BILL, RECORD LECTURE!!!!

BILL RECORD LECTURE!!!

Go Over Problems 4 and 6 from HW 01

October 10, 2020

The One-Time Pad and Trying to Fake It—and Failing to

October 10, 2020

The One-Time Pad

October 10, 2020

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 10001010001000111110111100

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 10001010001000111110111100

Alice wants to send Bob 1110.

She sends $1110 \oplus 1000 = 0110$.

Then Bob wants to send Alice 00111.

He sends $00111 \oplus 10100 = 10011$.

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!

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!
2. **CON** If Key is N bits long can only send N bits.

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!
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 10001010001000111110111110 0

Alice wants to send Bob 1110.

She sends $1110 \oplus 1000 = 0110$.

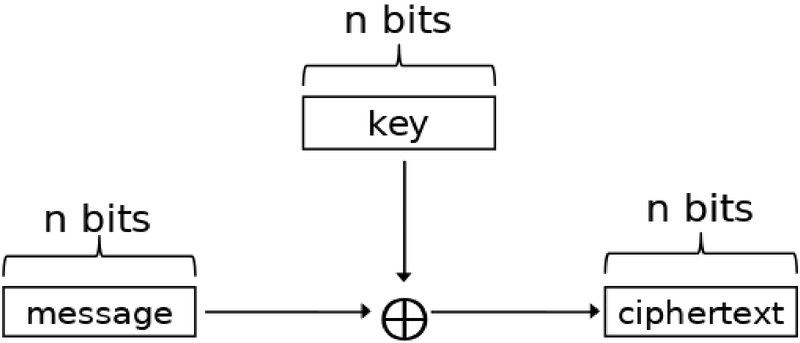
Then Bob wants to send Alice 00111.

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



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 perfectly secret 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 Just use the Random function in Java!

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 Just use the Random function in Java!

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 Just use the Random function in Java!

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 Just use the Random function in Java!

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 Just use the Random function in Java!

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 most languages) 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 most languages) 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 most languages) 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 most languages) 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 most languages) 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 most languages) 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 most languages) 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.

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

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

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

Our Running Example

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

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

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

Our Running Example

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

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

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

We will then crack it.

Our Running Example

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

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

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, B rel prime.

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 get a long random (looking?) sequence? Next slide.

Use Rec. x_0, A, B, M is Short Private Key

(Example from *"Cracking" a Random Number Generator by James Reed.* Paper on Course Website.)

Use Rec. x_0, A, B, M is Short Private Key

(Example from *"Cracking" a Random Number Generator by James Reed.* Paper on Course Website.)

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

Use Rec. x_0, A, B, M is Short Private Key

(Example from *"Cracking" a Random Number Generator by James Reed.* Paper on Course Website.)

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

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

Use Rec. x_0, A, B, M is Short Private Key

(Example from *"Cracking" a Random Number Generator by James Reed.* Paper on Course Website.)

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

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

We show that this random-looking sequence is NOT that random and, if used for a psuedo-one-time-pad, can be cracked.

Example 1

Example 1

$x_0 = 2134$
$x_1 = 2160$
$x_2 = 6905$
$x_3 = 3778$
They start with x_1.

Example 1

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

Example 1

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

Example 2

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

Example 2

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

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

Example 2

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.

Example 2

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.

Example 2

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.

Example 2

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

Eve Can Crack It!—Looks at ALL 8-letter Seq

For **every** 8-long sequence of letters, Eve spectates that its PAKISTAN

Eve Can Crack It!—Looks at ALL 8-letter Seq

For **every** 8-long sequence of letters, Eve spectates that its PAKISTAN

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

Eve Can Crack It!—Looks at ALL 8-letter Seq

For **every** 8-long sequence of letters, Eve spectates that its PAKISTAN

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's guess is correct then:

Key–Digits	18	65	76	48	08	25	25	82

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

Eve Can Crack It!—Looks at ALL 8-letter Seq

For **every** 8-long sequence of letters, Eve spectates that its PAKISTAN

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's guess is correct then:

