A Faster Attack on LFSR Recurrences

An Alternative to Berlekamp-Massey for Complete Ciphertexts

Timothy Tarter, Arsenii Herasymov, & Garrett Priebracha

Madison Institute for Mathematical Finance
James Madison University

Joint Mathematics Meetings
January 5, 2026

Overview

Consider the Sequence

Consider the Sequence

$$1001101001000010101110110001111|1001101001000010101110110\underline{\mathbf{0}}01\underline{\mathbf{1}}1\underline{\mathbf{1}}$$

Consider the Sequence

$$1001101001000010101110110001111|1001101001000010101110110\underline{\mathbf{0}}01\underline{\mathbf{1}}1\underline{\mathbf{1}}$$

- We claim that we can fully describe it using the recurrence polynomial (from right to left).

Consider the Sequence

1001101001000010101110110001111|1001101001000010101110110**0**01**1**1**1**

- We claim that we can fully describe it using the recurrence polynomial (from right to left).

$$x_{n+5} = x_n + x_{n+2} \; [\text{mod } 2]$$

Consider the Sequence

$$1001101001000010101110110001111|1001101001000010101110110\underline{\mathbf{0}}01\underline{\mathbf{1}}1\underline{\mathbf{1}}$$

- We claim that we can fully describe it using the recurrence polynomial (from right to left).

$$x_{n+5} = x_n + x_{n+2} \text{ [mod 2]}$$

- As it turns out, it repeats itself after 31 terms.

Question: What is the length of the recurrence?

Question: What is the length of the recurrence?

- Such recurrences are deterministic and are entirely defined by the initial seed and the recurrence polynomial.

Question: What is the length of the recurrence?

- Such recurrences are deterministic and are entirely defined by the initial seed and the recurrence polynomial.

$$\textbf{input}(seed, c(x)) \implies \textbf{recurrence}$$

Question: What is the length of the recurrence?

- Such recurrences are deterministic and are entirely defined by the initial seed and the recurrence polynomial.

$$\textbf{input}(seed, c(x)) \implies \textbf{recurrence}$$

$$(01111, x_{n+5} \equiv x_n + x_{n+2}) \implies 1001101001000010101110110001111$$

Setup

Setup

- Suppose we know the initial segment 011010111100.

Setup

- Suppose we know the initial segment 011010111100.
- How do we determine the polynomial coefficients $\{0, 1\}$?

Setup

- Suppose we know the initial segment 011010111100.
- How do we determine the polynomial coefficients $\{0, 1\}$?
- We start by guessing the length of the recurrence polynomial.

Setup

- Suppose we know the initial segment 011010111100.
- How do we determine the polynomial coefficients $\{0, 1\}$?
- We start by guessing the length of the recurrence polynomial.

$$x_{n+2} \equiv c_0 x_n + c_1 x_{n+1}. \tag{1}$$

Setup

- Suppose we know the initial segment 011010111100.
- How do we determine the polynomial coefficients $\{0, 1\}$?
- We start by guessing the length of the recurrence polynomial.

$$x_{n+2} \equiv c_0 x_n + c_1 x_{n+1}. \tag{1}$$

- Start with two cases $n = 1$ and $n = 2$, then $x_1 = 0, x_2 = 1, x_3 = 1, x_4 = 0$.

System of Linear Equations

System of Linear Equations

$$1 \equiv c_0 * 0 + c_1 * 1 \text{ [for the n=1 case]}$$
$$0 \equiv c_0 * 1 + c_1 * 1 \text{ [for the n=2 case]}$$

System of Linear Equations

$$1 \equiv c_0 * 0 + c_1 * 1 \text{ [for the n=1 case]}$$
$$0 \equiv c_0 * 1 + c_1 * 1 \text{ [for the n=2 case]}$$

$$\implies \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}. \tag{2}$$

System of Linear Equations

$$1 \equiv c_0 * 0 + c_1 * 1 \text{ [for the n=1 case]}$$
$$0 \equiv c_0 * 1 + c_1 * 1 \text{ [for the n=2 case]}$$

$$\implies \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}. \tag{2}$$

The solution to the system is $c_0 = 1$, $c_1 = 1$, thus we assume that the recurrence is $x_{n+2} \equiv x_n + x_{n+1}$, which **doesn't** fit the sixth digit, so make a new guess x_{n+3}.

Berlekamp-Massey

- The Gaussian Elimination Approach is computationally expensive.

[1] Wade Trappe and Lawrence C. Washington, *Introduction to Cryptography with Coding Theory*, 2nd ed., Pearson, 2006.

