UC SANTA CRUZ

Math 134: Cryptography

Lecture 16: Stream ciphers

Eoin Mackall

February 14th, 2025

University of California, Santa Cruz

Table of contents

Stream ciphers

Stream ciphers

In a stream cipher, bits of a plaintext message are combined with (pseudo)randomly generated bits in order to create a ciphertext message.

Stream ciphers

In a stream cipher, bits of a plaintext message are combined with (pseudo)randomly generated bits in order to create a ciphertext message.

If the bit string used to combine with the plaintext was truly randomly generated, then this string of bits would form the key for encryption and decryption.

Stream ciphers

In a stream cipher, bits of a plaintext message are combined with (pseudo)randomly generated bits in order to create a ciphertext message.

If the bit string used to combine with the plaintext was truly randomly generated, then this string of bits would form the key for encryption and decryption.

If the bit string used to combine with the plaintext is pseduorandomly generated, then the information used to generate this string forms the key.

Stream ciphers

Example

The plaintext message `Have a good day` can be converted to the binary string

```
01001000 01100001 01110110 01100101 00100000
01100001 00100000 01100111 01101111 01101111
01100100 00100000 01100100 01100001 01111001.
```

Stream ciphers

Example

The plaintext message `Have a good day` can be converted to the binary string

```
01001000 01100001 01110110 01100101 00100000
01100001 00100000 01100111 01101111 01101111
01100100 00100000 01100100 01100001 01111001.
```

We can combine this with the random bit string

```
01101000 11111101 10010111 01001110 11100010
00010110 11111101 10010010 10100011 00110111
01001001 00000010 01011110 01001000 00110100
```

adding each entry modulo 2.

Stream ciphers

Example

Taking the first blocks of both, we want to combine the message `01001000` with the key `01101000`.

Stream ciphers

Example

Taking the first blocks of both, we want to combine the message `01001000` with the key `01101000`.

We do this by

```
01001000
01101000
```

Stream ciphers

Example

Taking the first blocks of both, we want to combine the message `01001000` with the key `01101000`.

We do this by

```
01001000
01101000
    0
```

Stream ciphers

Example

Taking the first blocks of both, we want to combine the message `01001000` with the key `01101000`.

We do this by

```
01001000
01101000
00
```

Stream ciphers

Example

Taking the first blocks of both, we want to combine the message `01001000` with the key `01101000`.

We do this by

```
01001000
01101000
   001
```

Stream ciphers

Example

Taking the first blocks of both, we want to combine the message `01001000` with the key `01101000`.

We do this by

```
01001000
01101000
00100000
```

Stream ciphers

Example

Taking the first blocks of both, we want to combine the message `01001000` with the key `01101000`.

We do this by

```
01001000
01101000
00100000
```

The bit string `00100000` is then the corresponding ciphertext.

Stream ciphers

Example

Taking the first blocks of both, we want to combine the message `01001000` with the key `01101000`.

We do this by

```
01001000
01101000
00100000
```

The bit string `00100000` is then the corresponding ciphertext.

Remark

This is the same process as XOR for the binary strings.

Stream ciphers

Since addition of the same bit modulo 2 is always congruent with 0, decryption is gotten again by addition of the key (again).

Stream ciphers

Since addition of the same bit modulo 2 is always congruent with 0, decryption is gotten again by addition of the key (again).

Example

Using the same ciphertext and key as before:

$$\begin{matrix} 00100000 \\ 01101000 \end{matrix}$$

Stream ciphers

Since addition of the same bit modulo 2 is always congruent with 0, decryption is gotten again by addition of the key (again).

Example

Using the same ciphertext and key as before:

$$\begin{array}{r} 00100000 \\ 01101000 \\ \hline 0 \end{array}$$

Stream ciphers

Since addition of the same bit modulo 2 is always congruent with 0, decryption is gotten again by addition of the key (again).

Example

Using the same ciphertext and key as before:

$$\begin{array}{r} 00100000 \\ 01101000 \\ \hline 01 \end{array}$$

Stream ciphers

Since addition of the same bit modulo 2 is always congruent with 0, decryption is gotten again by addition of the key (again).

