and according to Lemma 3.2 $||\widetilde{HHL} - HHL||_2 \leq \mathcal{O}(\kappa/t_0)$. Therefore, $||HHL - HHL'||_2 \leq \mathcal{O}(\epsilon_\psi + \epsilon_H + \kappa/t_0)$. As described in Section 3.3.2, the errors ϵ_ψ and ϵ_H can be made smaller than ϵ within a running time of $\mathcal{O}(\log(N)s^2t_0)$. To get an overall error of $\mathcal{O}(\epsilon)$, $t_0 = \mathcal{O}\big(\frac{\kappa}{\epsilon}\big)$ should be used. Substituting this in the running time results in $\mathcal{O}(\log(N)s^2\kappa/\epsilon)$. However, to deal with the error induced by the measurement, the algorithm should be run $O(\kappa)$ times. This means that the running time is

$$\mathcal{O}(\log(N)s^2\kappa^2/\epsilon), \tag{65}$$

as given in [15].

3.4 Improvements

The algorithm above is a basic version of the HHL algorithm. It makes assumptions that could be relaxed by making adjustments to the algorithm. Previously is already discussed that A only needs to be sparse for the method of Hamiltonian simulation that is used, but that there are other methods possible that do not need this assumption on A [10, Ch. 2]. Also the assumption that A needs to be Hermitian can be omitted, since a non-Hermitian matrix can be transformed to a Hermitian matrix using zero-blocks and the complex conjugate of the matrix [15, Ch. 2]. The assumption that A is well-conditioned can be relaxed [10, Ch. 3].

Besides, the algorithm can be adjusted to solve more specific problems, like solving boolean systems [2, Ch. 4].

4 AES

This project aims to determine the complexity of a known-plaintext attack on AES, using HHL. However, as Section 12 shows, the system of equations derived for AES, becomes, after linearization, far too big to be workable. In order to be able to investigate the effects of all ideas, techniques, and computations, these will first be applied to Mini-AES [7]. Mini-AES has a very similar structure to AES, but, as the name suggests, is smaller than AES (16 key bits and 2 rounds). Due to this similar structure, it is expected that attacks that work on Mini-AES also work for AES. So, with respect to this report's attack strategy for AES (creating a system of equations and then solving this system), Mini-AES is a suitable toy example. However, it is only fair to note that Mini-AES, in contrast to AES, is easily solvable by a brute-force attack on the key. Since Mini-AES uses a key of only 16 bits, there are only 65536 possible keys. For any plaintext-ciphertext pair, it takes Sage about five minutes to check if encrypting the plaintext with every possible key results in the desired ciphertext.

Before diving into attack ideas on any of the two encryption systems, both systems will be explained in full detail in this section. Due to the correspondence between the structure of the systems, this is not much more work than explaining only AES.

First, some parameters need to be set:

	Generic	Mini-AES	AES-128	AES-192	AES-256
Size of an element in bits	S_e	4	8	8	8
Number of words in a block	N_b	2	4	4	4
Number of elements in a word	N_e	2	4	4	4
Number of words in a key	N_k	2	4	6	8
Number of rounds	N_r	2	10	12	14
Block size in bits	B_s	16	128	128	128

Note that:

- (Mini-)AES works with blocks, this means that both the key (initial key and round key) and the states are called blocks. Blocks consist of N_b words, which consist of N_e elements, which consist of S_e bits.
- The number of elements in a word is equal to number of words in a block.
- The number of rounds is denoted by N_r, including one final round and $N_r - 1$ middle rounds. Note that the initial key addition is not counted as a round.
- $r = 0, ..., N_r$ is the round number: $r = N_r$ for the final round and $r = 1, ..., N_r - 1$ for the middle rounds. Define $r = 0$ for the initial key addition.

Define the following variables:

- w_i and x_i are blocks, they represent the round key and the state after addRoundKey of round i respectively. These are specified further using $w_i(j, m)$ and $x_i(j, m)$ to describe bit m of element j. A visualization of a state is given in Figure 3. This state belongs to Mini-AES, but can easily be extended to a state for AES. Besides, by replacing x by w, the figure is an example of a round key.
- All round keys combined are called the extended key and denoted by w.
- $\bar{w}_i$ and $\bar{x}_i$ are the results of eltSub working on w_i and x_i respectively.

- $rcon \in GF(2^{S_e})^{10 \times N_e}$ are the round constants. So, $rcon_r$ is the round constant for round r, where $r = 1, ..., N_r$. Note that depending on whether Mini-AES, AES-128, AES-192 or AES-256 is used, too many constants are defined, which means that not all $rcon$ values are used.

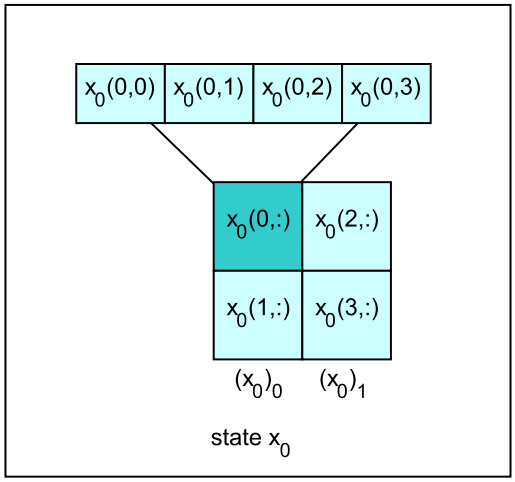


Figure 3: Example of a Mini-AES state

Note that:

- An element refers to a nibble when using Mini-AES and to a byte when using AES-128, AES-192, and AES-256.
- The encryption key $k \in GF(2^{S_e})^{N_k N_e}$ will be extended to the round keys $w_i(j, m)$. Note that in Mini-AES and in AES-128 the size of the key corresponds to the size of a state, whereas in AES-192 and in AES-256 the key is larger than a state. Therefore, different parameters are needed to describe the rounds in the key expansion for AES-192 and for AES-256 and the word 'round key' becomes a bit ambiguous: every round of encryption uses a round key and the round keys are created using the key expansion algorithm; in these contexts 'round' does not mean the same. In the encryption a round key means N_b words. In the key expansion a round key means N_k words. Therefore, other letters will be used to indicate these rounds. So, $w_i(j, m)$ for the encryption and $w_\imath(\jmath, m)$ for the key expansion, where $\imath = 0, ..., 10$ and $\jmath = 0, ..., N_k - 1$. Since $N_b(N_r + 1)$ words of round keys are needed for the encryption, and every round of the key expansion creates N_k new words, $\lceil N_b(N_r + 1)/N_k \rceil$ rounds of the key expansion are needed. Note that the key expansion algorithm now creates an extended key $w \in GF(2^{S_e})^{N_e \lceil N_b(N_r+1)/N_k \rceil N_k}$ even though only $w \in GF(2^{S_e})^{N_e N_b(N_r+1)}$ is needed. Therefore, the key expansions for AES-192 and AES-256 do not finish the last round, instead they stop as soon as enough words are created.
- $(w)_i$ is word i of the round key.
- Word $s(j : j + Nb, :) \in GF(2^{S_e})^{N_b}$ is word j of state s. Note that $j : j + Nb$ is used to denote entries $j, j + 1, ..., j + Nb$ and : is used to denote all entries.
- $(s)_{i,j}$ is element i, j of state s.

4.1 Algorithm overview

All algorithms that are needed for AES encryption can be found in Appendix B. First of all, the AES encryption algorithm is given in Algorithm 2. A visual representation of the Mini-AES encryption and of the AES encryption are given in Figure 4 and Figure 5 respectively. Both AES and Mini-AES are based on the same rounds, however, Mini-AES only has 1 middle round whereas AES performs the middle round multiple times.

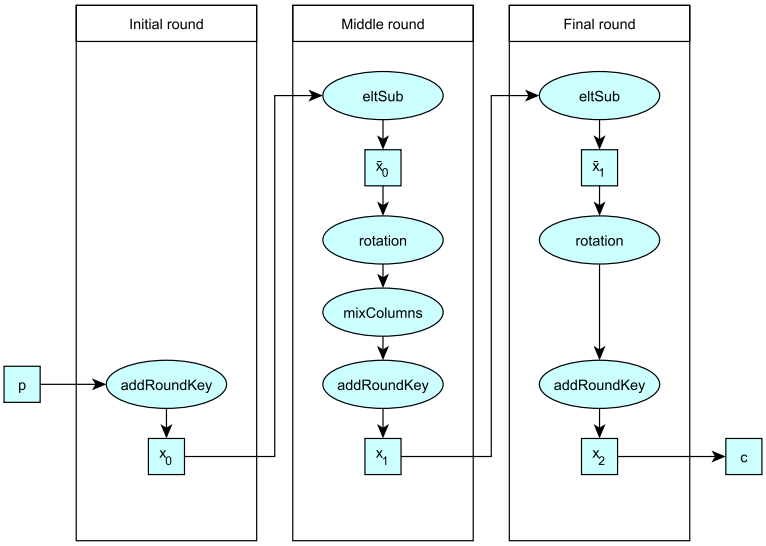


Figure 4: Mini-AES encryption

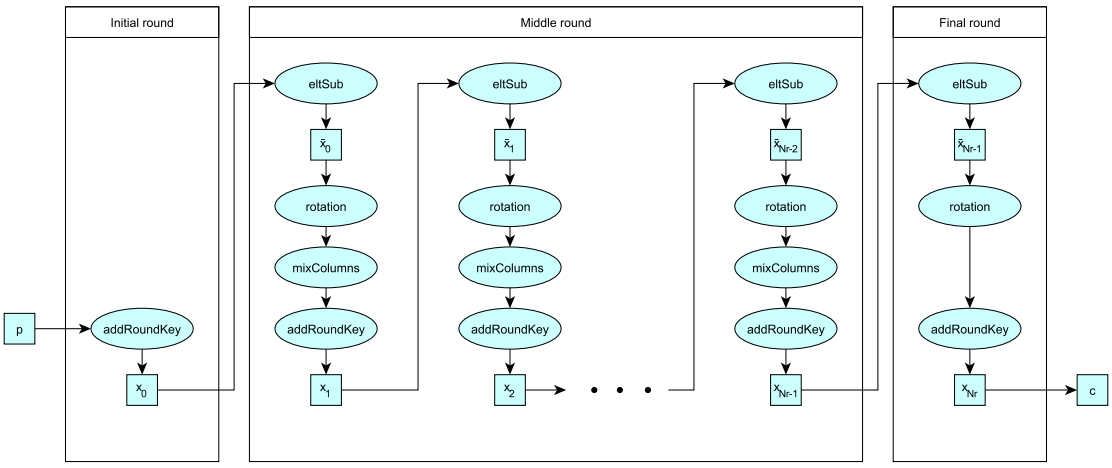


Figure 5: AES encryption

Both AES and Mini-AES work with an extended key (round keys). In the initial round of the encryption, only the round key is added. This is done by the linear function addRoundKey. In

addRoundKey, the new state is created by computing the element-wise XOR of the old state and the round key. A visual representation of addRoundKey can be found in Figure 6 and the corresponding algorithm can be found in Algorithm 7.

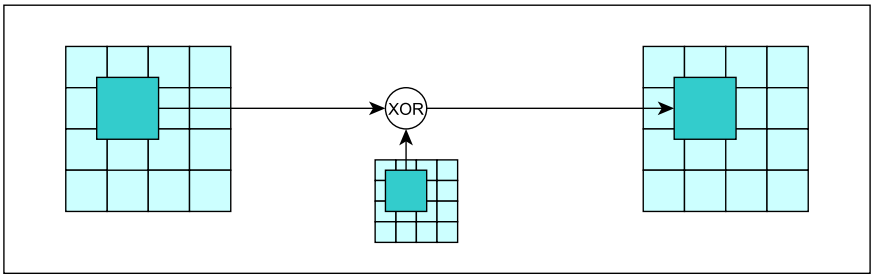


Figure 6: AES addRoundKey

The middle rounds of the encryption consist of applying four functions, always in the same order: first eltSub, then rotation, followed by mixColumns, and finally addRoundKey. The final round differs from the middle rounds; it omits the mixColumns function.

A visual representation of eltSub, rotation, and mixColumns can be found in Figure 7, Figure 8, and Figure 9 respectively. The corresponding algorithms can be found in Algorithm 8, Algorithm 6, and Algorithm 9 respectively. The eltSub algorithm uses a look-up table and the mixColumns algorithm uses a matrix. The look-up table and matrix can be found in Appendix A. The eltSub algorithm works element-wise. It replaces every element with the element it corresponds to in the S-box. This operation is non-linear, it is defined by a multiplicative inverse over GF(2^{S_e}) followed by an affine transformation over GF(2). The rotation algorithm is a linear transformation that works row-wise. For every row, it takes some of the elements on the left and inerts them on the right, shifting the other elements to the left. The number of elements that get rotated equals the row number. The mixColumns algorithm is described element-wise. It takes all the bits in an element and converts it to an element over $GF(2^{S_e})$. The columns of the new state are created by multiplying the old column by the mixColumnMatrix on the left. This results in elements of $GF(2^{S_e})$ which are then converted back to bits. However, as Lemma 5.1 states, these conversions are only a matter of notation, since all computations can also be done bit-wise. Therefore, the mixColumns algorithm can be considered a linear operation.

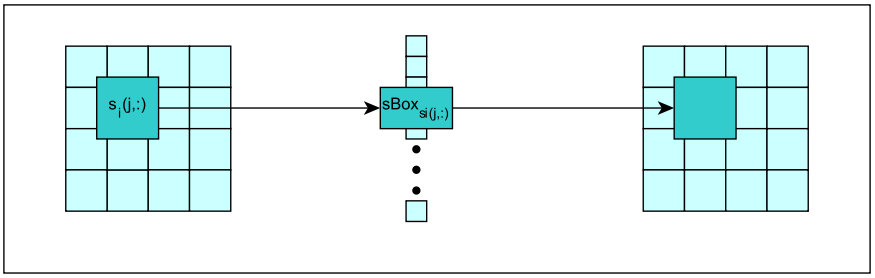


Figure 7: AES eltSub

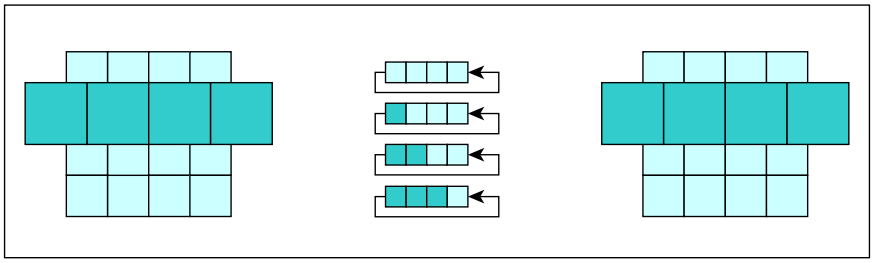


Figure 8: AES rotation

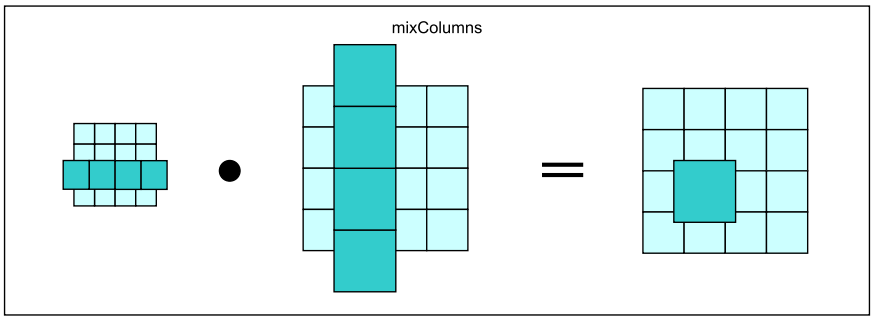


Figure 9: AES mixColumns

Since every round uses a round key, the AES encryption algorithm needs to start with determining the round constants, followed by creating the round keys. The key expansion algorithm is different for Mini-AES and for AES, since Mini-AES uses eltSub$((w)_{i-1})$, whereas AES uses eltSub(rotWord$((w)_{i-1})$). The rotWord algorithm is a linear transformation that works column-wise. It takes the element on the top and places it on the bottom, shifting all other elements one position up. A visualization of rotWord can be found in Figure 10 and the corresponding algorithm can be found in Algorithm 5. Note that in Mini-AES this eltSub works on a nibble and therefore rotWord is unnecessary.

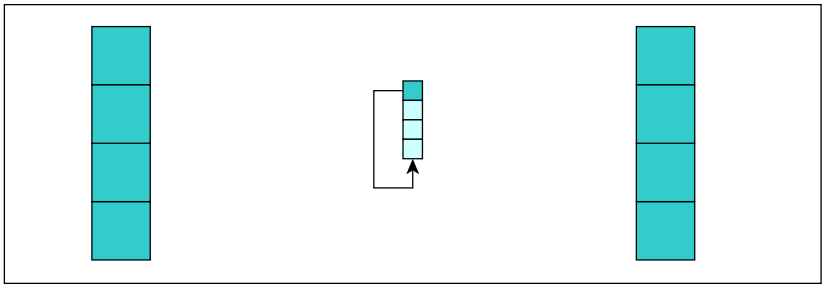


Figure 10: AES rotWord

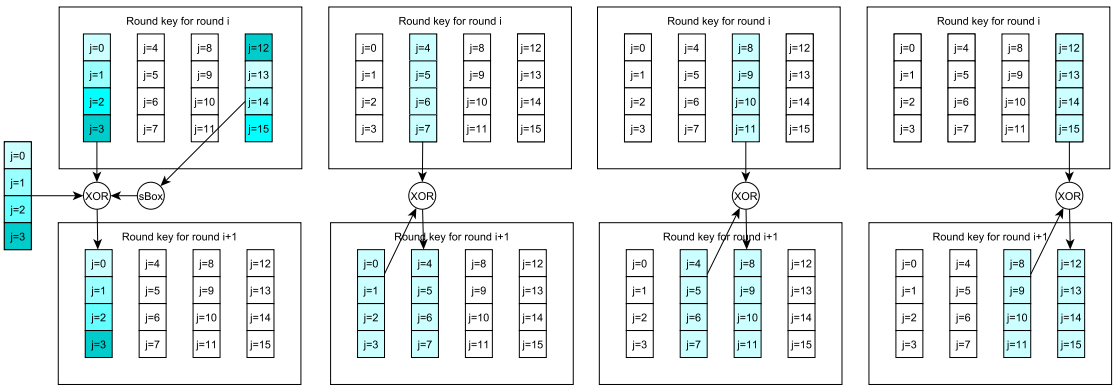


Figure 11: Key expansion AES-128 byte level

A visualization of the key expansion algorithm of Mini-AES and of AES-128, AES-192, and AES-256 can be found in Figure 12, Figure 11, Figure 13, and Figure 14 respectively. The corresponding algorithms can be found in Algorithm 4 and Algorithm 3 respectively.

In the key expansion of AES-128, the original key is exactly the same size as a state, which means it is exactly the size that we need for one round key. The first step is to copy the original key into the first round key. All other round keys are created based on the previous round key by the following computations. The first word of a new round key is created by computing the element-wise XOR of the first word of the previous round key, the round constant, and eltSub of the last word of the previous round key. However, before the eltSub is applied, the last word of the previous round key is rotated. This is done by the rotWord algorithm. All other words of the round key are created by computing the element-wise XOR of the corresponding word of the previous round key and the previous word of the same round key. A visual representation of the rotWord algorithm can be found in Figure 10 and the corresponding algorithm can be found in Algorithm 5.

As explained above, a Mini-AES state consists of two words, which each contain two elements. However, representing the Mini-AES round keys as four words, each containing just one element, underlines the similarities between the key expansion of AES and the key expansion of Mini-AES. The only difference between the two key expansions is that the key expansion of AES uses the rotWord algorithm and the key expansion of Mini-AES omits this operation. However, in the key expansion of Mini-AES, the rotWord algorithm is the identity, since a word consists of only one element. A visual representation of the key expansion of Mini-AES can be found in Figure 12 and the corresponding algorithm can be found in Algorithm 4.

The key expansions of AES-192 and AES-256 are similar to the key expansion of AES-128. The main difference is that the original key is now larger than one state. Therefore, the key expansions of AES-192 and AES-256 both work on states with more than four words. This means that the rounds of the key expansion and the rounds of the encryption are not synchronized anymore. One round of the key expansion creates more material than needed for one round of the encryption. Therefore, other variables for the number of rounds and the number of words are defined ($\imath$ and $\jmath$ respectively). By defining these notions of rounds and words, the key expansions of AES-192 and AES-256 essentially work the same as the key expansion of AES-128. The only difference is that in the key expansion for AES-256 an additional S-box is applied to the fourth word of the round key, when computing the fifth word of the round key. A visual representation of the key expansion of AES-192 and AES-256 can be found in Figure 13 and 14 and the corresponding algorithm is the same as the algorithm for the key expansion of AES-128, i.e. it can be found in Algorithm 3.

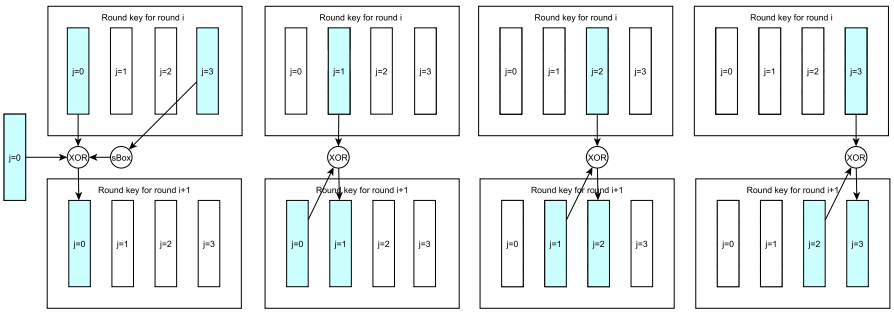


Figure 12: Key expansion Mini-AES nibble level

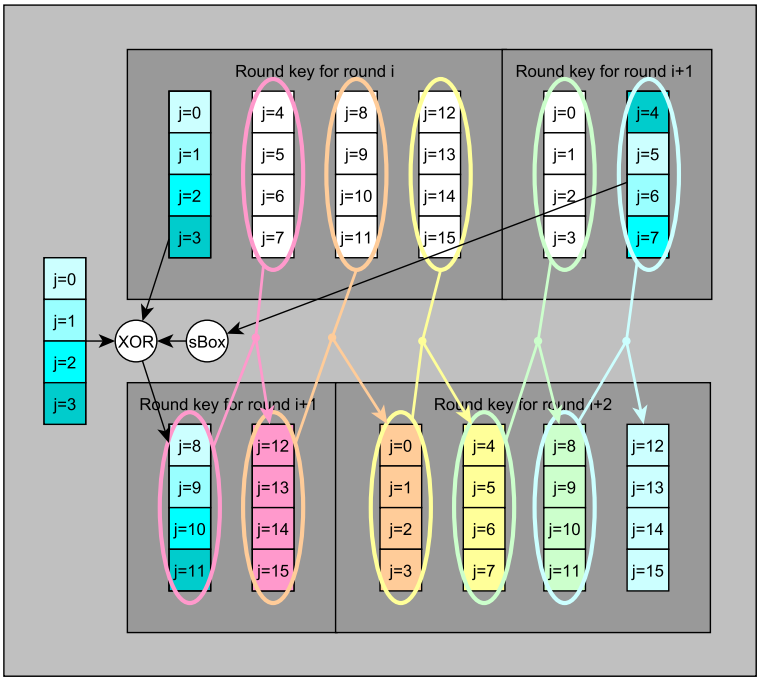


Figure 13: Key expansion AES-192 byte level

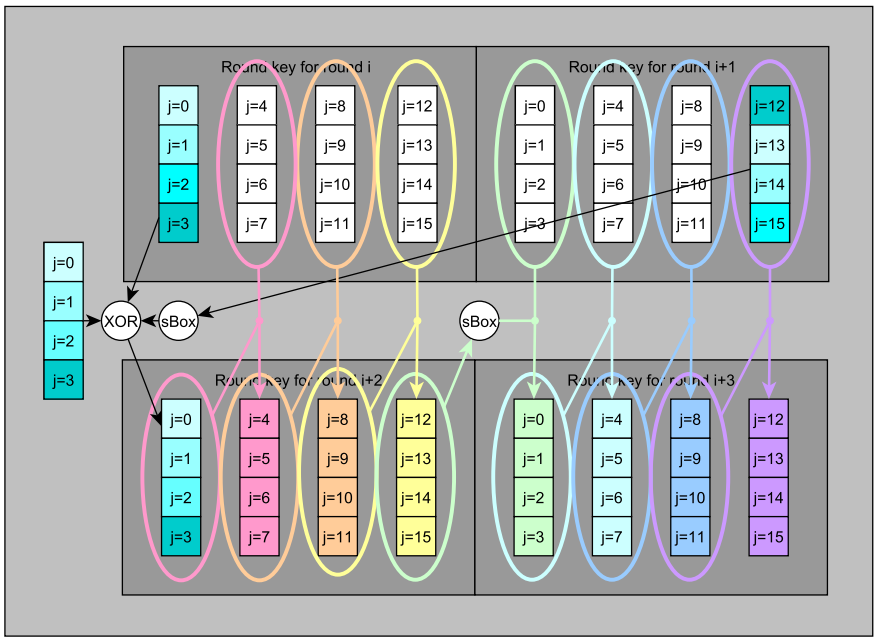


Figure 14: Key expansion AES-256 byte level

5 Mini-AES equations

This section derives a system of non-linear equations for a full Mini-AES encryption, to go from a key and a plaintext to a ciphertext, using Mini-AES encryption, which is given in Algorithm 2. At the end of this section, a list of bit-wise equations will be given, that fully describe the Mini-AES encryption. If a plaintext-ciphertext pair is inputted in these equations, then the solution of the resulting system results is a key that encrypts the chosen plaintext to the known ciphertext. Hence, solving the system of equations equals a successful chosen plaintext attack.

The mixColumns algorithm uses a matrix M. In this section this matrix equals M_0, as defined in Appendix A. However, the notation M is used, to underline the similarity with AES. If omitted: $j = 0, ..., N_b N_e - 1$, $m = 0, ..., S_e - 1$, and $i = 1, ..., N_r$. Hence, for Mini-AES: $j = 0, 1, 2, 3$, $m = 0, 1, 2, 3$, and $i = 1, 2$. To simplify notation, modulos are omitted, but indices work modulo $N_b N_e = 4$.

5.1 Encryption equations

The initial round only consists of the initial key addition:

$$x_0 = p \oplus w_0, \tag{66}$$

and can easily be written bit-wise:

$$x_0(j, m) = p(j, m) \oplus w_0(j, m). \tag{67}$$

The middle round computes:

$$x_i = \text{mixColumns}(\text{rotation}(\bar{x}_{i-1})) \oplus w_i. \tag{68}$$

In order to see what happens bit-wise, use the definitions of rotation and mixColumns:

$$\text{rotation}(\bar{x}_{i-1}) = \text{rotation}\left(\begin{pmatrix} \bar{x}_{i-1}(0,:) & \bar{x}_{i-1}(2,:) \\ \bar{x}_{i-1}(1,:) & \bar{x}_{i-1}(3,:) \end{pmatrix}\right) = \begin{pmatrix} \bar{x}_{i-1}(0,:) & \bar{x}_{i-1}(2,:) \\ \bar{x}_{i-1}(3,:) & \bar{x}_{i-1}(1,:) \end{pmatrix}, \tag{69}$$

and :

$$\begin{aligned} \text{mixColumns}(\text{rotation}(\bar{x}_{i-1})) &= M \cdot \text{rotation}(\bar{x}_{i-1}) \\ &= \begin{pmatrix} M(0,:) & M(2,:) \\ M(1,:) & M(3,:) \end{pmatrix} \cdot \begin{pmatrix} \bar{x}_{i-1}(0,:) & \bar{x}_{i-1}(2,:) \\ \bar{x}_{i-1}(3,:) & \bar{x}_{i-1}(1,:) \end{pmatrix}. \end{aligned} \tag{70}$$

When the round key is added, this results in:

$$x_i(j,:) = w_i(j,:) \oplus \sum_{j'=0}^{N_b-1} M(j_m + j' N_b, :) \bar{x}_{i-1}(j_p + j'(N_b + 1), :), \tag{71}$$

where $j_m = j \bmod N_b$ and $j_p = \lfloor j/N_b \rfloor N_b$. This can be written bit-wise by defining new notation $\sum_{j'} M(j', 0 : S_e) \bar{x}_{i-1}(j', 0 : S_e) = \sum_{j'} \sum_{m'=0}^{S_e-1} M(j', m') \bar{x}_{i-1}(j', m')$:

$$x_i(j, m) = w_i(j, m) \oplus \sum_{j'=0}^{N_b-1} \sum_{m'=0}^{S_e-1} M(j_m + j' N_b, m') \bar{x}_{i-1}(j_p + j'(N_b + 1), m'). \tag{72}$$

Lemma 5.1. *Multiplying an element by 2 or 3 can be done bitwise.*

Proof. Let $[b_0, b_1, b_2, b_3]$ be an element, where $b_0, b_1, b_2, b_3 \in \{0, 1\}$ are bits. Then:

$$\begin{aligned} 3 \cdot [b_0, b_1, b_2, b_3] &\equiv (x+1)(b_0x^3 + b_1x^2 + b_2x + b_3) \mod x^4 + x + 1 && \text{(73a)} \\ &\equiv b_0x^4 + (b_0 + b_1)x^3 + (b_1 + b_2)x^2 + (b_2 + b_3)x + b_3 + b_0(x^4 + x + 1) \\ &\equiv [b_0 + b_1, b_1 + b_2, b_0 + b_2 + b_3, b_0 + b_3], \\ 2 \cdot [b_0, b_1, b_2, b_3] &\equiv x(b_0x^3 + b_1x^2 + b_2x + b_3) \mod x^4 + x + 1 && \text{(73b)} \\ &\equiv b_0x^4 + b_1x^3 + b_2x^2 + b_3x + b_0(x^4 + x + 1) \\ &\equiv [b_1, b_2, b_0 + b_3, b_0]. \end{aligned}$$

□

The final round does not use the mixColumns, so for $i = N_r$:

$$c = x_{N_r} = \text{rotation } (\bar{x}_{N_r - 1}) \oplus w_{N_r}. \tag{74}$$

Using Equation (69), this can be written as:

$$c(j, :) = \bar{x}_{Nr-1}(j_r, :) \oplus w_{Nr}, \tag{75}$$

where $j_r = j + (j \bmod N_b)N_b$. This can be written bit-wise:

$$c(j, m) = \bar{x}_{Nr-1}(j_r, m) \oplus w_{Nr}. \tag{76}$$

Furthermore:

$$\bar{x}_{i-1} = \text{ eltSub } (x_{i-1}), \tag{77}$$

needs to be computed in the middle rounds and in the final round. So, compute Equation (77) for $i = 1, ..., N_r$. The equations needed to describe this step bit-wise are explained in Section 5.2.

5.2 sBox equations

Mini-AES uses a substitution box, called S-box, for the substitution step. This S-box can be found in Appendix A.

In order to describe the S-box as a function of the input and output bits, Karnaugh maps [5, Chapter 4] are used in which $[b_0, b_1, b_2, b_3]$ is the input and $[y_0, y_1, y_2, y_3]$ is the output, where $b_0, b_1, b_2, b_3, y_0, y_1, y_2, y_3 \in \{0, 1\}$ and are bits. A visualization of these Karnaugh maps can be found in Figure 16 in Appendix C. This results in:

$$\begin{aligned} y_0 =& b_3\bar{b}_1\bar{b}_0 + b_2\bar{b}_1b_0 + \bar{b}_3b_1\bar{b}_0 + \bar{b}_3\bar{b}_2b_1b_0 + b_3b_2b_1b_0 && \text{(78a)} \\ y_1 =& \bar{b}_3\bar{b}_1\bar{b}_0 + b_3\bar{b}_2\bar{b}_1 + \bar{b}_3b_2\bar{b}_1b_0 + b_3b_2b_1b_0 + \bar{b}_3b_2b_1\bar{b}_0 + b_3\bar{b}_2b_1\bar{b}_0 && \text{(78b)} \\ y_2 =& \bar{b}_3\bar{b}_2\bar{b}_0 + \bar{b}_3\bar{b}_1b_0 + b_3b_1b_0 + b_3\bar{b}_2b_1\bar{b}_0 + b_3b_2\bar{b}_1\bar{b}_0 && \text{(78c)} \\ y_3 =& \bar{b}_3\bar{b}_2\bar{b}_1 + b_2\bar{b}_1b_0 + b_3\bar{b}_2b_0 + \bar{b}_3b_2b_1, && \text{(78d)} \end{aligned}$$

which can be simplified by using $b' = b + 1$. Define the set of equations to describe the S-box:

$$\begin{aligned} S =& \{s_0, s_1, s_2, s_3\} && \text{(79)} \\ =& \begin{Bmatrix} 0 = y_0 + b_2b_3 + b_2b_1b_3 + b_3 + b_2 + b_2b_1 + b_2b_1b_0 + b_0 + 1, \\ 0 = y_1 + b_3b_1b_0 + b_3b_1 + b_3 + b_2b_0 + b_2 + b_1b_0 + 1, \\ 0 = y_2 + b_2b_3 + b_2b_1b_3 + b_1b_3 + b_2b_0b_3 + b_0b_3 + b_2b_1 + b_1 + b_2b_0 + b_1b_0 + b_0 + 1, \\ 0 = y_3 + b_3b_1b_0 + b_3b_0 + b_3 + b_2b_0 + b_1, \end{Bmatrix}. \end{aligned}$$

Doing more rewriting, the set S can even be written using only polynomials of degree two. Note that it is possible for any 4-bit to 4-bit S-box to be described by quadratic equations [4]. The method to create such equations for the AES S-box is described in [8]. For Mini-AES, this report extracted the quadratic equation that Sage uses for Mini-AES. These quadratic equations are given in Equation (142) in Appendix C.

