Gymnasium der Freien Katholischen Schulen Zurich
University of Zurich
Institute of Mathematics

Matura Project

Some Applications of Algebra in Coding Theory and Cryptography

Author:
Constantin V. A. Vlachos

Supervisor:
Niklas Gassner

December 2023

Constantin Vlachos
constantinosvasiliosargyris.vlachos@uzh.ch

Acknowledgements

I would like to express my profound gratitude to Professor Dr. Rosenthal, who not only introduced me to coding theory and gave me a starting point for this project but also inspired me with his excellent lectures on linear algebra.
My sincere gratitude goes to my supervisor Niklas Gassner for his willingness to take on the task of supervisor when no one in my school could, his guidance in coding theory, as well as in the intricacies of LaTeX and the immense amount of freedom he has given me throughout this whole project.
I thank Dr. Simran Tinani for inviting me to her seminar, all her help during the preparation of my talk and without whom I would not have been introduced to cryptography and been able to try and create my very own hash function.
Next, I thank Dr. Julia Schneider for all her help concerning p-adic numbers and finite fields.
Thank you to Mr. Daniel Bürgy for introducing me to the university and supporting me in my choice of topic.
Finally, I thank my dear family and friends for their continuous support.
Without the help and time of all of these people this project would never have been possible and I am forever grateful to them.

Constantin Vlachos
constantinosvasiliosargyris.vlachos@uzh.ch

Abstract

In this project we reverse-engineered the linear code known as the ISBN (international standrad book number) code. We searched for a generator matrix and parity-check matrix experimentally and used them to deduce certain properties of the ISBN code.
Next we reverse-engineered the non-linear IBAN (international bank account number) code. We came up with a decoding scheme utilizing p-adic numbers and used this very scheme to prove our theorem that the IBAN code is a single error detecting code.
Finally we also constructed a hash-function. In order to do so it was necessary to also construct a pseudo-random generator which was then combined with the pseudo-random function by Goldreich, Goldwasser and Micali, to get the desired result. We did not prove this or find experimental evidence to support a claim of first or second pre-image resistance, we did however test some known attacks.

The project is structured as follows. We begin with first examples of the two fields (coding theory and cryptography) in chapter one and some essential linear algebra results in chapter two. Next, we will introduce and study linear coding theory and put focus on BCH codes in chapter three. Chapter four is devoted to mathematical cryptography. In chapter five the reader will find the reverse-engineering of the ISBN code as well as additional information on its specifications. The sixth chapter is dedicated to explaining the IBAN code, why a non-linear code was a reasonable choice and how p-adic numbers can be utilized to let anyone decode an IBAN in a single glance. The final chapter is left to the hash function made by the author of this paper using the PRG's, PRF's and hash function knowledge given in chapter four.

Constantin Vlachos
constantinosvasiliosargyris.vlachos@uzh.ch

Contents

Constantin Vlachos
constantinosvasiliosargyris.vlachos@uzh.ch

Constantin Vlachos
constantinosvasiliosargyris.vlachos@uzh.ch

1. Introduction.

1.1. Cryptography and Coding Theory.

A common misconception is that coding theory is cryptography. Coding theory solves the problem of error correction and error detection over noisy channels. We explain this in greater detail below.
In communication information comes from a *source* and goes to a receiver through a medium called a channel. A channel could be a radio, telephone or in live conversation simply space. Channels which output every input exactly as it came in are called *noiseless*. These are quite rare. *Noise* distorts messages being sent over a channel and produces errors. To clarify what *noise* is we give some examples: *noise* in the radio is known as static, over the telephone it is sometimes caused by lightning and in face to face speech it can be understood as literal noise (loud drilling) that makes it harder for the recipient to understand the message.
Coding theory is used to detect and sometimes also correct errors resulting from noise.
Cryptography on the other hand is a tool to disguise a message (often by complex mathematical procedures) such that it is unintelligible to unwanted readers. Here we also send a message from a source over a channel to a recipient, however the noise aspect is neglected and we focus on the channel being secure or insecure and how one can protect messages over such channels.

1.2. A First Example in Coding Theory.

One has to navigate their way through the maze, but a second party is at a vantage point above the maze and can see the way out. The only issue is that the communication is over a noisy channel. Some rules are set up by both parties:
00 means go backwards.
11 means go forward.
10 means go right.
01 means go left.
Due to the noise,if the message 11 (a forward prompt) is sent, the message 01 (a left prompt) might be received. Depending on the frequency of errors, navigation might become difficult.
Alice (at the vantage point) and Bob (in the maze) use the following code instead:
0000 means forward.
1111 means backwards.
1010 means right.
0101 means left.
If a single error occurs Bob can detect it and ask for re-transmission. This holds because the code words differ to one another in at least 2 digits. We call this code a single-error detecting code.

1.3. A First Example in Cryptography.

Let us keep the same maze setup as in the previous example. However this time imagine a

Constantin Vlachos
constantinosvasiliosargyris.vlachos@uzh.ch

noiseless, insecure channel. Also imagine that Alice and Bob are in a race with Eve, who is also in the maze. Eve doesn't have anyone to guide her; what she does have though is access to the channel. Thus Alice and Bob must communicate using a secret cipher. They decide to use the same code as above. Eve knows this code and assumes you are using it. What Eve doesn't know is that Alice and Bob have agreed upon a secret cipher in which one must read every 0 as a 1 and every 1 as a 0.

Constantin Vlachos
constantinosvasiliosargyris.vlachos@uzh.ch

2. Preliminary Results.

This chapter begins with the presentation of some standard Algebra results that we use later on.
Both cryptography and coding theory often use finite fields and vector spaces over finite fields. This is why in the following chapter we will also explain what they are.
Finally we give some classical definitions and theorems which can be found in any book on Linear Algebra such as [Bos14].

2.1. Algebra Results.

THEOREM 2.1. *(The Euclidean Algorithm) [HPS06]*
Let a and b be positive integers. Assume $b \neq 0$ and $a \geq b$. The following algorithm computes $\gcd(a, b)$ *in a finite number of steps.*

(1) Let $r_0 = a$ and $r_1 = b$.
(2) Set $i = 1$
(3) Perform division with rest to obtain $r_{i-1} = q_i r_i + r_{i+1}$.
(4) If the remainder $r_{i+1} = 0$, then $r_i = \gcd(a, b)$ and the algorithm terminates.
(5) Otherwise, $r_{i+1} > 0$, so set $i = i + 1$ and go back to (3).

The division step is executed at most $2\log_2(b) + 2$ times.

Proof.
We only give the proof of the efficiency since we assume the reader is already familiar with the Euclidean algorithm.

$$\textbf{Claim:} \quad r_{i+2} \leq \frac{1}{2} r_i \quad \forall i \in \mathbb{N}.$$

Case 1: $r_{i+1} < \frac{1}{2} r_i$
We know the values are strictly decreasing, so it follows that

$$r_{i+2} < \frac{1}{2} r_i.$$

Case 2: $r_{i+1} > \frac{1}{2} r_i$
In this case we have that

$$r_i = r_{i+1} + r_{i+2},$$

and thus

$$r_{i+2} = r_i - r_{i+1} < r_i - \frac{1}{2} r_i = \frac{1}{2} r_i.$$

Using this inequality we find that

$$r_{2k+1} < \frac{1}{2} r_{2k-1} < \ldots < \frac{1}{2^k} r_1 = \frac{1}{2^k} b.$$

If $2^k \geq b$, then r_{2k+1} equals 0 and the algorithm terminates.
One can see that the Euclidean algorithm terminates in at most 2k iterations. Choose the smallest k, such that $2^k \geq b \geq 2^{k-1}$.

$$N \leq 2k = 2(k-1) + 2 < 2\log_2(b) + 2,$$

where N is the number of iterations. □

Constantin Vlachos
constantinosvasiliosargyris.vlachos@uzh.ch

THEOREM 2.2. *(Extended Euclidean Algorithm) [HPS06]*
Given four integers $a = r_0$, $b = r_1$, x, y one can write the $\gcd(a, b)$ *as the sum of* $ax + by = \gcd(a, b)$.

We give our own proof.

Proof.
From the Euclidean algorithm one takes that for some i

$$\begin{aligned} \gcd(a, b) &= r_{i+1} && (1)\\ &= r_{i-1} - q_i \cdot r_i && (2)\\ &= r_{i-1} - q_i \cdot (r_{i-2} - q_{i-1} ri - 1) && (3)\\ &= -q_i r_{i-2} + (1 - q_i q_{i-1}) r_{i-1}. && (4) \end{aligned}$$

It follows that

$$r_0 x + r_1 y = \gcd(a, b),$$

for some x, y.

□

THEOREM 2.3. *(Fermat's little theorem) [HPS06]*
Suppose p is prime. If $p \nmid a$, it holds that $a^{p-1} \equiv 1 \pmod{p}$.

Before giving the proof we look at a helpful lemma.

LEMMA 2.4.
Suppose p is prime. If $p \nmid a$, then $\{a, 2a, ..., (p-1)a\}$ are all distinct non-zero numbers modulo p.

Proof. Take two number ja and ka from the list.
If $ja \equiv ka$,then$(j - k)a \equiv 0$,it follows that$j - k \equiv 0$and thus$ja \equiv ka$.
But all numbers on the list are non-zero since $1, ..., p-1$ are not divisors of p and $p \nmid a$. □

Now back to the proof of Fermat's little theorem:

Proof.
One can observe that $a, 2a, ..., (p-1)a \pmod{p}$ is $1, 2, ..., (p-1) \pmod{p}$ in a different order. It follows that

$$a \cdot 2a \cdot ... \cdot (p-1)a \equiv 1 \cdot 2 \cdot ... \cdot (p-1) \pmod{p},$$

giving us

$$a^{p-1} \cdot (p-1)! \equiv (p-1)! \pmod{p}.$$

As both are non-zero and we are in a field as p is prime we may conclude that by multiplication with the inverse of $(p-1)!$ on both sides of the equation we get

$$\implies a^{p-1} \equiv 1 \pmod{p}.$$

□

Constantin Vlachos
constantinosvasiliosargyris.vlachos@uzh.ch

2.2. Vector Spaces over Finite Fields.

Definition 2.5. *(Fields)*
A field is a set $\mathbb{F}$ with two inner-operations $+ : \mathbb{F} \times \mathbb{F} \to \mathbb{F}$ and $\cdot : \mathbb{F} \times \mathbb{F} \to \mathbb{F}$ such that the following properties are satisfied:

(1) $(a + b) + c = a + (b + c)$ for all $a, b, c \in \mathbb{F}$. (additive associativity)
(2) There exists a neutral element e_{add} in $\mathbb{F}$ such that $e_{add} + a = a$ for all $a \in \mathbb{F}$.
(3) Every element $a \in \mathbb{F}$ has an inverse $b \in \mathbb{F}$ such that $a + b = e$.
(4) $a + b = b + a$ for all $a, b \in \mathbb{F}$. (additive commutativity)
(5) $(a \cdot b) \cdot c = a \cdot (b \cdot c)$ for all $a, b, c \in \mathbb{F}$.
(6) There exists a neutral element $e_{mult} \in \mathbb{F}$ such that $a \cdot e_{mult} = a$ for all $a \in \mathbb{F}$.
(7) Every element $a \in \mathbb{F}\backslash\{0\}$ has an inverse such that $a \cdot b = 1$.
(8) $a \cdot b = b \cdot a$ for all $a, b \in \mathbb{F}$. (multiplicative commutativity)
(9) $a \cdot (b + c) = a \cdot b + a \cdot c$ and $(a + b) \cdot c = a \cdot c + b \cdot c$ for all $a, b, c \in \mathbb{F}$. (distributivity)
(10) $1 \neq 0$

This defines any field. Some examples would be $\mathbb{C}$, the complex numbers (1, $1 + 3i$, $\sqrt{2}i$, ...); $\mathbb{Q}$, the rational numbers ($0, \frac{1}{5}, \frac{1}{2}, 1, \frac{4}{3}$...) or $\mathbb{R}$, the real numbers. We are however interested in finite fields. The following definition will give us a tighter grasp on them.

Definition 2.6. *(Finite Fields)*
A finite field or Galois field is a field which has a finite number of elements, this number being called the order of the field.

Definition 2.7.
Let m be a fixed integer. Two integers a, b are said to be congruent modulo m, symbolized

$$a \equiv b \pmod{m}$$

if $a - b$ is divisible by m.

Example 2.8.

$$1 \equiv -1 \pmod{2} \qquad 49 \equiv 0 \pmod{7} \qquad 5 \not\equiv 6 \pmod{8}$$

Modulo calculations are used in finite fields. An easy example is $\mathbb{F}_7$, also written as $\mathbb{Z}/7\mathbb{Z}$. It consists of the elements $\{0, 1, 2, 3, 4, 5, 6\}$. We calculate everything modulo 7.

Lemma 2.9.
A Field F satisfies the condition that for $a, b \in F$, $a \cdot b = 0$ if and only if a or b are equal to zero.

We give our own proof.

Proof.
Assume $a, b \in F^\star$. Then recalling that multiplication over a field is abelian we get

$$1 = aa^{-1}bb^{-1} = aba^{-1}b^{-1} = 0a^{-1}b^{-1} = 0,$$

which leads to a contradiction to the assumption that F is a field, as in a field $1 \neq 0$. □

Lemma 2.10.
Finite fields of the form $\mathbb{Z}/m\mathbb{Z}$, where $m \in \mathbb{Z}$ require that m is a prime number.

We give our own proof.

Proof.
If m is not prime we can choose a, b such that $ab = m$, where $a \not\equiv 0 \pmod{m}$ and $b \not\equiv 0 \pmod{m}$. We observe that ab can be equivalent to zero modulo m. Using Lemma 2.9 we may conclude that $\mathbb{Z}/m\mathbb{Z}$ is not a field. □

Definition 2.11.
Let R be a ring and ϕ be the unique ring homomorphism: $\mathbb{Z} \to R$.
The characteristic of R is the natural number n, such that the $ker(\phi) = n\mathbb{Z}$.

Notation 2.12. We denote the characteristic of a ring R as $\text{char}(R)$.

Remark 2.13.
Let L and K be two fields and let $K \subset L$. Then $\text{char}(K)$=$\text{char}(L)$.
We omit the proof.

Definition 2.14. *Prime Fields*
The smallest subfield of a field K is called the prime field of K.

Lemma 2.15.
Let K be a field. Then one of the following holds.

(1) char(K)=0 and the prime field of K is isomorphic to $\mathbb{Q}$.
(2) char(K)=p, a prime and the prime field of K is isomorphic to $\mathbb{Z}/p\mathbb{Z}$.

Proof.

(1) Let $\text{char}(K)$=0.
Consider the ring homomorphism $\phi : \mathbb{Z} \to K$, with $\mathbb{Z} = \mathbb{Z}/\ker\phi \simeq \text{Im}(\phi)$. It follows that K contains a subring, lets call it SR, isomorphic to $\mathbb{Z}$. As K is a field it contains $\text{Frac}(SR)$ (field of fractions of SR). Thus $\mathbb{Q} \simeq \text{Frac}(SR) \subset K$. This implies that $\mathbb{Q}$ is a prime field. If $F \subseteq \mathbb{Q}$ is a subfield of $\mathbb{Q}$, then $char(F) = 0$ meaning that F' contains a subring isomorphic to $\mathbb{Q}$. It follows that $F = \mathbb{Q}$.
(2) Let $\text{char}(K)$=$n > 0$.
Once more we consider the ring homomorphism $\phi : \mathbb{Z} \to K$. We know that $\text{Im}(\phi) \simeq \mathbb{Z}/n\mathbb{Z}$, so $\phi(\mathbb{Z}) \subset K$. It follows that for any $x, y \in \phi(\mathbb{Z})$

$$x \cdot y = 0 \text{ if and only if } y = 0 \text{ or } x = 0.$$

Recalling Lemma 2.10 this means that n must be prime and $\phi(\mathbb{Z}) \simeq \mathbb{Z}/p\mathbb{Z}$.

□

Theorem 2.16. *Galois Fields*
Let $q > 0$ be an integer. Then there exists a field with q elements if and only if $q = p^n$ for a prime p and an integer $n \geq 1$.

Proof.
We first prove the existence of a field with q elements assuming that $q = p^n$ for a prime p and an integer $n \geq 1$.

- If $n = 1$ then we cane take $\mathbb{F}_p$ to equal $\mathbb{Z}/p\mathbb{Z}$.
  To prove that $\mathbb{Z}/p\mathbb{Z}$ is indeed a field we show the existence of an inverse for every non-zero element, all other field axioms can be checked and hold true.
  Let $a \in \mathbb{Z}/p\mathbb{Z}$. Then by Lemma 2.4 it follows that $\{a, 2a, ..., (p-1)a\}$ all distinct non-zero numbers. As there are only $p-1$ non-zero in $\mathbb{Z}/p\mathbb{Z}$ we have that some $xa = 1$ for an $x \in \{1, ..., p-1\}$.
- If $n \geq 2$, consider the polynomial $f = T^q - T \in \mathbb{F}_p[T]$.
  Let F be the smallest field containing all roots of f. One can recall Fermat's little Theorem and observe that $\mathbb{F}_p \subset F$. Using Remark 2.13 we know that char(L)=p.
  Next, consider $L \subset F$, the set of the roots of f:
  $$L = \{a \in F | a^q = a\}$$
  We prove that L is a subfield of F which implies, by the assumption that F is the smallest field containing L, that $F = L$.

  - $1 \in L$, as $1^q - 1 = 0$.
  - Let $a, b \in L$, then $(-a)^q = (-1)^q a^q = (-1)^q a$. If q is odd then $(-a)^q = -a$ and if q is even then $p = 2$ and $2 \cdot x = 0$ for all $x \in F$. Thus $-1 = 1$ and $(-a)^q = a = -a$.
  - $\left(\frac{1}{a}\right)^q = \frac{1}{a^q} = \frac{1}{a} \in L$.
  - $(a+b)^q = \sum_{k=0}^{q} \binom{q}{k} a^k b^{p-k} = a^q + b^q = a + b \in L$, as $p \mid \binom{p^n}{k}$ for all k except $k = q$ and $k = 0$. (Recall that char(L)=p.
  - $(ab)^q = a^q b^q = ab \in L$

  Hence $L \subset F$ is a subfield and $L = F$. The set L consists of $q < \infty$ elements. It is clear that $|L| \leq \deg(f) = q$ and using the formal derivative one can show that $T^q - T \in \mathbb{F}_p[T]$ has no repeat roots. And so we conclude that there exists a field with q elements for $q = p^n$.

Next, we show that if $\mathbb{F}_q$ is a field with q elements, then q must be a prime power.
As $\mathbb{F}_q$ is a finite field char($\mathbb{F}_q$) cannot equal zero as this would imply that $\mathbb{Q}$ is contained in $\mathbb{F}_q$ contradicting the assumption that $\mathbb{F}_q$ has exactly q elements. Thus by Lemma 2.15 it holds that char(L)=p for p a prime.
It follows that $\mathbb{F}_p \subset \mathbb{F}_q$ and that $\mathbb{F}_q$ can be viewed as an $\mathbb{F}_p$-vector space of finite dimension n. Hence $\mathbb{F}_q$ consists of exactly p^n elements and $q = p^n$. (Note: vectorspaces and dimension are defined below.) □

After having extensively looked at finite fields we now introduce vector spaces, first in general and then over finite fields.

Constantin Vlachos
constantinosvasiliosargyris.vlachos@uzh.ch

Definition 2.17. *(Vector Spaces)* (p.34, [Bos14])
Let F be a field. A vector space over $\mathbb{F}$ is a set with an inner operation $V \times V \to V$, $(a, b) \mapsto a + b$ called addition and an outer operation $F \times V \to V$, $(\lambda, a) \mapsto \lambda \cdot v$ called scalar multiplication such that the following holds:

(1) Addition over V fulfills commutivity and associativity, the existence of a neutral and an inverse element.
(2) $(\lambda + \mu)a = \lambda a + \mu a$ for all $\lambda, \mu \in \mathbb{F}$ and $a \in V$.
(3) $\lambda(a + b) = \lambda a + \lambda b$ for all $\lambda \in \mathbb{F}$ and $a, b \in V$.
(4) $1 \cdot a = a$ for all $a \in V$.