Example

Using the same ciphertext and key as before:

```
00100000
01101000
   010
```

Stream ciphers

Since addition of the same bit modulo 2 is always congruent with 0, decryption is gotten again by addition of the key (again).

Example

Using the same ciphertext and key as before:

```
00100000
01101000
01001000
```

Stream ciphers

Since addition of the same bit modulo 2 is always congruent with 0, decryption is gotten again by addition of the key (again).

Example

Using the same ciphertext and key as before:

```
00100000
01101000
01001000
```

Stream ciphers

Since addition of the same bit modulo 2 is always congruent with 0, decryption is gotten again by addition of the key (again).

Example

Using the same ciphertext and key as before:

$$\begin{matrix} 00100000 \\ 01101000 \\ 01001000 \end{matrix}$$

This gives us back the original plaintext 01001000.

Stream ciphers

Diffusion and confusion

Stream ciphers

Diffusion and confusion

Stream ciphers aren't very diffuse: a change in one plaintext character does not signifcantly change the ciphertext.

Stream ciphers

Diffusion and confusion

Stream ciphers aren't very diffuse: a change in one plaintext character does not signifcantly change the ciphertext.

Stream ciphers (arguably) do provide confusion. Changing the initial conditions used to generate a key may significantly change the ciphertext.

Stream ciphers

Diffusion and confusion

Stream ciphers aren't very diffuse: a change in one plaintext character does not signifcantly change the ciphertext.

Stream ciphers (arguably) do provide confusion. Changing the initial conditions used to generate a key may significantly change the ciphertext.

Stream ciphers are valuable when security is not as important.

Pseudorandom bit generators

Pseudorandom bit generators

Ideally, one would produce a sequence of random bits to be used as they key for a stream cipher. However, this is often impractical.

Pseudorandom bit generators

Ideally, one would produce a sequence of random bits to be used as they key for a stream cipher. However, this is often impractical.

Definition

*A **pseudorandom number (or bit) generator** is an algorithm, or function, used to generate a sequence of numbers (or bits) which has properties that approximate the properties of a sequence of random numbers (or bits).*

Pseudorandom bit generators

Example (Linear congruential generator)

Pick three integers $a, b \in \mathbb{Z}$ and $n > 0$. Let x_0 be an initial seed.

Pseudorandom bit generators

Example (Linear congruential generator)

Pick three integers $a, b \in \mathbb{Z}$ and $n > 0$. Let x_0 be an initial seed.

For all $i > 0$, define

$$x_i \equiv a x_{i-1} + b \pmod{n}.$$

Then $x_0, x_1, x_2, \ldots$ is a pseudorandom sequence of numbers.

Pseudorandom bit generators

Example (Linear congruential generator)

Pick three integers $a, b \in \mathbb{Z}$ and $n > 0$. Let x_0 be an initial seed.

For all $i > 0$, define

$$x_i \equiv a x_{i-1} + b \pmod{n}.$$

Then $x_0, x_1, x_2, \ldots$ is a pseudorandom sequence of numbers.

Example (Linear congruential generator)

Let $n = 26$, $a = -4$ and $b = 5$.

Pseudorandom bit generators

Example (Linear congruential generator)

Pick three integers $a, b \in \mathbb{Z}$ and $n > 0$. Let x_0 be an initial seed.

For all $i > 0$, define

$$x_i \equiv a x_{i-1} + b \pmod{n}.$$

Then $x_0, x_1, x_2, \ldots$ is a pseudorandom sequence of numbers.

Example (Linear congruential generator)

Let $n = 26$, $a = -4$ and $b = 5$. If $x_0 = 13$ then

$$x_1 = -4 \cdot x_0 + 5 \equiv -2(26) + 5 \equiv 5 \pmod{26}$$

Pseudorandom bit generators

Example (Linear congruential generator)

Pick three integers $a, b \in \mathbb{Z}$ and $n > 0$. Let x_0 be an initial seed.

For all $i > 0$, define

$$x_i \equiv a x_{i-1} + b \pmod{n}.$$

Then $x_0, x_1, x_2, \ldots$ is a pseudorandom sequence of numbers.