As Section 5.1 explains, the S-box equations are used to compute Equation (77). So, add :

$$S(x_{i-1}(j,0),...,x_{i-1}(j,se-1),\bar{x}_{i-1}(j,0),...,\bar{x}_{i-1}(j,se-1)), \tag{80}$$

to the list of equations.

5.3 Key expansion equations

Mini-AES uses the addRoundKey algorithm. This algorithm needs round keys as input. These round keys are constructed using the key expansion algorithm. According to Algorithm 4, the key bits are first copied into the first roundkey:

$$w_0(j,:) \leftarrow k(j,:), \tag{81}$$

which can easily be written bit-wise:

$$w_0(j,:) = k(j,:). \tag{82}$$

However, this just creates unnecessary variables and equations. Therefore, this equation is omitted and the variables $w_0(j,m)$ are used instead of the variables $k(j,m)$.

The other roundkeys, $w_1(j,m),...,w_{N_r}(j,m)$, are computed as follows:

$$\begin{cases} w(j,:) = \text{eltSub}(w(j-1,:)) \oplus rcon_{j/N_b^2} \oplus w(j-N_b^2,:) & \text{if } j = 0 \mod N_b^2 \\ w(j,:) = w(j-1,:) \oplus w(j-N_b^2,:) & \text{for all other } j\text{'s.} \end{cases} \tag{83}$$

This means that if $j = 0 \mod N_b^2$:

$$w_i(0,:) = w_{i-1}(0,:) \oplus \bar{w}_{i-1}(N_b^2-1,:) \oplus rcon_i(0,:), \tag{84}$$

needs to be computed, where $\bar{w}_{i-1}(N_b^2-1,:)$ means, the eltSub of element N_b^2-1 of the roundkey of round $i-1$. Hence, Equation (79) needs to describe the substitution of $w_{i-1}(N_b^2-1,:)$. This means that Equation (79) needs to be added to the list of equations, with $w_{i-1}(N_b^2-1,:)$ as input and $\bar{w}_{i-1}(N_b^2-1,:)$ as output:

$$S(w_{i-1}(N_b^2-1,0),...,w_{i-1}(N_b^2-1,se-1),\bar{w}_{i-1}(N_b^2-1,0),...,\bar{w}_{i-1}(N_b^2-1,se-1)). \tag{85}$$

Besides, for the other $j's$:

$$w_i(j,:) = w_{i-1}(j,:) \oplus w_i(j-1,:), \tag{86}$$

needs to be computed, which can be written bit-wise:

$$w_i(j,m) = w_{i-1}(j,m) \oplus w_i(j-1,m). \tag{87}$$

This is the final equation that should be added to the list of equations.

The next subsection will collect all the equations derived above to create a system of equations.

5.4 System of equations for Mini-AES

If omitted: $j = 0, ..., N_b^2 - 1$, $m = 0, ..., se - 1$, and $i = 1, ..., N_r$. Hence, for Mini-AES: $j = 0, 1, 2, 3$, $m = 0, 1, 2, 3$, and $i = 1, 2$.

The previous sections derived a lot of equations. This section collects all equations needed to describe Mini-AES encryption and key expansion. This list consists of Equations (67), (72), (76), (84), (87), (80), and (85). Adding all the terms on the left-hand-side to both the left-hand-side and right-hand-side result in:

$$0 = x_0(j,m) \oplus p(j,m) \oplus w_0(j,m) \tag{88a}$$

$$0 = x_i(j,m) \oplus w_i(j,m) \oplus \sum_{j'=0}^{N_b-1} \sum_{m'=0}^{S_e-1} M(j_m + j'N_b, m') \bar{x}_{i-1}(j_p + j'(N_b+1), m') \quad \text{For } i = 1 \tag{88b}$$

$$0 = c(j,m) \oplus \bar{x}_{Nr-1}(j_r, m) \oplus w_{Nr} \tag{88c}$$

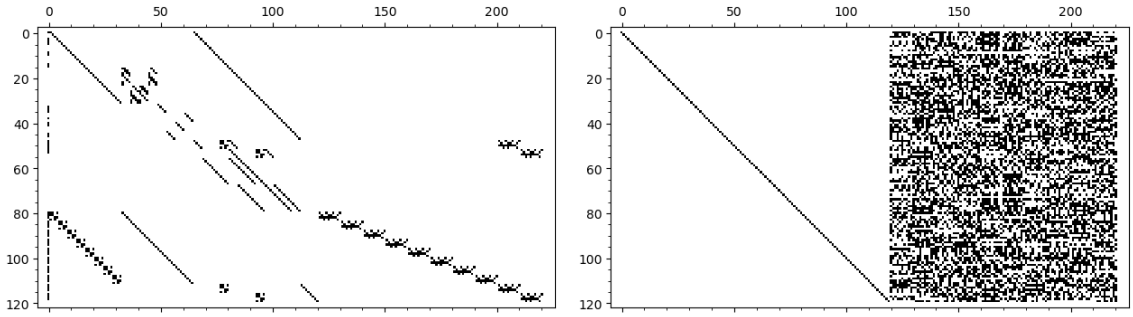
$$0 = w_i(0,m) \oplus w_{i-1}(0,m) \oplus \bar{w}_{i-1}(N_b^2 - 1, m) \oplus rcon_i(0,m) \tag{88d}$$

$$0 = w_i(j,m) \oplus w_{i-1}(j,m) \oplus w_i(j-1,m) \quad \text{For } j = 1, 2, 3 \tag{88e}$$

$$0 = S(x_{i-1}(j,0), ..., x_{i-1}(j, se-1), \bar{x}_{i-1}(j,0), ..., \bar{x}_{i-1}(j, se-1)) \tag{88f}$$

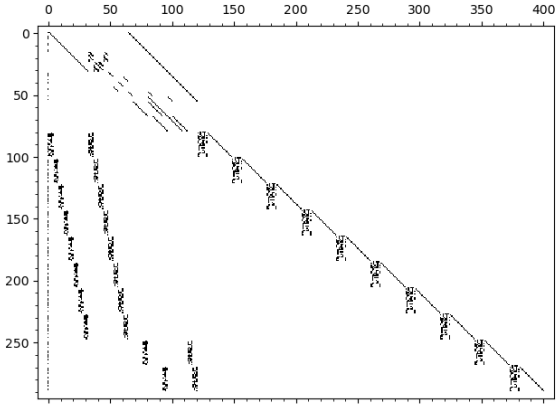
$$0 = S(w_{i-1}(N_b^2 - 1, se-1), ..., w_{i-1}(N_b^2 - 1, 0), \bar{w}_{i-1}(N_b^2 - 1, se-1), ..., \bar{w}_{i-1}(N_b^2 - 1, 0)) \tag{88g}$$

This system of equations can be linearized by defining a new variable for each non-linear term. Then, the system can be represented by the matrix in Figure 15. As an attempt to simplify this matrix, row reduction over GF(2) is applied. The result of this row reduction is included in the same figure. Note that both the degree two and degree three representation of the sBox are included in the figure, using Equation (142) and Equation (79) respectively. The degree two representation clearly results in more equations and more variables. The figure also shows that row reduction is not enough to find a solution of the system of equations, for neither representation of the S-box. The next section investigates whether it is possible to find a solution and, if so, what steps should be executed.

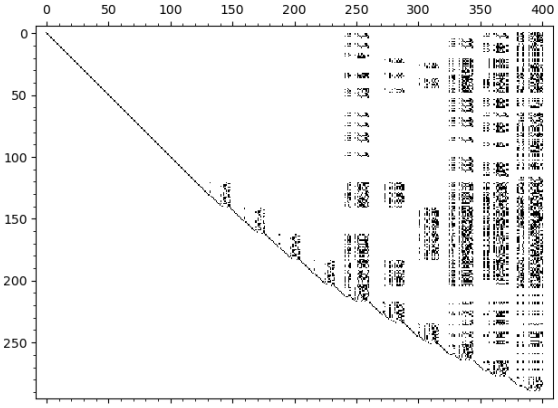


(a) Coefficient matrix, deg(sBox)=3

(b) Coefficient matrix row-reduced, deg(sBox)=3



(c) Coefficient matrix, deg(sBox)=2



(d) Coefficient matrix row-reduced, deg(sBox)=2

Figure 15: Mini-AES ($p = [9, 12, 6, 3], k = [12, 3, 15, 0]$)

6 Known plaintext attack for Mini-AES

The considered challenge is a known-plaintext attack on Mini-AES, as defined in Definition 6.1, using HHL.

Definition 6.1. *(Known-plaintext attack (KPA)) The attack model KPA assumes that the attacker knows as many plaintext-ciphertext pair as he needs and tries to gain information that reduces the security of the encryption scheme.*

To be able to use HHL for the known-plaintext attack, the system of equations, given in Equation (88), needs to be transformed into a well determined linear system $Ax = b$ over $\mathbb{C}$, since this is what HHL uses as input. This will be done in Section 8.

Before this can be done, a closer look needs to be taken at Equation (88). Section 6.1 first counts the number of equations and the number of variables for a known-plaintext attack to get an an idea of the size of this problem. Then it describes a method to linearize Equation (88) and concludes that this leads to an underdetermined linear system. Before trying to transform this system into a well determined linear system, the question whether this is possible will be answered in Section 6.2.

6.1 Size of the problem

Variables c and p are known, whereas all other variables are unknown. For the encryption this means that $x_0, ..., x_{N_r-1}$ and $\bar{x}_0, ..., \bar{x}_{N_r-1}$ are unknown, this gives $2 \cdot N_r \cdot N_b \cdot N_e \cdot S_e = 64$ variables. For the key expansion this means that $w_0, ..., w_{N_r}$ and $\bar{w}_0(S_e-1, :), ..., \bar{w}_{N_r-1}(S_e-1, :)$ are unknown, this gives another $((N_r + 1) \cdot N_b \cdot N_e + N_r) \cdot S_e = 56$ variables.

Now let's count the number of equations. Equations (88a), (88b), and (88c) are for the encryption. For each equation $j = 0, ..., N_eN_b - 1$ and $m = 0, ..., S_e - 1$, this gives another $N_e \cdot N_b \cdot S_e = 16$ equations per equation, i.e. 48 in total. Equations (88d) and (88e) describe the key expansion. In Equation (88d) it holds that $i = 1, ..., N_r$ and $m = 0, ..., S_e - 1$, this gives $N_r \cdot S_e = 8$ equations. In Equation (88e) $i = 1, ..., N_r$, $j = 1, 2, 3$, and $m = 0, ..., S_e - 1$. This gives another $N_r \cdot 3 \cdot S_e = 24$ equations. This means that $8 + 24 = 32$ equations are obtained for the key expansion.

Furthermore, Equations (88f) and (88g) describe the S-box equations for the encryption and the key expansion respectively. When computing these substitutions, nonlinear terms are introduced. In order to get a linear system, each nonlinear term is considered a new variable. For now, consider the degree three S-box polynomials as described in Equation (79). To compute one output bit of the substitution, one polynomial is used and four bits of input. Each S-box polynomial consists of $\sum_{i=2}^{S_e-1} \binom{S_e}{i} = \sum_{i=2}^{3} \binom{4}{i} = 10$ nonlinear terms (combinations of the four input bits). However, to compute all output bits of a nibble, the same ten nonlinear terms are used. So, a S-box polynomial introduces (at most) ten new variables, but, other polynomials to compute the other substituted bits in a nibble do not introduce new variables. Note that a different representation of the S-box may lead to a different number of variables and equations introduced by the S-box polynomials.

Equations (88f) and (88g) describe the S-box equations for the encryption and the key expansion respectively. For the encryption, $\bar{x}_0, ..., \bar{x}_{N_r-1}$ need to be computed, this gives $N_r \cdot N_b \cdot N_e \cdot S_e = 32$ equations, which leads to $10 \cdot N_r \cdot N_b \cdot N_e = 80$ new variables. For the key expansion this means $N_r \cdot 1 \cdot S_e = 8$ equations, which leads to $10 \cdot N_r \cdot 1 = 20$ new variables.

The above is summarized in Table 1. The most important thing to notice in this table is that there are 120 variables and equations, which means that the system of equations is well-determined. However, linearization introduces 100 more variables, resulting in 220 variables and only 120 equations. These extra variables come from the non-linear terms in the S-box equations. Therefore,

		Key expansion		Encryption		Total
		Linear	Non linear	Linear	Non linear	
S-box with degree three	Equations	32	8	48	32	120
	Variables	56	20	64	80	220
S-box with degree two	Equations	32	42	48	168	290
	Variables	56	56	64	224	400

Table 1: Mini-AES equations and variables

it might be convenient to choose another representation for the S-box. The number of non-linear terms decreases as the degree of the S-box equations decreases, which means that it might be most efficient to use S-box equations of degree two (note that linear equations are not possible without introducing more variables as the S-box operation is not linear). Even though [14] shows that a quadratic representation might not result in the most efficient attack, this report will use quadratic equations since [2] also does this for AES.

The quadratic representation of the Mini-AES S-box can be found in Equation (142), in Appendix C. It consists of 21 equations, with only quadratic terms, meaning that each equation will only introduce $7! = 28$ variables after linearization. For the key expansion this means $21 \cdot N_r \cdot N_b \cdot N_e = 168$ equations instead of 32, which leads to $28 \cdot N_r \cdot N_b \cdot N_e = 224$ new variables instead of 80. For the key expansion this means $21 \cdot N_r \cdot 1 = 42$ equations instead of 8 and $28 \cdot N_r \cdot 1 = 56$ new variables instead of 20.

These numbers can also be found in Table 1. From this table can be concluded that, when the degree two S-box equations are used, the number of equations is higher than the number of variables, before linearization. This means that not all equations are linearly independent. However, since the number of variables after linearization exceeds the number of equations, this does not really matter since it is clear that after linearization the system is under-determined. The next subsection will determine whether or not it is possible to transform this linear system into a well-determined linear system by answering the question whether the solution of this linear system is unique.

6.2 Unique solution

For both Mini-AES and AES, for a given key and plaintext, the ciphertext is uniquely defined. Besides, for a given key and ciphertext, the plaintext is uniquely defined. This is true because otherwise encryption and/or decryption would not be deterministic. However, the question at hand is the other way around: given a plaintext and a ciphertext, is the key uniquely defined?

Unfortunately, this is not the case for Mini-AES. For instance, one could encrypt plaintext $[1, 1, 1, 1]$ with keys $[0, 1, 2, 10]$ and $[0, 1, 3, 15]$ and find ciphertext $[3, 2, 2, 10]$ in both cases.

So, the answer to the question whether or not there is a unique solution to a chosen plaintext attack on AES, is generally: no. However, there will be a lot of plaintext-ciphertext pairs that have a unique key. To substantiate this claim, the probability that a certain plaintext-ciphertext pair has a unique key will be computed, as well as the expected number of keys.

Let T be the sets of all possible plaintexts and ciphertexts and let $K = \{k_1, ..., k_{|K|}\}$ be the set of all possible keys. Note that every plaintext, ciphertext and key consists of the same number of bits, hence $|T| = |K|$. Pick $p_1 \in T$ and let $c_1 \in T$ be the corresponding ciphertext, using without loss of generality key $k_1 \in K$. plaintext-ciphertext pair (p_1, c_1) has a unique key, when for all other keys in K hold that encrypting p_1 with this key does not result in c_1. Hence:

$$\begin{aligned}
&\mathbb{P}(\text{plaintext-ciphertext pair } (p_1, c_1) \text{ has a unique key}) \\
&= \mathbb{P}(\text{AES}(p_1, k_1) = c_1)\mathbb{P}(\text{AES}(p_1, k_2) \neq c_1)...\mathbb{P}(\text{AES}(p_1, k_{|K|}) \neq c_1) \\
&= \mathbb{P}(\text{AES}(p_1, k_1) = c_1)(1 - \mathbb{P}(\text{AES}(p_1, k_2) = c_1))^{|K|-1} \\
&= 1 \cdot \left(1 - \frac{1}{|T|}\right)^{|K|-1} = 1 \cdot \left(1 - \frac{1}{|K|}\right)^{|K|-1} \approx 0.4 \text{ for Mini-AES and AES-128}
\end{aligned} \tag{89}$$

It is clear that for each plaintext-ciphertext pair, the expected number of keys, that encrypt this specific plaintext to this specific ciphertext, is not very large.

To strengthen the result from these computations, a random sample of 100 plaintexts out of the 65536 possibilities was considered for Mini-AES. All these plaintexts were encrypted with all 65536 possible keys and it was counted how often the same ciphertext occurred. Suppose a ciphertext was found twice when encrypting the same plaintext, this means that there are two possible keys for this plaintext-ciphertext pair. So, the number of recurring ciphertexts per plaintext, equals the possible number of keys for the system. According to the sample, the minimum, mean, median, and maximal number of keys is respectively one, 1.580, one, and nine. Based on this, it is expected that around one in three plaintext-ciphertext pairs will have only one solution (key).

The fact that there might not be an unique key for a plaintext-ciphertext pair, could cause a problem. In this report's attack, a plaintext is considered, which is encrypted with a certain key. This results in the known ciphertext. The goal of the attack is to find the key that was used. However, since the plaintext-ciphertext pair does not uniquely determine the key that was used, it might be the case that another key is found, such that encryption of our plaintext leads to our ciphertext. This is a problem, because this is not the key that the other party used. The goal of the attack is to find the key that was used by the sender, because this has the practical applications: an attacker can decrypt more ciphertexts of the same sender (as long as the key is reused), the attacker can encrypt messages of his liking and impersonate the sender (as long as the receiver expects the sender to use the same key), etc. All of this is not possible if the attacker finds another key that coincidentally works for the known plaintext-ciphertext pair, so this defeats the purpose of a chosen plaintext attack.

However, the desired key is a solution of the system of equations. So, finding all the solutions of the system of equations would mean that the right key is among them. This means that if the number

of solutions is not too large, one could try all solutions (on a new plaintext-ciphertext pair) and see which key works. Note that the complexity of trying one of the keys is simply the complexity of one encryption with (Mini-)AES, which can be done efficiently. Above is determined that in the worst case, nine keys can work for a plaintext-ciphertext pair. This means that finding all possible keys for a certain plaintext-ciphertext pair will solve the problem, since it is no problem to check nine keys.

If the attack, on the other hand, does not find all possible keys, but just one, the problem can be solved by performing the attack multiple times. In a known plaintext attack, the attacker is assumed to have access to as many plaintext-ciphertext pairs as necessary. If the attacker has access to multiple plaintext-ciphertext pairs, that used the same key, he can perform the attack on all these pairs. As long as the result of each attack is a random pick out of the possible keys for this plaintext-ciphertext pair and the number of possible keys per plaintext-ciphertext pair is limited, the attacker will eventually find the desired key using this strategy.

This raises the idea that it would be beneficial to combine multiple plaintext-ciphertext pairs, with the same key, into one attack. However, this backfires for this report's attack, as explained in Appendix H.1.1.

The next section is devoted to reducing the solution space. Before doing so, the next subsection will describe how the solution space can be determined.

6.3 Determining solution space

The goal of this subsection is to create a strategy to determine whether applying HHL to a certain linearized system will result in a solution to the system of non-linear equations that the input for HHL is a linearization of. The strategy of this section will not be applied to the linearized system corresponding to (Mini-)AES, since this includes row-reduction and if row-reduction would be possible, the HHL algorithm would not be needed. The strategy is merely a way to evaluate the methods that are explained in later sections. These methods are applied to small examples and the strategy of this section is used to determine whether or not these methods result in solvable matrices. The hope is that this analysis can be extended for larger matrices, i.e. leads to an estimate of the size of the matrix vector system for (Mini-)AES to which HHL has to be applied.

First, a linear system will be considered. For a matrix vector system, corresponding to a linear system of equations, the number of solutions can be determined according to Theorem 6.1. The system of linear equations has a unique solution if and only if the matrix vector system has a unique solution.

The same line of reasoning can be extended to a linearized system. The difference between the two of them is that for the matrix corresponding to a system of linear equations, each column corresponds to an independent variable, whereas some columns of the matrix corresponding to a linearized system correspond to products of variables. Therefore, the linearized system $Ax = b$ might have more solutions than the system of non-linear equations. Example 6.1 clearly shows how this is possible.

Theorem 6.1. *(Number of solutions of a linear system) Let $Ax = b$ be a linear system over $\mathbb{C}$ and let A have dimensions $m \times n$.*

- *If rank$([A|b])$=rank$(A) = n$, then $Ax = b$ has a unique solution.*
- *If rank$([A|b])$ >rank(A), then $Ax = b$ has no solution.*
- *If rank$([A|b])$=rank$(A) < n$, then $Ax = b$ has multiple solutions.*

Example 6.1. *Consider the following system of non-linear equations over* $\mathbb{C}$*:*

$$\begin{cases} a & = 1 \\ b & = 1 \\ c & = 1 \\ ab - bc & = 0 \end{cases} \Rightarrow [A|b] = \left(\begin{array}{ccc|ccc|c} a & b & c & ab & ac & bc & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ \hline 0 & 0 & 0 & 1 & 0 & -1 & 0 \end{array}\right) \tag{90}$$

From the rows above the horizontal line can be concluded that $a = b = c = 1$*. This solves the system of non-linear equations and from this the values of* ab *and* bc *can also be determined. This results in the solution vector* $\dot{x} = (1, 1, 1, 1, 1, 1)^T$ *and indeed* $A\dot{x} = b$*. However,* $rank([A|b])$=$rank(A) = 4 < n = 6$ *which means according to Theorem 6.1 that there are multiple solutions. Indeed, for all vectors* $\dot{x} = (1, 1, 1, v_1, v_2, v_3)^T$*, where* $v_1 = v_2$*, it holds that* $A\dot{x} = b$*.*

From Example 6.1 can be concluded that Theorem 6.1 is not sufficient to determine whether a system of non-linear equations has a unique solution. Instead, the definition of a solvable matrix will be given in Definition 6.2 and Theorem 6.2 will tell when a matrix vector system is solvable.

Definition 6.2. *(Solvable matrix vector system) Let* $Ax = b$ *be a linear system, corresponding to a system of non-linear equations on* n' *variables and* m *equations, over* $\mathbb{C}$*. Without loss of generality, the first* n' *columns of* A *correspond to the linear terms. Then* A, x *and* b *can be written as:*

$$A = \left(\begin{array}{c|c} A_1 & A_2 \\ \hline A_3 & A_4 \end{array}\right), x = \left(\begin{array}{c} y \\ \hline z \end{array}\right) \text{ and } b = \left(\begin{array}{c} b_1 \\ \hline b_2 \end{array}\right), \tag{91}$$

where $A_1 \in \mathbb{C}^{n' \times n'}$*,* $y = (x_1, ..., x_{n'})^T$*,* $z = (x_{n'+1}, ..., x_n)^T$*, and* $b_1 \in \mathbb{C}^{n'}$*. The system* $Ax = b$ *is considered solvable if the system uniquely determines* y*. Note that* $z = (x_{n'+1}, ..., x_n)$ *can be uniquely determined if* $y = (x_1, ..., x_{n'})$ *is known.*

Note that this definition concludes that a matrix vector system is solvable if all linear variables can be uniquely determined. This is even more strict than needed for the matrix vector system for (Mini-)AES, since in this case only the key bits have to be uniquely determined. In fact, even if a small number of key bits can not be uniquely determined, these can be brute-forced. However, extending this idea to small examples in the following sections is quite vague. Therefore, this report will assume that all linear variables will have to be uniquely determined. Demanding that all variables need to be uniquely determined versus only the key bits will probably not lead to very different conclusions, because of how AES is defined: uniquely determined key bits (and a known plaintext-ciphertext) fixes all other variables.

Theorem 6.2. *(Number of solutions of a non-linear system) The system as defined in Definition 6.2, is solvable if and only if row-reduction of* $[A, b]$ *results in :*

$$[A', b'] = \left(\begin{array}{cc|c} A'_1 & A'_2 & b'_1 \\ A'_3 & A'_4 & b'_2 \end{array}\right), \tag{92}$$

where A'_1 *has rank* n' *and* $A'_2 = 0$*. Then the system of non-linear equations has a unique solution.*

Proof. Let row reduction of A result in:

$$[A', b'] = \left(\begin{array}{cc|c} A'_1 & A'_2 & b'_1 \\ A'_3 & A'_4 & b'_2 \end{array}\right), \tag{93}$$

Suppose A'_1 has rank n' and $A'_2 = 0$. Then rank($[A'_1|b'_1]$) =rank(A'_1) $= n' \leq m$. So according to Theorem 6.1 $A_1 y = b_1$ has a unique solution. This uniquely defines all variables $x_1, ..., x_{n'}$. So the system of non-linear equations has a unique solution.

Suppose A'_1 has rank $\neq n'$ (i.e. $< n'$). Then rank($[A'_1|b'_1]$) >rank(A_1) or rank($[A'_1|b'_1]$) =rank(A_1) < n'. According to Theorem 6.1, $A_1 y = b_1$ has no solution or multiple solutions respectively.

Suppose A'_1 has rank n' and $A'_2 \neq 0$. Then there is a linear variable or a combination of linear variables of which the value can only be determined, depending of the value of some non-linear term. However, the matrix vector system can not determine the value of this non-linear term (otherwise row-reduction would not be finished, since the non-zero entry in A'_2 could be removed by altering the corresponding value in b'_1 with the correct value of the non-linear term). This means that not all values of the linear terms can be extracted from the matrix vector system, i.e. y is not uniquely determined by the matrix vector system. Hence, this system is considered not solvable by Definition 6.2. □

According to Theorem 6.2, it can be determined whether a matrix vector system is solvable, based on the rank of the (sub)matrix. Therefore, the next step is to determine the rank of the matrix. This can be done by row-reduction of the matrix, therefore, a ring has to be determined over which row-reduction will take place.

Since HHL considers a complex matrix as input, the row-reduction should be done over the complex field. To underline this, Section 7 shows that even though the (Mini-)AES equations are defined over GF(2) row-reduction over GF(2) is not sufficient. This section also translates the equations from GF(2) to $\mathbb{C}$, resulting in a complex matrix. Row-reduction over $\mathbb{C}$ now has the desired effect. However, for row-reduction over $\mathbb{C}$ numerical methods are needed, using floating points, which might lead to floating point errors, whereas row-reduction over $\mathbb{Q}$ does not have this problem (it is precise, but works only as long as the numerators and denominators do not become so big that they lead to memory problems). Therefore, it is desired to use row-reduction over $\mathbb{Q}$ instead of over $\mathbb{C}$.

Translating equations from GF(2) to $\mathbb{C}$, using the method described in Section 7, results in a matrix over $\mathbb{C}$, but with entries that actually do not contain complex values (they can also be expressed in $\mathbb{Q}$). Therefore, Theorem 6.3 concludes that the row-reduction can be done over $\mathbb{Q}$ instead of over $\mathbb{C}$. This avoids floating point errors, especially for larger matrices. Therefore, the rest of this report will be using row-reduction over $\mathbb{Q}$.

Theorem 6.3. *Let A be a matrix with coefficients in $\mathbb{Q}$. Row-reduction of A over $\mathbb{C}$ results in the same matrix as row-reduction over $\mathbb{Q}$.*

Proof. The main idea of this proof is to show that for every step of the row-reduction over $\mathbb{C}$, the resulting matrix has only coefficients in $\mathbb{Q}$. From that it can be concluded that row-reduction over $\mathbb{Q}$ or over $\mathbb{C}$ results in the same matrix. Each step in row-reduction is one of the following operations:

1. Change order of rows.
2. Multiplying a row by a scalar.
3. Add a multiply of a row to another row.

The first option does not change the coefficients. For option two, the resulting matrix will have only coefficients in $\mathbb{Q}$, as long as one does not multiply by a scalar that is not in $\mathbb{Q}$, since $\mathbb{Q}$ is closed under multiplication. The goal of this step is to set the first non-zero entry of a row to one. Before this step the first non-zero entry was in $\mathbb{Q}$, hence one should multiply this row by a scalar in $\mathbb{Q}$. Therefore, after multiplication, all the coefficients in the row will still be in $\mathbb{Q}$. For option three similar reasoning can be used, since $\mathbb{Q}$ is also closed under addition. This step considers two rows, row i and row j. The goal of this step is to create a zero entry in row j and some

column k. Therefore, one searches for a scalar s, such that for entries (j,k) and (i,k) it holds that $(j,k) + s \cdot (i,k) = 0$. For $\mathbb{Q}$ is closed under addition and multiplication and both entries (j,k) and (i,k) are in $\mathbb{Q}$, the scalar s will also be in $\mathbb{Q}$. Therefore the entries $(j,l) + -s \cdot (i,l)$ for all other columns l will also be in $\mathbb{Q}$. □

From this subsection, a method can be extracted to check whether applying a linear system solver (such as HHL) to a certain matrix vector system will result in a solution for all the linear variables or not. To check whether a certain matrix will give the desired output, simply row-reduce over $\mathbb{Q}$. If A_1' has rank n' and $A_2' = 0$, then the linear system solver will give the desired output.

7 Transforming equations

The equations in Equation (88) are defined over GF(2) [2]. However, HHL takes a complex matrix as input. As Example 7.1 demonstrates, for a linear system that is defined over GF(2) , the corresponding matrix-vector system cannot directly be assumed to hold over $\mathbb{C}$. Therefore, Section 7.1 discusses how to transform a system over GF(2) to a system over $\mathbb{C}$. When transforming a system from GF(2) to $\mathbb{C}$, non-boolean solutions become possible. Section 8.1 describes how to handle this problem. Furthermore, depending on the approach used to transform a system from GF(2) to $\mathbb{C}$, the resulting system might not be sparse anymore, even if the original system was. Section 7.2 discusses how to deal with this.

Example 7.1. *Consider the following system of equations:*

$$\begin{cases} a+b+c & = 0 \mod 2 \\ b & = 1 \mod 2 \\ c & = 0 \mod 2 \\ a+c & = 1 \mod 2. \end{cases} \tag{94}$$

First transform this system of equations into a linear system $Ax = y$:

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \end{pmatrix}, x = \begin{pmatrix} a \\ b \\ c \end{pmatrix}, y = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}, \tag{95}$$

Consider $\dot{x} = (1, 1, 0)$ as a possible solution for this linear system. If this system is row-reduced over $\mathbb{C}$, the following might happen:

$$\left(\begin{array}{ccc|c} 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{array}\right) \sim \left(\begin{array}{ccc|c} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{array}\right) \tag{96}$$

This states that $a = -1$ and $a = 1$. This means that $\dot{x}$ is not a solution anymore, in fact, it means that there is no solution. This should never happen; if $\dot{x}$ is a solution before row-reduction then $\dot{x}$ should also be a solution after row-reduction. However, $\dot{x} = (1, 1, 0)$ is a solution to the system of equations over GF(2) and it is not a solution to the system of equations over $\mathbb{C}$. This means that if a system of equations is defined over GF(2) and transformed into a matrix-vector system, this matrix cannot be considered as a matrix over $\mathbb{C}$.

7.1 From GF(2) to $\mathbb{C}$

The mistake made in Example 7.1 is that the equations are defined over GF(2) and do not hold over $\mathbb{C}$. This means that the problem can be fixed by making sure that the equations do hold over $\mathbb{C}$. The next two subsections describe two different approaches to do so. The first approach is to transform the equations to a matrix first and then transform the matrix, whereas the second approach transforms the equations before they are transformed to a matrix.

7.1.1 Approach 1: matrix based

The first solution originates in the insight that if an equation equals zero modulo two, there is an integer k such that the equation minus $2k$ equals 0 over $\mathbb{C}$. The only downside is that this

number k is not known (assuming that the solution of the system is not known) and is therefore an additional variable in the system. Example 7.2 shows how this approach works.

Example 7.2. *System $Ax = y$ can be transformed to:*

$$(A|2I) = \left(\begin{array}{ccc|cccc} 1 & 1 & 1 & 2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 2 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 2 \end{array}\right), x' = \begin{pmatrix} a \\ b \\ c \\ k_1 \\ k_2 \\ k_3 \\ k_4 \end{pmatrix}, y = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}. \tag{97}$$

The solution $\dot{x}$ now corresponds to $\dot{x}' = (1, 1, 0, -1, 0, 0, 0)^T$, since only the first equation equals two for $\dot{x}$ and the other equations already equal 0 over $\mathbb{C}$. Row reduction now leads to:

$$\left(\begin{array}{ccc|cccc|c} 1 & 1 & 1 & 2 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 2 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 2 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 2 & 1 \end{array}\right) \sim \left(\begin{array}{ccc|cccc|c} 1 & 0 & 0 & 0 & 0 & -2 & 2 & 1 \\ 0 & 1 & 0 & 0 & 2 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & -1 & -1 \end{array}\right) \tag{98}$$

The solution $\dot{x}'$ is still a solution to this linear system. This means that this did solve the problem; it did transform the equations over GF(2) to equations over $\mathbb{C}$. However, it created as many new variables, in the process, as there were equations to start with.

Example 7.2 shows that this approach introduces as many new variables as there are equations. For the linear system of Mini-AES, this would mean that the system would never become well-determined; even if the system would be written with more independent equations that variables, adding a new variable per equation results in more variables than equations. Therefore, a different approach [2, Chapt 5] will be used to transform the equations over GF(2) to equations over $\mathbb{C}$.

Note that this matrix based approach is a useful approach in some other cases, for example the Coppersmith attack. In this case, the modulus is assumed to be big and the resulting problem is solved over the integers using LLL.

