CS 112 Introduction to Programming

(Spring 2012)

Lecture #20: LFSR Data Type

Zhong Shao

Department of Computer Science
Yale University
Office: 314 Watson

http://flint.cs.yale.edu/cs112

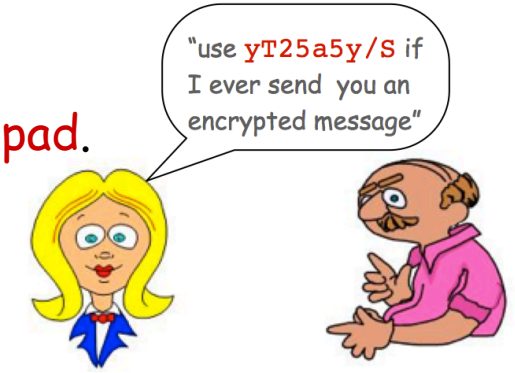
Acknowledgements: some slides used in this class are taken directly or adapted from those accompanying the textbook: **Introduction to Programming in Java: An Interdisciplinary Approach** by Robert Sedgewick and Kevin Wayne (Copyright 2002-2010)

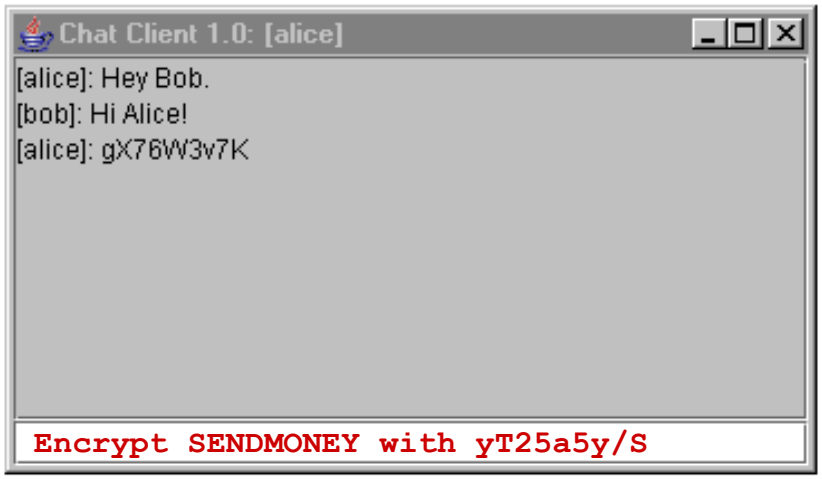


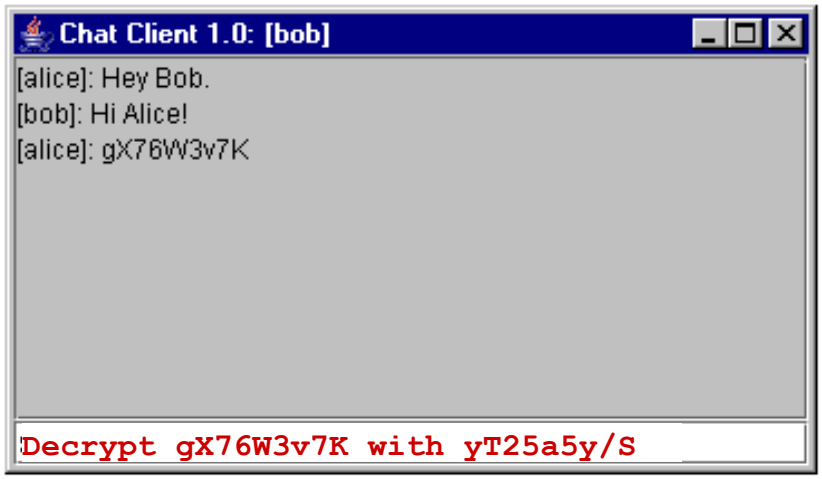
Secure Chat

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.