Example (Linear congruential generator)

Let $n = 26$, $a = -4$ and $b = 5$. If $x_0 = 13$ then

$$x_1 = -4 \cdot x_0 + 5 \equiv -2(26) + 5 \equiv 5 \pmod{26}$$

$$x_2 = -4 \cdot 5 + 5 \equiv 11 \pmod{26}.$$

Pseudorandom bit generators

Example (Linear congruential generator)

Pick three integers $a, b \in \mathbb{Z}$ and $n > 0$. Let x_0 be an initial seed.

For all $i > 0$, define

$$x_i \equiv a x_{i-1} + b \pmod{n}.$$

Then $x_0, x_1, x_2, \ldots$ is a pseudorandom sequence of numbers.

Example (Linear congruential generator)

Let $n = 26$, $a = -4$ and $b = 5$. If $x_0 = 13$ then

$$x_1 = -4 \cdot x_0 + 5 \equiv -2(26) + 5 \equiv 5 \pmod{26}$$

$$x_2 = -4 \cdot 5 + 5 \equiv 11 \pmod{26}.$$

Also $(x_3, x_4, x_5, \ldots) = (13, 5, 11, \ldots)$.

Pseudorandom bit generators

Example (Blum-Blum-Shub)

Pick large primes $p \equiv q \equiv 3 \pmod 4$.

Pseudorandom bit generators

Example (Blum-Blum-Shub)

Pick large primes $p \equiv q \equiv 3 \pmod 4$. Let x_0 be a random integer coprime to $n = pq$.

Pseudorandom bit generators

Example (Blum-Blum-Shub)

Pick large primes $p \equiv q \equiv 3 \pmod 4$. Let x_0 be a random integer coprime to $n = pq$.

For all $i > 0$, define $x_i \equiv x_{i-1}^2 \pmod n$. Take $b_i \equiv x_i \pmod 2$.

Pseudorandom bit generators

Example (Blum-Blum-Shub)

Pick large primes $p \equiv q \equiv 3 \pmod 4$. Let x_0 be a random integer coprime to $n = pq$.

For all $i > 0$, define $x_i \equiv x_{i-1}^2 \pmod n$. Take $b_i \equiv x_i \pmod 2$.

The sequence $b_0, b_1, b_2, \ldots$ is a pseudorandom sequence of bits.

Pseudorandom bit generators

Example (Blum-Blum-Shub)

Pick large primes $p \equiv q \equiv 3 \pmod 4$. Let x_0 be a random integer coprime to $n = pq$.

For all $i > 0$, define $x_i \equiv x_{i-1}^2 \pmod n$. Take $b_i \equiv x_i \pmod 2$.

The sequence $b_0, b_1, b_2, \ldots$ is a pseudorandom sequence of bits.

Example (Blum-Blum-Shub)

Let $n = 227 \cdot 479 = 108733$. Set $x_0 = 17$. Then

$$x_1 \equiv x_0^2 \equiv 289 \pmod{108733}$$

Pseudorandom bit generators

Example (Blum-Blum-Shub)

Pick large primes $p \equiv q \equiv 3 \pmod 4$. Let x_0 be a random integer coprime to $n = pq$.

For all $i > 0$, define $x_i \equiv x_{i-1}^2 \pmod n$. Take $b_i \equiv x_i \pmod 2$.

The sequence $b_0, b_1, b_2, \ldots$ is a pseudorandom sequence of bits.

Example (Blum-Blum-Shub)

Let $n = 227 \cdot 479 = 108733$. Set $x_0 = 17$. Then

$$x_1 \equiv x_0^2 \equiv 289 \pmod{108733}$$

$$x_2 \equiv x_1^2 \equiv 83521 \pmod{108733}$$

Pseudorandom bit generators

Example (Blum-Blum-Shub)

Pick large primes $p \equiv q \equiv 3 \pmod 4$. Let x_0 be a random integer coprime to $n = pq$.

For all $i > 0$, define $x_i \equiv x_{i-1}^2 \pmod n$. Take $b_i \equiv x_i \pmod 2$.

The sequence $b_0, b_1, b_2, \ldots$ is a pseudorandom sequence of bits.

