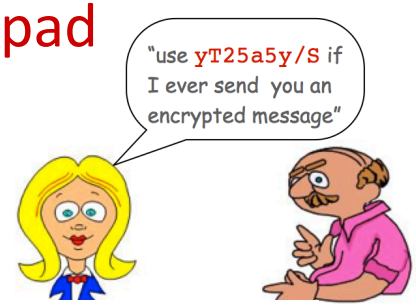
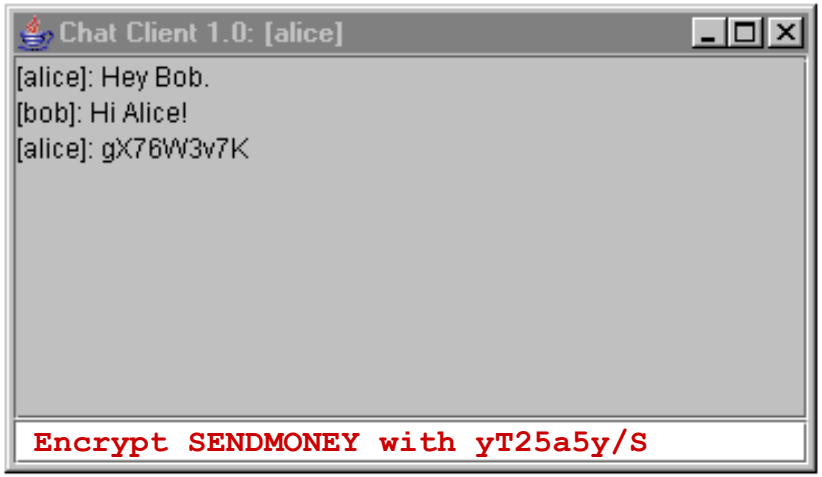
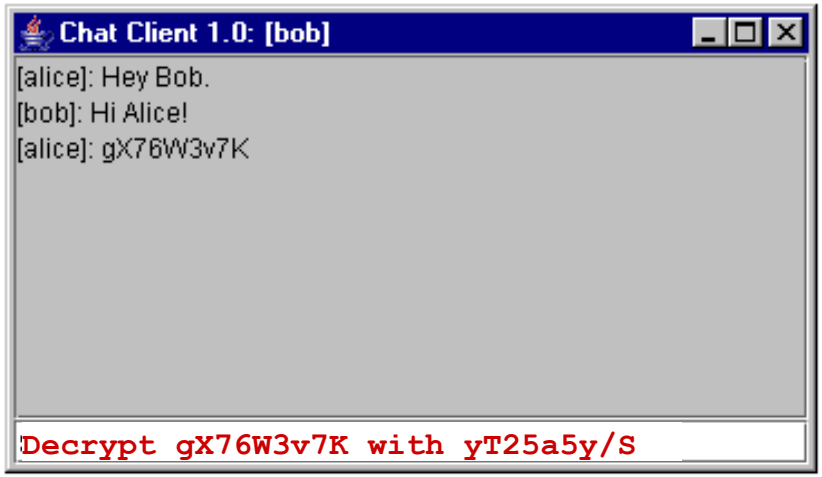
Secure Chat

Alice wants to send a secret message to Bob?

- Sometime in the past, they exchange a one-time pad
- Alice uses the pad to encrypt the message
- Bob uses the same pad to decrypt the message





