BILL, RECORD LECTURE!!!!

BILL RECORD LECTURE!!!

The One-Time Pad
Trying to Fake the OTP
Failing To Do So

The One-Time Pad

Notation Reminder: $\oplus$

Notation $\oplus$ on bits. This is often called XOR as well.

b	c	$b \oplus c$
0	0	0
0	1	1
1	0	1
1	1	0

Question Why do $\wedge$, $\vee$, $\oplus$ have symbols that are commonly used but NAND and NOR do not?

Notation Reminder: $\oplus$

Notation $\oplus$ on bits. This is often called XOR as well.

b	c	$b \oplus c$
0	0	0
0	1	1
1	0	1
1	1	0

Question Why do $\wedge$, $\vee$, $\oplus$ have symbols that are commonly used but NAND and NOR do not?

Answer $\wedge$, $\vee$, $\oplus$ are **associative** ; NAND and NOR are not.

Notation Reminder: $\oplus$

Notation $\oplus$ on bits. This is often called XOR as well.

b	c	$b \oplus c$
0	0	0
0	1	1
1	0	1
1	1	0

Question Why do $\wedge$, $\vee$, $\oplus$ have symbols that are commonly used but NAND and NOR do not?

Answer $\wedge$, $\vee$, $\oplus$ are **associative** ; NAND and NOR are not.

$$(\forall a, b, c \in \{0, 1\})[(a \oplus b) \oplus c = a \oplus (b \oplus c).$$

Useful Fact about $\oplus$

1. $(\forall b \in \{0,1\})[b \oplus b = 0]$
2. $(\forall b \in \{0,1\})[b \oplus 0 = b]$

Theorem $(\forall b, c \in \{0,1\})[b \oplus c \oplus c = b]$

Proof $b \oplus (c \oplus c) = b \oplus 0 = b$.

Useful Fact about $\oplus$

1. $(\forall b \in \{0,1\})[b \oplus b = 0]$
2. $(\forall b \in \{0,1\})[b \oplus 0 = b]$

Theorem $(\forall b, c \in \{0,1\})[b \oplus c \oplus c = b]$

Proof $b \oplus (c \oplus c) = b \oplus 0 = b$.

The Theorem is very important for the 1-time pad.

Extend $\oplus$ to Strings

Extend $\oplus$ to strings. If $x, y \in \{0,1\}^n$ then $x \oplus y$ is done bitwise.
Example $0010 \oplus 1110 = (0 \oplus 1)(0 \oplus 1)(1 \oplus 1)(0 \oplus 0) = 1100$.

1. $(\forall x \in \{0,1\}^n)[x \oplus x = 0^n]$
2. $(\forall x \in \{0,1\}^n)[x \oplus 0^n = x]$

Theorem $(\forall x, y \in \{0,1\}^n)[x \oplus y \oplus y = x]$
Proof $x \oplus (y \oplus y) = x \oplus 0^n = x$.

One-Time Pad

One-Time Pad

- Let $\mathcal{M} = \{0,1\}^n$, the set of all messages.

One-Time Pad

- Let $\mathcal{M} = \{0, 1\}^n$, the set of all messages.
- *Gen*: choose a uniform key $k \in \{0, 1\}^n$.

One-Time Pad

- Let $\mathcal{M} = \{0,1\}^n$, the set of all messages.
- *Gen*: choose a uniform key $k \in \{0,1\}^n$.
- $Enc_k(m) = k \oplus m$.

One-Time Pad

- Let $\mathcal{M} = \{0,1\}^n$, the set of all messages.
- *Gen*: choose a uniform key $k \in \{0,1\}^n$.
- $Enc_k(m) = k \oplus m$.
- $Dec_k(c) = k \oplus c$.

One-Time Pad

- Let $\mathcal{M} = \{0,1\}^n$, the set of all messages.
- *Gen*: choose a uniform key $k \in \{0,1\}^n$.
- $Enc_k(m) = k \oplus m$.
- $Dec_k(c) = k \oplus c$.
- Correctness:

$$\begin{aligned} Dec_k(Enc_k(m)) &= k \oplus (k \oplus m) \\ &= (k \oplus k) \oplus m \\ &= m \end{aligned}$$

Example Of One-Time Pad

Key is 100010100010001111101111100

Example Of One-Time Pad

Key is 100010100010001111101111100

Alice wants to send Bob 1110.

Example Of One-Time Pad

Key is 10001010001000111110111100

Alice wants to send Bob 1110.

She sends $1110 \oplus 1000 = 0110$.

Example Of One-Time Pad

Key is 1000101000100011111011111100

Alice wants to send Bob 1110.

She sends $1110 \oplus 1000 = 0110$.

Then Bob wants to send Alice 00111.

Example Of One-Time Pad

Key is 1000101000100011111101111100

Alice wants to send Bob 1110.

She sends $1110 \oplus 1000 = 0110$.

Then Bob wants to send Alice 00111.

He sends $00111 \oplus 10100 = 10011$.

Example Of One-Time Pad

Key is 10001010001000111110111100

Alice wants to send Bob 1110.

She sends $1110 \oplus 1000 = 0110$.

Then Bob wants to send Alice 00111.

He sends $00111 \oplus 10100 = 10011$.

1. **PRO** $\oplus$ is FAST!

Example Of One-Time Pad

Key is 100010100010001111101111100

Alice wants to send Bob 1110.

She sends $1110 \oplus 1000 = 0110$.

Then Bob wants to send Alice 00111.

He sends $00111 \oplus 10100 = 10011$.

1. **PRO** $\oplus$ is FAST!
2. **CON** If Key is N bits long can only send N bits.

Example Of One-Time Pad

Key is 10001010001000111110111111100

Alice wants to send Bob 1110.

She sends $1110 \oplus 1000 = 0110$.

Then Bob wants to send Alice 00111.

He sends $00111 \oplus 10100 = 10011$.

1. **PRO** $\oplus$ is FAST!
2. **CON** If Key is N bits long can only send N bits.

Is the one-time pad uncrackable:
VOTE: Yes, No, or Other.

Example Of One-Time Pad

Key is 1000101000100011111011111100

Alice wants to send Bob 1110.

She sends $1110 \oplus 1000 = 0110$.

Then Bob wants to send Alice 00111.

He sends $00111 \oplus 10100 = 10011$.

1. **PRO** $\oplus$ is FAST!
2. **CON** If Key is N bits long can only send N bits.

Is the one-time pad uncrackable:
VOTE: Yes, No, or Other.
Yes. Really!

Example Of One-Time Pad

Key is 100010100010001111101111100

Alice wants to send Bob 1110.

She sends $1110 \oplus 1000 = 0110$.

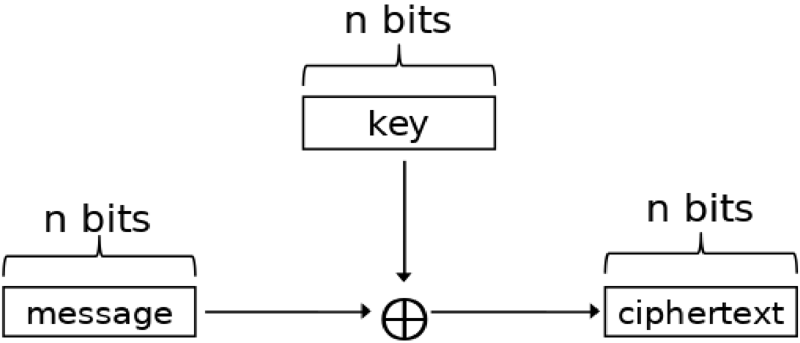
Then Bob wants to send Alice 00111.

He sends $00111 \oplus 10100 = 10011$.

1. **PRO** $\oplus$ is FAST!
2. **CON** If Key is N bits long can only send N bits.

Is the one-time pad uncrackable:
VOTE: Yes, No, or Other.
Yes. Really!
Caveat: Generating truly random bits is hard.

One-time pad



Alice and Bob Use the Psuedo One Time Pad

One-time pad (OTP)

One-time pad (OTP)

- The OTP was patented in 1917 by Vernam.

One-time pad (OTP)

- The OTP was patented in 1917 by Vernam.
- Historical research indicates the OTP was invented at least 35 years earlier.

One-time pad (OTP)

- The OTP was patented in 1917 by Vernam.
- Historical research indicates the OTP was invented at least 35 years earlier.
- The OTP was **proven** info-theoretic secure by Shannon in 1949.

Linear Cong. Generators

How Hard is it to Generate Truly Random Bits?

Paraphrase of a **Recent Piazza conversation**

Student You said that generating Random Bits is hard. Why?

How Hard is it to Generate Truly Random Bits?

Paraphrase of a **Recent Piazza conversation**

Student You said that generating Random Bits is hard. Why?

Bill *Truly* Rand Bits are hard. How would you do it?

How Hard is it to Generate Truly Random Bits?

Paraphrase of a **Recent Piazza conversation**

Student You said that generating Random Bits is hard. Why?

Bill *Truly* Rand Bits are hard. How would you do it?

Student Use the Random function in Java you muffinhead!

How Hard is it to Generate Truly Random Bits?

Paraphrase of a **Recent Piazza conversation**

Student You said that generating Random Bits is hard. Why?

Bill *Truly* Rand Bits are hard. How would you do it?

Student Use the Random function in Java you muffinhead!

Bill Okay. How does Java do it? Is it *Truly* Random?

How Hard is it to Generate Truly Random Bits?

Paraphrase of a **Recent Piazza conversation**

Student You said that generating Random Bits is hard. Why?

Bill *Truly* Rand Bits are hard. How would you do it?

Student Use the Random function in Java you muffinhead!

Bill Okay. How does Java do it? Is it *Truly* Random?

Student Oh.

How Hard is it to Generate Truly Random Bits?

Paraphrase of a **Recent Piazza conversation**

Student You said that generating Random Bits is hard. Why?

Bill *Truly* Rand Bits are hard. How would you do it?

Student Use the Random function in Java you muffinhead!

Bill Okay. How does Java do it? Is it *Truly* Random?

Student Oh. Okay, you tell me– how does Java do it?

How Hard is it to Generate Truly Random Bits?

Paraphrase of a **Recent Piazza conversation**

Student You said that generating Random Bits is hard. Why?

Bill *Truly* Rand Bits are hard. How would you do it?

Student Use the Random function in Java you muffinhead!

Bill Okay. How does Java do it? Is it *Truly* Random?

Student Oh. Okay, you tell me– how does Java do it?

Bill I will show what Java does and why it bytes.

