East Asian Journal on Applied Mathematics
doi: 10.4208/eajam.240522.200822

Vol. **13**, No. 1, pp. 177-193
February 2023

Cryptographic Systems Based on an Algebraic Structure

Łukasz Matysiak[1,*], Monika Chrzaniuk[1,2], Maximilian Duda[1,2], Marta Hanc[1,2], Sebastian Kowalski[1,2], Zoja Skotnicka[1,2] and Martin Waldoch[1,2]

[1] *Kazimierz Wielki University, Institute of Mathematics, ul. Powstancow Wielkopolskich 2, 85-090 Bydgoszcz, Poland.*
[2] *Institute of Informatics, ul. Mikolaja Kopernika 1, 85-074 Bydgoszcz, Poland.*

Received 24 May 2022; Accepted (in revised version) 20 August 2022.

Abstract. In this paper cryptographic systems based on the Dedekind and Galois structures are considered. We supplement the created cryptosystems based on the Dedekind structure with programs written in C ++ and discuss the inner structure of Galois in cryptography. It is well-known that such a structure is based on finite fields only. Our results reveals something more internal. The final section contains additional information about square-free and radical factorizations in monoids consisting in searching for a minimal list of counterexamples. As an open problem, we leave creating a program that would generate such a list and how to use such a list to create a cryptosystem.

AMS subject classifications: 94A60, 11R32

Key words: Monoid, cryptography, Dedekind domain, Galois extension, factorization.

1. Introduction

In this paper we present developed cryptographic programs written in C ++ (see Sections 2 and 3). These cryptographic systems are based on the Dedekind structure — cf. [1, 3]. This motivation is coming from the first author works [2, 3].

Dedekind domains are one of the most important rings in algebra. They have many valuable properties and applications. In such rings, any nonzero proper ideal factors into primes and any non-zero fractional ideal is invertible. In Section 2, we introduce a cryptographic program, where the key is an analog to a fractional ideal. In Section 3 we present another cryptographic program, where an alphabet is analog to the fractional ideal. In Section 4 we have complementary of [2, 3]. In addition, we consider the application of

*Corresponding author. *Email addresses:* `lukmat@ukw.edu.pl` (Ł. Matysiak), `marta.hanc@student.ukw.edu.pl` (M. Hanc), `monika.chrzaniuk@student.ukw.edu.pl` (M. Chrzaniuk), `mxdu0407@gmail.com` (M. Duda), `kspterna@student.ukw.edu.pl` (S. Kowalski), `skotnickazoja@gmail.com` (Z. Skotnicka), `martin.waldoch@student.ukw.edu.pl` (M. Waldoch) polynomial composites and monoid domains in cryptology in the form of certain cryptosystems. In this paper we present a program about this. Section 5 is devoted to a cryptosystem based on a Galois extension. Let us recall that a Galois extension is an algebraic field extension $K \subset L$ that is normal and separable. We introduce an example of such cryptosystem using $\mathbb{Q} \subset \mathbb{Q}(\sqrt{2}, \sqrt{3})$ which is a Galois extension. This cryptosystem can be freely modified while maintaining the idea of its operation. The motivation here is the development of a cryptosystem based on the Galois theory. They do exist, of course, but there is only talk of finite fields. To the best of our knowledge, there are no cryptosystems that go deeper into this science so far.

By a monoid we mean a commutative cancellative monoid. In Section 6, we provide a minimal list of possible counterexamples to find in monoids. Note that [4, Section 4] contains 24 square-free and radical factorizations and all dependencies in general monoids and in particular monoids (GCD-, pre-Schreier-, SR-, ACCP-, atomic, factorial monoids). We recall that a monoid is called GCD-monoid, if for any two elements there exists a greatest common divisor. A monoid H is called a pre-Schreier monoid, if any element $a \in H$ is primal — i.e. for any $b, c \in H$ such that $a \mid bc$ there exist $a_1, a_2 \in H$ such that $a = a_1a_2$, $a_1 \mid b$ and $a_2 \mid c$. A monoid H is called SR-monoid, if every square-free element is radical — cf. [4]. A monoid H is called ACCP-monoid if any ascending sequence of principal ideals of H stabilizes — i.e. for any sequence of principal ideals $I_1 \subset I_2 \subset \cdots$, there exists $n \in \mathbb{N}$ such that $I_n = I_{n+1} = \cdots$. A monoid H is called atomic, if every non-invertible element $a \in H$ is a finite product of irreducibles (atoms). A monoid H is factorial, if each non-invertible element can be written as a product of irreducible elements and this representation is unique. We also recall that any factorial monoid is ACCP-monoid, any ACCP-monoid is atomic, any factorial monoid is GCD-monoid, and any GCD-monoid is pre-Schreier. Since every pre-Schreier is AP-monoid — i.e each its irreducible element (atom) is prime, it yields that every atomic and AP-monoid is factorial.