7.1.2 Approach 2: equation based

This approach rewrites all equations one by one: $f \mapsto C(f)$. It first distinguishes special cases for equations with at most three terms, after which it solves the general case. One could use the general approach for every case, but this leads to unnecessary equations and variables.

Let f be an equation and $\dot{x}$ a boolean solution, meaning that $f(\dot{x}) = 0 \mod 2$. Note that each term of f can be at most one, since each entry of $\dot{x}$ is boolean. Furthermore, if $f(0) = 0 \mod 2$ or $f(0) = 1 \mod 2$, then also $f(0) = 0$ or $f(0) = 1$ over $\mathbb{C}$ respectively.

If f has at most one term, then $f(\dot{x})$ also equals zero over $\mathbb{C}$. So, take $C(f) = f$.

If f has two terms, then it can be written as $f = a + b$. Since $f(\dot{x}) = 0 \mod 2$, either $a = b = 0$ or $a = b = 1$ in $\dot{x}$. In both cases $a - b = 0$ over $\mathbb{C}$. So, in this case $C(f) = a - b$.

If f has three terms and $f(0) = 1$, then it can be written as $f = a + b + 1$. This means that in $\dot{x}$ either $a = 0, b = 1$ or $a = 1, b = 0$ and in both cases $f(\dot{x}) = 2$. This means that it works to take $C(f) = f - 2$.

If f has three terms and $f(0) = 0$, then it can be written as $f = a + b + c$. It can be verified by trying the four possible solutions that it works to pick $C(f) = 2ab + 2ac + 2bc - f$.

When f has more than three terms, no such approach is known. However, if f has t terms, it is known that $f(\dot{x})$ is at most t and that one of the equations $f(\dot{x}), f(\dot{x})-2, f(\dot{x})-4, ..., f(\dot{x})-2\lfloor\frac{t}{2}\rfloor$ equals zero over $\mathbb{C}$. If one of these equations equals zero over $\mathbb{C}$, then the product of these equations equals zero over $\mathbb{C}$. So, in this general case, use $C(f) = \prod_{k=f(0)}^{\lfloor t/2\rfloor}(f-2k)$. Note that if $f(0)=1$, then there is no reason to add f to the product, therefore it is sufficient to start at $k=f(0)$.

Example 7.3. *Above, several different cases are discussed:*

$$\left.\begin{array}{ll} t=1 & \\ t=2 & f(0)=1 \\ t=2, & f(0)=0 \\ t=3, & f(0)=1 \\ t=3 & f(0)=0 \\ t=4 & f(0)=1 \\ t=4 & f(0)=0 \end{array}\right\} \Rightarrow \left.\begin{array}{l} f_1 = a \\ f_2 = a+1 \\ f_3 = a+b \\ f_4 = a+b+1 \\ f_5 = a+b+c \\ f_6 = a+b+c+1 \\ f_7 = a+b+c+d \end{array}\right\} \Rightarrow \begin{array}{l} C(f_1) = a \\ C(f_2) = a-1 \\ C(f_3) = a-b \\ C(f_4) = a+b-1 \\ C(f_5) = 2ab+2ac+2bc-f_5 \\ C(f_6) = (f_6-2)(f_6-4) \\ C(f_7) = f_7(f_7-2)(f_7-4). \end{array} \tag{99}$$

Clearly, for $i=1,...,7$, if $\dot{x}\in\{0,1\}^4$ is a solution of function f_i, i.e. $f_i(\dot{x}) \equiv 0 \mod 2$, then $\dot{x}$ is a solution of $D(f_i)$, i.e. $C(f_i(\dot{x}))=0$. Note that f's with $t\geq 4$ work just like f_6 or f_7.

By transforming the equations over GF(2) to $\mathbb{C}$ so systematically, it also becomes possible to compute the number of terms of the resulting equations, using the multinomial theorem [12, Chapt.4]. Let f have t terms and $f(0)=0$ and take $q=\lceil\frac{t}{2}\rceil+1$. Then the number of terms in $C(f)$ is:

$$\mathcal{T}(C(f)) = \binom{t+q}{t} - 1. \tag{100}$$

Let f have t terms and $f(0)=1$ and take $q=\lceil\frac{t}{2}\rceil$ (when not confusing, f is said to have $t+1$ terms). Then the number of terms in $C(f)$ is:

$$\mathcal{T}(C(f)) = \binom{t+q}{t}. \tag{101}$$

This approach can be used to transform all equations in Equation (88) to equations over $\mathbb{C}$, one by one. A lot of these equations only have two or three terms to start with, meaning that transforming them will not add any variables or equations. However, there are also some equations with more terms and transforming these will result in equations with a larger number of terms. This increased number of terms increases the complexity of the problem, since it decreases the sparsity and the complexity of HHL depends on this sparsity. Section 7.2 will elaborate this point.

7.2 Decrease sparsity

Equation (65) states that the complexity of HHL depends on the sparsity (s) of the input matrix, where s is defined in Definition 3.1.

By transforming the equations according to Section 7.1, the equations are not as sparse anymore. Therefore, this section discusses two approaches of creating a sparser matrix. The first approach is to rewrite every single equation, introducing more variables (u's), as done in [2]. In the second approach, the non-sparse matrix is created first. The main idea of this approach is that every time a row contains to many non-zero's, some of these non-zero's are moved to a new row and column (note the correspondence with the u variables).

Both approaches are very similar, in both cases new rows and columns are added every time a row contains too many non-zero's. The main difference is that in the first approach new variables are defined, whereas the second approach does not lead to more variables. The effect of this difference will become clear in Section 8.

7.2.1 Approach 1: equation based

Every equation can be made 3-sparse by introducing new equations and new variables. A commonly known way to do so, creates as many new equations as new variables. This means that the solution space of the linear system is not affected by this approach. This method is also described in [2, Chapt 5]. However, this reports makes some adjustment and includes an example to show that the approach described by [2, Chapt 5] is not correct and the method described in this report is correct.

Let $s > 2, s \in \mathbb{N}$ be the desired sparsity and $f = \sum_{k=1}^{t} m_k$ be a polynomial with more than s terms, i.e. $t > s$. Set $S_t = \lceil \frac{t-s}{s-2} \rceil$. The following method will construct polynomial set $S(f,s) = \{\hat{f}_1, ..., \hat{f}_{S_t+1}\}$ that has the same solution set as f and that consists of polynomials that are all s-sparse, i.e. polynomials that have (at most) s terms.

Define a set of new variables $\mathbb{U}_f = \{u_1, ..., u_{S_t}\}$ and define $S(f,s) = \{\hat{f}_1, ..., \hat{f}_{S_t+1}\}$, where:

$$\hat{f}_1 = -u_1 + \sum_{k=1}^{s-1} m_k \tag{102a}$$

$$\hat{f}_j = u_{j-1} - u_j + \sum_{k=(j-1)(s-2)+2}^{j(s-2)+1} m_k \qquad \text{for } j = 2, ..., S_t \tag{102b}$$

$$\hat{f}_{S_t+1} = u_{S_t} + \sum_{k=S_t(s-2)+2}^{t} m_k. \tag{102c}$$

It is easy to verify that each $\hat{f}$ in Equation (102) is s-sparse. Besides, the solution of f is also a solution to all $\hat{f}$'s. This can be shown by picking:

$$u_1 = \sum_{k=1}^{s-1} m_k \tag{103a}$$

$$u_j = \sum_{k=1}^{j(s-2)+1} m_k \qquad \text{for } j = 2, ..., S_t. \tag{103b}$$

For this results in:

$$\hat{f}_1 = -\sum_{k=1}^{s-1} m_k + \sum_{k=1}^{s-1} m_k = 0 \tag{104a}$$

$$\begin{aligned} \hat{f}_j &= \sum_{k=1}^{(j-1)(s-2)+1} m_k - \sum_{k=1}^{j(s-2)+1} m_k + \sum_{k=(j-1)(s-2)+2}^{j(s-2)+1} m_k \\ &= -\sum_{(j-1)(s-1)+2}^{j(s-2)+1} m_k + \sum_{k=(j-1)(s-2)+2}^{j(s-2)+1} m_k = 0 \qquad \text{for } j = 2, ..., S_t \end{aligned} \tag{104b}$$

$$\hat{f}_{S_t+1} = \sum_{k=1}^{S_t(s-2)+1} m_k + \sum_{k=S_t(s-2)+2}^{t} m_k \tag{104c}$$
$$= \sum_{k=1}^{\lceil \frac{t-s}{s-2} \rceil (s-2)+1} m_k + \sum_{k=\lceil \frac{t-s}{s-2} \rceil (s-2)+2}^{t} m_k$$
$$= \sum_{k=1}^{t-s+1} m_k + \sum_{k=t-s+2}^{t} m_k = \sum_{k=1}^{t} m_k = f.$$

This means that $\hat{f}_j = 0$ for $j = 1, ..., S_t$ and that $\hat{f}_{S_t+1} = 0$ if and only if $f = 0$. So, this method results in a set of polynomials that are each s-sparse and have the same solution set as f.

Remark. In [2, Chapt 5], similar $\hat{f}$'s are defined. However, in the paper, the approach is applied to equations over GF(2). Therefore, the minus signs are omitted. In Section 9, this approach is both applied to equations over GF(2) and to equations over $\mathbb{C}$.

Example 7.4 shows the differences between the method described in this section and the one described in [2, Chapt 5]. The example uses $s = 3$ since that is what [2, Chapt 5] prescribes for the attack on AES.

Example 7.4. *Let $s = 3$ and $f = a + b + c + d + e$, i.e. $t = 5$. Then $S_t = \lceil \frac{t-s}{s-2} \rceil = 2$ and $\mathbb{U}_f = \{u_1, u_2\}$.*

For the approach described in [2, Chapt 5]:

$$\hat{f}_1 = a + b + u_1 \tag{105a}$$
$$\hat{f}_2 = c + u_1 + u_2 \tag{105b}$$
$$\hat{f}_3 = d + e + u_2. \tag{105c}$$

If each $\hat{f}$ has to be zero, one can conclude from the first two equations that $u_1 = -(a + b)$ and $u_2 = -c - u_1 = a + b - c$. Substituting this in the final equation results in $d + e + a + b - c = 0$. Since f also has to be zero, one could also write $a + b + c + d + e = 0$, hence $d + e + a + b = -c$. Combining the two results in $-2c = 0$ and this can not be what the authors have meant.

For the method in this report:

$$\hat{f}_1 = a + b - u_1 \tag{106a}$$
$$\hat{f}_2 = c + u_1 - u_2 \tag{106b}$$
$$\hat{f}_3 = d + e + u_2. \tag{106c}$$

From the first two of these equations can be concluded that $u_1 = a + b$ and $u_2 = c + u_1 = a + b + c$. Then the third equation reads $d + e + a + b + c$ which equals f. So, these equations are all zero if and only if f equals zero, which is probably what the authors meant. Also note the similarity with the u variables constructed in this example and the ones defined in Equation (103).

Remark. The reason that they prescribe this is not entirely clear. Using $s < 3$ does not work, since then $\hat{f}_j$ for $j = 2, ..., S_t$ will only consist of u variables and there will not be new terms of f added to the polynomials. However, one could of course use an $s > 3$. The paper probably uses $s = 3$ since the complexity of HHL depends on s, so it might pay of to decrease the sparsity as far as possible. However, the complexity only contains a quadratic term of s. This means that the total complexity might be lower if a bigger s is used to decrease the number of variables and equations added in this step.

7.2.2 Approach 2: matrix based

The main idea of the previous approach is that for each equation, one takes as many terms as possible (i.e. $s-1$) and then adds a new variable to create a correct equation again. Then a new equation is created, starting with the new variable and adding as much of the original equation as possible again. If needed, a new variable is added to create a correct equation again. This continues until the last term of the original equation is added. The same can be done with rows of a matrix. This is best explained by Example 7.5.

Example 7.5. *Assume $s = 3$, the same as in Example 7.4. Consider matrix A, this matrix will be altered, resulting in the 3-sparse matrix A'.*

The first three rows of A already have at most three non-zero entries, so these rows will just be copied. This is the top left part of A'. For row four, which is 4-sparse, one new row and column will be added. This results in the part of A' between the two line, both horizontally and vertically. For row five, which is 5-sparse, two new rows and columns will be added. This results in the part of A' below the second horizontal line and right of the second vertical line.

$$A = \left(\begin{array}{ccccc} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ \hline 1 & 1 & 0 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 \end{array}\right) \Rightarrow A' = \left(\begin{array}{ccccc|c|cc} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ \hline 1 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ \hline 1 & 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{array}\right). \tag{107}$$

8 Macaulay matrix approach

Section 6 shows that the matrix vector system derived from Mini-AES, is not full rank. An advanced linearization technique, that results in a full rank matrix, is to create a Macaulay matrix. This approach uses shifts of the polynomials. A shift of a polynomials is simply a multiplication of a polynomial with a monomial to create a new polynomial with degree d for some $d \in \mathbb{N}$. The idea is that by doing this for all polynomials and all monomials, with the only restriction that the polynomials have degree $d \leq D$, more equations than variables are created. Therefore, if D is chosen large enough, this approach should lead to a full rank matrix. This general approach of shifting the polynomials with all possible monomials will create an exponentially large matrix.

Note that it might be possible to chose these shifts smarter. The general approach is to shift each polynomial by each monomial (with the restriction on the degree). However, one can also try to find combinations of polynomials and monomials for which multiplication results in more new equations than new non-linear terms, without multiplying every equation with the monomial. These are called smart shifts. However, the effect of these are limited in the case of (Mini-)AES, as shown in Appendix H.1.2. Therefore, this idea will not be used in the rest of this report; only the general Macaulay approach will be considered.

Note furthermore, that Section 6 also shows that there might not be a unique solution. However, if a matrix can be created with a sufficiently large rank, the number of solutions will be limited enough to have reasonable probability of finding the desired solution.

This section gives a general definition of a Macaulay linear system and a Macaulay matrix. In the process of creating a Macaulay matrix, there are two decisions to be made; both consisting of two options, which leads to four slightly different Macaulay matrices. The first decision is the method to make sure that non-boolean solutions are excludes. These methods will first be explained in Section 8.1. Then the definition of a Macaulay matrix will be given in Section 8.2. This section ends with a comparison of these four Macaulay matrices in Section 8.3.

Remark. Even though this section has great similarities with [2], a slightly different notation is used to make sure that the other options can also be easily defined with the same notation.

8.1 Non-boolean solutions

As described in Section 7.1, by transforming a system of equations from GF(2) to $\mathbb{C}$, also non-boolean solutions are made acceptable. Example 8.1 shows that this does not have to be a problem, since some solutions can be easily transformed to boolean solutions. In contrast, Example 8.2 shows that some solutions are not that easily transformed.

Example 8.1. *Consider the following system of equations:*

$$\begin{cases} 0 = x^2 - 1 \\ 0 = y - x^2 y \\ 0 = y^2 - 2y. \end{cases} \tag{108}$$

Over $\mathbb{C}$ the following solutions are possible $(1,0), (1,2), (-1,0)$ and $(-1,2)$, while over GF(2) the unique solution is $(1,0)$. However, if these values for the variables are taken modulo 2, every solution boils down to the boolean solution $(1,0)$. So, irrespectively to which of these four solutions was found, the correct boolean solution can be derived.

Example 8.2. *Consider the following system of equations:*

$$\begin{cases} 0 = x^2 - y \\ 0 = y^2 - 2y. \end{cases} \tag{109}$$

Over $\mathbb{C}$ *both* $(-\sqrt{2}, 2), (\sqrt{2}, 2)$ *and* $(0, 0)$ *are correct solutions, while over GF(2) the unique solution is* $(0, 0)$*. If one would find* $(\sqrt{2}, 2)$ *or* $(-\sqrt{2}, 2)$*, it is not clear how to transform this into a correct boolean solution.*

8.1.1 Approach 1: add field equations

Variable x is a boolean over $\mathbb{C}$ if and only if $x^2 - x = 0$. Therefore, the first approach adds equations $x^2 - x = 0$ for each variable in the system. These equations are called field equations for they ensure that the solutions lie in the correct field. Example 8.3 shows that this approach works as expected.

Example 8.3. *This approach will first be applied to Example 8.1 and then to Example 8.2.*

$$\left.\begin{array}{l} 0 = x^2 - 1 \\ 0 = y - x^2 y \\ 0 = y^2 - 2y \end{array}\right\} \Rightarrow \left.\begin{array}{l} 0 = x^2 - 1 \\ 0 = y - x^2 y \\ 0 = y^2 - 2y \\ 0 = x^2 - x \\ 0 = y^2 - y \end{array}\right\} \Rightarrow (x, y) = (1, 0). \tag{110a}$$

$$\left.\begin{array}{l} 0 = x^2 - y \\ 0 = y^2 - 2y \end{array}\right\} \Rightarrow \left.\begin{array}{l} 0 = x^2 - y \\ 0 = y^2 - 2y \\ 0 = x^2 - x \\ 0 = y^2 - y \end{array}\right\} \Rightarrow (x, y) = (0, 0). \tag{110b}$$

One can easily verify that the given solutions are the only solutions (over $\mathbb{C}$*) of the system, which means that this approach finds all the solutions of the original system and excludes any non-boolean solutions.*

8.1.2 Approach 2: remove squares

This approach is also based on the idea of field equations. However, instead of adding the equations to the system, all equations will be simplified using these field equations. This means that every power greater than one will be replaced by one (since $x^2 - x = 0 \Rightarrow x = x^2 = x^3 = ...$). Example 8.4 shows that this approach works as expected.

Example 8.4. *This approach will first be applied to Example 8.1 and then to Example 8.2.*

$$\left.\begin{array}{l} 0 = x^2 - 1 \\ 0 = y - x^2 y \\ 0 = y^2 - 2y \end{array}\right\} \Rightarrow \left.\begin{array}{l} 0 = x - 1 \\ 0 = y - xy \\ 0 = y - 2y \end{array}\right\} \Rightarrow (x, y) = (1, 0). \tag{111a}$$

$$\left.\begin{array}{l} 0 = x^2 - y \\ 0 = y^2 - 2y \end{array}\right\} \Rightarrow \left.\begin{array}{l} 0 = x - y \\ 0 = y - 2y \end{array}\right\} \Rightarrow (x, y) = (0, 0). \tag{111b}$$

One can easily verify that the given solutions are the only solution (over $\mathbb{C}$*) of the system, which means that this approach finds all the solutions of the original system and excludes any non-boolean solutions.*

Remark. Note that, if one tries to solve a system of non-linear equations directly, this approach does not necessarily work. However, it does work when the Macaulay approach is used, as explained in the next subsection.

8.2 Definition of the Macaulay matrix

In order to create a Macaulay matrix, a decision has to made about the definition of the set of polynomials that will be considered. The AES equations are defined over GF(2). However, the HHL algorithm treats the input matrix as a complex matrix. Section 7.1 already showed how to make sure that the equations are correct over $\mathbb{C}$ and Section 8.1 described to ways to exclude non-boolean solutions.

In Section 5 the set of polynomials that need to be solved for Mini-AES are defined. The variables can be denoted as $\mathbb{X} = \{x_1, ..., x_n\}$, using lexicographic monomial ordering for $x_1 > ... > x_n$. The polynomials can then be denoted as $\mathcal{F}_1 = (f_{1_1}, ..., f_{1_r}) \subset \mathbb{B}[\mathbb{X}]$ and a is a solution of $\mathcal{F}_1$ if and only if $\forall_{i=1,...,r} : f_{1_i}(a) = 0$ and $\forall_{j=1,...,n} : a_j = 0 \vee a_j = 1$. Defining $\mathcal{F}_1$ over $\mathbb{B}[\mathbb{X}]$ essentially means simplifying all exponents to one (for example $x^2 \mapsto x$, $x^3 \mapsto x$). The same set of solutions can be defined using polynomials over the complex field $\mathcal{F}_2 = (f_{2_1}, ..., f_{2_r}) \subset \mathbb{C}[\mathbb{X}]$ and adding the field equations $\mathcal{H} = (x_1^2 - x_1, ..., x_n^2 - x_n) \subset \mathbb{C}[\mathbb{X}]$. This corresponds to the two approaches to make sure that non-boolean solutions are excluded, discussed in Section 8.1.

Clearly $\mathbb{V}_{\mathbb{B}}(\mathcal{F}_1) = \mathbb{V}_{\mathbb{C}}(\mathcal{F}_2, \mathcal{H})$, where $\mathbb{V}_{\mathbb{B}}(\mathcal{F}_1)$ denotes the set of boolean solutions of $\mathcal{F}_1$ and $\mathbb{V}_{\mathbb{C}}(\mathcal{F}_2, \mathcal{H})$ denotes the complex solutions that are both a solution to $\mathcal{F}_2$ and to $\mathcal{H}$. So, the first decision that needs to be made is whether to use the representation $\mathcal{F} = \mathcal{F}_1$ or $\mathcal{F} = \mathcal{F}_2 \cup \mathcal{H}$, i.e. whether to add field equations or to remove squares.

Remark. This method will always result in the same solution as the method in which the field equations are added. Consider a Macaulay matrix based on $\mathcal{F} = \mathcal{F}_2 \cup \mathcal{H}$. Every term that is found in (shifts of) an equation in $\mathcal{F}_2$ is also found in (a shift of) an equation in $\mathcal{H}$. This means that row-reduction will simplify all (shifts of all) equations in $\mathcal{F}_2$, essentially by substituting the field equations, until only the (shifts of) the equations in $\mathcal{H}$ contain squares. If the squares are only in (shifts of) the equations in $\mathcal{H}$, then these (shifts of) the equations in $\mathcal{H}$ do not give any additional information to the system about the rest of the the linearized system (normally the field equations give the information that $x = x^2$, but if this x^2 only appears in the field equations the linearized system only knows that x is equal to some other variable). Since, the (shifts of) the equations in $\mathcal{H}$ do not give any additional information at this point, they can be removed. The remaining linearized system is exactly the system that would have been created if the Macaulay matrix was based on $\mathcal{F}_1$ (note that $\mathcal{F}_1$ and $\mathcal{F}_2$ contain the same equations, the only difference is that the equations in $\mathcal{F}_1$ are defined over $\mathbb{B}[\mathbb{X}]$ and the equations in $\mathcal{F}_2$ are defined over $\mathbb{C}[\mathbb{X}]$) and the squares would have been removed in every step instead of at the end. Note that even though both methods (using the field equations or removing the squares) will result in the same solution, this does not mean that the choice for the method does not influence the dimensions of the Macaulay matrix.

Definition 8.1. *(Maximum degree) For variables $x_1, ..., x_n$ and powers $p_1, ..., p_n$, where $n, p_1, ..., p_n \in \mathbb{N}_+$, the max degree of monomial $x_1^{p_1} x_2^{p_2} \cdots x_n^{p_n}$ equals* $\max\{p_1, ..., p_n\}$.

Definition 8.2. *(Total degree) For variables $x_1, ..., x_n$ and powers $p_1, ..., p_n$, where $n, p_1, ..., p_n \in \mathbb{N}_+$, the total degree of monomial $x_1^{p_1} x_2^{p_2} \cdots x_n^{p_n}$ equals $p_1 + ... + p_n$.*

When a set $\mathcal{F} = (f_1, ..., f_k)$ is chosen, the key concept of the Macaulay matrix approach is to create a set of monomials and multiply (i.e. shift) all the polynomials by all these monomials, with the restriction that the resulting polynomial has a degree below a certain upper bound. Here degree

can either mean maximum degree or total degree, as defined in Definition 8.1 and Definition 8.2 respectively. This is the second decision that needs to be made. Denote $\deg_m(f)$ and $\deg_t(f)$ the maximum and total degree of polynomials f respectively. Define $\mathbb{X}^i = x_1^{i_1} x_2^{i_2} ... x_n^{i_n}$ and D_m and D_t the upper bound of the maximum degree and the total degree respectively. Then the set of monomials, for the max degree and the total degree respectively, can be defined as:

$$m_{\leq D_m} = \left\{ \mathbb{X}^i | i = (i_1, ..., i_n) \in \{0, ..., D_m\}^n \right\} \tag{112a}$$

$$m_{\leq D_t} = \left\{ \mathbb{X}^i | i = (i_1, ..., i_n) \in \{0, ..., D_t\}^n, 0 \leq i_1 + ... + i_n \leq D_t \right\}. \tag{112b}$$

Denote D the upper-bound for either the maximum degree or the total degree. The Macaulay matrix is created by multiplying polynomial $f_l \in \mathcal{F}$ for $l = 1, ..., r$ by monomial $(m_{\leq D})_{c,l}$ for $c = 1, ..., |m_{\leq D}|$ where $(m_{\leq D})_{c,l}$ is defined as:

$$(m_{\leq D})_{c,l} = \begin{cases} (m_{\leq D})_c & \text{if } \deg((m_{\leq D})_c \cdot f_l) \leq D \\ 0 & \text{otherwise.} \end{cases} \tag{113}$$

Remark. Note that using $(m_{\leq D})_{c,l}$ instead of $(m_{\leq D})_c$ ensures the restriction on the degree of the resulting polynomial. If the multiplication leads to a polynomial with degree $\leq D$ then the resulting polynomial is included in the Macaulay matrix, otherwise, a zero-row is included in the Macaulay matrix. The reason to include this zero-row and not skip this multiplication has to do with the query complexity of the HHL algorithm [2].

The Macaulay linear system is $\mathcal{M}_{\mathcal{F},D} \mathbf{m}_D = \mathbf{b}_{\mathcal{F},D}$, where $\mathcal{M}_{\mathcal{F},D}$ is called the Macaulay matrix, defined in Equation (114a), and $\mathbf{m}_D$ and $\mathbf{b}_{\mathcal{F},D}$ are defined in Equations (114b) and (114c).

$$\mathcal{M}_{\mathcal{F},D} = \begin{array}{c} \\ (m_{\leq D})_{1,1} \cdot f_1 \\ (m_{\leq D})_{2,1} \cdot f_1 \\ \vdots \\ (m_{\leq D})_{|m_{\leq D}|,1} \cdot f_1 \\ (m_{\leq D})_{1,2} \cdot f_2 \\ \vdots \\ (m_{\leq D})_{|m_{\leq D}|,r} \cdot f_r \end{array} \begin{array}{c} (m_{\leq D})_1 < (m_{\leq D})_2 < ... < (m_{\leq D})_{|m_{\leq D}|} \\ \begin{pmatrix} \dots \\ \dots \\ \dots \\ \dots \\ \dots \\ \dots \\ \dots \end{pmatrix} \end{array} \tag{114a}$$

$$\mathbf{m}_D = \begin{pmatrix} (m_{\leq D})_1 \\ (m_{\leq D})_2 \\ \vdots \\ (m_{\leq D})_{|m_{\leq D}|} \end{pmatrix} \tag{114b}$$

$$\mathbf{b}_{\mathcal{F},D} = \begin{array}{c} (m_{\leq D})_0 = 1 \\ \begin{pmatrix} (m_{\leq D})_{1,1} \cdot f_1(0) \\ (m_{\leq D})_{2,1} \cdot f_1(0) \\ \vdots \\ (m_{\leq D})_{|m_{\leq D}|,1} \cdot f_1(0) \\ (m_{\leq D})_{1,2} \cdot f_2(0) \\ \vdots \\ (m_{\leq D})_{|m_{\leq D}|,r} \cdot f_r(0) \end{pmatrix} \end{array} \tag{114c}$$

8.3 Four different Macaulay matrices

As explained above, there are four different Macaulay matrices possible; Macaulay matrix using

- max degree and field equations, or,
- max degree and removing squares, or,
- total degree and field equations, or,
- total degree and removing squares.

8.3.1 Dimensions

For each Macaulay matrix, the dimensions can be expressed in the number of variables n, the number of equations r, and the upper-bound on the solving degree. Note that an upper bound on the solving degree corresponds to an upper bound on D. These expressions are summarized in Table 2.

The number of columns in a Macaulay matrix equals the number of monomials in set $m_{\leq D}$. Recall Equation (112a) for the max degree. The are n entries in i, which can each take one of $D+1$ values. This are $(D+1)^n$ monomials, excluding the constant monomial results in $(D+1)^n - 1$ columns. Note that this only holds for the method of adding field equations. Removing squares of the monomials, essentially means that now $i \in \{0,1\}^n$. For $D \geq 1$ (which can be assumed without loss of generality, for $D = 0$ would only include the constant monomial and will not lead to a useful Macaulay matrix) this leads to 2^n monomials, excluding the constant monomial results in $2^n - 1$ columns.

Now recall Equation (112b) for the total degree. This essentially divides the total degree $d = 1, ..., D$ over n entries of i. The number of monomials of a certain degree $d \in \{1, ..., D\}$, equals the number of terms in $(x_1 + ... + x_n)^d$ (this product results in a sum, with coefficients, of the specific monomials). The number of terms in $(x_1 + ... + x_n)^d$ can be computed using the multinomial theorem [12, Chapt. 4]. For a specific $d \in \{1, ..., D\}$, the number of monomials is $\binom{d+n-1}{n-1}$, so the total number of columns is $\sum_{d=1}^{D} \binom{d+n-1}{n-1}$. Now for removing the squares, the total degree of a monomial is simply the number of variables in this monomial. For a certain degree $d \in \{1, ..., D\}$, this essentially picks d variables out of the total of n variables, this gives $\binom{n}{d}$ monomials. So, the total number of columns is $\sum_{d=1}^{D} \binom{n}{d}$.

The number of rows can be easily computed. The number of columns equals the number of monomials that each equation will be multiplied by and each multiplication leads to one row in the matrix. Note that the constant one is not considered as a column of the Macaulay matrix (it is included in the vector $\mathbf{b}_{\mathcal{F},D}$). However, the equations multiplied by one should also be included in the Macaulay matrix. This means that if the method of removing the squares is used, the Macaulay matrix will have $(\#cols+1)r$ rows, where $\#cols$ is the number of columns in the Macaulay matrix. When using field equations, also these equations are multiplied by all monomials and there are n field equations. Therefore, this option leads to $(\#cols + 1)(n + r)$ rows.

Above is already mentioned that the solving degree is at least one (i.e. D should be at least one) and there are more trivial bounds on the solving degree (i.e. on D). These trivial bounds are summarized in Table 3.

When using field equations, the solving degree is at least two (i.e. D should be at least two), for both using the max degree and the total degree. Field equations are of the form $x^2 - x = 0$, which means they have a total degree and a max degree of two. When an upper bound on the degree of less than two is used, the field equations will be excluded from the Macaulay matrix.

	Field equations		Remove squares	
	Number of columns	Number of rows	Number of columns	Number of rows
Max degree	$(D+1)^n - 1$	$(\#cols + 1)(n + r)$	$2^n - 1$	$(\#cols + 1)r$
Total degree	$\sum_{d=1}^{D} \binom{d+n-1}{n-1}$	$(\#cols + 1)(n + r)$	$\sum_{d=1}^{D} \binom{n}{d}$	$(\#cols + 1)r$

Table 2: Dimensions of Macaulay matrices

	Field equations	Remove squares
Max degree	$2 \leq D \leq 3n$	$D = 1$
Total degree	$2 \leq D \leq 3n^2$	$1 \leq D \leq n$

Table 3: Trival bounds on D for each Macaulay matrix

Removing squares leads to an upper-bound for the solving degree (i.e. for D) both when using max degree as when using total degree. When removing powers of the variables, each monomial will have a max degree of one, and total degree $\leq n$. So, one should use for the max degree $D = 1$ and for the total degree $D \leq n$.

When the max degree and field equations are used, one should use $D \leq 3n$, according to [2]. In according to [2] $3n$ is called the complete solving degree (CSD), meaning that for a general system of equations, the Macaulay matrix using $D = 3n$ will always be solvable. This translates to a total degree of $3n^2$, when using field equations.

Besides, note that the degree of the original equations does not have to be a lower-bound of the solving degree (i.e. D). Choosing a D lower than the degree of the original equations does mean that some equations are not included in the Macaulay matrix. However, this does not necessarily mean that information is lost, since the equations are not necessarily independent.

8.3.2 Comparison

The literature [2] uses the Macaulay with max degree and field equations. One of the benefits of working with a complex field and adding field equations is that this method is easy to implement.

The down side is that a lot more variables need to be defined. It might be more work to create a Macaulay matrix when working with a boolean field, since this means that after every multiplication a simplification needs to be done (essentially removing all the squares, which Sage can do automatically). However, this does result in less non-linear terms (for instance $x_1^2 x_2 \mapsto x_1 x_2$) and remember that for each non-linear term a new variable needs to be defined in order to linearize the system.

Furthermore, working with field equations results in more equations (the field equations). It is not trivial whether this is a benefit or not. More equations might result in a higher rank, which means a lower D can be used. However, adding field equations instead of removing squares results in an exponentially larger matrix. When max degree is used this results in $(D+1)^n - 1$ columns, for $D \geq 2$, versus $2^n - 1$ columns. When total degree is used and the same D is used, this results in $\sum_{d=1}^{D} \binom{d+n-1}{n-1}$ columns versus $\sum_{d=1}^{D} \binom{n}{d}$ columns. Note that the number of rows depends linearly on the number of columns.

Besides, if the same D is used, working with the max degree instead of the total degree means that more monomials will be added. For example when $D = 2$ the monomials $x_1^2 x_2$, $x_1 x_2^2$ and $x_1^2 x_2^2$ will be included when the maximum degree is used (since they all have maximum degree two) and will be excluded when the total degree is used (since they have total degree three, three and four respectively).