How Does Java Produce Random Numbers

Java (and many old langs) uses a **Linear Cong. Generator.**
When the computer is turned on (and once a month after that):

How Does Java Produce Random Numbers

Java (and many old langs) uses a **Linear Cong. Generator.**
When the computer is turned on (and once a month after that):

1. Pick M large. A power of 2 makes life easier for Alice and Bob, but might not want to do that— we'll see why later.

How Does Java Produce Random Numbers

Java (and many old langs) uses a **Linear Cong. Generator.**
When the computer is turned on (and once a month after that):

1. Pick M large. A power of 2 makes life easier for Alice and Bob, but might not want to do that— we'll see why later.
2. A, B, x_0 are random-looking. E.g. the number of nanoseconds mod M since last time reboot.

How Does Java Produce Random Numbers

Java (and many old langs) uses a **Linear Cong. Generator.**
When the computer is turned on (and once a month after that):

1. Pick M large. A power of 2 makes life easier for Alice and Bob, but might not want to do that— we'll see why later.
2. A, B, x_0 are random-looking. E.g. the number of nanoseconds mod M since last time reboot.
3. The computer has the recurrence

$$x_{i+1} = Ax_i + B \pmod{M}$$

How Does Java Produce Random Numbers

Java (and many old langs) uses a **Linear Cong. Generator.**
When the computer is turned on (and once a month after that):

1. Pick M large. A power of 2 makes life easier for Alice and Bob, but might not want to do that— we'll see why later.
2. A, B, x_0 are random-looking. E.g. the number of nanoseconds mod M since last time reboot.
3. The computer has the recurrence

$$x_{i+1} = Ax_i + B \pmod{M}$$

4. The ith time a random number is chosen, use x_i.

How Does Java Produce Random Numbers

Java (and many old langs) uses a **Linear Cong. Generator.**
When the computer is turned on (and once a month after that):

1. Pick M large. A power of 2 makes life easier for Alice and Bob, but might not want to do that— we'll see why later.
2. A, B, x_0 are random-looking. E.g. the number of nanoseconds mod M since last time reboot.
3. The computer has the recurrence

$$x_{i+1} = Ax_i + B \pmod{M}$$

4. The ith time a random number is chosen, use x_i.
5. Computer need only keep x_i, A, B, M in memory.

How Does Java Produce Random Numbers

Java (and many old langs) uses a **Linear Cong. Generator.**
When the computer is turned on (and once a month after that):

1. Pick M large. A power of 2 makes life easier for Alice and Bob, but might not want to do that— we'll see why later.
2. A, B, x_0 are random-looking. E.g. the number of nanoseconds mod M since last time reboot.
3. The computer has the recurrence

$$x_{i+1} = Ax_i + B \pmod{M}$$

4. The ith time a random number is chosen, use x_i.
5. Computer need only keep x_i, A, B, M in memory.

Depending on A, B, x_0 this can look random... or not.

Restrictions on A, B, M

What if M and A share a factor?

Restrictions on A, B, M

What if M and A share a factor?

Example

$x_0 = 5$

$x_{n+1} \equiv 2x_n + 5 \pmod{8}$

Restrictions on A, B, M

What if M and A share a factor?

Example

$x_0 = 5$

$x_{n+1} \equiv 2x_n + 5 \pmod{8}$

$x_1 = 2 * 5 + 5 = 15 \equiv 7$

Restrictions on A, B, M

What if M and A share a factor?

Example

$x_0 = 5$

$x_{n+1} \equiv 2x_n + 5 \pmod 8$

$x_1 = 2 * 5 + 5 = 15 \equiv 7$

$x_2 = 2 * 7 + 5 = 19 \equiv 3$

Restrictions on A, B, M

What if M and A share a factor?

Example

$x_0 = 5$

$x_{n+1} \equiv 2x_n + 5 \pmod 8$

$x_1 = 2 * 5 + 5 = 15 \equiv 7$

$x_2 = 2 * 7 + 5 = 19 \equiv 3$

$x_3 = 2 * 3 + 5 = 11 \equiv 3$

Restrictions on A, B, M

What if M and A share a factor?

Example

$x_0 = 5$

$x_{n+1} \equiv 2x_n + 5 \pmod 8$

$x_1 = 2 * 5 + 5 = 15 \equiv 7$

$x_2 = 2 * 7 + 5 = 19 \equiv 3$

$x_3 = 2 * 3 + 5 = 11 \equiv 3$

$(\forall i \geq 2)[x_i = 3]$.

Restrictions on A, B, M

What if M and A share a factor?

Example

$x_0 = 5$

$x_{n+1} \equiv 2x_n + 5 \pmod 8$

$x_1 = 2 * 5 + 5 = 15 \equiv 7$

$x_2 = 2 * 7 + 5 = 19 \equiv 3$

$x_3 = 2 * 3 + 5 = 11 \equiv 3$

$(\forall i \geq 2)[x_i = 3]$.

This is typical. If A is not rel prime to M then the numbers obtained will be only a small part of $\{0, \ldots, M-1\}$.

Restrictions on A, B, M

What if M and A share a factor?

Example

$x_0 = 5$

$x_{n+1} \equiv 2x_n + 5 \pmod{8}$

$x_1 = 2 * 5 + 5 = 15 \equiv 7$

$x_2 = 2 * 7 + 5 = 19 \equiv 3$

$x_3 = 2 * 3 + 5 = 11 \equiv 3$

$(\forall i \geq 2)[x_i = 3]$.

This is typical. If A is not rel prime to M then the numbers obtained will be only a small part of $\{0, \ldots, M-1\}$.

Eve will assume that A and M are rel prime. We need to assume more: next slide.

Conditions on x_0, A, B, M

1. $1 \leq x_0, A, B \leq 9999$.
2. $1000 \leq M \leq 9999$.
3. A, M are Rel Prime.

Example of Linear Cong. Gen

$x_0 = 21,\ A = 19,\ B = 30,\ M = 91$

$x_0 = 21$

$x_1 = 19 * 21 + 30 \pmod{91} = 65$

$x_2 = 19 * 65 + 30 \pmod{91} = 82$

$x_3 = 19 * 82 + 30 \pmod{91} = 41$

$x_4 = 19 * 41 + 30 \pmod{91} = 81$

$x_5 = 19 * 81 + 30 \pmod{91} = 22$

$x_6 = 19 * 22 + 30 \pmod{91} = 84$

$x_7 = 19 * 84 + 30 \pmod{91} = 79$

$x_8 = 19 * 79 + 30 \pmod{91} = 75$

Example of Linear Cong. Gen

$x_0 = 21,\ A = 19,\ B = 30,\ M = 91$
$x_0 = 21$
$x_1 = 19 * 21 + 30 \pmod{91} = 65$
$x_2 = 19 * 65 + 30 \pmod{91} = 82$
$x_3 = 19 * 82 + 30 \pmod{91} = 41$
$x_4 = 19 * 41 + 30 \pmod{91} = 81$
$x_5 = 19 * 81 + 30 \pmod{91} = 22$
$x_6 = 19 * 22 + 30 \pmod{91} = 84$
$x_7 = 19 * 84 + 30 \pmod{91} = 79$
$x_8 = 19 * 79 + 30 \pmod{91} = 75$
Does this sequence look random?

Example of Linear Cong. Gen

$x_0 = 21,\ A = 19,\ B = 30,\ M = 91$

$x_0 = 21$

$x_1 = 19 * 21 + 30 \pmod{91} = 65$

$x_2 = 19 * 65 + 30 \pmod{91} = 82$

$x_3 = 19 * 82 + 30 \pmod{91} = 41$

$x_4 = 19 * 41 + 30 \pmod{91} = 81$

$x_5 = 19 * 81 + 30 \pmod{91} = 22$

$x_6 = 19 * 22 + 30 \pmod{91} = 84$

$x_7 = 19 * 84 + 30 \pmod{91} = 79$

$x_8 = 19 * 79 + 30 \pmod{91} = 75$

Does this sequence look random? Hard to say.

Our Running Example

$x_0 = \mathbf{2134}$, $A = 4381$, $B = 7364$, $M = 8397$.

$$\begin{aligned} x_0 &= 2134 \text{ view as } 21, 34 \\ x_{n+1} &= 4381x_n + 7364 \pmod{8397} \end{aligned}$$

Our Running Example

$x_0 = \mathbf{2134}$, $A = 4381$, $B = 7364$, $M = 8397$.

$$\begin{aligned} x_0 &= 2134 \text{ view as } 21, 34 \\ x_{n+1} &= 4381x_n + 7364 \pmod{8397} \end{aligned}$$

We use this to gen rand-looking bits, so 1-time-pad with psuedo-random bits.

Our Running Example

$x_0 = \mathbf{2134}$, $A = 4381$, $B = 7364$, $M = 8397$.

$$\begin{aligned} x_0 &= 2134 \text{ view as } 21, 34 \\ x_{n+1} &= 4381x_n + 7364 \pmod{8397} \end{aligned}$$

We use this to gen rand-looking bits, so 1-time-pad with psuedo-random bits.

We will then crack it.

Our Running Example

$x_0 = \mathbf{2134}$, $A = 4381$, $B = 7364$, $M = 8397$.

$$\begin{aligned} x_0 &= 2134 \text{ view as } 21, 34 \\ x_{n+1} &= 4381x_n + 7364 \pmod{8397} \end{aligned}$$

We use this to gen rand-looking bits, so 1-time-pad with psuedo-random bits.

We will then crack it.

We will assume Eve knows that the random numbers are gen by a recurrence of the form

$$x_{i+1} = Ax_i + B \pmod{M}$$

but that Eve do not know x_0, A, B, M. Does know A, M rel prime.

Alice and Bob Use the Psuedo One Time Pad

Psuedo One-Time Pad

$A = 01$, $B = 02$, $\cdots$ $Z = 26$ (**Not our usual since $A = 01$.**)
View each letter as a two-digit number mod 26.

Psuedo One-Time Pad

$A = 01$, $B = 02$, $\cdots$ $Z = 26$ (**Not our usual since $A = 01$.**)
View each letter as a two-digit number mod 26.
Want a LONG sequence of 2-digit numbers $k_1, k_2, \ldots$

Psuedo One-Time Pad

$A = 01$, $B = 02$, $\cdots$ $Z = 26$ (**Not our usual since $A = 01$.**)
View each letter as a two-digit number mod 26.
Want a LONG sequence of 2-digit numbers $k_1, k_2, \ldots$

1. Will code $m_1, m_2, \ldots$ by, **by adding mod 10 to each digit**
   Example If key is 12 38 and message is 29 23 then send

$$\begin{array}{cc} 12 & 38 \\ 29 & 23 \\ \hline 31 & 51 \end{array}$$

   So send 31 51 (these do not correspond to letters, thats fine).