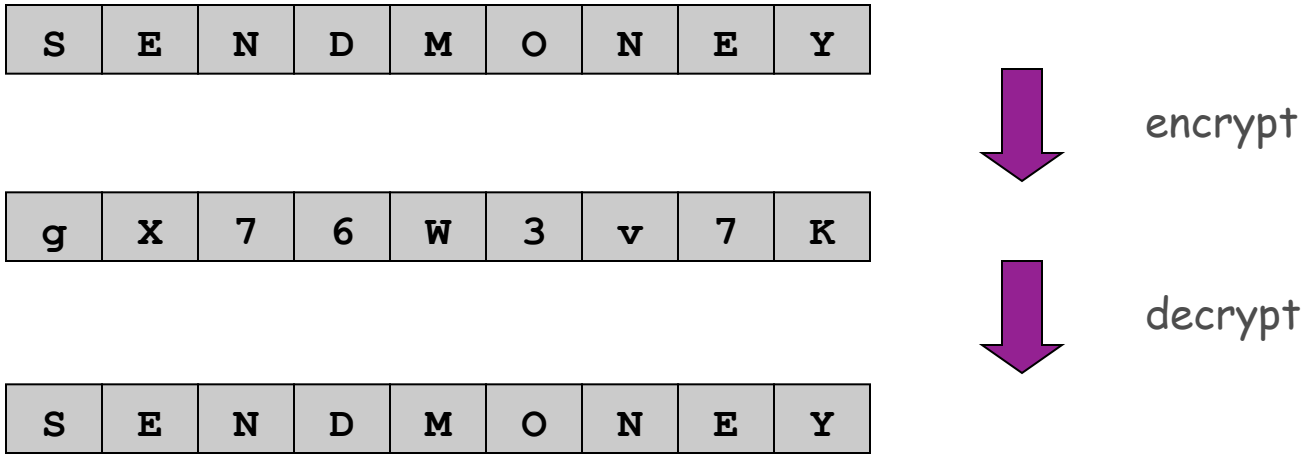


Key point Without the pad, Eve cannot understand the message

Penn Engineering

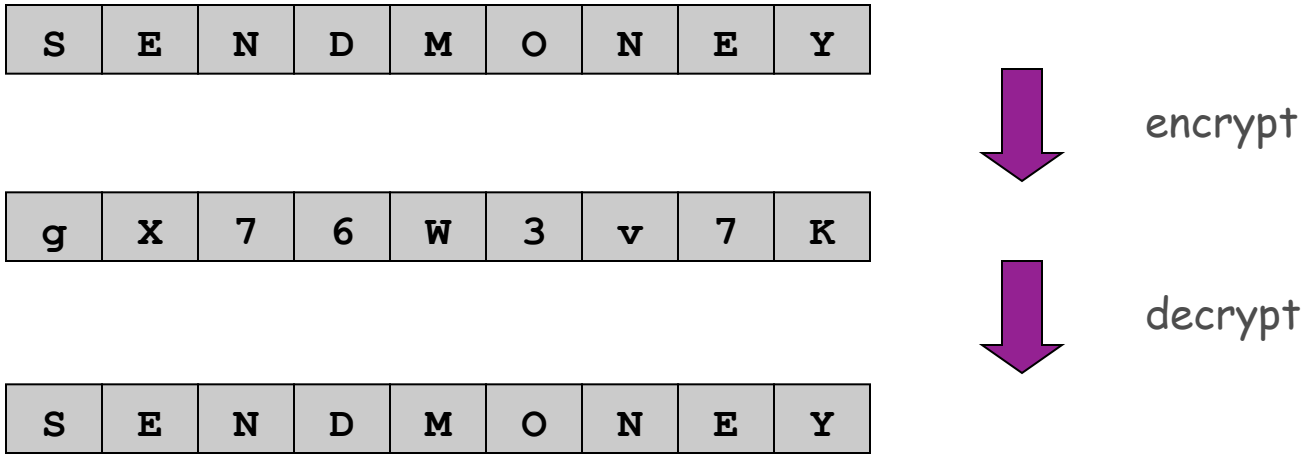
Encryption Machine

Goal: Design a machine to encrypt and decrypt data



Encryption Machine

Goal: Design a machine to encrypt and decrypt data



Enigma encryption machine

- "Unbreakable" German code during WWII
- Broken by Turing bombe
- One of first uses of computers
- Helped win Battle of Atlantic by locating U-boats

Penn Engineering

A Digital World

Data is a sequence of bits [bit = 0 or 1]

- Text
- Programs, executables
- Documents, pictures, sounds, movies, …

Base64 encoding Use 6 bits to represent each alphanumeric symbol.

Binary	Char	Binary	Char	Binary	Char	Binary	Char	Binary	Char	Binary	Char
000000	A	001011	L	010110	W	100001	h	101100	s	110111	3
000001	B	001100	M	010111	X	100010	i	101101	t	111000	4
000010	C	001101	N	011000	Y	100011	j	101110	u	111001	5
000011	D	001110	O	011001	Z	100100	k	101111	v	111010	6
000100	E	001111	P	011010	a	100101	l	110000	w	111011	7
000101	F	010000	Q	011011	b	100110	m	110001	x	111100	8
000110	G	010001	R	011100	c	100111	n	110010	y	111101	9
000111	H	010010	S	011101	d	101000	o	110011	z	111110	+
001000	I	010011	T	011110	e	101001	p	110100	0	111111	/
001001	J	010100	U	011111	f	101010	q	110101	1		
001010	K	010101	V	100000	g	101011	r	110110	2		

