Hidden flaw jeopardizes millions of online transactions

Mathematicians discover weakness in commonly used encryption schemes

Jump to discuss
comments below
X

Below: Discuss Related

By Paul Wagenseil

updated 2/15/2012 6:01:37 PM ET

Mathematicians based in Switzerland and the United States have discovered a small but
the most commonly used digital encryption schemes, a flaw that could undermine the secu
communication.

COMPUTERWORLD

White Papers Webcasts Newsletters

Topics News In Depth Reviews Blogs Opini

Blogs

IT Blogwatch

A Daily Digest of IT Blogs from Richi Jennings

More posts | Read bio

February 15, 2012 - 6:00 A.M.

RSA crypto: 'flawed', 'risky', 'quagmire of vulnerabilities'

2 Comments

Like 5 +1 1

TAGS: certificate, crypto, cryptography, Diffie-Hellman, encryption, key management, RSA, SSL

IT TOPICS: Applications, Cybercrime & Hacking, Financial IT, Government & Regulation, Internet, Privacy, Security, Security Hardware & Software

A group of six academic researchers have concluded that real-world RSA encryption keys are riskier than Diffie-Hellman-based ones. It seems that some of the random numbers used to generate them weren't, errm, random. In IT Blogwatch, bloggers wonder if the sky is falling.