We also note that Section 6 discusses a separate topic closely related to [4, Section 7]. More exactly, we consider a minimal list of counterexamples that we can look for (Theorem 6.1). Some of them are taken from [4, Section 7]. This list of counterexamples to be looked for concerns checking whether a given implication about whether we can obtain a different factorization from a given factorization (with respect to square-free or radical factorization) from [4, Section 7] is non-empty. Using the laws of logic, we can easily, although not quickly, generate such a list. Open search for such counterexamples is left as an open problem. It is well-known that cryptology has often used factorization of given numbers (or elements, more generally). The theory discussed in [4] and in this section can be successfully used to develop new cryptosystems.

2. First Cryptosystem with a Dedekind Structure

Consider an alphabet $A = \{a_0, a_1, \dots, a_n\}$ such that $|A|$ is a prime number, and let $x \in \{2, 3, \dots, |A|\}$ be the value of a letter in the alphabet, and $k \geqslant 2$ a key. Then

$$y = xk \pmod{|A|},$$

where y is the value of a letter in the alphabet. Assuming that y is an encrypted letter, we get the decrypted letter x by the formula

$$x = \big(y + (k - d)\cdot|A|\big)\cdot k^{-1}, \tag{2.1}$$

where d is the remainder of dividing y by k. The proof of (2.1) can be found in [3, Section 2].

The algorithm in C++ below, has a predefined 28-character alphabet. We encourage the reader to modify it.

```
#include <iostream>

using namespace std;

char pub_alphabet[29], message[100]={}, test=message[100];

int x[29], p=29,noc,k;

int calc_number_of_charactes()
{
noc=0;
for(int i=0; i<100; i++)
{
if(message[i]!=test)
noc++;
}
return noc;
}

int search_ch(char a)
{
char test_key='A';
for(int i=0; i<p; i++)
{
if(a==test_key)
return i+2;
else
{
test_key++;
}
}
}

int encrypt(int m)
```

```
{
return (m*k)%p;
}

int main()
{
pub_alphabet[0]='A';
x[0]=2;

for(int i=1; i<=28; i++)
{
x[i]=2+i;
pub_alphabet[i]='A'+i;
}
pub_alphabet[28]=' ';
do
{
cout << "Enter a key(must be equal or greater than 2)" <<
endl;
cin >> k;
}while(k<2);

cout << "Write a message to encrypt(max 100 characters)" <<
endl;
cin >> message;

calc_number_of_charactes();

for(int i=0; i<noc; i++)
{
message[i]=toupper(message[i]);
cout << encrypt(search_ch(message[i])) << " ";
}

return 0;
}
```

3. Second Cryptosystem with a Dedekind Structure

We now consider another cryptosystem — cf. [3, Section 3]. Let A be a set of characters. Assume that $|A|$ is a prime number and secretly establish a second alphabet A' such that $A' \subset A$ with a prime length.

Let $m_1 m_2 m_3 \ldots m_n$ be the message, we want to encrypt. The secret short alphabet A'

divides large public alphabet into zones. We skip extra characters such that $0, 1$. So we have a clean alphabet from 2. Let us move one over, so we have 1. If $p = |A|$, $q = |A'|$, then we have $\lceil p/q \rceil$ zones. The zero zone contains alphabet from 1 to q, the first zone the alphabet from $q + 1$ to $2q$, etc. The last zone ($\lceil p/q \rceil - 1$) contains alphabet from $\lceil p/q \rceil q$ to p.

We extend the message values by random numbers informing us about the given zone of a given letter and denote this information by z_i, viz.

$$z_1 m_1 z_2 m_2 \ldots z_n m_n.$$

Denote by k the key. Multiplying each value of the message (but not the information about the zone) by k and using the modulo q produces the ciphertext

$$z_1 d_1 z_2 d_2 \ldots z_n d_n, \tag{3.1}$$

where $d_1 d_2 \ldots d_n$ is the encrypted message.

