Prime Base, Prime Moduli PRN Generator

Intelligence and Information Systems, Raytheon Company, Aurora, Colorado, USA

Contact Author - Palak Thakkar, 16800 E. CentreTech Parkway, Aurora, CO 80012
Palak.P.Thakkar@Raytheon.com, (720)-858-4260
Scott Imhoff, 16800 E. CentreTech Parkway, Aurora, CO 80012
Scott_Imhoff@Raytheon.com, (720)-858-4287
John Harms, 16800 E. CentreTech Parkway, Aurora, CO 80012
John_M_Harms@Raytheon.com, (720)-858-4677

IPCV’13

Abstract – This paper presents an algorithm, which utilizes successive multiplications by a prime base and two stages of congruencing in order to generate a pseudorandom number (PRN) for use in encryption. The algorithm has advantages in one-way communication devices because only two prime numbers have to be known in order to decode the sequence. The efficiency of the algorithm stems from there being only a single multiplication tap. The code is determined by first generating the sequence in \( n \) of \( s_n = z^n \mod(p) \) where \( p \) and \( z \) are primes. Then \( s_n \mod(2) \) gives the binary sequence which can be used as a PRN code.

Keywords: Linear Feedback Shift Register, Pseudorandom Number, Prime Base, Prime Moduli

1 Introduction

This paper presents a new, innovative, and efficient pseudorandom noise (PRN) generator. The PRN code bits are generated with successive multiplications using a prime base and two stages of congruence.

PRN codes are considered “pseudorandom” because the sequence appears to be random. PRN is not actually random as it is completely known at the time of modulation. PRN has several major characteristics: PRN must be deterministic, meaning that the subscriber station must be able independently generate the code that matches the base station code. PRN must have the statistical properties of white noise, so that it appears random to a listener without prior knowledge of the code. The cross-correlation between any two sets of PRN code must be small. The PRN code must also have a long period.

The problem with generating PRN codes is in order to correlate the arriving signal the receiver often needs to possess a long reference code along with complex circuitry. If there is any error in the code, the correlation inner product can peak at the wrong sample, or if decoding, the decoding can break down. Some PRN codes may contain repeating sequences, which can hurt the orthogonality of the code.

There is a theorem in number theory that says that if \( p \) is a prime and \( (z, p) = 1 \), then the period is \( p - 1 \) in the sequence \( s(n) = z^n \mod(p) \). Thus two examples are shown in Figure 1.

<table>
<thead>
<tr>
<th>( 7^n \mod(13) )</th>
<th>( 7^n \mod(17) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>14</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>10</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>15</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>12</td>
<td>9</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
</tr>
</tbody>
</table>

Figure 1: Two examples of how \( s(n) = z^n \mod(p) \) have period \( p - 1 \).

A PRN code can be generated by taking the \( s(n) \) modulo 2. An example of this is presented in Figure 2. Here the PRN code is very short; however, we can generate longer PRN codes by using larger primes to get longer periods.
Apply the mod 2 operator to get a binary PRN code.

2 Linear Feedback Shift Registers

PRN generation is currently done by employing linear feedback shift registers (LSFR), powered by a transmitter clock. These shift registers are a row of cell initialized by a specified bit pattern. With each clock cycle, the contents of certain cells are extracted and submitted to a modulo-2 adder. The design of linear code registers can be described by the following polynomial:

\[ S_n = 7^n \pmod{17} \]

\[ PRN = S_n \pmod{2} \]

![Figure 2](image)

### 3 Prime Base, Prime Moduli PRN Generator

The PRN generator proposed in this paper is summarized in Figure 4. The PRN code is generated by successive multiplications by a prime base and two stages of congruencing. The first congruence uses a large prime modulus and the second congruence uses a modulus 2 operation. Because the PRN code is completely determined by the prime base and first prime modulus, or a tuple, it is advantageous for one-way communication in that very little information has to be retained by the receiver. The only piece of information that needs to be retained by the receiver is the tuple.

The PRN Prime Base, Prime Moduli PRN Generator along with the advantage of simplifying the information keeping also maintains good orthogonality by providing against the occurrence of repeating sub-sequences: In the first congruence stage there is a check to see if the product is congruent to unity in order to prevent the occurrence of a repeating sub-sequence.