Penn Engineering

One-Time Pad Encryption

Encryption

- Convert text message to N bits

Base64 Encoding

char	dec	binary
A	0	000000
B	1	000001
…	…	…
M	12	001100
…	…	…

S	E	N	D	M	O	N	E	Y	message
010010	000100	001101	000011	001100	001110	001101	000100	011000	base64

Penn Engineering

One-Time Pad Encryption

Encryption

- Convert text message to N bits
- Generate N random bits (one-time pad)

S	E	N	D	M	O	N	E	Y	message
010010	000100	001101	000011	001100	001110	001101	000100	011000	base64
110010	010011	110110	111001	011010	111001	100010	111111	010010	random bits

Penn Engineering

One-Time Pad Encryption

XOR Truth Table

x	y	x ^ y
0	0	0
0	1	1
1	0	1
1	1	0

Encryption

- Convert text message to N bits
- Generate N random bits (one-time pad)
- Take bitwise XOR of two bitstrings

sum corresponding pair of bits: 1 if sum is odd, 0 if even

S	E	N	D	M	O	N	E	Y	
010010	000100	001101	000011	001100	001110	001101	000100	011000	base64
110010	010011	110110	111001	011010	111001	100010	111111	010010	random bits
100000	010111	111011	111010	010110	110111	101111	111011	001010	XOR

0 ^ 1 = 1

One-Time Pad Encryption

Encryption

- Convert text message to N bits
- Generate N random bits (one-time pa
- Take bitwise XOR of two bitstrings
- Convert binary back into text

Base64 Encoding

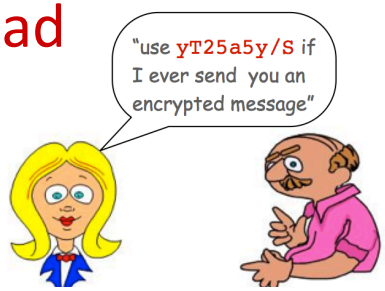
char	dec	binary
A	0	000000
B	1	000001
…	…	…
w	22	010110
…	…	…

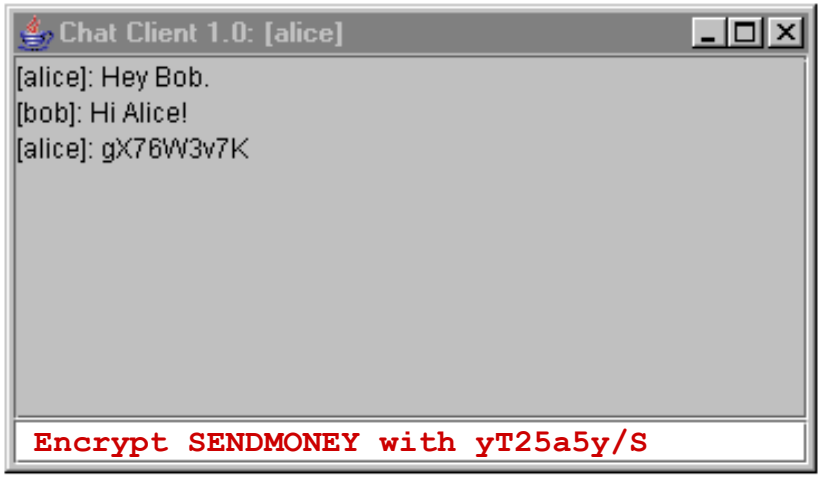
S	**E**	**N**	**D**	**M**	**O**	**N**	**E**	**Y**	message
010010	000100	001101	000011	001100	001110	001101	000100	011000	base64
110010	010011	110110	111001	011010	111001	100010	111111	010010	random bits
100000	010111	111011	111010	010110	110111	101111	111011	001010	XOR
g	**X**	**7**	**6**	**W**	**3**	**v**	**7**	**K**	encrypted

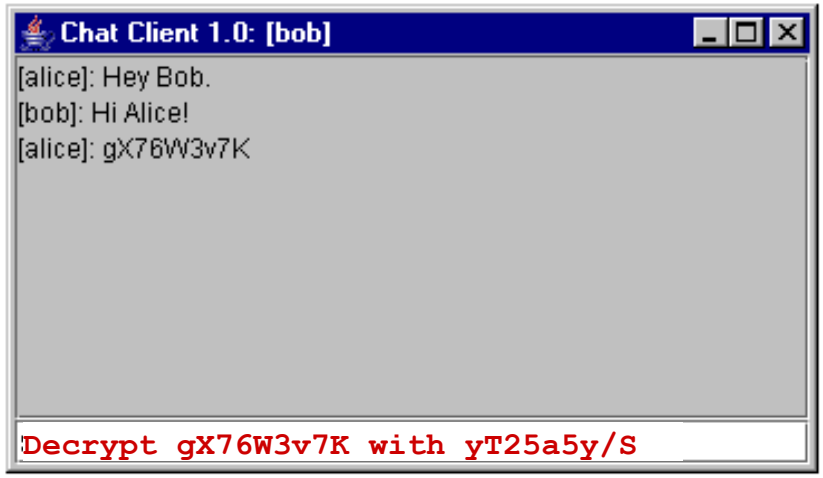
Secure Chat (review)

Alice wants to send a secret message to Bob?