Psuedo One-Time Pad

$A = 01$, $B = 02$, $\cdots$ $Z = 26$ (**Not our usual since $A = 01$.**)
View each letter as a two-digit number mod 26.
Want a LONG sequence of 2-digit numbers $k_1, k_2, \ldots$

1. Will code $m_1, m_2, \ldots$ by, **by adding mod 10 to each digit**
   Example If key is 12 38 and message is 29 23 then send

$$\begin{array}{cc} 12 & 38 \\ 29 & 23 \\ \hline 31 & 51 \end{array}$$

   So send 31 51 (these do not correspond to letters, thats fine).
2. View as One-time pad with psuedo-random sequence.

Psuedo One-Time Pad

$A = 01$, $B = 02$, $\cdots$ $Z = 26$ (**Not our usual since $A = 01$.**)
View each letter as a two-digit number mod 26.
Want a LONG sequence of 2-digit numbers $k_1, k_2, \ldots$

1. Will code $m_1, m_2, \ldots$ by, **by adding mod 10 to each digit**
   Example If key is 12 38 and message is 29 23 then send

$$\begin{array}{cc} 12 & 38 \\ 29 & 23 \\ \hline 31 & 51 \end{array}$$

   So send 31 51 (these do not correspond to letters, thats fine).
2. View as One-time pad with psuedo-random sequence.

How to code and decode? Next slide.

Running Example

From **Cracking a Random Number Generator** by James Reed. Paper on Course Website.

Running Example

From **Cracking a Random Number Generator** by James Reed.
Paper on Course Website.

$x_0 = 2134,\ A = 4381,\ B = 7364,\ M = 8397.$

Running Example

From **Cracking a Random Number Generator** by James Reed. Paper on Course Website.

$x_0 = 2134,\ A = 4381,\ B = 7364,\ M = 8397.$

$$\begin{aligned} x_0 &= 2134 \text{ view as } 21, 34 \\ x_{n+1} &= 4381 x_n + 7364 \pmod{8397} \end{aligned}$$

$x_0 = 2134$
$x_1 = 2160$
$x_2 = 6905$
$x_3 = 3778$

How Alice Codes: An Example

How Alice Codes: An Example

$x_0 = \mathbf{2134}$
$x_1 = \mathbf{2160}$
$x_2 = \mathbf{6905}$
$x_3 = \mathbf{3778}$
They start with x_1.

How Alice Codes: An Example

$x_0 = \mathbf{2134}$
$x_1 = \mathbf{2160}$
$x_2 = \mathbf{6905}$
$x_3 = \mathbf{3778}$
They start with x_1.
If the document began with the word **secret** then encode by adding columns base 10:

How Alice Codes: An Example

$x_0 = \mathbf{2134}$
$x_1 = \mathbf{2160}$
$x_2 = \mathbf{6905}$
$x_3 = \mathbf{3778}$
They start with x_1.
If the document began with the word **secret** then encode by adding columns base 10:

Text-Letter	S	E	C	R	E	T
Text-Digits	19	05	03	18	05	20
Key–Digits	**21**	**60**	**69**	**05**	**37**	**78**
Ciphertext	30	65	62	13	32	98

How Alice Codes: An Example

$x_0 = \mathbf{2134}$
$x_1 = \mathbf{2160}$
$x_2 = \mathbf{6905}$
$x_3 = \mathbf{3778}$
They start with x_1.
If the document began with the word **secret** then encode by adding columns base 10:

Text-Letter	S	E	C	R	E	T
Text-Digits	19	05	03	18	05	20
Key–Digits	**21**	**60**	**69**	**05**	**37**	**78**
Ciphertext	30	65	62	13	32	98

Note E is coded as 65 and then later as 32. Recall that the whole point of OTP is that a letter won't always be coded the same way.

How Alice Codes: General

The sequence is $x_0, x_1, x_2, \ldots$.
Each x_i is two **digits** : x_{i1}, x_{i2}.

How Alice Codes: General

The sequence is $x_0, x_1, x_2, \ldots$.
Each x_i is two **digits** : x_{i1}, x_{i2}.

Alice starts with x_1 (not with x_0).

How Alice Codes: General

The sequence is $x_0, x_1, x_2, \ldots$.
Each x_i is two **digits** : x_{i1}, x_{i2}.

Alice starts with x_1 (not with x_0).
Alice wants to send $m_1 m_2 \cdots$ where the m_i are letters.

How Alice Codes: General

The sequence is $x_0, x_1, x_2, \ldots$.
Each x_i is two **digits** : x_{i1}, x_{i2}.

Alice starts with x_1 (not with x_0).
Alice wants to send $m_1 m_2 \cdots$ where the m_i are letters.
Alice codes letters into 2-digits, so m_1 is $m_{1,1} m_{1,2}$, etc.

How Alice Codes: General

The sequence is $x_0, x_1, x_2, \ldots$.
Each x_i is two **digits** : x_{i1}, x_{i2}.

Alice starts with x_1 (not with x_0).
Alice wants to send $m_1 m_2 \cdots$ where the m_i are letters.
Alice codes letters into 2-digits, so m_1 is $m_{1,1} m_{1,2}$, etc.

All arithmetic is mod 10.

How Alice Codes: General

The sequence is $x_0, x_1, x_2, \ldots$.
Each x_i is two **digits** : x_{i1}, x_{i2}.

Alice starts with x_1 (not with x_0).
Alice wants to send $m_1 m_2 \cdots$ where the m_i are letters.
Alice codes letters into 2-digits, so m_1 is $m_{1,1} m_{1,2}$, etc.

All arithmetic is mod 10.

Plaintext	$m_{1,1} m_{1,2}$	$m_{2,1} m_{2,2}$
Key	$x_{1,1} x_{1,2}$	$x_{2,1} x_{2,2}$
Alice Sends	$(m_{1,1} + x_{1,1})(m_{1,2} + x_{1,2})$	$(m_{2,1} + x_{2,1})(m_{2,2} + x_{2,2})$

How Alice Codes: General

The sequence is $x_0, x_1, x_2, \ldots$.
Each x_i is two **digits** : x_{i1}, x_{i2}.

Alice starts with x_1 (not with x_0).
Alice wants to send $m_1 m_2 \cdots$ where the m_i are letters.
Alice codes letters into 2-digits, so m_1 is $m_{1,1} m_{1,2}$, etc.

All arithmetic is mod 10.

Plaintext	$m_{1,1} m_{1,2}$	$m_{2,1} m_{2,2}$
Key	$x_{1,1} x_{1,2}$	$x_{2,1} x_{2,2}$
Alice Sends	$(m_{1,1} + x_{1,1})(m_{1,2} + x_{1,2})$	$(m_{2,1} + x_{2,1})(m_{2,2} + x_{2,2})$

$(m_{1,1} + x_{1,1})(m_{1,2} + x_{1,2})$ is concatenation, not multiplication.

How Bob Decodes: An Example

Note Alice and Bob both know x_0, A, B, M so both know $x_1, x_2, \ldots,$.

How Bob Decodes: An Example

Note Alice and Bob both know x_0, A, B, M so both know $x_1, x_2, \ldots,$.

Bob Wants	$m_{1,1}\, m_{1,2}$	$m_{2,1}\, m_{2,2}$	$m_{3,1}\, m_{3,2}$
Bob Knows Key	21	60	69
Bob Sees	30	65	62

How Bob Decodes: An Example

Note Alice and Bob both know x_0, A, B, M so both know $x_1, x_2, \ldots,$.

Bob Wants	$m_{1,1}\, m_{1,2}$	$m_{2,1}\, m_{2,2}$	$m_{3,1}\, m_{3,2}$
Bob Knows Key	21	60	69
Bob Sees	30	65	62

Bob does the following, all mod 10:

How Bob Decodes: An Example

Note Alice and Bob both know x_0, A, B, M so both know $x_1, x_2, \ldots,$.

Bob Wants	$m_{1,1}\,m_{1,2}$	$m_{2,1}\,m_{2,2}$	$m_{3,1}\,m_{3,2}$
Bob Knows Key	21	60	69
Bob Sees	30	65	62

Bob does the following, all mod 10:

$m_{1,1} + 2 \equiv 3$ so $m_{1,1} \equiv 3 - 2 \equiv 1$.

How Bob Decodes: An Example

Note Alice and Bob both know x_0, A, B, M so both know $x_1, x_2, \ldots,$.

Bob Wants	$m_{1,1}\, m_{1,2}$	$m_{2,1}\, m_{2,2}$	$m_{3,1}\, m_{3,2}$
Bob Knows Key	21	60	69
Bob Sees	30	65	62

Bob does the following, all mod 10:

$m_{1,1} + 2 \equiv 3$ so $m_{1,1} \equiv 3 - 2 \equiv 1$.
$m_{1,2} + 1 \equiv 0$ so $m_{1,2} \equiv -1 \equiv 9$.

How Bob Decodes: An Example

Note Alice and Bob both know x_0, A, B, M so both know $x_1, x_2, \ldots,$.

Bob Wants	$m_{1,1} m_{1,2}$	$m_{2,1} m_{2,2}$	$m_{3,1} m_{3,2}$
Bob Knows Key	21	60	69
Bob Sees	30	65	62

Bob does the following, all mod 10:

$m_{1,1} + 2 \equiv 3$ so $m_{1,1} \equiv 3 - 2 \equiv 1$.
$m_{1,2} + 1 \equiv 0$ so $m_{1,2} \equiv -1 \equiv 9$.

Hence the first letter is 19 which is S.

How Bob Decodes: An Example

Note Alice and Bob both know x_0, A, B, M so both know $x_1, x_2, \ldots,$.

Bob Wants	$m_{1,1}\, m_{1,2}$	$m_{2,1}\, m_{2,2}$	$m_{3,1}\, m_{3,2}$
Bob Knows Key	21	60	69
Bob Sees	30	65	62

Bob does the following, all mod 10:

$m_{1,1} + 2 \equiv 3$ so $m_{1,1} \equiv 3 - 2 \equiv 1$.
$m_{1,2} + 1 \equiv 0$ so $m_{1,2} \equiv -1 \equiv 9$.

Hence the first letter is 19 which is S.
Bob can keep doing this to get the entire message.

How Bob Decodes: General

Note Alice and Bob both know x_0, A, B, M so both know $x_1, x_2, \ldots,$.
Bob starts with x_1 (not with x_0).

How Bob Decodes: General

Note Alice and Bob both know x_0, A, B, M so both know $x_1, x_2, \ldots,$.
Bob starts with x_1 (not with x_0).

All arithmetic is mod 10.

How Bob Decodes: General

Note Alice and Bob both know x_0, A, B, M so both know $x_1, x_2, \ldots,.$
Bob starts with x_1 (not with x_0).

All arithmetic is mod 10.