In order to decode the message, we split (3.1) into blocks, each of which contains the zone and message and apply the formula

$$m_i = \frac{d_i + (z_i + t_i \cdot k)|A|}{k},$$

where k is the key, m_i is the decoded letter, d_i the encrypted letter, t_i the zone, and z_i a number satisfying the congruence

$$|A|^{-1} z_i \equiv d_i \pmod{k}.$$

The algorithm in C ++ below is limited to 100 characters. We leave it open as to how this program can be improved and also encourage the reader to modify the algorithm.

```
#include <iostream>

using namespace std;

char pub_alphabet[29], priv_alphabet[3]={'A','B','C'},
message[100]={}, test=message[100];

int x[29], p=29, q=3,noc,k;

int calc_number_of_charactes()
{
noc=0;
for(int i=0; i<100; i++)
{
if(message[i]!=test)
noc++;
}
return noc;
}

int zones(int x)
{
int y=0;
while(x>=q)
{
y++;
x-=q;
}
return y;
}

int search_ch(char a)
{
char test_key='A';
for(int i=0; i<p; i++)
{
if(a==test_key)
return i;
else
{
test_key++;
}
}
}

int encrypt(int m)
{
return (m*k)%q;
}

bool key_check()
{
if(k%3!=0)
return true;

else
return false;
}
```

```
int main()
{
pub_alphabet[0]='A';
x[0]=2;

for(int i=1; i<=28; i++)
{
x[i]=2+i;
pub_alphabet[i]='A'+i;

}
pub_alphabet[28]=' ';

do
{
cout << "Enter a key (must not be divisible by 3)" << endl;
cin >> k;
}while(key_check()==false);

cout << "Write a message to encrypt(max 100 characters)" <<
endl;
cin >> message;

calc_number_of_charactes();

for(int i=0; i<noc; i++)
{
message[i]=toupper(message[i]);
cout <<  zones(search_ch(message[i])) <<
encrypt(search_ch(message[i])) << " ";
}

return 0;
}
```

4. Cryptosystem Based on Monoid Domains

In this section we use notions and results from [3, Section 5]. In particular, if F is a field and M a submonoid of $\mathbb{Q}_+$, then we can construct the following monoid domain:

$$F[M] = F[X;M] = \left\{a_0X^{m_0} + \cdots + a_nX^{m_n} \mid a_i \in F, m_i \in M\right\}.$$

Note that any alphabet of characters creates a finite set. Although most ciphers are based on finite sets, we can consider using an infinite alphabets $\mathbb{A}$. Nevertheless, in real applications such alphabets can be cyclical sets with an index — i.e. given cycles. For example, A_0 - 0, B_0 - 1, ... , Z_0 - 25, A_1 - 0, B_1 - 1,..., where A_i=A, ..., Z_i=Z for $i = 0, 1, \ldots$. We note that this is isomorphic to a monoid $\mathbb{N}_0$ of non-negative integers with the corresponding isomorphism

$$f : \mathbb{A} \to \mathbb{N}_0, \quad f(m_i) = i.$$

Now we can use the monoid domain by a map

$$\varphi : \mathbb{A} \to F[\mathbb{A}], \quad \varphi(m_0, m_1, \ldots, m_n) = a_0 X^{m_0} + \cdots + a_n X^{m_n}.$$

In order to encrypt the message $m_0 m_1 m_2 \ldots m_n$, i.e. to transform letters into the numbers by using a function φ, we need a secret key X. If F is a field, we choose any coefficients $a_0, a_1, \ldots, a_n$ from this field. The message $m_0 m_1 m_2 \ldots m_n$ can be then transformed into the polynomial

$$a_0 X^{m_0} + a_1 X^{m_1} + \cdots + a_n X^{m_n}.$$

Computing

$$d_i = a_i X^{m_i} \pmod{|\mathbb{A}|}, \quad i = 0, 1, \ldots, n$$

with a prime $|\mathbb{A}|$, we obtain an encrypted message $d_0 d_1 \ldots d_n$. To decrypt it, one can use the formulas