Notation 2.18.
We call the elements of a vector space vectors.

Remark 2.19. *Vector spaces over finite fields*
Vector spaces over finite field work completely analogous to general vector spaces defined above. The only difference is that the field $\mathbb{F}$ is finite.

Definition 2.20. *(Span)*
The span of set of vectors $\{v_1, ..., v_n | v_i \in V\}$ is the set

$$S = \{\sum_{i=1}^{n} \lambda_i v_i\},$$

where $\lambda_i \in K$ for all $i = 1, ..., n$.

Notice that if V is the span of v_1, v_2, v_3 then V is also the span of v_1, v_2, v_3, v_4 if $v_4 \in V$ that is to say if $v_4 = \lambda_1 v_1 + \lambda_2 v_2 + \lambda_3 v_3$.

Definition 2.21. *(Linear dependence)*
A set of vectors $\{v_1, ..., v_n\}$ are linearly independent if $\lambda_1 v_1 + ... + \lambda_n v_n = 0$ only holds when all coefficients are zero.

Definition 2.22. *(Basis)*
A set $B = \{v_1, ..., v_n\}$ is a basis of V if $\{v_1, ..., v_n\}$ span V and are linearly independent.

Lemma 2.23.
Let V be a K-vector space, $B = \{v_1, ..., v_n\}$ a basis of V and $v = v_i - v_j$. Then $\hat{B} = \{v_1, ..., v_{i-1}, v, v_{i+1}, ..., v_n\}$ is also a basis of V.
We give our own proof.

Proof.
To prove the statement we must show that

(1) the vectors in $\hat{B}$ are linearly independent,
(2) the vectors in $\hat{B}$ are span V.

(1) By assumption we know that for $\mu \in K$

$$\mu_1 v_1 + ... + \mu_{i-1} v_{i-1} + \mu_v v + \mu_{i+1} v_{i+1} + ... + \mu_n v_n \tag{5}$$

$$= \mu_1 v_1 + ... + \mu_v v_i - \mu_v v_j + ... + \mu_j v_j + ... + \mu_n v_n \quad (6)$$
$$= \mu_1 v_1 + ... + \mu_v v_i + ... + (\mu_j - \mu_v) v_j + ... + \mu_n v_n. \quad (7)$$

By assumption $v_1, ..., v_n$ are linearly independent. This means that for $\lambda \in K$

$$\lambda_1 v_1 + ... + \lambda_n v_n = 0 \iff \lambda_k = 0 \; \forall k.$$

It follows that all the coefficients in (3) must equal zero, specifically μ_v and $(\mu_j - \mu_v)$ in order for (3) to equal zero.

(2) By assumption any vector $a \in V$ can be written as

$$a = \lambda_1 v_1 + ... + \lambda_n v_n.$$

We observe that since $v = v_i - v_j$ that same vector a can be written as

$$a = \lambda_1 v_1 + ... + \lambda_i v + ... + (\lambda_i + \lambda_j) v_j + ... + \lambda_n v_n.$$

It follows that $\hat{B}$ spans V.

□

From Lemma 2.23 it follows that bases are not unique.

Definition 2.24. *(Dimensions)*
The dimension of V is the minimum number of vectors that span V.

Remark 2.25.
One can show that it is equivalent to define the dimension as the cardinality of any basis.

Definition 2.26. *(Subspaces)*
We define a linear subspace C of a vector space V to be a non-empty subset of V that is closed under addition and scalar multiplication of V.

Theorem 2.27.
Suppose V is an n-dimensional vector space with basis $v_1, ..., v_n$. Then every element of V can be expressed uniquely by a linear combination of the basis vectors.

Proof.
Since the basis vectors are linearly independent it follows that

$$\lambda_1 v_1 + ... + \lambda_n v_n = 0$$

if and only if all coefficients λ_i for all i=1,...,n equal 0.
So if $\lambda_1 v_1 + ... + \lambda_n v_n = \mu_1 v_1 + ... + \mu_n v_n$ it follows that $\sum_{i=1}^{n} (\lambda_i - \mu_i) v_i = 0$, which means $\lambda_i = \mu_i$ for all $i = 1, ..., n$.

□

Notation 2.28.
When writing $v \in V$ as the sum of all the basis vectors $v = \sum_{i=1}^{n} \lambda_i v_i$ the coefficients λ_i are also known as the vectors coordinates.

Constantin Vlachos
constantinosvasiliosargyris.vlachos@uzh.ch

Definition 2.29. *(Determinant)*
Let K-vectorspace $V = Mat_{n\times n}(K)$. Then the function $det : V \to K$. We define the determinant of an $n \times n$ matrix A as

$$det(A) = \sum_{\sigma \in S_n} Sign(\sigma) a_{1\sigma(1)} a_{2\sigma(2)} \cdot ... \cdot a_{n\sigma(n)}.$$

Alternatively one can define it axiomatically:

(1) The determinant function is multilinear in its rows i.e.

$$det \begin{pmatrix} ... & a_1 & ... \\ ... & a_2 & ... \\ ... & \vdots & ... \\ ... & \lambda a_i + \mu b_i & ... \\ ... & \vdots & ... \\ ... & a_n & ... \end{pmatrix} = \lambda \cdot det \begin{pmatrix} ... & a_1 & ... \\ ... & a_2 & ... \\ ... & \vdots & ... \\ ... & a_i & ... \\ ... & \vdots & ... \\ ... & a_n & ... \end{pmatrix} + \mu det \begin{pmatrix} ... & a_1 & ... \\ ... & a_2 & ... \\ ... & \vdots & ... \\ ... & b_i & ... \\ ... & \vdots & ... \\ ... & a_n & ... \end{pmatrix}$$

Note that the function is also multilinear in its columns. This also follows from Leibniz's definition.

(2) The determinant function is alternating, i.e.

$$det \begin{pmatrix} ... & a_1 & ... \\ ... & a_2 & ... \\ ... & \vdots & ... \\ ... & a_i & ... \\ ... & a_j & ... \\ ... & \vdots & ... \\ ... & a_n & ... \end{pmatrix} = -det \begin{pmatrix} ... & a_1 & ... \\ ... & a_2 & ... \\ ... & \vdots & ... \\ ... & a_j & ... \\ ... & a_i & ... \\ ... & \vdots & ... \\ ... & a_n & ... \end{pmatrix}.$$

Theorem 2.30.
Let A be a matrix with non-zero determinant and $\hat{A}$ be a matrix obtained via row or column operations on A, then the determinant of $\hat{A}$ is also non-zero.

We give our own proof.

Proof.
Gaussian row (or column) operations can be represented by a matrix multiplication. To multiply a row with a scalar λ is the same as to multiply the entire matrix with matrix

$$S = \begin{pmatrix} 1 & & & \\ & \ddots & & 0 \\ & & \lambda & \\ & 0 & & \ddots \\ & & & & 1 \end{pmatrix}.$$

The determinant of S is clearly λ.
Row permutation for some rows i, j can be achieved by multiplying the matrix with matrix

$$P = \begin{pmatrix} 1 & & & & & & 0 \\ & \ddots & & & & & \\ & & 0 & & 1 & & \\ & & & \ddots & & & \\ & & 1 & & 0 & & \\ & & & & & \ddots & \\ 0 & & & & & & 1 \end{pmatrix}.$$

The determinant of P is -1 (determinant is alternating).

Finally row the addition of a scalar multiple (λ) of one row to another can be viewed as the matrix multiplied with

$$Q = \begin{pmatrix} 1 & & & & 0 \\ & \ddots & & & \\ & \lambda & 1 & & \\ & & & \ddots & \\ 0 & & & & 1 \end{pmatrix}.$$

The determinant of Q is 1. Hence any matrix $\hat{A}$ attained by Gaussian row (or column) operations on a matrix A with $det(A) \neq 0$ has a non-zero determinant.

□

Remark 2.31.
Let A be an $n \times n$ matrix with $det(A) \neq 0$. Then the n rows (or columns) of A are linearly independent. This follows directly from Theorem 2.30.

Theorem 2.32. *(Vandermonde Matrix)*
The determinant of a Vandermonde Matrix V is equal to $\Pi_{i>j}(a_i - a_j)$, where

$$V = \begin{pmatrix} 1 & a_1 & a_1^2 & ... & a_1^{n-1} \\ 1 & a_2 & a_2^2 & ... & a_2^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & a_n & a_n^2 & ... & a_n^{n-1} \end{pmatrix}$$

and all a_i for $i = 1, ..., n$ are non-zero and distinct.

Proof. (By Induction)
Base Case: Over $V = Mat_{2x2}$.

$$det \begin{pmatrix} 1 & a_1 \\ 1 & a_2 \end{pmatrix} = a_2 - a_1.$$

Induction Hypothesis: The theorem holds for $n-1 \times n-1$ Vandermonde matrices for a fixed $n-1$.
Induction Step:
Case 1:
There exists i, j such that $a_i = a_j$. Then $det(A) = det \begin{pmatrix} 1 & a_1 & a_1^2 & ... & a_1^{n-1} \\ 1 & a_2 & a_2^2 & ... & a_2^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & a_n & a_n^2 & ... & a_n^{n-1} \end{pmatrix} = 0.$

It follows that $det(A) = \Pi_{i<j}(a_j - a_i)$.
Case 2:
There exist no i, j such that $a_i = a_j$. Let $a_n = x$.
We know from Leibniz's formula that

$$det(A) = c_{n-1}x^{n-1} + c_{n-2}x^{n-2} + ... + c_0.$$

We also know that $det(A) = 0$ if $x = a_i \quad i = 1, ..., n-1$. It follows that

$$det(A) = k \cdot \Pi_{i=1}^{n-1}(x - a_i),$$

where k is a costant.
Now the last question left to answer is what k equals. From the Leibniz formula we see that

$$det(A) = \sum_{\sigma \in S_n} sign(\sigma) \cdot a_{1\sigma(1)} \cdot ... \cdot a_{n\sigma(n)}.$$

However $\Pi_{i=1}^{n-1}(a_i - x)$ gives us all combinations with $x = a_n$ and it follows

$$det(A) = \sum_{\sigma \in S_n} sign(\sigma) \cdot a_{1\sigma(1)} \cdot ... \cdot a_{n-1\sigma(n-1)} \cdot \Pi_{i=1}^{n-1}(a_i - x).$$

This means that $k = det(\hat{A})$, where $\hat{A} = \begin{pmatrix} 1 & a_1 & a_1^2 & ... & a_1^{n-2} \\ 1 & a_2 & a_2^2 & ... & a_2^{n-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & a_{n-1} & a_{n-1}^2 & ... & a_{n-1}^{n-2} \end{pmatrix}$.

This concludes the proof as

$$det(A) = \Pi_{1 \le i < j \le n-1}(a_j - a_i) \cdot \Pi_{i=1}^{n-1}(a_i - x) = \Pi_{1 \le i < j \le n}(a_j - a_i).$$

□

Constantin Vlachos
constantinosvasiliosargyris.vlachos@uzh.ch

3. Coding Theory over Finite Fields.

As mentioned in chapter one coding theory solves the problem of the detection and correction of errors caused by noise. We start explaining how this is done in greater detail by introducing some terminology.

Definition 3.1. *(q-ary Codes)* [Hil86]
A q-ary code is a set of words, where each word is a string made up of elements of an alphabet $\mathbb{F}_q = \{\lambda_1, ..., \lambda_q\}$.

We call the words in a code code words.

Definition 3.2. *(Block Codes)* [Hil86]
A block code of length n is a code in which each code word is a string of a fixed length of n symbols.

code words of a q-ary blockcode of length n all reside in $(\mathbb{F}_q)^n$. In this paper we concern ourselves with nearest neighbour decoding. This means that we assume every symbol in the sequence of n symbols has the same probability which must be less than $\frac{1}{2}$ of being falsified by noise. Should an error occur we look for the closest code word to the received message. But how can we determine if two messages are "close" to one another? Hamming distance answers this question.

Definition 3.3. *(Hamming Distance)* [Hil86]
Hamming distance is a distance function on $(\mathbb{F}_q)^n$. The distance between two vectors $x, y \in (\mathbb{F}_q)^n$ is the number of entries in which they differ.

Notation 3.4.
The hamming distance of two vectors $x, y \in (\mathbb{F}_q)^n$ is denoted by $\mathrm{d}(x, y)$.

We note that the Hamming distance satisfies the conditions of a distance metric:

- $\mathrm{d}(x, y) = 0$ if and only if $x = y$
- $\mathrm{d}(x, y) = \mathrm{d}(y, x)$for all $x, y \in (\mathbb{F}_q)^n$.
- $\mathrm{d}(x, y) \leq \mathrm{d}(x, z) + \mathrm{d}(z, y)$for all $x, y, z \in (\mathbb{F}_q)^n$

Definition 3.5. *(Minimal distance)*
We define the minimum distance of a code C as

$$d(C) := \min\{d(x, y) \,|\, x \neq y\}.$$

Before moving on to linear codes we give the reader one very vital theorem that will also be used in the ISBN experiment which can be found in chapter four.

Theorem 3.6. *[Hil86]*
(1) A code C can detect a maximum of s errors given that $d(C) \geq s + 1$.

Constantin Vlachos
constantinosvasiliosargyris.vlachos@uzh.ch

(2) A code C can correct a maximum of t errors given that d(C)≥ 2t + 1.

Proof.

(1) If a message with s errors is received it cannot be mistaken for a different code word as d$(C)= s+1$, hence it is identified as an error. This also holds for less than s errors.

(2) Let d(C)=$2t + 1$. If a code word x is transmitted and the message y is received with t or less errors this means d$(x, y)\leq t$. For a code word $x' \neq x$, d$(x', y)\geq t+1$ holds. One can prove this by recalling the triangle inequality which gives us:

$$d(x, x') \leq d(x, y) + d(x', y).$$

If d$(x', y)$is less than $t + 1$ this would make d(x, y)+ d$(x', y)\leq 2t$ and contradict the assumption that d(C)=$2t + 1$.

□

3.1. The Advantages of Linear Codes.

Definition 3.7.
We define a q-ary linear block code of length n as a subspace of $\mathbb{F}_q^n$.

The vectors of such a subspace which we write $x_1x_2...x_n$ are the code words. Naturally the 0-vector belongs to every linear code. Regarding notation we specify:

- the alphabet (q-ary)
- the block size (n)
- the dimension of the subspace (k)
- the minimum distance (d)

Thus when describing a linear code one can speak of an [n,k,d] code.
In the following we give the reader some theorems, lemmas and defitions such that they might familiarize themselves with linear codes.

Definition 3.8. *(Weight)*
The weight $w(x)$ of a $x \in \mathbb{F}_q^n$ is defined as the number of non-zero entries.

We now give a lemma to help us prove the following theorem, that the minimum distance of a linear code equals the smallest weight of a non-zero code word.

Lemma 3.9. *[Hil86]*
If code words x and $y \in \mathbb{F}_q^n$, then

$$d(x, y) = w(x - y).$$

Proof.
The vector $x - y$ has non-zero entries exactly where x and y have different entries. □

THEOREM 3.10. *(Weight and minimum distance) [Hil86]*
Let C be a linear code and let $w(C)$ be the smallest of weights of the non-zero code words. Then

$$d(C) = w(C).$$

We give our own proof.

Proof.
Assume a code word x has the smallest weight of the non-zero code words. This would mean that w(C)=d$(x, 0)$ since C is linear and 0 therefore is a code word.
If code words z and y were to exist such that d$(z, y)<$ d$(x, 0)$ then the code word $b = z - y$ satisfies $w(z - y) =$ w$w(b) < w(x)$ which would be a contradiction to our assumption about x.

□

As shown in the first chapter vectorspaces, this also means subspaces can be described using a basis. We now show one of the great advantages of linear codes, the generator matrix.

DEFINITION 3.11. *(Generator matrix)*
A $k \times n$ matrix whose rows form a basis of code C is called a generator matrix.

EXAMPLE 3.12.
Let linear [3,2,1]-code C with code words

(1) 000
(2) 100
(3) 010
(4) 110.

C can be described with the generator matrix G:

$$G = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}.$$

Notice how the code words 110 and 000 can be written as sum of the rows of G.

The reader may recall Lemma 2.23, in which we proved that bases are not unique. Thus it follows that a code C can have multiple generator matrices.

The next topic we speak on is dual codes and the parity-check matrix. Once the reader is familiar with these we can look into the encoding and (syndrome) decoding of linear codes. After this we will give a quick summary of the advantages of linear codes.

Constantin Vlachos
constantinosvasiliosargyris.vlachos@uzh.ch

Definition 3.13.
For two vectors $x = x_1, ..., x_n, u = u_1, ..., u_n$ both in $\mathbb{F}_q^n$, $x \cdot u$ is defined as

$$u_1x_1 + u_2x_2 + ... + u_nx_n.$$

Definition 3.14. *(Orthogonal vectors)*
Two vectors $x, u \in \mathbb{F}_q^n$ are orthogonal if $x \cdot u = 0$.

Definition 3.15. *(Dual code)* [Hil86]
Let C be a k-dimensional linear code over $\mathbb{F}_q^n$. The dual code of C, denoted by $C^\perp$, is defined to be the set of those vectors of $\mathbb{F}_q^n$ which are orthogonal to every code word of C, i.e.

$$C^\perp = \{v \in \mathbb{F}_q^n \mid v \cdot x = 0 \; \forall x \in C\}.$$

From a classic linear algebra result it follows that if C is a k-dimensional linear code over $\mathbb{F}_q^n$ then $C^\perp$ is a $(n-k)$-dimensional linear code over $\mathbb{F}_q^n$ and $(C^\perp)^\perp = C$.

Definition 3.16. *(Parity-check matrix)* [Hil86]
A parity-check matrix H of linear k-dimensional code C over $\mathbb{F}_q^n$ is the $(n-k) \times n$ generator matrix of $C^\perp$.

By definition of H and $C^\perp$ for all $x \in C$

$$xH^T = 0.$$

This means a linear code can be described by its parity-check matrix as well as its generator matrix.

Before we move on to the encoding and decoding of linear codes we give a very important theorem, which will later be used in the ISBN experiment.

Theorem 3.17. *(Fundamental theorem) ([Hil86], Theorem 8.4 p.52)*
Suppose C a k-dimensional linear code over $\mathbb{F}_q^n$ with parity-check matrix H. Then the minimum distance of C is d if and only if any d-1 columns of H are linearly independent but some d columns are linearly dependent.

Proof.
We know in linear codes the minimum distance is equal to the weight of the smallest weighted code word. Let $x = x_1x_2...x_n \in \mathbb{F}_q^n$. We know

$$x \in C \iff xH^T = 0 \iff x_1H_1 + ... + x_nH_n = 0$$

where $H_1, ..., H_n$ are the columns of H hence the rows of H^T. As a code word of weight d exists it follows that there are d linearly dependent columns in H. If there were d-1 linear dependent columns in H this would contradict the fact that d is the minimum distance. □

Constantin Vlachos
constantinosvasiliosargyris.vlachos@uzh.ch

Encoding [Hil86]
Let C be a k-dimensional linear code over $\mathbb{F}_q^n$ and let G be its generator matrix. C contains q^k code words, hence there are q^k messages that can be sent. We view these messages as vectors in $\mathbb{F}_q^k$ and encode a vector $u \in \mathbb{F}_q^k$ by multiplying it on the right by G. G maps the vectorspace $\mathbb{F}_q^k$ onto a k-dimensional subspace of $\mathbb{F}_q^n$. If G is in standard form $[I_k \mid A]$ the encoding becomes easier:

$$uG = x = x_1 x_2 ... x_n,$$

where $x_i = u_i$ for all $1 \leq i \leq k$ and x_i with $i > k$ are the product of $\sum_{j=1}^{k} a_{ji} u_j$.
We call x_i's where $i \leq k$ *message digits*. If i were to be greater than k the according x would be called a *check digit*.