Example (Blum-Blum-Shub)

Let $n = 227 \cdot 479 = 108733$. Set $x_0 = 17$. Then

$$x_1 \equiv x_0^2 \equiv 289 \pmod{108733}$$

$$x_2 \equiv x_1^2 \equiv 83521 \pmod{108733}$$

$$x_3 \equiv x_2^2 \equiv 516246 \equiv 81314 \pmod{108733}.$$

Pseudorandom bit generators

Example (Blum-Blum-Shub)

Pick large primes $p \equiv q \equiv 3 \pmod 4$. Let x_0 be a random integer coprime to $n = pq$.

For all $i > 0$, define $x_i \equiv x_{i-1}^2 \pmod n$. Take $b_i \equiv x_i \pmod 2$.

The sequence $b_0, b_1, b_2, \ldots$ is a pseudorandom sequence of bits.

Example (Blum-Blum-Shub)

Let $n = 227 \cdot 479 = 108733$. Set $x_0 = 17$. Then

$$x_1 \equiv x_0^2 \equiv 289 \pmod{108733}$$

$$x_2 \equiv x_1^2 \equiv 83521 \pmod{108733}$$

$$x_3 \equiv x_2^2 \equiv 516246 \equiv 81314 \pmod{108733}.$$

So $(b_0, b_1, b_2, b_3, \ldots) = (1, 1, 1, 0, \ldots)$.

LFSR Sequences

Linear Feedback Shift Register Sequences

A *Linear Feedback Shift Register Sequence* is a sequence gotten from a recursive congruence

$$x_{n+m} \equiv c_0 x_n + c_1 x_{n+1} + \cdots + c_{m-1} x_{n+m-1} \pmod 2$$

for some $m > 0$, fixed starting values $c_0, c_1, ..., c_{m-1} \in \{0, 1\}$, and for initial values $x_0, ..., x_{m-1}$.

Linear Feedback Shift Register Sequences

A *Linear Feedback Shift Register Sequence* is a sequence gotten from a recursive congruence

$$x_{n+m} \equiv c_0 x_n + c_1 x_{n+1} + \cdots + c_{m-1} x_{n+m-1} \pmod 2$$

for some $m > 0$, fixed starting values $c_0, c_1, ..., c_{m-1} \in \{0, 1\}$, and for initial values $x_0, ..., x_{m-1}$.

Remark

This sequence can be easily implemented in hardware.

LFSR Sequences

LFSR sequences can be generated very easily and have long periods.

LFSR Sequences

LFSR sequences can be generated very easily and have long periods.

Example

The sequence generated by any nonzero initial vector $(x_0, ..., x_{30})$ using the relation

$$x_{n+31} \equiv x_n + x_{n+3} \pmod 2$$

has period $2^{31} - 1 = 2147483647$.

LFSR Sequences

LFSR sequences can be generated very easily and have long periods.

Example

The sequence generated by any nonzero initial vector $(x_0, ..., x_{30})$ using the relation

$$x_{n+31} \equiv x_n + x_{n+3} \pmod 2$$

has period $2^{31} - 1 = 2147483647$.

A stream cipher implemented using an LFSR sequence still succumbs to a known plaintext attack.

LFSR Sequences

Theorem

Let $x_1, x_2, x_3, \ldots$ be a sequence of bits produced by a linear recurrence relation mod 2.

LFSR Sequences

Theorem

Let $x_1, x_2, x_3, \ldots$ be a sequence of bits produced by a linear recurrence relation mod 2. For each $n \geq 1$, let

$$M_n = \begin{pmatrix} x_1 & x_2 & \cdots & x_n \\ x_2 & x_3 & \cdots & x_{n+1} \\ \vdots & \vdots & \ddots & \vdots \\ x_n & x_{n+1} & \cdots & x_{2n-1} \end{pmatrix}.$$

LFSR Sequences

Theorem

Let $x_1, x_2, x_3, \ldots$ be a sequence of bits produced by a linear recurrence relation mod 2. For each $n \geq 1$, let

$$M_n = \begin{pmatrix} x_1 & x_2 & \cdots & x_n \\ x_2 & x_3 & \cdots & x_{n+1} \\ \vdots & \vdots & \ddots & \vdots \\ x_n & x_{n+1} & \cdots & x_{2n-1} \end{pmatrix}.$$

Let N be the length of the shortest relation that generates our sequence.