Key–Digits	18	65	76	48	08	25	25	82

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

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

Eve Can Crack It!—Looks at ALL 8-letter Seq

For **every** 8-long sequence of letters, Eve spectates that its PAKISTAN

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's guess is correct then:

Key–Digits	18	65	76	48	08	25	25	82

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

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

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

Eve Can Crack It!—Looks at ALL 8-letter Seq

For **every** 8-long sequence of letters, Eve spectates that its PAKISTAN

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's guess is correct then:

Key–Digits	18	65	76	48	08	25	25	82

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

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

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

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

Eve Can Crack It!—Looks at ALL 8-letter Seq

For **every** 8-long sequence of letters, Eve spectates that its PAKISTAN

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's guess 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? (The title **Eve Can Crack It!** gives it away!)

Fortunately PAKISTAN Has 8 Letters

8 letters lead to 3 equations.

Fortunately PAKISTAN Has 8 Letters

8 letters lead to 3 equations.

More letters would lead to more equations. This is good since may find they are unsolvable quickly.

Fortunately PAKISTAN Has 8 Letters

8 letters lead to 3 equations.

More letters would lead to more equations. This is good since may find they are unsolvable quickly.

Less letters would lead to less equations. This is bad since may have to look at many false positives.

Fortunately PAKISTAN Has 8 Letters

8 letters lead to 3 equations.

More letters would lead to more equations. This is good since may find they are unsolvable quickly.

Less letters would lead to less equations. This is bad since may have to look at many false positives.

Leave as an exercise how many equations.

Eve Can Crack It!—Finding M (I)

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

Eve Can Crack It!—Finding 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

Eve Can Crack It!—Finding 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: $-5066 \equiv -1040A \pmod{M}$

Eve Can Crack It!—Finding M (II)

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

Eve Can Crack It!—Finding 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': $-5066 \times 5783 \equiv -1040 \times 5783 \times A \pmod{M}$

Eve Can Crack It!—Finding 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:

Eve Can Crack It!—Finding 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': $-29296678 \equiv -5783 \times 1040 \times A \pmod{M}$

Eve Can Crack It!—Finding 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?

Eve Can Crack It!—Finding 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!

Eve Can Crack It!—Finding M (III)

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

Eve Can Crack It!—Finding M (III)

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

1. M divides 36392598.

Eve Can Crack It!—Finding M (III)

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

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

Eve Can Crack It!—Finding 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.

Eve Can Crack It!—Finding 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 $7648 \leq M \leq 9999$.
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. If use 311 then need a 3: $2 \times 11 \times 311 = 6842 < 7648$.

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 $7648 \le M \le 9999$.

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: $\ge 311 \times 3 \times 11 = 10263 > 9999$.

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. 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:
   $311 \times 11 \times 9 = 30789 > 9999$.

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. 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: $311 \times 2 \times 9 = 5598 < 7648$.

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. 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!

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. 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.

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. 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 $M =$ **8397**.

That Last Slide was Old-Timey

That last slide was the sort of thing people did before computers.

That Last Slide was Old-Timey

That last slide was the sort of thing people did before computers.

Today we would just look at all the factors and see which one works.

That Last Slide was Old-Timey

That last slide was the sort of thing people did before computers.

Today we would just look at all the factors and see which one works.

In fact, today we would do something even less clever—we discuss later.

Reflect

We found $M = 8397$ is only M that works..

Reflect

We found $M = 8397$ is only M that works..

We might have found **no** M works. In that case, goto next 8-sequence.

Reflect

We found $M = 8397$ is only M that works..

We might have found **no** M works. In that case, goto next 8-sequence.

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

Eve Can Crack It—Finding A

EQ4: $-6823 \equiv 5783A \pmod{M}$
By either brute force of cleverness we found that $M = \mathbf{8397}$.

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

Eve Can Crack It—Finding A

EQ4: $-6823 \equiv 5783A \pmod{M}$
By either brute force of cleverness we found that $M = \mathbf{8397}$.