Bob Wants	$m_{1,1}m_{1,2}$	$m_{2,1}m_{2,2}$	$m_{3,1}m_{3,2}$
Bob Knows Key	$x_{1,1}x_{1,2}$	$x_{2,1}x_{2,2}$	$x_{3,1}x_{3,2}$
Bob Sees	$c_{1,1}c_{1,2}$	$c_{2,1}c_{2,2}$	$c_{3,1}c_{3,2}$

How Bob Decodes: General

Note Alice and Bob both know x_0, A, B, M so both know $x_1, x_2, \ldots,$.
Bob starts with x_1 (not with x_0).

All arithmetic is mod 10.

Bob Wants	$m_{1,1}m_{1,2}$	$m_{2,1}m_{2,2}$	$m_{3,1}m_{3,2}$
Bob Knows Key	$x_{1,1}x_{1,2}$	$x_{2,1}x_{2,2}$	$x_{3,1}x_{3,2}$
Bob Sees	$c_{1,1}c_{1,2}$	$c_{2,1}c_{2,2}$	$c_{3,1}c_{3,2}$

Bob does the following, all mod 10:

How Bob Decodes: General

Note Alice and Bob both know x_0, A, B, M so both know $x_1, x_2, \ldots,$.
Bob starts with x_1 (not with x_0).

All arithmetic is mod 10.

Bob Wants	$m_{1,1}m_{1,2}$	$m_{2,1}m_{2,2}$	$m_{3,1}m_{3,2}$
Bob Knows Key	$x_{1,1}x_{1,2}$	$x_{2,1}x_{2,2}$	$x_{3,1}x_{3,2}$
Bob Sees	$c_{1,1}c_{1,2}$	$c_{2,1}c_{2,2}$	$c_{3,1}c_{3,2}$

Bob does the following, all mod 10:

$m_{1,1} + x_{1,1} \equiv c_{1,1}$ so $m_{1,1} \equiv c_{1,1} - x_{1,1}$.

How Bob Decodes: General

Note Alice and Bob both know x_0, A, B, M so both know $x_1, x_2, \ldots,$.
Bob starts with x_1 (not with x_0).

All arithmetic is mod 10.

Bob Wants	$m_{1,1}m_{1,2}$	$m_{2,1}m_{2,2}$	$m_{3,1}m_{3,2}$
Bob Knows Key	$x_{1,1}x_{1,2}$	$x_{2,1}x_{2,2}$	$x_{3,1}x_{3,2}$
Bob Sees	$c_{1,1}c_{1,2}$	$c_{2,1}c_{2,2}$	$c_{3,1}c_{3,2}$

Bob does the following, all mod 10:

$m_{1,1} + x_{1,1} \equiv c_{1,1}$ so $m_{1,1} \equiv c_{1,1} - x_{1,1}$.
$m_{1,2} + x_{1,2} \equiv c_{1,2}$ so $m_{1,2} \equiv c_{1,2} - x_{1,2}$.

How Bob Decodes: General

Note Alice and Bob both know x_0, A, B, M so both know $x_1, x_2, \ldots,$.
Bob starts with x_1 (not with x_0).

All arithmetic is mod 10.

Bob Wants	$m_{1,1} m_{1,2}$	$m_{2,1} m_{2,2}$	$m_{3,1} m_{3,2}$
Bob Knows Key	$x_{1,1} x_{1,2}$	$x_{2,1} x_{2,2}$	$x_{3,1} x_{3,2}$
Bob Sees	$c_{1,1} c_{1,2}$	$c_{2,1} c_{2,2}$	$c_{3,1} c_{3,2}$

Bob does the following, all mod 10:

$m_{1,1} + x_{1,1} \equiv c_{1,1}$ so $m_{1,1} \equiv c_{1,1} - x_{1,1}$.
$m_{1,2} + x_{1,2} \equiv c_{1,2}$ so $m_{1,2} \equiv c_{1,2} - x_{1,2}$.

So first letter is $(c_{1,1} - x_{1,1})(c_{1,2} - x_{1,2})$.

He can keep on doing this.

Eve Can Crack The Psuedo One Time Pad

Credit Where Credit is Due

This presentation is based on the paper
Cracking a Random Number Generator by James Reed.
which is on the Course Website.

Eve Can Crack the Code

Alice sends Bob a document using the x_i as a two chars at a time.

Eve Can Crack the Code

Alice sends Bob a document using the x_i as a two chars at a time.

Eve knows rec of form $x_{n+1} = Ax_n + B \pmod{M}$.

Eve Can Crack the Code

Alice sends Bob a document using the x_i as a two chars at a time.

Eve knows rec of form $x_{n+1} = Ax_n + B \pmod{M}$.

Eve knows that A, B, M are all 4-digits. If she fails she may try again with 6-digits.

Eve Can Crack the Code

Alice sends Bob a document using the x_i as a two chars at a time.

Eve knows rec of form $x_{n+1} = Ax_n + B \pmod{M}$.

Eve knows that A, B, M are all 4-digits. If she fails she may try again with 6-digits.

Eve knows that the document is about **India** and **Pakistan.**

Eve Can Crack the Code

Alice sends Bob a document using the x_i as a two chars at a time.

Eve knows rec of form $x_{n+1} = Ax_n + B \pmod{M}$.

Eve knows that A, B, M are all 4-digits. If she fails she may try again with 6-digits.

Eve knows that the document is about **India** and **Pakistan.**

Eve thinks **Pakistan** will be in the document.
Eve thinks M is 4-digits.

Eve Can Crack the Code

Alice sends Bob a document using the x_i as a two chars at a time.

Eve knows rec of form $x_{n+1} = Ax_n + B \pmod{M}$.

Eve knows that A, B, M are all 4-digits. If she fails she may try again with 6-digits.

Eve knows that the document is about **India** and **Pakistan.**

Eve thinks **Pakistan** will be in the document.
Eve thinks M is 4-digits.

Text-Letter	P	A	K	I	S	T	A	N
Text-Digits	16	01	11	09	19	20	01	14

Thought Experiment

Eve sees

Thought Experiment

Eve sees

Ciphertext	24	66	87	47	17	45	26	96

Thought Experiment

Eve sees

Ciphertext	24	66	87	47	17	45	26	96

And thinks it is PAKISTAN.

Thought Experiment

Eve sees

Ciphertext	24	66	87	47	17	45	26	96

And thinks it is PAKISTAN.

So Eve thinks the following:

Thought Experiment

Eve sees

Ciphertext	24	66	87	47	17	45	26	96

And thinks it is PAKISTAN.

So Eve thinks the following:

Text-Letter	P	A	K	I	S	T	A
Text-Digits	16	01	11	09	19	20	01
Key-Digits	$k_{11}k_{12}$	$k_{21}k_{22}$	$k_{31}k_{32}$	$k_{41}k_{42}$	$k_{51}k_{52}$	$k_{61}k_{62}$	$k_{71}k_{7}$
Ciphertext	24	66	87	47	17	45	26

Thought Experiment

Eve sees

Ciphertext	24	66	87	47	17	45	26	96

And thinks it is PAKISTAN.

So Eve thinks the following:

Text-Letter	P	A	K	I	S	T	A
Text-Digits	16	01	11	09	19	20	01
Key-Digits	$k_{11} k_{12}$	$k_{21} k_{22}$	$k_{31} k_{32}$	$k_{41} k_{42}$	$k_{51} k_{52}$	$k_{61} k_{62}$	$k_{71} k_{7}$
Ciphertext	24	66	87	47	17	45	26

Can Eve find the Key-Digits?

Thought Experiment

Eve sees

Ciphertext	24	66	87	47	17	45	26	96

And thinks it is PAKISTAN.

So Eve thinks the following:

Text-Letter	P	A	K	I	S	T	A
Text-Digits	16	01	11	09	19	20	01
Key-Digits	$k_{11}k_{12}$	$k_{21}k_{22}$	$k_{31}k_{32}$	$k_{41}k_{42}$	$k_{51}k_{52}$	$k_{61}k_{62}$	$k_{71}k_{7}$
Ciphertext	24	66	87	47	17	45	26

Can Eve find the Key-Digits? Yes!

Thought Experiment

Eve sees

Ciphertext	24	66	87	47	17	45	26	96

And thinks it is PAKISTAN.

So Eve thinks the following:

Text-Letter	P	A	K	I	S	T	A
Text-Digits	16	01	11	09	19	20	01
Key-Digits	$k_{11}k_{12}$	$k_{21}k_{22}$	$k_{31}k_{32}$	$k_{41}k_{42}$	$k_{51}k_{52}$	$k_{61}k_{62}$	$k_{71}k_{7}$
Ciphertext	24	66	87	47	17	45	26

Can Eve find the Key-Digits? Yes! all $\equiv$ are mod 10.

Thought Experiment

Eve sees

Ciphertext	24	66	87	47	17	45	26	96

And thinks it is PAKISTAN.

So Eve thinks the following:

Text-Letter	P	A	K	I	S	T	A
Text-Digits	16	01	11	09	19	20	01
Key-Digits	$k_{11}k_{12}$	$k_{21}k_{22}$	$k_{31}k_{32}$	$k_{41}k_{42}$	$k_{51}k_{52}$	$k_{61}k_{62}$	$k_{71}k_{7}$
Ciphertext	24	66	87	47	17	45	26

Can Eve find the Key-Digits? Yes! all $\equiv$ are mod 10.

$1 + k_{11} \equiv 2$ so $k_{11} \equiv 2 - 1 \equiv 1$.

Thought Experiment

Eve sees

Ciphertext	24	66	87	47	17	45	26	96

And thinks it is PAKISTAN.

So Eve thinks the following:

Text-Letter	P	A	K	I	S	T	A
Text-Digits	16	01	11	09	19	20	01
Key-Digits	$k_{11}k_{12}$	$k_{21}k_{22}$	$k_{31}k_{32}$	$k_{41}k_{42}$	$k_{51}k_{52}$	$k_{61}k_{62}$	$k_{71}k_{7}$
Ciphertext	24	66	87	47	17	45	26

Can Eve find the Key-Digits? Yes! all $\equiv$ are mod 10.

$1 + k_{11} \equiv 2$ so $k_{11} \equiv 2 - 1 \equiv 1$.
$6 + k_{12} \equiv 4$ so $k_{12} \equiv 4 - 6 \equiv -2 \equiv 8$.
Etc.

Thought Experiment

Eve sees

Ciphertext	24	66	87	47	17	45	26	96

And thinks it is PAKISTAN.

So Eve thinks the following:

Text-Letter	P	A	K	I	S	T	A
Text-Digits	16	01	11	09	19	20	01
Key-Digits	$k_{11}k_{12}$	$k_{21}k_{22}$	$k_{31}k_{32}$	$k_{41}k_{42}$	$k_{51}k_{52}$	$k_{61}k_{62}$	$k_{71}k_{7}$
Ciphertext	24	66	87	47	17	45	26

Can Eve find the Key-Digits? Yes! all $\equiv$ are mod 10.