LFSR Sequences

Theorem

Let $x_1, x_2, x_3, \ldots$ be a sequence of bits produced by a linear recurrence relation mod 2. For each $n \geq 1$, let

$$M_n = \begin{pmatrix} x_1 & x_2 & \cdots & x_n \\ x_2 & x_3 & \cdots & x_{n+1} \\ \vdots & \vdots & \ddots & \vdots \\ x_n & x_{n+1} & \cdots & x_{2n-1} \end{pmatrix}.$$

Let N be the length of the shortest relation that generates our sequence.

Then $\det(M_N) \equiv 1 \pmod 2$ *and* $\det(M_n) \equiv 0 \pmod 2$ *for all* $n > N$.

LFSR Sequences

Example

Suppose we use a known plaintext attack to find values $x_1 = 0, x_2 = 1, x_3 = 1, x_4 = 0, x_5 = 1, x_6 = 0, x_7 = 1, x_8 = 1, x_9 = 1$ of an LFSR sequence.

LFSR Sequences

Example

Suppose we use a known plaintext attack to find values $x_1 = 0, x_2 = 1, x_3 = 1, x_4 = 0, x_5 = 1, x_6 = 0, x_7 = 1, x_8 = 1, x_9 = 1$ of an LFSR sequence.

If the recurrence relation used to generate this sequence had the form

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

then we get

$$1 \equiv c_0 \cdot 0 + c_1 \cdot 1 \pmod 2$$
$$0 \equiv c_0 \cdot 1 + c_1 \cdot 1 \pmod 2$$

LFSR Sequences

Example

Suppose we use a known plaintext attack to find values $x_1 = 0, x_2 = 1, x_3 = 1, x_4 = 0, x_5 = 1, x_6 = 0, x_7 = 1, x_8 = 1, x_9 = 1$ of an LFSR sequence.

If the recurrence relation used to generate this sequence had the form

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

then we get

$$1 \equiv c_0 \cdot 0 + c_1 \cdot 1 \pmod 2$$
$$0 \equiv c_0 \cdot 1 + c_1 \cdot 1 \pmod 2$$

This has solution $c_0 = 1 = c_1$.

LFSR Sequences

Example

Suppose we use a known plaintext attack to find values $x_1 = 0, x_2 = 1, x_3 = 1, x_4 = 0, x_5 = 1, x_6 = 0, x_7 = 1, x_8 = 1, x_9 = 1$ of an LFSR sequence.

If the recurrence relation used to generate this sequence had the form

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

then we get

$$1 \equiv c_0 \cdot 0 + c_1 \cdot 1 \pmod 2$$
$$0 \equiv c_0 \cdot 1 + c_1 \cdot 1 \pmod 2$$

This has solution $c_0 = 1 = c_1$. But $x_6 = 0 \not\equiv 0 + 1 \equiv x_4 + x_5 \pmod 2$.

LFSR Sequences

Example

Suppose we use a known plaintext attack to find values $x_1 = 0, x_2 = 1, x_3 = 1, x_4 = 0, x_5 = 1, x_6 = 0, x_7 = 1, x_8 = 1, x_9 = 1$ of an LFSR sequence.

LFSR Sequences

Example

Suppose we use a known plaintext attack to find values $x_1 = 0, x_2 = 1, x_3 = 1, x_4 = 0, x_5 = 1, x_6 = 0, x_7 = 1, x_8 = 1, x_9 = 1$ of an LFSR sequence.