Berlekamp-Massey

- The Gaussian Elimination Approach is computationally expensive.
- Berlekamp-Massey Algorithms utilizes properties of determinant over $\mathbb{Z}_2$

[1] Wade Trappe and Lawrence C. Washington, *Introduction to Cryptography with Coding Theory*, 2nd ed., Pearson, 2006.

Berlekamp-Massey

- The Gaussian Elimination Approach is computationally expensive.
- Berlekamp-Massey Algorithms utilizes properties of determinant over $\mathbb{Z}_2$
- Determinant $\in \{0, 1\}$. Zero for a failed guess and one for a successful guess.

[1] Wade Trappe and Lawrence C. Washington, *Introduction to Cryptography with Coding Theory*, 2nd ed., Pearson, 2006.

Berlekamp-Massey

- The Gaussian Elimination Approach is computationally expensive.
- Berlekamp-Massey Algorithms utilizes properties of determinant over $\mathbb{Z}_2$
- Determinant $\in \{0, 1\}$. Zero for a failed guess and one for a successful guess.

Let N be the length of the shortest recurrence, and M_N be the matrix containing the systems of equations for $[c_1, c_2, \ldots, c_n]^\top$, then $det(M_N) \equiv 1$ [mod 2], and for any $n > N$, $det(M_n) \equiv 0$ [mod 2][1].

[1] Wade Trappe and Lawrence C. Washington, *Introduction to Cryptography with Coding Theory*, 2nd ed., Pearson, 2006.

Theorem

Berlekamp-Massey Algorithm:

Let ε be acceptable error rate, and n be the length of the recurrence. Then for $2 \leq i \leq n$.

Theorem

Berlekamp-Massey Algorithm:

Let ε be acceptable error rate, and n be the length of the recurrence. Then for $2 \leq i \leq n$.

1. *Construct a coefficient matrix of size i*

Theorem

Berlekamp-Massey Algorithm:

Let ε be acceptable error rate, and n be the length of the recurrence. Then for $2 \leq i \leq n$.

1. *Construct a coefficient matrix of size i*
2. *Compute its determinant.*

Theorem

Berlekamp-Massey Algorithm:

Let ε be acceptable error rate, and n be the length of the recurrence. Then for $2 \leq i \leq n$.

1. *Construct a coefficient matrix of size i*
2. *Compute its determinant.*

$$1001010000\underline{\boldsymbol{1}}0000000000000000000 \tag{3}$$

Theorem

Berlekamp-Massey Algorithm:

Let ε be acceptable error rate, and n be the length of the recurrence. Then for $2 \leq i \leq n$.

1. *Construct a coefficient matrix of size i*
2. *Compute its determinant.*

$$1001010000\underline{\boldsymbol{\mathit{1}}}000000000000000000 \tag{3}$$

The first matrix with a determinant of one, with ε many zero determinants after the one, ***is*** *the coefficient matrix for the system of equations, equivalent to the recurrence polynomial.*

Main Results

Theorem (Tarter-Herasymov-Priebracha Algorithm)

*Given the **full** sequence of bits,*

1. *Compute the length of the string and factor it into a list of both prime and composite factors.*

Theorem (Tarter-Herasymov-Priebracha Algorithm)

*Given the **full** sequence of bits,*

1. *Compute the length of the string and factor it into a list of both prime and composite factors.*
2. *Starting with the largest proper factor k, chop off the last k digits of the sequence.*

Theorem (Tarter-Herasymov-Priebracha Algorithm)

*Given the **full** sequence of bits,*

1. *Compute the length of the string and factor it into a list of both prime and composite factors.*
2. *Starting with the largest proper factor k, chop off the last k digits of the sequence.*
3. *Since $k \mid n$, $n = q * k$. Concatenate the last k digits of the sequence q times.*

Theorem (Tarter-Herasymov-Priebracha Algorithm)

*Given the **full** sequence of bits,*

1. *Compute the length of the string and factor it into a list of both prime and composite factors.*
2. *Starting with the largest proper factor k, chop off the last k digits of the sequence.*
3. *Since $k \mid n$, $n = q * k$. Concatenate the last k digits of the sequence q times.*
4. *Check to see if this sequence matches the original. If it doesn't, continue to the next smallest factor.*

Theorem (Tarter-Herasymov-Priebracha Algorithm)

*Given the **full** sequence of bits,*

1. *Compute the length of the string and factor it into a list of both prime and composite factors.*
2. *Starting with the largest proper factor k, chop off the last k digits of the sequence.*
3. *Since $k \mid n$, $n = q * k$. Concatenate the last k digits of the sequence q times.*
4. *Check to see if this sequence matches the original. If it doesn't, continue to the next smallest factor.*

Once the bit recurrence string is obtained, recover the recurrence polynomial with BM.

