Fast Correlation Attacks and Linear Codes

Lauri Tarkkala

November 25, 2004

Brief Recap: Stream ciphers

Let P be the set of plaintext symbols. Let K be the set of keystream symbols. Let C be the set of ciphertext symbols. Let $P = K = C$.

A synchronous stream cipher produces a cyclic keystream K^* given as input a constant length key k. Encryption is performed by adding the synchronous keystream symbol by symbol to the plaintext modulo $|K|$. Decryption is performed by adding the inverse of each keystream symbol to the ciphertext symbol by symbol modulo $|K|$.

Note that stream ciphers by their very nature are vulnerable to chosen ciphertext attacks. Analysis of stream ciphers therefore limits itself often to considering known plaintext attacks, e.g. the computation of k given a keystream sequence.

Brief Recap: Linear Feedback Shift Registers

A Linear Feedback Shift Register (LFSR) is an n-bit register. A set of bit positions are designated as "taps". Every clock cycle the register is shifted towards the most significant bit. The least significant bit is set to the sum of the tap registers modulo 2. The most significant bit is the output.

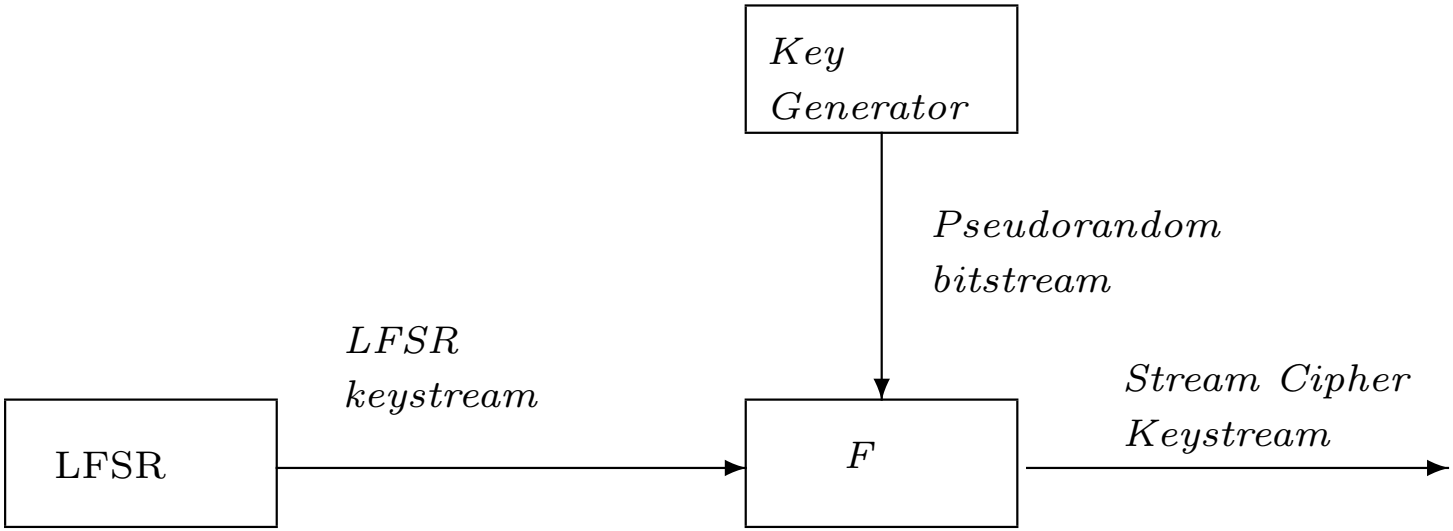
A LFSR is often described using a "feedback polynomial" $g(x) = 1 + \sum_{i=1}^{n-1} g_i x^i + a_n x^n$ where $g_i = 1$ if i corresponds to a "tap" and $g_i = 0$ otherwise. If the polynomial is irreducible and primitive then the LFSR cycle length is $2^n - 1$.

The amount of non-zero co-efficients in $g(x)$ is called the *weight* of the feedback polynomial.

A bitsequence output from an LFSR adheres to a set of linear equations over the bitstream. The output bits "are linear".