$1 + k_{11} \equiv 2$ so $k_{11} \equiv 2 - 1 \equiv 1$.
$6 + k_{12} \equiv 4$ so $k_{12} \equiv 4 - 6 \equiv -2 \equiv 8$.
Etc.

Next slide gives complete answer.

Thought Experiment Continued

Eve Thinks:

Thought Experiment Continued

Eve Thinks:

Text-Letter	P	A	K	I	S	T	A	N
Text-Digits	16	01	11	09	19	20	01	14
Ciphertext	24	66	87	47	17	45	26	96

Thought Experiment Continued

Eve Thinks:

Text-Letter	P	A	K	I	S	T	A	N
Text-Digits	16	01	11	09	19	20	01	14
Ciphertext	24	66	87	47	17	45	26	96

If Eve is correct then:

Thought Experiment Continued

Eve Thinks:

Text-Letter	P	A	K	I	S	T	A	N
Text-Digits	16	01	11	09	19	20	01	14
Ciphertext	24	66	87	47	17	45	26	96

If Eve is correct then:

Key–Digits	18	65	76	48	08	25	25	82

Thought Experiment Continued: Eve gets Equations

If Eve is correct then:

Thought Experiment Continued: Eve gets Equations

If Eve is correct then:

Key–Digits	18	65	76	48	08	25	25	82

Thought Experiment Continued: Eve gets Equations

If Eve is correct then:

Key–Digits	18	65	76	48	08	25	25	82

Since $x_{n+1} \equiv Ax_n + B \pmod{M}$

Thought Experiment Continued: Eve gets Equations

If Eve is correct then:

Key–Digits	18	65	76	48	08	25	25	82

Since $x_{n+1} \equiv Ax_n + B \pmod{M}$

$7648 \equiv 1865A + B \pmod{M}$

Thought Experiment Continued: Eve gets Equations

If Eve is correct then:

Key–Digits	18	65	76	48	08	25	25	82

Since $x_{n+1} \equiv Ax_n + B \pmod{M}$

$7648 \equiv 1865A + B \pmod{M}$

$825 \equiv 7648A + B \pmod{M}$

Thought Experiment Continued: Eve gets Equations

If Eve is correct then:

Key–Digits	18	65	76	48	08	25	25	82

Since $x_{n+1} \equiv Ax_n + B \pmod{M}$

$7648 \equiv 1865A + B \pmod{M}$

$825 \equiv 7648A + B \pmod{M}$

$2582 \equiv 825A + B \pmod{M}$

Thought Experiment Continued: Eve gets Equations

If Eve is correct then:

Key–Digits	18	65	76	48	08	25	25	82

Since $x_{n+1} \equiv Ax_n + B \pmod{M}$

$7648 \equiv 1865A + B \pmod{M}$

$825 \equiv 7648A + B \pmod{M}$

$2582 \equiv 825A + B \pmod{M}$

Can we solve these?

Thought Experiment Continued: Eve gets Equations

If Eve is correct then:

Key–Digits	18	65	76	48	08	25	25	82

Since $x_{n+1} \equiv Ax_n + B \pmod{M}$

$7648 \equiv 1865A + B \pmod{M}$

$825 \equiv 7648A + B \pmod{M}$

$2582 \equiv 825A + B \pmod{M}$

Can we solve these? Yes!

Thought Exp: Eve Can Find M (I)

EQ1: $7648 \equiv 1865A + B \pmod{M}$
EQ2: $825 \equiv 7648A + B \pmod{M}$
EQ3: $2582 \equiv 825A + B \pmod{M}$

Thought Exp: Eve Can Find M (I)

EQ1: $7648 \equiv 1865A + B \pmod{M}$
EQ2: $825 \equiv 7648A + B \pmod{M}$
EQ3: $2582 \equiv 825A + B \pmod{M}$

By looking at EQ2−EQ1 and EQ3−EQ1 get 2 equations and no B

Thought Exp: Eve Can Find M (I)

EQ1: $7648 \equiv 1865A + B \pmod{M}$
EQ2: $825 \equiv 7648A + B \pmod{M}$
EQ3: $2582 \equiv 825A + B \pmod{M}$

By looking at EQ2−EQ1 and EQ3−EQ1 get 2 equations and no B

EQ4: $-6823 \equiv 5783A \pmod{M}$
EQ5: $-5066 \equiv -1040A \pmod{M}$

Thought Exp: Eve can Find M (II)

EQ4: $-6823 \equiv 5783A \pmod{M}$
EQ5: $-5066 \equiv -1040A \pmod{M}$

Thought Exp: Eve can Find M (II)

EQ4: $-6823 \equiv 5783A \pmod{M}$
EQ5: $-5066 \equiv -1040A \pmod{M}$

Mult EQ4 by 1040 and EQ5 by 5783 to get:

EQ4': $-6823 \times 1040 \equiv \quad 5783 \times 1040 \times A \pmod{M}$
EQ5': $-5066 \times 5783 \equiv -1040 \times 5783 \times A \pmod{M}$

Thought Exp: Eve can Find M (II)

EQ4: $-6823 \equiv 5783A \pmod{M}$
EQ5: $-5066 \equiv -1040A \pmod{M}$

Mult EQ4 by 1040 and EQ5 by 5783 to get:

EQ4': $-6823 \times 1040 \equiv \quad 5783 \times 1040 \times A \pmod{M}$
EQ5': $-5066 \times 5783 \equiv -1040 \times 5783 \times A \pmod{M}$

We rewrite a bit:

Thought Exp: Eve can Find M (II)

EQ4: $-6823 \equiv 5783A \pmod{M}$
EQ5: $-5066 \equiv -1040A \pmod{M}$

Mult EQ4 by 1040 and EQ5 by 5783 to get:

EQ4': $-6823 \times 1040 \equiv \quad 5783 \times 1040 \times A \pmod{M}$
EQ5': $-5066 \times 5783 \equiv -1040 \times 5783 \times A \pmod{M}$

We rewrite a bit:

EQ4': $-7095920 \equiv \quad 5783 \times 1040 \times A \pmod{M}$
EQ5': $-29296678 \equiv -5783 \times 1040 \times A \pmod{M}$

Thought Exp: Eve can Find M (II)

EQ4: $-6823 \equiv 5783A \pmod{M}$
EQ5: $-5066 \equiv -1040A \pmod{M}$

Mult EQ4 by 1040 and EQ5 by 5783 to get:

EQ4': $-6823 \times 1040 \equiv \quad 5783 \times 1040 \times A \pmod{M}$
EQ5': $-5066 \times 5783 \equiv -1040 \times 5783 \times A \pmod{M}$

We rewrite a bit:

EQ4': $-7095920 \equiv \quad 5783 \times 1040 \times A \pmod{M}$
EQ5': $-29296678 \equiv -5783 \times 1040 \times A \pmod{M}$

Add EQ4' and EQ5' to get:

$$-36392598 \equiv 0 \pmod{M}$$

Can we use this?

Thought Exp: Eve can Find M (II)

EQ4: $-6823 \equiv 5783A \pmod{M}$
EQ5: $-5066 \equiv -1040A \pmod{M}$

Mult EQ4 by 1040 and EQ5 by 5783 to get:

EQ4': $-6823 \times 1040 \equiv \quad 5783 \times 1040 \times A \pmod{M}$
EQ5': $-5066 \times 5783 \equiv -1040 \times 5783 \times A \pmod{M}$

We rewrite a bit:

EQ4': $-7095920 \equiv \quad 5783 \times 1040 \times A \pmod{M}$
EQ5': $-29296678 \equiv -5783 \times 1040 \times A \pmod{M}$

Add EQ4' and EQ5' to get:

$$-36392598 \equiv 0 \pmod{M}$$

Can we use this? Yes We Can!

Thought Exp: Eve Finds M (III)

$$36392598 \equiv 0 \pmod{M}$$

Thought Exp: Eve Finds M (III)

$$36392598 \equiv 0 \pmod{M}$$

1. M divides 36392598.

Thought Exp: Eve Finds M (III)

$$36392598 \equiv 0 \pmod{M}$$

1. M divides 36392598.
2. M is 4 digits long.

Thought Exp: Eve Finds M (III)

$$36392598 \equiv 0 \pmod{M}$$

1. M divides 36392598.
2. M is 4 digits long.
3. The cipher used 7648, so $M > 7648$, hence $7649 \le M \le 9999$.

Hence a SMALL number of possibilities for M.

Thought Exp: Eve Finds M (III)

$$36392598 \equiv 0 \pmod{M}$$

1. M divides 36392598.
2. M is 4 digits long.
3. The cipher used 7648, so $M > 7648$, hence $7649 \leq M \leq 9999$.

Hence a SMALL number of possibilities for M.
Two ways to find possibilities for M on next few slides.

Eve Factors to Find M

Eve factors 36392598.

$$36392598 = 2 \times 3^3 \times 11 \times 197 \times 311$$

Eve Factors to Find M

Eve factors 36392598.

$36392598 = 2 \times 3^3 \times 11 \times 197 \times 311$
Factoring? Really? Eve has to Factor?

Eve Factors to Find M

Eve factors 36392598.

$36392598 = 2 \times 3^3 \times 11 \times 197 \times 311$
Factoring? Really? Eve has to Factor?
(Sarcastic) does she have a quantum computer?

Eve Factors to Find M

Eve factors 36392598.

$36392598 = 2 \times 3^3 \times 11 \times 197 \times 311$
Factoring? Really? Eve has to Factor?
(Sarcastic) does she have a quantum computer?
We will address this point later.

Eve Factors to Find M

Eve factors 36392598.

$36392598 = 2 \times 3^3 \times 11 \times 197 \times 311$
Factoring? Really? Eve has to Factor?
(Sarcastic) does she have a quantum computer?
We will address this point later.

1. M is a divisor of 36392598.

Eve Factors to Find M

Eve factors 36392598.

$36392598 = 2 \times 3^3 \times 11 \times 197 \times 311$
Factoring? Really? Eve has to Factor?
(Sarcastic) does she have a quantum computer?
We will address this point later.

1. M is a divisor of 36392598.
2. M is 4 digits long.

Eve Factors to Find M

Eve factors 36392598.

$36392598 = 2 \times 3^3 \times 11 \times 197 \times 311$
Factoring? Really? Eve has to Factor?
(Sarcastic) does she have a quantum computer?
We will address this point later.

1. M is a divisor of 36392598.
2. M is 4 digits long.
3. The cipher used 7648, so $M > 7648$.

Eve Can Crack It!–Finding M

$36392598 = 2 \times 3^3 \times 11 \times 197 \times 311$
M is a factor of 36392598 such that $7648 \leq M \leq 9999$.
How many factors of $2 \times 3^3 \times 11 \times 197 \times 311$?

