2-ADIC NUMBERS IN CRYPTOGRAPHY

SIDDHARTH K

Abstract. This paper explains how 2-adic numbers can be used to study certain stream ciphers. A stream cipher encrypts a message by combining it with a long pseudorandom sequence of bits. One way to generate such a sequence is with a shift register. The usual kind, called a linear feedback shift register, uses addition modulo 2. A related kind, called a feedback with carry shift register, also keeps track of carries. These carries make the connection to the 2-adic numbers. In this paper, we make use of the fact that an infinite binary sequence can be viewed as a 2-adic integer. Under this point of view, the output of a feedback with carry shift register is a rational 2-adic number.

1. Introduction

Suppose Alice wants to send a secret message to Bob. If Eve sees the message while it is being sent, Alice and Bob still want Eve to learn nothing useful.

One common method for this is a *stream cipher*, where Alice has the message written as bits:

$$P_0, P_1, P_2, \ldots,$$

where P_i is either 0 or 1. Alice also has a secret-looking stream of bits

$$K_0, K_1, K_2, \ldots$$

This is called the *keystream*. Alice encrypts the message by adding the message bits and the keystream bits modulo 2:

$$C_i = P_i + K_i \pmod 2.$$

Bob can decrypt because adding the same bit twice modulo 2 gives back the original sequence:

$$P_i = C_i + K_i \pmod 2.$$

So the main problem is that Alice and Bob need to create the same keystream without Eve being able to predict it.

Date: May 2026.

One solution to this problem is the *shift register*, which is a simple machine that stores a few bits and uses them to produce more bits.

This paper introduces a type of shift register called a *feedback with carry shift register*, or *FCSR*. It turns out that FCSRs are naturally connected to arithmetic in the 2-adic numbers, and the goal of this paper is to explain this connection.

2. Preliminaries

A 2-adic integer is an infinite sum of the form

$$a_0 + a_1 2 + a_2 2^2 + a_3 2^3 + \cdots ,$$

where each a_i is either 0 or 1. The set of all 2-adic integers is $\mathbb{Z}_2$. These numbers converge under the p-adic absolute value because the powers 2^i become divisible by larger powers of 2, and are therefore smaller.

3. Infinite Bit Strings as 2-adic Integers

The 2-adic integers give us a clean way to represent infinite bit strings. Given a binary sequence

$$a_0, a_1, \ldots ,$$

we can associate it to the 2-adic integer

$$\alpha = a_0 + a_1 2 + a_2 2^2 + \cdots .$$

For example, we can represent the sequence $1, 1, 1, 1 \ldots$ with the 2-adic integer $1 + 2 + 2^2 + \cdots = -1$. This also agrees with how negative numbers are often stored inside computers.

Another example is the sequence: $1, 0, 1, 0, \ldots$. This corresponds to

$$1 + 2^2 + 2^4 + \cdots = 1 + 4 + 4^2 + \cdots = \frac{1}{1-4} = -\frac{1}{3}.$$

So the repeating binary sequence $101010 \ldots$ can be represented by the 2-adic number $-1/3$.

4. Ordinary Shift Registers

Now, we move toward cryptography.

A shift register stores a finite list of bits. At each step, it shifts the bits over and creates a new bit using the old ones. The simplest important kind is called a *linear feedback shift register*, or *LFSR*.

For example, suppose a register stores three bits. We might define the next bit by the rule

$$a_{n+3} = a_{n+1} + a_n \pmod 2.$$

For instance, if the starting bits are

$$a_0 = 1, \quad a_1 = 0, \quad a_2 = 0,$$

the remaining bits would be

$$\begin{aligned} a_3 &= a_1 + a_0 = 1 \pmod 2, \\ a_4 &= a_2 + a_1 = 0 \pmod 2, \\ a_5 &= a_3 + a_2 = 1 \pmod 2. \end{aligned}$$

Continuing this process gives a long sequence of bits. While LFSRs are fast, simple, can be built into hardware, and can produce sequences with long periods, the fact that they are linear makes them attackable. If Eve sees enough output bits, she can recover the recurrence rule and predict all future bits.

5. Feedback with Carry Shift Registers

This motivates the next idea, which is the *feedback with carry shift register*, or *FCSR*. This is like an LFSR, except that it stores a small memory value called the carry.

The register still stores bits, but when it computes the next bit, it adds selected old bits as ordinary integers, not just modulo 2. Then it outputs the parity of the sum and saves the rest as the new carry.

5.1. **The definition.** Suppose the register contains r bits:

$$a_{n-r}, a_{n-r+1}, \ldots, a_{n-1}.$$

There is also an integer memory value m_{n-1}. We choose tap values

$$q_1, q_2, \ldots, q_r,$$

where each q_i is either 0 or 1.

At each step n, compute

$$\sigma_n = q_1 a_{n-1} + q_2 a_{n-2} + \cdots + q_r a_{n-r} + m_{n-1}.$$

Then the new bit is

$$a_n = \sigma_n \pmod 2.$$

Define the new memory

$$m_n = \frac{\sigma_n - a_n}{2},$$

which is an integer because $a_n \equiv \sigma_n \pmod 2$.