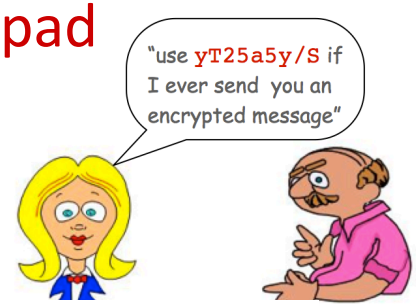
Penn Engineering

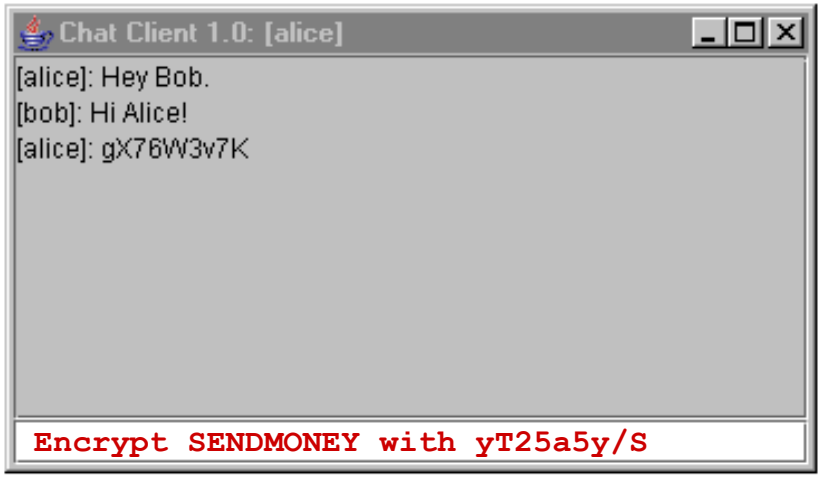
Penn Engineering

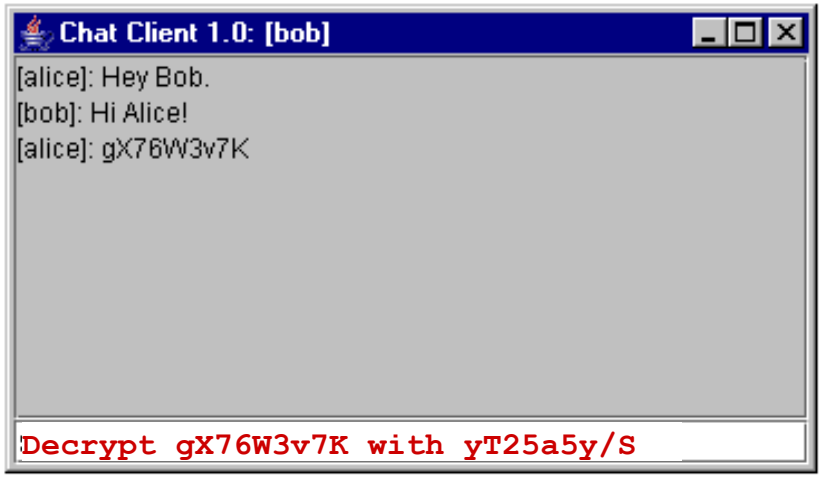
Secure Chat

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





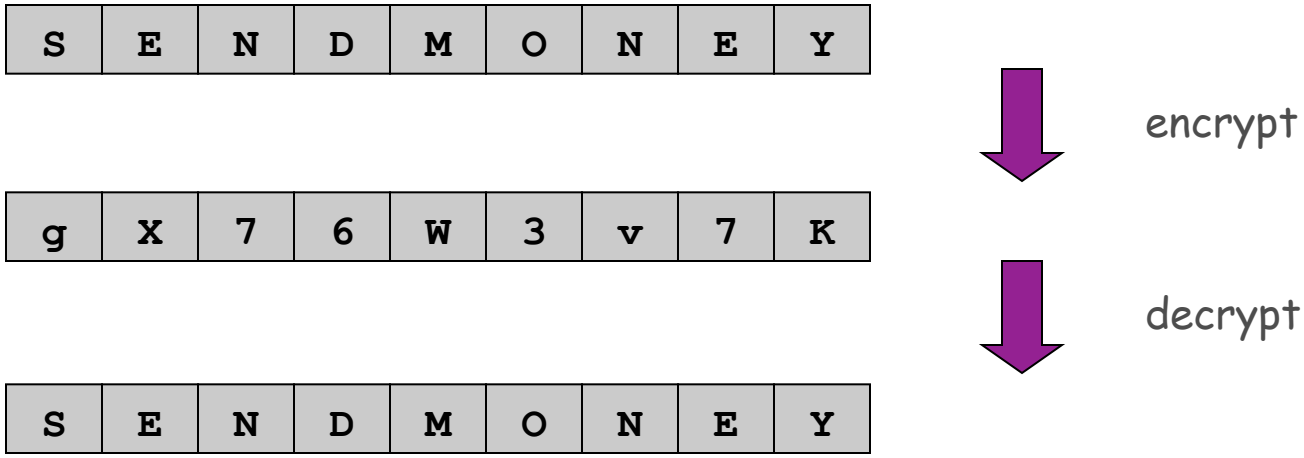


Key point Without the pad, Eve cannot understand the message

Penn Engineering

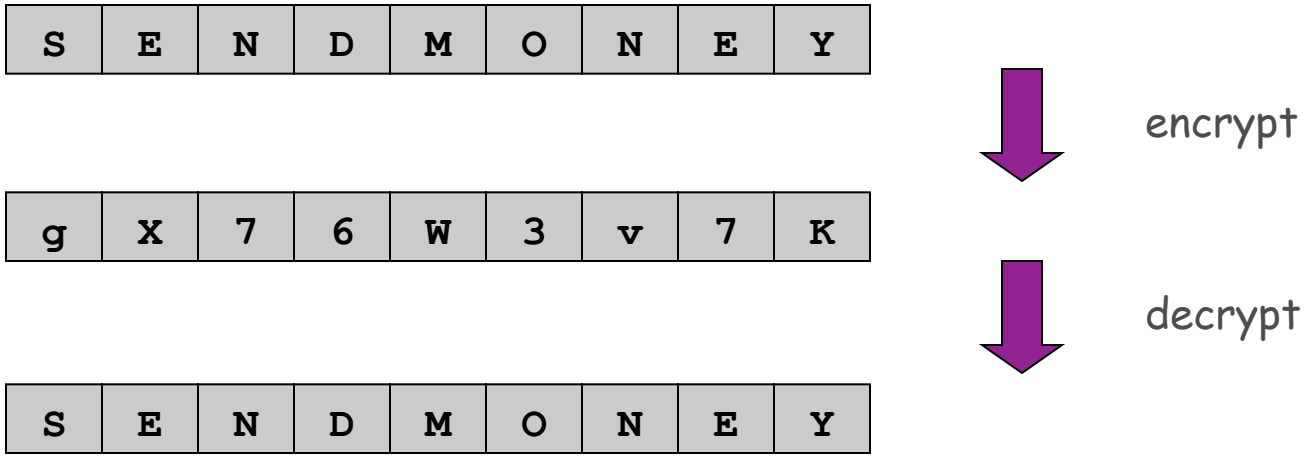
Encryption Machine

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, …

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

computer with a lens

computer with earbuds

computer with a radio

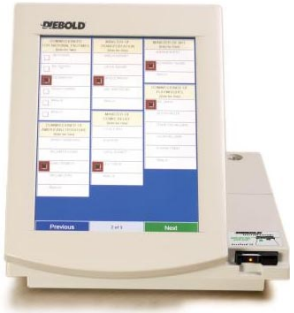
A Digital World

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

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

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

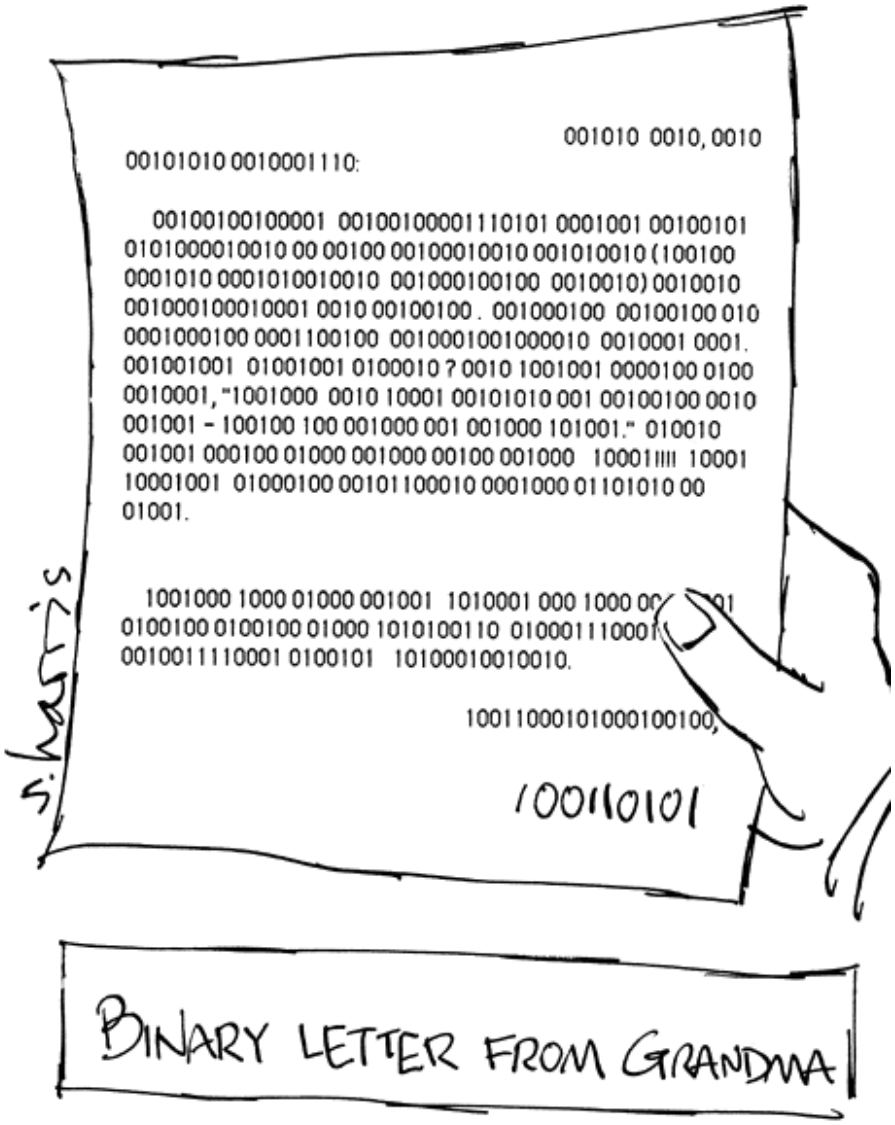
computer with
a cash dispenser



computer with
a ballot box

computer with
a heating element

Penn Engineering



NUMBER SYSTEMS AND DATA REPRESENTATION

There are 10 kinds of people in the world, those who use binary, and those who don't.

Penn Engineering

Base 10 (Decimal numbers)

What does 157 mean?

$$157 = 1 \times 100 + 5 \times 10 + 7 \times 1$$
$$= 1 \times 10^2 + 5 \times 10^1 + 7 \times 10^0$$

Decimal Numbering System

The Arabic numbering system uses ten symbols and is a base ten numbering system. In any numbering system the base is raised to consecutive powers to determine positional (place) values.

10^3	10^2	10^1	10^0	base^{powers}
1000	100	10	1	place values
	2	3	1	
	4	1	3	
7	8	6	0	

$231 = 2 * 10^2 + 3 * 10^1 + 1 * 10^0$

Penn Engineering

Computer Representation
Binary System

At the hardware level computers must store numbers using either the presence or absence of a voltage. Then there are only two states (symbols) available for number representation. So base 2 (binary) is a suitable system for computers.

2^6	2^5	2^4	2^3	2^2	2^1	2^0	base$^{\text{powers}}$
64	**32**	**16**	**8**	**4**	**2**	**1**	place values

Positional Systems

What value does 1110 represent?

Undetermined until the numbering system base is stated.

Base 10 place values:	10,000	1,000	100	10	1
Base 2 place values:	16	8	4	2	1
		1	1	1	0

base		**base 10**
10?	1*1000 + 1*100 + 1*10 + 0*1	1,110
2?	1*8 + 1*4 + 1*2 + 0*1	14