It can be concluded that for the same D, working with field equations instead of removing squares results in a larger Macaulay matrix and working with max degree instead of with total degree results in a larger Macaulay matrix. This raises the question whether this method, as used in [2], actually results in the most efficient Macaulay matrix. The most efficient Macaulay matrix, means the Macaulay matrix with smallest dimensions, that results in a solvable system according to Definition 6.2. To be able to answer this question, for a specific set of r equations on n variables, one has to known what the smallest D is for which the Macaulay matrix is solvable. This is called the solving degree (SD). Then the dimensions of each Macaulay matrix, for the corresponding solving degree, can be computed and the most efficient one can be picked. The hope is that one of the Macaulay matrices is the most efficient, for each set of r equations on n variables, or at least for similar sets. If this is the case, then one can try a small example, similar to the real problem at hand, pick the most efficient Macaulay matrix and create this Macaulay matrix for the real problem. In order to investigate which Macaulay matrix is the most efficient, a program is written that decides for each options what upper-bound for the degree is needed, i.e. what the solving degree is. All options are tested for two sets of polynomials. These tests can be found in Examples 8.5 and 8.6.

Example 8.5. *Consider the following system of non-linear equations in two variables:*

$$\begin{cases} 0 = x_0 - 1 \\ 0 = x_0 x_1 - 1 \end{cases} \tag{115}$$

The corresponding matrix vector system is

$$[A|b] = \left(\begin{array}{cc|c|c} 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{array}\right) \tag{116}$$

This matrix is clearly not solvable according to Definition 6.2. Therefore, the four different Macaulay matrices are created. The dimensions of these matrices are given in Table 4, this table also states for each matrix whether or not it is solvable. Some example matrices of this table are given in Appendix I.

		Field equations		Remove squares	
		Dimensions	Solvable	Dimensions	Solvable
Max degree	1	-	-	8×3	Y
	2	36×8	Y	-	-
	3	64×15	Y	-	-
Total degree	1	-	-	6×2	N
	2	24×5	Y	8×3	Y
	3	40×9	Y	8×3	Y

Table 4: Macaulay matrices for system 1

Example 8.6. *Consider the following system of non-linear equations in three variables:*

$$\begin{cases} 0 = x_0 - 1 \\ 0 = x_0 x_1 - 1 \\ 0 = x_0 x_2 + x_1 x_2. \end{cases} \tag{117}$$

The corresponding matrix vector system is

$$[A|b] = \left(\begin{array}{ccc|ccc|c} 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{array}\right) \tag{118}$$

This matrix is clearly not solvable according to Definition 6.2. Therefore, the four different Macaulay matrices are created. The dimensions of these matrices are given in Table 5, this table also states for each matrix whether or not it is solvable. Some example matrices of this table are given in Appendix I.

		Field equations		Remove squares	
		Dimensions	Solvable	Dimensions	Solvable
Max degree	1	-	-	24×7	Y
	2	162×26	Y	-	-
	3	384×63	Y	-	-
Total degree	1	-	-	12×3	N
	2	60×9	N	21×6	N
	3	120×19	Y	24×7	Y

Table 5: Macaulay matrices for system 2

The conclusion of Examples 8.5 and 8.6 is that the smallest solvable matrix is found when either the max or the total degree is used and the squares are removed. Also when the total degree is used and the squares are removed, a smaller matrix is sufficient than when the max degree and field equations are used. This means that for these examples, the option used by [2] is the least efficient. This is not enough evidence to conclude that the literature uses the least efficient Macaulay matrix and that removing squares with either the max or the total degree leads to the most efficient Macaulay matrix. However, it is clear that for the same D the Macaulay matrix with field equations and max degree has the largest dimensions, so it is wise to seriously doubt the strategy chosen by [2] and do further research on Macaulay matrices for AES specifically.

9 Applying HHL to Mini-AES

As Sections 7 and 8 show, to go from (Mini-)AES to input that is suitable for HHL, four things should be handled: translating from GF(2) to $\mathbb{C}$, decrease sparsity, and creating a Macaulay matrix, excluding non-boolean solutions in the process of creating a Macaulay matrix.

Section 7.1 discusses two approaches to transform the equations. However, it is also shown that the matrix based approach is not suitable for this report's attack. Therefore, only the equation based approach is considered. This means that this has to be done before creating the Macaulay matrix.

Section 7.2 discusses two approaches to decrease the sparsity. The equation based approach has to be done before creating the Macaulay matrix. It can be done before or after transforming the equations from GF(2) to $\mathbb{C}$. The matrix based approach has to be done after creating the Macaulay matrix. Besides, one could also choose to not decrease the sparsity. The HHL algorithm will still work on a less sparse matrix, only the complexity will be higher. Since the complexity depends on both the sparsity and the dimensions of the input matrix, where decreasing the sparsity will increase the dimensions of the matrix, this is a trade-off. Note that decreasing sparsity actually means decreasing sparsity to some value $s > 2 \in \mathbb{N}$, where s is the maximum number of non-zeros per row of the matrix (or terms per equation since s terms in an equation results in s non-zeros in a row of the matrix). So one could try some different values for s to check which leads to de lowest complexity. However, for simplicity reasons, this project will only consider decreasing the sparsity to $s = 3$ or not decreasing the sparsity at all.

Remark. The complexity of HHL is minimal for $s = 3$ when the matrix based approach of decreasing the sparsity is used, assuming that this does not influence κ too much. If A is an s_0-sparse Macaulay matrix of dimensions $N \times N$ and A' is the corresponding s_1-sparse Macaulay matrix, where $s_1 < s_0$, created using the matrix based approach, then A' has dimensions $N' \times N'$, where $N' \leq N + (s_0 - s_1)N$. The complexity of HHL applied to A' is $\mathcal{O}\big(\log(N')s_1^2\kappa^2/\epsilon\big) = \mathcal{O}\big(\log(N + (s_0 - s_1)N)s_1^2\kappa^2/\epsilon\big)$. Since $\log(N + (s_0 - s_1)N)s_1^2$ is an increasing function for increasing s_1, the complexity is minimal for the smallest possible s_1, i.e. $s_1 = 3$.

Section 8.1 discusses two approaches for excluding the non-boolean solutions. Both approaches are executed in the process of making the Macaulay matrix. Here the field equations are added to the set of equations, or the squares are removed when new equations are formed.

9.1 Four strategies to create the input for HHL

This means that there are four strategies to go from the (Mini-)AES equations to input for HHL. Strategy:

1. Transform equations from GF(2) to $\mathbb{C}$, then create Macaulay matrix.
2. Transform equations from GF(2) to $\mathbb{C}$, then create Macaulay matrix, then decrease sparsity of Macaulay matrix.
3. Transform equations from GF(2) to $\mathbb{C}$, then decrease sparsity of equations, then create Macaulay matrix.
4. Make equations 3-sparse, transform sparse equations from GF(2) to $\mathbb{C}$, then create Macaulay matrix.

In the strategies above, there are two points in which the dimensions of the Macaulay matrix are heavily influenced. Transforming non-sparse equations from GF(2) to $\mathbb{C}$, which is done in Stratgies

1, 2 and 3, results in a lot of terms. This means that it is a lot of work to make the equations sparse afterwards. When the equations are made sparse before creating the Macaulay matrix, which is done in Strategy 3, this introduces a lot of new variables. This is the second point that heavily influences the dimensions of the Macaulay matrix. This can be avoided by creating the Macaulay matrix first, which is done in Strategies 1 and 2. It is possible to make the Macaulay matrix sparse at this point, which is done in Strategy 2.

Also the first point in which the dimensions of the Macaulay matrix are heavily influenced can be avoided. Transforming 3-sparse equations from GF(2) to $\mathbb{C}$ results in equations of at most six terms. Therefore, it might be beneficial to make the equations 3-sparse first, which is done in Strategy 4, and then transform the equations from GF(2) to $\mathbb{C}$. This way, the equations that are made three sparse have less terms than in the strategies above, which results in introducing less variables in the step of decreasing the sparsity. This strategy is done in [2].

Intuitively, it is best to not start by transforming the equations from GF(2) to $\mathbb{C}$, since a blow-up here results in a way bigger blow-up in the next step. To evaluate the strategies, the dimensions of the matrix resulting of Strategies 2, 3, and 4 will be investigated using a small example. Also Strategy 1 will be investigated as a reference. All four strategies will be applied on all four possible Macaulay matrices as defined in Section 8.3. This experiment is given in Example 9.1.

Example 9.1. *Consider the following system of non-linear equations on two variables, over GF(2):*

$$\mathcal{F} = \left\{ \begin{array}{l} 0 = x_0x_1 + x_0 + x_1 + 1, \\ 0 = x_0 + 1, \\ 0 = x_2 + 1, \\ 0 = x_0x_1 + 1 \end{array} \right\}. \tag{119}$$

Transforming these equations from GF(2) to $\mathbb{C}$, using the equation based approach from Section 7.1, results in:

$$C(\mathcal{F}) = \left\{ \begin{array}{l} 0 = x_0^2x_1^2 + 2x_0^2x_1 + 2x_0x_1^2 + x_0^2 - 2x_0x_1 + x_1^2 - 4x_0 - 4x_1 + 3, \\ 0 = x_0 - 1, \\ 0 = x_2 - 1, \\ 0 = x_0x_1 - 1 \end{array} \right\}. \tag{120}$$

Then making the equations 3-sparse, using the equation based approach from Section 7.2, results in:

$$S(C(\mathcal{F}), 3) = \left\{ \begin{array}{l} 0 = x_0^2x_1^2 + 2x_0^2x_1 - u_1, \\ 0 = u_1 + 2x_0x_1^2 - u_2, \\ 0 = u_2 + x_0^2 - u_3, \\ 0 = u_3 - 2x_0x_1 - u_4, \\ 0 = u_4 + x_1^2 - u_5, \\ 0 = u_5 - 4x_0 - u_6, \\ 0 = u_6 - 4x_1 + 3, \\ 0 = x_0 - 1, \\ 0 = x_2 - 1, \\ 0 = x_0x_1 - 1 \end{array} \right\}. \tag{121}$$

Making the equations in $\mathcal{F}$ 3-sparse, using the equation based approach from Section 7.2, results in:

$$S(\mathcal{F}, 3) = \left\{ \begin{array}{l} 0 = x_0x_1 + x_0 - u \\ 0 = u + x_1 + 1 \\ 0 = x_0 + 1 \\ 0 = x_2 + 1 \\ 0 = x_0x_1 + 1. \end{array} \right. \tag{122}$$

Then transforming these equations from GF(2) to $\mathbb{C}$, using the equation based approach from Section 7.1, results in:

$$C(S(\mathcal{F},3)) = \begin{cases} 0 = 2x_0x_1x_0 + 2x_0x_1u + 2x_0u - x_0x_1 - x_0 - u \\ 0 = u + x_1 - 1 \\ 0 = x_0 - 1 \\ 0 = x_2 - 1 \\ 0 = x_0x_1 - 1. \end{cases} \tag{123}$$

Strategies 2 and 1 create Macaulay matrices of $C(\mathcal{F})$, Strategy 2 creates a Macaulay matrix of $S(C(\mathcal{F}),3)$, and Strategy 4 creates a Macaulay matrix of $C(S(\mathcal{F},3))$. The results of creating all four Macaulay matrices, using all four strategies, are summarized in Table 6. The D's in this table the minimum values such that the corresponding Macaulay matrix is solvable.

		Field equations		Remove squares	
		Dimensions	D	Dimensions	D
Str. 1	M_d	189×26	2	32×7	1
	T_d	70×9	2	28×6	2
Str. 2	M_d	201×38	2	34×9	1
	T_d	71×10	2	29×7	2
Str. 3	M_d	255879×19683	2	6656×511	1
	T_d	550×54	2	520×51	2
Str. 4	M_d	729×80	2	80×15	1
	T_d	135×14	2	55×10	2

Table 6: Macaulay matrices for strategies 1,2,3 and 4

The conclusion of Example 9.1 for each strategy is that the Macaulay matrix with total degree and removing the squares is the most efficient. For this Macaulay matrix, Strategy 3 is the least efficient.

Strategies 1 and 2 include the same equations and monomials, but the Macaulay matrices in Strategy 1 are 4-sparse whereas the Macaulay matrices in Strategy 2 are made 3-sparse, resulting in a slight difference in dimensions. The reason for this minimal difference , is the limited number of variables in this example. Only two variables were used, meaning that after removing squares, only four monomials exist: x_0, x_1, x_0x_1, and 1. Therefore, the matrix for removing squares is only 4-sparse and decreasing the sparsity of this matrix only includes one more column and row. However, the more variables are used, the more new columns and rows have to be added to create a 3-sparse matrix. Since the complexity of HHL depends quadratically on the sparsity s and logarithmically on the size of the matrix, and the matrix grows only linearly in Strategy 2, this strategy is considered more efficient than Strategy 1, when applied to (Mini-)AES.

Strategy 3 transforms the equations from GF(2) to $\mathbb{C}$, then makes the equations 3-sparse and creates the Macaulay matrix afterwards. Therefore, the Macaulay matrix is created using more

variables, resulting in more monomials for the same degree. This means that the number of rows and the number of columns is way larger in Strategy 3 than in Strategies 1 and 2.

Strategy 4 also creates a Macaulay matrix using more variables than Strategies 1 and 2, but less variables than Strategy 3. Therefore, the dimensions of the Macaulay matrices in Strategy 4 are larger than in Strategies 1 and 2, but smaller than in Strategy 3. Furthermore, the Macaulay matrices in Strategy 4 are always at most 6-sparse, since the equations are first made 3-sparse and transforming equations that are 3-sparse from GF(2) to $\mathbb{C}$ result in equations that are at most 6-sparse. This means that it would not be fair to compare the dimensions of the matrices in this strategy with the dimensions of the matrices in Strategy 2, since these are 3-sparse. However, the matrices in Strategy 1 are 4-sparse in this example, meaning that this strategy result in sparser matrices with smaller dimensions than Strategy 4, hence Strategy 1 is the better strategy for this example.

To conclude the analysis, it most be noted that Strategy 4 scores worse in this example than Strategies 1 and 2 due to the limited number of terms of the equations in $\mathcal{F}$. The more terms in the equations, the bigger the blow-up in the number of terms when transforming these from GF(2) to $\mathbb{C}$, which results in a way bigger blow-up when the equations are made sparse again. When the equations of $\mathcal{F}$ have more terms, it might become beneficial to first decrease the sparsity and than transform them from GF(2) to $\mathbb{C}$. This means that unlike this example, Strategy 4 might still be the best strategy for Mini-AES.

9.2 Four strategies applied to Mini-AES

Evaluating the three strategies for (Mini-)AES is a bit harder, since it is not possible anymore to create all four Macaulay matrices, with increasing degree, until a solvable matrix is found. The resulting matrices are too big to fit in the memory of a regular laptop.

However, from Table 2 can be concluded that it is sufficient to look at the number of columns of the Macaulay matrices, to evaluate the strategies, since this is the dominant factor in computing the number of rows. Furthermore, Table 2 summarized the the dimensions of the Macqaulay matrices and Table 3 summarized the trivial bounds on D. Combining these results in the trivial bounds on the number of columns of the Macaulay matrices given in Table 7. Note that these do not depend on D anymore, so they can be used to bound the number of columns even if the solving degree, i.e. the smallest D for which the corresponding Macaulay matrix is solvable, is unknown.

	Field equations	Remove squares
Max degree	$3^n - 1 \leq \#cols \leq (3n+1)^n - 1$	$\#cols = 2^n - 1$
Total degree	$\frac{1}{2}n(n+3) \leq \#cols \leq (3n+\frac{1}{n})\binom{n(3n+1)}{n-1} - 1$	$n \leq \#cols \leq 2^n - 1$

Table 7: Trival bounds on $\#cols$ for each Macaulay matrix

This means that to evaluate the three strategies, only the number of variables needs to be determined. According to Table 1, Mini-AES consists of $n = 56+64 = 120$ variables, $32+48 = 80$ linear

equations, and $42 + 168 = 210$ non-linear equations. The linear equations consist of 8 equations with three terms and 72 equations with four terms. The non-linear equations come from ten nibble subs, each of which needs 21 equations with a different number of terms. This is summarized in Table 8. All strategies start by transforming the equations to $\mathbb{C}$, according to the equation based approach discussed in Section 7.1. This results in the same number of equations, but more terms per equation. The number of terms per equation is given in Table 8. Depending on the strategy, the sparsity of the equations over $\mathbb{C}$ will be decreased to three. For a function of $t > 3$ terms, this introduces $t - 3$ new equations and variables.

		Over GF(2)		Over $\mathbb{C}$	
		# eq.	# terms	# eq.	# terms
Linear equations		8	3	8	6
		72	4	72	34
Per nibble sub (10 in total)		1	4	1	34
		1	5+1	1	56
		2	6	2	209
		4	7+1	4	330
		4	8	4	1286
		1	8+1	1	495
		2	9	2	2001
		3	9+1	3	2002
		2	10	2	8007
		1	11+1	1	12376
		1	12+1	1	18564

Table 8: Mini-AES equations, transformed to $\mathbb{C}$ and decreased sparsity

Strategies 1 and 2 will create a Macaulay matrix on $n_1 = 120$ variables and $r_1 = 80 + 210 = 290$ equations. Strategy 3 will create a Macaulay matrix with:

$$\begin{aligned} 8 \cdot 3 + 72 \cdot 31 + 10 \cdot (31 + 53 + 2 \cdot 206 + 4 \cdot 327 + 4 \cdot 1283 + 492 + \\ 2 \cdot 1998 + 3 \cdot 1999 + 2 \cdot 8004 + 12373 + 18561) = 642886, \end{aligned} \tag{124}$$

additional variables and equations, i.e. $n_3 = 120 + 642886 = 643006$ and $r_3 = 290 + 642886 = 643176$.

Strategy 4 will create a Macaulay matrix with:

$$72 \cdot 1 + 1 \cdot 1 + 3 \cdot 3 + 8 \cdot 5 + 3 \cdot 6 + 5 \cdot 7 + 1 \cdot 9 + 1 \cdot 10 = 194, \tag{125}$$

additional variables and equations, i.e. $n_4 = 120 + 194 = 314$ and $r_4 = 290 + 194 = 484$.

These n_1, n_3 and n_4 substituted in Table 7 directly result in bounds on the number of columns for the Macaulay matrices for Strategies 1 and 3. Strategy 2 starts with the same Macaulay matrix as Strategy 1, but to decrease the sparsity it adds as many columns (and rows) as the number of variables added in Strategy 3, i.e. 642886. This results in the bounds on the number of columns given in Table 9.

		Field equations	Remove squares
Str. 1	M_d	$1.8 \times 10^{57} \leq \#cls \leq 8.0 \times 10^{306}$	$\#cls = 1.3 \times 10^{36}$
	T_d	$7.4 \times 10^{3} \leq \#cls \leq 3.2 \times 10^{357}$	$1.2 \times 10^{2} \leq \#cls \leq 1.3 \times 10^{36}$
Str. 2	M_d	$1.8 \times 10^{57} \leq \#cls \leq 8.0 \times 10^{306}$	$\#cls = 1.3 \times 10^{36}$
	T_d	$6.5 \times 10^{5} \leq \#cls \leq 3.2 \times 10^{357}$	$6.4 \times 10^{5} \leq \#cls \leq 1.3 \times 10^{36}$
Str. 3	M_d	$6.8 \times 10^{306791} \leq \#cls \leq 1.2 \times 10^{4041509}$	$\#cls = 1.2 \times 10^{193564}$
	T_d	$2.1 \times 10^{11} \leq \#cls \leq 3.0 \times 10^{4320765}$	$6.4 \times 10^{5} \leq \#cls \leq 1.2 \times 10^{193564}$
Str. 4	M_d	$6.5 \times 10^{149} \leq \#cls \leq 1.8 \times 10^{47042}$	$\#cls = 3.3 \times 10^{94}$
	T_d	$5.0 \times 10^{4} \leq \#cls \leq 4.4 \times 10^{1068}$	$3.1 \times 10^{2} \leq \#cls \leq 3.3 \times 10^{94}$

Table 9: Number of columns of Macaulay matrices for Mini-AES

Table 9 shows that the lower-bound on the number of columns for the Macaulay matrix with max degree and field equations is already higher than the upper-bound on the number of columns for Macaulay matrices with removing squares. Therefore, it can be concluded that Macaulay matrices with removing squares are more efficient for Mini-AES than the Macaulay matrix with max degree and field equations. It is harder to compare the Macaulay matrix with total degree and field equations to the other Macaulay matrices, since this Macaulay matrix has quite a low lower-bound and the highest upper-bound. In Example 9.1, the Macaulay matrix with total degree and field equations did not have the best result nor the worst. However, this example is too small to derive a conclusion from.

Furthermore, Table 9 shows that Strategies 1 and 2 result in Macaulay matrices of similar size. This is, as explained above, due to the fact that decreasing the sparsity of the matrices created in Strategy 1, only increases the size linearly in the sparsity.

Next, it can be concluded that Strategy 3 results in way bigger matrices, for similar values of D, than any other strategy. This conclusions excludes the case that total degree is used with removing squares and $D = 1$ appears to be enough. However, if $D = 1$ would be enough, this would mean that only the linear equations of Mini-AES would be enough to uniquely determine the variables. It is safe to assume that this is not the case. When total degree is used with removing squares and $D = 2$, the number of columns in Strategies 1, 2, 3, and 4 equals 7.3×10^3, 6.5×10^5, 2.1×10^{11}, and 4.9×10^4 respectively. Hence, for Mini-AES, Strategy 3 results in way bigger matrices, for similar values of D.
This is as expected since these matrices are created with way more variables and this is the dominant term in the number of columns. However, the upper bound on the number of columns in Strategies 1, 2, and 4 are not always lower than the lower-bound on the number of columns in the corresponding Macaulay matrix of Strategy 3. Therefore, it is not proven that Strategy 3 is the least efficient. However, this strategy only results in a smaller matrix if a lower D is used and there is no reason to believe that this strategy will work with a lower D than the other strategies.

This is based on the reasoning that even though decreasing the sparsity of the equations does add new equations (and variables), it does add not more information about the variables, nor does it influence the linear dependency between equations. Therefore, it should have no influence on the solvability of the matrix, nor on the solvability of the Macaulay matrix. Assuming that similar D's are needed for all strategies, it is clear that Strategy 3 is least efficient.

Now that Strategy 3 is out of the picture, the question remains whether Strategy 2 or Strategy 4 is more efficient. When the max degree is used and the squares are removed it is clear that Strategy 2 is more efficient than Strategy 4. When max degree and field equations are used, the lower bound and the upper bound on the number of columns in Strategy 2 are smaller than the corresponding bounds in the Strategy 4. The reason for this is that the number of monomials, in the case of max degree, for $D \geq 2$ already becomes very large for $n_4 = 314$ with respect to the number of variables that are added to make the matrices in Strategy 3-sparse. So, for similar values of D and using max degree, Strategy 2 is more efficient than Strategy 4. However, since the upper bound of Strategy 2 is higher than the lower bound of Strategy 4, when using max degree and field equations, it can not be concluded that Strategy 2 is necessarily more efficient.
When the total degree is used, Strategy 4 has a lower lower bound and a higher upper bound on the number of columns than Strategy 2, which means that no conclusion can be made other than: for small values of D is Strategy 4 more efficient and for large values of D Strategy 2 is more efficient. This is also as expected, since Strategy 4 creates a Macaulay matrix on more variables, but does not have to be made sparse afterwards and Strategy 2 adds a constant number of columns at the end to make the matrix sparse. For small values of D, the number of variables does not influence the dimension as much, meaning that it is better to create a Macaulay matrix on some more variables than to add 642886 columns in the end to make the matrix sparse.
Note that it is not entirely fair to compare the matrices of Strategy 2 with the matrices of Strategy 4, since in the first case the matrices are 3-sparse and in the second case they are 6-sparse. However, as explained above, this does not influence the result that much.

10 AES Equations

Section 5 provides a list of equations, of which the solution is the solution of a chosen plaintext attack for Mini-AES, meaning that the solution of the system of equations equals a key that encrypts the chosen plaintext to the known ciphertext. This section does the same for AES instead of Mini-AES. Therefore, this section has great similarities with Section 5, the main difference is that the constants in this section are bigger. Other differences between the equations for Mini-AES and AES will be explained below.

The mixColumns algorithm uses a matrix M. In this section this matrix equals M_1, as defined in Appendix A. However, the notation M is used, to underline the similarity with AES. If omitted: $j = 0, ..., N_b N_e - 1$, $m = 0, ..., se-1$, $i = 1, ..., N_r$, $\jmath = 0, ..., N_k N_e - 1$, and $\imath = 1, ..., \lceil N_b(N_r+1)/N_k \rceil$. Hence, for AES $j = 0, ..., 7$, $m = 0, ..., 7$, and $i = 1, ..., N_r$. To simplify notation, modulos are omitted, but indices work modulo $N_b N_e = 16$.

10.1 Encryption equations

For the initial round for Mini-AES Equation (67) was derived. This equation remains the same for AES. However, where Mini-AES used a 2×2 matrix of which every element was a nibble to define x_0, p and w_0, AES uses 4×4 matrices that are filled with bytes. This difference is captured in the different values for j and m.

For the middle rounds, Equation (72) was derived. Again this equation stays the same for AES. However, by definition:

$$\begin{aligned} \text{rotation}(y_i) &= \text{rotation}\left(\begin{pmatrix} y_i(0,:) & y_i(4,:) & y_i(8,:) & y_i(12,:) \\ y_i(1,:) & y_i(5,:) & y_i(9,:) & y_i(13,:) \\ y_i(2,:) & y_i(6,:) & y_i(10,:) & y_i(14,:) \\ y_i(3,:) & y_i(7,:) & y_i(11,:) & y_i(15,:) \end{pmatrix}\right) \\ &= \begin{pmatrix} y_i(0,:) & y_i(4,:) & y_i(8,:) & y_i(12,:) \\ y_i(5,:) & y_i(9,:) & y_i(13,:) & y_i(1,:) \\ y_i(10,:) & y_i(14,:) & y_i(2,:) & y_i(6,:) \\ y_i(15,:) & y_i(3,:) & y_i(7,:) & y_i(11,:) \end{pmatrix}, \end{aligned} \tag{126}$$

and :

$$\begin{aligned} \text{mixColumns}(\text{rotation}(y_i)) =& M_1 \cdot \text{rotation}(y_i) \\ =& \begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix} \cdot \begin{pmatrix} y_i(0,:) & y_i(4,:) & y_i(8,:) & y_i(12,:) \\ y_i(5,:) & y_i(9,:) & y_i(13,:) & y_i(1,:) \\ y_i(10,:) & y_i(14,:) & y_i(2,:) & y_i(6,:) \\ y_i(15,:) & y_i(3,:) & y_i(7,:) & y_i(11,:). \end{pmatrix} \end{aligned} \tag{127}$$

This difference is again captured in the different values for j and m that are used for AES versus the values that were used for Mini-AES.

For the final round, Equation (76) was derived. However, this was the addition of 2×2 matrices filled with nibbles and this means now the addition of 4×4 matrices filled with bytes. This difference is once more captured by the different values for j and m.

The final equation that was derived for the encryption in Mini-AES is Equation (77). In order to write this bit-wise, Equation (79) was used to derive Equation (80). However, AES uses a different sBox, so, a significantly different system of equations arises. These equations are derived in Section 10.2.

10.2 sBox equations

To describe $\bar{x}_{i-1}(j,m) = \bar{x}_{i-1}(j,m)$, a set S of sBox equations is defined. The equations in this set are given in Appendix D. The following equations are needed for the encryption:

$$S(x_{i-1}(j,0),...,x_{i-1}(j,se-1),\bar{x}_{i-1}(j,0),...,\bar{x}_{i-1}(j,se-1)). \tag{128}$$

Since AES also uses the sBox to substitute the last word of each round key, also $\bar{w}_{\imath-1}(\jmath,m) = \text{eltSub}(w_{\imath-1}(\jmath,m))$ for $\jmath = N_e(N_k-1),...,N_eN_k-1$ need to be described. Define for these parameters:

$$S(w_{\imath-1}(j,0),...,w_{\imath-1}(j,S_e-1),\bar{w}_{\imath-1}(j,0),...,\bar{w}_{\imath-1}(j,S_e-1)). \tag{129}$$

10.3 Key expansion equations

In Algorithm 3 the key expansion is described in the words of the key and the words of the round keys. This description will be used to express the round keys by their bytes. If omitted: $\imath = 1,...,\lceil N_b(N_r+1)/N_k\rceil - 1$, and $\jmath = 0,...,N_kN_e-1$.

For $\jmath = 0,...,N_kN_e-1$

$$w_0(\jmath,:) \leftarrow k(\jmath,:) \tag{130a}$$

For $l = N_k,...,N_b(N_r+1)-1$

$$w_\imath(0:N_e,:) \leftarrow \text{eltSub}(\text{rotWord}(w_{\imath-1}(-N_e:0,:)))\oplus \tag{130b}$$
$$rcon_\imath(:,:)\oplus w_{\imath-1}(0:N_e,:)$$

$$w_\imath(lN_e:(l+1)N_e,:) \leftarrow \text{eltSub}(w_\imath((l-1)N_e:lN_e,:))\oplus \quad \text{if } N_k>6, l=4 \bmod N_k \tag{130c}$$
$$w_{\imath-1}(lN_e:(l+1)N_e,:)$$

$$w_\imath(lN_e:(l+1)N_e,:) \leftarrow w_\imath((l-1)N_e:lN_e,:)\oplus \quad \text{otherwise} \tag{130d}$$
$$w_{\imath-1}(lN_e:(l+1)N_e,:)$$

Equation (130a) tells to copy the words of the key $\jmath = 0,...,N_kN_e-1$ into the first words of the round key. This can be written bit-wise:

$$w_0(\jmath,m) = k(\jmath,m). \tag{131}$$

However, this again only creates unnecessary variables and equations. Therefore, these equations will be omitted and the variables $w_0(\jmath,m)$ will be used instead of the variables $k(\jmath,m)$.

Equation (130b) tells that for the first word of every round key ($\imath = 1,...,\lceil N_b(N_r+1)/N_k\rceil - 1$) the eltSub of the rotWord of the last word (which is the last word of the previous round key) needs to be taken and this needs to be XORed with the same (first) word of the previous round key and the round constant corresponding to this round. Applying eltSub and rotWord to the last word of the previous round key, results in (for $\jmath = 0,...,N_e-1$):

$$w_\imath(\jmath,:) = \bar{w}_{\imath-1}(\jmath-N_e+1,:))\oplus rcon_\imath(\jmath,:)\oplus \quad \text{For } \jmath = 0,...,N_e-2 \tag{132a}$$
$$w_{\imath-1}(\jmath,:)$$

$$w_\imath(N_e-1,:) = \bar{w}_{\imath-1}(N_e(N_k-1),:))\oplus \tag{132b}$$
$$rcon_\imath(N_e-1,:)\oplus w_{\imath-1}(N_e-1,:),$$

which can easily be written bit-wise (for $\jmath = 0,...,N_e-1$):

$$w_\imath(\jmath,m) = \bar{w}_{\imath-1}(\jmath-N_e+1,m))\oplus rcon_\imath(\jmath,m)\oplus \quad \text{For } \jmath = 0,...,N_e-2 \tag{133a}$$
$$w_{\imath-1}(\jmath,m)$$

$$w_\imath(N_e-1,m) = \bar{w}_{\imath-1}(N_e(N_k-1),m)) \oplus rcon_\imath(N_e-1,m) \oplus w_{\imath-1}(N_e-1,m). \tag{133b}$$

For Equations (130c) and (130d), a distinction needs to be made between $N_k \geq 6$ and $N_k < 6$. First consider the case that $N_k < 6$. Then Equation (130c) can be ignored and Equation (130d) prescribes to XOR the previous word with the same word of the previous round. This can be written bit-wise (for $\jmath = N_e, ..., N_k N_e - 1$):

$$w_\imath(\jmath,m) = w_{\imath-1}(\jmath,m) \oplus w_\imath(\jmath-N_e,m). \tag{134}$$

Now consider the case that $N_k \geq 6$. Then Equation (130c) describes that for every fourth word of a round key (excluding the first round key) one should take the XOR of the eltSub of the previous word and the same word of the previous round. So, the only difference is that now the eltSub needs to be computed before we XOR. Equation (130d) describes again that in the other cases (if $l \neq 4 \bmod N_k$) the XOR of the previous word and the same word of the previous round needs to be computed. So this can be written bit-wise:

$$w_\imath(\jmath,m) = w_{\imath-1}(\jmath,m) \oplus \bar{w}_\imath(\jmath-N_e,m) \qquad \text{For } \jmath = 4N_e, ..., 5N_e-1 \tag{135a}$$

$$w_\imath(\jmath,m) = w_{\imath-1}(\jmath,m) \oplus w_\imath(\jmath-N_e,m) \qquad \text{For } \jmath = N_e, ..., 4N_e-1 \text{ or } \jmath = 5N_e, ..., N_k N_e - 1. \tag{135b}$$

10.4 System of equations for AES

If omitted: $i = 1, ..., N_r$,$j = 0, ..., N_e N_b - 1, m = 0, ..., S_e - 1$, $\jmath = 0, ..., N_k N_e - 1$, and $\imath = 1, ..., \lceil N_b(N_r+1)/N_k \rceil - 1$.