The register then shifts, and the process repeats.

In an LFSR, we only care about the sum mod 2. So

$$1 + 1 = 0 \pmod 2.$$

The extra 1 disappears. But in an FCSR, this information is stored as memory. If the tapped bits add to $1 + 1 = 2$, then the output bit is 0 and the carry is 1.

5.2. **The Connection Integer.** Each FCSR has a number attached to it called its connection integer. Given taps

$$q_1, q_2, \ldots, q_r,$$

define

$$q = -1 + q_1 2 + q_2 2^2 + \cdots + q_r 2^r.$$

If $q_r = 1$, then q is a positive odd integer. This number controls the behavior of the sequence.

6. Main Theorem

Theorem 6.1. *Let*

$$a_0, a_1, a_2, \ldots$$

be the output of an FCSR with connection integer

$$q = -1 + q_1 2 + q_2 2^2 + \cdots + q_r 2^r,$$

with $q_i \in \{0, 1\}$ *and* $q_r = 1$. *Define*

$$\alpha = \sum_{n=0}^{\infty} a_n 2^n \in \mathbb{Z}_2.$$

Then there is an integer p *such that*

$$\alpha = \frac{p}{q}$$

inside $\mathbb{Z}_2$.

Proof. For $n \geq r$, the FCSR rule says

$$\sigma_n = q_1 a_{n-1} + q_2 a_{n-2} + \cdots + q_r a_{n-r} + m_{n-1}.$$

Also, $a_n \equiv \sigma_n \pmod 2$, and $m_n = \frac{\sigma_n - a_n}{2}$. This means

$$\sigma_n = a_n + 2m_n,$$

and therefore,

$$a_n + 2m_n = q_1 a_{n-1} + q_2 a_{n-2} + \cdots + q_r a_{n-r} + m_{n-1},$$

or

$$a_n = q_1 a_{n-1} + q_2 a_{n-2} + \cdots + q_r a_{n-r} + m_{n-1} - 2m_n.$$

Now form the 2-adic number

$$\alpha = a_0 + a_1 2 + a_2 2^2 + \cdots .$$

Split it into its first r terms and the rest:

$$\alpha = x + \sum_{n=r}^{\infty} a_n \cdot 2^n,$$

where

$$x = a_0 + a_1 2 + \cdots + a_{r-1} 2^{r-1}.$$

Now, substitute the FCSR recurrence into the tail:

$$\alpha = x + \sum_{n=r}^{\infty} (q_1 a_{n-1} + q_2 a_{n-2} + \cdots q_r a_{n-r} + m_{n-1} - 2m_n) 2^n.$$

Distributing the sum gives us

$$\alpha = x + \sum_{i=1}^{r} q_i \sum_{n=r}^{\infty} a_{n-i} 2^n + \sum_{n=r}^{\infty} m_{n-1} 2^n - \sum_{n=r}^{\infty} 2m_n 2^n.$$

Now we handle these sums one at a time.

First, consider

$$\sum_{n=r}^{\infty} a_{n-i} 2^n.$$

We can write

$$\sum_{n=r}^{\infty} a_{n-i} 2^n = 2^i \sum_{n=r}^{\infty} a_{n-i} 2^{n-i} = 2^i \sum_{j=r-i}^{\infty} a_j 2^j,$$

where we've set $j = n - i$. Since

$$\alpha = \sum_{j=0}^{\infty} a_j 2^j,$$

we can write

$$\sum_{j=r-i}^{\infty} = \alpha - \sum_{j=0}^{r-i-1} a_j 2^j.$$

Therefore

$$\sum_{n=r}^{\infty} a_{n-i} 2^n = 2^i \left(\alpha - \sum_{j=0}^{r-i-1} a_j 2^j \right).$$

For convenience, define

$$X_i = \sum_{j=0}^{r-i-1} a_j 2^j.$$

This gives us

$$\sum_{n=r}^{\infty} a_{n-i}2^n = 2^i(\alpha - X_i),$$

so

$$\sum_{i=1}^{r} q_i \sum_{n=r}^{\infty} a_{n-i}2^n = \sum_{i=1}^{r} q_i 2^i(\alpha - X_i)$$

$$= \sum_{i=1}^{r} q_i 2^i \alpha - \sum_{i=1}^{r} q_i 2^i X_i = \alpha \sum_{i=1}^{r} q_i 2^i - \sum_{i=1}^{r} q_i 2^i X_i.$$

But

$$q = -1 + q_1 2 + q_2 2^2 + \cdots + q_r 2^r,$$

so the sum becomes

$$(q+1)\alpha - \sum_{i=1}^{r} q_i 2^i X_i.$$

Now, we look at the memory part:

$$\sum_{n=r}^{\infty} m_{n-1}2^n - \sum_{n=r}^{\infty} 2m_n 2^n.$$

We can rewrite this as

$$\sum_{n=r}^{\infty} m_{n-1}2^n - \sum_{n=r}^{\infty} m_n 2^{n+1}.$$

Everything telescopes except the first term. To show this, consider the partial sums:

$$\sum_{n=r}^{N} m_{n-1}2^n - \sum_{n=r}^{N} m_n 2^{n+1} = m_{r-1}2^r - m_N 2^{N+1}.$$

