A Fast Correlation Attack Implementation

AZHAR DESAI
University of Cape Town
Supervisors
DR ANNE KAYEM - Department of Computer Science, University of Cape Town
DR CHRISTINE SWART - Department of Mathematics, University of Cape Town

1. INTRODUCTION

This Honours project is centered on an implementation of a fast correlation attack. This is a cryptographic attack that works on certain stream ciphers. Strengthening the design of stream ciphers motivates studying such attacks against them. Attacks have been made that exploit correlations to attack streams ciphers currently in use in GSM and Bluetooth. The rest of this introduction explains broadly what a fast correlation attack is.

1.1 Stream Ciphers

A stream cipher aims to allow two parties to exchange information in secret, given that both know some secret key, a bit string, which only they know. Fast correlation attacks work on certain synchronous stream ciphers. This stream cipher works by first having its keystream generator expand the secret key into a keystream. This is done independently of the data to be encrypted. The keystream is then combined with the information to be sent, producing ciphertext. On receiving the message, the other party expands the key the same way into the keystream and uses this to decrypt the ciphertext, retrieving the original plaintext. In principle, it should be infeasible to retrieve the plaintext without knowledge of the secret key.

1.2 Fast Correlation Attack

Fast Correlation Attacks exploit a weakness of some keystream generators. The output of some keystream generators are correlated with the predictable sequence produced by some component of the system such as a Linear Feedback Shift Register (LFSR). The details are made precise in section 2. By determining the initial state of the LFSR, information can be obtained about the shared key in certain stream ciphers.

By exploiting weaknesses in a cipher's design, parties other than the communicating two, may gain knowledge of plaintext or the shared key. A possible motivation for this is to decrypt intercepted messages. Another is that knowledge of attacks help to design more secure ciphers.

1.3 Proposal Overview

Fast correlation attacks are explained in Section 2 and we build on this to discuss related work in Section 3. Section 4 outlines the methodology that will be used to tackle the problem and expected outcomes are discussed in Section 5. Finally in Section 7, concluding remarks are offered.

2. THE FAST CORRELATION ATTACK PROBLEM

To introduce the problem the definition of a Linear Feedback Shift Register is required.

2.1 Linear Feedback Shift Registers

An LFSR is a fixed size register of bits which produces a deterministic sequence of bits. Each step of the register generates an update bit. This is the sum modulo 2 (or XOR) of bits from some fixed positions in the register. To complete the step, the whole register is shifted up one bit. The bit that gets shifted out the register is the next bit of the LFSR sequence. The new position that opens up is filled with update bit. The state of the LFSR is the contents of the register. Each summand position is called a tap of the LFSR. The LFSR in figure 2.1 has 2 taps. The contents of the register is shifted to the right to make the contents of the register $\sqcup 011$ where $\sqcup$ is the space that opens up. The output bit is the 0 that was shifted out of the register. The update bit, $1 = 1 + 0$, is inserted in place of $\sqcup$, making the next state 1011.

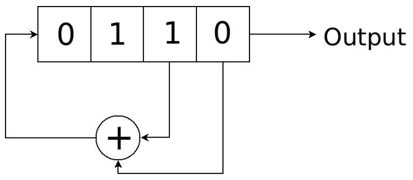

Fig. 1. An LFSR of length 4 in state 0110.

The linear relationship between the bits is important because it allows the initial state to be recovered with sufficient bits in the output sequence.

2.2 An Example Correlated LFSR

An example keystream generator is the combination generator shown in figure 2.2. Each bit of the keystream is f applied to the corresponding bits of each LFSR. Parts of the symmetric key are commonly used as the initial state of the constituent LFSR. The function f is non-linear and computes each bit of the keystream independently from all the others.

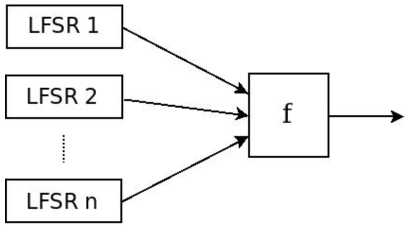

Fig. 2. A keystream Generator which combines the output of several LFSR with a nonlinear binary function f.