This section collects all equations needed to describe AES encryption and key expansion. This list consists of Equations (67), (72), (76), (128), (129), (133a), (134), and (135a). Adding all the terms on the left-hand-side to both the left-hand-side and right-hand-side, results in:

$$0 = x_0(j,m) \oplus p(j,m) \oplus w_0(j,m) \tag{136a}$$

$$0 = x_i(j,m) \oplus w_i(j,m) \oplus \sum_{j'=0}^{N_b-1} \sum_{m'=0}^{S_e-1} M(j_m + j' N_b, m') \bar{x}_{i-1}(j_p + j'(N_b+1), m') \qquad \text{For } i = 1, ..., N_r - 1 \tag{136b}$$

$$0 = c(j,m) \oplus \bar{x}_{Nr-1}(j_r,m) \oplus w_{N_r}(j,m) \tag{136c}$$

$$0 = S(x_{i-1}(j,0), ..., x_{i-1}(j,S_e-1), \bar{x}_{i-1}(j,0), ..., \bar{x}_{i-1}(j,S_e-1)) \tag{136d}$$

$$0 = S((w_{\imath-1}(j,0), ..., w_{\imath-1}(j,S_e-1), \bar{w}_{\imath-1}(j,0), ..., \bar{w}_{\imath-1}(j,S_e-1)) \qquad \text{For } \jmath = N_e(N_k-1), ..., N_e N_k - 1 \tag{136e}$$

$$0 = w_\imath(\jmath,m) \oplus \bar{w}_{\imath-1}(\jmath - N_e + 1, m)) \oplus rcon_\imath(\jmath,m) \oplus w_{\imath-1}(\jmath,m) \qquad \text{For } \jmath = 0, ..., N_e - 2 \tag{136f}$$

$$0 = w_\imath(N_e-1,m) \oplus \bar{w}_{\imath-1}(N_e(N_k-1),m)) \oplus rcon_\imath(N_e-1,m) \oplus w_{\imath-1}(N_e-1,m) \tag{136g}$$

For $N_k \leq 6$

$$0 = w_\imath(\jmath, m) \oplus w_{\imath-1}(\jmath, m) \oplus w_\imath(\jmath - N_e, m) \quad \text{For } \jmath = N_e, ..., N_k N_e - 1 \quad (136h)$$

For $N_k > 6$

$$0 = w_\imath(\jmath, m) \oplus w_{\imath-1}(\jmath, m) \oplus \bar{w}_\imath(\jmath - N_e, m) \quad \text{For } \jmath = 4N_e, ..., 5N_e - 1 \quad (136i)$$

$$0 = w_\imath(\jmath, m) \oplus w_{\imath-1}(\jmath, m) \oplus w_\imath(\jmath - N_e, m) \quad \text{For } \jmath = N_e, ..., 4N_e - 1 \quad (136j)$$
$$\jmath = 5N_e, ..., N_k N_e - 1$$

11 Known plaintext attack for AES-128

Section 6 established the size of the chosen plain-text attack for Mini-AES, this section will do the same for AES-128. The parameters for AES-128 are $S_e = 8$, $N_b = 4$, $N_e = 4$, $N_k = 4$, $N_r = 10$, and $B_s = 128$. Note that for AES-128 $N_k \leq 6$, so only the last word of a round key goes into the S-box.

Variables c and p are known, whereas all other variables are unknown. For the encryption $x_0, ..., x_{N_r-1}$ and $\bar{x}_0, ..., \bar{x}_{N_r-1}$ are unknown, i.e $2 \cdot N_r \cdot N_b \cdot N_e \cdot S_e = 2560$ variables. For the key expansion $w_0, ..., w_{\lceil N_b(N_r+1)/N_k \rceil - 1}$ and $\bar{w}_0(\jmath, m), ..., \bar{w}_{\lceil N_b(N_r+1)/N_k \rceil - 2}(\jmath, m)$ for $\jmath = (N_k - 1)N_e, ..., N_kN_e - 1$ and $m = 0, ..., S_e - 1$ are unknown, i.e. $\lceil N_b(N_r + 1)/N_k \rceil \cdot N_eN_k \cdot S_e + (\lceil N_b(N_r + 1)/N_k \rceil - 1) \cdot N_e \cdot S_e = 1728$ variables.

Now let's count the number of equations. Equations (136a), (136b), and (136c) are for the encryption, in each line $j = 0, ..., N_eN_b - 1$ and $m = 0, ..., S_e - 1$ and in the middle line $i = 1, ..., N_r - 1$, this gives another $N_e \cdot N_b \cdot S_e \cdot (2 + N_r - 1) = 1408$ equations. Equations (136f), (136g), and (136h) describe the key expansion for AES-128 (note that $N_k = 4 \leq 6$ for AES-128). In both Equations (136f) and (136g) it holds that $\imath = 1, ..., \lceil N_b(N_r + 1)/N_k \rceil - 1$ and $m = 0, ..., S_e - 1$. Furthermore, in Equation (136f) it holds that $\jmath = 0, ..., N_e - 2$. This gives $(\lceil N_b(N_r + 1)/N_k \rceil - 1) \cdot N_e \cdot S_e = 320$ equations. In Equation (136h) it holds that $\imath = 1, ..., \lceil N_b(N_r + 1)/N_k \rceil - 1$, $\jmath = N_e, ..., N_kN_e - 1$, and $m = 0, ..., S_e - 1$, this gives $(\lceil N_b(N_r + 1)/N_k \rceil - 1) \cdot N_e(N_k - 1) \cdot S_e = 960$ equations. This means that $320 + 960 = 1280$ equations are used for the key expansion.

Furthermore, Equations (136d) and (136e) describe the S-box equations for the encryption and the key expansion respectively. When computing these substitutions, nonlinear terms are introduced. In order to get a linear system, each nonlinear term is considered as a new variable. If a similar representation (with no restriction on the degree) of the S-box is used for AES as for the S-box of Mini-AES as presented in Section 5.2, this will lead to the following number of equations and variables. To compute one output bit of the substitution, one polynomial is used and eight bits of input. Each S-box polynomial consists of (at most) $\sum_{i=2}^{2S_e-1} \binom{2S_e}{i} = \sum_{i=2}^{15} \binom{16}{i} = 246$ non-linear terms (terms of degree two) and possibly some linear terms. However, to compute all output bits of a byte, the same 246 nonlinear terms are used. So, each S-box polynomial introduces (at most) 246 new variables, but other polynomials to compute the other substituted bits in a byte do not introduce new variables.

Equations (136d) and (136e) describe the S-box equations for the encryption and the key expansion respectively. For the encryption, $\bar{x}_0, ..., \bar{x}_{N_r-1}$ need to be computed, this gives $N_r \cdot N_b \cdot N_e \cdot S_e = 1280$ equations. Which leads to $246 \cdot N_r \cdot N_b \cdot N_e = 39360$ new variables. For the key expansion $\bar{w}_0(\jmath, m), ..., \bar{w}_{\lceil N_b(N_r+1)/N_k \rceil - 2}(\jmath, m)$ for $\jmath = (N_k - 1)N_e, ..., N_kN_e - 1$ and $m = 0, ..., S_e - 1$ need to be computed, this gives $(\lceil N_b(N_r + 1)/N_k \rceil - 1) \cdot N_e \cdot S_e = 320$ equations. This leads to $246 \cdot (\lceil N_b(N_r + 1)/N_k \rceil - 1) \cdot N_e = 9840$ variables.

The above is summarized in Table 10. From this table it can be concluded that the number of variables (before linearization) equals the number of equations and since all equations are linearly independent, this means that the system is well-determined. However, the number of non-linear variables is very high and it turns out that a different representation of the S-box can result in way less non-linear terms (i.e. variables after linearization). This representation is given in Appendix D. It consists of 39 equations, with only quadratic and linear terms. Meaning that each equation will only introduce (at most) 120 variables after linearization. For the encryption this means $39 \cdot N_r \cdot N_b \cdot N_e = 6240$ equations instead of 1280 and $120 \cdot N_r \cdot N_b \cdot N_e = 19200$ variables instead of 39360. For the key expansion this means $39(\lceil N_b(N_r + 1)/N_k \rceil - 1) \cdot N_e = 1560$ equations instead of 320 and $120 \cdot (\lceil N_b(N_r + 1)/N_k \rceil - 1) \cdot N_e = 4800$ variables instead of 9840.

The above is also summarized in Table 10. From this table it can be concluded that the number

		Key expansion		Encryption		Total
		Linear	Non linear	Linear	Non linear	
S-box with high degree	Equations	1280	320	1408	1280	4288
	Variables	1728	9840	2560	39360	53488
S-box with degree two	Equations	1280	1560	1408	6240	10488
	Variables	1728	4800	2560	19200	28288

Table 10: AES-128 equations and variables

of equations is higher than the number of variables, before linearization. This means that not all equations are linearly independent anymore. However, since the number of variables after linearization exceeds the number of equations, this does not really matter since it is clear that after linearization the system is under-determined.

12 Applying HHL to AES-128

Similar to Section 9.2, this section will investigate the bounds on the dimensions of the four Macaulay matrices for all three strategies. Based on these bounds, an estimation for κ is derived. This bound on κ is used to compare the complexity of the attack described in this report to a Grover's search attack on the AES-128 key. The literature [11] claims that, depending on the Hamming weight of the solution, a Grover search is more efficient than the HHL algorithm. This section investigates what this claim means for a known plaintext attack on AES. Therefore, the most efficient Macaulay matrix is used and all variables are chosen as optimistic as possible (resulting in the lowest complexity). The reason behind this is that if the complexity of the most optimistic case of this attack is higher than the complexity of the complexity of Grover's search (which considers an average case), then it can be concluded that Grover's search would be the better choice.

12.1 Input of HHL

According to Table 10, AES-128 results in $n_1 = 1728 + 2560 = 4288$ variables, $1280 + 1408 = 2688$ linear equations, and $1560 + 6240 = 7800$ non-linear equations, so $r_1 = 2688 + 7800 = 10488$ equations in total. The linear equations consist of 2368 equations with three terms and 320 equations with four terms. The non-linear equations come from 200 byte substitutions, each of which needs 39 equations with a different number of terms. This is summarized in Table 11.

Recall Table 7 for the trivial bounds on the number of columns of the four Macaulay matrices. Combining this with Table 11 concludes that Strategy 3 will create a Macaulay matrix on $n_3 = 4288 + 1.27 \times 10^{22} = 1.27 \times 10^{22}$ variables and $r_3 = 10488 + 1.27 \times 10^{22} = 1.27 \times 10^{22}$ equations and Strategy 4 will create a Macaulay matrix on $n_4 = 4288 + 1395 = 5683$ variables and $r_4 = 10488 + 1395 = 11833$ equations. This results in dimensions of the Macaulay matrices as summarized in Table 12.

The conclusions drawn from Table 9, also hold for Table 12. This means that for AES:

- Macaulay matrices that remove squares are more efficient than Macaulay matrices that use max degree and field equations.
- No hard conclusion can be made about Macaulay matrices that use field equations and total degree.
- Strategy 3 is the least efficient.
- When using max degree and removing squares, Strategy 2 is more efficient than Strategy 4.
- When using max degree and field equations Strategy 2 is more efficient, for similar values of D, than Strategy 4.
- When using total degree, for small values of D Strategy 4 is more efficient and for large values of D Strategy 2 is more efficient.

Next subsection will use the most efficient Macaulay matrices. It distinguishes between max degree and total degree. According to Table 12, the most efficient Macaulay matrix for max degree results from Strategy 2 and removing squares (6.6×10^{1290} columns) and the most efficient Macaulay matrix for total degree results from Strategy 4 and removing squares (5.7×10^3 columns).

		Over GF(2)		Over $\mathbb{C}$	
		# eq.	# terms	# eq.	# terms
Linear equations		2368	3	2368	6
		320	4	320	34
Per nibble sub (200 in total)		3	24	3	$3.56 \cdot 10^9$
		1	$24+1$	1	$1.25 \cdot 10^9$
		3	25	3	$15.08 \cdot 10^9$
		2	26	2	$23.21 \cdot 10^9$
		1	27	1	$98.67 \cdot 10^9$
		3	28	3	$151.53 \cdot 10^9$
		1	$28+1$	1	$52.86 \cdot 10^9$
		1	29	1	$646.63 \cdot 10^9$
		2	30	2	$991.49 \cdot 10^9$
		1	$30+1$	1	$344.87 \cdot 10^9$
		4	31	4	$4.24 \cdot 10^{12}$
		1	$31+1$	1	$1.50 \cdot 10^{12}$
		3	32	3	$6.50 \cdot 10^{12}$
		2	$32+1$	2	$2.25 \cdot 10^{12}$
		3	33	3	$27.90 \cdot 10^{12}$
		1	$34+1$	1	$14.77 \cdot 10^{12}$
		1	35	1	$183.65 \cdot 10^{12}$
		3	36	3	$280.58 \cdot 10^{12}$
		2	37	2	$1.21 \cdot 10^{15}$
		1	$38+1$	1	$636.98 \cdot 10^{15}$

Table 11: AES-128 equations, translated to $\mathbb{C}$ and decreased sparsity

		Field equations	Remove squares
Str. 1	M_d	$7.9 \times 10^{2045} \leq \#cls \leq 1.4 \times 10^{17621}$	$\#cls = 6.6 \times 10^{1290}$
	T_d	$9.2 \cdot 10^6 \leq \#cls \leq 1.3 \times 10^{19481}$	$4.3 \times 10^3 \leq \#cls \leq 6.60 \times 10^{1290}$
Str. 2	M_d	$7.9 \times 10^{2045} \leq \#cls \leq 1.4 \times 10^{17621}$	$\#cls = 6.6 \times 10^{1290}$
	T_d	$1.27 \times 10^{22} \leq \#cls \leq 1.3 \times 10^{19481}$	$1.27 \times 10^{22} \leq \#cls \leq 6.6 \times 10^{1290}$
Str. 3	M_d	$10^{10^{21.78}} \leq \#cls \leq 10^{10^{23.46}}$	$\#cls = 10^{10^{21.59}}$
	T_d	$10^{10^{21.89}} \leq \#cls \leq 10^{10^{23.47}}$	$1.27 \times 10^{22} \leq \#cls \leq 10^{10^{21.59}}$
Str. 4	M_d	$3.0 \times 10^{2711} \leq \#cls \leq 10^{10^{7.188}}$	$\#cls = 5.7 \times 10^{1710}$
	T_d	$1.6 \times 10^7 \leq \#cls \leq 4.3 \times 10^{26514}$	$5.7 \times 10^3 \leq \#cls \leq 5.7 \times 10^{1710}$

Table 12: Number of columns of Macaulay matrices for AES-128

12.2 Estimation of κ

According to [11, Theorem 4.5], the condition number κ for Macaulay linear system $\mathcal{M}_{\mathcal{F},D}\mathbf{m}_D = \mathbf{b}_{\mathcal{F},D}$ is bounded by:

$$\kappa \geq \begin{cases} \sqrt{(D+1)^h - 1} & \text{if max degree is used} \\ \sqrt{\binom{D+h}{h} - 1} & \text{if total degree is used} \end{cases}, \tag{137}$$

where h is the Hamming weight of the solution.

Remark. Equation (137) uses that the number of solutions (called t in [11, Theorem 4.5]) is equal to one. This is a safe lower-bound that might actually by achieved by AES, i.e. there might be a plaintext-ciphertext pair with a unique key. Even if there is no such plaintext-ciphertext pair (or the instance that is tried to be solved has multiple keys that work), the lower-bound on κ still holds if this assumption is made.

Table 3 already gave trivial bounds on D: $D \geq 2$ when field equations are used and $D \geq 1$ when squares are removed. The previous subsection already established that the most efficient Macaulay matrices were created by removing squares and not by adding field equations. Therefore, only the lower bound of $D \geq 1$ for the remove squares option will be considered:

$$\kappa \geq \begin{cases} \sqrt{2^h - 1} & \approx 2^{h/2} & \text{if max degree is used} \\ \sqrt{\binom{2+h}{h} - 1} & = \sqrt{h} & \text{if total degree is used.} \end{cases} \tag{138}$$

Remark. Note that [11] uses the upper bound of $D \leq 3n$ given by [2], to evaluate this lower bound on κ. Obviously, this results in a way more pessimistic lower bound on κ. However, at this point, there is no reason to believe that $D = 3n$ is indeed needed to get a solvable Macaulay linear system. It is very well possible that AES is not a worst-case problem in that sense, i.e. a lower D might already be sufficient. Therefore, there should be reasonable doubt that the lower bound $\kappa \geq (3n)^{h/2}$, given by [11], might not hold for AES.

The only unknown in the equations above is the Hamming weight. The system described in Section 11 has $n_1 = 4288$ variables. By construction of AES, the solution of this system is expected to have a Hamming weight of approximately $n_1/2 = 2144$. This must hold since this includes the

plaintext-ciphertext, key, and all intermediate states of the AES encryption. Note that if this would not be the case, there might be a certain plaintext and key which leads to an out standing Hamming weight, which would seriously decrease the security of AES.
However, the input matrix for HHL will be the result of the Macaulay matrix created in one of the four strategies explained in Section 9. This means that some columns correspond to products of variables and some u variables are considered. Since HHL treats each column as a variable, the Hamming weight of the solution is the Hamming weight of the vector that includes all the n_1 variables, the u variables, and all the product terms.

Creating a Macaulay matrix, i.e. extending the solution vector with products of variables, decreases the Hamming weight of the solution vector, relatively speaking. If each variable is non-zero with probability $1/2$ then the product of d such variables is non-zero with the probability $1/2^d$. However, decreasing the sparsity of either the equations or the matrix, increases the Hamming weight. If polynomial $a + b + c + ...$ will be made 3-sparse, then $u_1 = a + b$ is only zero if both a and b are zero, i.e. this happens with probability $1/4$. Furthermore, $u_2 = u_1 + c$ which means that u_2 is only zero if both u_1 and c are zero, which happens with probability $1/8$. Similar reasoning can be used for the case that the matrix will be made sparse.

Since this section considers a best case and the Hamming weight of the solution to the Macaulay system is at least the Hamming weight of the solution to the system described in Section 11, an approximated Hamming weight of $h = 2144$ will be used. Combining this with Equation 138 results in the following approximation of κ:

$$\kappa \approx \begin{cases} 2^{1072} & \text{if max degree is used} \\ \sqrt{2144} \leq 2^6 & \text{if total degree is used.} \end{cases} \tag{139}$$

12.3 Comparison to Grover Search

The estimation of κ can give more insight in the complexity of HHL applied to the attack described in Section 11, which makes it possible to compare this complexity, to a known plain-text attack on AES using Grover search. The complexity of HHL is given in Equation 65. It uses $N \geq n$ where N is a power of 2. For simplicity reasons, set N equal to the number of columns of the input matrix. For the most optimistic case defined above, this means $N = 6.6 \times 10^{1290}$ when max degree is used and $N = 5.7 \times 10^3$ when max degree is used.
Furthermore, the complexity uses the sparsity of the input matrix. Note that for max degree, Strategy 2 should be used, whereas for total degree the most efficient Macaulay matrix was derived when using Strategy 4. Recall that Strategy 2 and Strategy 4 result in a 3-sparse matrix and a 6-sparse matrix respectively.
Finally, ϵ has to be at least $1/2$, so use $\epsilon = 1/2$ and combine all of this with Equation 65 and Equation 139 to find:

$$\mathcal{O}(\log(N)s^2\kappa^2/\epsilon) \approx \begin{cases} \mathcal{O}(2.97 \times 10^3 \cdot 9 \cdot 2^{2144}/\epsilon) \approx \mathcal{O}(2^{2157}) & \text{if max degree is used} \\ \mathcal{O}(8.65 \cdot 36 \cdot 2^{12}/2) \approx \mathcal{O}(2^{19.28}) & \text{if total degree is used.} \end{cases} \tag{140}$$

According to [11] the complexity of Grover search is $\mathcal{O}\left(h\sqrt{\binom{n}{h}}\right)$. However, in any brute-force kind of attack such as Grover search, only the key bits of AES will be brute-forced. This means that in case of AES-128 $n = 128$. Furthermore, when only the key bits are considered, the expected Hamming weight of the solution is $h = n/2$ as explained above. This results in a complexity of:

$$\mathcal{O}\left(\frac{n}{2}\sqrt{\binom{n}{n/2}}\right) = \mathcal{O}\left(\frac{128}{2}\sqrt{\binom{128}{128/2}}\right) = \mathcal{O}(2^{68.09}). \tag{141}$$

This $\mathcal{O}$-notation is unpolished and [13] derives the more precise complexity of 1.19×2^{86}. In an attempt to determine which attack (using HHL or Grover's search) is more efficient, this complexity is compared to Equation 140. This is difficult since the complexity of HHL is only given in a $\mathcal{O}$-notation and not precisely for a specific case. This means the actual complexity is a multiplication of the term in the $\mathcal{O}$-notation, multiplied by an unknown scalar. However, it is clear that if max degree is used, this scalar has to be at most $\approx 2^{-2100}$,which is unexpectedly small, for HHL to be the more efficient attack. On the other hand, if total degree is used and $D = 2$ is sufficient, HHL is more efficient than Grover's search, if this scalar is at most 1.446×10^{20}, which is way more conventional.

13 Discussion

The goal of this report is to investigate the complexity of a known-plaintext attack on AES using HHL. The most important thing to note is that this report chooses a specific method (Macaulay matrices) to translate a known-plaintext attack on AES to an input for HHL. This means that the outcome of this complexity only holds for this method and not for a known-plaintext attack on AES using HHL. It might be possible to use a different method which leads to a different, maybe lower, complexity. Therefore, the complexity discussed in this report should be seen as an upper-bound on the complexity of a known-plaintext attack on AES using HHL and should not be used to blindly conclude that AES is quantum-safe with respect to the HHL algorithm.

To take a step back out from the Macaulay matrices, Sections 13.1 and 13.2 shortly point out some important aspects that this report did not consider (in detail) so far, but are very important for everyone who wants to construct a practical attack against AES using HHL. During the study on HHL needed for this report, other ideas for applications of the HHL algorithm did arise. Some of these are mentioned in Section 13.3.

13.1 Input of HHL

In this report's attack, the input matrix of HHL is a Macaulay matrix. It is not yet proven that the Macaulay matrix meets all the requirements of the input matrix of HHL: it might not be Hermitian. However, Section 3.4 describes that the HHL algorithm can be adjusted as necessary.

Furthermore, it is worth noting that creating these matrices on a classical computer will be problematic because of memory issues. Theoretically, a classical computer should be able to store $n \times n$ matrices for n up to 2^{128}, in practice it will only be doable for n up to 2^{64} or maybe 2^{80}. This means that independent on D, the matrices using max degree will not fit in memory, the same for the matrices using field equations and total degree in Strategy 3. The other matrices only fit for very limited values of D. However, it is not needed to create the entire Macaulay matrix on a classical computer. The only requirement is that the matrix can be queried efficiently. This holds for the Macaulay matrix described in [2]. This report does some adjustments to their Macaulay matrix, that are not expected to influence the query complexity. To find an entry, one can simply multiply the desired polynomial by the desired monomial and look for the coefficient of this monomial in the resulting polynomial. Which set of monomials is used, max degree or total degree, does not influence this process. Removing squares instead of adding field equations does require an extra step: the result of every multiplication should be simplified before the coefficient of the desired polynomial is determined ($x^2 + x \rightarrow 2x$, so the coefficient of x is two and not one). However, these simplifications can be done efficiently and therefore this will likely not influence the query complexity.

Besides, the Macaulay matrix can possibly be improved to decrease the complexity of the HHL algorithm. For instance, the complexity depends on κ which is believed to be related to the Hamming weight of the solution. This Hamming weight can perhaps be decreased by combining this attack with a brute-force attack, guessing some bits. Every one that is guessed correctly, does not only decrease the size of the matrix but it especially decreases the Hamming weight of the solution. Every zero that is guessed correctly, decreases the number of non-zeros in the Macaulay matrix drastically, since guessing only one zero will already lead to several monomials that will be zero. It would be very interesting to investigate how the dimensions of the smallest solvable matrix would behave as a function of the number of guessed bits and how this translates to the overall complexity of this attack.

One last note on the Macaulay matrices should be made: this report considers the smallest solvable Macaulay matrix to be the most efficient Macaulay matrix. However, the complexity of HHL does

not only depend on the size of the matrix, it also depends on the condition number of the matrix. It might be possible that the condition number of a certain small matrix is so much bigger than the condition number of a certain larger matrix, that HHL applied to the larger matrix will be more efficient than HHL applied to the smaller matrix. With the limited knowledge that there is about condition numbers of matrices, it is not possible to draw any conclusions on this. Therefore, this report assumes that the size of the matrices is dominant in comparing the complexity of applying HHL to different matrices.

HHL also requires some ϵ as input. What value should be chosen depends on the desired output and is discussed in the next subsection.

13.2 Output of HHL

This report does not consider the output of the HHL algorithm at all. The output of HHL is a quantum state $|x\rangle$. Since x should contain all the key bits (and other variables), this x should be boolean. To make sure that the output is boolean, one should use the boolean variant of the HHL algorithm [2].

When this $|x\rangle$ is created, there are two options to use this; either get classical information about x (by doing multiple rounds of HHL and measurements on $|x\rangle$), or use quantum state $|x\rangle$ in a quantum AES en-/decryption. However, before doing so, it is important to note that state $|x\rangle$ includes an unknown normalization factor. This normalization factor can be found by the measuring the first register multiple times to estimate the probability of obtaining $|1\rangle$. Then the normalization factor can be accounted for.

First of all, it is worth investigating applications of the key as a quantum state. In [17] a quantum version of the AES cryptosystem is explained, which uses a quantum state of the key as input. It might be possible to, instead of creating this quantum state from a classical input, copy the (right part of) the output of the HHL algorithm to this register. Further research on this part is very fitting, since this known plaintext attack on AES already assumes that the attacker has access to a quantum computer. Several practical attack scenarios are possible. For instance a MITM attack in which an attacker intercepts a plaintext-ciphertext pair, applies HHL to construct the key as a quantum state and uses this to encrypt a message of his choice to send to the receiver instead. It would be interesting to see what the influence is of the error on the key, induced by the HHL algorithm.

On the other hand, when the goal is to extract classical information about the key bits, one would need to run the HHL algorithm multiple times, because one measurement only gives limited information. Even if one would do enough measurements to extract all bits of x, this might not give the desired result as the entries of x might differ quite a bit from the desired key bits. Section 3 bounds that error of HHL by $||A\tilde{x} - b||_2 \leq \epsilon$, which is bounded by the condition number of A. However, no matter how small ϵ is chosen, this does not ensure that $\tilde{x} - x$ is small as well (consider for example an A with a zero row). This can cause a problem when one tries to extract the key bits from the output of HHL. For certain bits this output might differ more than $1/2$ from the correct key bit, making it impossible to ensure that the correct key bit is guessed correctly. Furthermore, one can only learn $\log N$ bits of x from one run of the algorithm. This problem might be overcome by running the algorithm multiple times, or not trying to get classical information about x at all. Since for AES-128 only 128 key bits need to be determined, this does not seem to influence the overall complexity too much.

Furthermore, not entire $|x\rangle$ is interesting, as it not only contains key bits, but also consists of all intermediate states. Therefore, extracting classical information about the key bits is not as complex as trying to determine all bits in $|x\rangle$. This might also be a starting point for improvements on

the HHL algorithm. The algorithm now constructs all qubits of $|x\rangle$, with a certain error. It might be possible to decrease the complexity of the algorithm by constructing only the key bits, or by allowing a greater error in the other bits.

Whether the goal is to extract classical information about x, or to use $|x\rangle$ in another quantum algorithm, it is important to choose between the regular HHL algorithm or the boolean variant discussed in Section 3.4. When classical information has to be extracted it might be more efficient to use the boolean variant as this might give smaller errors on the entries since the error would either be zero or one. However, when using $|x\rangle$ as input for another quantum algorithm, it might not be that important that the entries are boolean.

13.3 HHL applied to other cryptosystems

This report focused on breaking AES with the use of HHL. However, it is also interesting if HHL might be used to break other cryptosystems. Especially if these systems are assumed to be quantum safe. Since there seems to be a connection between the Hamming weight of the solution and κ, it is interesting to focus on cryptosystems with a low Hamming weight, such as a public-key cryptosystem via Mersenne Numbers [3]. This system is particularly interesting because the system contains a lot of linearity, just as AES. Also lattice based cryptosystems contain a lot of linearity, which means it is also worth investigating how HHL can be used to attack those systems. Furthermore, cryptographic systems are created, based on cryptographic hashes, since these are assumed to be quantum safe. Therefore, it is also interesting to see if HHL can be used to affect the security of hashes.

14 Conclusion

At this point, no hard conclusion can be made, since there is not enough information about the precise complexity of HHL applied to a specific matrix-vector system. However, it seems unrealistic that an attack using max degree will result in an attack that is more efficient than Grover's search. On the other hand, there is more hope for an attack using total degree. However, this hope is based on a best case scenario, using a lower bound on D and on κ. On the other hand, there might be another way to translate a known-plaintext attack on AES to an input for HHL, yielding a lower complexity for HHL.

So, this report does raise reasonable doubt about the bounds on κ found in the literature, questioning the conclusion that the complexity of HHL will be high, but it did not find evidence that a certain attack using HHL will be more efficient than the known Grover's search attack. Therefore, this report is on the one hand another reassurance that HHL does not harm the security of AES. On the other hand, this report underlines that it should not be forgotten that it is still not proven that a Macaulay attack using HHL is less efficient than Grover's search and that there might be other attacks possible using HHL. Therefore, the main conclusion is that further research in HHL is necessary, encouraging and above all very fascinating.

References

[1] Federal information processing standards publication 197 announcing the advanced encryption standard (aes), 2001.

[2] Quantum algorithms for boolean equation solving and quantum akgebraic attack on cryptosystems. 2021.

[3] D. Aggarwal, A. Joux, A. Parkash, and M. Santha. A new public-key cryptosystem via mersenne numbers. 2017.

[4] A. Bogdanov, L. Knudsen, G. Leander, C. Paar, A. Poschmann, M. Robshaw, Y. Seurin, and C. Vikkelsoe. Present: An ultra-lightweight block cipher. 2007.

[5] S. Brown and Z. Vranesic. *Fundamentals of digital logic with VHDL design*. McGraw-Hill, 2009.

[6] Y.-A. Chen and X.-S. Gao. Quantum algorithms for boolean equation solving and quantum akgebraic attack on cryptosystems. 2017.

[7] R. Chung and W. Phan. Mini advanced encryption standard (mini-aes): A testbed for cryptanalysis students, 2002.

[8] N. T. Courtois and J. Pieprzyk. Cryptanalysis of block ciphers with overdefined systems of equations. 2002.

[9] R. de Wolf. Quantum computing: Lecture notes. 2019.

[10] D. Dervoric, M. Herbster, P. Mountney, S. Severini, N. Usher, and L. Wossnig. Quantum linear systems algorithms: a primer. 2018.

[11] J. Ding, V. Gheorghiu, A. Gilyén, S. Hallgren, and J. Li. Limitations of the macaulay matrix approach for using the hhl algorithm to solve multivariate polynomial systems. 2021.

[12] J. Gallier. *Discrete Mathematics*. Springer, 2011.

[13] M. Grassl, B. Langenberg, M. Roetteler, and R. Steinwandt. Applying grover's search algorithm to aes: quantum resource estimates. 2015.

[14] B. S. H. Arabnezhad-Khanoki and J. Pieprzyk. S-boxes representation and efficiency of algebraic attack. 3 2019.

[15] A. W. Harrow, A. Hassidim, and S. Lloyd. Quantum algorithm for solving linear systems of equations. 2009.

[16] M. A. Nielsen and I. L. Chuang. *Quantum computing and quantum information*. Cambridge University Press, 2020.

[17] B. G. S. *Selected Constructive and Destructive Approaches to Post-Quantum Cryptography*. Printservice Technische Universiteit Eindhoven, 2019.

[18] R. D. V. Buzek and S. Massar. Optimal quantum clocks. 12 1998.

Appendices

A AES matrices

Matrix $M \in GF(2^{S_e})^{N_e N_b}$ for mixColumn function:

$$M_0 = \begin{pmatrix} 3 & 2 \\ 2 & 3 \end{pmatrix} \in GF(2^4)^{2\times 2} \qquad \text{for Mini-AES}$$

$$M_1 = \begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix} \in GF(2^8)^{4\times 4} \qquad \text{for AES.}$$

Look-up table S-box, with elements in $GF(2^{S_e})^{2^{S_e}}$ for eltSub function. Note that for Mini-AES, a bit representation is used, whereas for AES a hexadecimal representation is used: 13

Mini-AES		x_0x_1			
		00	01	10	10
x_2x_3	00	1110	0010	0011	0101
	01	0100	1111	1010	1001
	10	1101	1011	0110	0000
	11	0001	1000	1100	0111

Table 13: S-box for Mini-AES

AES		$x_0x_1x_2x_3$															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$x_4x_5x_6x_7$	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Table 14: S-box for AES

B Algorithms

This appendix states generic algorithms that describe both AES and Mini-AES, denoted by (Mini-)AES, as well as algorithms solely used for AES or Mini-AES.