Brief Recap: LFSRs in stream ciphers

Stream ciphers often contain a least one LFSR as a primitive. One can in these cases consider the stream cipher to consist of a pseudorandom bit generator, the LFSR and a function F that combines the two component keystreams into a keystream.



Binary Symmetric Channel

A Binary Symmetric Channel (BSC) is a communication channel that with probability p flips a bit. The probability $1 - p$ is called the *cross-over probability*.

Error-correcting codes have been designed for reliable data transmission over these channels.

The cryptanalysis problem in this case can be understood as an attempt to correctly decode the "code" generated by the LFSR. The probability $1 - p$ is the "correlation" probability between the LFSR output and the F output.

Due to trade-offs in the resiliency and non-linearity of F it is assumed that $p < 0.5$ in practice. Exploiting this to compute the initial state of the LFSR is called a 'correlation attack'.

Convolutional Codes

A convolutional encoder when input a sequence of $B + 1$ input symbols outputs a code for the first input symbol in the sequence. The parameter B is called the "memory" of the encoder.

A convolutional code is linear. The relation between an an output symbol and the $B + 1$ input symbols is a linear equation.

A binary convolutional encoder for each input bit outputs c output bits. The ratio $R = 1/c$ is called the *rate* of the code.

Convolutional Codes

The structure of a binary convolutional code can be described using a set of binary linear equations. The codewords are linear combinations of $B+1$ different c-bit components that are labeled G_i.

If the plaintext was a N bits in length then the encoder could be written as the following $N \times cN$-matrix G and the plaintext as an N-element row vector.

$$G = \begin{pmatrix} G_0 & G_1 & ... & G_B & & & \\ & G_0 & G_1 & ... & G_B & & \\ & & G_0 & G_1 & ... & G_B & \\ & & & ... & ... & ... & \\ & & & & ... & ... & G_B \end{pmatrix}$$

Convolutional Codes

A binary convolutional code has 2^B different states.

The decoding operation is quite trivial, assuming the channel is error-free. If the channel is a binary symmetric channel with cross-over probability greater than 0 then a maximum-likelihood (ML) decoding algorithm is used.

The decoder receives as input a sequence of received bits $r = r_0^0 r_0^1 ... r^{c-1} r_1^0 .....$

The decoder now for each codeword $r_i = r_i^0 ... r_i^{c-1}$ attempts to compute the plaintext symbol y_i such that the conditional probability $p(r_i|y_i)$ is maximal when $y_i \in \{0, 1\}$.

The Viterbi algorithm decodes a binary convolutional code. The runtime grows expontentially in B.

Stream Ciphers and Convolutional Codes

The stream cipher is assumed to be of the form described earlier consisting of an LFSR, a pseudorandom bit-generator and a combination function F.

Let l be the length of the LFSR under analysis.

Let $g(x) = 1 + g_1 x^1 + ... + g^l x^l$ be the feedback polynomial. Let t be the number of taps and $t + 1$ be the weight.

Let $\mathcal{L}$ denote the set of LFSR sequences ($|\mathcal{L}| = 2^l$).

Truncate the LFSR sequences in $\mathcal{L}$ to length N. These sequences form a $[N, l]$ block code. Call this code $\mathcal{C}$. Assume $N >> l/(1 + p \log_2 p)$ s.t. a unique decoding is feasible.

Denote the keystream sequence by $z = (z_1, z_2, ..., z_N)$ as the output of the BSC F. Denote the output of the LFSR as $u = (u_1, u_2, ..., u_N)$.

Fast Correlation Attacks

If the feedback polynomial has low weight, then *fast correlation attacks* may be possible. This is performed by writing out sets of linear "parity check" equations that have only a few binary variables and then using these to decode the code.

Write out the equations for LFSR involving output index n, e.g. $u_n = g_1 u_{n-1} + g_2 u_{n-2} + ... + g_{n-l} u_{n-l}$. There are $t+1$ equations that contain u_n as a variable.

Note that $g(x)^j = g(x^j)$ when $j = 2^k$. Use this relation to create new parity check equations untill the degree of $g(x)^{2^k}$ is greater than N. The above relation guarantees that each polynomial has only weight $t+1$. This creates again $t+1$ equations involving u_n for each value of k when shifting $g(x)^{2^k}$.