If f is poorly chosen, it could happen that the keystream is correlated to output sequence of one of the constituent LFSR. A cryptanalyst, who has observed the keystream, can use this to recover the initial state of the correlated LFSR, in far less time than an exhaustive search over the initial states of all the LFSR together. The aim of the project is to implement a system for recovering this initial state.

2.3 Recovering the Initial State of Correlated LFSR Problem

This project aims to create a system that recovers the initial state of the correlated LFSR from a given keystream sequence. It is assumed that the taps and length of the LFSR are known. (This is not a great restriction, as an exhaustive search over all likely LFSR taps is possible.)

A successful recovery of the initial state is required under the widest possible range of the following parameters:

- the length of the LFSR,
- the number of LFSR taps, and
- the probability of the correlation.

The last requirement is correlation attack must take far less time than an exhaustive search over all the initial states of the correlated LFSR. These parameters determine required length of the keystream sequence.

2.4 Importance of the Technique

Apart from the obvious use of this to recover the keys used in certain stream ciphers, this also informs the design of stream ciphers. A proposed keystream generator must avoid a correlation of the keystream with an LFSR sequence.

There have been applications of fast correlation attacks on stream ciphers in current use. A fast correlation attack can be made on E0, the stream cipher in the Bluetooth protocol, as done in [Lu and Vaudenay 2004]. A technique based on correlation attacks is used in [Ekdahl and Johansson 2003] on A5/1, a version of the stream cipher used in GSM.

3. RELATED WORK

The implementation and probability results will mainly follow the iterative decoding method of Meier and Staffelbach [Meier and Staffelbach 1989; 1988]. Given a keystream that is correlated an LFSR output sequence, the method expects that a certain amount of digits in the keystream are correct. A digit in the keystream sequence is correct if it is the same as the digit in the corresponding position in the LFSR sequence. In each iteration of the method, the probabilities of each keystream bit being correct are estimated. If the probability of a bit being correct is below a certain threshold, then that bit is flipped. These iterations converge on a sequence of digits that can be produced by the correlated LFSR. This is used to obtain the a initial state of the correlated LFSR.

Should time allow it, a technique for finding all parity check equations of certain weight may be used from Canteaut and Trabbia's algorithm in [Canteaut and Trabbia 2000]. These equations are used to compute the probability that a given digit in the keystream sequence is the same as the corresponding digit in the correlated LFSR output sequence. This technique extends the method for generating parity check equations that Meier and Staffelbach use. A discussion of this technique is beyond the scope of this document.

Data will be used from Mihaljevic and Golic's comparison between the method of Meier and Staffelbach and their own in [Mihaljevic and Golic 1990] to test for correctness for parts of this project's implementation.

4. METHODOLOGY

This section considers details concerning the implementation of the initial state recovering algorithm. (Explain parity check) First, the requirements of the statistical model are detailed. Following that is the implementation details, and lastly, the tests are described.

4.1 Mathematical Foundations

The implementation of the attack requires several mathematical results. Meier and Staffelbach derive these in [Meier and Staffelbach 1989; 1988]. In particular, the following needs to be derived:

- the number of parity check equations obtained,
- the conditional probability that a digit in the observed sequence matches the corresponding LFSR digit, given how many parity check equations it satisfies, and
- the probability threshold to determine whether a given digit is correct, or should be flipped.

Once these details are worked out, the implementation can be started.

4.2 Implementation

The implementation will be done in Python. The code will be written to run on Linux (32 bit architecture). Common math functions needed will be used from SciPy library. The bit manipulation will be done using a third party python library called bitstring.

Should time allow, some of the slower code could be rewritten as C modules to be called from Python. This may allow larger LFSR lengths to tested more easily. However, this may not improve the recovery for a larger range of probabilities of the correlation, or an increased number of taps.

Graph plotting of the results of the recovery will be done using Matplotlib.