Base 10 vs Base 2

Base 10

157

$157 = 1 \times 100 + 5 \times 10 + 7 \times 1$

$= 1 \times 10^2 + 5 \times 10^1 + 7 \times 10^0$

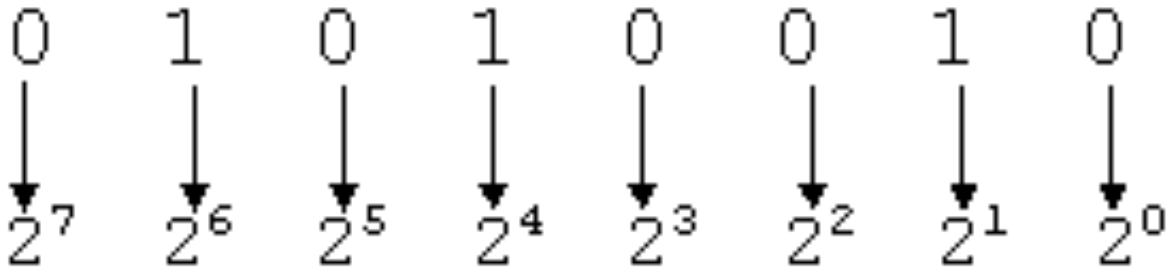
Base 2

$1011 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

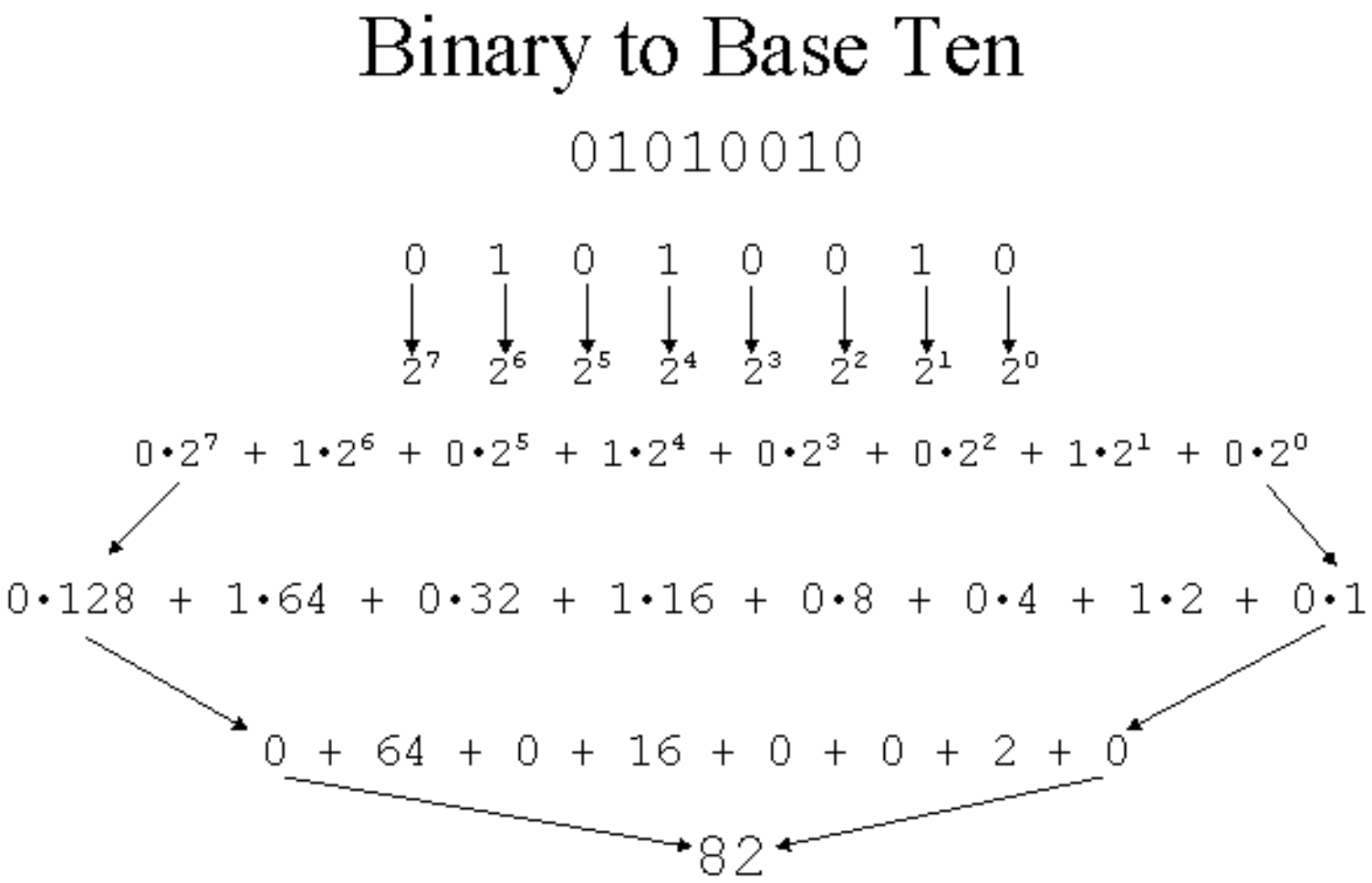
$1011 = 1 \times 8 + 0 \times 4 + 1 \times 2 + 1 \times 1$

Binary to Base Ten

01010010



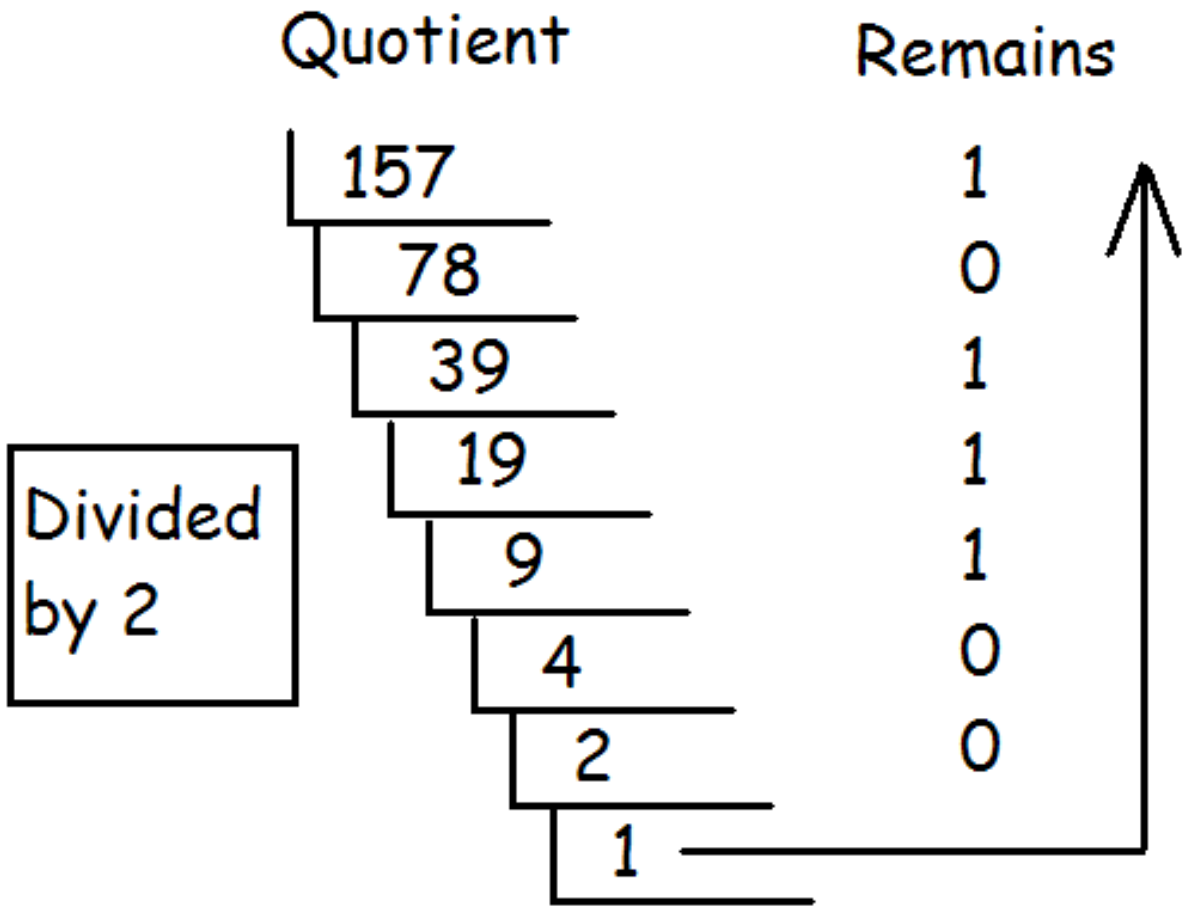
Binary to Base Ten



Penn Engineering

Base 10 to Binary

What is 157_{10} in binary?



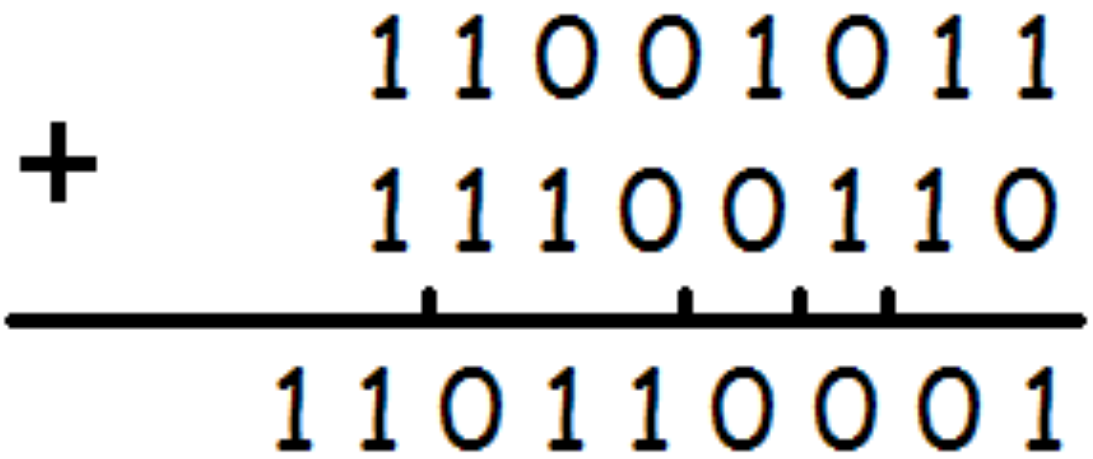
Answer: 10011101

Binary mathematics

0+0=0

1+0=1

1+1=10



Representing Negative Integers

TOY words are 16 bits each.

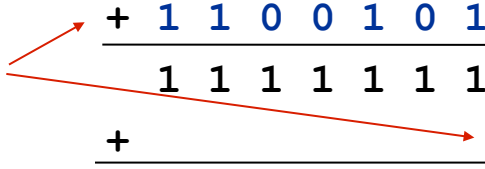
- We could use 16 bits to represent 0 to $2^{16} - 1$.
- We want negative integers too.
- Reserving half the possible bit-patterns for negative seems fair.