Algorithm 2: (Mini-)AES encryption

Input: $N_r \in \mathbb{R}, p \in GF(2^{S_e})^{N_e N_b}, k \in GF(2^{S_e})^{N_k N_e}$

Set global variables such that other functions can use them

- **if** $N_k = 2$ **then**
  - $sBox \leftarrow sBox_0$
  - $\alpha \leftarrow \alpha_0$
  - $w \leftarrow$ Mini-AESkeyExpansion (k, N_r, N_k)
- **else**
  - $sBox \leftarrow sBox_1$
  - $\alpha \leftarrow \alpha_1$
  - **Determine round constants**
    - $rcon_1 = \begin{pmatrix} 01_{16} & 00_{16} & 00_{16} & 00_{16} \end{pmatrix}^T$
    - **for** $i = 2, ..., \lceil N_b(N_r + 1)/N_k \rceil$ **do**
      - **if** $rcon_{i-1}[0] < 80B_{16}$ **then**
        - $rcon_i = \begin{pmatrix} 2rcon_{i-1}[0] & 00_{16} & 00_{16} & 00_{16} \end{pmatrix}$
      - **else**
        - $rcon_i = \begin{pmatrix} 2rcon_{i-1}[0] \oplus 11B_{16} & 00_{16} & 00_{16} & 00_{16} \end{pmatrix}$
  - $w \leftarrow$ Mini-AESkeyExpansion (k, N_r, N_k)

Encryption

- **Initial round** $(r = 0)$
  - $x_0 \leftarrow \text{addRoundKey}(p, w_0)$
- **Middle round(s)**
  - **for** $i = 1, ..., N_r - 1$ **do**
    - $\bar{x}_{i-1} \leftarrow$ eltSub (x_{i-1})
    - $x_i \leftarrow \text{mixColumns}(\text{rotation}(\bar{x}_{i-1})) \oplus w_i$
- **Final round** $(i = N_r)$
  - $\bar{x}_{N_r-1} \leftarrow \text{eltSub}(x_{N_r-1})$
  - $x_{N_r} \leftarrow \text{rotation}(\bar{x}_{N_r-1}) \oplus w_{N_r}$
  - $c \leftarrow x_{N_r}$

Output: ciphertext $c \in GF(2^{S_e})^{N_e N_b}$

Algorithm 3: AES keyExpansion

Input: $N_r \in \mathbb{N}, N_k \in \mathbb{N}, k \in GF(2^{S_e})^{N_k}$

First words of the round keys are copies of the words of the key

for $\jmath = 0, ..., N_k N_e - 1$ **do**

$w_0(\jmath, :) \leftarrow k(\jmath, :)$

Other words of the round keys

for $\imath = 1, ..., \lceil N_b(N_r + 1)/N_k \rceil - 1$ **do**

$w_\imath(0 : N_e, :) \leftarrow \text{invSub}(\text{rotword}(w_{\imath-1}(-N_e : 0, :))) \oplus rcon_\imath(:, :) \oplus w_{\imath-1}(0 : N_e, :)$

for $l = 1, ..., N_k - 1$ **do**

if $N_k > 6 \ \wedge \ l = 4 \mod N_k$ **then**

$w_\imath(lN_e : (l+1)N_e, :) \leftarrow \text{invSub}(w_\imath((l-1)N_e : lN_e, :)) \oplus w_{\imath-1}(lN_e : (l+1)N_e, :)$

else

$w_\imath(lN_e : (l+1)N_e, :) \leftarrow w_\imath((l-1)N_e : lN_e, :) \oplus w_{\imath-1}(lN_e : (l+1)N_e, :)$

Output: extended key $w \in GF(2^{S_e})^{\lceil N_e N_b (N_r+1)/N_k \rceil N_k}$

Algorithm 4: Mini-AES keyExpansion

Input: $N_r \in \mathbb{R}, N_k \in \mathbb{R}, k \in GF(2^{S_e})^{N_k}$

First nibbles of the round keys are copies of the nibbles of the key

for $j = 0, ..., N_e N_b - 1$ **do**

$w(j, :) \leftarrow k(j, :)$

Other words of the round keys

for $j = N_e N_b, ..., N_e N_b (N_r + 1) - 1$ **do**

if $j = 0 \mod N_e N_b$ **then**

$w(j, :) \leftarrow \text{eltSub}(w(j - 1, :)) \oplus rcon_{j/N_e N_b} \oplus w(j - N_e N_b, :)$

else

$w(j, :) \leftarrow w(j - 1, :) \oplus w(j - N_e N_b, :)$

Output: extended key $w \in GF(2^{S_e})^{N_e N_b (N_r+1)}$

Algorithm 5: AES rotWord

Input: word $s(j : j + N_e, :) \in GF(2^{S_e})^{N_e}$

for $k = 0, ..., N_e - 1$ **do**
  $s(j + k, :) = s(j + k + 1 \mod N_e, :)$

Output: word $s(j : j + N_e, :) \in GF(2^{S_e})^{N_e}$

Algorithm 6: (Mini-)AES rotation

Input: state matrix $s \in GF(2^{S_e})^{N_e N_b}$

for $l = 0, ..., N_e - 1$ **do**
  for $j = 0, ..., N_b - 1$ **do**
    $(s)_{l,j} \leftarrow (s)_{l,j+l \mod N_b}$

Output: $s \in GF(2^{S_e})^{N_e N_b}$

Algorithm 7: (Mini-)AES addRoundkey

Input: state matrix $s \in GF(2^{S_e})^{N_b \times N_b}$, round key $w_i \in GF(2^{S_e})^{N_b}$

for $j = 0, ..., N_b - 1$ **do**
  for $m = 0, ..., S_e$ **do**
    $x_i(j, m) \leftarrow s_i(j, m) \oplus w_i(j, m)$

Output: state matrix $s \in GF(2^{S_e})^{N_b \times N_b}$

Algorithm 8: (Mini-)AES eltSub

Input: collection of elements, might be state $s \in GF(2^{S_e})^{N_e N_b}$, might be single element $s \in GF(2^{S_e})$ etc.

for *every element* e *in* s **do**
  $e \leftarrow sBox_e$
  Note that this is equal to computing an inverse first and then applying an affine transformation.

Output: s of the same size as the input

Algorithm 9: (Mini-)AES mixColumns

Input: state matrix $s \in GF(2^{S_e})^{N_e N_b}$, matrix $M \in GF(2^{S_e})^{N_e N_b}$

for $j = 0, ..., N_e N_b - 1$ **do**
  compute matrix vector multiplication

$$s(j, :) \leftarrow \sum_{j'=0}^{N_b - 1} M((j \mod N_b) + N_b j', :) \cdot s(\lfloor j/N_e \rfloor N_e + j', :)$$

Output: s of the same size as the input

C Mini-AES S-box

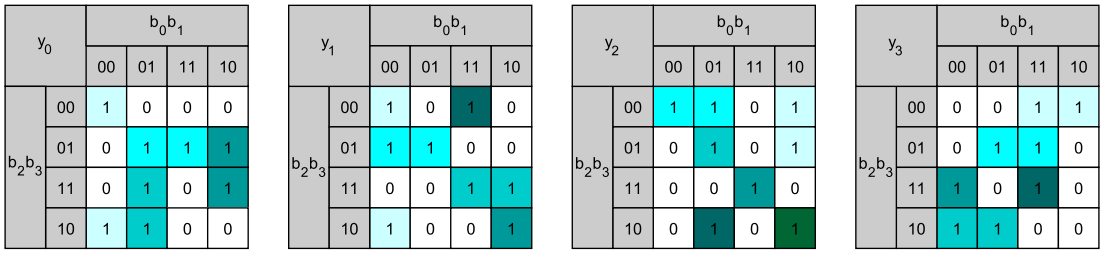


Figure 16: Karnaugh maps for S-box of Mini-AES

The S-box of Mini-AES, written as polynomials of degree two:

$$
\begin{aligned}
0 =& b_0b_3 + b_0y_1 + b_0y_2 + b_0y_3 \\
0 =& b_0b_1 + b_0b_3 + b_1b_3 + b_0y_2 + b_2 + y_3 \\
0 =& b_0b_3 + b_1y_0 + b_0y_2 + b_0 + b_1 + b_2 + b_3 + y_0 + y_3 + 1 \\
0 =& b_0b_1 + b_1b_2 + b_0y_0 + b_1y_1 + b_0y_2 + b_0 + b_1 + b_2 + b_3 + y_0 + y_3 + 1 \\
0 =& b_0b_1 + b_0b_2 + b_0b_3 + b_0y_1 + b_0y_2 + b_1y_2 + b_0 + b_1 + b_2 + y_0 + y_1 + y_2 + 1 \\
0 =& b_0b_1 + b_0b_2 + b_1b_2 + b_0y_0 + b_0y_1 + b_0y_2 + b_1y_3 + b_2 + y_3 \\
0 =& b_0b_2 + b_0b_3 + b_2b_3 + b_1 + b_2 + y_1 + y_3 + 1 \\
0 =& b_0b_2 + b_2y_0 + b_0y_1 + b_0y_2 + b_0 + b_1 + b_2 + y_0 + y_2 + y_3 \\
0 =& b_0b_1 + b_1b_2 + b_0y_1 + b_2y_1 + b_1 + b_2 + y_0 + y_2 + y_3 \\
0 =& b_0b_1 + b_1b_2 + b_0y_0 + b_0y_1 + b_2y_2 + b_2 + y_0 + y_1 + y_2 + 1 \\
0 =& b_0b_3 + b_0y_1 + b_0y_2 + b_2y_3 + b_0 + y_0 + y_1 + y_2 + 1 \\
0 =& b_0b_2 + b_0b_3 + b_0y_0 + b_3y_0 + b_0y_1 + b_0y_2 + b_2 + y_3 \\
0 =& b_0b_2 + b_3y_1 + b_1 + b_2 + b_3 + y_1 + y_3 + 1 \\
0 =& b_0b_2 + b_0y_0 + b_0y_1 + b_0y_2 + b_3y_2 + y_0 + y_1 + y_2 + y_3 + 1 \\
0 =& b_0b_2 + b_3y_3 + b_0 + b_1 + y_1 + 1 \\
0 =& b_0b_2 + b_0y_1 + y_0y_1 + b_0 + b_1 + b_2 + b_3 + y_0 + y_1 + y_2 \\
0 =& b_0y_2 + y_0y_2 + b_2 + b_3 + y_0 + y_2 + y_3 + 1 \\
0 =& b_0b_2 + b_0b_3 + b_0y_1 + b_0y_2 + y_0y_3 + b_1 + y_0 + y_2 \\
0 =& b_0b_1 + b_0b_2 + b_1b_2 + y_1y_2 + b_0 + b_3 + y_1 + y_3 \\
0 =& b_0b_1 + b_1b_2 + y_1y_3 + b_0 + b_1 + b_2 + y_1 + 1 \\
0 =& b_0b_1 + b_0b_2 + b_1b_2 + y_2y_3 + b_2 + y_3
\end{aligned}
\tag{142}
$$

D AES S-box

The literature [6] states that the S-box equations of AES can be written as polynomials of degree two:

$$0 = b_5b_7 + b_5b_6 + b_3b_7 + b_3b_6 + b_2b_4 + b_1b_7 + b_1b_6 + b_1b_5 + b_1b_3 + b_1b_2 + b_0b_7 + b_0b_3 + b_0b_2 +$$

$$
\begin{aligned}
& b_6y_7 + b_7y_6 + b_6y_6 + b_7y_5 + b_5y_5 + b_7y_4 + b_1y_4 + b_2y_3 + b_0y_3 + b_6y_2 + b_4y_2 + b_3y_2 + b_0y_2 + \\
& b_4y_0 + b_2y_0 + b_7 + b_5 + b_3 + y_7 + y_2 + y_0 + 1 \\
0 = & b_6b_7 + b_5b_7 + b_4b_7 + b_4b_6 + b_4b_5 + b_3b_4 + b_2b_5 + b_1b_7 + b_1b_6 + b_1b_5 + b_1b_4 + b_1b_3 + b_1b_2 + \\
& b_0b_5 + b_0b_1 + b_6y_6 + y_5y_7 + b_3y_4 + y_4y_7 + \\
& y_4y_5 + b_5y_3 + b_0y_3 + y_3y_6 + y_3y_4 + b_3y_2 + b_0y_2 + y_2y_4 + \\
& y_2y_3 + b_5y_0 + b_3y_0 + b_1y_0 + y_0y_7 + y_0y_3 + y_0y_1 + b_5 + b_3 + b_0 + y_2 + 1 \\
0 = & b_1y_7 + b_0y_7 + y_6y_7 + b_7y_5 + b_6y_5 + y_5y_7 + b_7y_4 + b_5y_3 + b_2y_3 + y_3y_6 + b_2y_2 + b_0y_2 + y_2y_5 + b_6y_1 + \\
& b_4y_1 + b_1y_1 + y_1y_2 + b_6y_0 + b_5y_0 + b_4y_0 + y_0y_7 + y_0y_6 + y_0y_5 + y_0y_4 + y_0y_3 + b_3 + b_1 + y_3 + y_2 + y_1 + 1 \\
0 = & b_6b_7 + b_4b_6 + b_3b_7 + b_2b_7 + b_1b_4 + b_0b_6 + b_0b_3 + b_6y_7 + b_4y_7 + b_3y_7 + b_7y_6 + b_3y_6 + b_7y_5 + \\
& b_7y_4 + b_1y_4 + b_5y_3 + b_4y_3 + b_1y_3 + b_6y_2 + b_2y_2 + b_6y_1 + b_5y_1 + b_3y_1 + b_1y_1 + b_0y_1 + b_7y_0 + \\
& b_6y_0 + b_5y_0 + b_3y_0 + b_2y_0 + b_1y_0 + b_6 + b_1 \\
0 = & b_6y_7 + b_5y_7 + b_1y_7 + b_0y_7 + b_5y_6 + b_4y_6 + b_3y_6 + b_4y_4 + b_3y_4 + b_2y_4 + b_4y_3 + b_3y_3 + y_3y_5 + \\
& b_2y_2 + y_2y_7 + y_2y_4 + y_2y_3 + b_7y_1 + b_4y_1 + b_3y_1 + b_1y_1 + y_1y_7 + y_1y_6 + y_1y_5 + y_1y_2 + b_7y_0 + b_5y_0 + \\
& b_4y_0 + b_3y_0 + y_0y_4 + y_0y_3 + y_0y_2 + b_6 + b_4 + b_3 + y_2 \\
0 = & b_2y_7 + y_5y_6 + b_1y_4 + b_7y_3 + b_2y_3 + b_1y_3 + b_0y_3 + y_3y_6 + y_3y_4 + b_4y_2 + b_2y_2 + b_0y_2 + y_2y_6 + \\
& y_2y_5 + y_2y_3 + b_5y_1 + b_3y_1 + y_1y_7 + y_1y_5 + y_1y_4 + y_1y_3 + y_1y_2 + b_5y_0 + b_4y_0 + b_3y_0 + b_0y_0 + y_0y_6 + \\
& y_0y_1 + b_7 + b_6 + b_3 + b_2 + b_1 + y_4 + y_2 + y_1 + y_0 \\
0 = & b_5y_7 + b_3y_6 + b_2y_6 + b_0y_6 + b_6y_5 + b_5y_5 + b_0y_5 + b_6y_4 + b_5y_4 + b_3y_4 + b_2y_4 + b_0y_4 + y_4y_7 + \\
& y_3y_6 + y_3y_5 + b_3y_2 + b_2y_2 + b_0y_2 + y_2y_7 + y_2y_6 + y_2y_5 + b_3y_1 + b_2y_1 + y_1y_7 + y_1y_6 + y_1y_3 + b_5y_0 + \\
& b_4y_0 + b_1y_0 + y_0y_7 + y_0y_2 + b_6 + b_1 + y_4 + y_3 + y_1 \\
0 = & b_6y_7 + b_3y_7 + b_0y_7 + b_2y_6 + b_4y_4 + b_2y_4 + b_0y_4 + b_7y_3 + b_6y_3 + b_3y_3 + b_2y_3 + b_1y_3 + b_0y_3 + \\
& b_5y_2 + b_2y_2 + b_1y_2 + b_3y_1 + b_2y_1 + b_1y_1 + b_3y_0 + b_6 + b_2 + b_1 + b_0 + y_2 \\
0 = & b_7y_7 + b_4y_7 + b_1y_7 + b_7y_6 + b_6y_6 + b_1y_6 + y_6y_7 + b_7y_5 + b_6y_5 + b_2y_5 + b_0y_5 + b_6y_4 + b_4y_4 + \\
& b_2y_4 + y_4y_5 + b_4y_3 + b_3y_3 + b_2y_3 + b_1y_3 + b_0y_3 + y_3y_6 + y_3y_5 + b_3y_2 + y_2y_4 + b_6y_1 + b_5y_1 + b_4y_1 + \\
& y_1y_5 + y_1y_2 + b_6y_0 + b_2y_0 + b_1y_0 + y_0y_6 + y_4 + y_3 \\
0 = & b_4y_7 + b_3y_7 + b_4y_6 + b_2y_6 + b_1y_6 + b_0y_4 + b_3y_3 + b_1y_3 + b_0y_3 + b_7y_2 + b_3y_2 + b_2y_2 + b_1y_2 + \\
& b_0y_2 + b_7y_1 + b_6y_1 + b_5y_1 + b_4y_1 + b_3y_1 + b_0y_1 + b_4 + b_2 + b_1 + y_2 \\
0 = & b_3b_6 + b_2b_5 + b_1b_4 + b_1b_2 + b_0b_4 + b_0b_1 + b_4y_6 + b_2y_6 + b_1y_6 + b_7y_5 + b_7y_4 + b_2y_4 + b_6y_3 + \\
& b_4y_3 + b_3y_3 + b_2y_3 + b_7y_2 + b_0y_2 + b_4y_1 + b_3y_1 + b_5y_0 + b_2y_0 + b_1y_0 + b_4 + b_1 + y_4 + y_3 + y_0 + 1 \\
0 = & b_4b_7 + b_2b_3 + b_1b_5 + b_1b_4 + b_0b_7 + b_0b_6 + b_0b_5 + b_0b_4 + b_0b_1 + b_7y_7 + b_4y_7 + b_1y_6 + b_2y_4 + \\
& b_1y_4 + b_0y_4 + b_7y_3 + b_4y_3 + b_3y_3 + b_4y_2 + b_3y_2 + b_0y_2 + b_7y_1 + b_5y_1 + b_2y_1 + b_1y_1 + b_0y_1 + \\
& b_1y_0 + b_0y_0 + b_7 + b_1 + b_0 + y_2 \\
0 = & b_6b_7 + b_4b_5 + b_3b_7 + b_3b_5 + b_2b_5 + b_2b_4 + b_2b_3 + b_1b_7 + b_0b_6 + b_0b_4 + b_2y_7 + b_0y_7 + b_1y_6 + \\
& b_6y_5 + b_2y_4 + b_6y_3 + b_5y_3 + b_2y_3 + b_6y_2 + b_4y_2 + b_3y_2 + b_2y_2 + b_1y_2 + b_7y_1 + b_5y_1 + b_6y_0 + \\
& b_5y_0 + b_4y_0 + b_3y_0 + b_0y_0 \\
0 = & b_5b_7 + b_3b_6 + b_1b_7 + b_1b_2 + b_0b_4 + b_0b_3 + b_1y_7 + b_2y_6 + b_1y_6 + b_6y_5 + b_4y_5 + b_2y_5 + b_0y_4 + \\
& b_5y_3 + b_2y_3 + b_6y_2 + b_5y_2 + b_1y_2 + b_0y_2 + b_7y_1 + b_6y_0 + b_5y_0 + b_0y_0 + b_6 + b_1 + y_6 + y_2 + y_1 \\
0 = & b_1y_7 + b_0y_7 + b_2y_6 + b_6y_5 + b_2y_5 + b_4y_4 + b_3y_4 + b_2y_4 + b_1y_4 + b_0y_4 + b_5y_3 + b_2y_3 + b_1y_3 + \\
& b_7y_2 + b_4y_2 + b_3y_2 + b_1y_2 + b_6y_1 + b_3y_1 + b_2y_1 + b_1y_1 + b_0y_1 + b_5y_0 + b_0y_0 + b_3 + y_2 \\
0 = & b_5b_7 + b_3b_6 + b_1b_7 + b_1b_2 + b_0b_4 + b_0b_3 + b_6y_7 + b_3y_7 + b_0y_7 + b_4y_6 + b_5y_5 + b_2y_5 + b_4y_4 + \\
& b_3y_3 + b_1y_2 + b_0y_2 + b_7y_1 + b_5y_1 + b_4y_1 + b_2y_1 + b_7y_0 + b_6y_0 + b_1y_0 + b_6 + b_5 + b_4 + b_2 + b_0 + y_7
\end{aligned}
$$

$$
\begin{aligned}
0 =& b_7y_7 + b_7y_6 + b_6y_6 + b_4y_6 + b_2y_6 + b_1y_6 + b_7y_5 + b_2y_5 + b_7y_4 + b_1y_4 + b_0y_4 + y_4y_6 + b_7y_3 + \\
& b_3y_3 + y_3y_5 + y_3y_4 + b_7y_2 + b_4y_2 + b_3y_2 + y_2y_4 + y_2y_3 + y_1y_6 + b_3y_0 + b_2y_0 + b_1y_0 + y_0y_6 + y_0y_4 + \\
& y_0y_2 + y_0y_1 + b_5 + b_4 + b_3 + b_1 + y_4 + y_2 + y_1 \\
0 =& b_7y_6 + y_6y_7 + b_7y_5 + b_3y_5 + y_5y_7 + b_7y_4 + b_0y_4 + y_4y_7 + y_4y_5 + b_6y_3 + b_4y_3 + b_3y_3 + b_1y_3 + \\
& y_3y_7 + y_3y_6 + y_3y_5 + y_3y_4 + b_6y_2 + b_5y_2 + b_4y_2 + b_1y_2 + b_0y_2 + y_2y_5 + b_6y_1 + b_5y_1 + b_4y_1 + b_0y_1 + \\
& y_1y_7 + y_1y_4 + y_1y_2 + b_5y_0 + b_4y_0 + y_0y_7 + y_0y_5 + y_0y_4 + b_0 + y_0 \\
0 =& b_6y_6 + b_1y_6 + b_5y_5 + b_3y_4 + b_0y_4 + b_7y_3 + b_6y_3 + b_5y_3 + b_1y_3 + b_0y_3 + b_7y_2 + b_7y_1 + b_6y_1 + \\
& b_4y_1 + b_3y_1 + b_0y_1 + b_4y_0 + b_2y_0 + b_7 + b_6 + b_4 + b_3 + b_0 + y_2 \\
0 =& b_6b_7 + b_5b_7 + b_4b_6 + b_3b_7 + b_2b_7 + b_2b_5 + b_1b_7 + b_0b_6 + b_0b_1 + b_7y_7 + b_3y_7 + b_1y_7 + b_5y_6 + b_0y_4 + \\
& b_6y_3 + b_1y_3 + b_7y_2 + b_6y_2 + b_5y_2 + b_4y_2 + b_3y_2 + b_4y_1 + b_2y_1 + b_5y_0 + b_3y_0 + b_5 + b_2 + b_1 + b_0 + y_3 + y_2 \\
0 =& b_0y_7 + b_2y_6 + b_0y_6 + b_5y_5 + b_0y_5 + b_6y_4 + b_3y_4 + b_2y_4 + b_0y_4 + b_7y_3 + b_6y_3 + b_4y_3 + b_3y_3 + \\
& b_2y_3 + b_4y_2 + b_3y_2 + b_4y_1 + b_2y_1 + b_7y_0 + b_2y_0 + b_0y_0 + b_7 + b_4 + b_2 + b_1 + y_4 + y_1 + y_0 \\
0 =& b_5b_7 + b_5b_6 + b_4b_6 + b_3b_5 + b_2b_6 + b_2b_5 + b_1b_7 + b_1b_6 + b_0b_7 + b_0b_6 + b_0b_3 + b_0b_1 + b_6y_7 + b_3y_6 + \\
& b_3y_4 + b_0y_4 + b_7y_3 + b_4y_3 + b_3y_3 + b_5y_2 + b_4y_2 + b_6y_1 + b_4y_1 + b_2y_1 + b_1y_1 + b_1y_0 + b_5 + b_0 + y_4 + y_2 \\
0 =& b_5b_7 + b_5b_6 + b_3b_7 + b_3b_4 + b_2b_6 + b_2b_4 + b_1b_4 + b_1b_3 + b_1b_2 + b_0b_6 + b_7y_7 + b_6y_7 + b_4y_7 + \\
& b_3y_7 + b_1y_6 + b_1y_4 + b_7y_3 + b_3y_3 + b_2y_3 + b_6y_2 + b_2y_2 + b_0y_2 + b_7y_1 + b_5y_1 + b_4y_1 + b_3y_1 + \\
& b_6y_0 + b_3y_0 + b_1y_0 + b_6 + b_3 + b_2 + y_4 \\
0 =& b_5b_7 + b_4b_6 + b_4b_5 + b_3b_5 + b_2b_7 + b_2b_4 + b_2b_3 + b_0b_4 + b_0b_1 + b_4y_6 + b_2y_6 + b_7y_5 + b_7y_4 + \\
& b_6y_4 + b_1y_4 + b_6y_3 + b_5y_3 + b_2y_3 + b_1y_3 + b_7y_2 + b_5y_2 + b_7y_1 + b_6y_1 + b_0y_1 + b_4y_0 + b_3y_0 + \\
& b_1y_0 + b_3 + b_1 + y_4 + y_3 \\
0 =& b_5b_7 + b_4b_6 + b_3b_7 + b_3b_5 + b_3b_4 + b_2b_4 + b_1b_6 + b_1b_3 + b_1b_2 + b_0b_7 + b_4y_7 + b_3y_7 + b_0y_7 + \\
& b_4y_6 + b_2y_6 + b_0y_6 + b_5y_5 + b_0y_5 + b_6y_4 + b_6y_3 + b_6y_2 + b_4y_2 + b_3y_2 + b_1y_2 + b_5y_1 + b_4y_1 + \\
& b_5y_0 + b_1y_0 + b_0y_0 + b_7 + b_5 + y_1 + y_0 \\
0 =& b_6b_7 + b_4b_6 + b_4b_5 + b_2b_6 + b_2b_5 + b_2b_3 + b_1b_7 + b_1b_5 + b_1b_4 + b_1b_3 + b_0b_2 + b_1y_7 + b_2y_6 + b_1y_6 + \\
& b_6y_5 + b_2y_5 + b_2y_4 + b_3y_3 + b_7y_2 + b_3y_2 + b_2y_2 + b_7y_1 + b_5y_1 + b_1y_1 + b_7 + b_3 + b_0 + y_6 + y_4 + y_2 + y_1 \\
0 =& b_7y_5 + b_7y_4 + b_2y_4 + b_7y_3 + b_6y_3 + b_0y_3 + b_5y_2 + b_4y_2 + b_0y_2 + b_7y_1 + b_3y_1 + b_2y_1 + b_0y_1 + \\
& b_7y_0 + b_5y_0 + b_4y_0 + b_2y_0 + b_1y_0 + b_2 + b_1 + b_0 + y_4 + y_2 + y_1 + 1 \\
0 =& b_5b_6 + b_3b_7 + b_3b_6 + b_2b_5 + b_2b_4 + b_1b_6 + b_1b_5 + b_1b_4 + b_1b_3 + b_1b_2 + b_0b_7 + b_0b_2 + b_0b_1 + \\
& b_5y_7 + b_1y_6 + b_4y_4 + b_2y_4 + b_0y_4 + b_5y_3 + b_4y_3 + b_1y_3 + b_0y_2 + b_5y_1 + b_2y_1 + b_1y_1 + b_3y_0 + \\
& b_0y_0 + b_6 + b_5 + b_1 + y_3 + y_2 \\
0 =& b_6y_7 + b_3y_7 + b_1y_7 + b_6y_6 + b_5y_6 + b_1y_6 + b_5y_5 + b_1y_5 + b_6y_4 + b_6y_3 + b_5y_3 + b_4y_3 + b_3y_2 + \\
& b_7y_1 + b_6y_1 + b_1y_1 + b_6y_0 + b_3y_0 + b_2y_0 + b_5 + y_7 + y_3 + y_2 + y_1 + y_0 \\
0 =& b_5b_6 + b_4b_6 + b_3b_6 + b_3b_5 + b_3b_4 + b_1b_7 + b_1b_5 + b_0b_3 + b_0b_2 + b_5y_7 + b_1y_6 + b_6y_5 + b_2y_5 + \\
& b_2y_4 + b_0y_4 + b_6y_2 + b_4y_2 + b_2y_2 + b_1y_2 + b_6y_1 + b_5y_1 + b_4y_1 + b_2y_1 + b_5y_0 + b_3y_0 + b_7 + b_6 + \\
& b_4 + b_0 + y_6 + y_4 + y_2 + 1 \\
0 =& b_5y_7 + b_4y_7 + b_3y_7 + b_1y_7 + b_0y_7 + b_7y_6 + b_4y_6 + b_7y_5 + b_3y_5 + b_7y_4 + b_4y_4 + b_3y_4 + b_7y_3 + \\
& b_6y_3 + b_4y_3 + b_3y_3 + b_1y_3 + b_6y_2 + b_5y_2 + b_2y_2 + b_2y_1 + b_7y_0 + b_3y_0 + b_2y_0 + b_1y_0 + y_6 + y_2 + y_0 \\
0 =& b_7y_7 + b_4y_7 + b_1y_7 + b_6y_6 + b_5y_6 + b_5y_5 + b_2y_3 + b_1y_3 + b_0y_3 + b_5y_2 + b_4y_2 + b_2y_2 + b_1y_2 + \\
& b_7y_1 + b_0y_1 + b_6y_0 + b_5y_0 + b_4y_0 + b_0y_0 + b_6 + b_2 + b_1 + b_0 + y_7 \\
0 =& b_7y_7 + b_4y_7 + b_2y_7 + b_4y_6 + b_6y_5 + b_5y_5 + b_1y_5 + b_6y_4 + b_4y_4 + b_7y_3 + b_6y_3 + b_6y_2 + b_5y_2 + \\
& b_0y_2 + b_6y_1 + b_2y_1 + b_6y_0 + b_4y_0 + b_5 + b_4 + b_3 + b_1 + b_0 + y_7 + y_4 + y_2
\end{aligned}
$$

$$
\begin{aligned}
0 =& b_4b_6 + b_4b_5 + b_3b_5 + b_3b_4 + b_2b_7 + b_2b_6 + b_2b_4 + b_2b_3 + b_1b_6 + b_1b_5 + b_1b_4 + b_1b_2 + b_0b_7 + \\
& b_0b_6 + b_0b_2 + b_0b_1 + b_0y_5 + b_6y_4 + b_3y_4 + b_0y_4 + b_6y_3 + b_3y_3 + b_2y_3 + b_6y_2 + b_5y_2 + b_4y_2 + \\
& b_0y_2 + b_7y_1 + b_5y_1 + b_0y_0 + b_7 + y_1 + 1 \\
0 =& b_6y_7 + y_6y_7 + b_7y_5 + y_5y_7 + b_7y_4 + y_4y_7 + b_7y_3 + b_6y_3 + b_2y_3 + b_1y_3 + y_3y_5 + y_3y_4 + b_7y_2 + \\
& b_6y_2 + b_0y_2 + y_2y_3 + b_4y_1 + b_3y_1 + b_2y_1 + b_1y_1 + b_0y_1 + y_1y_5 + y_1y_2 + b_7y_0 + b_1y_0 + y_0y_7 + y_0y_6 + \\
& y_0y_3 + y_0y_1 + b_2 + y_5 + y_4 \\
0 =& b_4y_7 + b_3y_7 + b_1y_7 + b_5y_6 + b_4y_6 + b_1y_6 + b_4y_5 + b_0y_5 + b_7y_4 + b_6y_4 + b_4y_4 + b_2y_4 + b_1y_4 + \\
& b_5y_3 + b_4y_3 + b_3y_3 + b_0y_3 + b_3y_2 + b_0y_1 + b_6y_0 + b_1y_0 + b_6 + b_4 + b_2 + y_6 + y_2 + y_1 \\
0 =& b_5b_7 + b_4b_7 + b_2b_5 + b_2b_3 + b_1b_7 + b_1b_5 + b_0b_7 + b_0b_6 + b_0b_5 + b_0b_4 + b_0b_3 + b_4y_5 + b_6y_4 + b_5y_3 + \\
& b_4y_3 + b_0y_3 + b_7y_2 + b_3y_2 + b_2y_2 + b_5y_1 + b_3y_1 + b_0y_1 + b_5y_0 + b_3y_0 + b_0y_0 + b_7 + b_5 + b_1 + b_0 + y_4 + \\
& y_0 + 1 \\
0 =& b_7y_7 + b_4y_7 + b_0y_7 + b_6y_6 + b_5y_6 + b_0y_6 + b_0y_5 + b_6y_4 + b_5y_4 + b_3y_4 + b_7y_3 + b_6y_3 + b_0y_3 + \\
& b_7y_2 + b_3y_2 + b_2y_2 + b_6y_1 + b_3y_1 + b_1y_1 + b_5y_0 + b_4y_0 + b_3y_0 + b_1y_0 + b_3 + y_3 \\
0 =& b_4b_5 + b_3b_7 + b_3b_6 + b_3b_4 + b_2b_7 + b_2b_5 + b_2b_3 + b_1b_6 + b_1b_4 + b_1b_3 + b_0b_7 + b_1y_7 + b_2y_6 + b_1y_6 + \\
& b_6y_5 + b_2y_5 + b_4y_4 + b_3y_4 + b_7y_3 + b_3y_2 + b_5y_1 + b_2y_1 + b_0y_1 + b_4y_0 + b_3y_0 + b_0y_0 + b_3 + y_3 + y_2 + \\
& y_1 + y_0
\end{aligned}
$$

Remark. Note that [6] uses x instead of b, but because x is already used as a variable for the states of (Mini-)AES this might cause confusion. Therefore, this report uses a new variable b for the input bits of the S-box.