4.3 Testing and Analysis

The two main products of the project are the practical implementation, and the derivation of the attack. The derivations will be looked over by supervisors, and should agree with [Meier and Staffelbach 1989; 1988].

The implementation will be tested to see if it can recover the initial states of different correlated LFSR. The correlated LFSR will have a few number of taps (less than 5) and have lengths of up to 1000 bits where the probability of the correlation is greater than 0.5. To test a representative set of test values for the correlated LFSR will be chosen from these ranges. This will be used as the third LFSR in the Geffe keystream generator. The output of the Geffe generator is known to be correlated to the output of its third LFSR.

Should time allow further testing will be for larger LFSR lengths and more taps to see if it is possible to still recover the sequence. In this case more testing data can be generated by perturbing LFSR sequences. Some of the testing data will be sourced from the papers mentioned in the related work section.

5. ANTICIPATED OUTCOMES

The two main anticipated outcomes of this Honours project are the derivation of the results needed for the attack and the implementation of the attack.

The explanation will be included in the report. It should be a self-contained derivation of the results needed that can be followed by another honours student.

The implementation will be judged successful if it can successfully recover the initial state of various correlated LFSR under the expected testing conditions described in the previous section.

The expected challenges are in understanding the attack, and a suitably powerful implementation that manages to recover the initial state. Once the correlated LFSR length and the number of taps gets too large, it may quickly become infeasible for the attack to work.

Once the project is completed, it could be used as part of a larger system to recover keys of vulnerable stream ciphers.

6. PROJECT PLAN

This section outlines a plan of what must be done for the project and how it is intended to be achieved.

6.1 Deliverables

The following need to be done for this Honours project:

- write up of the mathematical foundations of the attack,
- implementation of the fast correlation attack,
- testing of the implementation,
- report detailing the entire project,
- poster summarising the project,
- reflection paper on the work on the project,
- web page presenting the project.

6.2 Risks

There are risks affecting the two main parts of the project. The first risk is I may not be able to understand the arguments presented in the relevant papers. This is offset by consulting supervisors and background reading to get the relevant context. This needs to be done quickly enough to avoid delaying the implementation part of the project, and hence the final deadlines.

There are two risks involved in the implementation. One is that debugging of the entire application using the probabilities output at the end will be very difficult. To keep that debugging to a minimum there will be unit testing done on the probability formulae, LFSR implementation.

Another is that the complexity of the algorithm may make testing large LFSR lengths very time consuming to the point of being impractical. The theoretical analysis will help predict easier values to begin testing on.

A backup of the project will be maintained, version controlled on an off-site server.

6.3 Resources

This project only requires easily available resources: an average desktop computer, internet access and the server the author uses at his host for backup.

Table I. Milestones and their Due Dates in 2011

Milestone	Due Date
Theoretical Foundations Completed	4 August
Initial Feasibility Demonstration	29 July
First Prototype Implementation	19 September
Final Prototype Implementation and Testing Completed	28 September
Final Implementation, Testing and Write Up	3 October
Written Chapters on Implementation and Testing	3 October
Report Final Draft and Mark Weighting Decided	24 October
Final Submission of Project and Report	31 October
Poster Due	3 November
Web page due	7 November
Reflection Paper	11 November
Final Project Presentation	17,18 November

6.4 Timeline and Milestones

The milestones and their due dates are summarised in table 6.4. The project is intended to stick to these deadlines to ensure the project meets the final deadline.

The tasks required to meet these milestones have been broken down into smaller tasks, and scheduled in the Gantt chart in figures at the end of this document. Roughly four weeks have been scheduled for working through the mathematical foundations, and a bit more than five for the main implementation. Most of the mathematical foundation work will be completed before the second semester begins. The implementation will be started before all the foundation work is done, to have code ready for the feasibility demonstration. Two weeks after testing has been dedicated to writing the report. However, it is intended that some writing will be done along the way.