Decoding [Hil86]
Let $x = x_1 x_2 ... x_n$ be a code word sent through a channel and let $y = y_1 y_2 ... y_n$. The recipient now decides based on y which code word was sent.

Definition 3.18. *(Coset)*
Suppose that C is k-dimensional linear code over $\mathbb{F}_q^n$ and $a \in \mathbb{F}_q^n$. Then $a + C$ is defined as $a + C =: \{a + c \mid c \in C\}$.

Lemma 3.19.
Suppose $a + C$ is a coset of C and $b \in a + C$. Then $b + C = a + C$.

We give our own proof.

Proof.
Since $b \in a + C$ it follows that $b = a + x$ where $x \in C$. It also follows that $a = b - x$. Every element in $a + C$ can be written $b - x + x'$ and every element in $b + C$ can be written $a + x + x''$, where $x', x'' \in C$. □

Theorem 3.20. *(Lagrange)*
Suppose that C is k-dimensional linear code over $\mathbb{F}_q^n$ and $a \in \mathbb{F}_q^n$. Then

(1) every coset contains q^k vectors,
(2) two cosets are disjoint or coincide.

We give our own proof.

Proof.

(1) There are $|C|$ of distinct vectors in C. Hence it follows that there are $|C|$ different combinations of a vector in C and some vector $a \in \mathbb{F}_q^n$.
(2) Assume $y \in a + C$ and $y \in b + C$ with $b \neq a$. This would mean that for some $x, x' \in C$

$$a + x = b + x'.$$

From this it would follow that $a = b + x' - x = b + x''$. This would mean that $a \in b + C$ and by the lemma above that $a + C = b + C$.

□

Syndrome Decoding

Using the lemmas and theorems above we introduce a method of finding the code word x that was sent using a linear code C from a received message y.

Definition 3.21. *(Syndrome)* [Hil86]
Let H be a parity-check of some linear code C over $\mathbb{F}_q^n$ and let y be some received message vector in $\mathbb{F}_q^n$. We call

$$s(y) = yH^T$$

the syndrome of y.

From the definition one can notice that $s(y) = y \cdot h_1, y \cdot h_2, ..., y \cdot h_n$ with h_i denoting the rows of H. We also can see that if $y \in C$, then $s(y) = 0$.

Lemma 3.22. *[Hil86]*
If two vectors are in the same coset they have the same syndrome.

Proof.
If two vectors, y and y' are in the same coset $a + C$ we can write them as $y = a + x$, $y' = a + x'$.

$$s(y) = (a + x)H^T = aH^T + xH^T = aH^T + 0 = aH^T + x'H^T = s(y').$$

This holds because $x, x' \in C$ and $s(x) = 0$ for all $x \in C$.

□

In this lemma we essentially proved that there is a link from a specific coset to a specific syndrome. So in order to decode a received message y we use the following algorithm.

(1) Create a look-up table where the code words of C are listed in the first row, the cosets of C are listed in the rows beneath and the syndromes of the respective cosets are denoted next to them, e.g.

$$\begin{pmatrix} x_1 & x_2 & x_3 & ... \\ a+x_1 & a+x_2 & a+x_3 & ... \\ b+x_1 & b+x_2 & b+x_3 & ... \\ c+x_1 & c+x_2 & c+x_3 & ... \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix} \begin{pmatrix} 00 \\ S(a) \\ S(b) \\ S(c) \\ \vdots \end{pmatrix}.$$

It is clear that the maximum weight of $a, b, c, ...$ is $\frac{d(C)-1}{2}$.
(2) Calculate $S(y)$ of received vector y.
(3) Locate $S(y)$ in the syndrome column of the look-up table, if $S(y)$ is not in the table ask for re-transmission.

(4) Compare y to the vectors listed in the row in which $S(y)$ is listed. If a match is found assume the sent vector was the code word listed in the first row of the column in which the match was found.

3.2. BCH-Codes.

A BCH code C is a t-error correcting code over a finite field $\mathbb{F}_q$ where $2t+1 \leq d(C)$. The decoding of such codes requires solving a certain system of simultaneous non-linear equations. We use Ramanujan's method.
Typical for BCH codes is that the parity check matrix looks like a Vandermonde matrix as introduced in Theorem 2.32. The difference is that Vandermonde matrices are always $m \times m$ matrices for some m and the parity check matrix of a BCH code is not necessarily. Constructing such a matrix can be done as follows. If one desires a t-error correcting code of length n over $\mathbb{F}_q$ one finds an n such that

$$2t+1 \leq n \leq q-1.$$

The parity check matrix H meeting the conditions above has the following form

$$H = \begin{pmatrix} 1 & 1 & ... & 1 \\ a_1 & a_2 & ... & a_n \\ \vdots & \vdots & \ddots & \vdots \\ a_1^{d-2} & a_2^{d-2} & ... & a_n^{d-2} \end{pmatrix},$$

where $d \leq n \leq q-1$ and $a_i \;\; i = 1, ..., n$ are distinct non-zero elements of $\mathbb{F}_q$. This is important because:

(1) $a_i, \;\; i = 1, ..., n$ are distinct non-zero elements of $\mathbb{F}_q$ because we want the determinant of a Vandermonde matrix with columns from H to have a non-zero determinant.
(2) $d-1$ must be less than n, such that $d-1$ columns are linearly independent but d columns are linearly dependent, since we want $d(C) = \mathrm{d}$ (see Theorem 3.17). We know that the rows and columns are linearly independent in a Vandermonde matrix with a non-zero determinant.
(3) n cannot be greater than $q-1$ since we want the entries a_i of the parity check matrix to be distinct and non-zero. $q-1$ is the number of distinct non-zero elements in $\mathbb{F}_q$.
(4) We know the zero vector is in C, hence if $n < 2t+1$ it follows that $d(C) < 2t+1$, resulting in C not being a t-error correcting code.

Remark 3.23.
With the given parity-check matrix, a code C can be described as follows:

$$C = \left\{ x_1 x_2 ... x_n \in \mathbb{F}_q^n \mid \sum_{i=1}^{n} a_i^j x_i = 0 \;\; j = 0, 1, ..., d-2 \right\}.$$

Constantin Vlachos
constantinosvasiliosargyris.vlachos@uzh.ch

3.2.1. Decoding Algorithm. The following section is based on chapter 11 of Raymond Hill's *"A first course in coding theory"* [Hil86]. In this paper you will find an almost identical theoretical part, however with different examples.
We present a decoding algorithm for BCH codes and follow it up by giving an example where we receive a vector y with a specific syndrome and attempt to find the vector x that was sent.
Before giving the decoding algorithm the reader may note that encoding works exactly the same as with the previous linear codes ($u \cdot G = x$).
For the decoding algorithm we assume $d = 2t + 1$ is odd and that H has 2t rows (0 until $2t - 1$).
We assume code word $x = x_1x_2...x_n$ was sent and received as $y = y_1y_2...y_n$ with t errors (in y). Suppose these errors have occurred in positions $X_1, X_2, ...X_t$ with respective magnitudes $m_1, m_2, ..., m_t$.
First the syndrome of y is computed, that is

$$S(y) = yH^T = (S_1, S_2, ..., S_{2t})$$

where

$$S_j = \sum_{i=1}^{n} a_i^{j-1} y_i = \sum_{i=1}^{t} m_i X_i^{j-1},$$

for $j = 1, ..., 2t$ (compare rows of H).
Next we have to solve the following non-linear system of equations for m_i and X_i:

$$\begin{cases} S_1 = m_1 + m_2 + ... + m_t, \\ S_2 = m_1X_1 + m_2X_2 + ... + m_tX_t, \\ S_3 = m_1X_1^2 + m_2X_2^2 + ... + m_tX_t^2, \\ \vdots \qquad \vdots \qquad \vdots \qquad \vdots \\ S_{2t} = m_1X_1^{2t-1} + m_2X_2^{2t-1} + ... + m_tX_t^{2t-1}. \end{cases}$$

This is done by using Ramanujan's method where on sets

$$\phi(\theta) = \frac{m_1}{1 - X_1\theta} + \frac{m_2}{1 - X_2\theta} + ... + \frac{m_t}{1 - X_t\theta}.$$

This is the sum of

$$m_1(1 + X_1\theta + X_1^2\theta^2 + ...) + m_2(1 + X_2\theta + X_2^2\theta^2 + ...) + ... + m_t(1 + X_t\theta + X_t^2\theta^2 + ...).$$

This however is equal to

$$(m_1 + m_2 + ... + m_t) + (m_1X_1 + m_2X_2 + ... + m_tX_t)\theta + (m_1X_1^2 + m_2X_2^2 + ... + m_tX_t^2)\theta^2 + ...,$$

which can also be written as the sum of the syndromes above multiplied by θ to some power. Hence

$$\phi(\theta) = S_1 + S_2\theta + S_3\theta^2 + ... + S_{2t}\theta^{2t-1} + ...$$

Constantin Vlachos
constantinosvasiliosargyris.vlachos@uzh.ch

From the original way $\phi(\theta)$ was written one can receive the following equation by reducing the fractions to a common denominator:

$$\phi(\theta) = \frac{A_1 + A_2\theta + A_3\theta^2 + ... + A_t\theta^{t-1}}{1 + B_1\theta + B_2\theta^2 + ... + B_t\theta^t}.$$

It follows that

$$(S_1+S_2\theta+S_3\theta^2+...)(1+B_1\theta+B_2\theta^2+...+B_t\theta^t) = A_1+A_2\theta+...+A_t\theta^{t-1}+0\cdot\theta^t+0\cdot\theta^{t+1}+...$$

Now one has a new system of equations that is considerably easier to solve, namely

$$\begin{cases} A_1 = S_1, \\ A_2 = S_1B_1 + S_2, \\ A_3 = S_1B_2 + S_2B_1 + S_3, \\ \vdots \\ A_t = S_1B_{t-1} + S_2B_{t-2} + ... + S_{t-1}B_1 + S_t, \\ 0 = S_1B_t + S_2B_{t-1} + ... + S_tB_1 + S_{t+1}, \\ 0 = S_2B_t + S_3B_{t-1} + ... + S_{t+1}B_1 + S_{t+2}, \\ \vdots \\ 0 = S_tB_t + S_{t+1}B_{t-1} + ... + S_{2t-1}B_1 + S_{2t}. \end{cases}$$

From the last t equations one can find the B_i's and since the syndromes are known the A_i's can be calculated with ease.
The last step is to split the rational function of $\phi(\theta) = \frac{A_1+A_2\theta+A_3\theta^2+...+A_t\theta^{t-1}}{1+B_1\theta+B_2\theta^2+...+B_t\theta^t}$ into partial fractions to get

$$\phi(\theta) = \frac{m_1}{1 - X_1\theta} + \frac{m_2}{1 - X_2\theta} + ... + \frac{m_t}{1 - X_t\theta}.$$

Definition 3.24. *(Error-locator Polynomial)*
We define the error locator polynomial as

$$\sigma(\theta) = 1 + B_1\theta + B_2\theta^2 + ... + B_t\theta^t.$$

Note that the zeros of the error-locator polynomial are the inverses of the error locations $X_1, ..., X_t$.

Definition 3.25. *(Error-evaluator Polynomial)*
We define the error-evaluator polynomial to be

$$\omega(\theta) = A_1 + A_2\theta + ... + A_t\theta^{t-1}.$$

Remark 3.26.
$\omega(\theta) = A_1 + A_2\theta + ... + A_t\theta^{t-1}$ is referred to as the error-evaluator polynomial since one can calculate the error magnitudes with it in the following way:

$$\frac{m_1}{1 - X_1\theta} + \frac{m_2}{1 - X_2\theta} + ... + \frac{m_t}{1 - X_t\theta} = \frac{A_1 + A_2\theta + ... + A_t\theta^{t-1}}{(1 - X_1)(1 - X_2\theta) \cdot ... \cdot (1 - X_t\theta)}.$$

Consequently

$$m_1 + \frac{m_2(1 - X_1\theta)}{1 - X_2\theta} + \frac{m_3(1 - X_1\theta)}{1 - X_3\theta} + ... + \frac{m_t(1 - X_1\theta)}{1 - X_t\theta} = \frac{A_1 + A_2\theta + ... + A_t\theta^{t-1}}{(1 - X_2) \cdot ... \cdot (1 - X_t)}.$$

Since θ is variable we can choose it to $\theta = X_1^{-1}$.
It follows that

$$m_1 = \frac{A_1 + A_2\theta + ... + A_t\theta^{t-1}}{(1 - X_2\theta) \cdot ... \cdot (1 - X_t\theta)}$$

and as $X_1, ..., X_t$ are known to us, we can compute this.

Remark 3.27. (Less than t errors)
One can determine the number of errors that have occurred by regarding the system of linear equations. If $e < t$ errors have occurred then $t - e \neq 0$ equations will be linearly dependent of the first e. This naturally follows from the fact that if e error occur the error locator polynomial has a degree of e. Hence all coefficients (the B_i with $i > e$) equal 0. But then the system of equations, which is

$$\begin{cases} 0 = S_1B_t + S_2B_{t-1} + ... + S_tB_1 + S_{t+1}, \\ 0 = S_2B_t + S_3B_{t-1} + ... + S_{t+1}B_1 + S_{t+2}, \\ \vdots \\ 0 = S_tB_t + S_{t+1}B_{t-1} + ... + S_{2t-1}B_1 + S_{2t}, \end{cases}$$

turns into

$$\begin{cases} 0 = S_{t-e+1}B_e + S_{t-e+2}B_{e-1} + ... + S_tB_1 + S_{t+1}, \\ 0 = S_{t-e+2}B_e + S_{t-e+3}B_{e-1} + ... + S_{t+1}B_1 + S_{t+2}, \\ \vdots \\ 0 = S_{2t-e}B_e + S_{2t-e+1}B_{e-1} + ... + S_{2t-1}B_1 + S_{2t}. \end{cases}$$

This system of equations has t equations and $e < t$ unknowns. Hence the rank of matrix $M \in Mat_{t\times t}$, where

$$M \begin{pmatrix} B_1 \\ B_2 \\ \vdots \\ B_e \\ 0 \\ \vdots \end{pmatrix} = \begin{pmatrix} -S_{t+1} \\ -S_{t+2} \\ \vdots \\ -S_{2t} \end{pmatrix},$$

is not full, since we only have e pivot elements.

Summary of decoding algorithm (for a received vector y assuming $\leq t$ errors)

(1) Calculate the syndrome $(S_1, S_2, ..., S_{2t})$.
(2) Determine the number of errors, e and set all $B_i = 0$ where $i > e$.
(3) Solve the second system of equations (with A_i's, S_i's and B_i's.

Constantin Vlachos
constantinosvasiliosargyris.vlachos@uzh.ch

(4) Find zeros of $\sigma(\theta)$, which is a polynomial of degree e.
(5) Calculate the error magnitudes using $\omega(\theta)$ and $X_1, ..., X_e$.

3.2.2. Example: Decoding a vector with a BCH code.
Consider the BCH code C over $\mathbb{F}_{11}^7$ with parity-check matrix

$$H = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & 4 & 9 & 5 & 3 & 3 & 5 \\ 1 & 8 & 5 & 9 & 4 & 4 & 9 \\ 1 & 5 & 4 & 5 & 9 & 3 & 5 \\ 1 & 10 & 1 & 9 & 1 & 7 & 10 \end{pmatrix}.$$

We know there are exactly $2t$ rows in the parity-check matrix of a t error correcting code, so C corrects 3 errors.
Imagine a vector y was received with syndrome $S(y) = (S_1, ..., S_6) = (9, 7, 9, 0, 5, 7)$
We receive the system of equations

$$\begin{cases} 0 = 0 + 9B_1 + 7B_2 + 9B_3, \\ 0 = 5 + 0B_1 + 9B_2 + 7B_3, \\ 0 = 7 + 5B_1 + 0B_2 + 9B_3, \end{cases}$$

giving us

$$B_1 = 0, B_2 = 1, B_3 = 9.$$

It is clear from the system that three errors have occurred and not one or two. Using the values of B_1, B_2B_3 we solve

$$\begin{cases} A_1 = 9, \\ A_2 = 7 + 0 \cdot 9, \\ A_3 = 9 + 0 \cdot 7 + 1 \cdot 9, \end{cases}$$

and receive

$$A_1 = 9, A_2 = 7, A_3 = 7.$$

Next is the factoring of the *error-locator* polynomial

$$\sigma(\theta) = 1 + \theta^2 + 9\theta^3.$$

We search for an r such that $\sigma(r) = 0$. As $|\mathbb{F}| = 11$ we can do this by trying every element. 1 does the trick. It follows that $(\theta - 1)$ is a zero and divides $\sigma(\theta)$.
Next we divide by $(\theta - 1)$ and receive the polynomial $\hat{\sigma}(\theta) = 9\theta^2 + 10\theta + 10$. This time $\hat{\sigma}(2) = 0$. After dividing $\hat{\sigma}$ by $(\theta + 9)$, which equals $(x - 2)$. So the factorisation in linear terms over $\mathbb{F}_{11}$ is

$$\sigma(\theta) = (\theta - 1)(\theta - 2)(\theta - 3).$$

Now that we can find the error locations since

$$\sigma(\theta) = (1 - X_1\theta)(1 - X_2\theta)(1 - X_3\theta)$$

Constantin Vlachos
constantinosvasiliosargyris.vlachos@uzh.ch

and $\sigma(\theta) = 0$ for $\theta = 1, 2, 3$. Hence $X_1 = 1, X_2 = 4, X_3 = 6$, the multiplicative inverses of the zeros of $\sigma(\theta)$. Finally the magnitudes m_1, m_2, m_3 can be calculated as follows:

$$m_1 = \frac{9 + 7 + 7}{(1 - 4)(1 - 6)} = \frac{1}{4} = 3,$$
$$m_2 = \frac{9 + 28 + 112}{(1 - 3)(1 - 18)} = \frac{5}{1} = 5,$$
$$m_3 = \frac{9 + 14 + 28}{(1 - 2)(1 - 8)} = \frac{7}{7} = 1.$$

4. An Introduction to Cryptography.

In this chapter we introduce the reader to the concept of *one way functions*, *randomness* and *digital signatures*. We also look at some examples of *cryptographic protocols*.

4.1. Public- and Private-Key Cryptography.

Let us recall the example made in the first chapter, where we had the scenario of two people needing to communicate over an insecure channel using a known code. We determined that every 0 must be read as a 1 and every 1 as a 0. In the following section we will give the reader a more mathematical way of viewing encryption and decryption.

4.1.1. Symmetric Ciphers (private-key cryptography).

Definition 4.1. We formally define encryption as

$$e : \mathcal{M} \times \mathcal{K} \to \mathcal{C}$$

where one maps (m, k) with m plaintext and k a key to the ciphertext c.

Similarly we define decryption as

$$d : \mathcal{C} \times \mathcal{K} \to \mathcal{M}$$

where one maps (c, k) with c ciphertext and k a key to the plaintext m.
Note that for some encryption function e the decryption function is $d = e^{-1}$, the inverse of e.

Symmetric Ciphers (private-key cryptosystems)
Suppose Alice and Bob are using a symmetric cipher.
Alice wants to send Bob plaintext m. She first maps it to a ciphertext c, using an algorithm which is publicly known, and the private key. It is vital to the security of their private-key cryptosystem that only trusted people know the private key.
Bob receives the ciphertext and decrypts it, using the inverse function of Alice's encryption function.
Written as in 4.1:

$$e_k(m) = c$$
$$d_k(c) = m$$

Remark 4.2.
Notice that d is the inverse function of e.

