Lecture Notes

Advanced Discrete Structures

COT 4115.001 S15

2015-01-27

Recap

- ADFGX Cipher
- Block Cipher
  - Modes of Operation
- Hill Cipher
- Inverting a Matrix $\pmod{n}$

Hill Cipher Example

- <u>Encryption</u>:
  - Multiple coded vector by matrix (mod 26)

Coded Vectors: $(18,4,4),\ (18,15,14),\ (19,17,20),\ (13,23,23)$

$$(18 \quad 4 \quad 4)\begin{pmatrix} 3 & 22 & 17 \\ 8 & 2 & 11 \\ 23 & 5 & 19 \end{pmatrix} = (178 \quad 424 \quad 426) \equiv (22 \quad 8 \quad 10) \pmod{26}$$

$$(18 \quad 15 \quad 14)\begin{pmatrix} 3 & 22 & 17 \\ 8 & 2 & 11 \\ 23 & 5 & 19 \end{pmatrix} = (496 \quad 496 \quad 737) \equiv (2 \quad 2 \quad 9) \pmod{26}$$

$$(19 \quad 17 \quad 20)\begin{pmatrix} 3 & 22 & 17 \\ 8 & 2 & 11 \\ 23 & 5 & 19 \end{pmatrix} = (653 \quad 552 \quad 890) \equiv (3 \quad 6 \quad 6) \pmod{26}$$

$$(13 \quad 23 \quad 23)\begin{pmatrix} 3 & 22 & 17 \\ 8 & 2 & 11 \\ 23 & 5 & 19 \end{pmatrix} = (752 \quad 447 \quad 911) \equiv (24 \quad 5 \quad 1) \pmod{26}$$

Encrypted Vectors: $(22,8,10),\ (2,2,9),\ (3,6,6),\ (24,5,1)$

Ciphertext: `W I K  C C J  D G G  Y F B`

Hill Cipher Example

Using the Gauss-Jordan elimination method:

$$\left(\begin{array}{ccc|ccc} 3 & 22 & 17 & 1 & 0 & 0 \\ 8 & 2 & 11 & 0 & 1 & 0 \\ 23 & 5 & 19 & 0 & 0 & 1 \end{array}\right) \rightarrow \left(\begin{array}{ccc|ccc} 1 & 0 & 0 & 11 & 9 & 0 \\ 0 & 1 & 0 & 5 & 2 & 19 \\ 0 & 0 & 1 & 10 & 5 & 6 \end{array}\right) \pmod{26}$$

which means that if

$$\begin{pmatrix} a & b & c \end{pmatrix} \begin{pmatrix} 3 & 22 & 17 \\ 8 & 2 & 11 \\ 23 & 5 & 19 \end{pmatrix} \equiv \begin{pmatrix} A & B & C \end{pmatrix} \pmod{26}$$

then

$$\begin{pmatrix} a & b & c \end{pmatrix} \equiv \begin{pmatrix} A & B & C \end{pmatrix} \begin{pmatrix} 11 & 9 & 0 \\ 5 & 2 & 19 \\ 10 & 5 & 6 \end{pmatrix} \pmod{26}$$

Hill Cipher Example

- Decryption:
  - Multiply the encrypted vectors by the inverse matrix:

Encrypted Vectors: $(22,8,10),\ (2,2,9),\ (3,6,6),\ (24,5,1)$

$$\begin{pmatrix} 22 & 8 & 10 \end{pmatrix} \begin{pmatrix} 11 & 9 & 0 \\ 5 & 2 & 19 \\ 10 & 5 & 6 \end{pmatrix} \equiv \begin{pmatrix} 18 & 4 & 4 \end{pmatrix} \pmod{26}$$

$$\begin{pmatrix} 2 & 2 & 9 \end{pmatrix} \begin{pmatrix} 11 & 9 & 0 \\ 5 & 2 & 19 \\ 10 & 5 & 6 \end{pmatrix} \equiv \begin{pmatrix} 18 & 15 & 14 \end{pmatrix} \pmod{26}$$

$$\begin{pmatrix} 3 & 6 & 6 \end{pmatrix} \begin{pmatrix} 11 & 9 & 0 \\ 5 & 2 & 19 \\ 10 & 5 & 6 \end{pmatrix} \equiv \begin{pmatrix} 19 & 17 & 20 \end{pmatrix} \pmod{26}$$

$$\begin{pmatrix} 24 & 5 & 1 \end{pmatrix} \begin{pmatrix} 11 & 9 & 0 \\ 5 & 2 & 19 \\ 10 & 5 & 6 \end{pmatrix} \equiv \begin{pmatrix} 13 & 23 & 23 \end{pmatrix} \pmod{26}$$