Proof

Proof of THP

- Let β be a ciphertext with an unknown, minimal recurrence γ, and let $n = |\beta|$.

Proof of THP

- Let β be a ciphertext with an unknown, minimal recurrence γ, and let $n = |\beta|$.
- Let $\theta = \{$last 'k' digits of n $\mid |k||n$, $k \neq 1, n$, and $\bigoplus_{i=1}^{\frac{n}{|k|}}$ last k digits of $\beta = \beta\}$.

Proof of THP

- Let β be a ciphertext with an unknown, minimal recurrence γ, and let $n = |\beta|$.
- Let $\theta = \{$last 'k' digits of n $\mid |k| | n$, $k \neq 1, n$, and $\bigoplus_{i=1}^{\frac{n}{|k|}}$ last k digits of $\beta = \beta\}$.
- We want to show that $\gamma \in \theta$ and we want to show that there does not exist something smaller than γ in θ.

Proof of THP

- Let β be a ciphertext with an unknown, minimal recurrence γ, and let $n = |\beta|$.
- Let $\theta = \{$last 'k' digits of n $\mid |k| | n$, $k \neq 1, n$, and $\bigoplus_{i=1}^{\frac{n}{|k|}}$ last k digits of $\beta = \beta\}$.
- We want to show that $\gamma \in \theta$ and we want to show that there does not exist something smaller than γ in θ.
- Since $|\gamma|$ divides n by definition, our algorithm will identify it as a potential k. Since it is also a recurrence, it can be reconstructed into β by concatenating it with itself $\frac{n}{|\gamma|}$ times, and thus is contained in θ.

Proof of THP

- Let β be a ciphertext with an unknown, minimal recurrence γ, and let $n = |\beta|$.
- Let $\theta = \{\text{last 'k' digits of n} \mid |k| | n,\ k \neq 1, n, \text{ and } \bigoplus_{i=1}^{\frac{n}{|k|}} \text{last k digits of } \beta = \beta\}$.
- We want to show that $\gamma \in \theta$ and we want to show that there does not exist something smaller than γ in θ.
- Since $|\gamma|$ divides n by definition, our algorithm will identify it as a potential k. Since it is also a recurrence, it can be reconstructed into β by concatenating it with itself $\frac{n}{|\gamma|}$ times, and thus is contained in θ.
- To show that there is not a smaller element in θ than γ, we assume there exists $\eta \in \theta$ with $|\eta| < |\gamma|$. But if $\eta \in \theta$, then η can be reconstructed as a recurrence of β by concatenating itself $\frac{n}{|\eta|}$ times, and is therefore a smaller recurrence than γ, which we assumed to be the minimal recurrence. Therefore, there does not exist such an η, and $\min \theta = \gamma$.

Computational Complexity

Time Complexity

Algorithm	Complexity
THP	$O(n + log(n))$
BM	$O(n^2)$

Time Complexity

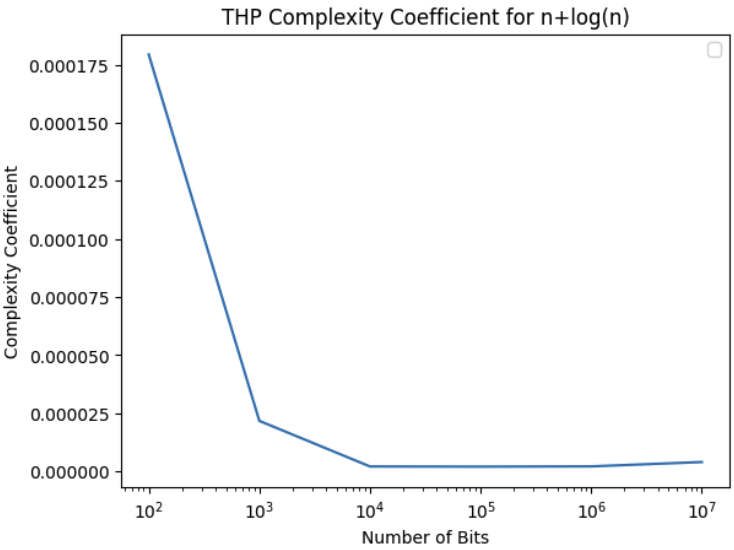
Algorithm	Complexity
THP	$O(n + log(n))$
BM	$O(n^2)$