Example 4.3.
We now rewrite the example discussed in the first part of this chapter using the new notation. First e must be defined. We set e to the function that adds k to every digit of m. Note that we are over the field $\mathbb{F}_2$. By taking $k = 1$, we already have the desired encryption.
For this example let $m = 01$. It follows that

$$e_k(01) = 10$$

since $0 + 1 \equiv 1 \pmod 2$ and $1 + 1 \equiv 0 \pmod 2$. To decrypt we use the inverse function of e, d, that subtracts k (the same as above) from every digit of m. It follows that

$$d_k(10) = 01.$$

This type of cipher is useful if its users have had a chance to exchange keys in secret. If this is not the case a different solution must be found for secure communication. This will be discussed in the following chapter.

4.1.2. Asymmetric Ciphers (public-key cryptography).
Assume Alice and Bob would like to communicate securely but they've never had a chance to exchange a private-key. Their only means of communication is over an insecure channel. Their solution is public-key cryptography, also known as asymmetric cryptography.
The main difference between asymmetric and symmetric cryptography is the amount of keys. In asymmetric cryptography one uses a public and a private key. The public one is for encryption and the private one for decryption, allowing anyone to send Bob an encrypted message but just Bob to decrypt it.
Written as in Notation 4.1:

$$e_{k_{pub}}(m) = c$$
$$d_{k_{priv}}(c) = m$$

The concept of asymmetric cryptography is similar to that of a mailbox: anyone wishing to contact Alice can send a letter which will be delivered to her mailbox, to which only Alice possesses the key. In this analogy the address would be the public key of an asymmetric cipher and the key to the mailbox the private one.

Remark 4.4.
One may observe that in order for this to work it must be infeasible for an attacker to find the inverse function to $e_{k_{pub}}$. In cryptography $d_{k_{priv}}$ is commonly also referred to as a trapdoor function. k_{priv} contains additional information allowing the holder to invert $e_{k_{pub}}$ in an acceptable amount of time.

In the next section the reader will be introduced to the *discrete logarithm problem*, also known as the DLP, and learn more about the special functions mentioned in Remark 4.4.

4.2. A One-Way Function.
The concept behind most cryptographic protocols is as follows.
One would like a function that is easily computable and hard to invert. Additionally a trapdoor, that is to say a means of computing the inverse if one has a certain piece of additional information, is important. Thus one can encrypt using a public key and be secure in knowing that the encrypted message cannot be decrypted by inversion of the (public) encryption function using the public key. In this chapter we focus on the *Discrete logarithm*, which is an essential part of the *Diffie-Hellman Key Exchange*. This will be further described in the next section.

4.2.1. The Discrete Logarithm Problem.

In the discrete logarithm problem we try to calculate the logarithm $\log_g(h)$ over a finite field. This is easier said than done.

Definition 4.5. [HPS06]
Let g be the primitive root of $\mathbb{F}_p$ and let h be a non-zero element of $\mathbb{F}_p$. The discrete logarithm problem (DLP) is the problem of finding an exponent x such that $g^x \equiv h \pmod{p}$.

In this section the hardness of the DLP will be examined.
Note that in computer science the hardness of a problem is defined by the amount of operations that are necessary to solve it, using the most efficient method currently known (for non quantum computers). A problem is deemed hard in computer science if it cannot be solved efficiently.
The first and weakest algorithm is the trivial bound algorithm.

Proposition 4.6.
Let $g \in G$ be a non-zero element of order N, then the discrete logarithm can be solved in $\mathcal{O}(N)$steps.

Proof.
Compute $g, g^2, g^3, ..., g^N$. If a solution exists, then h appears before one reaches g^N. Note that $g^i = g^{i-1} \cdot g$ thus one must only store one number and the number g. □

Remark 4.7.
$\mathcal{O}(N)$ looks linear however if one chooses p between 2^k and 2^{k+1} then the problem is stated in $\mathcal{O}(k)$ bits.
As we know a brute-force attack takes $\mathcal{O}(N)$ which is about $\mathcal{O}(p)$ (N=p-1 for p prime) and $\mathcal{O}(p) = \mathcal{O}(2^k)$ which is exponential.

We now present Shank's Algorithm, also know as the Babystep-Giantstep Algorithm

Proposition 4.8. *(Babystep-Giantstep Algorithm) [Dan71]*
Let G be a group and let $g \in G$ be an element of order $N \geq 2$.
Babystep-Giantstep solves $g^x \equiv h$ in $\sqrt{N} \log N$ steps.

(1) Let $n = 1 + \lceil \sqrt{N} \rceil$.
(2) Create two lists:
List 1: $e, g, g^2, ..., g^{n-1}$
List 2: $h, hg^{-n}, hg^{-2n}, ..., hg^{-(n-1)n}$
(3) Find a match such that $g^i = hg^{-jn}$
(4) $x = i + jn$ is a solution for $g^x = h$.

Proof.
Creating both lists requires 2n multiplications. These can be neglected.
Finding a match with an efficient searching and sorting algorithm has a time complexity of $\mathcal{O}(n \log n)$. Since $n \approx \sqrt{N}$ one can say the time complexity of the Babystep-Giantstep

Constantin Vlachos
constantinosvasiliosargyris.vlachos@uzh.ch

algorithm is $\mathcal{O}(\sqrt{N}\log\sqrt{N})$.
Now the proof that a match will always be found in both lists if an x exists such that $g^x = h$:
Write $x = i + jn$ where $0 \leq i < n$
It is clear that $1 \leq x < N \text{implies} x - i < N$
From

$$x = i + jn \implies j = \frac{x-i}{n} < \frac{N}{n} < n,$$

one sees that $q < n$ meaning one can always find a match in the two lists.

□

We now present the Pohlig-Hellman Algorithm. In order to do this we first state and prove the Chinese Remainder theorem.

THEOREM 4.9. *(Chinese Remainder theorem) [HPS06]*
Let $m_1, m_2, ..., m_k$ be a collection of pairwise relative prime integers, that is $\gcd(m_i, m_j) = 1$ *for $i \neq j$.*
Let $a_1, ..., a_k$ be arbitrary integers. Then the system of simultaneous congruences

$$x \equiv a_1 \pmod{m_1}, ..., x \equiv a_k \pmod{m_k}$$

has a solution $x = c$.
The solution c is unique up to equivalence modulo $m_1 \cdot m_2 \cdot ... \cdot m_k$.

EXAMPLE 4.10.
$x \equiv 1 \pmod 5$ and $x \equiv 9 \pmod{11}$. Does there exist such an x, if so what value does x have?

$$x \equiv 1 + 5y$$

with y an arbitrary integer.

$$x \equiv 9 \pmod{11} \implies 5y \equiv 8 \pmod{11}$$
$$\implies y \equiv 8 \cdot 9^{-1} \equiv 8 \cdot 9 \equiv 6 \pmod{11}$$
$$\implies x = 1 + 5 \cdot 6$$

Now to the proof of the theorem:

Proof.
Suppose one has the solution $x = c_i$ for the first i congurences. One can find further solutions of the form $x = c_i + m_1 \cdot m_2 \cdot ... \cdot m_i \cdot y$.
One chooses y such that x also satisfies $x \equiv a_{i+1} \pmod{m_{i+1}}$ so

$$c_i + m_1 \cdot m_2 \cdot ... \cdot m_i \cdot y \equiv a_{i+1} \pmod{m_{i+1}}.$$

□

Constantin Vlachos
constantinosvasiliosargyris.vlachos@uzh.ch

The idea of Pohlig and Hellman was as follows: when one solves the DLP for x, x is in modulo $p-1$. What if one could break the factorisation of $p-1$ down, such that one can solve simultaneous congruences to find x?

THEOREM 4.11. *(Pohlig Hellman Algorithm) [HPS06]*
Let G be a group. Suppose one can solve the DLP for any $g \in G$ whose order is a prime power. If some $g \in G$ have order q^e, where q is prime, we may assume the DLP for g can be solved in $\mathcal{O}(S_{q^e})$ steps.
Let $g \in G$ be of order N and let $N = q_1^{e_1} \cdot ... \cdot q_t^{e_t}$, where the q_i's are distinct primes. Then the DLP for g can be solved in $\mathcal{O}\left(\sum_{i=1}^{t} S_{q_i^{e_i}} + \log(N)\right)$ steps.

Procedure:

(1) For each $1 \leq i \leq t$, let

$$g_i = g^{\frac{N}{q_i^{e_i}}} \quad \text{and} \quad h_i = h^{\frac{N}{q_i^{e_i}}}.$$

One can observe that g_i has order $q_i^{e_i}$.

(2) Solve $g_i^{x_i} = h_i$ with an efficient algorithm such as the Babystep-Giantstep Algorithm.

(3) Use Chinese remainder theorem to solve the simultaneous congruences for $x \equiv x_1 \pmod{q_1^{e_1}}, ..., x \equiv x_n \pmod{q_t^{e_t}}$.

Proof.
It holds that:
(2) takes $\mathcal{O}\left(\sum_{i=1}^{t} S_{q_i^{e_i}}\right)$ steps,
(3) takes $\mathcal{O}\left(\log(N)\right)$ steps.
All that is left to be shown is that one always gets a valid solution x for $g^x = h$.
Let x be the solution of the system of congruences.
For each i we can write $x = x_i + q_i^{e_i} \cdot z_i$ for some z_i and compute,

$$\begin{aligned}
(g^x)^{\frac{N}{q_i^{e_i}}} &= \left(g^{x_i + q_i^{e_i} \cdot z_i}\right)^{\frac{N}{q_i^{e_i}}} && (8)\\
&= \left(g^{\frac{N}{q_i^{e_i}}}\right)^{x_i} \cdot g^{N z_i} && (9)\\
&= \left(g^{\frac{N}{q_i^{e_i}}}\right)^{x_i} && (10)\\
&= g_i^{x_i} && (11)\\
&= h_i && (12)\\
&= h^{\frac{N}{q_i^{e_i}}}. && (13)
\end{aligned}$$

Constantin Vlachos
constantinosvasiliosargyris.vlachos@uzh.ch

It follows that

$$\frac{N}{q_i^{e_i}} \cdot x \equiv \frac{N}{q_i^{e_i}} \cdot \log_g(h) \pmod{N}.$$

The gcd of $\frac{N}{q_1^{e_1}}, ..., \frac{N}{q_n^{e_n}}$ is one, so by Theorem 2.2 one can find integers $c_1, ..., c_t$ such that

$$\frac{N}{q_1^{e_1}} \cdot c_1 + ... + \frac{N}{q_t^{e_t}} \cdot c_t = 1.$$

Hence

$$\sum_{i=1}^{t} \frac{N}{q_i^{e_i}} \cdot c_i \cdot x \equiv \sum_{i=1}^{t} \frac{N}{q_i^{e_i}} \cdot c_i \cdot \log_g(h) \pmod{N},$$

giving us

$$x \equiv \log_g(h) \pmod{N}.$$

Thus one can conclude that it is wise to choose prime p as $2 \cdot q + 1$, where q is a large prime. □

Constantin Vlachos
constantinosvasiliosargyris.vlachos@uzh.ch

4.3. RSA and Diffie-Hellman Protocols.

In this section we present the RSA and Diffie-Hellman protocols and show how one can make use of hard problems.

4.3.1. Diffie-Hellman Key Exchange. [HPS06]

Cryptography's two most prominent stars Alice and Bob would like to exchange a secret key over an insecure channel using the Diffie-Hellman key exchange.
The procedure:

(1) They agree upon a large prime p and a non-zero integer g, whose order is a large prime. These can be known to an adversary.
(2) Alice and Bob both pick secret integers a and b and compute:
$$A \equiv g^a \pmod{p} \qquad B \equiv g^b \pmod{p}$$
(3) They exchange the values A and B publicly.
(4) Next they compute:
$$A' \equiv B^a \pmod{p}, \qquad B' \equiv A^b \pmod{p}.$$
(5) The values they compute are the same since
$$A' \equiv B^a \equiv (g^b)^a \equiv (g^a)^b \equiv A^b \equiv B' \pmod{p}.$$
(6) This common value is the exchanged key.

Definition 4.12. *(DHP)*
Let p be a prime number and g an integer. The Diffie-Hellman problem (DHP) is the problem of computing g^{ab} from the public values g^a and g^b.

If one can solve the DLP one can also solve the DHP. The other direction is not known.

4.3.2. RSA.

The RSA-Protocol is a further example of asymmetric cryptography. It relies on the problem of factoring a large number N=pq into primes p and q, or obtaining the value of $(p-1)(q-1)$ from N (also knows as the totient of N). This paper skips the exact examination of the *factoring problem* and instead only focuses on the RSA protocol. Before being able to present the actual RSA protocol however we must first introduce a theorem and a lemma.

Theorem 4.13. *(Euler's theorem) [HPS06]*
Let p and q be distinct primes and let
$$g = \gcd(p-1, q-1).$$
Then $a^{(p-1)(q-1)/g} \equiv 1 \pmod{pq}$ *for all a with* $\gcd(a, pq) = 1$.

Proof. It holds that
$$a^{(p-1)(q-1)/g} = (a^{(p-1)})^{(q-1)/g}.$$

Constantin Vlachos
constantinosvasiliosargyris.vlachos@uzh.ch

In modulo p:

$$(a^{(p-1)})^{(q-1)/g} \equiv 1^{(q-1)/g} \pmod{p}. \tag{14}$$

Further,

$$a^{(p-1)(q-1)/g} = (a^{(q-1)})^{(p-1)/g},$$

so modulo q we get

$$(a^{(q-1)})^{(p-1)/g} \equiv 1^{(q-1)/g} \pmod{q}. \tag{15}$$

One can observe that $p-1$ or $q-1$ is always divisible by g by definition of the gcd and since $a^{(p-1)(q-1)/g} - 1$ is both divisible by p as by q, it follows that $a^{(p-1)(q-1)/g} - 1$ is divisible by pq. □

Lemma 4.14.
Let p and q be distinct primes and let $e \geq 1$ be an integer satisfying $\gcd(e,(p-1)(q-1)) = 1$, *this means that by Theorem 2.2 it follows that e has a multiplicative inverse, d modulo $(p-1)(q-1)$. Also assume that* $\gcd(c,pq) = 1$. *Then the congruence $x^e \equiv c \pmod{pq}$ has a unique solution $x \equiv c^d \pmod{pq}$.*

Proof.
It holds that: If $c \equiv 0$ then $x \equiv 0$.
As $de \equiv 1 \pmod{(p-1)(q-1)}$ we know that $de = 1 + k(p-1)(q-1)$ for some integer k.
We now show that $x \equiv c^d \pmod{pq}$ is a valid solution.

$$\begin{aligned}
(c^d)^e &\equiv c^{de} \pmod{pq} && (16)\\
&\equiv c^{1+k(p-1(q-1)} \pmod{pq} && (17)\\
&\equiv c \cdot (c^{(p-1)(q-1)})^k \pmod{pq} && (18)\\
&\equiv c \cdot 1^k \pmod{pq} && (19)\\
&\equiv c \pmod{pq} && (20)
\end{aligned}$$

It follows that $(c^d)^e \equiv c \pmod{pq}$, thus $x \equiv c^d \pmod{pq}$ is a valid solution.
Next we will show the uniqueness of the solution presented above. Assume $u = x$ is a solution for $x^e \equiv c \pmod{pq}$ and recall that $de = 1 + k(p-1)(q-1)$. Also note that since c is invertible modulo pq we also know that u is invertible.

$$\begin{aligned}
u \equiv u^{de-k(p-1)(q-1)} \pmod{pq} && (21)\\
\equiv (u^e)^d \cdot (u^{(p-1)(q-1)})^{-k} \pmod{pq} && (22)\\
\equiv (u^e)^d \cdot 1^{-k} \pmod{pq} && (23)\\
\equiv c^d \pmod{pq} && (24)
\end{aligned}$$

The last step is possible because one assumes $u^e \equiv c \pmod{pq}$.
It follows that every solution is of the form c^d.

Constantin Vlachos
constantinosvasiliosargyris.vlachos@uzh.ch

□

The RSA cryptosystem is based on the hardness of finding $(p-1)(q-1)$ from a number N=pq with p and q large primes without knowing the values p and q, merely N.
It hasn't been broken yet since there is no (non-quantum) polynomial-time algorithm that

(1) factors N into pq,
(2) finds $(p-1)(q-1)$ only knowing the value of $N = pq$.

Now back to Alice and Bob.
Suppose Alice and Bob would like to communicate using the RSA cryptosystem.
This is their setup: let p and q be large primes and let N be their product pq. Let e, m and c be integers. Let $\gcd(e, (p-1)(q-1)) = 1$.
This is their procedure:

(1) Bob's public key is (N,e) and his secret key is (p,q).
(2) Alice encrypts plaintext m by computing $c \equiv m^e \pmod{N}$
(3) Bob solves $x^e \equiv$ to recover m, which can be done because Bob knows p and q. (Lemma 4.14).

Remark 4.15.
An adversary only needs $(p-1)(q-1)$ to efficiently solve $x^e \equiv c \pmod{N}$ (not the full secret key (p, q)).
If the adversary knows the sum $p+q$ they can find $(p-1)(q-1)$. Indeed, we have that

$$(p-1)(q-1) = pq - p - q + 1 = N - (p+q) + 1.$$

4.4. Pseudo-random Functions and Generators.

In cryptography one often uses "random" numbers, selected by algorithms. But how could an algorithm possibly generate random numbers, wouldn't it just output the numbers it was programmed to?
Due to the need of random numbers in cryptosystems such as RSA and NTRUE, pseudo-random generators (PRGs) and pseudo-random functions (PRFs) have been designed. They take an input and give back a seemingly unrelated and random output. In reality however the output is not random at all.

4.4.1. Pseudo-random Generators.

Idea: The goal is a deterministic algorithm where one can input a seed with significantly smaller bit-length than the output. Given the output it should be hard to tell if it was randomly chosen or it was generated by inputting a seed in said algorithm.
Construction: (Blum Blum Shub) [[Bon11]]
Suppose a non-zero seed $x_0 \neq 1$ and the two large prime numbers p and q are given and $N = pq$. Let x_0 be coprime to p and q and define the sequence (x_n) by:

$$x_{n+1} = x_n^2 \pmod{N}.$$

The security of the PRG is based on the hardness of the quadratic residue problem (QRP), which states that given an integer a and $N = pq$ where p and q are large primes, it is hard to decide whether a is a quadratic residue or not.
Here an example to illustrate the QRP:
Let $N = 10$. We can observe that $2, 3, 7, 8$ are not quadratic residues modulo 10:

- $0^2 \equiv 0 \pmod{10}$
- $1^2 \equiv 1 \pmod{10}$
- $2^2 \equiv 4 \pmod{10}$
- $3^2 \equiv 9 \pmod{10}$
- $4^2 \equiv 6 \pmod{10}$
- $5^2 \equiv 5 \pmod{10}$
- $6^2 \equiv 6 \pmod{10}$
- $7^2 \equiv 9 \pmod{10}$
- $8^2 \equiv 4 \pmod{10}$
- $9^2 \equiv 1 \pmod{10}$.

There is no efficient way known to see if 8 was really a quadratic residue without computing the squares of 2, 3, 4 and 5 modulo 10 (we only need squares 0-5 because the curve of x^2 is symmetrical).

Remark 4.16.
One must choose a modulo N that isn't prime. If N=p, an odd prime, then determining if $a^k \pmod{p}$ is a quadratic residue can be done efficiently.

LEMMA 4.17. *(Legendre Symbol)*
Let p be an odd prime, $a \in \mathbb{F}_p^\star$ and let $\gcd(a,p) = 1$. Then $a^{\frac{p-1}{2}} \equiv 1 \pmod{p}$ if and only if a is a quadratic residue modulo p.