Eve Can Crack It!–Finding M

$36392598 = 2 \times 3^3 \times 11 \times 197 \times 311$
M is a factor of 36392598 such that $7648 \leq M \leq 9999$.
How many factors of $2 \times 3^3 \times 11 \times 197 \times 311$?
$2 \times 4 \times 2 \times 2 \times 2 = 64$.

Eve Can Crack It!–Finding M

$36392598 = 2 \times 3^3 \times 11 \times 197 \times 311$
M is a factor of 36392598 such that $7648 \leq M \leq 9999$.
How many factors of $2 \times 3^3 \times 11 \times 197 \times 311$?
$2 \times 4 \times 2 \times 2 \times 2 = 64$.

1. Can't use 197 AND 311: $197 \times 311 = 61267 > 9999$.

Eve Can Crack It!–Finding M

$36392598 = 2 \times 3^3 \times 11 \times 197 \times 311$
M is a factor of 36392598 such that $7648 \le M \le 9999$.
How many factors of $2 \times 3^3 \times 11 \times 197 \times 311$?
$2 \times 4 \times 2 \times 2 \times 2 = 64$.

1. Can't use 197 AND 311: $197 \times 311 = 61267 > 9999$.
2. Could continue to do this by hand.

Eve Can Crack It!–Finding M

$36392598 = 2 \times 3^3 \times 11 \times 197 \times 311$
M is a factor of 36392598 such that $7648 \leq M \leq 9999$.
How many factors of $2 \times 3^3 \times 11 \times 197 \times 311$?
$2 \times 4 \times 2 \times 2 \times 2 = 64$.

1. Can't use 197 AND 311: $197 \times 311 = 61267 > 9999$.
2. Could continue to do this by hand.

   We won't—**we are busy people** and we have computers to do it for us.

Eve Can Crack It!–Finding M

$36392598 = 2 \times 3^3 \times 11 \times 197 \times 311$
M is a factor of 36392598 such that $7648 \leq M \leq 9999$.
How many factors of $2 \times 3^3 \times 11 \times 197 \times 311$?
$2 \times 4 \times 2 \times 2 \times 2 = 64$.

1. Can't use 197 AND 311: $197 \times 311 = 61267 > 9999$.
2. Could continue to do this by hand.

   We won't—**we are busy people** and we have computers to do it for us.

   The original article did do it by hand. It was written in 1977.

Eve Can Crack It!–Finding M

$36392598 = 2 \times 3^3 \times 11 \times 197 \times 311$
M is a factor of 36392598 such that $7648 \le M \le 9999$.
How many factors of $2 \times 3^3 \times 11 \times 197 \times 311$?
$2 \times 4 \times 2 \times 2 \times 2 = 64$.

1. Can't use 197 AND 311: $197 \times 311 = 61267 > 9999$.
2. Could continue to do this by hand.

   We won't—**we are busy people** and we have computers to do it for us.

   The original article did do it by hand. It was written in 1977.

   The next slide shows how to do it by hand. We won't go over it, but you can if you want.

Eve Can Crack It!–Finding M OLD WAY

THIS SLIDE IS OPTIONAL

$36392598 = 2 \times 3^3 \times 11 \times 197 \times 311$

M is a factor of 36392598 such that $7648 \leq M \leq 9999$.

How many factors of $2 \times 3^3 \times 11 \times 197 \times 311$?

Eve Can Crack It!–Finding M OLD WAY

THIS SLIDE IS OPTIONAL

$36392598 = 2 \times 3^3 \times 11 \times 197 \times 311$

M is a factor of 36392598 such that $7648 \leq M \leq 9999$.

How many factors of $2 \times 3^3 \times 11 \times 197 \times 311$?

$2 \times 4 \times 2 \times 2 \times 2 = 64$.

Eve Can Crack It!–Finding M OLD WAY

THIS SLIDE IS OPTIONAL

$36392598 = 2 \times 3^3 \times 11 \times 197 \times 311$

M is a factor of 36392598 such that $7648 \le M \le 9999$.

How many factors of $2 \times 3^3 \times 11 \times 197 \times 311$?

$2 \times 4 \times 2 \times 2 \times 2 = 64$.

1. Can't use 197 AND 311: $197 \times 311 = 61267 > 9999$.
2. If use 311 then need a 3: $2 \times 11 \times 311 = 6842 < 7648$.
3. If use 311 and exactly one 3 does not work:
   (a) Use 2 but not 11: $311 \times 3 \times 2 = 1866 < 7648$
   (b) Use 11: $\ge 311 \times 3 \times 11 = 10263 > 9999$.
4. If use 311, at least two 3's, and 11:
   $311 \times 11 \times 9 = 30789 > 9999$.
5. If use 311 and 9 does not work: $311 \times 2 \times 9 = 5598 < 7648$.
6. If use 311 and 27: $311 \times 27 = 8397$. WORKS!
7. Leave it to you to show that using 197 does not work.
8. So $M = \mathbf{8397}$.

How to do it in 2021

Recall

M is a factor of 36392598 such that $7648 \leq M \leq 9999$.

How to do it in 2021

Recall
M is a factor of 36392598 such that $7648 \leq M \leq 9999$.

$$36392598 = 2 \times 3^3 \times 11 \times 197 \times 311$$

How to do it in 2021

Recall
M is a factor of 36392598 such that $7648 \leq M \leq 9999$.

$$36392598 = 2 \times 3^3 \times 11 \times 197 \times 311$$

36392598 has $2 \times 4 \times 2 \times 2 \times 2 = 64$ factors.
Two ways to find **possibilities for M**

1. Look at all 64 factors and see which ones are in $[7648, 9999]$.

How to do it in 2021

Recall

M is a factor of 36392598 such that $7648 \leq M \leq 9999$.

$$36392598 = 2 \times 3^3 \times 11 \times 197 \times 311$$

36392598 has $2 \times 4 \times 2 \times 2 \times 2 = 64$ factors.

Two ways to find **possibilities for M**

1. Look at all 64 factors and see which ones are in $[7648, 9999]$.
2. Even less clever: Look at ALL numbers in $[7648, 9999]$ and see which ones are factors of M.

Reflect

If we do this we find that the only candidate that works is $M = 8397$.

Reflect

If we do this we find that the only candidate that works is $M = 8397$.

We might have found **no** M works. So Eve was wrong.

Reflect

If we do this we find that the only candidate that works is $M = 8397$.

We might have found **no** M works. So Eve was wrong.

We might have found **several** M works. In that case, do what is on the next few slides with each one.

Eve Determines Which M Is Correct, If Any

EQ4: $-6823 \equiv 5783A \pmod{M}$
By either brute force of cleverness we found that
If Eve's Guess Is Correct then $M = 8397$.

EQ4: $-6823 \equiv 5783A \pmod{8397}$

Eve Determines Which M Is Correct, If Any

EQ4: $-6823 \equiv 5783A \pmod{M}$
By either brute force of cleverness we found that
If Eve's Guess Is Correct then $M = 8397$.

EQ4: $-6823 \equiv 5783A \pmod{8397}$
Use Euclid algorithm to find that $5783^{-1} \pmod{8397} \equiv 1982$.

Eve Determines Which M Is Correct, If Any

EQ4: $-6823 \equiv 5783A \pmod{M}$
By either brute force of cleverness we found that
If Eve's Guess Is Correct then $M = 8397$.

EQ4: $-6823 \equiv 5783A \pmod{8397}$
Use Euclid algorithm to find that $5783^{-1} \pmod{8397} \equiv 1982$.
Reflect It is possible the inverse does not exist. Then Eve is wrong. In the case at hand, the inverse exists.

Eve Determines Which M Is Correct, If Any

EQ4: $-6823 \equiv 5783A \pmod{M}$
By either brute force of cleverness we found that
If Eve's Guess Is Correct then $M = 8397$.

EQ4: $-6823 \equiv 5783A \pmod{8397}$
Use Euclid algorithm to find that $5783^{-1} \pmod{8397} \equiv 1982$.
Reflect It is possible the inverse does not exist. Then Eve is wrong. In the case at hand, the inverse exists.
Multiply both sides of EQ4 by 1982 to get:

$$-6823 \times 1982 \equiv A \pmod{8397}$$

$$A \equiv -6823 \times 1982 \equiv \mathbf{4381} \pmod{8397}$$

Eve Checks M

Now want to find B. Recall:

Eve Checks M

Now want to find B. Recall:

EQ1: $7648 \equiv 1865A + B \pmod{M}$

Eve Checks M

Now want to find B. Recall:

EQ1: $7648 \equiv 1865A + B \pmod{M}$

By plugging in $M = 8397$ and $A = 4381$ we get

$$7648 \equiv 1865 * 4381 + B \pmod{8397}$$

Eve Checks M

Now want to find B. Recall:

EQ1: $7648 \equiv 1865A + B \pmod{M}$

By plugging in $M = 8397$ and $A = 4381$ we get

$$7648 \equiv 1865 * 4381 + B \pmod{8397}$$

$$B \equiv 7648 - 1865 * 4381 \equiv \mathbf{7364} \pmod{8397}$$

Eve Checks M

Now want to find B. Recall:

EQ1: $7648 \equiv 1865A + B \pmod{M}$

By plugging in $M = 8397$ and $A = 4381$ we get

$$7648 \equiv 1865 * 4381 + B \pmod{8397}$$

$$B \equiv 7648 - 1865 * 4381 \equiv \mathbf{7364} \pmod{8397}$$

Upshot **If Eve's Guess Is Correct Then** $A = 4381$, $B = 7364$, $M = 8397$.

Eve Can Find x_0

Eve wants to test $A = 4381, B = 7634, M = 8397$.

Eve Can Find x_0

Eve wants to test $A = 4381, B = 7634, M = 8397$.

$$x_{n+1} \equiv 4381 x_n + 7364 \pmod{8397}$$

Eve Can Find x_0

Eve wants to test $A = 4381, B = 7634, M = 8397$.

$$x_{n+1} \equiv 4381x_n + 7364 \pmod{8397}$$

Need x_0.

Eve Can Find x_0

Eve wants to test $A = 4381, B = 7634, M = 8397$.

$$x_{n+1} \equiv 4381x_n + 7364 \pmod{8397}$$

Need x_0.

4381 is rel prime to 8397 so $(4381)^{-1} \pmod{8397}$ exists.
It is 8374. Mult equation by 8374.

Eve Can Find x_0

Eve wants to test $A = 4381, B = 7634, M = 8397.$

$$x_{n+1} \equiv 4381 x_n + 7364 \pmod{8397}$$

Need x_0.