$$m_i = \log_X \frac{d_i}{a_i} \pmod{|\mathbb{A}|}, \quad i = 0, 1, \ldots, n.$$

The following program is a modification of the presented mathematical algorithm. After entering a key, the program determine the coefficients, which are successive powers of 2 modulo 10.

```
#include <iostream>
#include <math.h>

using namespace std;

char message[100]={}, test=message[100];

int p=29,noc,k;

struct polynomial
{
char a;
int x;
};

int calc_number_of_charactes()
{
noc=0;
```

```
for(int i=0; i<100; i++)
{
if(message[i]!=test)
noc++;
}
return noc;
}

int search_ch(char a)
{
char test_key='A';
for(int i=0; i<p; i++)
{
if(a==test_key)
return i;
else
{
test_key++;
}
}
}

int encrypt(int m, int x)
{
int en;
en=x*pow(k,m);
return en%p;
}

int main()
{
polynomial table[29];
table[0].a='A';

for(int i=1; i<=28; i++)
{
table[i].x=2;
table[i].x=pow(table[i].x,i);
table[i].x%=10;
table[i].a='A'+i;
}
table[28].a=' ';
```

```
cout << "Enter a key" << endl;
cin >> k;

cout << "Write a message to encrypt(max 100 characters)" << endl;
cin >> message;

calc_number_of_charactes();

for(int i=0; i<noc; i++)
{
message[i]=toupper(message[i]);
cout << encrypt(search_ch(message[i]),
table[search_ch(message[i])].x) << " ";
}
cout << endl;
int de;

return 0;
}
```

5. A Cryptosystem Based on a Galois Extension

Let L be a Galois extension field of $\mathbb{Q}$. Recall, if $K \subset L$ is a Galois extension, then $Aut_K L$ is called the Galois group of $K \subset L$ and denoted by $G(L \mid K)$. It is well-known how to form a Galois group of such an extension. We consider a simple example. Let $L = \mathbb{Q}(\sqrt{2}, \sqrt{3})$. Of course L is a Galois extension field of $\mathbb{Q}$. Then $Gal(L \mid \mathbb{Q}) = \{\sigma_1, \sigma_2, \sigma_3, \sigma_4\}$, where

$$\begin{aligned}
\sigma_1\left(a + b\sqrt{2} + c\sqrt{3} + d\sqrt{6}\right) &= a + b\sqrt{2} + c\sqrt{3} + d\sqrt{6} = id,\\
\sigma_2\left(a + b\sqrt{2} + c\sqrt{3} + d\sqrt{6}\right) &= a + b\sqrt{2} - c\sqrt{3} - d\sqrt{6},\\
\sigma_3\left(a + b\sqrt{2} + c\sqrt{3} + d\sqrt{6}\right) &= a - b\sqrt{2} + c\sqrt{3} - d\sqrt{6},\\
\sigma_4\left(a + b\sqrt{2} + c\sqrt{3} + d\sqrt{6}\right) &= a - b\sqrt{2} - c\sqrt{3} + d\sqrt{6}
\end{aligned}$$

with $a, b, c, d \in \mathbb{Q}$. Note that the automorphisms $\sigma_1, \sigma_2, \sigma_3, \sigma_4$ can be conveniently written as

$$\begin{aligned}
\sigma_1(a, b, c, d) &= (a, b, c, d) = id,\\
\sigma_2(a, b, c, d) &= (a, b, -c, -d),\\
\sigma_3(a, b, c, d) &= (a, -b, c, -d),\\
\sigma_4(a, b, c, d) &= (a, -b, -c, d).
\end{aligned}$$

Consider the English alphabet of 26 letters. Let $m_1m_2m_3m_4$ be a 4-letter block of a message. In the Galois group we ignore id because it does not change anything. Let us encode the block by using the automorphism σ_2, i.e.

$$\sigma_2(m_1, m_2, m_3, m_4) = (m_1, m_2, -m_3, -m_4).$$

After that we replace $-m_3$ by the letter that has the opposite position in the alphabet — e.g. D is the 4-th letter in the alphabet, hence $-D$ will be replaced by W because W is the 4-th one to last letter in the alphabet. Similarly, we deal with $-m_4$.

If a letter such as m_1 does not change as a result, we check what place in the alphabet does it take, divide it by 2, and replace by the letter from that position. However, if the remainder of dividing of this letter's position by 2 is 1, then we round down and instead of placing a capital letter, we place a small one. The letter m_2 can be handled analogously.