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

Eve Can Crack It—Finding A

EQ4: $-6823 \equiv 5783A \pmod{M}$
By either brute force of cleverness we found that $M = \mathbf{8397}$.

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 move on to next 8-sequence. In the case at hand, the inverse exists.

Eve Can Crack It—Finding A

EQ4: $-6823 \equiv 5783A \pmod{M}$
By either brute force of cleverness we found that $M = \mathbf{8397}$.

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 move on to next 8-sequence. 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 Can Crack It!—Finding B

Now want to find B. Recall:

Eve Can Crack It!—Finding B

Now want to find B. Recall:

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

Eve Can Crack It!—Finding B

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 Can Crack It!—Finding B

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 Can Crack It!—Finding B

Now want to find B. Recall:

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

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

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

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

So..., are we done? Do we have correct A, B, M? Do we need more?

Eve Can Crack It!—Finding x_0

We have $A = 4381$, $B = 7634$, $M = 8307$ so we have

Eve Can Crack It!—Finding x_0

We have $A = 4381, B = 7634, M = 8307$ so we have

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

Eve Can Crack It!—Finding x_0

We have $A = 4381, B = 7634, M = 8307$ so we have

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

Need x_0.

Eve Can Crack It!—Finding x_0

We have $A = 4381, B = 7634, M = 8307$ so we have

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

Need x_0.

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

Eve Can Crack It!—Finding x_0

We have $A = 4381, B = 7634, M = 8307$ so we have

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

Need x_0.

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

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

Eve Can Crack It!—Finding x_0

We have $A = 4381, B = 7634, M = 8307$ so we have

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

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 Crack It!—Finding x_0

We have $A = 4381$, $B = 7634$, $M = 8307$ so we have

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

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 Can Crack It!—Finding x_0 (cont)

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

Eve Can Crack It!—Finding 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.

Eve Can Crack It!—Finding 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 Can Crack It!—Finding 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 YES, then 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 YES, then done.

If NO, then go to next 8-seq or next M if there was one.

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 knows appears in the text and is ≥ 8 letters. $w = w_1 \cdots w_8$ is first 8 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 knows appears in the text and is ≥ 8 letters. $w = w_1 \cdots w_8$ is first 8 letters.
2. For EVERY 8-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 knows appears in the text and is ≥ 8 letters. $w = w_1 \cdots w_8$ is first 8 letters.
2. For EVERY 8-letter seq Eve does the following:
   2.1 Assuming 8-letter seq is $w_1 \cdots w_8$ form equations and try to solve them. If can't then goto next 8-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 knows appears in the text and is ≥ 8 letters. $w = w_1 \cdots w_8$ is first 8 letters.
2. For EVERY 8-letter seq Eve does the following:
   2.1 Assuming 8-letter seq is $w_1 \cdots w_8$ form equations and try to solve them. If can't then goto next 8-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 8-let-seq.

Eve Can Factor Fast?

Eve had to factor:

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

Eve Can Factor Fast?

Eve had to factor:

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

We usually say

Factoring is Hard

Eve Can Factor 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** ?

Eve Can Factor 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.

Eve Can Factor 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.

Eve Can Factor 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** .

With Modern Computers do not Need to be Clever

Recall

(1) M div 36392598, (2) M 4 digs long, (3) $7649 \le M \le 9999$.

How to find M?

With Modern Computers do not Need to be Clever

Recall

(1) M div 36392598, (2) M 4 digs long, (3) $7649 \le M \le 9999$.

How to find M?

Eve Tries All $\mathbf{7649 \le M \le 9999}$

With Modern Computers do not Need to be Clever

Recall
(1) M div 36392598, (2) M 4 digs long, (3) $7649 \le M \le 9999$.
How to find M?
Eve Tries All $\mathbf{7649 \le M \le 9999}$

This gives a small set of possibilities for M.

With Modern Computers do not Need to be Clever

Recall

(1) M div 36392598, (2) M 4 digs long, (3) $7649 \le M \le 9999$.
How to find M?
Eve Tries All $\mathbf{7649 \le M \le 9999}$

This gives a small set of possibilities for M.