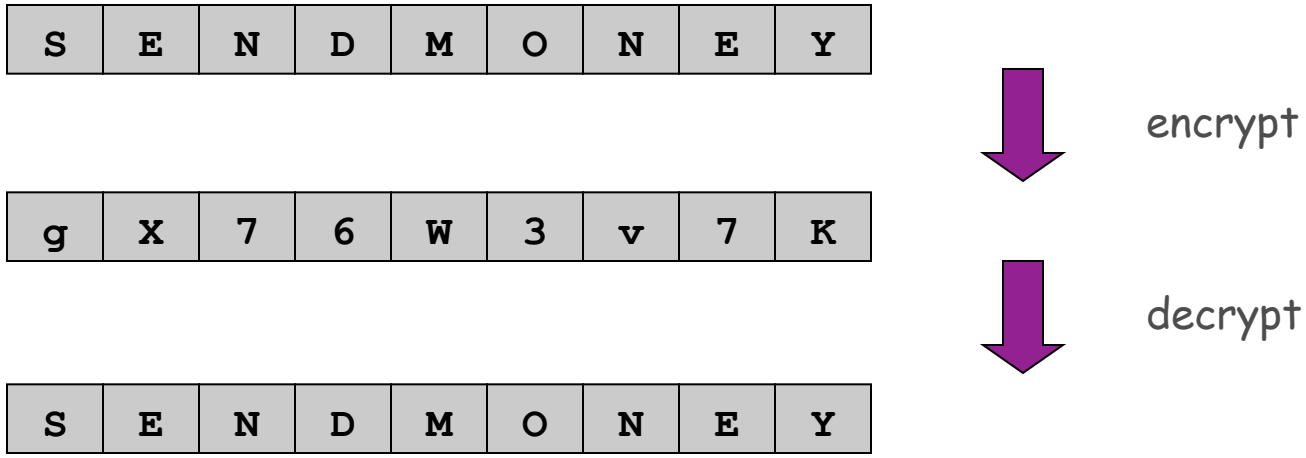


Key point. Without the pad, Eve cannot understand the message.



Encryption Machine

Goal. Design a machine to encrypt and decrypt data.



Encryption Machine

Goal. Design a machine to encrypt and decrypt data.

S	E	N	D	M	O	N	E	Y

encrypt

g	X	7	6	W	3	v	7	K

decrypt

S	E	N	D	M	O	N	E	Y

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.



A Digital World

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

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

File formats. txt, pdf, java, exe, docx, pptx, jpeg, mp3, divx, …



computer with a lens



computer with earbuds



computer with a radio

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.

very weak type of encryption

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		

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

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

One-Time Pad Encryption

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

XOR Truth Table

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

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 pad).
- Take bitwise XOR of two bitstrings.
- Convert binary back into text.

Base64 Encoding

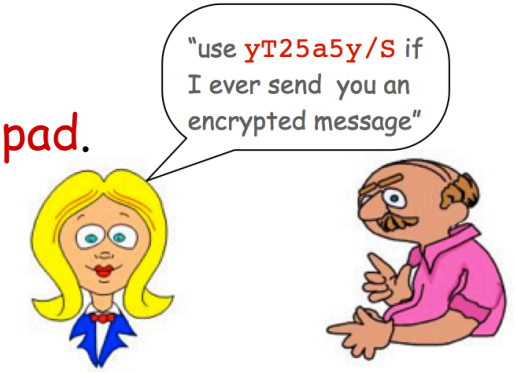
char	dec	binary
A	0	000000
B	1	000001
…	…	…
w	22	010110
…	…	…

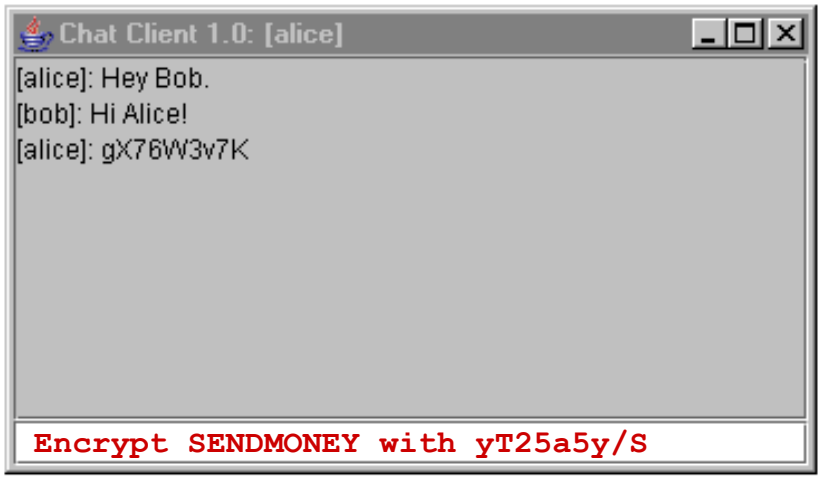
S	**E**	**N**	**D**	**M**	**O**	**N**	**E**	**Y**	message
010010	000100	001101	000011	001100	001110	001101	000100	011000	base64
110010	010011	110110	111001	011010	111001	100010	111111	010010	random bits
100000	010111	111011	111010	010110	110111	101111	111011	001010	XOR
g	**X**	**7**	**6**	**W**	**3**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