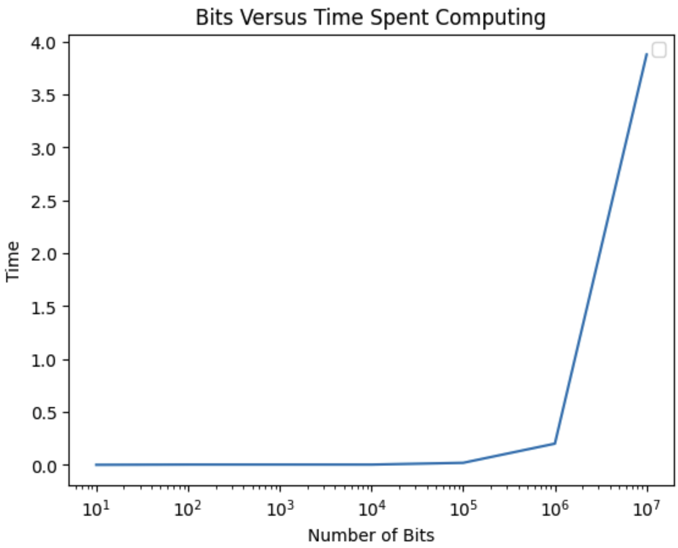
1. THP requires a full string whereas Berlekamp-Massey works for partial string as well.

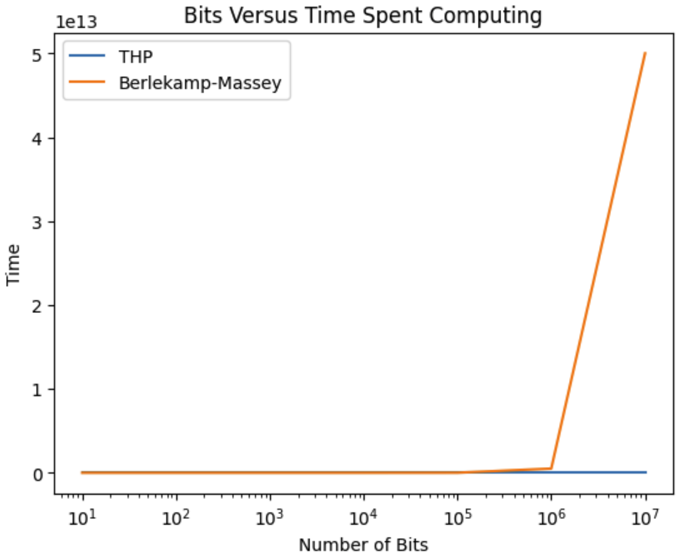
Time Complexity

Algorithm	Complexity
THP	$O(n + log(n))$
BM	$O(n^2)$

1. THP requires a full string whereas Berlekamp-Massey works for partial string as well.
2. Our time complexity prevails for large sequences, unfeasable for Berlekamp-Massey.


THP Complexity Coefficient for n+log(n)
Complexity Coefficient
0.000175
0.000150
0.000125
0.000100
0.000075
0.000050
0.000025
0.000000
10^2
10^3
10^4
10^5
10^6
10^7
Number of Bits


Bits Versus Time Spent Computing
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Time
10^1
10^2
10^3
10^4
10^5
10^6
10^7
Number of Bits


Bits Versus Time Spent Computing
1e13
5
4
3
2
1
0
Time
THP
Berlekamp-Massey
10^1
10^2
10^3
10^4
10^5
10^6
10^7
Number of Bits

GitHub Repository

W. Trappe and L. C. Washington, *Introduction to Cryptography with Coding Theory*, 2nd ed., Pearson Education, 2006.

M. Tomlinson, C. J. Tjhai, M. A. Ambroze, M. Ahmed, M. Jibril, *Error-Correction Coding and Decoding: Bounds, Codes, Decoders, Analysis and Applications*, Springer International Publishing, 2017.

S. Nikolaev, *Advanced Berlekamp–Massey Method as a Basis for Finding Periodicities in Bitstreams*, Cybernetics and Systems Analysis, 61, pp. 3–10, 2025

S. D. Galbraith, *Cryptography and Coding: 11th IMA International Conference*, Cirencester, UK, December 18–20, 2007, Proceedings, Springer, pp. 220-232, 2007

A. Kreczmar and G. Mirkowska, *Mathematical Foundations of Computer Science 1989*, Lecture Notes in Computer Science, vol. 379, Springer, pp. 431–444, 1989.

L. J. Guibas, *Periods in Strings*, Journal Combinatorial Theory, Series A, vol. 30, Issue 1, January 1981, pp. 19-42.

K. Yu, *On binary recurrence sequences*, Indagationes Mathematicae, vol. 6, Issue 3, 1995, pp. 341-354.

Y. Wu, High-Speed *LFSR Decoder Architectures for BCH and GII Codes*, IEEE Journal on Selected Areas in Information Theory, vol. 4, pp. 331-350, 2023