Proof. We begin by regarding the group homomorphism

$$\phi : \mathbb{F}_p^\star \to \mathbb{F}_p^\star,$$

where

$$x \mapsto x^2.$$

As p is prime it follows that $\ker(\phi) = \{1, -1\}$. Thus we see that half of the element of $\mathbb{F}_p^\star$ are quadratic residues (that is exactly $\frac{p-1}{2}$ elements).
By Theorem 2.3 we know that for any element a of $\mathbb{F}_p^\star$, $a^{p-1} \equiv 1 \pmod{p}$ holds. This means any element in $\mathbb{F}_p^\star$ is a root of the following polynomial in $\mathbb{F}_p[T]$

$$T^{p-1} - 1.$$

Recalling that p is odd this polynomial can also be written as

$$T^{p-1} - 1 = (T^{\frac{p-1}{2}} + 1)(T^{\frac{p-1}{2}} - 1).$$

It follows that for $a \in \mathbb{F}_p^\star$ satisfies

$$a^{\frac{p-1}{2}} \equiv \pm 1 \pmod{p}.$$

Finally we finish the proof by showing that the roots of $(T^{\frac{p-1}{2}} - 1)$ are all the $\frac{p-1}{2}$ quadratic residues of $\mathbb{F}_p^\star$.
To do this consider an element $a \in \mathbb{F}_p^\star$ and b such that $b^2 = a$ and a is a quadratic residue. Then it follows that

$$a^{\frac{p-1}{2}} = b^{p-1} \equiv 1 \pmod{p}.$$

This concludes the proof as we have showed that quadratic residue is a root of $(T^{\frac{p-1}{2}} - 1)$ giving us that every quadratic residue raised to the power of $\frac{p-1}{2}$ equals one. We also found that the other $\frac{p-1}{2}$ elements are all not quadratic residues and that they equal -1 when raised to the power of $\frac{p-1}{2}$. □

4.4.2. Pseudo-random Functions.

The goal is a function where one can input a number and receive back a random looking bit-string. A way of simplifying this function is to think of it as linked to a PRG. If one inputs i it outputs the *i'th* generated string by the PRG.

Construction: (Goldreich, Goldwasser, Micali) [[GGM86]]
Let G be a PRG where $G : \{0,1\}^s \to \{0,1\}^{2s}$.
Define G^0 and G^1 to be the left and right halves of G, both of length s.
$G(x) = G^0(x) \parallel G^1(x)$.
Take any secret key $k \in \{0,1\}^s$ (k's binary representation is limited to max. length s).

Define $f_k : \{0,1\}^n \mapsto \{0,1\}^s$ by $f_k(x_1, ..., x_n) = G^{x_n}(G^{x_{n-1}}(...(G^{x_1}(k))))$
In simpler terms:

(1) $G(k) : \{0,1\}^s \mapsto \{0,1\}^{2s}$.
(2) $G(k) = G^0(k) \parallel G^1(k)$.
(3) For $x_1 = 0$ take $G^0(k)$ (left side).
For $x_1 = 1$ take $G^1(k)$ (right side).
(4) Loop back to 1) for x_i with $i = 1, ..., n$.

4.5. Digital Signatures.

Alice would like to show her approval of a document by putting a digital signature on it. Anyone receiving her document can check if the signature on the document is indeed hers. The encryption key of a digital signature function is private, since she doesn't want other people to be able to sign using her signature.
The decryption (or verification) key is public in order to let anyone validate a signature.

Before presenting a more formal description of digital signatures we give a few definitions and introduce hash functions.

4.5.1. Hash Functions.

Definition 4.18. *(Collision Resistance)*
A function f is called collision resistant if it is hard to find x and y, such that $f(x) = f(y)$ where $x \neq y$.

Definition 4.19. *(Pre-Image Resistance)*
A function f is pre-image resistant when given an x, it is hard to find a y such that $f(y) = x$.

Definition 4.20. *(Second Pre-Image Resistance)*
A function f is called second pre-image resistant if given an x, it is hard to find a $y \neq x$, such that $f(x) = f(y)$.

Definition 4.21. *(Hash Functions)* A hash function is an algorithm that maps a binary string of arbitrary length to a binary string of fixed length of n-bits. This function must also be:

- collision resistant
- pre-image resistant
- second pre-image resistant
- computable in almost linear time.

Remark 4.22.
Collision resistance implies second pre-image resistance. The latter however does not imply the former.

Construction:

Constantin Vlachos
constantinosvasiliosargyris.vlachos@uzh.ch

Suppose $h(x_1, ..., x_n, x_{n+1}, ...x_s) = (x_1, ..., x_n)$, for $x_1 \in \{0,1\}$ for $i = 1, ..., s$
The proposed function h fulfills the condition of mapping a string of arbitrary length to fixed length n.
However there is:

- no pre-image resistance: $h(x_1, ..., x_n, ...) = (x_1, ..., x_n)$ so we can easily find a pre-image for example $(x_1, ...x_n, y)$,where $y \in \{0, 1\}$.
- no collision resistance: $h(x_1, ..., x_n, x') = (x_1, ..., x_n) = h(x_1, ..., x_n, x^*)$ for $x' \neq x^*$.

Therefore h is not an appropriate hash function for cryptographic applications.

Now consider this example and the algorithm H [HPS06]. To compute H(D) with D a binary string of arbitrary length we do the following.

(1) First D is lengthened with 0-bits until its bit-length is divisible by n.
(2) Next we write D as the concatenation $D = D_1 \parallel D_2 \parallel ... \parallel D_k$ where D_i has length n for every $i = 1, .., k$.
(3) One starts with an initial bit-string H_0 of length n and computes $M(D_1)$, where M is a mixing algorithm and sets $H_1 = M(D_1) \oplus H_0$
(4) One repeats this until $H_k = M(D_k) \oplus H_{k-1}$.
(5) Hash(D) = H_k.

Remark 4.23.
One can observe that H does not fulfill the property of collision resistance. We illustrate this now.
Assuming we are given $H(D), H_0, M$ we can re-write the algorithm:

$$H(D) = H_k = H_0 \oplus M(D_1) \oplus ... \oplus M(D_k)$$

One can construct an algorithm that finds a collision by finding a second pre-image. The algorithm works as follows.

(1) Solve $H_0 \oplus X = H(D)$ for X, a bit-string of length n.
(2) $X = M(Y)$, where Y is our second pre-image. So to get Y one applies the inverse function of M.

4.5.2. Digital Signatures using Hash Functions.

We are given the following setup:

- K_{priv}: signing key
- K_{pub}: verification key
- Sign: signing algorithm with input D, k_{priv} and output D^{sig} for D.
- Verify: verification algorithm with input D, D^{sig}, k_{pub} and output True / False.

It is very important that:

(1) It is hard to find a $D^{sig'} \neq D^{sig}$ for which Verify gives the output True.
(2) Given k_{pub} one cannot find k_{priv}

Since signing big documents would take very long, one only signs the hash of a document instead of the entire document. The security is not weakened since hash functions are collision and pre-image resistant.

Constantin Vlachos
constantinosvasiliosargyris.vlachos@uzh.ch

4.5.3. RSA Digital Signature. [RSA78]
To illustrate the theory we give an example of the RSA-signature.
Setup:
Alice chooses two large primes p, q which she keeps secret.
She computes $pq = N$ and chooses a public verification exponent e, a non-zero integer with $\gcd(e, N) = 1$.
She solves the congruence $de \equiv 1 \pmod{(p-1)(q-1)}$ and takes d as her secret signing key.
To sign D (assuming D is an integer between $1 < D < N$) Alice computes: $S \equiv D^d \pmod{N}$.
To verify one can compute $S^e \pmod{N}$ and may assume the document was signed by Alice if $S^e \equiv D \pmod{N}$.

5. The ISBN Experiment.

The aim of this section is to find a parity-check matrix for the IBAN code that we assume is linear. We also assume that the dimension of the dual-code is non-zero.

5.1. Recapitulation of Linear Codes.

We begin this chapter with a short recapitulation of linear codes.

We know: Every linear code has a generator matrix G and a parity-check matrix H. The parity-check matrix of an (n, k, d)-code C is an $n - k \times n$ matrix H over the chosen field K such that the following holds:

$$c \in C \iff cH^t = 0,$$

it follows that

$$GH^t = 0.$$

This means that the rows of any parity-check matrix H span the kernel of a generator matrix G.
Hence to find H we find $Ker(G)$ by collecting code words of C and computing a basis for them. The basis elements are the rows of a generator matrix G. Next we calculate a basis of $Ker(G)$. Finally we take the basis vectors of $Ker(G)$ as the rows of H.

5.2. The ISBN Experiment. ISBN stands for International Standard Book Number. The code is used to identify books. Now a brief scenario to give the reader a feel for where coding theory is hidden in these book numbers.
Say you are on the phone with the book store and are ordering a book by reading out its ISBN number. Imagine the store employee gets one digit wrong. The ISBN code detects such an error and produces an error or no results when the store employee looks up the false number.

5.2.1. Our own Experiment.

We assume that the ISBN code is linear and we want to find a parity-check matrix for it. We employ the method described above. However since we don't have G and only know that the ISBN code is an $(10, k, d)$-code we have to guess k. We start by guessing low, taking one code word c_1 and calculating a basis of the kernel of the generator matrix. We know this is also a basis for the row span of H meaning that for all $u \in Ker(G)$

$$\langle c, u \rangle = 0.$$

The next step is choosing a particular basis vector of $span_r(H)$ and attempting to find a code word c such that $\langle c, u \rangle \neq 0$ over the field K. In our case we are over the field $\mathbb{F}_{11} = \mathbb{Z}/11\mathbb{Z}$. If we find such a c, let's call it c_2, then we conclude that $k > 1$ and that the supposed generator matrix was too small and make a new one. The rows of the new generator matrix, let's call it G_2, are c_1 and c_2. We carry on with this procedure until we cannot find any code word c such that $\langle c, u \rangle \neq 0$. We assume $k \leq 9$ otherwise the code would not correct or detect any errors.

Constantin Vlachos
constantinosvasiliosargyris.vlachos@uzh.ch

5.2.2. Conducting our Experiment.

We begin with assuming the H has rank 9. This is the highest possible guess since if rank(H)=10 the generator matrix would be the zero-matrix, which it is obviously not. In this experiment we take our samples from the back of books. So whenever a new code word is introduced it is taken from the back of a book.
We also note that throughout this experiment the website [Mat] was used to facilitate matrix calculations.

Assume Rank(H)=9

We want to clarify that every time a new code word is introduced this means the author has simply copied it from the back of a book.
Let our code word be called $c_1 = \begin{pmatrix}0 & 4 & 7 & 1 & 9 & 9 & 4 & 6 & 2 & 6\end{pmatrix}$.
As described above we take c_1 as the row of G_1 (in the case our supposed generator matrix has rank 1).

$$G_1 = \begin{pmatrix}0 & 4 & 7 & 1 & 9 & 9 & 4 & 6 & 2 & 6\end{pmatrix}$$

We calculate a basis of the kernel of G using matrixcalc.org and receive

$$B_{Ker(G_1)} = \left\{ \begin{pmatrix}1\\0\\0\\0\\0\\0\\0\\0\\0\\0\end{pmatrix}, \begin{pmatrix}0\\\frac{-7}{4}\\1\\0\\0\\0\\0\\0\\0\\0\end{pmatrix}, \begin{pmatrix}0\\\frac{-1}{4}\\0\\1\\0\\0\\0\\0\\0\\0\end{pmatrix}, \begin{pmatrix}0\\\frac{-9}{4}\\0\\0\\1\\0\\0\\0\\0\\0\end{pmatrix}, \begin{pmatrix}0\\\frac{-9}{4}\\0\\0\\0\\1\\0\\0\\0\\0\end{pmatrix} \begin{pmatrix}0\\-1\\0\\0\\0\\0\\1\\0\\0\\0\end{pmatrix} \begin{pmatrix}0\\\frac{-3}{2}\\0\\0\\0\\0\\0\\1\\0\\0\end{pmatrix} \begin{pmatrix}0\\\frac{-1}{2}\\0\\0\\0\\0\\0\\0\\1\\0\end{pmatrix} \begin{pmatrix}0\\\frac{-3}{2}\\0\\0\\0\\0\\0\\0\\0\\1\end{pmatrix} \right\}$$

Since we have established that $Span_r(H) = Ker(G)$ it follows that if and only if $G_1 = G$ then for all $c \in C$ $\langle c, u\rangle = 0 \pmod{11}$ holds, where u is a linear combination of the basis vectors above.
We choose u_2 to be the first basis vector $\begin{pmatrix}1 & 0 & \cdots & 0\end{pmatrix}$ and introduce a new code word $c_2 = \begin{pmatrix}1 & 5 & 8 & 4 & 8 & 8 & 0 & 1 & 8 & X\end{pmatrix}$. However it is clear that $\langle c_2, u_2\rangle \neq 0 \pmod{11}$. Hence $G_1 \neq G$ and the search for generator and parity-check matrices goes on. (Also notice that in the ISBN code over $\mathbb{F}_{11}$, we denote 10 as X. This lets every code word have 10 digits.)

Constantin Vlachos
constantinosvasiliosargyris.vlachos@uzh.ch

Assume Rank(H)=8
The code words

- $c_1 = \begin{pmatrix}0 & 4 & 7 & 1 & 9 & 9 & 4 & 6 & 2 & 6\end{pmatrix}$
- $c_2 = \begin{pmatrix}1 & 5 & 8 & 4 & 8 & 8 & 0 & 1 & 8 & X\end{pmatrix}$

lead to the generator matrix

$$G_2 = \begin{pmatrix}0 & 4 & 7 & 1 & 9 & 9 & 4 & 6 & 2 & 6\\ 1 & 5 & 8 & 4 & 8 & 8 & 0 & 1 & 8 & X\end{pmatrix}.$$

Thus, we set

$$B_{Ker(G_2)} = \left\{ \begin{pmatrix}\frac{3}{4}\\ \frac{-7}{4}\\ 1\\ 0\\ 0\\ 0\\ 0\\ 0\\ 0\\ 0\end{pmatrix}, \begin{pmatrix}\frac{-11}{4}\\ \frac{-1}{4}\\ 0\\ 1\\ 0\\ 0\\ 0\\ 0\\ 0\\ 0\end{pmatrix}, \begin{pmatrix}\frac{13}{4}\\ \frac{-9}{4}\\ 0\\ 0\\ 1\\ 0\\ 0\\ 0\\ 0\\ 0\end{pmatrix}, \begin{pmatrix}\frac{13}{4}\\ \frac{-9}{4}\\ 0\\ 0\\ 0\\ 1\\ 0\\ 0\\ 0\\ 0\end{pmatrix} \begin{pmatrix}5\\ -1\\ 0\\ 0\\ 0\\ 0\\ 1\\ 0\\ 0\\ 0\end{pmatrix} \begin{pmatrix}\frac{13}{2}\\ \frac{-3}{2}\\ 0\\ 0\\ 0\\ 0\\ 0\\ 1\\ 0\\ 0\end{pmatrix} \begin{pmatrix}\frac{-11}{2}\\ \frac{-1}{2}\\ 0\\ 0\\ 0\\ 0\\ 0\\ 0\\ 1\\ 0\end{pmatrix} \begin{pmatrix}\frac{-5}{2}\\ \frac{-3}{2}\\ 0\\ 0\\ 0\\ 0\\ 0\\ 0\\ 0\\ 1\end{pmatrix} \right\}.$$

We choose another u, this time u_3 and let $u_3 = \begin{pmatrix}5 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0\end{pmatrix}$. Next we introduce $c_3 = \begin{pmatrix}0 & 2 & 7 & 3 & 0 & 8 & 6 & 7 & 8 & 2\end{pmatrix}$. And since $\langle c_3, u_3\rangle \neq 0 \pmod{11}$ it follows that $G_2 \neq G$.
Assume Rank(H)=7
The code words

- $c_1 = \begin{pmatrix}0 & 4 & 7 & 1 & 9 & 9 & 4 & 6 & 2 & 6\end{pmatrix}$
- $c_2 = \begin{pmatrix}1 & 5 & 8 & 4 & 8 & 8 & 0 & 1 & 8 & X\end{pmatrix}$
- $c_3 = \begin{pmatrix}0 & 2 & 7 & 3 & 0 & 8 & 6 & 7 & 8 & 2\end{pmatrix}$

lead to the generator matrix

$$G_3 = \begin{pmatrix}0 & 4 & 7 & 1 & 9 & 9 & 4 & 6 & 2 & 6\\ 1 & 5 & 8 & 4 & 8 & 8 & 0 & 1 & 8 & X\\ 0 & 2 & 7 & 3 & 0 & 8 & 6 & 7 & 8 & 2\end{pmatrix}.$$

Constantin Vlachos
constantinosvasiliosargyris.vlachos@uzh.ch

Thus, we set

$$B_{Ker(G_3)} = \left\{ \begin{pmatrix} \frac{-23}{7} \\ 1 \\ \frac{-5}{7} \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \frac{59}{14} \\ \frac{-9}{2} \\ \frac{9}{7} \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \frac{5}{2} \\ \frac{-1}{2} \\ -1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} \frac{29}{7} \\ 1 \\ \frac{-8}{7} \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} \frac{79}{14} \\ \frac{1}{2} \\ \frac{-8}{7} \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} -7 \\ 3 \\ -2 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \begin{pmatrix} \frac{-16}{7} \\ -2 \\ \frac{2}{7} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \right\}.$$

Let $u_4 = \begin{pmatrix} \frac{5}{2} & \frac{-1}{2} & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$. We introduce $c_4 = \begin{pmatrix} 0 & 5 & 8 & 2 & 0 & 1 & 8 & 6 & 1 & 7 \end{pmatrix}$. After calculating $\langle c_4, u_4 \rangle = 8 - 8 + 1 = 1 \pmod{11}$ it follows that $G_3 \neq G$.