PROS and **CONS**

With Modern Computers do not Need to be Clever

Recall

(1) M div 36392598, (2) M 4 digs long, (3) $7649 \le M \le 9999$.

How to find M?

Eve Tries All $\mathbf{7649 \le M \le 9999}$

This gives a small set of possibilities for M.

PROS and **CONS**

1. **PRO** Easy to code.

With Modern Computers do not Need to be Clever

Recall
(1) M div 36392598, (2) M 4 digs long, (3) $7649 \le M \le 9999$.
How to find M?
Eve Tries All $\mathbf{7649 \le M \le 9999}$

This gives a small set of possibilities for M.

PROS and **CONS**

1. **PRO** Easy to code.
2. **CON** Might take a long time if M is more digits long.

With Modern Computers do not Need to be Clever

Recall
(1) M div 36392598, (2) M 4 digs long, (3) $7649 \le M \le 9999$.
How to find M?
Eve Tries All $\mathbf{7649 \le M \le 9999}$

This gives a small set of possibilities for M.

PROS and **CONS**

1. **PRO** Easy to code.
2. **CON** Might take a long time if M is more digits long.
3. CAVEAT: For this example it's fine.

With Modern Computers do not Need to be Clever

Recall
(1) M div 36392598, (2) M 4 digs long, (3) $7649 \le M \le 9999$.
How to find M?
Eve Tries All $\mathbf{7649 \le M \le 9999}$

This gives a small set of possibilities for M.

PROS and **CONS**

1. **PRO** Easy to code.
2. **CON** Might take a long time if M is more digits long.
3. CAVEAT: For this example it's fine.
4. CAVEAT: For the Class Prog Assignment it will be fine.

Real World Versus What I Teach (I)

Paraphrase of a **Recent conversation with Zan**

Real World Versus What I Teach (I)

Paraphrase of a **Recent conversation with Zan**

Bill Have you proofread my slides on the Linear Cong Gen?

Real World Versus What I Teach (I)

Paraphrase of a **Recent conversation with Zan**

Bill Have you proofread my slides on the Linear Cong Gen?
Zan Yes, and they are stupid.

Real World Versus What I Teach (I)

Paraphrase of a **Recent conversation with Zan**

Bill Have you proofread my slides on the Linear Cong Gen?
Zan Yes, and they are stupid.

Bill Is there a mistake in them I should fix?

Real World Versus What I Teach (I)

Paraphrase of a **Recent conversation with Zan**

Bill Have you proofread my slides on the Linear Cong Gen?
Zan Yes, and they are stupid.

Bill Is there a mistake in them I should fix?

Zan You say that Java and other langs use an LCG with some **mysterious M** as the mod.

Real World Versus What I Teach (I)

Paraphrase of a **Recent conversation with Zan**

Bill Have you proofread my slides on the Linear Cong Gen?
Zan Yes, and they are stupid.

Bill Is there a mistake in them I should fix?

Zan You say that Java and other langs use an LCG with some **mysterious M** as the mod. The mod is always 2^{32} or 2^{64} you moron.

Real World Versus What I Teach (I)

Paraphrase of a **Recent conversation with Zan**

Bill Have you proofread my slides on the Linear Cong Gen?
Zan Yes, and they are stupid.

Bill Is there a mistake in them I should fix?

Zan You say that Java and other langs use an LCG with some **mysterious M** as the mod. The mod is always 2^{32} or 2^{64} you moron.

Bill But if Alice and Bob use a power of 2 that will cut down on Eve's search space!

Real World Versus What I Teach (I)

Paraphrase of a **Recent conversation with Zan**

Bill Have you proofread my slides on the Linear Cong Gen?
Zan Yes, and they are stupid.

Bill Is there a mistake in them I should fix?

Zan You say that Java and other langs use an LCG with some **mysterious M** as the mod. The mod is always 2^{32} or 2^{64} you moron.

Bill But if Alice and Bob use a power of 2 that will cut down on Eve's search space!
This exciting conversation continued on next slide!