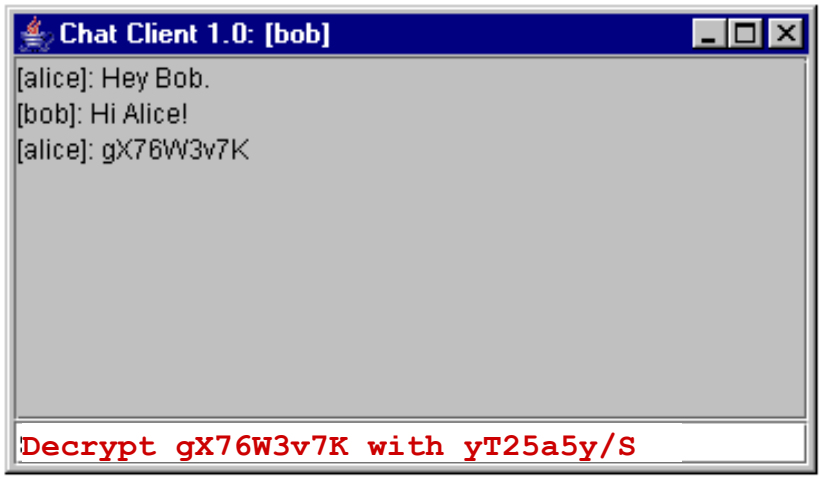
Secure Chat (review)

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.







Key point. Without the pad, Eve cannot understand the message.



One-Time Pad Decryption

Decryption.

- Convert encrypted message to binary.

g	X	7	6	W	3	v	7	K

encrypted

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**	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

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

XOR Truth Table

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

Why is crucial property true?

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

associativity of ^ | always 0 | identity

One-Time Pad Decryption (with the wrong pad)

Decryption.

- Convert encrypted message to binary.

g	X	7	6	W	3	v	7	K

encrypted

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]

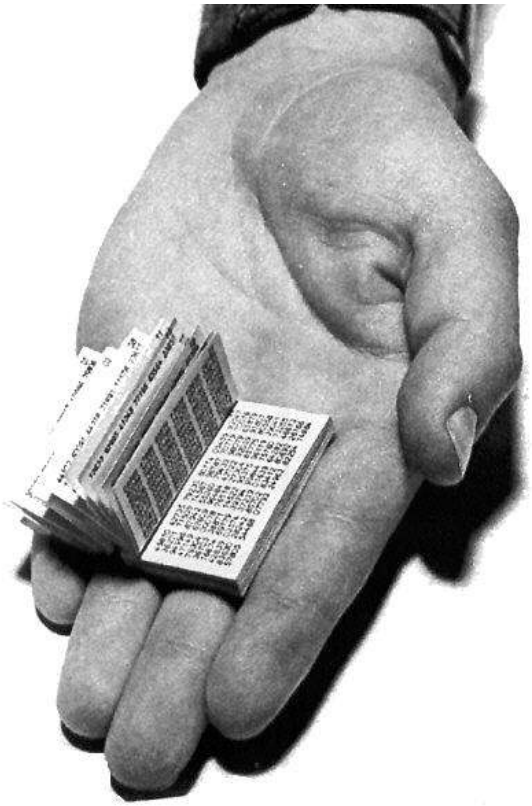
eavesdropper Eve sees only random bits

"one time" means one time only

Bad.

- Easily breakable if pad is re-used.
- Pad must be as long as the message.
- Truly random bits are very hard to come by.
- 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.

instead of identical large one-time pads

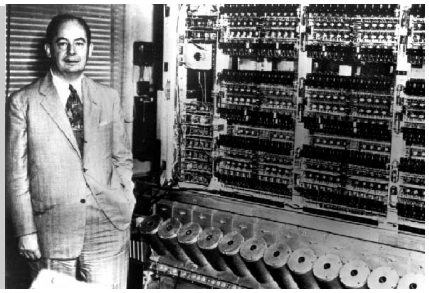
How to make small gadget that produces "random" bits.