Highly desirable property. If x is an integer, then the representation of -x, when added to x, is zero.

```
    x          0 0 1 1 0 1 0 0
+(-x)       + ? ? ? ? ? ? ? ?
-------    ------------------
    0          0 0 0 0 0 0 0 0
```

```
    x          0 0 1 1 0 1 0 0
               + 1 1 0 0 1 0 1 1
               -----------------
+(-x)          1 1 1 1 1 1 1 1
               +               1
-------    ------------------
    0          0 0 0 0 0 0 0 0
```

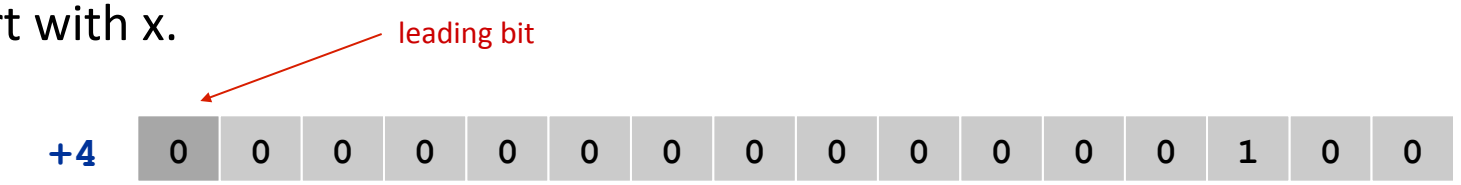
flip bits and add 1



Two's Complement Integers

To compute -x from x:

- Start with x.



- Flip bits.

- Add one.

Two's Complement Integers

		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
dec	hex	binary															
+32767	7FFF	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
		. . .															
+4	0004	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
+3	0003	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
+2	0002	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
+1	0001	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
+0	0000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-1	FFFF	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
-2	FFFE	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
-3	FFFD	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
-4	FFFC	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
		. . .															
-32768	8000	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Hexadecimal (base 16)

Binary code is too long in representation.
Hex is much shorter.

Converting a binary number to a Hex number is relatively easy

- Every 4 bits can convert to a single Hex value

Problem: we are short of numbers

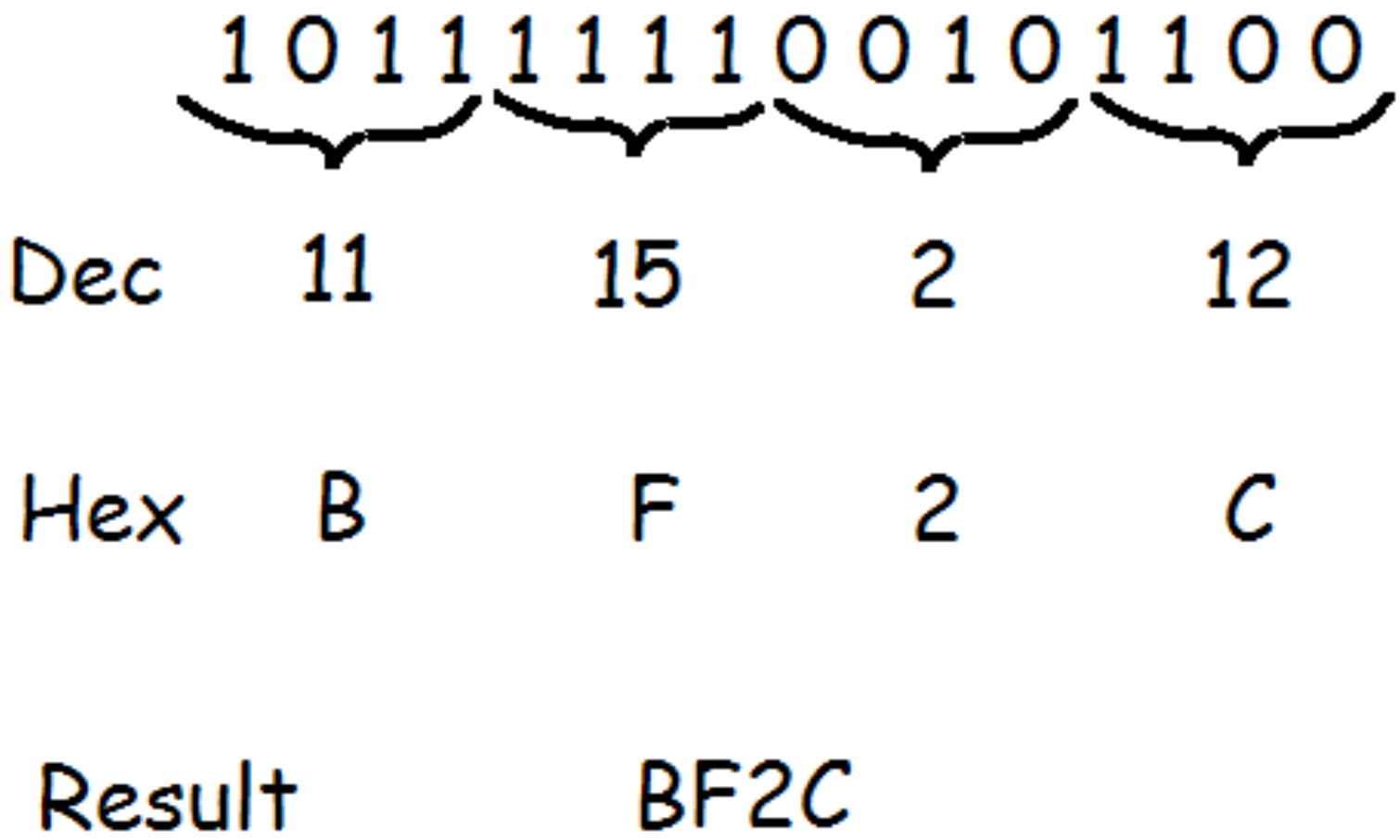
- A=10 B=11 C=12 D=13 E=14 F=15

Hexadecimal (Base 16) Representation

Binary	HEX
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7

Binary	HEX
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Example



Bits, Bytes, and Words

Information in the computer are stored as a groups of binary digits. An individual digit is called a **bit**.

Bits are grouped into 8-bit collections called **bytes**. Memory is normally measured in terms of bytes.

- 256 possible values from 0-255 base 10 or
- 00000000 to 11111111 base 2

Bytes are further grouped into 4 or more byte groupings to make up a computer **word**. The most common word sizes in modern computers is 32 bits (4 8-bit bytes) or 64-bits

BACK TO ENCRYPTION

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

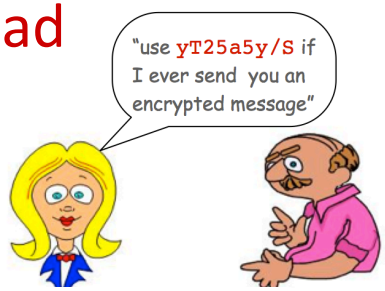
char	dec	binary
A	0	000000
B	1	000001
…	…	…
w	22	010110
…	…	…

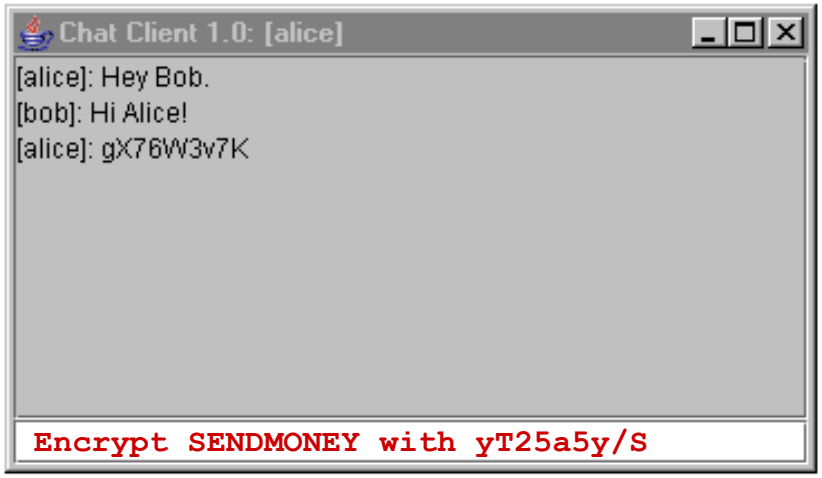
S	**E**	**N**	**D**	**M**	**O**	**N**	**E**	**Y**	message
010010	000100	001101	000011	001100	001110	001101	000100	011000	base64
110010	010011	110110	111001	011010	111001	100010	111111	010010	random bits
100000	010111	111011	111010	010110	110111	101111	111011	001010	XOR
g	**X**	**7**	**6**	**W**	**3**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