Real World versus What I Teach

Paraphrase of a **Recent conversation with Zan (cont)**

Zan Get real man!

Real World versus What I Teach

Paraphrase of a **Recent conversation with Zan (cont)**

Zan Get real man!

Bill I will teach them how to crack LCG in the general case, but then comment that often M is a power of 2.

Real World versus What I Teach

Paraphrase of a **Recent conversation with Zan (cont)**

Zan Get real man!

Bill I will teach them how to crack LCG in the general case, but then comment that often M is a power of 2.

Zan Okay, that works. You are truly the master of education (NOTE: Zan did not say that, but he did call me a moron again.)

Real World Versus What I Teach (II)

Paraphrase of a **Recent conversation with a Student**

Real World Versus What I Teach (II)

Paraphrase of a **Recent conversation with a Student**

Bill All langs use Linear Cong Gens for Rand Numbs.

Real World Versus What I Teach (II)

Paraphrase of a **Recent conversation with a Student**

Bill All langs use Linear Cong Gens for Rand Numbs.

Student Actually Python uses the Mersenne Twister.

Real World Versus What I Teach (II)

Paraphrase of a **Recent conversation with a Student**

Bill All langs use Linear Cong Gens for Rand Numbs.

Student Actually Python uses the Mersenne Twister.

Bill OH. I wonder if that would be good for crypto.

Real World Versus What I Teach (II)

Paraphrase of a **Recent conversation with a Student**

Bill All langs use Linear Cong Gens for Rand Numbs.

Student Actually Python uses the Mersenne Twister.

Bill OH. I wonder if that would be good for crypto.

Student They say to NOT use it for crypto.

Real World Versus What I Teach (II)

Paraphrase of a **Recent conversation with a Student**

Bill All langs use Linear Cong Gens for Rand Numbs.

Student Actually Python uses the Mersenne Twister.

Bill OH. I wonder if that would be good for crypto.

Student They say to NOT use it for crypto.

Bill OH. Well, I will look into it and present it to next years class.

Real World Versus What I Teach (II)

Paraphrase of a **Recent conversation with a Student**

Bill All langs use Linear Cong Gens for Rand Numbs.

Student Actually Python uses the Mersenne Twister.

Bill OH. I wonder if that would be good for crypto.

Student They say to NOT use it for crypto.

Bill OH. Well, I will look into it and present it to next years class.

Student Why not this semester?

Real World Versus What I Teach (II)

Paraphrase of a **Recent conversation with a Student**

Bill All langs use Linear Cong Gens for Rand Numbs.

Student Actually Python uses the Mersenne Twister.

Bill OH. I wonder if that would be good for crypto.

Student They say to NOT use it for crypto.

Bill OH. Well, I will look into it and present it to next years class.

Student Why not this semester?

Bill Why not indeed! Okay! I accept your challenge!

Real World Versus What I Teach (II)

Paraphrase of a **Recent conversation with a Student**

Bill All langs use Linear Cong Gens for Rand Numbs.

Student Actually Python uses the Mersenne Twister.

Bill OH. I wonder if that would be good for crypto.

Student They say to NOT use it for crypto.

Bill OH. Well, I will look into it and present it to next years class.

Student Why not this semester?

Bill Why not indeed! Okay! I accept your challenge!

Student Challenge? What challenge?

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 $\{0, \ldots, 26\}$).

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 $\{0, \ldots, 26\}$).

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 $\{0, \ldots, 26\}$).

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 $\{0, \ldots, 26\}$).

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.

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 $\{0, \ldots, 26\}$).

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.

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 $\{0, \ldots, 26\}$).

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 $\{0, \ldots, 26\}$).

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
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.

An Approach To Generating Random Bits

Random-number generation

1. Continually collect ‘unpredictable” data.
2. This data may be biased.
3. Correct biases in it to make it more random.
4. Called **smoothing** .

Unpredictable: Different models. Our Model: There is a $0 < p < 1$ such that each bit has

$$\Pr(1) = p, \Pr(0) = 1 - p.$$

Bits are independent. p is not known.