- 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)

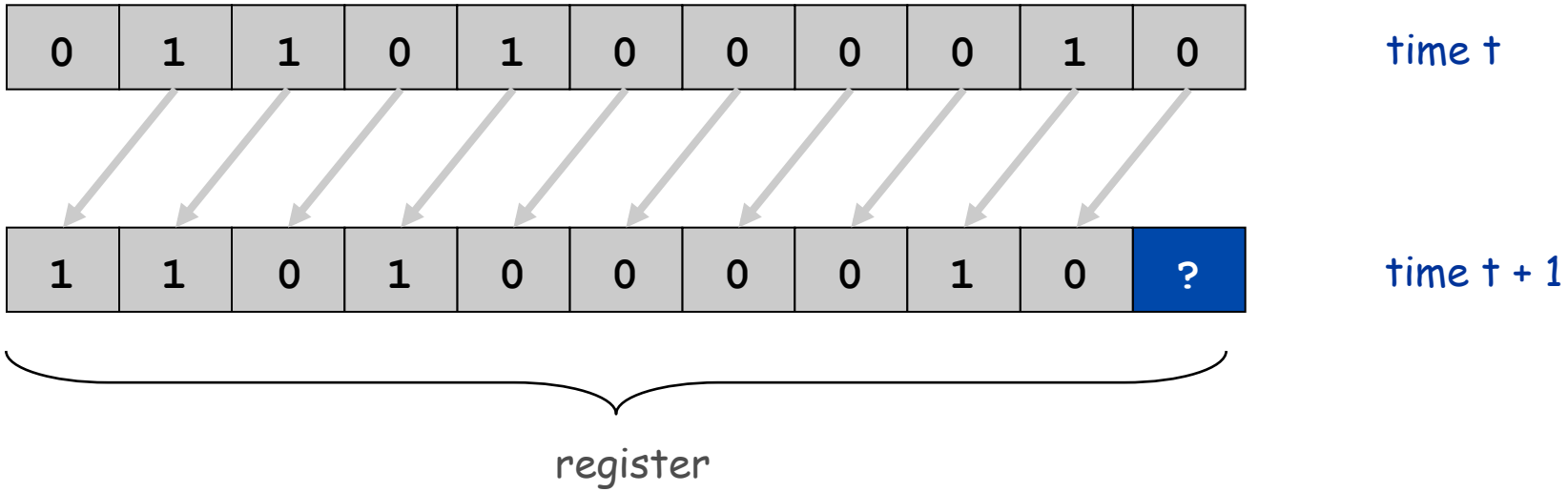
– ENIAC (right)



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.