The other automorphisms can be encoded analogously. Thus using Table 1 and the Galois group, we can decode the encoded message without any problems.

Table 1: Action table.

Place	Letter	Negative letter	The letter it turns into if it does not change
1	A	Z	z
2	B	Y	A
3	C	X	a
4	D	W	B
5	E	V	b
6	F	U	C
7	G	T	c
8	H	S	D
9	I	R	d
10	J	Q	E
11	K	P	e
12	L	O	F
13	M	N	f
14	N	M	G
15	O	L	g
16	P	K	H
17	Q	J	h
18	R	I	I
19	S	H	i
20	T	G	J
21	U	F	j
22	V	E	K
23	W	D	k
24	X	C	L
25	Y	B	l
26	Z	A	M

In a very similar way, we can construct an analogous cryptosystem, where the key is an extension field of rational numbers, and if the addressee knows the Galois theory, he can easily calculate the Galois group of such extension and perform appropriate steps.

It is known — cf. [5, Theorem 2.2], that every finite group is a Galois group of a extension field of $\mathbb{Q}$. This means that instead of extension field of $\mathbb{Q}$, we can pass a finite group as a key. This will further increase the security of the cryptosystem.

6. Minimal List of Counterexamples in Monoids

Recall that a 24 square-free and radical factorizations and all dependencies in general monoids and in particular monoids has been discussed in [4, Section 4]. In this section, we consider a minimal list of possible counterexamples, which can be found in commutative cancellative monoids. Some of them are presented in [4, Section 7].

Theorem 6.1. *The statement that there are (in general) no other implications than the ones stated in* [4] *is equivalent to the existence of the following counter-examples:*

1. *Any monoid satisfies the conditions*

   $4s \land \neg 0s,\ 5s \land \neg 0s,\ 5.1s \land \neg 0s,\ 3s \land \neg 1s,\ 5.1s \land \neg 4s,\ 4s \land \neg 4.1s,\ 2s \land \neg 4.2s,\ 5.3s \land \neg 4.2s,\ 1s \land \neg 5s,\ 3s \land \neg 5s,\ 4s \land \neg 5s,\ 5.1s \land \neg 5s,\ 2s \land \neg 5.3s,\ 4s \land \neg 5.3s,\ 4.1s \land \neg 5.3s,\ 1s \land \neg 6s,\ 4s \land \neg 6s,\ 5s \land \neg 6s,\ 5.1s \land \neg 6s,\ 5r \land \neg 0r,\ 0r \land \neg 1r,\ 3r \land \neg 1r,\ 5.1r \land \neg 4r,\ 5.2r \land \neg 4.1r,\ 1r \land \neg 4.2r,\ 3r \land \neg 4.2r,\ 5.3r \land \neg 4.2r,\ 1r \land \neg 5.3r,\ 3r \land \neg 5.3r,\ 4r \land \neg 5.3r,\ 1r \land \neg 6r,\ 4r \land \neg 6r.$

2. *Non-factorial GCD-monoids satisfy the conditions*

   $4/5s \land \neg 0s/1s/2s/3s,\ 6s \land \neg 4.1s\ /5.1s,\ 5.3s \land \neg 4.2s/5.2s,\ 4.1s/5.1s \land \neg 6s.$

3. *Pre-Schreier non-GCD-monoids satisfy the conditions*

   $4s/5s \land \neg 0,\ 4.1s/5.1s \land \neg 4s/5s,\ 4.2s/5.2s \land \neg 4.1s/5.1s,\ 5.3s \land \neg 4.2s/5.2s,\ 3s \land \neg 5.3s,\ 0s \land \neg 6s,\ 4.1s/5.1s \land \neg 6s.$

4. *SR-non-pre-Schreier monoids satisfy the conditions*

   $5s \land \neg 0s,\ 0s \land \neg 1s,\ 3s \land \neg 1s,\ 5.1s \land \neg 4s,\ 5.2s \land \neg 4.1s,\ 1s \land \neg 4.2s,\ 3s \land \neg 4.2s,\ 5.3s \land \neg 4.2s,\ 1s \land \neg 5.3s,\ 3s \land \neg 5.3s,\ 4s \land \neg 5.3s,\ 1s \land \neg 6s,\ 4s \land \neg 6s.$