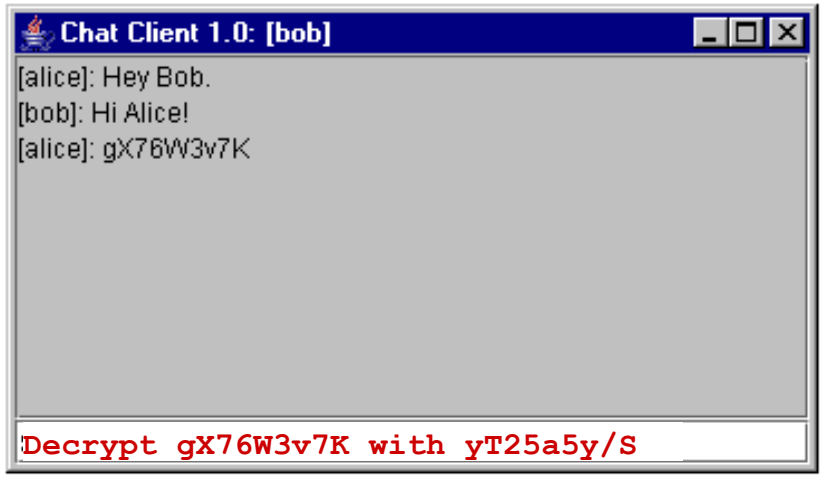
Secure Chat (review)

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



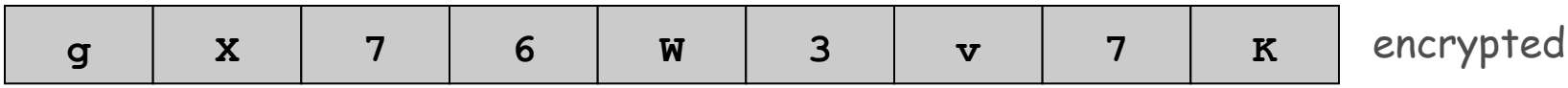




One-Time Pad Decryption

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

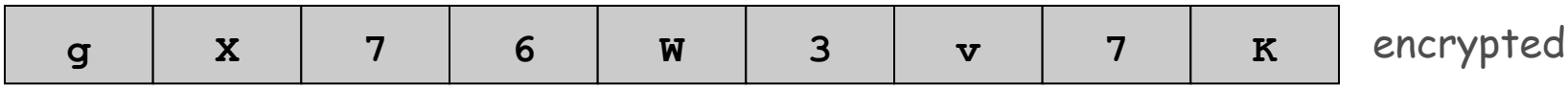
x	y	x ^ y
0	0	0
0	1	1
1	0	1
1	1	0

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

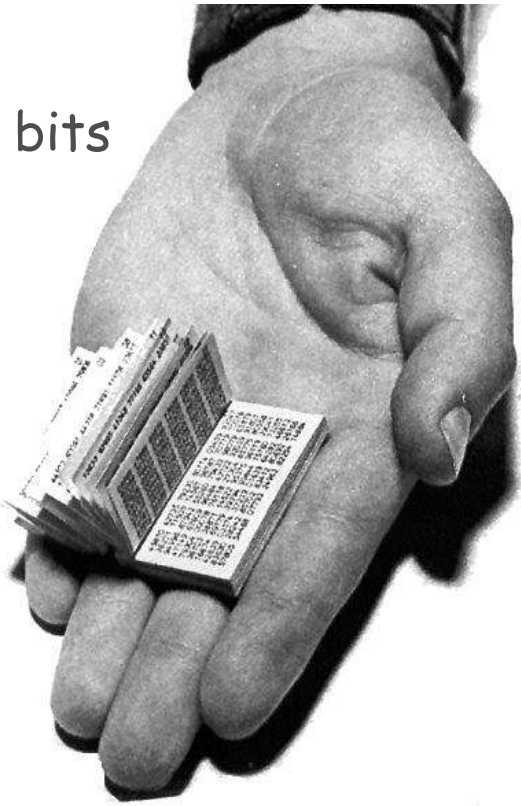
Good

- 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

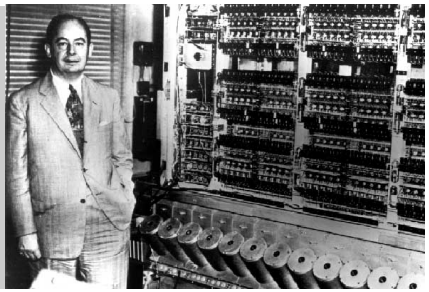
Practical middle-ground

- 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)

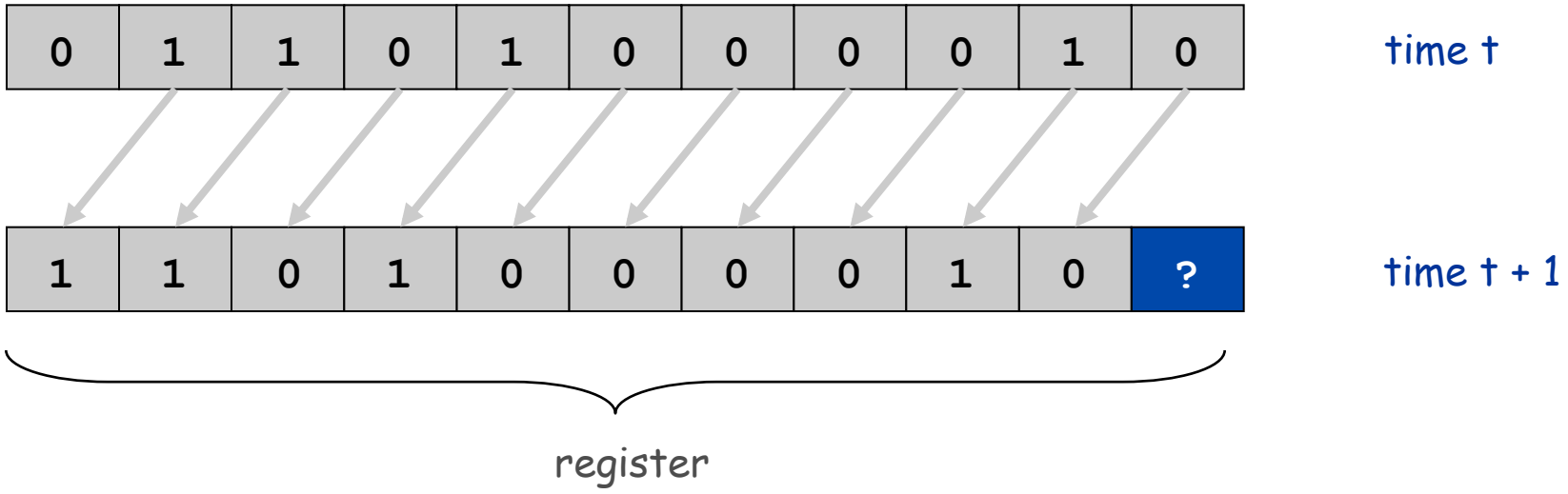


Penn Engineering

Shift Register

Shift register terminology.