Another important property of the Prime Base, Prime Moduli PRN Generator is by using a prime number as the modulus in the first stage of the congruence and another prime number for the base tends to increase the period of the sequence, allowing for longer orthogonal codes.

### 4 Results of Algorithm

Prime Base, Prime Moduli PRN Generator resulted in autocorrelation signal to noise ratios as high as 20.6 decibels. LSFRs typically have lower values. Using a typical LSFR of length 30, the SNR is 3.0 dB. Using the 3.0 dB as a measuring stick, Prime Base, Prime Moduli PRN Generator equals or exceeds 3.0 dB in 78.01% of cases for the first 100 prime bases and the first 300 prime moduli. Figure 5 shows the SNR plot for a typical LSFR. Figure 6 shows a case of when Prime Base, Prime Moduli PRN Generator exceeds the LSFR SNR.

### 5 Application of Prime Base, Prime Moduli PRN Generator

There are several identified applications of the Prime Base, Prime Moduli PRN Generator. The applications provided in this section are not exhaustive.

One important application is for the Federal Aviation Administration (FAA). The FAA can allow airport
unique code with one way transmission to aircraft for better integrity, positioning, and ease of implementation to the fleet. The aircraft receives only information then can report through normal channels more accurate position, allowing aircrafts to fly closer to reduce airport congestion in the air space.

Another potential application is use in localized GPS accuracies and one way communication. Receivers would only need to know two prime numbers. The ability to send messages without complex and energy intensive devices would be useful for remote or rugged applications.

The algorithm would also be beneficial to increase the fidelity of secure banking between individuals and mobile devices.

6 Conclusion

The Prime Base, Prime Moduli PRN Generator uses single multiplication by a prime base and two stages of congruencing to generate each bit of PRN code, which allows for implementation using a single tap as opposed to multiple taps in other devices.

Because the PRN code is completely determined by the prime base and the first prime modulus, demodulation becomes easier in the case of one way transmission.

In order to remove the possibility of the occurrence of repeating sub-sequences, there is a check in the first congruence stage to check to see if the product is congruent to unity.

Finally, the Prime Base, Prime Moduli PRN Generator is able to increase the period of the sequence by using a prime number as the modulus in the first stage of the congruence and another prime number for the base. The increase in the period allows for longer codes.

7 References


Figure 3: Typical LSFR

\[ x = \prod p_{i} \text{ (prime)} \; i = 1, j = 1, k = 0, \text{prime} \Rightarrow i = \text{size(prime) - 1}; \text{Discard} = \text{False} \]

Figure 4: Prime Base, Prime Moduli PRN Generator algorithm
Figure 5: Typical LSFR plot

Figure 6: SNF plot for Prime Base, Prime Moduli PRN Generator with a base of 7 and modulus of 61

8 MATLAB Implementation

```matlab
% Initialize variables
% Initialize the first n prime numbers
primeNumbers = primes(7920);
% Initialize x
x = [1 primeNumbers];
% Create z array
z = zeros(1,size(primeNumbers,2));
% Choose a delta
delta = 1;
% Initialize counter variable
k = 1;
% Initialize loop size
l = size(primeNumbers,2) - delta;

for i = 1:l
    % Reset disregard variable
    disregard = 0;
    % Reset x variable
    x(1) = x(i);
    % Create congruence series
    for j = 1:i-1
        z(j) = mod(x(j),
        primeNumbers(i));
        x(j+1) = z(j) * x(i);
        if (z(j) == 1)
            % Disregard if series is repeating
            disregard = 1;
        end
    end
    if (disregard == 0)
        % Convert series to binary
        B(k,:) = mod(z, 2);
        % Increase counter variable
        k = k + 1;
    end
end

% Determine Correlation
% Number of data points in correlation
length = size(primeNumbers,2);
% First array
a = B(677,1:length);
% Second array
b = B(678,1:length);
% Convolution of two arrays
cl = convolution(a, b, length);
% Convolution of one array with itself
c2 = convolution(a, a, length);

% Plot Correlation
plot(1:length, cl, 'g')
hold on
plot(1:length, c2, 'm')
legend('cross correlation', 'autocorrelation')
```

0 10 20 30 40 50 60 70
8
9
10
11
12
13
14
15
16

0 20 40 60 80 100 120 140
-40
-20
0
20
40
60
80