Smoothing via Von Neumann Technique (VN)

- Need to eliminate *bias*.
- VN technique for eliminating bias:
  - Collect two bits per output bit
    - $01 \mapsto 0$
    - $10 \mapsto 1$
    - $00, 11 \mapsto \text{skip}$
  - Note that this assumes *independence* (as well as constant bias)
  - This gives truly random bits (next slide) but takes time.

Prob of 0, Prob of 1

$$\Pr(1) = p,\ \Pr(0) = 1 - p.$$

Prob of 0, Prob of 1

$$\Pr(1) = p, \Pr(0) = 1 - p.$$

Flip 2 coins

first bit	second bit	Prob
0	0	$(1-p)^2$
0	1	$(1-p)p$
1	0	$p(1-p)$
1	1	p^2

Prob of 0, Prob of 1

$$\Pr(1) = p, \Pr(0) = 1 - p.$$

Flip 2 coins

first bit	second bit	Prob
0	0	$(1-p)^2$
0	1	$(1-p)p$
1	0	$p(1-p)$
1	1	p^2

$$\Pr(01) = \Pr(10) = p(1-p).$$

Prob of 0, Prob of 1

$$\Pr(1) = p, \Pr(0) = 1 - p.$$

Flip 2 coins

first bit	second bit	Prob
0	0	$(1-p)^2$
0	1	$(1-p)p$
1	0	$p(1-p)$
1	1	p^2

$$\Pr(01) = \Pr(10) = p(1-p).$$

Hence if we toss out the 00 and 11 then

Prob of 0, Prob of 1

$$\Pr(1) = p, \ \Pr(0) = 1 - p.$$

Flip 2 coins

first bit	second bit	Prob
0	0	$(1-p)^2$
0	1	$(1-p)p$
1	0	$p(1-p)$
1	1	p^2

$$\Pr(01) = \Pr(10) = p(1-p).$$

Hence if we toss out the 00 and 11 then

$$\Pr(01) = \Pr(10) = \frac{1}{2}.$$

Prob of 0, Prob of 1

$$\Pr(1) = p, \Pr(0) = 1 - p.$$

Flip 2 coins

first bit	second bit	Prob
0	0	$(1-p)^2$
0	1	$(1-p)p$
1	0	$p(1-p)$
1	1	p^2

$$\Pr(01) = \Pr(10) = p(1-p).$$

Hence if we toss out the 00 and 11 then

$$\Pr(01) = \Pr(10) = \frac{1}{2}.$$

Perfect Randomness!

How Many Random Bits Can We Expect?

Assume that $\Pr(b = 0) = p$ and $\Pr(b = 1) = 1 - p$.

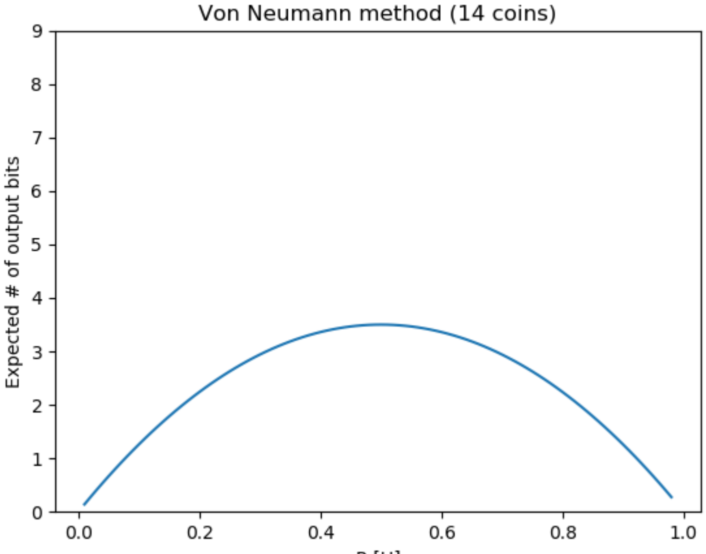
If flip 2 coins then expected numb of rand bits is

$$\Pr(01) + \Pr(10) = p(1 - p) + (1 - p)p = 2p(1 - p).$$

If flip $2n$ coins then expected number of rand bits is $2np(1 - p)$.