If the recurrence relation used to generate this sequence had the form

$$x_{n+3} \equiv c_0 x_n + c_1 x_{n+1} + c_2 x_{n+2} \pmod 2$$

then we get

$$0 \equiv 0 \cdot c_0 + 1 \cdot c_1 + 1 \cdot c_2 \pmod 2$$
$$1 \equiv 1 \cdot c_0 + 1 \cdot c_1 + 0 \cdot c_2 \pmod 2$$
$$0 \equiv 1 \cdot c_0 + 0 \cdot c_1 + 1 \cdot c_2 \pmod 2$$

LFSR Sequences

Example

Suppose we use a known plaintext attack to find values $x_1 = 0, x_2 = 1, x_3 = 1, x_4 = 0, x_5 = 1, x_6 = 0, x_7 = 1, x_8 = 1, x_9 = 1$ of an LFSR sequence.

If the recurrence relation used to generate this sequence had the form

$$x_{n+3} \equiv c_0 x_n + c_1 x_{n+1} + c_2 x_{n+2} \pmod 2$$

then we get

$$0 \equiv 0 \cdot c_0 + 1 \cdot c_1 + 1 \cdot c_2 \pmod 2$$
$$1 \equiv 1 \cdot c_0 + 1 \cdot c_1 + 0 \cdot c_2 \pmod 2$$
$$0 \equiv 1 \cdot c_0 + 0 \cdot c_1 + 1 \cdot c_2 \pmod 2$$

But this system of equations has no solution.

LFSR Sequences

Example

Suppose we use a known plaintext attack to find values $x_1 = 0, x_2 = 1, x_3 = 1, x_4 = 0, x_5 = 1, x_6 = 0, x_7 = 1, x_8 = 1, x_9 = 1$ of an LFSR sequence.

LFSR Sequences

Example

Suppose we use a known plaintext attack to find values $x_1 = 0, x_2 = 1, x_3 = 1, x_4 = 0, x_5 = 1, x_6 = 0, x_7 = 1, x_8 = 1, x_9 = 1$ of an LFSR sequence.

If the recurrence relation used to generate this sequence had the form

$$x_{n+4} \equiv c_0 x_n + c_1 x_{n+1} + c_2 x_{n+2} + c_3 x_{n+3} \pmod 2$$

then we get a system of equations similarly.

LFSR Sequences

Example

Suppose we use a known plaintext attack to find values $x_1 = 0, x_2 = 1, x_3 = 1, x_4 = 0, x_5 = 1, x_6 = 0, x_7 = 1, x_8 = 1, x_9 = 1$ of an LFSR sequence.

If the recurrence relation used to generate this sequence had the form

$$x_{n+4} \equiv c_0 x_n + c_1 x_{n+1} + c_2 x_{n+2} + c_3 x_{n+3} \pmod 2$$

then we get a system of equations similarly.

We can try to solve for c_0, c_1, c_2, c_3 and check if this relation generates all of the x_i values that we know.

LFSR Sequences

Example

Suppose we use a known plaintext attack to find values $x_1 = 0, x_2 = 1, x_3 = 1, x_4 = 0, x_5 = 1, x_6 = 0, x_7 = 1, x_8 = 1, x_9 = 1$ of an LFSR sequence.

If the recurrence relation used to generate this sequence had the form

$$x_{n+4} \equiv c_0 x_n + c_1 x_{n+1} + c_2 x_{n+2} + c_3 x_{n+3} \pmod 2$$

then we get a system of equations similarly.

We can try to solve for c_0, c_1, c_2, c_3 and check if this relation generates all of the x_i values that we know.

If this relation does generate all x_i that we know, we then verify that higher recurrence relations indeed have no solutions (as per the theorem).

LFSR Sequences

Example

Suppose we use a known plaintext attack to find values $x_1 = 0, x_2 = 1, x_3 = 1, x_4 = 0, x_5 = 1, x_6 = 0, x_7 = 1, x_8 = 1, x_9 = 1$ of an LFSR sequence.

If the recurrence relation used to generate this sequence had the form

$$x_{n+4} \equiv c_0 x_n + c_1 x_{n+1} + c_2 x_{n+2} + c_3 x_{n+3} \pmod 2$$

then we get a system of equations similarly.

We can try to solve for c_0, c_1, c_2, c_3 and check if this relation generates all of the x_i values that we know.

If this relation does generate all x_i that we know, we then verify that higher recurrence relations indeed have no solutions (as per the theorem).

If this is the case, then we can be (somewhat) confident that this is the generating recurrence relation.