- Sometime in the past, they exchange a one-time pad
- Alice uses the pad to encrypt the message
- Bob uses the same pad to decrypt the message



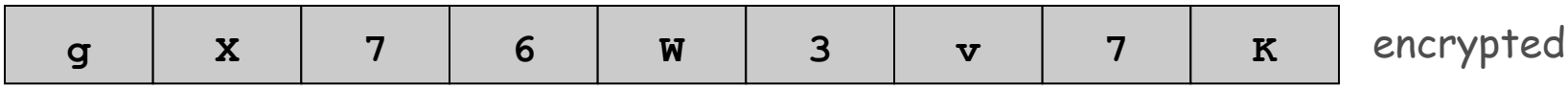




One-Time Pad Decryption

Decryption

- Convert encrypted message to binary



One-Time Pad Decryption

Decryption

- Convert encrypted message to binary

Base64 Encoding

char	dec	binary
A	0	000000
B	1	000001
…	…	…
W	22	010110
…	…	…

g	x	7	6	W	3	v	7	K	encrypted
100000	010111	111011	111010	010110	110111	101111	111011	001010	base64

One-Time Pad Decryption

Decryption

- Convert encrypted message to binary
- Use same N random bits (one-time pad)

g	x	7	6	w	3	v	7	k	encrypted
100000	010111	111011	111010	010110	110111	101111	111011	001010	base64
110010	010011	110110	111001	011010	111001	100010	111111	010010	random bits

One-Time Pad Decryption

Decryption

- Convert encrypted message to binary
- Use same N random bits (one-time pad)
- Take bitwise XOR of two bitstrings

XOR Truth Table

x	y	x ^ y
0	0	0
0	1	1
1	0	1
1	1	0

g	X	7	6	W	3	v	7	K	
100000	010111	111011	111010	010110	110111	101111	111011	001010	base64
110010	010011	110110	111001	011010	111001	100010	111111	010010	random bits
010010	000100	001101	000011	001100	001110	001101	000100	011000	XOR

encrypted

1 ^ 1 = 0

One-Time Pad Decryption

Decryption

- Convert encrypted message to binary
- Use same N random bits (one-time pad)
- Take bitwise XOR of two bitstrings
- Convert back into text

Base64 Encoding

char	dec	binary
A	0	000000
B	1	000001
…	…	…
M	12	001100
…	…	…

g	**X**	**7**	**6**	**W**	**3**	**v**	**7**	**K**	encrypted
100000	010111	111011	111010	010110	110111	101111	111011	001010	base64
110010	010011	110110	111001	011010	111001	100010	111111	010010	random bits
010010	000100	001101	000011	001100	001110	001101	000100	011000	XOR
S	**E**	**N**	**D**	**M**	**O**	**N**	**E**	**Y**	message

Why Does It Work?

Crucial property Decrypted message = original message

Notation	Meaning
a	original message bit
b	one-time pad bit
^	XOR operator
a ^ b	encrypted message bit
(a ^ b) ^ b	decrypted message bit

Why is crucial property true?

- Use properties of XOR.
- (a ^ b) ^ b = a ^ (b ^ b) = a ^ 0 = a

associativity of ^ always 0 identity

XOR Truth Table

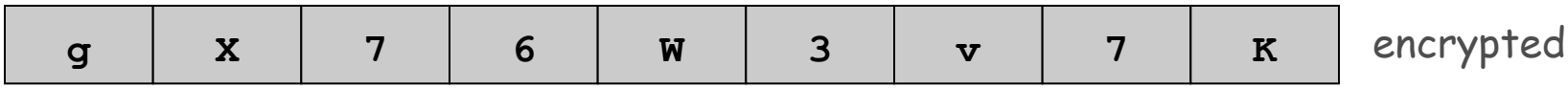
x	y	x ^ y
0	0	0
0	1	1
1	0	1
1	1	0

Penn Engineering

One-Time Pad Decryption (with the wrong pad)