How Good is VN Method?

If flip 14 coins ($n = 7$) then we get the following graph:



How Good is VN Method? Not Very Good

How Good is VN Method? Not Very Good

1. If $p = 0.2$ or 0.8 then from 14 flips we only get around 2 truly random bits.

How Good is VN Method? Not Very Good

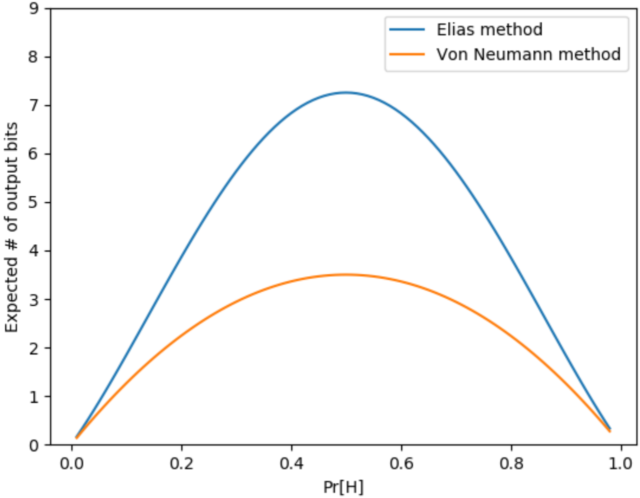
1. If $p = 0.2$ or 0.8 then from 14 flips we only get around 2 truly random bits. Sad.

How Good is VN Method? Not Very Good

1. If $p = 0.2$ or 0.8 then from 14 flips we only get around 2 truly random bits. Sad.
2. The method can be extended, called **The Elias Method.** We won't present it but will show graph on next slide.

VN vs GMS

If we flip 14 bits:



How Good is Elias Method?

How Good is Elias Method?

1. If $p = 0.2$ or 0.8 then from 14 flips we only get around 4 truly random bits. Better than VN.

How Good is Elias Method?

1. If $p = 0.2$ or 0.8 then from 14 flips we only get around 4 truly random bits. Better than VN. Still sad.

How Good is Elias Method?

1. If $p = 0.2$ or 0.8 then from 14 flips we only get around 4 truly random bits. Better than VN. Still sad.
2. For both VN and Elias we are assuming that there is a steady source of independent biased coins with the same bias. This is unrealistic. Still, a good attempt.

How Good is Elias Method?

1. If $p = 0.2$ or 0.8 then from 14 flips we only get around 4 truly random bits. Better than VN. Still sad.
2. For both VN and Elias we are assuming that there is a steady source of independent biased coins with the same bias. This is unrealistic. Still, a good attempt.
3. So can we get truly random bits?

Sources of True Random Bits

Sources of True Random Bits

1. Radioactivity

Sources of True Random Bits

1. Radioactivity
2. Atmospheric noise

Sources of True Random Bits

1. Radioactivity
2. Atmospheric noise
3. Last bit of the atomic clock

Sources of True Random Bits

1. Radioactivity
2. Atmospheric noise
3. Last bit of the atomic clock
4. Thermal Heat-entropy.

Sources of True Random Bits

1. Radioactivity
2. Atmospheric noise
3. Last bit of the atomic clock
4. Thermal Heat-entropy.
5. Lasers

These are all expensive.

Sources of True Random Bits

1. Radioactivity
2. Atmospheric noise
3. Last bit of the atomic clock
4. Thermal Heat-entropy.
5. Lasers

These are all expensive.

What is used Psuedo-random generator that are more sophisticated than what I showed here.

Sources of True Random Bits

1. Radioactivity
2. Atmospheric noise
3. Last bit of the atomic clock
4. Thermal Heat-entropy.
5. Lasers

These are all expensive.

What is used Psuedo-random generator that are more sophisticated than what I showed here.