E Proof of Lemma 2.3

Proof. It needs to be shown that $\big||\Psi_0\rangle\big| = 1$. First note that:

$$\begin{aligned}
\big||\Psi_0\rangle\big|^2 &= \left|\sqrt{\frac{2}{T}}\right|^2 \cdot \left|\sum_{\tau=0}^{T-1} \sin\!\left(\frac{\pi(\tau+\frac{1}{2})}{T}\right)|\tau\rangle\right|^2 \\
&= \frac{2}{T}\cdot\left|\left(\sin\!\left(\frac{\pi(0+\frac{1}{2})}{T}\right),\ldots,\sin\!\left(\frac{\pi((T-1)+\frac{1}{2})}{T}\right)\right)^{\top}\right|^2 \\
&= \frac{2}{T}\cdot\left|\sum_{\tau=0}^{T-1}\sin^2\!\left(\frac{\pi(\tau+\frac{1}{2})}{T}\right)\right| \\
&\text{note that } \sin(x) = \frac{1}{2} - \frac{1}{2}\cos(2x) \\
&= \frac{2}{T}\cdot\left|\sum_{\tau=0}^{T-1}\left[\frac{1}{2} - \frac{1}{2}\cos\!\left(\frac{2\pi(\tau+\frac{1}{2})}{T}\right)\right]\right| \\
&= \frac{2}{T}\cdot\left|\sum_{\tau=0}^{T-1}\left[\frac{1}{2}\right] - \sum_{\tau=0}^{T-1}\left[\frac{1}{2}\cos\!\left(\frac{2\pi(\tau+\frac{1}{2})}{T}\right)\right]\right|.
\end{aligned} \tag{143}$$

Using $\cos(x) = \frac{1}{2}(e^{ix} + e^{-ix})$, note that:

$$\begin{aligned}
\sum_{\tau=0}^{T-1}\frac{1}{2}\cos\!\left(\frac{2\pi(\tau+\frac{1}{2})}{T}\right) &= \frac{1}{4}\left(\sum_{\tau=0}^{T-1} e^{i\frac{2\pi(\tau+\frac{1}{2})}{T}} + \sum_{\tau=0}^{T-1} e^{i\frac{-2\pi(\tau+\frac{1}{2})}{T}}\right) \\
&= \frac{1}{4}\left(e^{\frac{i\pi}{T}}\sum_{\tau=0}^{T-1}\left(e^{i\frac{2\pi i}{T}}\right)^{\tau} + e^{-\frac{i\pi}{T}}\sum_{\tau=0}^{T-1}\left(e^{-i\frac{2\pi i}{T}}\right)^{\tau}\right) \\
&= \frac{1}{4}\left(e^{\frac{i\pi}{T}}\frac{1-e^{2\pi i}}{1-e^{2\pi i/T}} + e^{-\frac{i\pi}{T}}\frac{1-e^{-2\pi i}}{1-e^{-2\pi i/T}}\right) \\
&= \frac{1}{4}\left(e^{\frac{i\pi}{T}}\frac{1-1}{1-e^{2\pi i/T}} + e^{-\frac{i\pi}{T}}\frac{1-1}{1-e^{-2\pi i/T}}\right) = 0.
\end{aligned} \tag{144}$$

So, it holds that:

$$\begin{aligned}
\big||\Psi_0\rangle\big|^2 &= \frac{2}{T}\cdot\left|\sum_{\tau=0}^{T-1}\left[\frac{1}{2}\right] - \sum_{\tau=0}^{T-1}\left[\frac{1}{2}\cos\!\left(\frac{2\pi(\tau+\frac{1}{2})}{T}\right)\right]\right| \\
&= \frac{2}{T}\cdot\left|\frac{T}{2} - 0\right| = 1.
\end{aligned} \tag{145}$$

□

F Derivation of uncompute

First the relabeling is undone. So, write $|k\rangle$ instead of $|\tilde{\lambda_k}\rangle$ and substitute back Equation (38):

$$\sum_{j=1}^{N}\beta_j\sum_{k=0}^{T-1}\alpha_{k|j}\left(\sqrt{1-\frac{c^2}{\tilde{\lambda_k}^2}}\,|0\rangle+\frac{c}{\tilde{\lambda_k}}\,|1\rangle\right)\otimes|\tilde{\lambda_k}\rangle\otimes|u_j\rangle= \tag{146}$$
$$\sum_{j=1}^{N}\beta_j\sum_{k=0}^{T-1}\frac{\sqrt{2}}{T}\sum_{\tau=0}^{T-1}e^{-2\pi ik\tau/T}\sin\left(\frac{\pi(\tau+1/2)}{T}\right)\left(\sqrt{1-\frac{c^2}{\tilde{\lambda_k}^2}}\,|0\rangle+\frac{c}{\tilde{\lambda_k}}\,|1\rangle\right)\otimes$$
$$|k\rangle\otimes e^{i\lambda_j\tau t_0/T}\,|u_j\rangle\,.$$

Then the $QFT_t^\dagger$ needs to be reversed. So, apply QFT_t as defined in Definition 2.9:

$$\sum_{j=1}^{N}\beta_j\sum_{k=0}^{T-1}\frac{\sqrt{2}}{T}\sum_{\tau=0}^{T-1}e^{-2\pi ik\tau/T}\sin\left(\frac{\pi(\tau+1/2)}{T}\right)\left(\sqrt{1-\frac{c^2}{\tilde{\lambda_k}^2}}\,|0\rangle+\frac{c}{\tilde{\lambda_k}}\,|1\rangle\right)\otimes \tag{147}$$
$$|k\rangle\otimes e^{i\lambda_j\tau t_0/T}\,|u_j\rangle\xrightarrow{QFT_t}$$
$$\sum_{j=1}^{N}\beta_j\sum_{k=0}^{T-1}\frac{\sqrt{2}}{T}\sum_{\tau=0}^{T-1}e^{-2\pi ik\tau/T}\sin\left(\frac{\pi(\tau+1/2)}{T}\right)\left(\sqrt{1-\frac{c^2}{\tilde{\lambda_k}^2}}\,|0\rangle+\frac{c}{\tilde{\lambda_k}}\,|1\rangle\right)\otimes$$
$$\frac{1}{\sqrt{T}}\sum_{\tau'=0}^{T-1}e^{2\pi i\tau' k/T}\,|\tau'\rangle\otimes e^{i\lambda_j\tau t_0/T}\,|u_j\rangle=$$
$$\sum_{j=1}^{N}\beta_j\sum_{k=0}^{T-1}\frac{\sqrt{2}}{T}\frac{1}{\sqrt{T}}\sum_{\tau=0}^{T-1}\sum_{\tau'=0}^{T-1}e^{2\pi i(\tau'-\tau)k/T}\sin\left(\frac{\pi(\tau+1/2)}{T}\right)\left(\sqrt{1-\frac{c^2}{\tilde{\lambda_k}^2}}\,|0\rangle+\frac{c}{\tilde{\lambda_k}}\,|1\rangle\right)\otimes$$
$$|\tau'\rangle\otimes e^{i\lambda_j\tau t_0/T}\,|u_j\rangle\,.$$

Assuming phase estimation was perfect, the substitution $\tilde{\lambda_k}=\lambda_j$, can be done and results in:

$$\sum_{j=1}^{N}\beta_j\sum_{k=0}^{T-1}\frac{\sqrt{2}}{T}\frac{1}{\sqrt{T}}\sum_{\tau=0}^{T-1}\sum_{\tau'=0}^{T-1}e^{2\pi i(\tau'-\tau)k/T}\sin\left(\frac{\pi(\tau+1/2)}{T}\right)\left(\sqrt{1-\frac{c^2}{\tilde{\lambda_k}^2}}\,|0\rangle+\frac{c}{\tilde{\lambda_k}}\,|1\rangle\right)\otimes \tag{148}$$
$$|\tau'\rangle\otimes e^{i\lambda_j\tau t_0/T}\,|u_j\rangle=$$
$$\sum_{j=1}^{N}\beta_j\sum_{k=0}^{T-1}\frac{\sqrt{2}}{T}\frac{1}{\sqrt{T}}\sum_{\tau=0}^{T-1}\sum_{\tau'=0}^{T-1}e^{2\pi i(\tau'-\tau)k/T}\sin\left(\frac{\pi(\tau+1/2)}{T}\right)\left(\sqrt{1-\frac{c^2}{\lambda_j^2}}\,|0\rangle+\frac{c}{\lambda_j}\,|1\rangle\right)\otimes$$
$$|\tau'\rangle\otimes e^{i\lambda_j\tau t_0/T}\,|u_j\rangle=$$
$$\sum_{j=1}^{N}\beta_j\left(\sqrt{1-\frac{c^2}{\lambda_j^2}}\,|0\rangle+\frac{c}{\lambda_j}\,|1\rangle\right)\otimes\frac{\sqrt{2}}{T}\frac{1}{\sqrt{T}}\sum_{\tau=0}^{T-1}\sin\left(\frac{\pi(\tau+1/2)}{T}\right)\sum_{\tau'=0}^{T-1}\sum_{k=0}^{T-1}e^{2\pi i(\tau'-\tau)k/T}\,|\tau'\rangle\otimes$$
$$e^{i\lambda_j\tau t_0/T}\,|u_j\rangle=$$
$$\sum_{j=1}^{N}\beta_j\left(\sqrt{1-\frac{c^2}{\lambda_j^2}}\,|0\rangle+\frac{c}{\lambda_j}\,|1\rangle\right)\otimes\frac{\sqrt{2}}{T}\frac{1}{\sqrt{T}}\sum_{\tau=0}^{T-1}\sin\left(\frac{\pi(\tau+1/2)}{T}\right)\sum_{\tau'=0}^{T-1}T\cdot\delta_{\tau'\tau}\,|\tau'\rangle\otimes$$
$$e^{i\lambda_j\tau t_0/T}\,|u_j\rangle=$$
$$\sum_{j=1}^{N}\beta_j\left(\sqrt{1-\frac{c^2}{\lambda_j^2}}\,|0\rangle+\frac{c}{\lambda_j}\,|1\rangle\right)\otimes\sqrt{\frac{2}{T}}\sum_{\tau=0}^{T-1}\sin\left(\frac{\pi(\tau+1/2)}{T}\right)|\tau\rangle\otimes e^{i\lambda_j\tau t_0/T}\,|u_j\rangle\,.$$

Since, $\sum_k \delta_{k\tau} f(k) = f(\tau)$ and $f(A)\left|u_j\right\rangle = f(\lambda_j)\left|u_j\right\rangle$, it holds that:

$$\sum_{j=1}^{N} \beta_j \left(\sqrt{1-\frac{c^2}{\lambda_j^2}}\left|0\right\rangle + \frac{c}{\lambda_j}\left|1\right\rangle \right) \otimes \sqrt{\frac{2}{T}} \sum_{\tau=0}^{T-1} \sin\left(\frac{\pi(\tau+1/2)}{T}\right) \left|\tau\right\rangle \otimes e^{i\lambda_j \tau t_0/T} \left|u_j\right\rangle = \tag{149}$$

$$\sum_{j=1}^{N} \beta_j \left(\sqrt{1-\frac{c^2}{\lambda_j^2}}\left|0\right\rangle + \frac{c}{\lambda_j}\left|1\right\rangle \right) \otimes \sqrt{\frac{2}{T}} \sum_{k=0}^{T-1}\sum_{\tau=0}^{T-1} \sin\left(\frac{\pi(\tau+1/2)}{T}\right) \left|k\right\rangle \delta_{k\tau} \otimes e^{iA\tau t_0/T} \left|u_j\right\rangle =$$

$$\sum_{j=1}^{N} \beta_j \left(\sqrt{1-\frac{c^2}{\lambda_j^2}}\left|0\right\rangle + \frac{c}{\lambda_j}\left|1\right\rangle \right) \otimes \sqrt{\frac{2}{T}} \sum_{k=0}^{T-1}\sum_{\tau=0}^{T-1} \sin\left(\frac{\pi(\tau+1/2)}{T}\right) \left|k\right\rangle \left\langle k|\tau\right\rangle \otimes e^{iA\tau t_0/T} \left|u_j\right\rangle =$$

$$\sum_{j=1}^{N} \beta_j \left(\sqrt{1-\frac{c^2}{\lambda_j^2}}\left|0\right\rangle + \frac{c}{\lambda_j}\left|1\right\rangle \right) \otimes \sqrt{\frac{2}{T}} \sum_{k=0}^{T-1}\sum_{\tau=0}^{T-1} \left|k\right\rangle \left\langle k\right| \sin\left(\frac{\pi(\tau+1/2)}{T}\right) \left|\tau\right\rangle \otimes e^{iA\tau t_0/T} \left|u_j\right\rangle =$$

$$\sum_{j=1}^{N} \beta_j \left(\sqrt{1-\frac{c^2}{\lambda_j^2}}\left|0\right\rangle + \frac{c}{\lambda_j}\left|1\right\rangle \right) \otimes \sqrt{\frac{2}{T}} \sum_{\tau=0}^{T-1}\sum_{k=0}^{T-1} \left[\left|k\right\rangle \left\langle k\right| \otimes e^{iA\tau t_0/T} \right]$$

$$\left[\sin\left(\frac{\pi(\tau+1/2)}{T}\right) \left|\tau\right\rangle \otimes \left|u_j\right\rangle \right] =$$

$$\sum_{j=1}^{N} \left(\sqrt{1-\frac{c^2}{\lambda_j^2}}\left|0\right\rangle + \frac{c}{\lambda_j}\left|1\right\rangle \right) \otimes \sqrt{\frac{2}{T}} \sum_{\tau=0}^{T-1} \left[\sum_{k=0}^{T-1} \left|k\right\rangle \left\langle k\right| \otimes e^{iA\tau t_0/T} \right]$$

$$\sin\left(\frac{\pi(\tau+1/2)}{T}\right) \left|\tau\right\rangle \otimes \beta_j \left|u_j\right\rangle .$$

Reversing U_t' on the third register, controlled on the second, makes the term in the brackets disappear:

$$\sum_{j=1}^{N} \left(\sqrt{1-\frac{c^2}{\lambda_j^2}}\left|0\right\rangle + \frac{c}{\lambda_j}\left|1\right\rangle \right) \otimes \sqrt{\frac{2}{T}} \sum_{\tau=0}^{T-1} \left[\sum_{k=0}^{T-1} \left|k\right\rangle \left\langle k\right| \otimes e^{iA\tau t_0/T} \right] \sin\left(\frac{\pi(\tau+1/2)}{T}\right) \left|\tau\right\rangle \tag{150}$$

$$\otimes \beta_j \left|u_j\right\rangle \xrightarrow{I_1 \otimes U_{t,n}'^{\dagger}}$$

$$\sum_{j=1}^{N} \left(\sqrt{1-\frac{c^2}{\lambda_j^2}}\left|0\right\rangle + \frac{c}{\lambda_j}\left|1\right\rangle \right) \otimes \sqrt{\frac{2}{T}} \sum_{\tau=0}^{T-1} \sin\left(\frac{\pi(\tau+1/2)}{T}\right) \left|\tau\right\rangle \otimes \beta_j \left|u_j\right\rangle .$$

Now substitute back $\left|\Psi_0\right\rangle = \sqrt{\frac{2}{T}} \sum_{\tau=0}^{T-1} \sin\left(\frac{\pi(\tau+1/2)}{T}\right) \left|\tau\right\rangle$:

$$\sum_{j=1}^{N} \left(\sqrt{1-\frac{c^2}{\lambda_j^2}}\left|0\right\rangle + \frac{c}{\lambda_j}\left|1\right\rangle \right) \otimes \sqrt{\frac{2}{T}} \sum_{\tau=0}^{T-1} \sin\left(\frac{\pi(\tau+1/2)}{T}\right) \left|\tau\right\rangle \otimes \beta_j \left|u_j\right\rangle = \tag{151}$$

$$\sum_{j=1}^{N} \left(\sqrt{1-\frac{c^2}{\lambda_j^2}}\left|0\right\rangle + \frac{c}{\lambda_j}\left|1\right\rangle \right) \otimes \left|\Psi_0\right\rangle \otimes \beta_j \left|u_j\right\rangle .$$

Finally, undo the H'_t on the second register to see that this register is set back to zero. It results in:

$$\sum_{j=1}^{N}\left(\sqrt{1-\frac{c^2}{\lambda_j^2}}\,|0\rangle+\frac{c}{\lambda_j}\,|1\rangle\right)\otimes|\Psi_0\rangle\otimes\beta_j\,|u_j\rangle\xrightarrow{I_1\otimes H_t'^{\dagger}\otimes I_n} \tag{152}$$
$$\sum_{j=1}^{N}\left(\sqrt{1-\frac{c^2}{\lambda_j^2}}\,|0\rangle+\frac{c}{\lambda_j}\,|1\rangle\right)\otimes|0\rangle\otimes\beta_j\,|u_j\rangle\,.$$

Note that from Equation (150) on, all register were uncoupled. So, the remaining steps of uncompute were unnecessary. These were only done to make the notation easier.

G Proof of Lemma 3.4

Proof. Rewrite Equation (38) as follows:

$$\alpha_{k|j} = \frac{\sqrt{2}}{T}\sum_{\tau=0}^{T-1} e^{i\lambda_j \tau t_0/T} e^{-2\pi i k\tau/T} \sin\left(\frac{\pi(\tau+1/2)}{T}\right) \tag{153}$$

$$\text{note that } \sin(x) = \frac{1}{2i}(e^{ix} - e^{-ix})$$

$$= \frac{1}{i\sqrt{2}T}\sum_{\tau=0}^{T-1} e^{\frac{i\tau}{T}(\lambda_j t_0 - 2\pi k)}\left(e^{i\frac{\pi(\tau+1/2)}{T}} - e^{-i\frac{\pi(\tau+1/2)}{T}}\right)$$

$$\text{define } \delta = \lambda_j t_0 - 2\pi k$$

$$= \frac{1}{i\sqrt{2}T}\sum_{\tau=0}^{T-1}\left(e^{\frac{i\pi}{2T}} e^{i\tau\frac{\delta+\pi}{T}} - e^{\frac{-i\pi}{2T}} e^{i\tau\frac{\delta-\pi}{T}}\right)$$

$$\text{two geometric sequences}$$

$$= \frac{1}{i\sqrt{2}T}\left(e^{\frac{i\pi}{2T}}\frac{1-e^{i(\delta+\pi)}}{1-e^{i\frac{\delta+\pi}{T}}} - e^{\frac{-i\pi}{2T}}\frac{1-e^{i(\delta-\pi)}}{1-e^{i\frac{\delta-\pi}{T}}}\right)$$

$$= \frac{1}{i\sqrt{2}T}\left(e^{\frac{i\pi}{2T}}\frac{1+e^{i\delta}}{1-e^{i\frac{\delta+\pi}{T}}} - e^{\frac{-i\pi}{2T}}\frac{1+e^{i\delta}}{1-e^{i\frac{\delta-\pi}{T}}}\right)$$

$$= \frac{1+e^{i\delta}}{i\sqrt{2}T}\left(e^{\frac{i\pi}{2T}}\frac{1}{1-e^{i\frac{\delta+\pi}{T}}} - e^{\frac{-i\pi}{2T}}\frac{1}{1-e^{i\frac{\delta-\pi}{T}}}\right)$$

$$= \frac{1+e^{i\delta}}{i\sqrt{2}T}\left(\frac{1}{e^{\frac{-i\pi}{2T}}(1-e^{i\frac{\delta+\pi}{T}})} - \frac{1}{e^{\frac{i\pi}{2T}}(1-e^{i\frac{\delta-\pi}{T}})}\right)$$

$$= \frac{1+e^{i\delta}}{i\sqrt{2}T}\left(\frac{e^{\frac{-i\delta}{2T}}}{e^{\frac{-i\pi}{2T}}}\frac{1}{e^{\frac{-i\pi}{2T}}(1-e^{i\frac{\delta+\pi}{T}})} - \frac{e^{\frac{-i\delta}{2T}}}{e^{\frac{-i\delta}{2T}}}\frac{1}{e^{\frac{i\pi}{2T}}(1-e^{i\frac{\delta-\pi}{T}})}\right)$$

$$= \frac{1+e^{i\delta}}{i\sqrt{2}T}\left(\frac{e^{\frac{-i\delta}{2T}}}{e^{-\frac{i}{2T}(\delta+\pi)} - e^{\frac{i}{2T}(\delta+\pi)}} - \frac{e^{\frac{-i\delta}{2T}}}{e^{-\frac{i}{2T}(\delta-\pi)} - e^{\frac{i}{2T}(\delta-\pi)}}\right)$$

$$= \frac{(1+e^{i\delta})e^{\frac{-i\delta}{2T}}}{i\sqrt{2}T}\left(\frac{1}{e^{-\frac{i}{2T}(\delta+\pi)} - e^{\frac{i}{2T}(\delta+\pi)}} - \frac{1}{e^{-\frac{i}{2T}(\delta-\pi)} - e^{\frac{i}{2T}(\delta-\pi)}}\right)$$

$$= \frac{(1+e^{i\delta})e^{\frac{-i\delta}{2T}}}{i\sqrt{2}T}\left(\frac{1}{-2i\sin(\frac{\delta+\pi}{2T})} - \frac{1}{-2i\sin(\frac{\delta-\pi}{2T})}\right)$$

$$= \frac{(1+e^{i\delta})e^{\frac{-i\delta}{2T}}}{-2i\cdot i\sqrt{2}T}\left(\frac{1}{\sin(\frac{\delta+\pi}{2T})} - \frac{1}{\sin(\frac{\delta-\pi}{2T})}\right)$$

$$= \frac{e^{-\frac{i\delta}{2T}} + e^{\frac{i\delta}{2T}(2T-1)}}{-2i\cdot i\sqrt{2}T}\left(\frac{1}{\sin(\frac{\delta+\pi}{2T})} - \frac{1}{\sin(\frac{\delta-\pi}{2T})}\right)$$

$$= \frac{e^{\frac{i\delta}{2}(-\frac{1}{T})} + e^{\frac{i\delta}{2}(2-\frac{1}{T})}}{-2i\cdot i\sqrt{2}T}\left(\frac{1}{\sin(\frac{\delta+\pi}{2T})} - \frac{1}{\sin(\frac{\delta-\pi}{2T})}\right)$$

$$= \frac{e^{\frac{i\delta}{2}(1-\frac{1}{T}-1)} + e^{\frac{i\delta}{2}(1-\frac{1}{T}+1)}}{-2i\cdot i\sqrt{2}T}\left(\frac{1}{\sin(\frac{\delta+\pi}{2T})} - \frac{1}{\sin(\frac{\delta-\pi}{2T})}\right)$$

$$= \frac{e^{\frac{i\delta}{2}(1-\frac{1}{T})}(e^{-\frac{i\delta}{2}} + e^{\frac{i\delta}{2}})}{-2i\cdot i\sqrt{2}T}\left(\frac{1}{\sin(\frac{\delta+\pi}{2T})} - \frac{1}{\sin(\frac{\delta-\pi}{2T})}\right)$$

$$
\begin{aligned}
&= \frac{e^{\frac{i\delta}{2}(1-\frac{1}{T})}2\cos(\frac{\delta}{2})}{-2i\cdot i\sqrt{2}T}\left(\frac{1}{\sin(\frac{\delta+\pi}{2T})}-\frac{1}{\sin(\frac{\delta-\pi}{2T})}\right)\\
&= \frac{e^{\frac{i\delta}{2}(1-\frac{1}{T})}\cos(\frac{\delta}{2})}{\sqrt{2}T}\left(\frac{1}{\sin(\frac{\delta+\pi}{2T})}-\frac{1}{\sin(\frac{\delta-\pi}{2T})}\right)\\
&= \frac{e^{\frac{i\delta}{2}(1-\frac{1}{T})}\cos(\frac{\delta}{2})}{\sqrt{2}T}\frac{\sin(\frac{\delta-\pi}{2T})-\sin(\frac{\delta+\pi}{2T})}{\sin(\frac{\delta+\pi}{2T})\sin(\frac{\delta-\pi}{2T})}\\
&= \frac{e^{\frac{i\delta}{2}(1-\frac{1}{T})}\cos(\frac{\delta}{2})}{\sqrt{2}T}\frac{2\cos(\frac{\delta}{2T})\sin(\frac{-\pi}{2T})}{\sin(\frac{\delta+\pi}{2T})\sin(\frac{\delta-\pi}{2T})}\\
&= -\frac{e^{\frac{i\delta}{2}(1-\frac{1}{T})}\cos(\frac{\delta}{2})}{\sqrt{2}T}\frac{2\cos(\frac{\delta}{2T})\sin(\frac{\pi}{2T})}{\sin(\frac{\delta+\pi}{2T})\sin(\frac{\delta-\pi}{2T})}\\
&= -\frac{e^{\frac{i\delta}{2}(1-\frac{1}{T})}\cos(\frac{\delta}{2})}{T}\frac{\sqrt{2}\cos(\frac{\delta}{2T})\sin(\frac{\pi}{2T})}{\sin(\frac{\delta+\pi}{2T})\sin(\frac{\delta-\pi}{2T})}.
\end{aligned}
$$

Remark. Note that this differs a factor -2 from [15] and [10]. However, these papers start with the same equation as this report, so one who believes the steps used in this report, doubts the correctness of those papers.

Now use $|e^{\frac{i\delta}{2}(1-\frac{1}{T})}| = 1$ since δ and T are real, $\sin(x) \leq x$ for $x \geq 0$, and $\cos(x) \leq 1$ in the nominator. Use that $\sin(x) \geq x - x^3/6$ for $x \geq 0$ in the denominator. This results in:

$$
\begin{aligned}
&\left|-\frac{e^{\frac{i\delta}{2}(1-\frac{1}{T})}\cos(\frac{\delta}{2})}{T}\frac{\sqrt{2}\cos(\frac{\delta}{2T})\sin(\frac{\pi}{2T})}{\sin(\frac{\delta+\pi}{2T})\sin(\frac{\delta-\pi}{2T})}\right| \qquad (154)\\
&\leq \left|\frac{\sqrt{2}\frac{\pi}{2T}}{T\Big((\frac{\delta+\pi}{2T})-(\frac{\delta+\pi}{2T})^3/6\Big)\Big((\frac{\delta-\pi}{2T})-(\frac{\delta-\pi}{2T})^3/6\Big)}\right|\\
&= \left|\frac{\sqrt{2}\pi}{2T^2(\delta+\pi)(\delta-\pi)\Big(\frac{1}{4T^2}-\frac{(\delta+\pi)^2}{96T^4}-\frac{(\delta-\pi)^2}{96T^4}+\frac{(\delta+\pi)^2(\delta-\pi)^2)}{2304T^6}\Big)}\right|\\
&= \left|\frac{2\sqrt{2}\pi}{(\delta^2-\pi^2)\Big(1-\frac{\delta^2+\pi^2}{24T^2}+\frac{(\delta^2-\pi^2)^2)}{576T^4}\Big)}\right|\\
&\leq \left|\frac{2\sqrt{2}\pi}{(\delta^2-\pi^2)\Big(1-\frac{\delta^2+\pi^2}{24T^2}\Big)}\right|\\
&\text{use } T \geq 10\delta\\
&\leq \left|\frac{2\sqrt{2}\pi}{(\delta^2-\pi^2)\Big(1-\frac{\delta^2+\pi^2}{24(10\delta)^2}\Big)}\right|\\
&\text{use } 2\pi \leq \delta\\
&\leq \left|\frac{2\sqrt{2}\pi}{(\delta^2-(\delta/2)^2)\Big(1-\frac{\delta^2+(\delta/2)^2}{24(10\delta)^2}\Big)}\right|\\
&= \left|\frac{2\sqrt{2}\pi}{\frac{3}{4}\delta^2\frac{1919}{1920}}\right| = \left|\frac{\frac{5120}{1919}\sqrt{2}\pi}{\delta^2}\right| \leq \left|\frac{12}{\delta^2}\right|.
\end{aligned}
$$

Remark. Note that a different and tighter upper bound is derived than in [15] and [10].

From this it can be concluded that $|\alpha_{k|j}|^2 \leq \frac{144}{\delta^4}$, when $|k - \frac{\lambda_j t_0}{2\pi}| \geq 1$. A visualization of the tightness of this upper-bound is given in Figure 17.

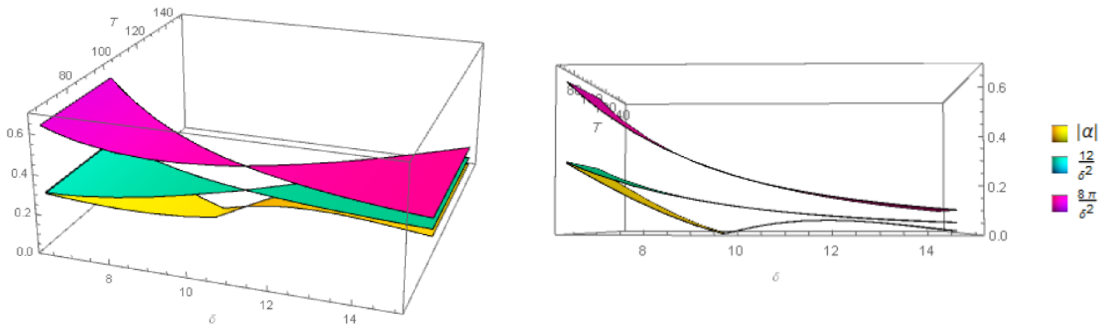


Figure 17: Tightness of upper bounds alpha (two plots with same bounds, different point of view)

□

H Failed attempts

Since there might not be a unique solution to the chosen plaintext attack, it makes no sense to take one plaintext-ciphertext pair and try to create as many (independent) equations as variables. So, the next step is figuring out how far the number of solutions can be limited, if only one plaintext-ciphertext pair is considered and what happens if multiple plaintext-ciphertext pairs are considered. The next subsection will do both.

H.1 Creating a system of equations with a limited number of solutions

Intuitively, adding more plaintext-ciphertext pairs should give more information about the key. Section H.1.1 looks into the question whether or not adding more plaintext-ciphertext indeed limits the number of solutions of the system of equations.

A general approach from linear algebra to create a full rank matrix out of an under-determined polynomial system is to create a Macaulay matrix. This general approach of shifting the polynomials with all possible monomials will create an exponentially large matrix. In contrast to a general system of equations, the systems of equations in this report are well defined, which means that there might be a smarter choice in polynomials and monomials for the shifts. Section H.1.2 will look into these, so called, smart shifts.

H.1.1 Adding more plaintext-ciphertext pairs

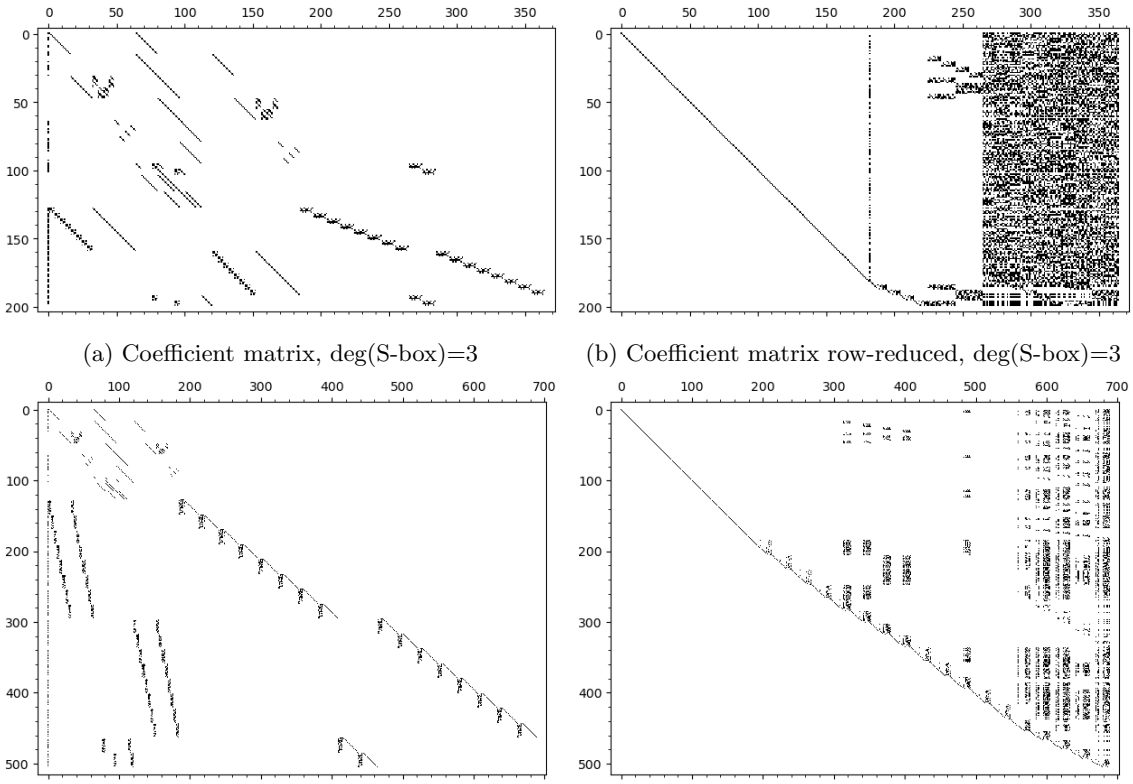
One option to create more equations is to simply add more plaintext-ciphertext pairs. The idea is that only the key bits are of interest, and intuitively, adding more equations with the same key bits as answer would make the system less under-determined. However, new variables need to be introduced for all the encryption variables. Note that since the key did not change, it is not necessary to introduce new variables for the key expansion, unfortunately this does also not lead to new equations.

According to Table 1 this would mean that $64 + 80 = 144$ new variables and $48 + 32 = 80$ new equations would need to be created for the representation of the S-box with degree three. So, adding a new plaintext-ciphertext pair leads to 365 variables (if one is also counted as a variable) and only 200 equations. Similarly, for the representation of the S-box with degree two, according to Table 1 this would create $64 + 224 = 288$ new variables and $48 + 168 = 216$ new equations. So, adding a new plaintext-ciphertext pair leads to 689 new variables (if one is also counted as a variable) and only 506 new equations.

The above is visualized in Figure 18. It can be concluded that for both representations of the S-box, the ratio between variables and equations gets worse by adding new plaintext-ciphertext pairs and the problem did not get any easier. This can be seen by comparing the row reduction of the matrix with two sets of plaintext-ciphertext, which is also given in Figure 18, to the row reduction of the coefficient matrix with only one set of plaintext-ciphertext, which is given in Figure 15, in Section 5.4.