Decryption

- Convert encrypted message to binary



Penn Engineering

One-Time Pad Decryption (with the wrong pad)

Decryption

- Convert encrypted message to binary

g	X	7	6	W	3	v	7	K	encrypted
100000	010111	111011	111010	010110	110111	101111	111011	001010	base64

One-Time Pad Decryption (with the wrong pad)

Decryption

- Convert encrypted message to binary
- Use wrong N bits (bogus one-time pad)

g	x	7	6	w	3	v	7	k	encrypted
100000	010111	111011	111010	010110	110111	101111	111011	001010	base64
101000	011100	110101	101111	010010	111001	100101	101010	001010	wrong bits

One-Time Pad Decryption (with the wrong pad)

Decryption

- Convert encrypted message to binary
- Use wrong N bits (bogus one-time pad)
- Take bitwise XOR of two bitstrings

g	x	7	6	W	3	v	7	K	encrypted
100000	010111	111011	111010	010110	110111	101111	111011	001010	base64
101000	011100	110101	101111	010010	111001	100101	101010	001010	wrong bits
001000	001011	001110	010101	000100	001110	001010	010001	000000	XOR

One-Time Pad Decryption (with the wrong pad)

Decryption

- Convert encrypted message to binary
- Use wrong N bits (bogus one-time pad)
- Take bitwise XOR of two bitstrings
- Convert back into text: Oops

g	**X**	**7**	**6**	**W**	**3**	**v**	**7**	**K**	encrypted
100000	010111	111011	111010	010110	110111	101111	111011	001010	base64
101000	011100	110101	101111	010010	111001	100101	101010	001010	wrong bits
001000	001011	001110	010101	000100	001110	001010	010001	000000	XOR
I	**L**	**O**	**V**	**E**	**O**	**K**	**R**	**A**	wrong message

Goods and Bads of One-Time Pads

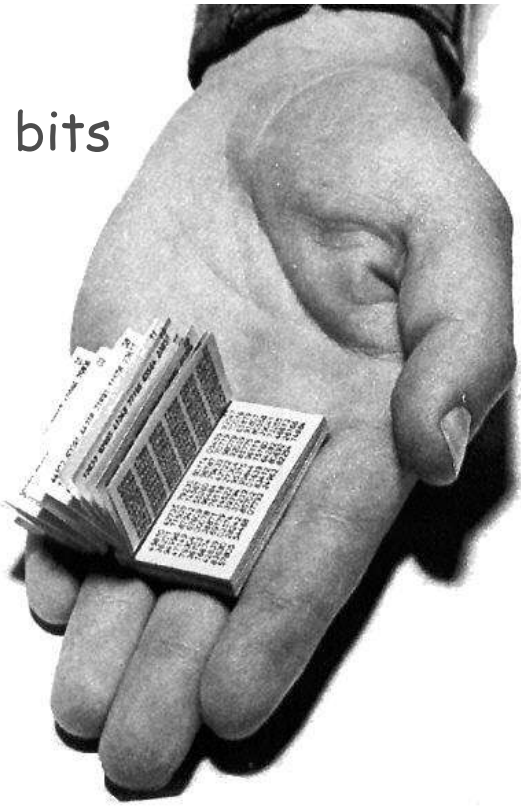
Good

- Easily computed by hand
- Very simple encryption/decryption processes
- Provably unbreakable if bits are truly random
  [Shannon, 1940s]

eavesdropper Eve sees only random bits

Bad

- Easily breakable if pad is re-used
- Pad must be as long as the message
- Truly random bits are hard to generate
- Pad must be distributed securely
  - impractical for Web commerce



a Russian one-time pad

Pseudo-Random Bit Generator

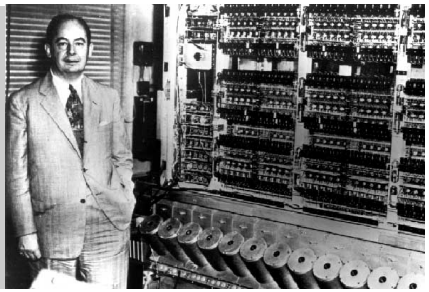
Practical middle-ground

- Let's make a "random"-bit generator gadget
- Alice and Bob each get identical small gadgets

How to make a gadget that produces "random" bits

- Enigma machine
- Linear feedback shift register
- Linear congruential generator
- Blum-Blum-Shub generator

" Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin. "
– Jon von Neumann (left)
– ENIAC (right)