5. *Non-factorial ACCP-monoids satisfy the conditions*

   $4s \land \neg 0s,\ 4.1s \land \neg 0s,\ 5.2s \land \neg 0s,\ 6s \land \neg 0s,\ 0s \land \neg 1s,\ 5.1s \land \neg 4s,\ 4s \land \neg 4.1s,\ 5.2s \land \neg 4.1s,\ 2s \land \neg 4.2s,\ 5.3s \land \neg 4.2s,\ 6s \land \neg 4.2s,\ 1s \land \neg 5s,\ 4s \land \neg 5s,\ 4.1s \land \neg 5s,\ 5.3s \land \neg 5s,\ 6s \land \neg 5s,\ 2s/3s \land \neg 5.3s,\ 4s \land \neg 5.3s,\ 4.1s \land \neg 5.3s,\ 1s \land \neg 6s,\ 4s \land \neg 6s,\ 4.1s \land \neg 6s,\ 5.2s \land \neg 6s,\ 4r \land \neg 0r,\ 5.3r \land \neg 0r,\ 0r \land \neg 1r,\ 4.1r \land \neg 4r,\ 4.2r \land \neg 4.1r,\ 1r \land \neg 4.2r,\ 5.3r \land \neg 4.2r,\ 1r \land \neg 5.3r,\ 4r \land \neg 5.3r.$

6. *An atomic non-ACCP monoid satisfy exactly the same conditions as in item* 1.

Proof. We will only give a sketch of the proof. Let us demonstrate the proof ideas by the diagram below. Of course, appropriate diagrams and tables from [4] should be included.

$$\begin{tikzcd}
 & 4 \arrow[r, Rightarrow] \arrow[dr, Rightarrow] & 6 \\
A \arrow[ur, Rightarrow] \arrow[r, Rightarrow] & B \arrow[r, Rightarrow] & 1 \\
2 \arrow[r, Rightarrow] \arrow[dr, Rightarrow] & 3 \arrow[ur, Rightarrow] & \\
 & 5 &
\end{tikzcd}$$

Arranging all properties in the order $A, B, 1, 2, 3, 4, 5, 6$, we analyze all implications in this order, starting from the first stage.

I. A. (1) We have implications:

$$A \Rightarrow B, \quad A \Rightarrow 1, \quad A \Rightarrow 4, \quad A \Rightarrow 6.$$

(2) Remaining implications from A are

$$A \Rightarrow 2, \quad A \Rightarrow 3, \quad A \Rightarrow 5.$$

(3) Dependencies between them are

$$(A \Rightarrow 2) \Rightarrow (A \Rightarrow 3), \quad (A \Rightarrow 2) \Rightarrow (A \Rightarrow 5).$$

(4) It is enough to have counter-examples to the implications

$$A \Rightarrow 3, \quad A \Rightarrow 5.$$

I. B. (1) We have implication

$$B \Rightarrow 1.$$

(2) Remaining implications from B are

$$B \Rightarrow A, \quad B \Rightarrow 2, \quad B \Rightarrow 3, \quad B \Rightarrow 4, \quad B \Rightarrow 5, \quad B \Rightarrow 6.$$

(3) Dependencies between them are

$$(B \Rightarrow A) \Rightarrow (B \Rightarrow 4) \Rightarrow (B \Rightarrow 6), \quad (B \Rightarrow 2) \Rightarrow (B \Rightarrow 3), \quad (B \Rightarrow 2) \Rightarrow (B \Rightarrow 5).$$

(4) It is enough to have counter-examples to the implications

$$B \Rightarrow 3, \quad B \Rightarrow 5, \quad B \Rightarrow 6.$$

I. 1. (1) We have no implications starting from 1.

(2) Remaining implications from 1 are

$$1 \Rightarrow A, \quad 1 \Rightarrow B, \quad 1 \Rightarrow 2, \quad 1 \Rightarrow 3, \quad 1 \Rightarrow 4, \quad 1 \Rightarrow 5, \quad 1 \Rightarrow 6.$$

(3) Dependencies between them are

$$\begin{aligned}&(1 \Rightarrow A) \Rightarrow (1 \Rightarrow B), \quad (1 \Rightarrow A) \Rightarrow (1 \Rightarrow 4) \Rightarrow (1 \Rightarrow 6),\\ &(1 \Rightarrow 2) \Rightarrow (1 \Rightarrow 3), \quad (1 \Rightarrow 2) \Rightarrow (1 \Rightarrow 5).\end{aligned}$$