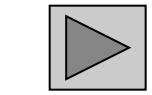
- Bit: 0 or 1
- Cell: storage element that holds one bit
- Register: sequence of cells
- Seed: initial sequence of bits
- Shift register: when clock ticks, bits propagate one position to left



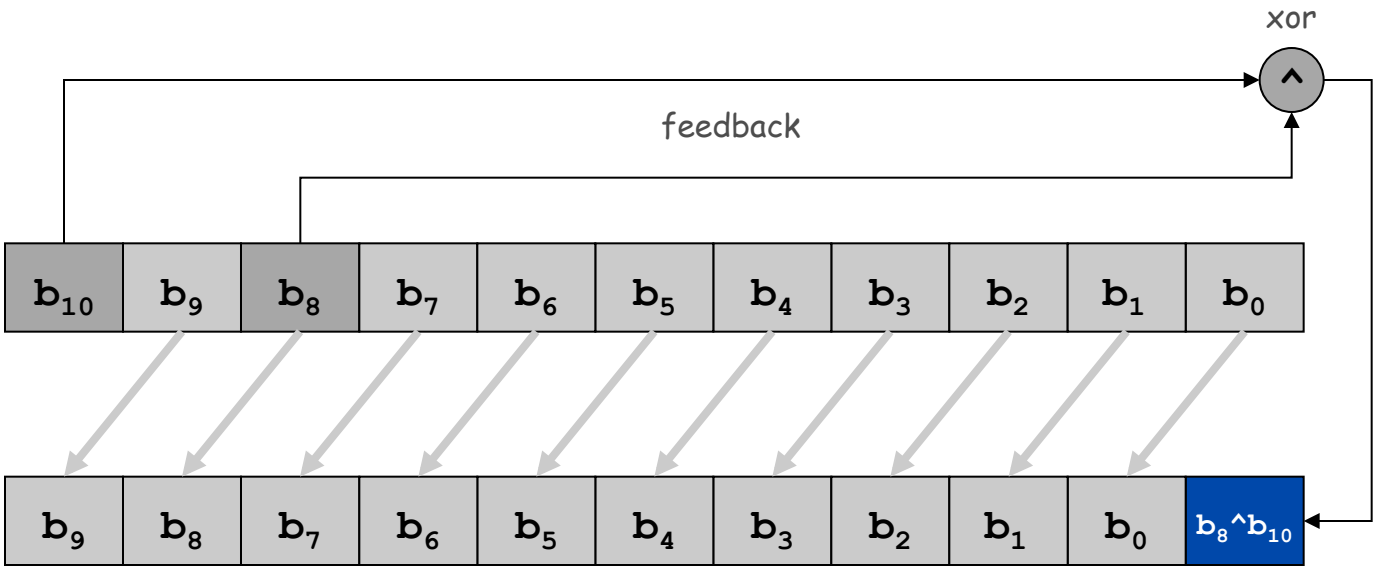
Linear Feedback Shift Register (LFSR)

{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

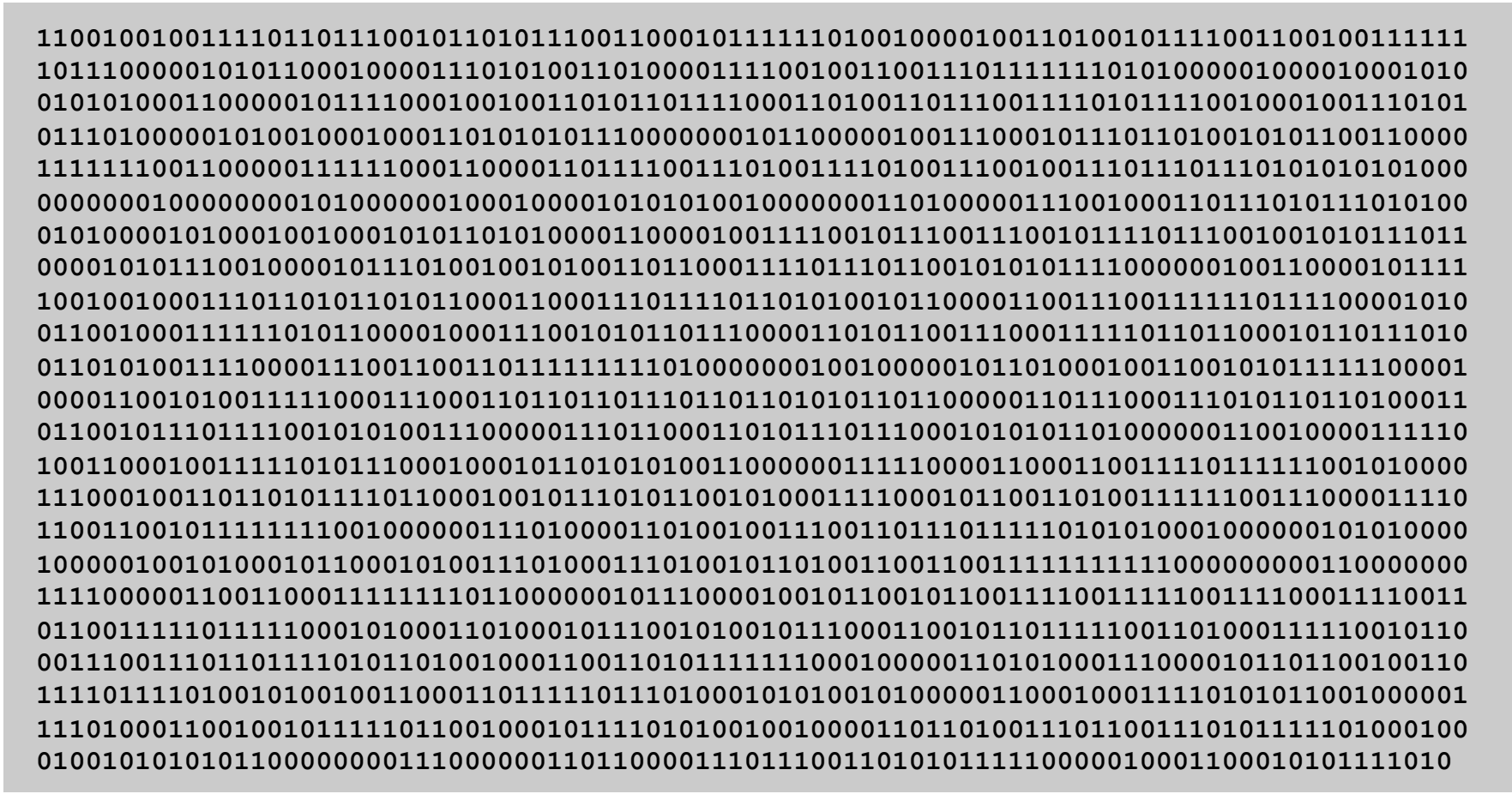


LFSR demo



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

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

A Profound Idea

Programming Can write a Java program to simulate the operations of any abstract machine

- Basis for theoretical understanding of computation
- Basis for bootstrapping real machines into

```
public class LFSR {
   private int seed[];
   private final int tap;
   private final int N;

   public LFSR(String seed, int tap) { … }

   public int step() { … }

   public static void main(String[] args) {
      LFSR lfsr = new LFSR("01101000010", 8);
      for (int i = 0; i < 2000; i++)
         StdOut.println(lfsr.step());
   }
}
```

```
% java LFSR
11001001001111011011100101101
01110011000101111110100100001
00110100101111001100100111...
```

Penn Engineering

A Closing Question

Q. What is a random number?

LFSR does not produce random numbers

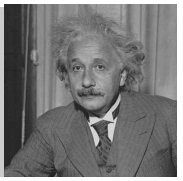
- It is a very simple deterministic machine
- But not obvious how to distinguish the bits it produces from random

Q. Are truly random processes found in nature?

- Motion of cosmic rays or subatomic particles?
- Mutations in DNA?