As $N \to \infty$,

$$m_N 2^{N+1} \to 0$$

in $\mathbb{Z}_2$ because m_N is an integer and

$$|m_N 2^{N+1}|_2 \leq 2^{-(N+1)}.$$

Therefore,

$$\sum_{m=r}^{\infty} m_{n-1}2^n - \sum_{m=r}^{\infty} m_n 2^{n+1} = m_{r-1}2^r.$$

Putting everything back together, we get

$$\alpha = x + \sum_{i=1}^{r} q_i \sum_{n=r}^{\infty} a_{n-i}2^n + \sum_{n=r}^{\infty} m_{n-1}2^n - \sum_{n=r}^{\infty} 2m_n 2^n$$

$$= x + (q+1)\alpha - \sum_{i=1}^{r} q_i 2^i X_i + 2^r m_{r-1}.$$

Rearranging gives us

$$\alpha - (q+1)\alpha = -q\alpha = x - \sum_{i=1}^{r} q_i 2^i X_i + 2^r m_{r-1}.$$

Define

$$p = -x + \sum_{i=1}^{r} q_i 2^i X_i - 2^r m_{r-1},$$

which is an integer. Then

$$-q\alpha = -p,$$

or

$$\alpha = \frac{p}{q}.$$

This proves that the FCSR output sequence is the 2-adic expansion of a rational number where the denominator is the connection integer. ■

Example 6.1. Take the connection integer $q = 5$. Then

$$q + 1 = 6 = 2 + 4,$$

so the taps are on the previous two bits. Starting with

$$a_0 = 1, \quad a_1 = 1, \quad m_1 = 0,$$

we get

$$\sigma_2 = a_1 + a_0 + m_1 = 2,$$

so

$$a_2 = 0, \quad m_2 = 1.$$

For $n = 3$,

$$\sigma_3 = a_2 + a_1 + m_2 = 2,$$

$$a_3 = 0, \quad m_3 = 1.$$

Continuing in this way, we get the sequence

$$11001100\cdots.$$

As a 2-adic number, this is

$$\begin{aligned} 11001100\cdots &= 1 + 2 + 2^4 + 2^5 + 2^8 + 2^9 + \cdots \\ &= (1+2)(1 + 2^4 + 2^8 + \cdots) \\ &= 3(1 + 16 + 16^2 + \cdots) \\ &= 3\left(-\frac{1}{15}\right) \\ &= -\frac{1}{5}. \end{aligned}$$

So this FCSR has connection integer 5, and its output sequence represents $-1/5$. This is exactly what the theorem predicts.

7. Connection to Cryptography

A stream cipher needs a keystream:

$$K_0, K_1, K_2, \ldots$$

An FCSR can generate such a stream. The secret key could determine the starting bits, starting carry, the connection integer, or some combination of these. Then encryption works as

$$C_i = P_i + K_i \pmod 2,$$

and decryption works the same way:

$$P_i = C_i + K_i \pmod 2.$$

Someone might want to use FCSRs instead of LFSRs because the carries make the recurrence nonlinear. But from the 2-adic point of view, the sequence still has structure. It is arithmetic in $\mathbb{Z}_2$, making it vulnerable to 2-adic rational approximation attacks.

7.1. **2-adic span.**

For LFSRs, there is an important idea called linear span or linear complexity. It measures the size of the smallest LFSR that can generate a given sequence. If a sequence has small linear complexity, Eve may only need to see a short part of the sequence to reconstruct the whole generator.

For FCSRs, there is an analogous idea called 2-adic span (see [1]). This roughly measures the size of the smallest FCSR that can generate a given sequence. If the 2-adic span is small, an attacker may be able to guess the rational number p/q from seeing many bits of the keystream.

7.2. **Rational Approximation Attacks.** The main theorem states that FCSR outputs are rational 2-adic numbers $\alpha = p/q$. So if Eve sees the first several output bits, she knows an approximation to α modulo a high power of 2.

So whether or not Eve is able to attack depends on how well she can recover a small rational approximation p/q from $a_0, a_1, \ldots, a_N$, or equivalently, $\alpha \pmod{2^N}$.

Klapper and Goresky studied this type of attack ([1]). Their paper explains that a 2-adic rational approximation method can construct the smallest FCSR that generates a sequence, using only a limited number of bits depending on the 2-adic span. So while the mathematical pattern behind FCSRs make them efficient, it also makes them attackable if the pattern is too visible.

References

[1] Andrew Klapper and Mark Goresky, "2-adic Shift Registers," *Fast Software Encryption*, Lecture Notes in Computer Science, vol. 809, Springer, pp. 174–178.