(4) It is enough to have counter-examples to implications

$$1 \Rightarrow B, \quad 1 \Rightarrow 3, \quad 1 \Rightarrow 5, \quad 1 \Rightarrow 6.$$

I. 2. (1) We have implications

$$2 \Rightarrow 1, \quad 2 \Rightarrow 3, \quad 2 \Rightarrow 5.$$

(2) Remaining implications from 2 are

$$2 \Rightarrow A, \quad 2 \Rightarrow B, \quad 2 \Rightarrow 4, \quad 2 \Rightarrow 6.$$

(3) Dependencies between them are

$$(2 \Rightarrow A) \Rightarrow (2 \Rightarrow B), \quad (2 \Rightarrow A) \Rightarrow (2 \Rightarrow 4) \Rightarrow (2 \Rightarrow 6).$$

(4) It is enough to have counter-examples to implications

$$2 \Rightarrow B, \quad 2 \Rightarrow 6.$$

I. 3. (1) We have implication

$$3 \Rightarrow 1.$$

(2) Remaining implications from 3 are

$$3 \Rightarrow A, \quad 3 \Rightarrow B, \quad 3 \Rightarrow 2, \quad 3 \Rightarrow 4, \quad 3 \Rightarrow 5, \quad 3 \Rightarrow 6.$$

(3) Dependencies between them are

$$(3 \Rightarrow A) \Rightarrow (3 \Rightarrow B), \quad (3 \Rightarrow A) \Rightarrow (3 \Rightarrow 4) \Rightarrow (3 \Rightarrow 6), \quad (3 \Rightarrow 2) \Rightarrow (3 \Rightarrow 5).$$

(4) It is enough to have counter-examples to implications

$$3 \Rightarrow B, \quad 3 \Rightarrow 5, \quad 3 \Rightarrow 6.$$

I. 4. (1) We have implications

$$4 \Rightarrow 1, \quad 4 \Rightarrow 6.$$

(2) Remaining implications from 4 are

$$4 \Rightarrow A, \quad 4 \Rightarrow B, \quad 4 \Rightarrow 2, \quad 4 \Rightarrow 3, \quad 4 \Rightarrow 5.$$

(3) Dependencies between them are

$$(4 \Rightarrow A) \Rightarrow (4 \Rightarrow B), \quad (4 \Rightarrow 2) \Rightarrow (4 \Rightarrow 3), \quad (4 \Rightarrow 2) \Rightarrow (4 \Rightarrow 5).$$

(4) It is enough to have counter-examples to implications

$$4 \Rightarrow B, \quad 4 \Rightarrow 3, \quad 4 \Rightarrow 5.$$

I. 5. (1) We have no implications starting from 5.

(2) Remaining implications from 5 are

$$5 \Rightarrow A, \quad 5 \Rightarrow B, \quad 5 \Rightarrow 1, \quad 5 \Rightarrow 2, \quad 5 \Rightarrow 3, \quad 5 \Rightarrow 4, \quad 5 \Rightarrow 6.$$

(3) Dependencies between them are

$$\begin{aligned}&(5 \Rightarrow A) \Rightarrow (5 \Rightarrow B) \Rightarrow (5 \Rightarrow 1),\\ &(5 \Rightarrow A) \Rightarrow (5 \Rightarrow 4) \Rightarrow (5 \Rightarrow 6),\\ &(5 \Rightarrow 2) \Rightarrow (5 \Rightarrow 3).\end{aligned}$$

(4) It is enough to have counter-examples to implications

$$5 \Rightarrow 1, \quad 5 \Rightarrow 3, \quad 5 \Rightarrow 6.$$

I. 6. (1) We have no implications starting from 6.

(2) Remaining implications from 6 are

$$6 \Rightarrow A, \quad 6 \Rightarrow B, \quad 6 \Rightarrow 1, \quad 6 \Rightarrow 2, \quad 6 \Rightarrow 3, \quad 6 \Rightarrow 4, \quad 6 \Rightarrow 5.$$

(3) Dependencies between them are

$$\begin{aligned}&(6 \Rightarrow A) \Rightarrow (6 \Rightarrow B) \Rightarrow (6 \Rightarrow 1),\\ &(6 \Rightarrow A) \Rightarrow (6 \Rightarrow 4) \Rightarrow (6 \Rightarrow 1),\\ &(6 \Rightarrow 2) \Rightarrow (6 \Rightarrow 3), \quad (6 \Rightarrow 2) \Rightarrow (6 \Rightarrow 5).\end{aligned}$$