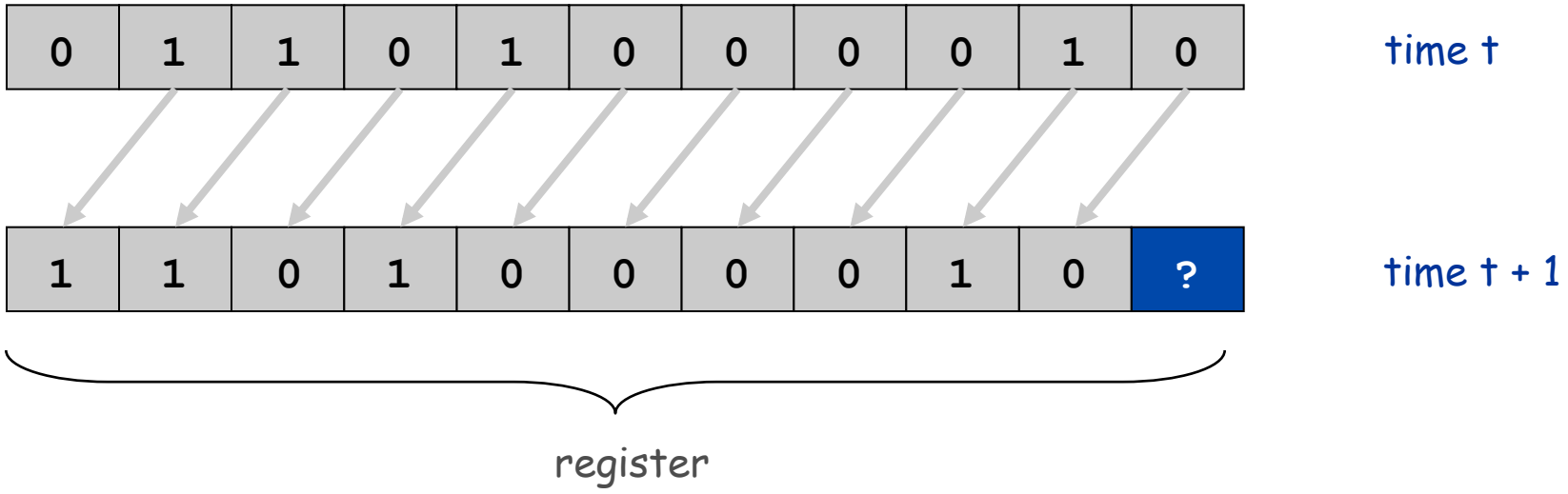


Penn Engineering

Shift Register

Shift register terminology.

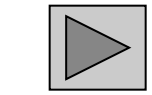
- Bit: 0 or 1
- Cell: storage element that holds one bit
- Register: sequence of cells
- Seed: initial sequence of bits
- Shift register: when clock ticks, bits propagate one position to left



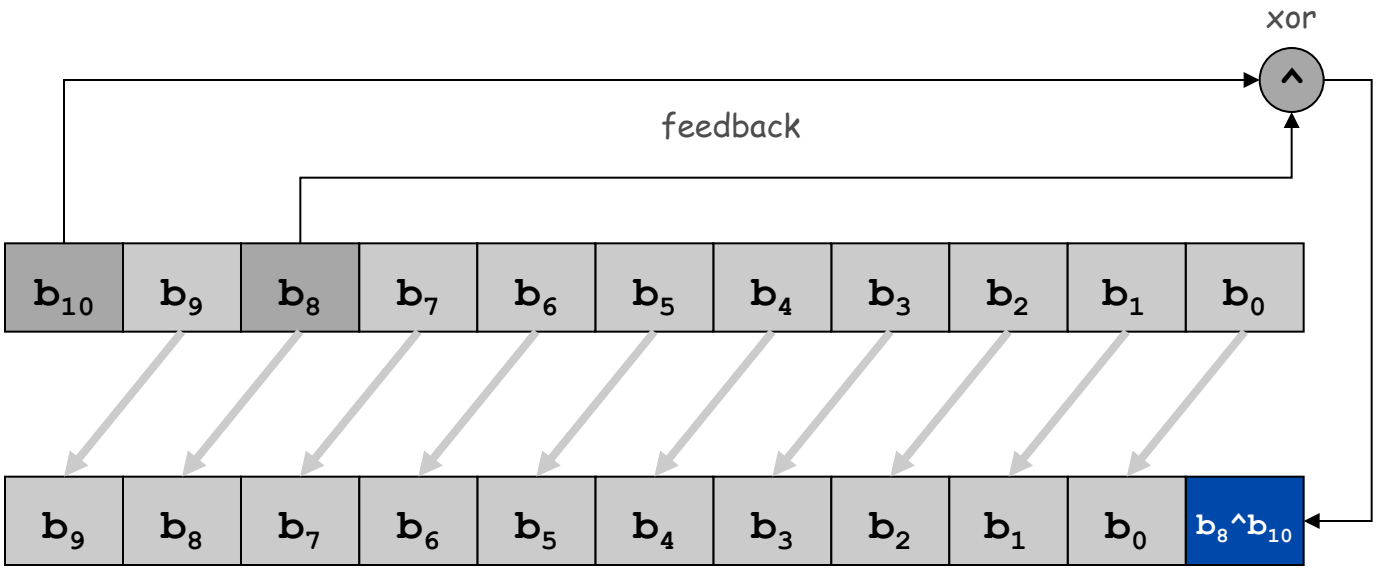
Linear Feedback Shift Register (LFSR)