Assume Rank(H) = 6

The code words

- $c_1 = \begin{pmatrix} 0 & 4 & 7 & 1 & 9 & 9 & 4 & 6 & 2 & 6 \end{pmatrix}$
- $c_2 = \begin{pmatrix} 1 & 5 & 8 & 4 & 8 & 8 & 0 & 1 & 8 & X \end{pmatrix}$
- $c_3 = \begin{pmatrix} 0 & 2 & 7 & 3 & 0 & 8 & 6 & 7 & 8 & 2 \end{pmatrix}$
- $c_4 = \begin{pmatrix} 0 & 5 & 8 & 2 & 0 & 1 & 8 & 6 & 1 & 7 \end{pmatrix}$

lead to the generator matrix

$$G_4 = \begin{pmatrix} 0 & 4 & 7 & 1 & 9 & 9 & 4 & 6 & 2 & 6 \\ 1 & 5 & 8 & 4 & 8 & 8 & 0 & 1 & 8 & X \\ 0 & 2 & 7 & 3 & 0 & 8 & 6 & 7 & 8 & 2 \\ 0 & 5 & 8 & 2 & 0 & 1 & 8 & 6 & 1 & 7 \end{pmatrix}.$$

Thus, we set

$$B_{Ker(G_4)} = \left\{ \begin{pmatrix} -27 \\ 5 \\ \frac{-11}{2} \\ \frac{19}{2} \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \frac{-196}{6} \\ \frac{62}{9} \\ \frac{-113}{18} \\ \frac{133}{18} \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 14 \\ -2 \\ 1 \\ -3 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 4 \\ 1 \\ \frac{-3}{2} \\ \frac{1}{2} \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} -7 \\ 3 \\ -2 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} \frac{-37}{9} \\ \frac{-13}{9} \\ \frac{-1}{9} \\ \frac{5}{9} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \right\}.$$

Let $u_5 = \begin{pmatrix} -7 & 3 & -2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$. We introduce $c_5 = \begin{pmatrix} 0 & 8 & 2 & 8 & 4 & 0 & 2 & 6 & 8 & X \end{pmatrix}$.

$$\langle c_5, u_5 \rangle \equiv 4 \pmod{11}$$

Constantin Vlachos
constantinosvasiliosargyris.vlachos@uzh.ch

hence

$$G_4 \neq G.$$

Assume Rank(H)=5

The code words

- $c_1 = \begin{pmatrix} 0 & 4 & 7 & 1 & 9 & 9 & 4 & 6 & 2 & 6 \end{pmatrix}$
- $c_2 = \begin{pmatrix} 1 & 5 & 8 & 4 & 8 & 8 & 0 & 1 & 8 & X \end{pmatrix}$
- $c_3 = \begin{pmatrix} 0 & 2 & 7 & 3 & 0 & 8 & 6 & 7 & 8 & 2 \end{pmatrix}$
- $c_4 = \begin{pmatrix} 0 & 5 & 8 & 2 & 0 & 1 & 8 & 6 & 1 & 7 \end{pmatrix}$
- $c_5 = \begin{pmatrix} 0 & 8 & 2 & 8 & 4 & 0 & 2 & 6 & 8 & X \end{pmatrix}$

lead to the generator matrix

$$G_5 = \begin{pmatrix} 0 & 4 & 7 & 1 & 9 & 9 & 4 & 6 & 2 & 6 \\ 1 & 5 & 8 & 4 & 8 & 8 & 0 & 1 & 8 & X \\ 0 & 2 & 7 & 3 & 0 & 8 & 6 & 7 & 8 & 2 \\ 0 & 5 & 8 & 2 & 0 & 1 & 8 & 6 & 1 & 7 \\ 0 & 8 & 2 & 8 & 4 & 0 & 2 & 6 & 8 & X \end{pmatrix}.$$

Thus, we set

$$B_{Ker(G_5)} = \left\{ \begin{pmatrix} \frac{3341}{981} \\ \frac{2183}{981} \\ \frac{-1126}{981} \\ \frac{-1444}{981} \\ \frac{-305}{327} \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \frac{554}{109} \\ \frac{-38}{109} \\ \frac{-89}{109} \\ \frac{15}{109} \\ \frac{36}{109} \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \frac{841}{109} \\ \frac{34}{109} \\ \frac{-81}{109} \\ \frac{-88}{109} \\ \frac{-15}{109} \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \frac{-7}{109} \\ \frac{187}{109} \\ \frac{-64}{109} \\ \frac{-266}{109} \\ \frac{-28}{109} \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} \frac{-3385}{981} \\ \frac{-1537}{981} \\ \frac{23}{981} \\ \frac{317}{981} \\ \frac{-8}{327} \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \right\}.$$

We take $u_6 = \begin{pmatrix} 554 & -38 & -89 & 15 & 36 & 0 & 109 & 0 & 0 & 0 \end{pmatrix}$ and introduce $c_6 = \begin{pmatrix} 3 & 5 & 4 & 0 & 5 & 2 & 0 & 0 & 0 & 7 \end{pmatrix}$. To simplify the dot product calculations we give the reader $u_6 = \begin{pmatrix} 4 & 6 & X & 4 & 3 & 0 & X & 0 & 0 & 0 \end{pmatrix} \pmod{11}$.

$$\langle c_6, u_6 \rangle \equiv 9 \pmod{11}$$

hence

$$G_5 \neq G.$$

Assume Rank(H)=4

The code words

- $c_1 = \begin{pmatrix} 0 & 4 & 7 & 1 & 9 & 9 & 4 & 6 & 2 & 6 \end{pmatrix}$
- $c_2 = \begin{pmatrix} 1 & 5 & 8 & 4 & 8 & 8 & 0 & 1 & 8 & X \end{pmatrix}$
- $c_3 = \begin{pmatrix} 0 & 2 & 7 & 3 & 0 & 8 & 6 & 7 & 8 & 2 \end{pmatrix}$
- $c_4 = \begin{pmatrix} 0 & 5 & 8 & 2 & 0 & 1 & 8 & 6 & 1 & 7 \end{pmatrix}$

- $c_5 = \begin{pmatrix}0 & 8 & 2 & 8 & 4 & 0 & 2 & 6 & 8 & X\end{pmatrix}$
- $c_6 = \begin{pmatrix}3 & 5 & 4 & 0 & 5 & 2 & 0 & 0 & 0 & 7\end{pmatrix}$

lead to the generator matrix

$$G_6 = \begin{pmatrix}0 & 4 & 7 & 1 & 9 & 9 & 4 & 6 & 2 & 6\\ 1 & 5 & 8 & 4 & 8 & 8 & 0 & 1 & 8 & X\\ 0 & 2 & 7 & 3 & 0 & 8 & 6 & 7 & 8 & 2\\ 0 & 5 & 8 & 2 & 0 & 1 & 8 & 6 & 1 & 7\\ 0 & 8 & 2 & 8 & 4 & 0 & 2 & 6 & 8 & X\\ 3 & 5 & 4 & 0 & 5 & 2 & 0 & 0 & 0 & 7\end{pmatrix}.$$

Thus, we set

$$B_{Ker(G_6)} = \left\{ \begin{pmatrix} \frac{10174}{4607} \\ \frac{-10258}{4607} \\ \frac{701}{4607} \\ \frac{6357}{4607} \\ \frac{5148}{4607} \\ \frac{-3888}{4607} \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \frac{36323}{13821} \\ \frac{-41632}{13821} \\ \frac{13427}{13821} \\ \frac{19232}{4607} \\ \frac{5785}{4607} \\ \frac{-6882}{4607} \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \frac{-16765}{13821} \\ \frac{13337}{13821} \\ \frac{-2764}{13821} \\ \frac{-26866}{13821} \\ \frac{266}{460} \\ \frac{-1554}{4607} \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} \frac{-3408}{4607} \\ \frac{942}{4607} \\ \frac{-4101}{4607} \\ \frac{-3909}{4607} \\ \frac{-3533}{4607} \\ \frac{3667}{4607} \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \right\}.$$

We take $u_7 = \begin{pmatrix}\frac{-3408}{4607} & \frac{942}{4607} & \frac{-4101}{4607} & \frac{-3909}{4607} & \frac{-3533}{4607} & \frac{3667}{4607} & 0 & 0 & 0 & 1\end{pmatrix} \equiv \begin{pmatrix}X & 2 & X & 2 & 1 & 9 & 0 & 0 & 0 & 9\end{pmatrix} \pmod{11}$ and introduce $c_7 = \begin{pmatrix}3 & 5 & 4 & 0 & 1 & 3 & 6 & 1 & 6 & 9\end{pmatrix}$.

$$\langle c_7, u_7 \rangle = 189 \equiv 2 \pmod{11}.$$

hence

$$G_6 \neq G.$$

Assume Rank(H)=3
The code words

- $c_1 = \begin{pmatrix}0 & 4 & 7 & 1 & 9 & 9 & 4 & 6 & 2 & 6\end{pmatrix}$
- $c_2 = \begin{pmatrix}1 & 5 & 8 & 4 & 8 & 8 & 0 & 1 & 8 & X\end{pmatrix}$
- $c_3 = \begin{pmatrix}0 & 2 & 7 & 3 & 0 & 8 & 6 & 7 & 8 & 2\end{pmatrix}$
- $c_4 = \begin{pmatrix}0 & 5 & 8 & 2 & 0 & 1 & 8 & 6 & 1 & 7\end{pmatrix}$
- $c_5 = \begin{pmatrix}0 & 8 & 2 & 8 & 4 & 0 & 2 & 6 & 8 & X\end{pmatrix}$
- $c_6 = \begin{pmatrix}3 & 5 & 4 & 0 & 5 & 2 & 0 & 0 & 0 & 7\end{pmatrix}$
- $c_7 = \begin{pmatrix}3 & 5 & 4 & 0 & 1 & 3 & 6 & 1 & 6 & 9\end{pmatrix}$

Constantin Vlachos
constantinosvasiliosargyris.vlachos@uzh.ch

lead to the generator matrix

$$G_7 = \begin{pmatrix} 0 & 4 & 7 & 1 & 9 & 9 & 4 & 6 & 2 & 6 \\ 1 & 5 & 8 & 4 & 8 & 8 & 0 & 1 & 8 & X \\ 0 & 2 & 7 & 3 & 0 & 8 & 6 & 7 & 8 & 2 \\ 0 & 5 & 8 & 2 & 0 & 1 & 8 & 6 & 1 & 7 \\ 0 & 8 & 2 & 8 & 4 & 0 & 2 & 6 & 8 & X \\ 3 & 5 & 4 & 0 & 5 & 2 & 0 & 0 & 0 & 7 \\ 3 & 5 & 4 & 0 & 1 & 3 & 6 & 1 & 6 & 9 \end{pmatrix}.$$

Thus, we set

$$B_{Ker(G_7)} = \left\{ \begin{pmatrix} \frac{32218}{1581} \\ \frac{-11019}{527} \\ \frac{2313}{1054} \\ \frac{39469}{3162} \\ \frac{5395}{527} \\ \frac{-4362}{527} \\ \frac{1495}{186} \\ 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \frac{-29549}{1581} \\ \frac{9795}{527} \\ \frac{-740}{527} \\ \frac{-20338}{1581} \\ \frac{-4630}{527} \\ \frac{3342}{527} \\ \frac{-736}{93} \\ 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} \frac{-30997}{1581} \\ \frac{30397}{1581} \\ \frac{-6925}{3162} \\ \frac{-13319}{1054} \\ \frac{-5435}{527} \\ \frac{4219}{527} \\ \frac{-1589}{186} \\ 0 \\ 0 \\ 1 \end{pmatrix} \right\}.$$

We choose the transpose of the third vector to be u_8 and taking modulo 11 to account $u_8 = \begin{pmatrix} 7 & 6 & 1 & X & 1 & 5 & 5 & 0 & 0 & 1 \end{pmatrix}$.
Next we introduce $c_8 = \begin{pmatrix} 0 & 4 & 1 & 2 & 0 & 7 & 3 & 3 & 1 & 5 \end{pmatrix}$.

$$\langle c_8, u_8 \rangle = 100 \equiv 1 \pmod{11}$$

hence

$$G_7 \neq G.$$

Assume Rank(H)=2
The code words

- $c_1 = \begin{pmatrix} 0 & 4 & 7 & 1 & 9 & 9 & 4 & 6 & 2 & 6 \end{pmatrix}$
- $c_2 = \begin{pmatrix} 1 & 5 & 8 & 4 & 8 & 8 & 0 & 1 & 8 & X \end{pmatrix}$
- $c_3 = \begin{pmatrix} 0 & 2 & 7 & 3 & 0 & 8 & 6 & 7 & 8 & 2 \end{pmatrix}$
- $c_4 = \begin{pmatrix} 0 & 5 & 8 & 2 & 0 & 1 & 8 & 6 & 1 & 7 \end{pmatrix}$
- $c_5 = \begin{pmatrix} 0 & 8 & 2 & 8 & 4 & 0 & 2 & 6 & 8 & X \end{pmatrix}$
- $c_6 = \begin{pmatrix} 3 & 5 & 4 & 0 & 5 & 2 & 0 & 0 & 0 & 7 \end{pmatrix}$
- $c_7 = \begin{pmatrix} 3 & 5 & 4 & 0 & 1 & 3 & 6 & 1 & 6 & 9 \end{pmatrix}$
- $c_8 = \begin{pmatrix} 0 & 4 & 1 & 2 & 0 & 7 & 3 & 3 & 1 & 5 \end{pmatrix}$

Constantin Vlachos
constantinosvasiliosargyris.vlachos@uzh.ch

lead to the generator matrix

$$G_8 = \begin{pmatrix} 0 & 4 & 7 & 1 & 9 & 9 & 4 & 6 & 2 & 6 \\ 1 & 5 & 8 & 4 & 8 & 8 & 0 & 1 & 8 & X \\ 0 & 2 & 7 & 3 & 0 & 8 & 6 & 7 & 8 & 2 \\ 0 & 5 & 8 & 2 & 0 & 1 & 8 & 6 & 1 & 7 \\ 0 & 8 & 2 & 8 & 4 & 0 & 2 & 6 & 8 & X \\ 3 & 5 & 4 & 0 & 5 & 2 & 0 & 0 & 0 & 7 \\ 3 & 5 & 4 & 0 & 1 & 3 & 6 & 1 & 6 & 9 \\ 0 & 4 & 1 & 2 & 0 & 7 & 3 & 3 & 1 & 5 \end{pmatrix}.$$

Thus, we set

$$B_{Ker(G_8)} = \left\{ \begin{pmatrix} \frac{-25749}{9859} \\ \frac{41289}{19718} \\ \frac{12887}{39436} \\ \frac{-119031}{39436} \\ \frac{-14015}{19718} \\ \frac{-1845}{9859} \\ \frac{-62077}{39436} \\ \frac{15553}{19718} \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} \frac{13836}{69013} \\ \frac{-453737}{414078} \\ \frac{-47329}{828156} \\ \frac{-417875}{828156} \\ \frac{-50125}{138026} \\ \frac{-2698}{69013} \\ \frac{-605309}{828156} \\ \frac{134153}{138026} \\ 0 \\ 1 \end{pmatrix} \right\}.$$

In modulo 11 this reads

$$B_{Ker(G_8)} = \left\{ \begin{pmatrix} 4 \\ 1 \\ 6 \\ 0 \\ 9 \\ 1 \\ 7 \\ 9 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 2 \\ 7 \\ 7 \\ 7 \\ X \\ 3 \\ 1 \\ 7 \\ 0 \\ 1 \end{pmatrix} \right\}.$$

We choose the transpose of the first vector to be u_9 and introduce $c_9 = 0412406500$.

$$\langle c_9, u_9 \rangle = 133 \equiv 1 \pmod{11}$$

hence

$$G_8 \neq G.$$

Assume Rank(H) = 1

The code words

- $c_1 = \begin{pmatrix} 0 & 4 & 7 & 1 & 9 & 9 & 4 & 6 & 2 & 6 \end{pmatrix}$
- $c_2 = \begin{pmatrix} 1 & 5 & 8 & 4 & 8 & 8 & 0 & 1 & 8 & X \end{pmatrix}$

Constantin Vlachos
constantinosvasiliosargyris.vlachos@uzh.ch

- $c_3 = \begin{pmatrix}0 & 2 & 7 & 3 & 0 & 8 & 6 & 7 & 8 & 2\end{pmatrix}$
- $c_4 = \begin{pmatrix}0 & 5 & 8 & 2 & 0 & 1 & 8 & 6 & 1 & 7\end{pmatrix}$
- $c_5 = \begin{pmatrix}0 & 8 & 2 & 8 & 4 & 0 & 2 & 6 & 8 & X\end{pmatrix}$
- $c_6 = \begin{pmatrix}3 & 5 & 4 & 0 & 5 & 2 & 0 & 0 & 0 & 7\end{pmatrix}$
- $c_7 = \begin{pmatrix}3 & 5 & 4 & 0 & 1 & 3 & 6 & 1 & 6 & 9\end{pmatrix}$
- $c_8 = \begin{pmatrix}0 & 4 & 1 & 2 & 0 & 7 & 3 & 3 & 1 & 5\end{pmatrix}$
- $c_9 = \begin{pmatrix}0 & 4 & 1 & 2 & 4 & 0 & 6 & 5 & 0 & 0\end{pmatrix}$
  lead to the generator matrix

$$G_9 = \begin{pmatrix}0 & 4 & 7 & 1 & 9 & 9 & 4 & 6 & 2 & 6\\ 1 & 5 & 8 & 4 & 8 & 8 & 0 & 1 & 8 & X\\ 0 & 2 & 7 & 3 & 0 & 8 & 6 & 7 & 8 & 2\\ 0 & 5 & 8 & 2 & 0 & 1 & 8 & 6 & 1 & 7\\ 0 & 8 & 2 & 8 & 4 & 0 & 2 & 6 & 8 & X\\ 3 & 5 & 4 & 0 & 5 & 2 & 0 & 0 & 0 & 7\\ 3 & 5 & 4 & 0 & 1 & 3 & 6 & 1 & 6 & 9\\ 0 & 4 & 1 & 2 & 0 & 7 & 3 & 3 & 1 & 5\\ 0 & 4 & 1 & 2 & 4 & 0 & 6 & 5 & 0 & 0\end{pmatrix}.$$

Thus, we set

$$B_{Ker(G_9)} = \left\{ \begin{pmatrix} \frac{4956618}{1567405} \\ \frac{-16299307}{4702215} \\ \frac{-2008273}{4702215} \\ \frac{13694644}{4702215} \\ \frac{138398}{313481} \\ \frac{54157}{313481} \\ \frac{4942513}{4702215} \\ \frac{123817}{1567405} \\ \frac{-1774413}{1567405} \\ 1 \end{pmatrix} \equiv \begin{pmatrix} X \\ 9 \\ 8 \\ 7 \\ 6 \\ 5 \\ 4 \\ 3 \\ 2 \\ 1 \end{pmatrix} \right\}.$$

It follows that

$$H = \begin{pmatrix}X & 9 & 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1\end{pmatrix},$$

as otherwise rank(H) would be zero which would contradict the error-detection property of the ISBN code.
One may recall that we can find an equivalent code by using Gauss-elimination. If we multiply the row by 10 we receive the equivalent parity-check matrix

$$\hat{H} = \begin{pmatrix}1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & X\end{pmatrix}.$$

This is the parity-check matrix of the ISBN code. Hence by applying Gauss-eliminations

$$G_9 = G.$$

This experiment took 15 ISBN code words to execute, 6 of which were in the Kernel of a G_i with $i < 9$. This experiment shows us that it is possible to reverse-engineer a linear code assuming one has access to a great amount of code words. Knowing the d of the linear (n, k, d)-code could also be helpful. We now demonstrate why.

PROPOSITION 5.1.
If d, the minimum distance of C, is known one can find H in one step.

Proof.
We know that the minimum distance of code C is d if and only if any $d-1$ columns of the parity-check matrix are linearly independent but d are linearly dependent.
From $d = 2$ one one can deduce that rank(H)=1. Next we make a matrix with at least 9 code words and perform Gauss-eliminations on it. One then adds code words until our matrix has rank 9. Any vector spanning the kernel then is the parity-check matrix of the ISBN code. □

We now demonstrate this using completely different ISBN code words.
We take the code words:

- $c_1 = 3540069607$
- $c_2 = 3540069038$
- $c_3 = 0387069038$
- $c_4 = 3540068600$
- $c_5 = 0306449641$
- $c_6 = 0471540250$
- $c_7 = 3540902260$
- $c_8 = 0387902368$
- $c_9 = 052170040X$

Next we denote with G the potential generator matrix consisting of row-vectors $c_1, ..., c_9$ and check its rank. If the rank equals 9 it is indeed a generator matrix of the ISBN code.

$$G = \begin{pmatrix} 3 & 5 & 4 & 0 & 0 & 6 & 9 & 6 & 0 & 7 \\ 3 & 5 & 4 & 0 & 0 & 6 & 9 & 0 & 3 & 8 \\ 0 & 3 & 8 & 7 & 0 & 6 & 9 & 0 & 3 & 8 \\ 3 & 5 & 4 & 0 & 0 & 6 & 8 & 6 & 0 & 0 \\ 0 & 3 & 0 & 6 & 4 & 4 & 9 & 6 & 4 & 1 \\ 0 & 4 & 7 & 1 & 5 & 4 & 0 & 2 & 5 & 0 \\ 3 & 5 & 4 & 0 & 9 & 0 & 2 & 2 & 6 & 0 \\ 0 & 3 & 8 & 7 & 9 & 0 & 2 & 3 & 6 & 8 \\ 0 & 5 & 2 & 1 & 7 & 0 & 0 & 4 & 0 & X \end{pmatrix}$$

One can see that $Rank(G) = 9$, hence it is a generator matrix of our code. Since the rank is full we know that the vector y spanning the kernel is simultaneously the desired parity

Constantin Vlachos
constantinosvasiliosargyris.vlachos@uzh.ch

check matrix, that is

$$y = \left\{ \begin{pmatrix} \frac{2351}{91} \\ \frac{-7454}{273} \\ \frac{-227}{39} \\ \frac{1801}{273} \\ \frac{6379}{273} \\ \frac{5651}{182} \\ -7 \\ -8 \\ \frac{-49}{3} \\ 1 \end{pmatrix} \equiv \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ X \end{pmatrix} \right\}$$

It follows that

$$H = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & X \end{pmatrix}.$$

After having reverse-engineered the ISBN code we give the reader some additional information about it.