(4) It is enough to have counter-examples to implications

$$6 \Rightarrow 1, \quad 6 \Rightarrow 3, \quad 6 \Rightarrow 5.$$

In the first stage we have selected the following implications:

$$\begin{aligned}
&A \Rightarrow 3, \quad A \Rightarrow 5,\\
&B \Rightarrow 3, \quad B \Rightarrow 5, \quad B \Rightarrow 6,\\
&1 \Rightarrow B, \quad 1 \Rightarrow 3, \quad 1 \Rightarrow 5, \quad 1 \Rightarrow 6,\\
&2 \Rightarrow B, \quad 2 \Rightarrow 6,\\
&3 \Rightarrow B, \quad 3 \Rightarrow 5, \quad 3 \Rightarrow 6,\\
&4 \Rightarrow B, \quad 4 \Rightarrow 3, \quad 4 \Rightarrow 5,\\
&5 \Rightarrow 1, \quad 5 \Rightarrow 3, \quad 5 \Rightarrow 6,\\
&6 \Rightarrow 1, \quad 6 \Rightarrow 3, \quad 6 \Rightarrow 5.
\end{aligned}$$

In the second stage we will analyze dependencies between implications from this list ending at each of the consecutive properties.

II. B. (1) We consider the following implications ending at B:

$$1 \Rightarrow B, \quad 2 \Rightarrow B, \quad 3 \Rightarrow B, \quad 4 \Rightarrow B.$$

(2) Dependencies between them are

$$(1 \Rightarrow B) \Rightarrow (4 \Rightarrow B), \quad (1 \Rightarrow B) \Rightarrow (3 \Rightarrow B) \Rightarrow (2 \Rightarrow B).$$

(3) It is enough to have counter-examples to implications

$$2 \Rightarrow B, \quad 4 \Rightarrow B.$$

II. 1. (1) We consider the following implications ending at 1:

$$5 \Rightarrow 1, \quad 6 \Rightarrow 1.$$

(2) No dependencies between them.

II. 3 (1) We consider the following implications ending at 3:

$$A \Rightarrow 3, \quad B \Rightarrow 3, \quad 1 \Rightarrow 3, \quad 4 \Rightarrow 3, \quad 5 \Rightarrow 3, \quad 6 \Rightarrow 3.$$

(2) Dependencies between them are

$$\begin{aligned}
&(1 \Rightarrow 3) \Rightarrow (B \Rightarrow 3) \Rightarrow (A \Rightarrow 3),\\
&(1 \Rightarrow 3) \Rightarrow (4 \Rightarrow 3) \Rightarrow (A \Rightarrow 3),\\
&(6 \Rightarrow 3) \Rightarrow (4 \Rightarrow 3) \Rightarrow (A \Rightarrow 3).
\end{aligned}$$

(3) It is enough to have counter-examples to implications

$$A \Rightarrow 3, \quad 5 \Rightarrow 3.$$

The proof is complete. □

We strongly encourage the reader to write a program generating the minimum set of counterexamples for any set of implications. Besides, we leave the question about how to use such a list to create a cryptosystem.

7. Code Availability

The code in this paper is available for any use.

References

[1] M. Jankowska and Ł. Matysiak, *A polynomial composites and monoid domains as algebraic structures and their applications*, Global Journal of Science Frontier Research: F Mathematics and Decision Sciences **21**(3), (2021).

[2] Ł. Matysiak, *Generalized RSA cipher and Diffie-Hellman protocol*, J. Appl. Math.& Informatics **39**(1-2), 93–103 (2021).

[3] Ł. Matysiak, *A structure of Dedekind in the cryptosystem*, SCIREA Journal of Mathematics **7**(1), 1–8 (2022).

[4] Ł. Matysiak, *On square-free and radical factorizations and relationships with the Jacobian conjecture*, accepted in The Asian Journal of Mathematics (2022), `https://lukmat.ukw.edu.pl/files/14square-free.pdf`

[5] Ł. Matysiak, *The inverse Galois problem*, J. Appl. Math. Inform. **40**(3-4), 765–767 (2022).