{8, 10} linear feedback shift register

- Shift register with 11 cells
- Bit b_0 is is XOR of previous bits b_8 and b_{10}
- Pseudo-random bit = b_0

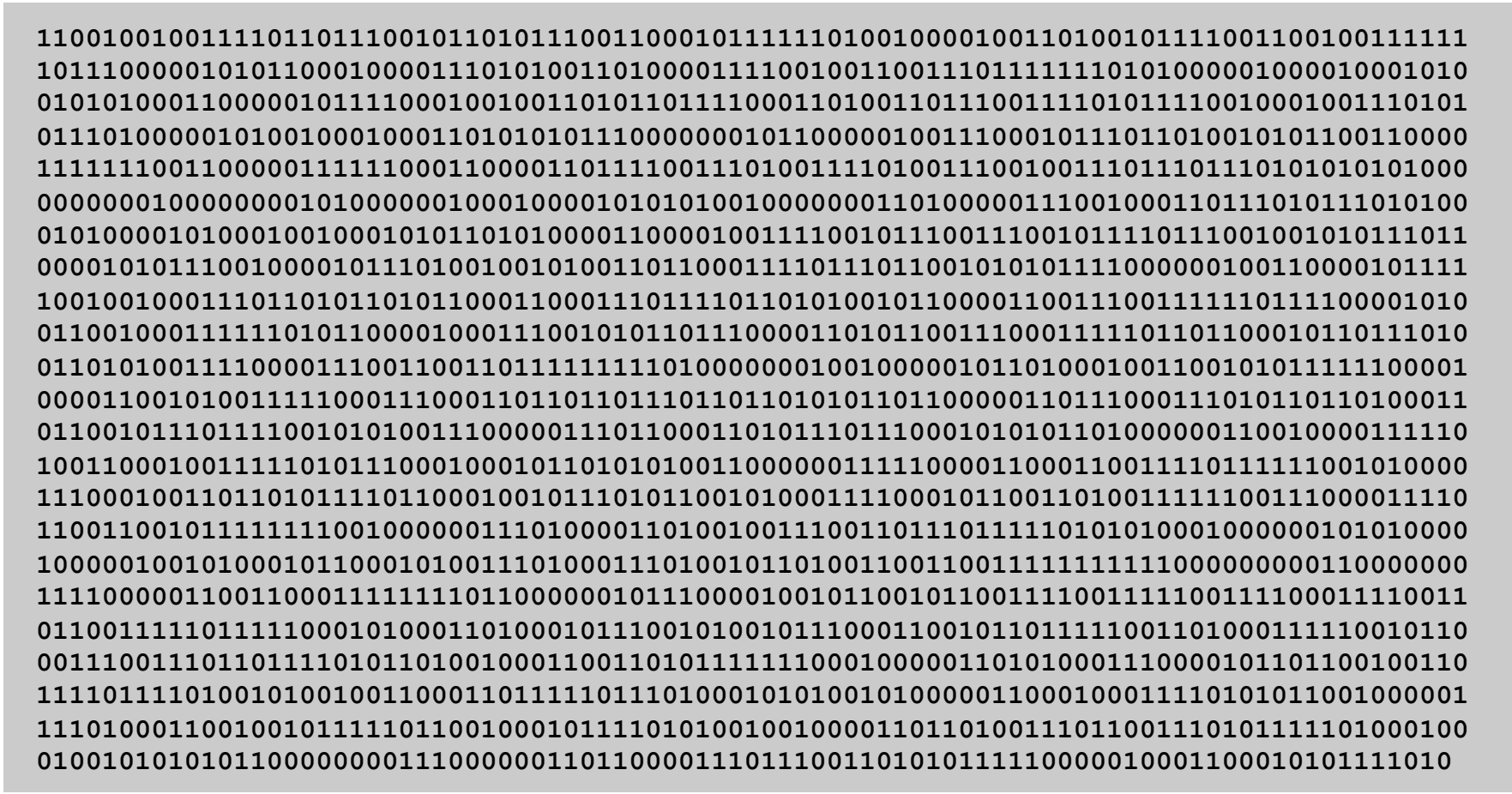


LFSR demo



Random Numbers

Q. Are these 2000 numbers random?



A. No. This is output of {8, 10} LFSR with seed 01101000010!

Penn Engineering

LFSR Encryption

LFSR encryption

- Convert text message to N bits
- Initialize LFSR with small seed
- Generate N bits with LFSR
- Take bitwise XOR of two bitstrings
- Convert binary back into text

Base64 Encoding

char	dec	binary
A	0	000000
B	1	000001
…	…	…
w	22	010110
…	…	…

S	**E**	**N**	**D**	**M**	**O**	**N**	**E**	**Y**	message
010010	000100	001101	000011	001100	001110	001101	000100	011000	base64
110010	010011	110110	111001	011010	111001	100010	111111	010010	LFSR bits
100000	010111	111011	111010	010110	110111	101111	111011	001010	XOR
g	**X**	**7**	**6**	**W**	**3**	**v**	**7**	**K**	encrypted

LFSR Decryption

LFSR Decryption

- Convert encrypted message to N bits
- Initialize identical LFSR with same seed
- Generate N bits with LFSR
- Take bitwise XOR of two bitstrings
- Convert binary back into text

Base64 Encoding

char	dec	binary
A	0	000000
B	1	000001
…	…	…
M	12	001100
…	…	…

g	**X**	**7**	**6**	**W**	**3**	**v**	**7**	**K**	encrypted
100000	010111	111011	111010	010110	110111	101111	111011	001010	base64
110010	010011	110110	111001	011010	111001	100010	111111	010010	LFSR bits
010010	000100	001101	000011	001100	001110	001101	000100	011000	XOR
S	**E**	**N**	**D**	**M**	**O**	**N**	**E**	**Y**	message

Goods and Bads of LFSR Encryption

a commercially available LFSR

Goods

- Easily computed with simple machine
- Very simple encryption/decryption process