4381 is rel prime to 8397 so $(4381)^{-1} \pmod{8397}$ exists.
It is 8374. Mult equation by 8374.

$$8374 x_{n+1} \equiv 8374 * 4381 x_n + 8374 * 7364 \pmod{8397}$$

Eve Can Find x_0

Eve wants to test $A = 4381, B = 7634, M = 8397$.

$$x_{n+1} \equiv 4381x_n + 7364 \pmod{8397}$$

Need x_0.

4381 is rel prime to 8397 so $(4381)^{-1} \pmod{8397}$ exists.
It is 8374. Mult equation by 8374.

$$8374x_{n+1} \equiv 8374 * 4381x_n + 8374 * 7364 \pmod{8397}$$

$$8374x_{n+1} \equiv x_n + 6965 \pmod{8397}$$

Eve Can Find x_0

Eve wants to test $A = 4381, B = 7634, M = 8397$.

$$x_{n+1} \equiv 4381x_n + 7364 \pmod{8397}$$

Need x_0.

4381 is rel prime to 8397 so $(4381)^{-1} \pmod{8397}$ exists.
It is 8374. Mult equation by 8374.

$$8374x_{n+1} \equiv 8374 * 4381x_n + 8374 * 7364 \pmod{8397}$$

$$8374x_{n+1} \equiv x_n + 6965 \pmod{8397}$$

$$x_n \equiv 8374x_{n+1} - 6965 \equiv 8374x_{n+1} + 1432$$

How will this help us?

Eve Finds x_0 (cont)

$$x_n \equiv 8374x_{n+1} + 1432$$

Eve Finds x_0 (cont)

$$x_n \equiv 8374 x_{n+1} + 1432$$

PAKISTAN had the P on the (say) 191st spot. We know the key at 191 spot. Hence can use recurrence above to get key at 190th, 189th, ..., 0th spot.

Eve Finds x_0 (cont)

$$x_n \equiv 8374x_{n+1} + 1432$$

PAKISTAN had the P on the (say) 191st spot. We know the key at 191 spot. Hence can use recurrence above to get key at 190th, 189th, . . ., 0th spot.

So can get x_0.

Eve Finds x_0 (cont)

$$x_n \equiv 8374x_{n+1} + 1432$$

PAKISTAN had the P on the (say) 191st spot. We know the key at 191 spot. Hence can use recurrence above to get key at 190th, 189th, ..., 0th spot.

So can get x_0.

Are we done yet? No.

Eve Uses Is-English

Eve has x_0, A, B, M so Eve can generate the **entire** key.

Eve Uses Is-English

Eve has x_0, A, B, M so Eve can generate the **entire** key.

She uses it to recover the **entire** plaintext.

Eve Uses Is-English

Eve has x_0, A, B, M so Eve can generate the **entire** key.

She uses it to recover the **entire** plaintext.

Use IS-ENGLISH.

Eve Uses Is-English

Eve has x_0, A, B, M so Eve can generate the **entire** key.

She uses it to recover the **entire** plaintext.

Use IS-ENGLISH.

If Eve's Guess Is Correct then it will return YES-IS ENGLISH.
So Eve is done!

Eve Uses Is-English

Eve has x_0, A, B, M so Eve can generate the **entire** key.

She uses it to recover the **entire** plaintext.

Use IS-ENGLISH.

If Eve's Guess Is Correct then it will return YES-IS ENGLISH. So Eve is done!

If Eve's Guess Is Not Correct then either the procedure would have failed long before this point OR we find ISNOT-English.

But This Was All Predicated on Eve's Guess

We just showed that **IF** Eve thinks that PAKISTAN occurred in (say) spaces 190 to 197 then:

But This Was All Predicated on Eve's Guess

We just showed that **IF** Eve thinks that PAKISTAN occurred in (say) spaces 190 to 197 then:

1. She can test if the guess is correct.

But This Was All Predicated on Eve's Guess

We just showed that **IF** Eve thinks that PAKISTAN occurred in (say) spaces 190 to 197 then:

1. She can test if the guess is correct.
2. If the guess is correct then she can find A, B, M, x_0 and decode the message

But This Was All Predicated on Eve's Guess

We just showed that **IF** Eve thinks that PAKISTAN occurred in (say) spaces 190 to 197 then:

1. She can test if the guess is correct.
2. If the guess is correct then she can find A, B, M, x_0 and decode the message

How can Eve use this to break the cipher?

But This Was All Predicated on Eve's Guess

We just showed that **IF** Eve thinks that PAKISTAN occurred in (say) spaces 190 to 197 then:

1. She can test if the guess is correct.
2. If the guess is correct then she can find A, B, M, x_0 and decode the message

How can Eve use this to break the cipher?
For **every** 8-letter sequence Eve **guess's** that it is PAKISTAN and does out the procedure above.

But This Was All Predicated on Eve's Guess

We just showed that **IF** Eve thinks that PAKISTAN occurred in (say) spaces 190 to 197 then:

1. She can test if the guess is correct.
2. If the guess is correct then she can find A, B, M, x_0 and decode the message

How can Eve use this to break the cipher?
For **every** 8-letter sequence Eve **guess's** that it is PAKISTAN and does out the procedure above.

Most of the time she will be wrong. But the one time she is right, she will have decoded the message.

Putting it All Together

Putting it All Together

1. Input is long ciphertext T that Eve knows was coded with recurrence. Eve knows a word w that she is sure appears in the text and is L letters.

Putting it All Together

1. Input is long ciphertext T that Eve knows was coded with recurrence. Eve knows a word w that she is sure appears in the text and is L letters.
2. For EVERY L-letter seq Eve does the following:

Putting it All Together

1. Input is long ciphertext T that Eve knows was coded with recurrence. Eve knows a word w that she is sure appears in the text and is L letters.
2. For EVERY L-letter seq Eve does the following:
   2.1 Assuming L-letter seq is w form equations and try to solve them. If can't then goto next L-letter seq.

Putting it All Together

1. Input is long ciphertext T that Eve knows was coded with recurrence. Eve knows a word w that she is sure appears in the text and is L letters.
2. For EVERY L-letter seq Eve does the following:
   2.1 Assuming L-letter seq is w form equations and try to solve them. If can't then goto next L-letter seq.
   2.2 Use A, B, M, x_0 to generate **entire** key. Decode **entire** text. If IS-ENGLISH=YES, DONE! Else goto next L-let-seq.

About Eve Factoring Fast

Eve had to factor:

$$36,392,598 = 2 \times 3^3 \times 11 \times 197 \times 311$$

About Eve Factoring Fast

Eve had to factor:

$$36,392,598 = 2 \times 3^3 \times 11 \times 197 \times 311$$

We usually say

Factoring is Hard

About Eve Factoring Fast

Eve had to factor:

$$36,392,598 = 2 \times 3^3 \times 11 \times 197 \times 311$$

We usually say

Factoring is Hard

But what do we mean by **Factoring is Hard** ?

About Eve Factoring Fast

Eve had to factor:

$$36,392,598 = 2 \times 3^3 \times 11 \times 197 \times 311$$

We usually say

Factoring is Hard

But what do we mean by **Factoring is Hard** ?

1. If **Alice** picks two **primes** p, q of length n and picks $N = pq$ then factoring N is hard.

About Eve Factoring Fast

Eve had to factor:

$$36,392,598 = 2 \times 3^3 \times 11 \times 197 \times 311$$

We usually say

Factoring is Hard

But what do we mean by **Factoring is Hard** ?

1. If **Alice** picks two **primes** p, q of length n and picks $N = pq$ then factoring N is hard.
2. If a **random** number is given then half the time it's even. A third of the time is divided by 3. Not so hard to factor.

About Eve Factoring Fast

Eve had to factor:

$$36,392,598 = 2 \times 3^3 \times 11 \times 197 \times 311$$

We usually say

Factoring is Hard

But what do we mean by **Factoring is Hard** ?

1. If **Alice** picks two **primes** p, q of length n and picks $N = pq$ then factoring N is hard.
2. If a **random** number is given then half the time it's even. A third of the time is divided by 3. Not so hard to factor.

Our scenario is closer to **random** than to **Alice** .

Some Real World Notes

Some Real World Notes

1. Java and other langs use an LCG with some A, B, M. Actually the M is always 2^{32} or 2^{64}. This makes the LCG even easier to crack.

Some Real World Notes

1. Java and other langs use an LCG with some A, B, M. Actually the M is always 2^{32} or 2^{64}. This makes the LCG even easier to crack.
2. Python and other modern language use **The Mersenne Twister** to generate random numbers. It is also not secure. (I will discuss it very soon.)

Some Real World Notes

1. Java and other langs use an LCG with some A, B, M. Actually the M is always 2^{32} or 2^{64}. This makes the LCG even easier to crack.
2. Python and other modern language use **The Mersenne Twister** to generate random numbers. It is also not secure. (I will discuss it very soon.)
3. Why do Java and Python and other langs have such bad random number generators?

Some Real World Notes

1. Java and other langs use an LCG with some A, B, M. Actually the M is always 2^{32} or 2^{64}. This makes the LCG even easier to crack.
2. Python and other modern language use **The Mersenne Twister** to generate random numbers. It is also not secure. (I will discuss it very soon.)
3. Why do Java and Python and other langs have such bad random number generators?

Some Real World Notes

1. Java and other langs use an LCG with some A, B, M. Actually the M is always 2^{32} or 2^{64}. This makes the LCG even easier to crack.
2. Python and other modern language use **The Mersenne Twister** to generate random numbers. It is also not secure. (I will discuss it very soon.)
3. Why do Java and Python and other langs have such bad random number generators?
    3.1 They are bad for crypto.

Some Real World Notes

1. Java and other langs use an LCG with some A, B, M. Actually the M is always 2^{32} or 2^{64}. This makes the LCG even easier to crack.
2. Python and other modern language use **The Mersenne Twister** to generate random numbers. It is also not secure. (I will discuss it very soon.)
3. Why do Java and Python and other langs have such bad random number generators?
   3.1 They are bad for crypto.
   3.2 They are fine for randomized algorithms (like quicksort).

Mersenne Twister

We do a very small example with a smaller word size than is used. The **Mersenne Twister** generates a sequence of 10-bit numbers (two 5-bit numbers, so for us 2 numbers in $\{01, \ldots, 26\}$).

Mersenne Twister

We do a very small example with a smaller word size than is used. The **Mersenne Twister** generates a sequence of 10-bit numbers (two 5-bit numbers, so for us 2 numbers in $\{01, \ldots, 26\}$).

We give an example:

Params: **7** ,**5** ,**3** ,**5** ,**3** ,$x_0, \ldots, x_6$, unknown to Eve.