- From the ISBN's parity-check matrix it follows that

  $c_1 + 2c_2 + 3c_3 + 4c_4 + 5c_5 + 6c_6 + 7c_7 + 8c_8 + 9c_9 + 10c_{10} \equiv 0 \pmod{11}$.
- The ISBN code is a single error detecting code. Indeed by Theorem 3.6, a code C can detect up to s errors in any code word provided $d(C) \geq s + 1$, where d(C) is the minimal distance and correct up to any t errors provided $d(C) \geq 2t + 1$.
- The ISBN code can detect a double permutation error. Indeed, if the above holds then for

  $$c_1 + 2c_2 + ... + ic_i + jc_j + ... + 10c_{10} \equiv 0 \pmod{11}$$

  any equation of the form

  $$c_1 + 2c_2 + ... + jc_i + ic_j + ... + 10c_{10} \not\equiv 0 \pmod{11}$$

  does not hold (assuming $c_i \not\equiv c_j$). This can be proven easily by assuming both are equivalent to zero and subtracting the second equation from the first. One is left with

  $$(ic_i + jc_j) - (jc_i + ic_j) \equiv 0 \pmod{11}.$$

  Making use of the distributive law we get

  $$(c_i - c_j)(i - j) \equiv 0 \pmod{11}.$$

  This is clearly a contradiction as $c_i - c_j \not\equiv 0$ and $i - j \not\equiv 0 \pmod{11}$.

Constantin Vlachos
constantinosvasiliosargyris.vlachos@uzh.ch

6. The IBAN Experiment.

In this chapter we present the IBAN code. To do this we make use of *p-adic numbers* during the breakdown of this code as they allow us to determine validity of an IBAN in a quick glace. The following chapter is divided into three subsections: in the first we take a look at the significance of IBAN numbers, in the second there is a extensive introduction to *p-adic numbers* and finally, in the third, the IBAN code is explained to the reader. We also note that the IBAN has similarities to one of the most used codes in the world, the *Cyclic-Redundancy-Check code*.

6.1. The significance if IBAN. [Wik23a]

IBAN stands for *International Bank Account Number*. IBANs were introduced as a means to ease the identification of international bank accounts and facilitate international transactions. Additionally the IBAN system reduces the risk of transcription errors.
IBANs vary depending on country:

- In Switzerland an IBAN number consists of 21 characters, the first two of which are the country code *CH*. The following 19 are digits resulting in a Swiss IBAN number being of the form

  $$CHcc\ bbbb\ baaa\ aaaa\ aaaa\ a,$$

  where the c's are check digits,the b's are the clearing number and the a's the account number. For example a Swiss account with the account number 5266 7129 0032 has an IBAN of the form CHcc bbbb b526 6712 9003 2). Note that the number *bbbbb* identifies the financial institute in which the account number *aaaaaaaaaaaa* lies.
- French IBANs are constructed differently. They consist of 27 characters, where the first two are the letters *FR*. The next 25 characters are numeric. A French IBAN is of the form

  $$FRcc\ bbbb\ bsss\ ssaa\ aaaa\ aaaa\ axx,$$

  where the c's are the check digits, the b's the clearing number, the s's from the *code guichinet*, the a's the account number and finally the x's the national check digits.

The fact that a "reduced risk of transcription errors" is mentioned leads us to believe that coding theory is involved in the way the IBANs were set up.

6.2. P-adic Numbers.

In Subsection 6.1 the construction of a Swiss IBAN was almost fully explained. The key part that was left out: the check digits. These vary from number to number and will be further addressed in Subsection 6.3. This subsection is for the reader unfamiliar with *p-adic numbers* as they will also be of importance in Subsection 6.3.

6.2.1. What is a P-adic Number?

Example 6.1.
41 can be written as a polynomial $a_0 + a_1x^1 + a_2x^2 + ...$ evaluated at $x = 5$ with all a_i chosen such that $0 \leq a_i < p$ for all $i \in I$,

$$41 = 1 \cdot 5^2 + 3 \cdot 5^1 + 1 \cdot 5^0.$$

Hence 41's 5-adic representation is 131.

The example above is easily understood as $41 = 1 + 3 \cdot 5 + 5^2$ and there isn't that much left to ponder about. -1 on the other hand is much harder, as this cannot be denoted as a finite sum like 41 can (see full example below).

In general we write a p-adic number s in the form $s = \sum_{i=k}^{\infty} a_i p^i$, where $0 \leq a_i < p$.

6.2.2. The Construction of the P-adic Numbers from the Rationals.

Before we get to the construction of the p-adic numbers from $\mathbb{Q}$ we must look at the p-adic valuation and p-adic absolute value on $\mathbb{Q}$ as these are used in the construction proof.

Definition 6.2. *(P-adic Valuation)* [POM]
Define the p-adic valuation on $\mathbb{Q}$ as the function $\nu_p : \mathbb{Q} \to \mathbb{Z} \cup \{\infty\}$, where $\nu_p(x)$ is:

- if $x \in \mathbb{Z}\backslash\{0\}$, then it is the integer such that $x = p^{\nu_p(x)}x'$, where $x' \in \mathbb{Z}$, $p \nmid x'$ in the case that $x \neq 0$,
- $\nu_p(a) - \nu_p(b)$ for any $x = \frac{a}{b} \in \mathbb{Q}$ (a, b are integers so do as above),
- $+\infty$ if $x = 0$.

Lemma 6.3. *(P-adic Valuation Properties) [POM]*
From Definition 6.2 the following properties follow for all $x, y \in \mathbb{Q}$:

(1) $\nu_p(xy) = \nu_p(x) + \nu_p(y)$ and
(2) $\nu_p(x + y) \geq min\{\nu_p(x), \nu_p(y)\}$.

Proof.

(1) We give no proof for part one. It follows from Definition 6.2 point one.
(2) Let $x, y \in \mathbb{Z}$.
If $x+y = 0$ then $\nu_p(x+y) = +\infty$, which must be larger or equal to $\min\{\nu_p(x), \nu_p(y)\}$.
In the case that $x + y \neq 0$ the proof goes as follows. WLOG one may set $\nu_p(x) \leq \nu_p(y)$. It follows that $x = p^{\nu_p(x)}m$ and $y = p^{\nu_p(y)}n$, where $p \nmid m, p \nmid n$. Thus

$$x + y = p^{\nu_p(x)}m + p^{\nu_p(y)}n.$$

As $\nu_p(x) \leq \nu_p(y)$ it holds true that $p^{\nu_p(x)} \mid p^{\nu_p(y)}$. Hence

$$p^{\nu_p(x)} \mid x + y$$

and

$$\nu_p(x) \leq \nu_p(x + y).$$

Constantin Vlachos
constantinosvasiliosargyris.vlachos@uzh.ch

Since $\nu_p(x)$ was chosen to be less or equal to $\nu_p(y)$, $\min\{\nu_p(x), \nu_p(y)\} = \nu_p(x)$. Thus the claim holds for all $x, y \in Z$.
To generalize this to all $x, y \in \mathbb{Q}$ we simply observe that any rational number can be denoted as the fraction of two integers i.e. for $x, y \in \mathbb{Q}, x = \frac{a}{b}, y = \frac{c}{d}$, where $a, b, c, d \in \mathbb{Z}$. Thus

$$\nu_p(x) = \nu_p\left(\frac{a}{b}\right) = \nu_p(a) - \nu_p(b).$$

Similarly one can obtain

$$\nu_p(y) = \nu_p\left(\frac{c}{d}\right) = \nu_p(c) - \nu_p(d).$$

WLOG we may set $\nu_p(x) \leq \nu_p(y)$. This implies that

$$\nu_p(a) - \nu_p(b) \leq \nu_p(c) - \nu_p(d).$$

In turn this means that

$$\nu_p(a) + \nu_p(d) \leq \nu_p(c) + \nu_p(b).$$

Recalling that $\nu_p(a) + \nu_p(d) = \nu_p(ad)$ we have that

$$\nu_p(ad) \leq \nu_p(bc).$$

Similarly to the proof for integers, we can deduce the first of these inequalities from the inequality above.

$$\begin{aligned}
\nu_p(ad) &\leq \nu_p(ad + bc) && (25)\\
\nu_p(a) + \nu_p(d) &\leq \nu_p(ad + bc) && (26)\\
\nu_p(a) &\leq \nu_p(ad + bc) - \nu_p(d) && (27)\\
\nu_p(a) - \nu_p(b) &\leq \nu_p(ad + bc) - \nu_p(d) - \nu_p(b) && (28)\\
& && (29)
\end{aligned}$$

Recalling that $x + y = \frac{ad+bc}{bd}$ and $x = \frac{a}{b}$ we can see by inequality (29) that our claim also holds for rationals x and y.

□

We can now define an absolute value on $\mathbb{Q}$.

Definition 6.4. *(P-adic Absolute Value)*
Using ν_p from 6.2, define the *p-adic absolute value* $|\cdot|_p : \mathbb{Q} \to \mathbb{R}_{\geq 0}$ by

$$|x|_p = \begin{cases} p^{-\nu_p(x)} & x \neq 0, \\ 0 & x = 0. \end{cases}$$

Next we check if Definition 6.4 truly gives us an absolute value, recalling that an absolute value $|\cdot|$ must fulfill four conditions:

Constantin Vlachos
constantinosvasiliosargyris.vlachos@uzh.ch

(1) $|\cdot| : \mathbb{K} \to \mathbb{R}_{\geq 0}$,
(2) $|x| = 0$ if and only if $x = 0$,
(3) $|xy| = |x||y|$ for $x, y \in \mathbb{K}$,
(4) $|x + y| \leq |x| + |y|$ for $x, y \in \mathbb{K}$.

THEOREM 6.5. *[POM]*
$|\cdot|_p$ *from Definition* 6.4 *is an absolute value.*

Note that instead of proving (4) we will prove a stronger property from which (4) follows.

Proof.
In this proof we show that the absolute value $||_p$ possesses all of the properties an absolute value must have.

(1) Let $x = 0$ then by definition it follows that $|x|_p = 0$.
Let $|x|_p = 0$ then it follows that $x = 0$ as $p^y \neq 0$ for all $y \in \mathbb{Z}$.
(2) $|x|_p|y|_p = p^{-\nu_p(x)} \cdot p^{-\nu_p(y)} = p^{-(\nu_p(x)+\nu_p(y))} = p^{-(\nu_p(xy))} = |xy|_p$, for $x, y \in \mathbb{Q}$.
(3) For $x + y = 0$: $|x + y|_p = 0 \leq \max\{|x|, |y|\}$.
For $x + y \neq 0$ we can set $|x|_p \leq |y|_p$ resulting in $\nu_p(x) \geq \nu_p(y)$.
We write $|x + y|_p = p^{-\nu_p(x+y)}$ and know that $\nu_p(x + y) \geq \min\{|x|, |y|\} = \nu_p(x)$.
It follows that $|x + y|_p \leq |x|_p = \max\{|x|_p, |y|_p\}$.

□

The Construction of the P-adic numbers from the Rationals
The construction is divided into four small theorems in which one creates a field $\mathbb{F}$ with an absolute value and a final proof in which one proves that $(\mathbb{F}, |\cdot|_p)$ is a metric space. Before beginning with the first proof we define Cauchy Sequences.

DEFINITION 6.6. *(Cauchy Sequences)* A sequence $(a_n)_{n\in\mathbb{N}}$ is Cauchy with respect to an absolute value $|\cdot|$ if for every $\epsilon \in \mathbb{R}_{>0}$, there is some $N_0 \in \mathbb{N}$ such that for all $m, n \geq N_0$

$$|a_n - a_m| < \epsilon.$$

Note that

- all Cauchy sequences have an upper bound, this means that for a Cauchy sequence $(f_n)_{n\in\mathbb{N}}$ and any integer $i \geq \hat{C}_0$, where $\hat{C}_0$ is positive integer, $f_i < \hat{C}_0$,
- all Cauchy sequences have a lower bound, this means that for a Cauchy sequence $(f_n)_{n\in\mathbb{N}}$ and any integer $i \geq C_0$, where C_0 is positive integer, $f_i > C_0$,
- a Cauchy sequence is a nullsequence if $\lim_{n\to\infty} |a_n| = 0$.

Also note that in the following theorems we heavily rely on the fact that the *p-adic absolute value* is an absolute value and possesses the properties proven above. There is no need to apply definition of $|\cdot|_p$ to prove the theorems as they simply require the properties of an absolute value.

THEOREM 6.7.
Consider the set $S = \{(a_n)_{n\in\mathbb{N}} | a_n \in \mathbb{Q}, (a_n)_{n\in\mathbb{N}}$ *is a Cauchy sequence with respect to* $|\cdot|_p\}$.

Constantin Vlachos
constantinosvasiliosargyris.vlachos@uzh.ch

Then S, endowed with pointwise addition and multiplication, forms a commutative ring R with identity.

We give our own proof.

Proof.
Let $\epsilon > 0$ and choose N_{0_a}, N_{0_b} such that $|a_n - a_m| < \frac{\epsilon}{2}$ for $m, n > N_{0_a}$, $|b_n - b_m| < \frac{\epsilon}{2}$ for $m, n > N_{0_b}$.

- Additive closure: Let $(a_n)_{n\in\mathbb{N}}, (b_n)_{n\in\mathbb{N}} \in R$ then
$$(a_n)_{n\in\mathbb{N}} + (b_n)_{n\in\mathbb{N}} = (a_n + b_n)_{n\in\mathbb{N}}.$$
We have that $(a_n + b_n)_{n\in\mathbb{N}} \in \mathbb{Q}$ for all n. $(a_n + b_n)_{n\in\mathbb{N}}$ is also Cauchy because for m, n chosen such that for $m, n > \max\{N_{0_a}, N_{0_b}\}$, it holds that
$$|a_n + b_n - (a_m + b_m)| = |a_n - a_m + b_n - b_m| \leq |a_n - a_m| + |b_n - b_m| < \frac{\epsilon}{2} + \frac{\epsilon}{2}.$$
- Additive neutral element: Let $(a_n)_{n\in\mathbb{N}} \in R$. $(a_n)_{n\in\mathbb{N}} + (0, 0, 0, ...) = (a_n)_{n\in\mathbb{N}}$ as we have pointwise addition from $\mathbb{Q}$. All that is left to show, is that $(0, 0, 0, ...)$ is Cauchy, which it is because the difference of any two entries equals zero and $\epsilon > 0$.
- Multiplicative closure: Let M be such that $M > |a_n|, |b_n|$ for all $n \in \mathbb{N}$ and pick N_{0_a}, N_{0_b} such that $|a_n - a_m| < \frac{\epsilon}{2M}$ for all $m, n > N_{0_a}$ and $|b_n - b_m| < \frac{\epsilon}{2M}$ for all $m, n > N_{0_b}$. Let $(a_n)_{n\in\mathbb{N}}, (b_n)_{n\in\mathbb{N}} \in R$, then $(a_n)_{n\in\mathbb{N}}(b_n)_{n\in\mathbb{N}} = (a_n b_n)_{n\in\mathbb{N}}$. It is clear that $ab \in \mathbb{Q}$.$(a_n b_n)_{n\in\mathbb{N}}$ is Cauchy because for m, n chosen such that for $m, n > \max\{N_{0_a}, N_{0_b}\}$
$$\begin{aligned} |a_n b_n - a_m b_m| =& |a_n b_n - a_m b_n + a_m b_n - a_m b_m| && (30)\\ \leq& |b_n||a_n - a_m| + |a_m||b_n - b_m| && (31)\\ <& M \cdot \frac{\epsilon}{M} = \epsilon && (32)\end{aligned}$$
and since $(a_n)_{n\in\mathbb{N}}$ and $(b_n)_{n\in\mathbb{N}}$ are Cauchy, thus bounded it follows that $(a_n b_n)_{n\in\mathbb{N}}$ is Cauchy.
- Multiplicative neutral element: Let $(a_n)_{n\in\mathbb{N}} \in R$ then $(a_n)_{n\in\mathbb{N}} \cdot (1, 1, 1, ...) = (a_n)_{n\in\mathbb{N}}$. $(1, 1, 1, ...)$ is Cauchy because the difference of any two entries equals zero and is smaller than any $\epsilon > 0$.
- The additive and multiplicative inverses, as well as commutativity, associativity and distributivity, all follow from pointwise addition and multiplication and that all entries lie in $\mathbb{Q}$.

□

Theorem 6.8.
The nullsequences of the ring R form an ideal I.

We give our own proof.

Proof.

- Let $(a_n)_{n\in\mathbb{N}}, (b_n)_{n\in\mathbb{N}} \in R$ be nullsequences. It follows that
$$\lim_{n\to\infty}(a_n + b_n) = \lim_{n\to\infty}(a_n) + \lim_{n\to\infty}(b_n) = 0.$$
Hence the sum of two nullsequences is also a nullsequence and in I.
- Let $(a_n)_{n\in\mathbb{N}} \in R$ with upper bound be a nullsequence and $(b_n)_{n\in\mathbb{N}} \in R$ have the upper bound M. Then
$$\lim_{n\to\infty}(a_n b_n) = \lim_{n\to\infty}(a_n) \cdot M = 0 \cdot M = 0.$$

□

Theorem 6.9.
R/I is a field.

We give our own proof.

Proof.
Take any non null-sequence $(a_n)_{n\in\mathbb{N}} \in R$. This means $a_n \notin I$.
We know that since $(a_n)_{n\in\mathbb{N}}$ is not a null-sequence there can only be a finite number of a_i's where $a_i = 0$ and that after some $Z_0 \in \mathbb{N}$ every entry is non-zero.
Hence we can construct a null-sequence $(b_n)_{n\in\mathbb{N}}$, where
$$\begin{cases} b_i = 1 \text{if} a_i = 0, \\ b_i = 0 \qquad \text{else.} \end{cases}$$
It follows that the sequence $(c_n)_{n\in\mathbb{N}} = (\frac{1}{a_n+b_n})$ is well-defined, that is to say there is no entry c_i where $a_i + b_i = 0$ giving us $\frac{1}{0}$.
To show the sequence $(c_n)_{n\in\mathbb{N}}$ is Cauchy we must prove that $\exists N \in \mathbb{N}$, such that for
$$m, n \geq N, \left|\frac{1}{a_n + b_n} - \frac{1}{a_m - b_m}\right| < \epsilon$$
for any $\epsilon \in \mathbb{R}_{>0}$. This holds because
$$\exists N_0 \in \mathbb{N} \text{ such that for any } m, n \geq N_0, \text{it holds that} |a_n - a_m| < \epsilon',$$
where $\epsilon' \in \mathbb{R}_{>0}$ and because $\exists Z_0$ as above. Finally one may recall that Cauchy sequences have an upper and lower bound, i.e. for any $n \geq C \in \mathbb{N}$ it holds that $A < a_n < B$, where $A, B \in \mathbb{N}$. We choose m and n to be greater than N_0, C_0 and Z_0.
$$\left|\frac{1}{a_n + b_n} + \frac{1}{a_m + b_m}\right| \leq \frac{|a_n - a_m| + |b_n - b_m|}{|a_n a_m + b_n a_m + b_m a_n + b_n b_m|} \leq \frac{|a_n - a_m|}{|a_n a_m|} < \epsilon' \cdot \frac{1}{A^2}.$$
We may take
$$\epsilon = \frac{\epsilon'}{A^2}$$
and conclude that $(c_n)_{n\in\mathbb{N}}$ is in fact a Cauchy sequence.
Now one may observe that $(a_n)_{n\in\mathbb{N}} \cdot (c_n)_{n\in\mathbb{N}} = (111...1...) + (\hat{b}_n)_{n\in\mathbb{N}}$, the multiplicative identity and a null-sequence. Since we chose $(a_n)_{n\in\mathbb{N}}$ as any non-zero Cauchy sequence it

Constantin Vlachos
constantinosvasiliosargyris.vlachos@uzh.ch

follows that one could similarly find an inverse to any other sequence in R/I. Hence every element in R/I is a unit making R/I a field. □

THEOREM 6.10.
One can extend the absolute value $|\cdot|_p$ to one on $\mathbb{F}$ as follows:
Let $a \in \mathbb{F}$, set $|a| := \lim_{n\to\infty} |a_n|$, where $a = (a_n)_{n\in\mathbb{N}} + I$.

