Queen Mary
University of London

MTH6115 Cryptography

Solutions 4

1 It is enough to find a primitive polynomial of degree 5 and look at the output sequence of the corresponding shift register. You have seen in the previous exercise sheet how systematically find such polynomials, but here I will do it by trial and error: pick a polynomial that looks irreducible (!) and check whether it is primitive. Let's try $1 + x + x^5$ and take 00001 as input. The periodic part of the output sequence is:

$$000010001100101011111$$

The period is 21, so the polynomial is not primitive; it is actually divisible by $1 + x + x^2$. (Does the sequence satisfy Golomb's postulates?)

I try again with $1 + x^2 + x^5$ and 00001:

$$000010010110011111000110111010 1$$

The period is 31. So the polynomial is primitive and the sequence satisfies Golomb's postulates.

2 I quote the following from Dr. John Bray:

"My strings were

$$\begin{aligned} s_1 &= 10011100110100111101000010110110010110 01, \\ s_2 &= 00011110110100100110110010100000101010110, \\ \text{and} \quad s_3 &= 10000010000000011011110000010111000011 11, \end{aligned}$$

where s_1 was me trying to write down a random sequence of 0s and 1s, and s_2 was a 'mechanically' produced 'random' sequence, produced by computer rather than coin toss. I also calculated s_3 as the bit addition of s_1 and s_2. This has just 16 ones and 24 zeros and looks strikingly non-random. (I chose s_1 before the computer chose s_2, by the way.) The out-of-phase autocorrelations are:

$$\begin{aligned} 7^2, 8^2, 9^2, 10^9, 11^{12}, 12^6, 13^6 &\text{ for } s_1, \\ 7^8, 8^8, 9^{10}, 10^{11}, 11^2 &\text{ for } s_2, \\ \text{and } 3^4, 4^4, 5^6, 6^{12}, 7^2, 8^5, 9^4, 10^2 &\text{ for } s_3, \end{aligned}$$

where exponentiation denotes the number of times each value comes up. So s_2 seems to perform a little better than s_1 for postulate (G3). The 0–1 balances are 19–21 for s_1 and 21–19 for s_2, so neither quite satisfies (G1), but are quite close. For true randomness we shouldn't expect perfect balances all of the time. Here are some probabilities. Of course, an m–n balance and an n–m balance have the same probabilities, so for example a 22–18 balance has probability $\approx 10.312\%$.

Balance	14–26	15–25	16–24	17–23	18–22	19–21	20–20
Probability (%)	2.111	3.658	5.716	8.070	10.312	11.940	12.537

The probability that the 0–1 distribution is 13–27, 27–13 or more extreme is about 3.848%. I'll leave you to wonder about how well, or not, (G2) is satisfied."

3 The first 14 bits of the message are 10000111110010 (the ASCII encryption of `Cr`) and the first 14 bits of the ciphertext are 00100110111001. Since we are working over $\mathbb{Z}_2$, we have $k = p + z$ (z is the ciphertext), and so the first 14 bits of the key are:

$$10000111110010 + 00100110111001 = 10100001001011.$$

We thus get a system of equations represented by the matrix equation (with a_i unknown, over $\mathbb{Z}_2$)

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \end{pmatrix}.$$

This system is easy to solve, and then (non-unique) solution is

$$(a_0, a_1, a_2, a_3, a_4, a_5, a_6) = (y, z, y + 1, z, 1, y, z),$$

where y and z are arbitrary elements of $\mathbb{Z}_2$. Pick an arbitrary solution, say $(0, 0, 1, 0, 1, 0, 0)$. This corresponds to the shift register with polynomial $x^2+x^4+x^7$. We now use the first 7 bits of the key that we know, that is 1010000, as an input for the shift register. By iterating, we generate

$$k = 10100001001011001111100011011101010000100101100111110001$$

(We stop the iteration as soon as the length of the key we are generating becomes equal to the length of the ciphertext, that is, 56.)

We find the binary code for the plaintext as follows:

$$\begin{aligned} z &= 00100110111001101011011011111010100010010001010100001010 0 \\ k &= 10100001001011001111100011011101010000100101100111110001 \\ \therefore p = z + k &= 10000111110010100100111000101000010100000111001111100101. \end{aligned}$$

The decimal codes for p are $67, 114, 73, 98, 66, 65, 103, 101$, and so the message was `CrIbBAge` (with capitals and lower case exactly like this).

[**Remark.** We had more than one solution for our system of equations above, of which we picked one arbitrarily. What if we picked another solution? This would give us another shift register. And a different shift register can generate a different key. But can it? In fact, in our case, it can not. One can show that if two n-bit shift registers, fed with the same binary n-tuple input, generate the same output in the first n iterations, then they will generate the same output forever. I leave the proof of this relatively easy result to you. One way of proving it is to use Cayley-Hamilton's theorem in linear algebra.]