Q. Or, is the natural world a (not-so-simple) deterministic machine?

" God does not play dice. "
– Albert Einstein



Penn Engineering

Image Steganography

The art of hiding secret messages in images

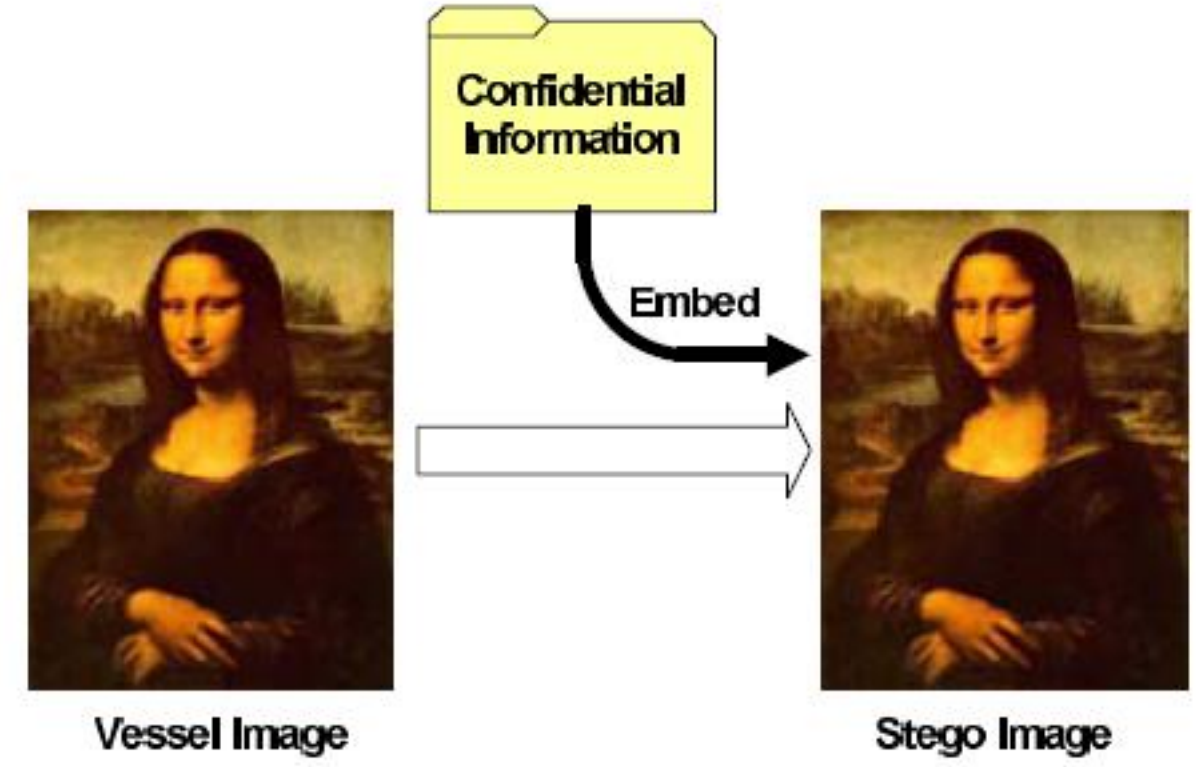


Image Steganography

The art of hiding secret messages in images

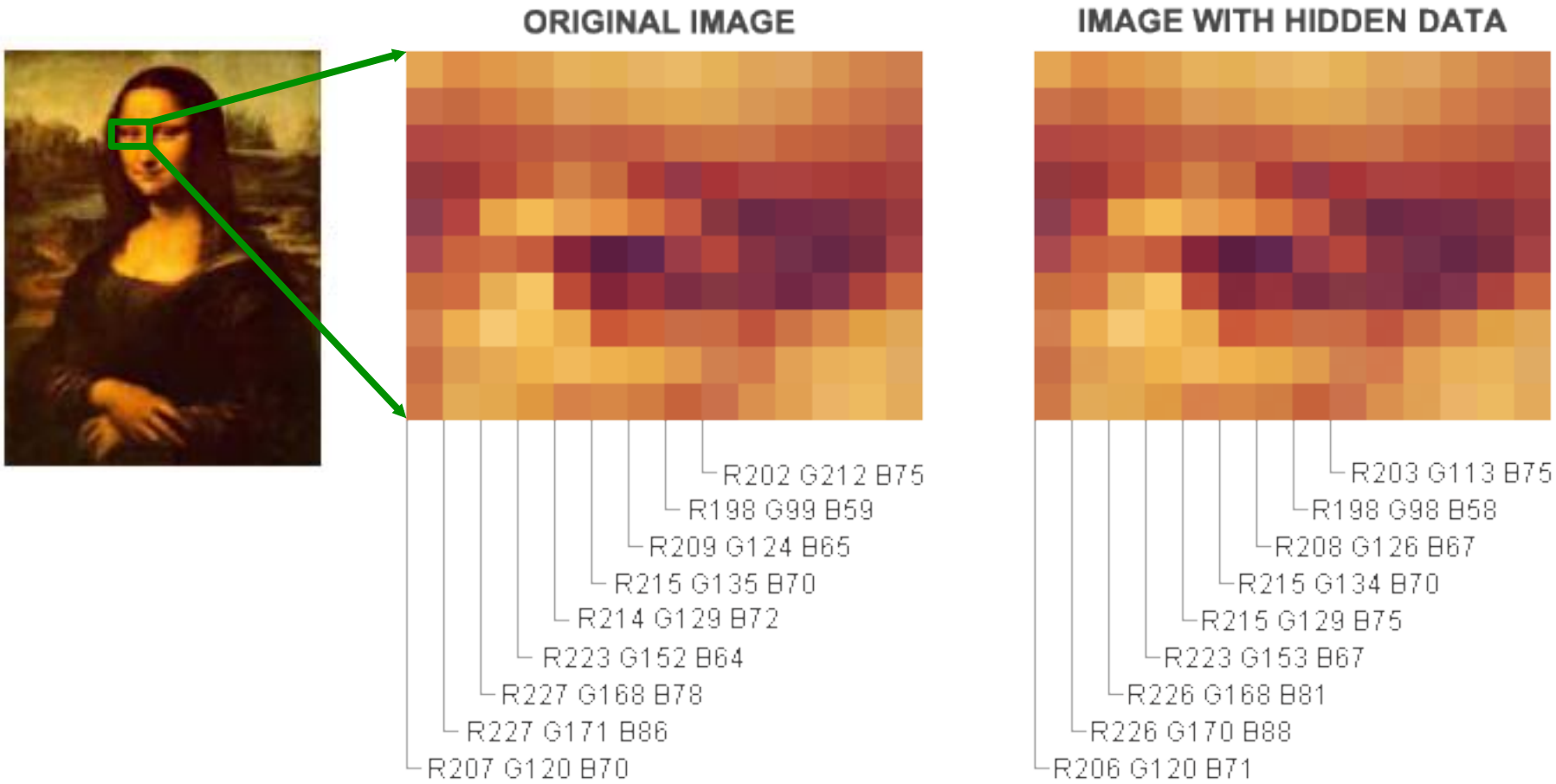
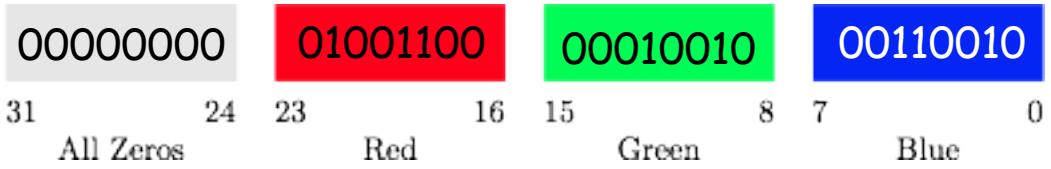


Image Steganography: How it Works

- Recall that each pixel of an image is 32 bits



- The right-most bit of each color is the least significant
  - Changing it by a small amount results in an imperceptible change to the color

- So, we can hide information in it!

- For simplicity, we'll only hide information in the blue channel

Image Steganography: How It Works

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

R207 G120 B71 ← 1000111

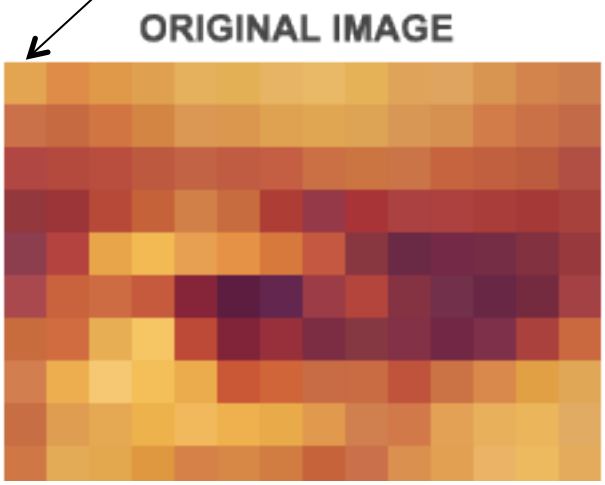
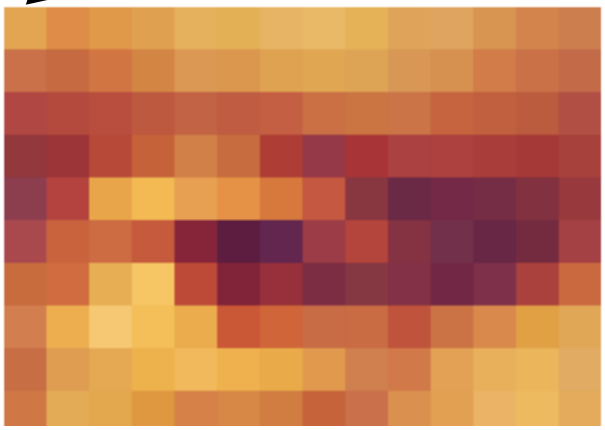


Image Steganography: How It Works

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

R207 G120 B71 ← ~~1000111~~



R207 G120 B70 ← 1000110

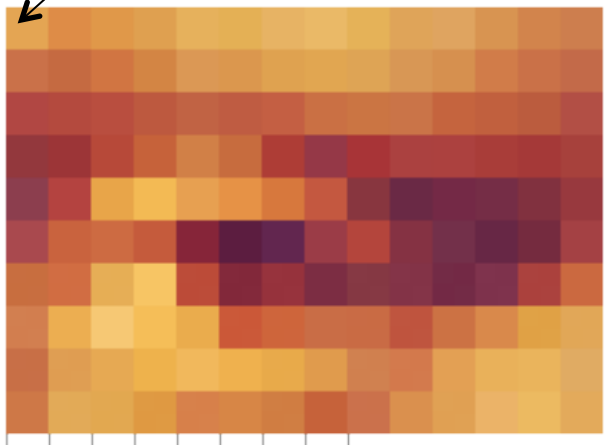


Image Steganography on Printers

- It has been alleged that laser printer manufacturers, perhaps in collusion with federal agencies, insert steganographic signatures on every piece of paper printed on a laser printer.
  - These signatures encode information identifying the serial number of the printer as well as a timestamp and other information that can be used to trace the originator of a printout.