- Scalable: 20 cells = 1 million bits; 30 cells = 1 billion bits

  [need theory of finite groups to know where to put taps]

Bads

- Need secure, independent way to distribute LFSR seed
- The bits are not truly random

  [bits in our 11-bit LFSR cycle after $2^{11} - 1 = 2047$ steps]
- Experts have cracked LFSR

  [more complicated machines needed]

Other LFSR Applications

What else can we do with a LFSR?

- DVD encryption with CSS
- DVD decryption with DeCSS!
- Subroutine in military cryptosystems

DVD Jon
(Norwegian hacker)

```
/*      efdtt.c      Author:  Charles M. Hannum <root@ihack.net>              */
/*      Usage is:  cat title-key scrambled.vob | efdtt >clear.vob             */

#define m(i)(x[i]^s[i+84])<<

                    unsigned char x[5]      ,y,s[2048];main(
                    n){for( read(0,x,5     );read(0,s ,n=2048
                             ); write(1   ,s,n)         )if(s
                   [y=s      [13]%8+20]  /16%4  ==1      ){int
                   i=m(      1)17  ^256 +m(0)   8,k      =m(2)
                   0,j=      m(4)   17^ m(3)   9^k*     2-k%8
                   ^8,a     =0,c    =26;for   (s[y]    -=16;
                  --c;j  *=2)a=     a*2^i&    1,i=i /2^j&1
                  <<24;for(j=        127;     ++j<n;c=c>
                                     y)
                                     c

                           +=y=i^i/8^i>>4^i>>12,
                    i=i>>8^y<<17,a^=a>>14,y=a^a*8^a<<6,a=a
                  >>8^y<<9,k=s[j],k         ="7Wo~'G_\216"[k
                   &7]+2^"cr3sfw6v;*k+>/n."[k>>4]*2^k*257/
                         8,s[j]=k^(k&k*2&34)*6^c+~y
                                  ;}}
```

http://www.cs.cmu.edu/~dst/DeCSS/Gallery