H.1.2 Smart shifts

Section 8 explains that multiplying the equations with monomials results in new variables, but, introduces more new equations than variables. However, this leads to a very large matrix and the larger the matrix, the more complex the computations. Therefore, it is worth trying to find smart



(a) Coefficient matrix, deg(S-box)=3

(b) Coefficient matrix row-reduced, deg(S-box)=3

(c) Coefficient matrix, deg(S-box)=2

(d) Coefficient matrix row-reduced, deg(S-box)=2

Figure 18: Mini-AES ($p_1 = [9, 12, 6, 3], p_2 = [8, 4, 15, 1], k = [12, 3, 15, 0]$)

shifts. The term smart shifts refers to shifts that introduce more new equations than variables, without having to multiply all equations with all monomials.

An example of a smart shift can be found using the Equation (67). For $j = N_e N_b - 1$ this reads $0 = x_0(3,m) \oplus p(3,m) \oplus w_0(3,m)$. Remember that $p(j,m)$ are known, therefore, multiplying these with variables does not lead to non-linear terms, i.e. new variables. Besides, the S-box equations already introduced the non linear terms $x_0(j,m) \cdot x_0(j,m')$ and $w_0(j,m) \cdot w_0(j,m')$ for $j = N_e N_b - 1$ and $m, m' = 0, ..., S_e - 1$. Multiplying $0 = x_0(j,m) \oplus p(j,m) \oplus w_0(j,m)$ with $x_0(j,m')$ and $w_0(j,m')$ creates 32 new equations and introduces 16 new variables, namely $x_0(j,m) \cdot w_0(j,m')$ for $j = N_e N_b - 1$ and $m, m' = 0, ..., S_e - 1$.

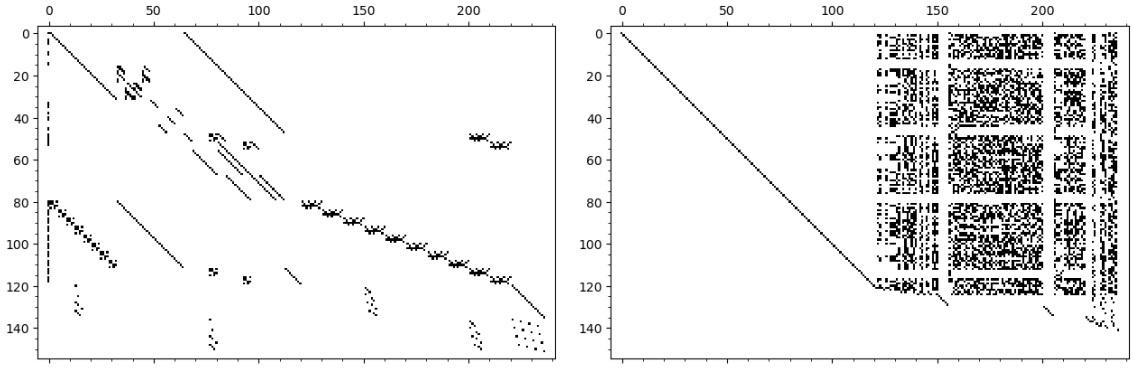
This smart shift decreases the gap between the number of variables and the number of equations, note that the size of the gap is not influenced by the representation of the S-box since the S-box equations play no role in this smart shift. The gap is decreased from $221 - 120 = 101$ to $237 - 152 = 85$. However, after adding the new equations, not all equations are linearly independent anymore. This results in 10 zero rows after row reduction, which means that the gap is actually only decreased by 6. A visual representation of the shifted matrix before and after row reduction, for both representations of the S-box, can be found in Figure 19. The matrix after row reduction shows some zero rows. Furthermore, it is worth noting that the right side of the matrix already contains fewer non-zero entries compared to the row-reduced coefficient matrix without the shift (Figure 15). This means that even though the zero rows can be removed and the problem did get easier by doing this smart shift, the winning is neglectable.

The reason that row reduction results in 10 zero rows, might not be very clear to see. Therefore, an experiment is conducted with multiple different plaintexts and keys, of which some results are shown in Table 15. Even though this table does not provide insight in the reason that 10 zero rows appear, it does show that the number 10 is independent of the plaintext or the key.

This smart shift resulted in a decrease of the gap of only 6. So, more work has to be done and it is worth investigating whether or not more smart shifts are possible. The previously described smart shift worked, because the equation had only two variables and these variables were already an input to the S-box equations. This means that every time that the equation was multiplied by an x, this created products of the x terms which were introduced before, and a mix term of x and w. Since the same equation was also multiplied by w, this created products of the w terms which were already introduced before and the same mix term as with the first multiplication.

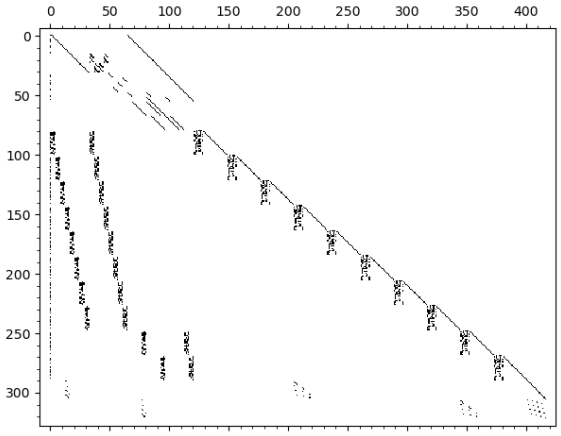
		Number of rows	Number of columns	Gap	Gap change
Plaintext Key	$[9,12,6,3]$ $[12,3,15,0]$	142	237	95	6
Plaintext Key	$[8,4,15,1]$ $[1,11,7,2]$	142	237	95	6
Plaintext Key	$[0,0,0,0]$ $[12,3,15,0]$	142	237	95	6
Plaintext Key	$[15,15,15,15]$ $[12,3,15,0]$	142	237	95	6
Plaintext Key	$[9,12,6,3]$ $[0,0,0,0]$	142	237	95	6
Plaintext Key	$[9,12,6,3]$ $[15,15,15,15]$	142	237	95	6

Table 15: Mini-AES smart shift change in gap, based on deg(S-box)=3

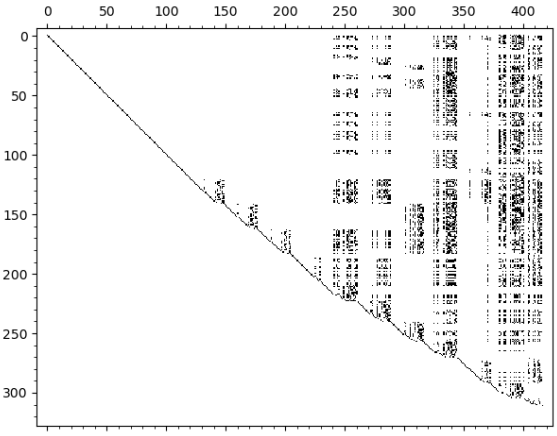


(a) Coefficient matrix, deg(S-box)=3

(b) Coefficient matrix row-reduced, deg(S-box)=3



(c) Coefficient matrix, deg(S-box)=2



(d) Coefficient matrix row-reduced, deg(S-box)=2

Figure 19: Mini-AES smart shift ($p = [9, 12, 6, 3], k = [12, 3, 15, 0]$)

Unfortunately, this is the only equation with this property and, therefore, this is the only possible smart shift. All other equations also contain variables that are not input of an S-box equation (note that w_0 is, in contradiction to w_2, the input of an S-box, which means that whereas the smart shift works for the equation that contains the plaintext, it does not work for the equation that contains the ciphertext even though both the plaintext and the ciphertext are known). This means that every multiplication, not only creates mix terms (for instance an x term and a w term), but also products of terms (for instance an x term times another x term) that lead to new variables. Therefore, the smart shift that worked for the first equation will not work for any other equation. A visualization of this idea can be found in Figure 20.

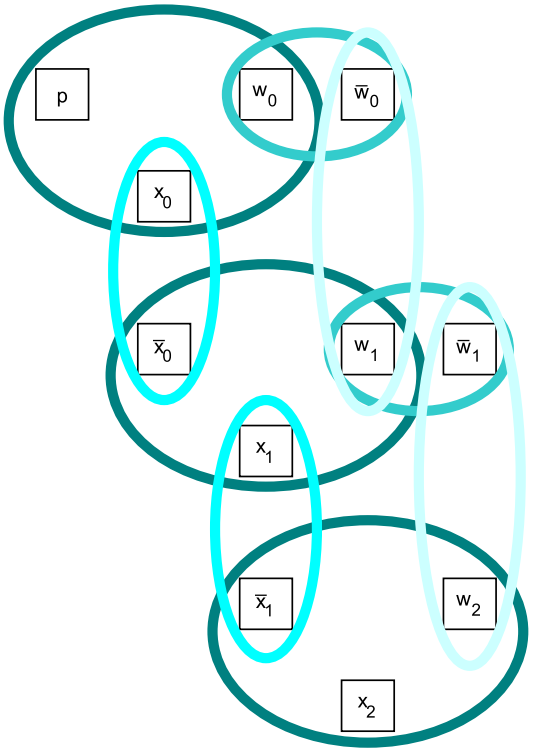


Figure 20: Mini AES equation visualization

Another option is combining equations to find smart shifts. The idea is that as much new equations as possible are created; but as little new variables as possible. Therefore, if a new variable is created, another multiplication is sought that introduces the same variable. This means that, starting with the smart shift mentioned above, equations are sought that also contain $w_0(j,m)$ and/or $x_0(j,m)$. Both variables occur in exactly one other equation, Equation (85) and Equation (80) respectively. This means that these equations need to be multiplied by $w_0(j,m)$ and $x_0(j,m)$ as well and that Equation (67) needs to be multiplied by the variables of Equation (85) and Equation (80). However, Equation (85) and Equation (80) also contained other terms besides $w_0(j,m)$ and $x_0(j,m)$ and for these terms, new variables are created. The next step would than be to find a new equation, containing the same variables, multiplying this equation with the same variables that are already used and multiplying the equations that are already used with the new terms in the new equation. With the way that Mini-AES is structured, one could easily verify that this process has to be repeated until every equation is multiplied with every variable. This means that this is no longer

a smart shift, but a Macaulay matrix approach.

I Example Macaulay matrices

Each (sub)figure contains a matrix on the left hand side and the corresponding row-reduced matrix on the right hand side.

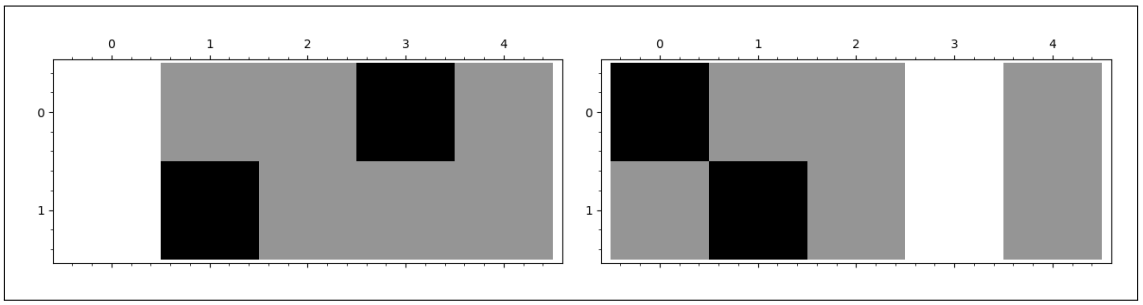


Figure 21: System 1 - coefficient matrix.

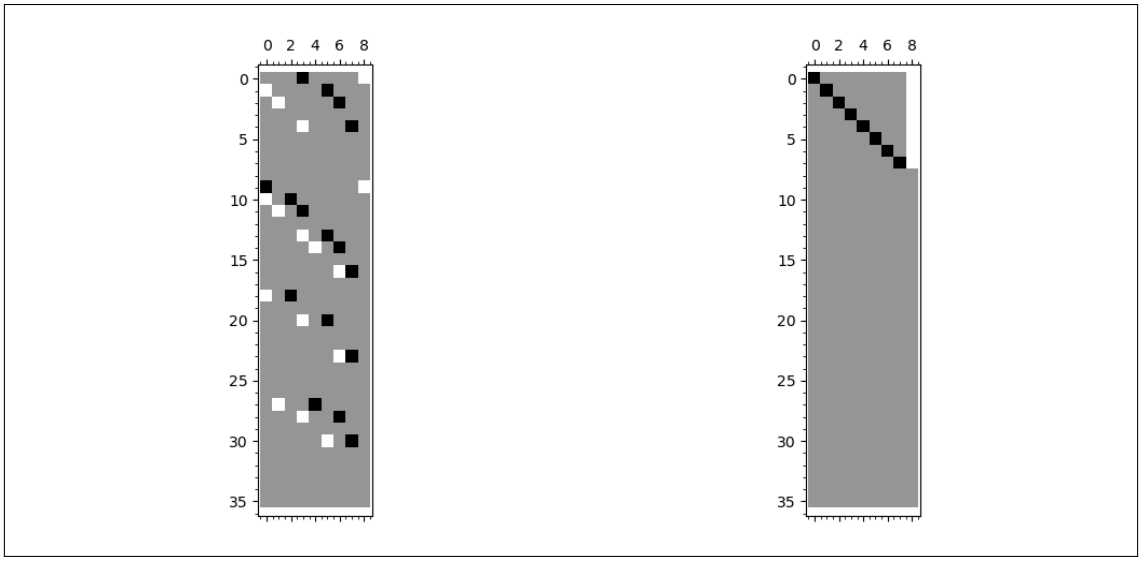


Figure 22: System 1 - max degree and field equations - $D = 2$ is solvable.

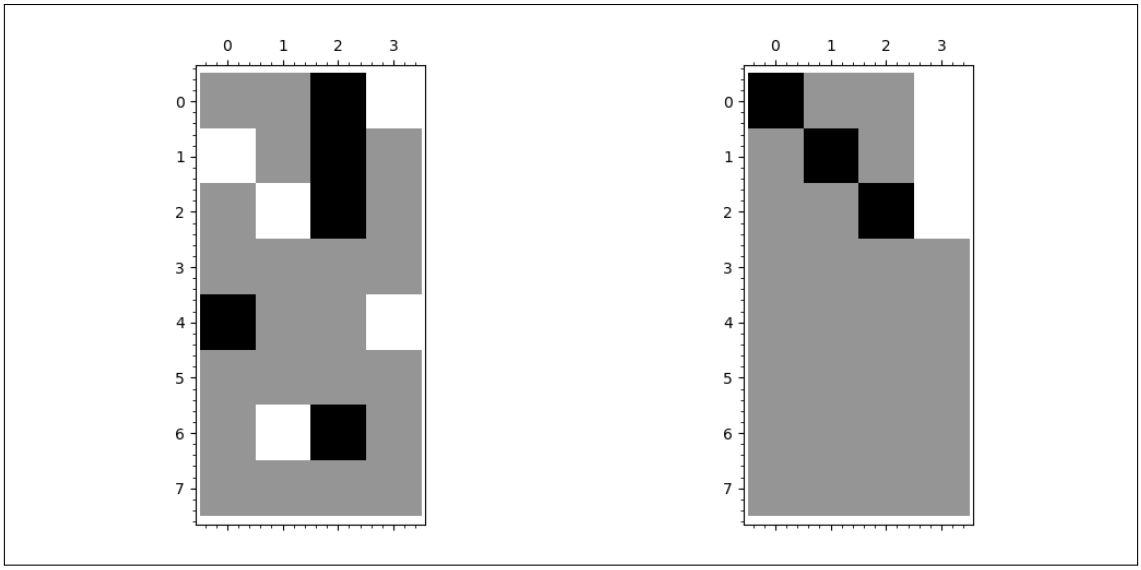


Figure 23: System 1 - max degree and remove squares - $D = 1$ is solvable.

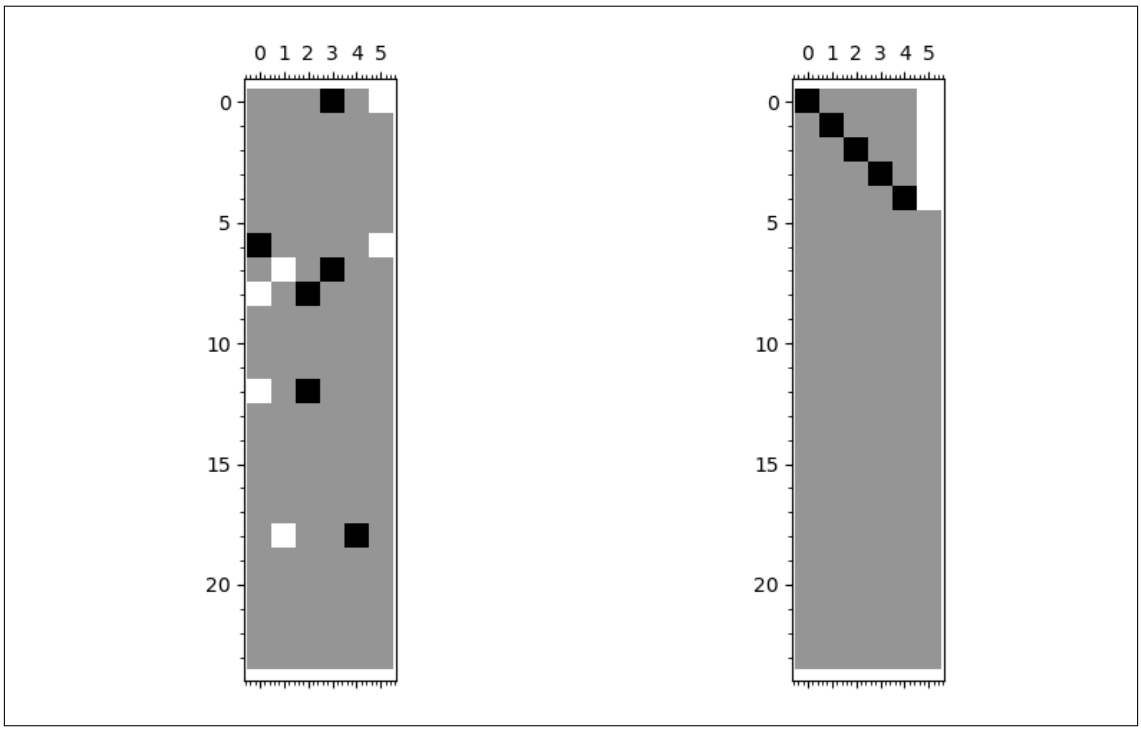
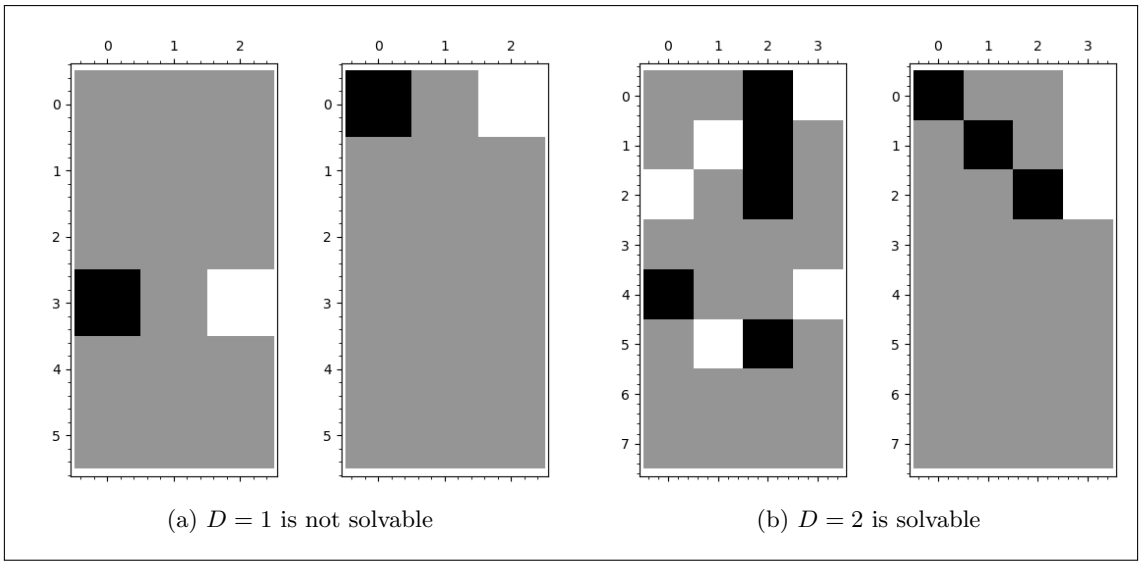


Figure 24: System 1 - total degree and field equations - $D = 2$ is solvable.



(a) $D = 1$ is not solvable

(b) $D = 2$ is solvable

Figure 25: System 1 - total degree and remove squares.

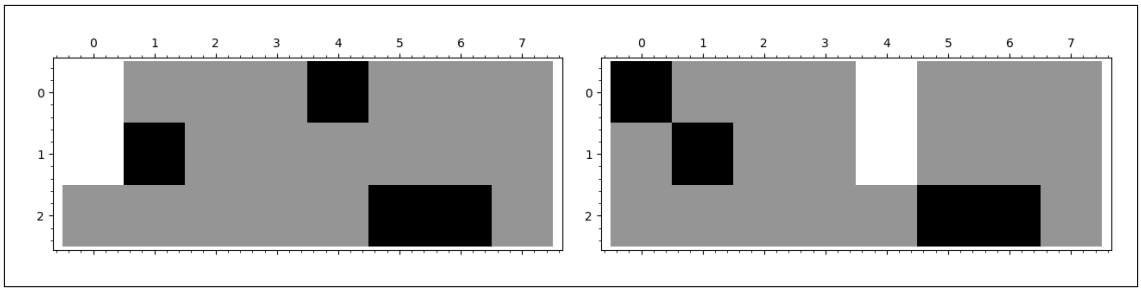


Figure 26: System 2 - coefficient matrix.

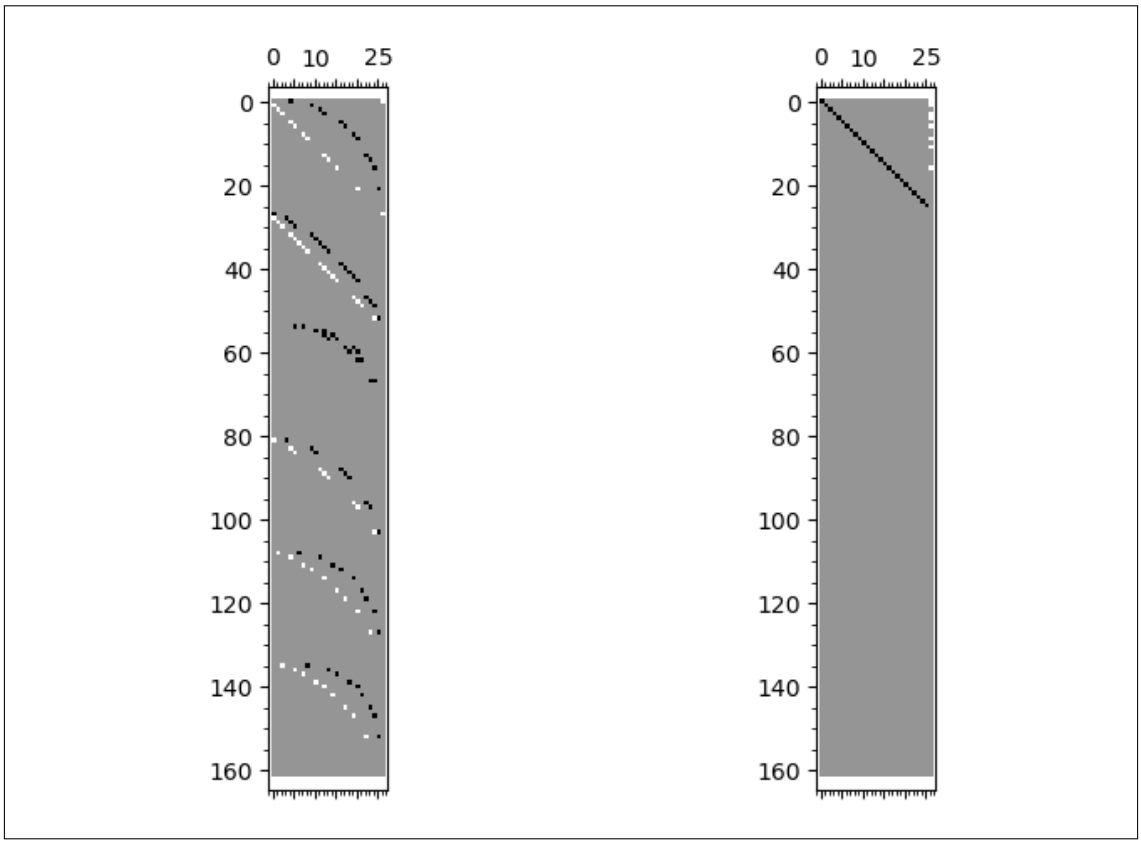


Figure 27: System 2 - max degree and field equations - $D = 2$ is solvable.

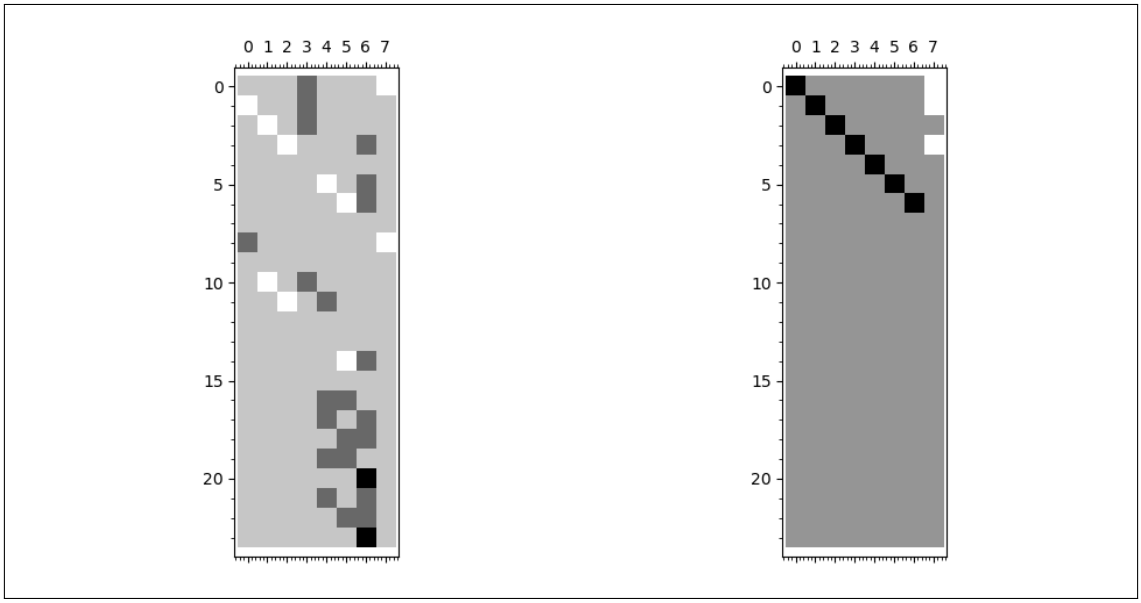
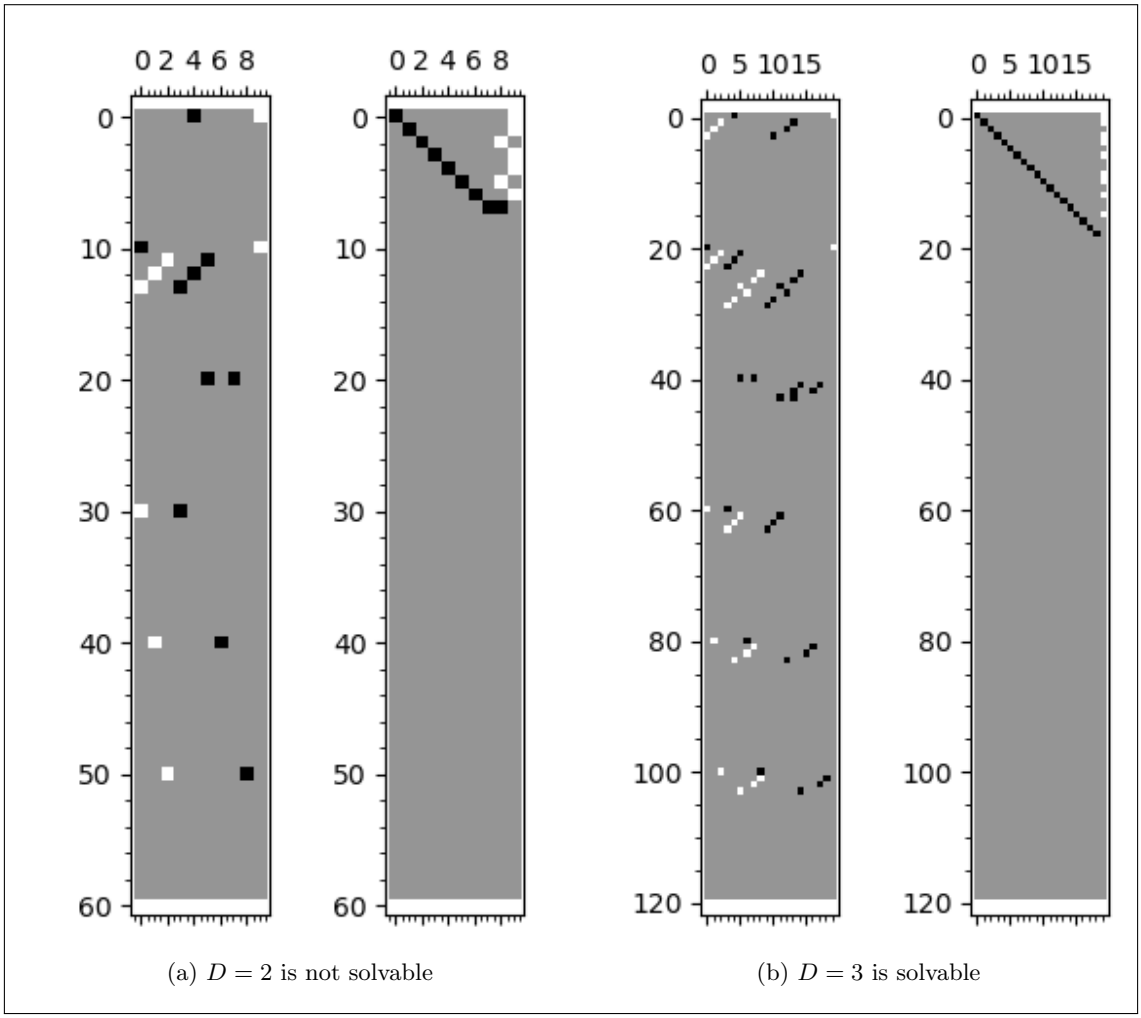


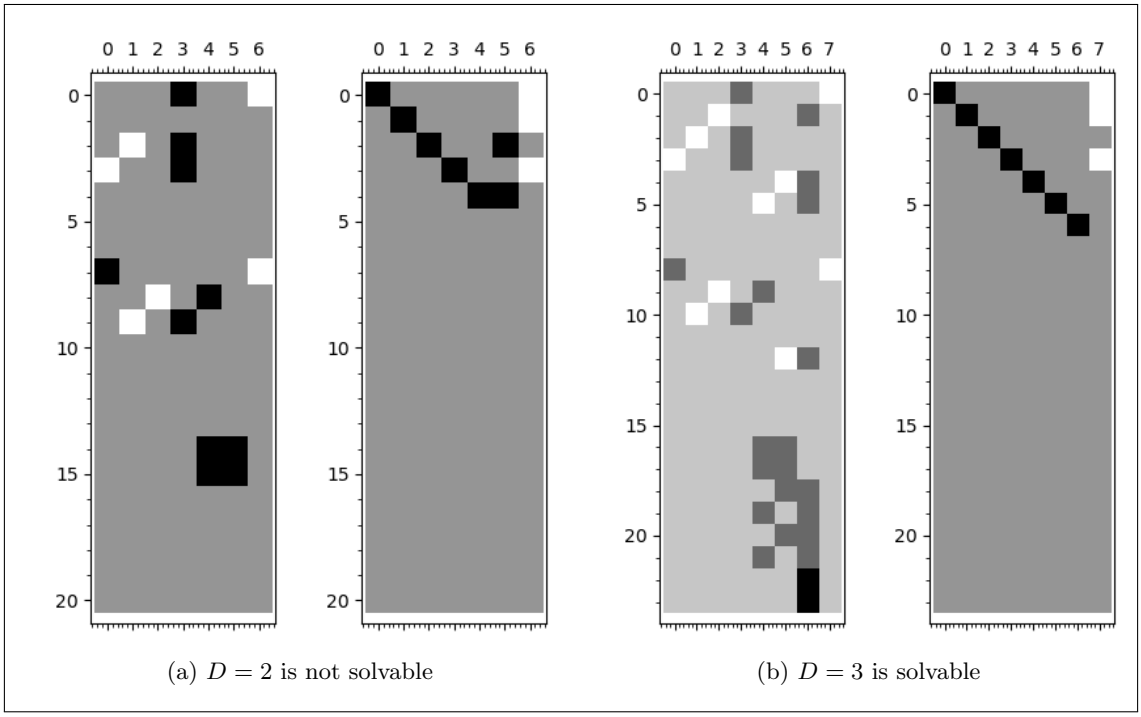
Figure 28: System 2 - max degree and remove squares - $D = 1$ is solvable.



(a) $D = 2$ is not solvable

(b) $D = 3$ is solvable

Figure 29: System 2 - total degree and field equations.



(a) $D = 2$ is not solvable

(b) $D = 3$ is solvable

Figure 30: System 2 - total degree and remove squares.