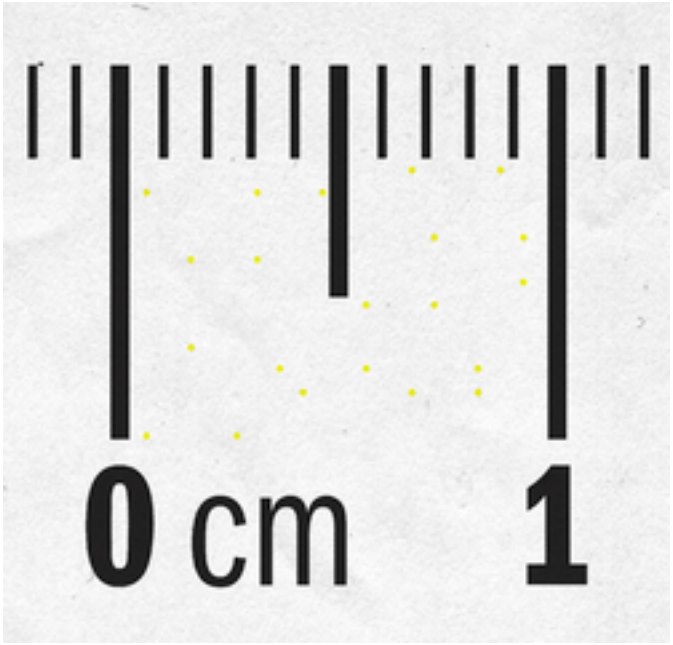
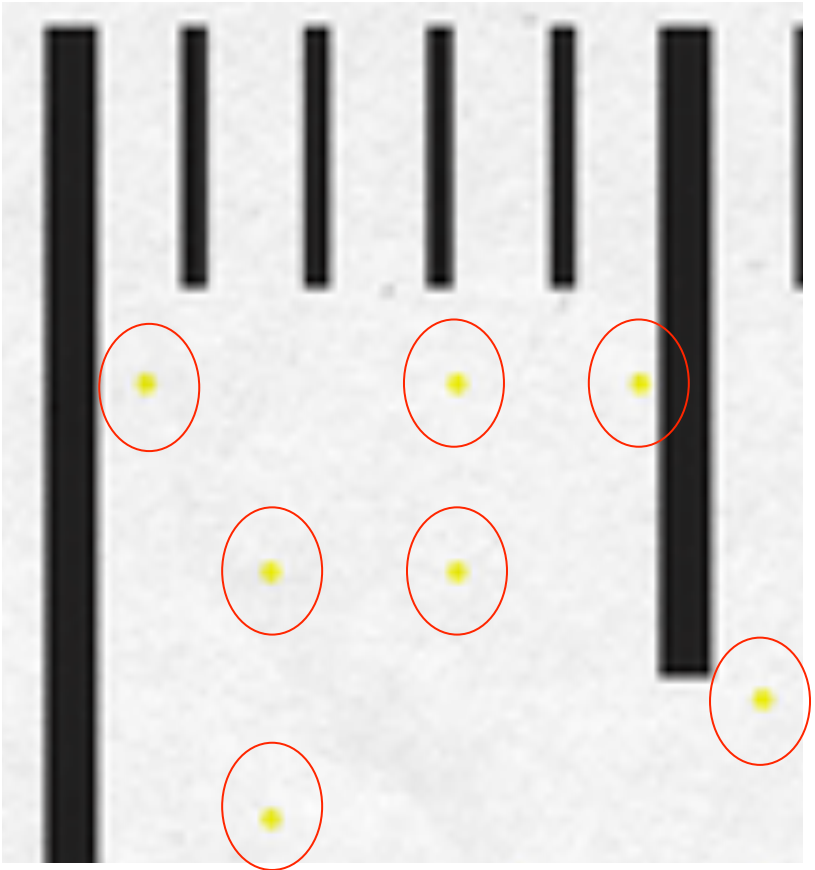


Image Steganography on Printers

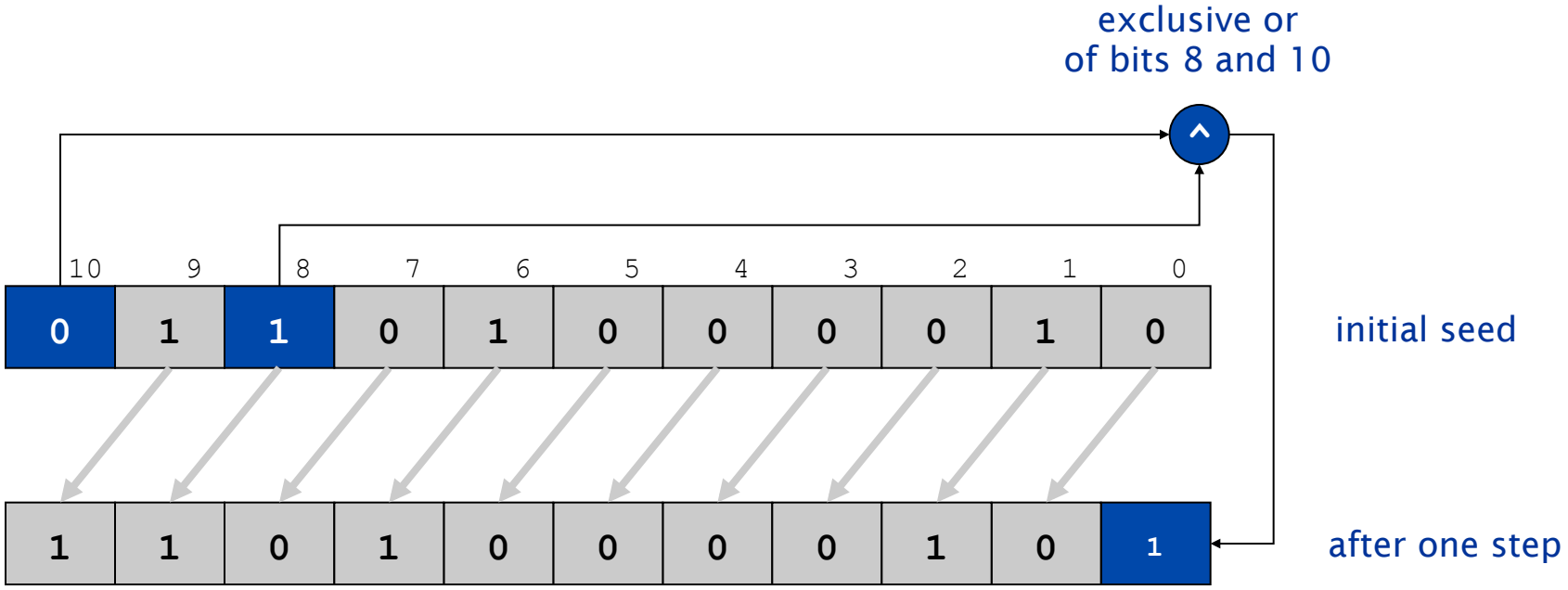
- It has been alleged that laser printer manufacturers, perhaps in collusion with federal agencies, insert steganographic signatures on every piece of paper printed on a laser printer.
  - These signatures encode information identifying the serial number of the printer as well as a timestamp and other information that can be used to trace the originator of a printout.



Penn Engineering

Extra Slides

Linear Feedback Shift Register



One step of an 11-bit LFSR with initial seed 01101000010 and tap at position 8