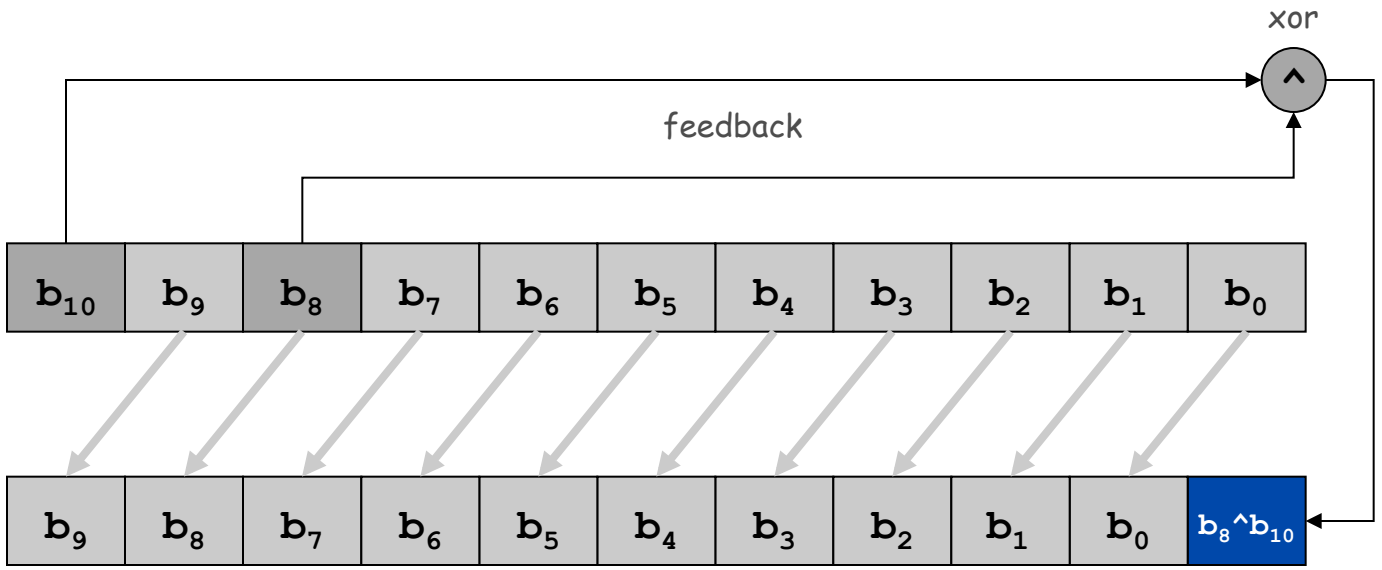


Linear Feedback Shift Register (LFSR)

{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? If not, what is the pattern?

```
11001001001111011011100101101011100110001011111101001000010011010010111100110010011111
11011100000101011000100001110101001101000011110010011001110111111101010000010000100010
10010101000110000010111100010010011010110111100011010011011100111101011110010001001110
10101110100000101001000100011010101011100000001011000001001110001011101101001010110011
00001111111001100000111111000110000110111100111010011110100111001001110111011010101010
01000000000010000000010100000010001000010101010010000000110100000111001000110111010111
01010001010000101000100100010101101010000110000100111100101110011100101111011100100101
01110110000101011100100001011101001001010011011000111101111011001010101111000000100110
00101111100100100011101101011010110001100011101111011010100101100001100111001111110111
10000101001100100011111101011000010001110010101101110000110101100111000111110110110001
01101110100110101001111000011100110011011111111101000000010010000010110100010011001010
11111100001000011001010011111000111000110110110111011011010101101100000110111000111010
11011010001101100101110111100101010011100000111011000110101110111000101010110100000011
00100001111101001100010011111010111000100010110101010011000000111100000110001100111101
11111001010000111000100110110101111011000100101110101100101000111100010110011010011111
10011100001111011001100101111111100100000011101000011010010011100110111011111010101000
10000001010100001000001001010001011000101001110100011101001011010011001100111111111110
00000000110000000111100000110011000111111111011000000101110000100101100101100111100111
10011110001111001101100111111011111000101000110100010111001010010111000110010110111100
11010001111100101100011100111011011110101101001000110011010111111100010000011010100011
10000101101100100110111101111010010100100110001101111101110100010101001010000011000100
01111010101100100000111010001100100101111101100100010111101010010010000110110100111101
10011101011111010001000100101010101100000000011100000011011000011101110011010101111000
00100011000101011111010
```

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

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

Goods.

- Easily computed with simple machine.
- Very simple encryption/decryption process.
- Scalable: 20 cells for 1 million bits; 30 cells for 1 billion bits.
  [but need theory of finite groups to know where to put taps]



a commercially available LFSR

Bads.

- Still 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

LFSR vs. General-Purpose Computer

Important properties.

- Built from simple components.
- Scales to handle huge problems.
- Requires a deep understanding to use effectively.

Basic Component	LFSR	Computer
control	start, stop, load	same
clock	regular pulse	2.8 GHz pulse
memory	11 bits	4 GB
input	seed	sequence of bits
computation	shift, XOR	logic, arithmetic, …
output	pseudo-random bits	Sequence of bits

Critical difference. General-purpose computer can be programmed to simulate any abstract machine.

LFSR Data Type

```
public class LFSR {
    // declare instance variables
    private int N;       // number of bits in the LFSR
    private int[] reg;   // reg[i] = ith bit of LFSR, reg[0] is rightmost bit
    private int tap;     // index of the tap bit

    // constructor to create LFSR with the given initial seed and tap
    public LFSR(String seed, int t) {
        // PUT YOUR CODE HERE
    }

    // simulate one step and return the new bit as 0 or 1
    public int step() {
        // PUT YOUR CODE HERE
    }

    // simulate k steps and return k-bit integer
    public int generate(int k) {
        // PUT YOUR CODE HERE
    }

    // return a string representation of the LFSR
    public String toString()  {
        // PUT YOUR CODE HERE
    }

    // test client
    public static void main(String[] args)  {
       // PUT YOUR TEST CODE HERE
    }

}
```