We now have approx $\log_2(N/2l)(t+1)$ equations. Assume these equations hold for any bit in u. Decode z.

Fast Correlation Attacks

The decoding is done using a memoryless decoder.

One algorithm ("A") attempts to maximize
$p^* = P(u_n = z_n|\text{h equations holds})$.

Another algorithm ("B") iteratively flips bits in z_n untill for a sufficient amount of bits p^* exceeds a set treshold.

Simulation results by Johansson and Jönsson.

N/l	Algorithm B	Algorithm A
10^3	0.092	0.096
10^4	0.104	0.122

Fast Correlation Attack using Convolutional Codes

Attack proposed by Thomas Johansson and Fredrik Jönsson.

This attack improves the decoding process by adding a memory of the B previous bits to the decoder. The attack is based on the observation that a LFSR creates a very low-rate convolutional code and the decoder used is the Viterbi algorithm. The memory required is 10 states and each codeword is assumed to be 4 bits.

The N-bit code output by a l-bit LFSR can be written as the product of $1 \times l$ vector and a $l \times N$ generator matrix called G_{LFSR}. Then $u = u_0 G_L FSR$ where u_0 is the LFSR initial state.

$$G_{LFSR} = \begin{pmatrix} I_{B+1} & Z_{B+1} \\ 0_{l-B-1} & Z_{l-B-1} \end{pmatrix}$$

I_x denotes an $x \times x$ identity matrix.

Fast Correlation Attack using Convolutional Codes

The code generated by the LFSR is considered to be systematic convolutional code.

Parity check equations are generated for $u_n = u_{B+1}$ by considering the bits NOT in the initial state. Find linear combinations of columns of Z_{l-B-1} that add to the all zero column vectors (e.g. $u_{j_{11}} = u_0 * [...]$ and $u_{j_{21}} = u_0 * [....]$) s.t. the value of u_n differs in these equations. Sum these two equations to generate a parity check equation.

This technique finds parity check equations with weight $t = 2$.

Write these equations as $u_n = \sum_{i=1}^{B} c_{i1} u_{n-1} + u_{j_{1l}} + u_{j_{2l}}$ where l is the index of equation.

Fast Correlation Attack using Convolutional Codes

Based on the m equations $u_n = \sum_{i=1}^{B} c_{i1} u_{n-1} + u_{j_{1l}} + u_{j_{2l}}$ construct a convolutional code. Write the parity equations so that they hold when a bitstream is encoded using the constructed encoder.

$$\begin{pmatrix} G_0 \\ G_1 \\ ... \\ G_B \end{pmatrix} = \begin{pmatrix} 1 & 1 & ... & 1 \\ 0 & c_{11} & ... & c_{1m} \\ ... & ... & ... & ... \\ 0 & c_{B1} & ... & c_{Bm} \end{pmatrix}$$

$$G = \begin{pmatrix} ... & ... & ... & ... & & \\ & G_0 & G_1 & ... & G_B & \\ & & G_0 & G_1 & ... & G_B \\ & & & ... & ... & ... & ... \end{pmatrix}$$

Fast Correlation Attack using Convolutional Codes

If a codeword $v_n^i = u_n$ (non-parity bit) then $P(v_n^i = z_n) = 1 - p$. If a codeword $v_n^i = u_{j_{1i}} + u_{j_{21}}$ then $P(v_n^i = z_{j_{1i}} + z_{j_{2i}}) = (1-p)^2 + p^2$.

Let $r = r_n^0 r_n^1 ... r_n^m r_{n+1}^0 ... r_{n+1}^m ...$ be the bitsequence received by the decoder and let $r_n^0 = z_n$ and $r_n^i = z_{j_{1i}} + z_{j_{21}}$, $1 \leq i \leq m$.

Now we only have to decode l consecutive codewords correctly to be able to backtrack to the initial state. This is performed using the Viterbi algorithm.

Simulation Results

Maximum correlation probability p for a realistic probability for a succesful attack according to simulations by Johansson and Jönsson. Simulation used a 40 bit LFSR with a weight 17 feedback polynomial.

N/l	B = 13	B = 14	B = 15
10^3	0.19	0.22	0.26
10^4	0.37	0.39	0.40