Mersenne Twister

We do a very small example with a smaller word size than is used. The **Mersenne Twister** generates a sequence of 10-bit numbers (two 5-bit numbers, so for us 2 numbers in $\{01, \ldots, 26\}$).

We give an example:
Params: **7** ,**5** ,**3** ,**5** ,**3** ,$x_0, \ldots, x_6$, unknown to Eve.

$$x_{n+\mathbf{7}} = x_{n+\mathbf{5}} \oplus f(x_n^{\text{first}\mathbf{3}\text{bits}} x_{n+1}^{\text{last}\mathbf{5}\text{bits}})$$

f shifts bits **3** to the left (its more complicated).

Mersenne Twister

We do a very small example with a smaller word size than is used. The **Mersenne Twister** generates a sequence of 10-bit numbers (two 5-bit numbers, so for us 2 numbers in $\{01, \ldots, 26\}$).

We give an example:
Params: **7** ,**5** ,**3** ,**5** ,**3** ,$x_0, \ldots, x_6$, unknown to Eve.

$$x_{n+7} = x_{n+5} \oplus f(x_n^{\text{first}3\text{bits}} x_{n+1}^{\text{last}5\text{bits}})$$

f shifts bits **3** to the left (its more complicated).

1. Very fast since $\oplus$ and concat and shift are fast.

Mersenne Twister

We do a very small example with a smaller word size than is used. The **Mersenne Twister** generates a sequence of 10-bit numbers (two 5-bit numbers, so for us 2 numbers in $\{01, \ldots, 26\}$).

We give an example:

Params: **7** ,**5** ,**3** ,**5** ,**3** ,$x_0, \ldots, x_6$, unknown to Eve.

$$x_{n+\mathbf{7}} = x_{n+\mathbf{5}} \oplus f(x_n^{\mathrm{first}\mathbf{3}\mathrm{bits}} x_{n+1}^{\mathrm{last}\mathbf{5}\mathrm{bits}})$$

f shifts bits **3** to the left (its more complicated).

1. Very fast since $\oplus$ and concat and shift are fast.
2. Has same problem for crypto that LCG does: its a recurrence. Can guess that a word or phrase is in the text.

Mersenne Twister

We do a very small example with a smaller word size than is used. The **Mersenne Twister** generates a sequence of 10-bit numbers (two 5-bit numbers, so for us 2 numbers in $\{01, \ldots, 26\}$).

We give an example:
Params: **7** ,**5** ,**3** ,**5** ,**3** ,$x_0, \ldots, x_6$, unknown to Eve.

$$x_{n+\mathbf{7}} = x_{n+\mathbf{5}} \oplus f(x_n^{\text{first}\mathbf{3}\text{bits}} x_{n+1}^{\text{last}\mathbf{5}\text{bits}})$$

f shifts bits **3** to the left (its more complicated).

1. Very fast since $\oplus$ and concat and shift are fast.
2. Has same problem for crypto that LCG does: its a recurrence. Can guess that a word or phrase is in the text.
3. Would need to be a very long phrase so that the recurrence produces equations.

Mersenne Twister

We do a very small example with a smaller word size than is used. The **Mersenne Twister** generates a sequence of 10-bit numbers (two 5-bit numbers, so for us 2 numbers in $\{01, \ldots, 26\}$).

We give an example:
Params: **7** ,**5** ,**3** ,**5** ,**3** ,$x_0, \ldots, x_6$, unknown to Eve.

$$x_{n+\mathbf{7}} = x_{n+\mathbf{5}} \oplus f(x_n^{\text{first}\mathbf{3}\text{bits}} x_{n+1}^{\text{last}\mathbf{5}\text{bits}})$$

f shifts bits **3** to the left (its more complicated).

1. Very fast since $\oplus$ and concat and shift are fast.
2. Has same problem for crypto that LCG does: its a recurrence. Can guess that a word or phrase is in the text.
3. Would need to be a very long phrase so that the recurrence produces equations.
4. The larger the parameter which we have as 7, the longer the phrase has to be.

Mersenne Twister Example with Digits

Text-Letter	P	A	K	I	S	T	A	N	B	O
Text-Digits	16	01	11	09	19	20	01	14	02	15
Cipher-text	24	66	87	47	17	45	26	96	06	11
Key	18	65	76	48	08	25	25	82	04	04

Mersenne Twister Example with Digits

Text-Letter	P	A	K	I	S	T	A	N	B	O
Text-Digits	16	01	11	09	19	20	01	14	02	15
Cipher-text	24	66	87	47	17	45	26	96	06	11
Key	18	65	76	48	08	25	25	82	04	04

Text-Letter	R	D	E	R	S	I	N	D	I	A
Text-Digits	18	04	05	18	19	09	14	04	09	01
Cipher-text	23	16	01	11	09	19	20	01	14	02
Key	95	12	04	03	90	10	16	07	15	09

Mersenne Twister Example with Digits

Text-Letter	P	A	K	I	S	T	A	N	B	O
Text-Digits	16	01	11	09	19	20	01	14	02	15
Cipher-text	24	66	87	47	17	45	26	96	06	11
Key	18	65	76	48	08	25	25	82	04	04

Text-Letter	R	D	E	R	S	I	N	D	I	A
Text-Digits	18	04	05	18	19	09	14	04	09	01
Cipher-text	23	16	01	11	09	19	20	01	14	02
Key	95	12	04	03	90	10	16	07	15	09

Eve will guess the 7 and 5, does not know f, a, b

$$x_{n+7} = x_{n+5} \oplus f(x_n^{\text{first a digs}} x_{n+1}^{\text{last b digs}})$$

Mersenne Twister Example with Digits

Text-Letter	P	A	K	I	S	T	A	N	B	O
Text-Digits	16	01	11	09	19	20	01	14	02	15
Cipher-text	24	66	87	47	17	45	26	96	06	11
Key	18	65	76	48	08	25	25	82	04	04

Text-Letter	R	D	E	R	S	I	N	D	I	A
Text-Digits	18	04	05	18	19	09	14	04	09	01
Cipher-text	23	16	01	11	09	19	20	01	14	02
Key	95	12	04	03	90	10	16	07	15	09

Eve will guess the 7 and 5, does not know f, a, b

$$x_{n+7} = x_{n+5} \oplus f(x_n^{\text{first a digs}} x_{n+1}^{\text{last b digs}})$$

$$1509 = 9010 \oplus f(0825^{\text{first a digs}}, 2528^{\text{last b digs}})$$

Mersenne Twister Example with Digits

Text-Letter	P	A	K	I	S	T	A	N	B	O
Text-Digits	16	01	11	09	19	20	01	14	02	15
Cipher-text	24	66	87	47	17	45	26	96	06	11
Key	18	65	76	48	08	25	25	82	04	04

Text-Letter	R	D	E	R	S	I	N	D	I	A
Text-Digits	18	04	05	18	19	09	14	04	09	01
Cipher-text	23	16	01	11	09	19	20	01	14	02
Key	95	12	04	03	90	10	16	07	15	09

Eve will guess the 7 and 5, does not know f, a, b

$$x_{n+7} = x_{n+5} \oplus f(x_n^{\text{first a digs}} x_{n+1}^{\text{last b digs}})$$

$1509 = 9010 \oplus f(0825^{\text{first a digs}}, 2528^{\text{last b digs}})$
$1607 = 0403 \oplus f(7648^{\text{first a digs}}, 4808^{\text{last b digs}})$

Mersenne Twister Example with Digits

Text-Letter	P	A	K	I	S	T	A	N	B	O
Text-Digits	16	01	11	09	19	20	01	14	02	15
Cipher-text	24	66	87	47	17	45	26	96	06	11
Key	18	65	76	48	08	25	25	82	04	04

Text-Letter	R	D	E	R	S	I	N	D	I	A
Text-Digits	18	04	05	18	19	09	14	04	09	01
Cipher-text	23	16	01	11	09	19	20	01	14	02
Key	95	12	04	03	90	10	16	07	15	09

Eve will guess the 7 and 5, does not know f, a, b

$$x_{n+7} = x_{n+5} \oplus f(x_n^{\text{first a digs}} x_{n+1}^{\text{last b digs}})$$

$1509 = 9010 \oplus f(0825^{\text{first a digs}}, 2528^{\text{last b digs}})$
$1607 = 0403 \oplus f(7648^{\text{first a digs}}, 4808^{\text{last b digs}})$
$9010 = 9512 \oplus f(1865^{\text{first a digs}}, 6576^{\text{last b digs}})$

Mersenne Twister Example with Digits

Text-Letter	P	A	K	I	S	T	A	N	B	O
Text-Digits	16	01	11	09	19	20	01	14	02	15
Cipher-text	24	66	87	47	17	45	26	96	06	11
Key	18	65	76	48	08	25	25	82	04	04

Text-Letter	R	D	E	R	S	I	N	D	I	A
Text-Digits	18	04	05	18	19	09	14	04	09	01
Cipher-text	23	16	01	11	09	19	20	01	14	02
Key	95	12	04	03	90	10	16	07	15	09

Eve will guess the 7 and 5, does not know f, a, b

$$x_{n+7} = x_{n+5} \oplus f(x_n^{\text{first a digs}} x_{n+1}^{\text{last b digs}})$$

$1509 = 9010 \oplus f(0825^{\text{first a digs}}, 2528^{\text{last b digs}})$
$1607 = 0403 \oplus f(7648^{\text{first a digs}}, 4808^{\text{last b digs}})$
$9010 = 9512 \oplus f(1865^{\text{first a digs}}, 6576^{\text{last b digs}})$

Can use recurrences to find f, a, b.

Mersenne Twister Example with Digits

Text-Letter	P	A	K	I	S	T	A	N	B	O
Text-Digits	16	01	11	09	19	20	01	14	02	15
Cipher-text	24	66	87	47	17	45	26	96	06	11
Key	18	65	76	48	08	25	25	82	04	04

Text-Letter	R	D	E	R	S	I	N	D	I	A
Text-Digits	18	04	05	18	19	09	14	04	09	01
Cipher-text	23	16	01	11	09	19	20	01	14	02
Key	95	12	04	03	90	10	16	07	15	09

Eve will guess the 7 and 5, does not know f, a, b

$$x_{n+7} = x_{n+5} \oplus f(x_n^{\text{first a digs}} x_{n+1}^{\text{last b digs}})$$

$1509 = 9010 \oplus f(0825^{\text{first a digs}}, 2528^{\text{last b digs}})$
$1607 = 0403 \oplus f(7648^{\text{first a digs}}, 4808^{\text{last b digs}})$
$9010 = 9512 \oplus f(1865^{\text{first a digs}}, 6576^{\text{last b digs}})$

Can use recurrences to find f, a, b.Will need more equations and some guesswork, but crackable!

Upshot

Any pseudo-random generator that is based on recurrences is crackable.