7. CONCLUSION

This concludes the proposal for an implementation of a fast correlation attack. Given that an observed keystream sequence is correlated to an LFSR, the proposed system is intended to recover the initial state of the LFSR under the conditions described in Section 4.3. The limitations of the conditions involve limits on the length of the LFSR, the number of taps, the probability of the correlation and the length of the observed sequence. The two main products of the project are a detailed account of the mathematical foundations of, and an implementation of the attack.

Details of the implementation were described, as well as methods for testing its ability to successfully recover the initial states. Sources of test data, and generating test data were also discussed.

This proposal detailed the requirements for the project, which include a project report, poster and website. Lastly a plan was outlined, along with estimated times, for completing the project in the Honours year schedule. Possible risks are considered along with potential mitigations.

REFERENCES

CANTEAUT, A. AND TRABBIA, M. 2000. *Improved Fast Correlation Attacks Using Parity-Check Equations of Weight 4 and 5*. Lecture Notes in Computer Science, vol. 1807. Springer Berlin / Heidelberg, 573–588.

EKDAHL, P. AND JOHANSSON, T. 2003. Another attack on A5/1. *Information Theory, IEEE Transactions on 49,* 1, 284–289.

LU, Y. AND VAUDENAY, S. 2004. *Faster Correlation Attack on Bluetooth Keystream Generator E0*. Lecture Notes in Computer Science, vol. 3152. Springer Berlin / Heidelberg, 35–49.

MEIER, W. AND STAFFELBACH, O. 1988. Fast Correlation Attacks on Stream Ciphers. *Advances in Cryptology EUROCRYPT 88 Lecture Notes in Computer Science 330*, 301–314.

MEIER, W. AND STAFFELBACH, O. 1989. Fast Correlation Attacks on Certain Stream Ciphers. *J. Cryptology 1,* 3, 159–176.

MIHALJEVIC, M. AND GOLIC, J. 1990. *A fast iterative algorithm for a shift register initial state reconstruction given the noisy output sequence*. Lecture Notes in Computer Science, vol. 453. Springer Berlin / Heidelberg, 165–175.

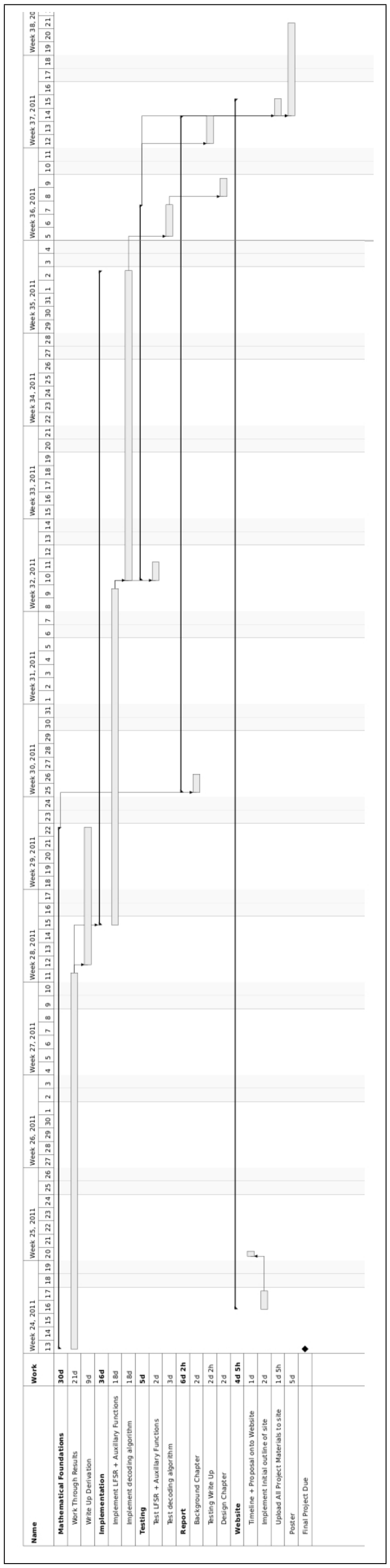

Fig. 3. Timeline of Project Tasks