Proof.

- $a = 0$, then $\lim_{n\to\infty}(a_n)_{n\in\mathbb{N}} = 0$. It follows that $|a| = 0$.
  $|a| = 0$ implies that $(a_n)_{n\in\mathbb{N}} \in I$, thus $a = 0$ over $\mathbb{F} = R/I$.
- For $x, y \in \mathbb{F}$ it holds that

$$\begin{aligned} |xy| &= \lim_{n\to\infty}(x_n y_n) && (33)\\ &= \lim_{n\to\infty}(x_n) \cdot \lim_{n\to\infty}(y_n) && (34)\\ &= (\lim_{n\to\infty}(x_n)) \cdot (\lim_{n\to\infty}(y_n) && (35)\\ &= |x||y| && (36) \end{aligned}$$

- For $x, y \in \mathbb{F}$:

$$\begin{aligned} |x+y| &= \lim_{n\to\infty}(x_n + y_n) && (37)\\ &= \lim_{n\to\infty}(x_n) + \lim_{n\to\infty}(y_n) && (38)\\ &= |x| + |y|. && (39) \end{aligned}$$

  The triangle equality follows.

□

We introduce two definitions of metric spaces.

DEFINITION 6.11. *(Metric Space)* [Wik23b]
A metric space is a pair (X, d) where X is a set and $d : X \times X \to \mathbb{R}_{\geq 0}$ is a metric, that is, for all $x, y, z \in \mathbb{F}$:

(1) $d(x, y) = 0$ if and only if x=y,
(2) $d(x, y) = d(y, x)$,
(3) $d(x, z) \leq d(x, y) + d(y, z)$.

DEFINITION 6.12. A metric space X, together with a distance function $d(X, d)$ is complete if every Cauchy sequence of points in X has a limit in X.

Using the definitions above we show that $(\mathbb{F}, d)$ is a metric space.

THEOREM 6.13. *[POM]*
$\mathbb{F}$, constructed in Theorem 6.9, together with d, defined such that $d(x, y) = |x - y|_p$, is a metric space.

Proof.

(1) If $d(x,y) = 0$ then $|x-y|_p = 0$ and this is only the case if $x - y = 0$.
(2) $d(x,y) = p^{-\nu_p(x-y)}$. We know that

$$\begin{aligned}\nu_p(x-y) =& \nu_p((-1)(y-x)) && (40)\\ =& \nu_p(-1) + \nu_p(y-x) && (41)\\ =& \nu_p(-1) + \nu_p(y-x) && (42)\\ =& \nu_p(y-x) && (43)\end{aligned}$$

as $\nu_p(-1) = 0$ since p is prime.
Hence

$$d(x,y) = p^{-\nu_p(x-y)} = p^{-\nu_p(y-x)} = d(y,x).$$

(3) It holds that

$$\begin{aligned}d(x,z) =& |x-z|_p && (44)\\ =& |x-y+y-z|_p && (45)\\ \leq& |x-y|_p + |y-z|_p && (46)\\ =& d(x,y) + d(y,z). && (47)\\ & && (48)\end{aligned}$$

□

One can proceed to prove that $(\mathbb{F}, d)$ is complete but we omit this.

Example 6.14. (Elaborate Example)
We finish the subsection on p-adic numbers with an example of our own, in which we wish to find the 5-adic representation of -1.
We begin with two observations:

(1) $\lim_{n\to\infty} |p^n|_p = 0$.
This follows because $\lim_{n\to\infty} |p^n|_p = \lim_{n\to\infty} \frac{1}{p^n} = 0$.
(2) We use that $(x-1)(x^{k-1} + ... + x^2 + x + 1) = (x^k - 1)$.
To find the 5-adic representation of -1 we use observation (1) and obtain

$$\lim_{j\to\infty} 5^j - 1 = -1.$$

Next, one applies observations (2) to get the equation

$$\lim_{j\to\infty} 4\sum_{i=0}^{j-1} 5^i = \lim_{j\to\infty} (5-1)\sum_{i=0}^{j-1} 5^i = \lim_{j\to\infty} (\sum_{i=1}^{j} 5^i - \sum_{i=1}^{j-1} 5^i = \lim_{j\to\infty} 5^j - 1 = -1.$$

Hence the 5-adic representation of -1 is

$$...444.$$

Constantin Vlachos
constantinosvasiliosargyris.vlachos@uzh.ch

6.3. The coding theory of IBAN.

In this subsection we build off the results and observations in the previous two subsections and show that the IBAN code is non-linear and single error correcting.

6.3.1. Why the IBAN code is not linear. [Wik23a]

In this experiment the goal was to reverse engineer the IBAN code. To do this we had to first gather information about said code. We found that given an IBAN number

$$CHcc\ bbbb\ baaa\ aaaa\ aaaa\ a,$$

one can

(1) permute the code such that after permutation it has the form $bbbb\ baaa\ aaaa\ aaaa\ aCHc\ c$,
(2) turn the alphanumeric code into a numeric one by decoding $a = 10, b = 11, c = 12, ..., z = 25$,
(3) and observe that the newly obtained integer over $\mathbb{Z}/97\mathbb{Z}$ sits in the equivalence class of 1.

Thus the IBAN code is non-linear, as the IBANs lie in a coset of $\mathbb{Z}/97\mathbb{Z}$, which is not a subvectorspace. This is clear because 0 is not in it. If we take a step back and consider the build of an IBAN it makes sense that the code is not linear. Below we illustrate why even if every IBAN were in the ideal $97\mathbb{Z}$ the IBAN code would not be linear.
Before we begin explaining why finding a generator matrix is impossible we note that a **PDIBAN** is an IBAN post permutation and decoding of letters.

- It is impossible to find a generator matrix because the way PDIBANs are built, digits 6,5,4 and 3 counting from right to left are the decoded letters. Hence if we add one Swiss IBAN to another and assume that the addition of the check digits does not carry a one into the third digit, the digits will be 2434. If we were to try and translate this back to a country code we would get **O,Y**. As no country that uses IBANs has the country code **OY**, we conclude that the sum of two Swiss PDIBANs is not a Swiss PDIBAN or even an PDIBAN at all and the code is not linear, hence one cannot find a generator matrix.
- The 23-digit number that lies in the ideal $97\mathbb{Z}$ and has a zero as its sixth digit is not an PDIBAN, as there is no letter that decodes to $0x$ for any positive integer x.

6.3.2. The IBAN code.

Now that we've argued why the choice of a linear code is not fitting, we go back to the algorithm provided to us by Wikipedia that is denoted above.
If any IBAN (post permutation and decoding of the letters) must be equivalent to one modulo 97 it means that any IBANs 97-adic representation begins with a one (recall p-adic numbers are infinite towards the left).
This allows us to set up a theorem, that claims that the IBAN is a single error detecting code.

THEOREM 6.15.
The IBAN code is a single error detecting code.

Proof.
Let $c = c_1c_2...c_{23}$ be a Swiss IBAN (post permutation and decoding of letters).
It follows that the 97-adic representation of

$$c = a_{12}97^{12} + a_{11}97^{11} + a_{10}97^{10} + ... + a_197 + 1$$

is as follows

$$a_{12}a_{11}a_{10}...a_11,$$

where each a_i for $i = 1, ..., 11$ is a number between zero and 96 and a_{12} is either zero or one (note that 97^{13} has more than 23 digits). If a single error of magnitude $y \in \{1, ..., 9\}$ is made in position x(counting right to left) this results in the addition of $\pm y10^{x-1}$ to c. Instead of doing this we look at the 97-adic representation of $\pm y10^{x-1}$ and realize that it never ends in a zero. Hence if we add it to the 97-adic representation of c, we know that the error will be detected.
We show that the code is not double error correcting with a counterexample.
$c = 300000011568199401217 97$ is an IBAN (post permutation and decoding of letters) and $e = 300000011568199401218 94$ the denotation of c, where two errors have occurred in positions 1 and 3 (counting from right). Notice, without even calculating the 97-adic representations of c and e that they both have a one as their leftmost digit. This is obvious for c as c is an IBAN and follows for e due to $100 - 3 = 97$. So no error is detected.
□

REMARK 6.16. We insert this remark during preparation for the presentation of this paper. It has become evident that the proof given above is incomplete. We simply have proven that every single error in the PDIBAN is detected. False transcription of the country code of the IBAN however causes a double error. To conclusively prove that this double error in the PDIBAN is found we must show that a double error caused by false transcription of a country code letter is detected.
We recall that a double error in positions $x, \hat{x}$ with magnitudes $y, \hat{y}$ would result in the addition of $\pm y10^{x-1}$ and $\pm \hat{y}10^{\hat{x}-1}$ to the PDIBAN.
Also recalling the way the country code letters decode to numbers (A=10, B=11,..., Z=35) we notice that, assuming $x > \hat{x}$, $x = \hat{x} + 1$. We also see that $y \in \{1, 2\}$ and $\hat{y} \in \{1, ..., 9\}$. As none of the possible sums of $\pm y10^{x-1}$ and $\pm \hat{y}10^{\hat{x}-1}$ lie in the ideal $97\mathbb{Z}$ we conclude that the p-adic representation of the sum of $\pm y10^{x-1}$ and $\pm \hat{y}10^{\hat{x}-1}$ has a non zero a_0

Constantin Vlachos
constantinosvasiliosargyris.vlachos@uzh.ch

coefficient and that the p-adic representation of the erroneous PDIBAN does not begin with a 1 (that is begin from the right). Thus the theorem holds.

7. Constructing a Hash-Function.

In the final chapter of this project we design a hash-function. The reason for the development of a new hash-function was the lack of second pre-image resistance of the second hash-function example in the Subsubsection 4.5.1.
First we recall the definition of a hash-function:

DEFINITION 7.1. *(Hash Functions)* A hash function is an algorithm that maps a binary string of arbitrary length to a binary string of fixed length of n-bits. This function must also be:

- collision resistant
- pre-image resistant,
- second pre-image resistant,
- computable in almost linear time.

Note that a function f being collision resistant if it is hard to find two inputs D_1, D_2 such that $f(D_1) = f(D_2)$. A function f is pre-image resistant if it is hard to find the input D from $f(D)$ alone for any input D. A function is second pre-image resistant if given $f(D_1)$ for any input D_1 it is hard to find an input D_2 such that $f(D_1) = f(D_2)$ (we see that if a function is not second pre-image resistant it is automatically not collision resistant).

In the hash-function we give below, focus was put on the properties of pre-image resistance and second pre-image resistance. Computational efficiency was completely neglected and although we hope the function is collision resistant we did not manage to prove it or find experimental evidence. One somewhat unorthodox specification of this hash-function is that one must also include a secret integer N, the product of two large primes. The last step we must take before getting to the hash-function itself is to consider the pseudo-random generator G, which we constructed. The algorithm G uses the secret integer N mentioned above and calculating $G(k)$ works as follows:

(1) Let $k = k_1...k_n$, where $k_i \in \{0, 1\}$ for all $i = 1, ..., n$. One takes the number whose binary representation is k to be $\hat{k}$.
(2) In order to compute $G(k)$ one first computes $y_1, ..., y_n$ as follows:
$$y_1 \equiv \hat{k}^2 \pmod{N},$$
$$y_{i+1} \equiv y_i^2 \pmod{N}.$$
(3) Next we write the y_i's in their binomial representation as bit strings, extend them by n zeros and compute $h(y_i)$ for every $i = 1, ..., n$, where h is the *"bad hash"* mentioned in Subsubsection 4.5.1 that simply chops off any bit after the $2n$'th bit.
(4) Finally one has $G(k) = r = h(y_1) \oplus ... \oplus h(y_n)$.

Our own Hash-Function

Consider the algorithm H, a document of arbitrary length D and a bit string k of length n that can be public. H runs as follows:

(1) D is extended by adding 0's to the end of it until it has length $s \cdot n$, where both $s, n \in \mathbb{N}$. Note that n will be the final length of $H(D)$ once computed and s is just some integer that varies depending on the length of D such that $s \cdot n$ is the nearest multiple of n to the length of D.

(2) Concatenate $D = d_1 d_2 ... d_{sn}$ such that $D = h_1 \parallel h_2 \parallel ... \parallel h_s$, where all h_i for $i = 1, ..., s$ have length n.

(3) Compute H_i for each h_i for $i = 1, ..., s$.
$H_i = f_k(x_1, ..., x_n) \oplus h_i$, where $h_i = x_1 x_2 ... x_n$.
One views $f_k(x_1, x_2, ..., x_n)$ as the pseudo-random function constructed by Goldreich, Goldwasser and Micali in 1986. We use the the PRG G, defined above and define $G^0(k)$ and $G^1(k)$ to be the left and right halves of the bit string generated by $G(k)$ for some k. Both $G^0(k)$ and $G^1(k)$ have length n.

$$G(k) = G^0(k) \parallel G^1(k).$$

Now define $f_k(x_1, ..., x_n)$ to be $G^{x_n}(...(G^{x_2}(G^{x_1}(k))))$.

(4) The final output $H(D)$ is then given to us by

$$H(D) = H_1 \oplus ... \oplus H_n.$$

We give a few examples of possible attacks on first pre-image and second pre-image. Again we did not manage to prove or find experimental evidence that the hash-function is in fact first pre-image and second pre-image resistant. In this paper we simply present our own hash-function, study some known attacks and argue why they are ineffective.

Example 7.2. (Second Pre-Image Attack 1)
Attack the second pre-image by inserting $D = 111...1$.
It follows that all h_i are equal, making all H_i equal. If the number of H_i's, so essentially, s is odd

$$H(D) = \underbrace{111...1}_{\text{n 1's}} \oplus f_k(x_1 x_2 ... x_n).$$

A problem of the second pre-image security of this hash-function would be choosing large primes p, q such that their product is greater than y_{n-1}^2. This would mean all the y_i's modulo N and without modulo would be identical. So N could be neglected and an attacker would know what $f_k(z_1, ..., z_n)$ is for $z_i \in \{0, 1\}$. Hence by trial and error one could find a $\hat{h} = z_1 ... z_n$ such that $f_k(z_1 ... z_n) \oplus \hat{h} = H(D)$. Then one could let $D = \hat{h} \parallel 111..1 \parallel 111...1 \parallel ....$ This D would be a second pre-image.
We conclude from this example that the choice of p and q is not to be taken lightly.

Example 7.3. (Second Pre-Image Attack 2)
By inserting the same sequence of n zeros and ones, $x_1 ... x_n$, s times (where s is an odd number) we can find $f_k(x_1, ..., x_n) \oplus x_1 ... x_n$ and obviously $f_k(x_1 ... x_n)$ itself. Hence someone

could make a database of different h_i's and their resulting H_i's which would allow a user to quickly come up with a second pre-image.
This serious issue can be combated by generating multiple k's. That way if we insert the same sequence $x_1...x_n$ s times we don't get $H(D) = f_k(x_1...x_n) \oplus x_1...x_n$ outputted, as the $f_k(x_1...x_n)$ vary.
To generate multiple k's, let's call them $k_1, ..., k_n$ one uses the k embedded in the function and lets

$$k_i \equiv k^{2^i} \pmod{N}.$$

Note that the binary representation of every k_i doesn't have to be of length n as we use the *"bad hash"* later in the PRG.

Remark 7.4. (Pre-Image Resistance)
Finding a pre-image from $H(D)$ is difficult as it would involve first finding the H_i's (presumably). The brute-force algorithm for this alone runs in an exponential time complexity.

Proof. (Time Complexity of Brute-Force Algorithm)
Let $A = a_1, a_2, ..., a_k; B = b_1, b_2, ..., b_k; C = c_1, c_2, ..., c_k)$, where $a_i, b_i, c_i \in \{0, 1\}$ and finally let $A = B \oplus C$. Assume $a_1 = 1$. It follows that $b_1 = 1, c_1 = 0$ or $b_1 = 0, c_1 = 1$. We take it that A is known to an attacker. If the attacker wishes to find B and C by brute-force (so by trying every option) they would have to go through up to 2^n possibilities.
Hence it is hard for an attacker to get the H_i's by trying every possibility of $H(D) = H_1 \oplus ... \oplus H_s$ as $H(D) = (H_1 \oplus ... \oplus H_s)$ can be written

$$H(D) = (((H_1 \oplus H_2) \oplus H_3)... \oplus H_{s-1}) \oplus H_s$$

giving us a similar problem to the one above with A, B and C, the core difference being that there are more possibilities. Instead of there being 2^n different options there are now $\left(\frac{s!}{k!(n-k)!}\right)^n$ possibilities, where for s an odd number k must be chosen as any odd number between 1 and s, and for s an even number k must be chosen as any even number between 1 and s.
We conclude that the brute-force algorithm to find the H_i's from $H(D)$ does not run in polynomial time. □

Constantin Vlachos
constantinosvasiliosargyris.vlachos@uzh.ch

References.

[Bos14] Siegfried Bosch. "Vektorräume". In: *Lineare Algebra*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 1–49. ISBN: 978-3-642-55260-1. DOI: 10.1007/978-3-642-55260-1_1. URL: https://doi.org/10.1007/978-3-642-55260-1_1.

[HPS06] Jeffrey Hoffstein, Jill Pipher, and Joseph H Silverman. "NTRU: A ring-based public key cryptosystem". In: *Algorithmic Number Theory: Third International Symposiun, ANTS-III Portland, Oregon, USA, June 21–25, 1998 Proceedings*. Springer, 2006, pp. 267–288.

[Hil86] Raymond Hill. *A first course in coding theory*. Oxford University Press, 1986.

[Dan71] Shanks Daniel. "Class number, a theory of factorization, and genera". In: *Proc. Sympos. Pure Math.* Vol. 20. 1971, pp. 415–440.

[Bon11] Dan Boneh. "Blum–Blum–Shub Pseudorandom Bit Generator". In: *Encyclopedia of Cryptography and Security*. Ed. by Henk C. A. van Tilborg and Sushil Jajodia. Boston, MA: Springer US, 2011, pp. 160–161. ISBN: 978-1-4419-5906-5. DOI: 10.1007/978-1-4419-5906-5_141. URL: https://doi.org/10.1007/978-1-4419-5906-5_141.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. "How to construct random functions". In: *Journal of the ACM (JACM)* 33.4 (1986), pp. 792–807.

[RSA78] Ronald L Rivest, Adi Shamir, and Leonard Adleman. "A method for obtaining digital signatures and public-key cryptosystems". In: *Communications of the ACM* 21.2 (1978), pp. 120–126.

[Mat] *Matrix Calculator*. https://matrixcalc.org. Accessed: 19. 07. 2023.

[Wik23a] Wikipedia contributors. *International Bank Account Number — Wikipedia, The Free Encyclopedia*. [Online; accessed 17-October-2023]. 2023. URL: https://en.wikipedia.org/w/index.php?title=International_Bank_Account_Number&oldid=1175069833.

[POM] ALEXA POMERANTZ. *An introduction to the p-adic numbers*.

[Wik23b] Wikipedia contributors. *Metric space — Wikipedia, The Free Encyclopedia*. [Online; accessed 19-October-2023]. 2023. URL: https://en.wikipedia.org/w/index.php?title=Metric_space&oldid=1180325997.