Decrypted Vectors: $(18,4,4),\ (18,15,14),\ (19,17,20),\ (13,23,23)$

Plaintext: S E E S P O T R U N X X

Classical Cryptosystems - Section 2.7

BLOCK CIPHERS

Hill Cipher Attacks

- **Ciphertext only:**
  - Changing one letter of the plaintext usually changes whole block
  - If block sizes are small (<4), the little is changed and frequency analysis can be run on the whole blocks
- **Easy to break with:**
  - Known plaintext
  - Chosen plaintext
  - Chosen ciphertext

Hill Cipher - Known Plaintext

Plaintext:

Plaintext:	d	o	n	t	b	e	e	v	i	l
	3	14	13	19	1	4	4	21	8	11
Ciphertext:	B	J	E	Z	X	T	N	I	B	G
	1	9	4	25	23	19	13	8	1	6

Try various block sizes: For $n = 2$,

$$\begin{pmatrix} 3 & 14 \\ 13 & 19 \end{pmatrix} \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} \equiv \begin{pmatrix} 1 & 9 \\ 4 & 25 \end{pmatrix} \pmod{26}$$

and

$$\begin{pmatrix} 3 & 14 \\ 13 & 19 \end{pmatrix}^{-1} \equiv \begin{pmatrix} 9 & 18 \\ 13 & 11 \end{pmatrix} \pmod{26}$$

so

$$\begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} \equiv \begin{pmatrix} 9 & 18 \\ 13 & 11 \end{pmatrix} \begin{pmatrix} 1 & 9 \\ 4 & 25 \end{pmatrix} \equiv \begin{pmatrix} 3 & 11 \\ 5 & 2 \end{pmatrix} \pmod{26}$$

Hill Cipher - Known Plaintext

- If the plaintext matrix does not have an inverse, try a different set of plaintext blocks
  - Recall that matrix M is invertible if
  $$\det M \not\equiv 0 \pmod{26}.$$
- If none of the plaintext matrices are invertible, try a different block size.

Hill Cipher – Chosen Plaintext

1. Try out different possibilities for n until you find the right one

2. Encrypt the following plaintext:

   – 1st Block: `baaaa…a = 10000…0`

   – 2nd Block: `abaaa…a = 01000…0`

   – nth Block: `aaaaa…b = 00000…1`

3. Encryption matrix is the output

Hill Cipher – Chosen Plaintext

For $n = 2$,

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} \equiv \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} \pmod{26}$$

Invert the encryption matrix to find the decryption matrix.

Hill Cipher – Chosen Ciphertext

- Similar to chosen plaintext
  - Guess and check the key size
- Choose to decrypt the identity matrix

$$\begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix}^{-1} \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} \equiv \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \pmod{26}$$

- Result is the decryption matrix
  - Invert to find encryption matrix

Confusion and Diffusion

- Claude Shannon: *Communication Theory of Secrecy Systems* (1949)
  - **Confusion:** Each letter of the cipher should depend on multiple parts of the key
  - **Diffusion:** Changing a letter of the plaintext should change multiple letters of the ciphertext
    - Diffusion causes error-propogation

Classical Cryptosystems - Section 2.8

BINARY NUMBERS AND ASCII

Base Systems

<u>Example</u>: (Base 10)

$$2{,}345_{10} = \mathbf{2} \cdot 10^3 + \mathbf{3} \cdot 10^2 + \mathbf{4} \cdot 10^1 + \mathbf{5} \cdot 10^0$$

<u>Example</u>: (Base 2 or Binary)

$$1{,}101_2 = \mathbf{1} \cdot 2^3 + \mathbf{1} \cdot 2^2 + \mathbf{0} \cdot 2^1 + \mathbf{1} \cdot 2^0$$

$$= 8 + 4 + 1 = 13_{10}$$

Binary

- Binary number: a string of 0's and 1's
- Each 0 or 1 is called a **bit**
- A representation that takes 8 bits is called an **8-bit number** or a **byte**
- The largest number a byte can represent is

$$11111111_2 = 2^8 - 1 = 255$$

ASCII

- American Standard Code for Information Interchange
- A standard way to represent common alphabet characters
- Each character is represented using 7 bits
  - Extra bit in the byte used for a parity check (to see if an error occurred in transmission)

ASCII

Symbol	!	“	#	$	%	&	‘
Decimal	33	34	35	36	37	38	39
Binary	0100001	0100010	0100011	0100100	0100101	0100110	0100111
(	)	*	+	,	-	.	/
40	41	42	43	44	45	46	47
0101000	0101001	0101010	0101011	0101100	0101101	0101110	0101111
0	1	2	3	4	5	6	7
48	49	50	51	52	53	54	55
0110000	0110001	0110010	0110011	0110100	0110101	0110110	0110111
8	9	:	;	<	=	>	?
56	57	58	59	60	61	62	63
0111000	0111001	0111010	0111011	0111100	0111101	0111110	0111111
@	A	B	C	D	E	F	G
64	65	66	67	68	69	70	71
1000000	1000001	1000010	1000011	1000100	1000101	1000110	1000111

ASCII

- Each message can be coded in binary:

Message: `Yes!`

Binary:

```
   Y       e       s       !
1011001 1100101 1110011 1000001
```

Classical Cryptosystems - Section 2.9

ONE-TIME PADS

One-Time Pad (1918)

- An “unbreakable” cryptosystem
- Developed by Gilbert Vernam and Joseph Mauborgne
- Reported that during the Cold War the “hot line” between Washington, D.C., and Moscow used one-time pads

One-Time Pad

- Encryption:
  - Code the message in binary
  - Choose a binary key as long as the message
  - Ciphertext is XOR between message and key

p	q	p XOR q
0	0	0
0	1	1
1	0	1
1	1	0

One-Time Pad

- **Example (Encryption)**

```
plaintext:     0101011100101110 (mod 2)
     key:    + 1110001110001110 (mod 2)
---------------------------------------
ciphertext:    1011010010100000 (mod 2)
```

One-Time Pad

- **Example (Decryption)**

```
ciphertext:     1011010010100000 (mod 2)
      key:  +   1110001110001110 (mod 2)
-----------------------------------------
 plaintext:     0101011100101110 (mod 2)
```

One-Time Pad

- **Example (Decryption)**

ciphertext:	1011010010100000	(mod 2)
key: +	1110001110001110	(mod 2)
plaintext:	0101011100101110	(mod 2)

One-Time Pad

The Good:

- Brute force and frequency attacks *do not work*
- All that can be gained from the ciphertext is the message and key length
- Knowing part of the key doesn't give any information about the rest of the key

The Bad:

- Knowing the key $\Leftrightarrow$ Knowing the plaintext
- Key can only be used once (otherwise can attack with crib-dragging)

Key Generation

- A “random” key is desirable over a phrase
  - Knowledge of part of a key gives no knowledge of the rest of the key
  - The seed for a random key may be small, but can generate a longer pseudo-random string

Classical Cryptosystems - Section 2.10

PSEUDO-RANDOM BIT GENERATION

Hardware Random Number Generators

- Sampling from a natural process
  - may not be sufficiently random
  - may take too long/expensive to sample
  - observable to others
- Examples
  - Thermal noise
  - Radiation / Photoelectric effect
  - Quantum states

Pseudo-Random Bit Generation

- For many purposes a computer algorithm can generate pseudo-random bits
  - Initial input called the **seed** and produces output bitstream

Example:

- C library contains `rand()` function, which is a **linear congruential generator**
  - produces a sequence of numbers $x_1, x_2, \ldots$ such that

$$x_n \equiv a x_{n-1} + b \pmod{m}$$

from a seed x_0 and parameters a, b, and m.

Pseudo-Random Bit Generation

- *Any* polynomial congruence generator is insecure
  - Can be predicted even when parameters a, b, m are unknown
- **One-way functions**: Easy to compute in one direction and hard to find the inverse
  - Digital Encryption Standard (DES)
  - Secure Hash Algorithm (SHA)
    - SSL is based on SHA

Blum-Blum-Shub (BBS)

1. Generate two large primes:
$$p, q \equiv 3 \pmod 4)$$
2. Let: $$n = pq$$
3. Choose a large x relatively prime to n.
4. Set: $$x_0 \equiv x^2 \pmod n$$
5. Let: $x_j \equiv {x_{j-1}}^2 \pmod n$ for $j = 1,2,\ldots$
6. Let b_j be the parity of the last bit of x_j.

Blum-Blum-Shub (BBS)

Example:

Choose $p, q \equiv 3 \pmod 4$:

$p = 134095639079$, $q = 3435316544843$

$$n = pq = 460660967519384246719597$$

Choose x relatively prime to n:

$$x = 151984354862728938426826$$

Blum-Blum-Shub (BBS)

$$\begin{aligned} x_0 &\equiv x^2 \pmod{n} \\ &\equiv 22489642930339275646808\mathbf{6} \pmod{n} \end{aligned}$$

$$\begin{aligned} x_1 &\equiv {x_0}^2 \pmod{n} \\ &\equiv 276799756549439688565\mathbf{8}\mathbf{2} \pmod{n} \end{aligned}$$

$$\begin{aligned} x_2 &\equiv {x_1}^2 \pmod{n} \\ &\equiv 389970735289039492719842\mathbf{3} \pmod{n} \end{aligned}$$

$$\begin{aligned} x_3 &\equiv {x_2}^2 \pmod{n} \\ &\equiv 239541338864411504485899\mathbf{9} \pmod{n} \end{aligned}$$

Blum-Blum-Shub (BBS)

$$b_0 = 6 \equiv 0 \pmod{2}$$

$$b_1 = 2 \equiv 0 \pmod{2}$$

$$b_2 = 3 \equiv 1 \pmod{2}$$

$$b_3 = 9 \equiv 1 \pmod{2}$$

$$\vdots$$

Or can take several bits ($k \leq \log_2 \log_2 n$) from each x_i:

$$b_0 = 086 \to 000, \quad b_1 = 582 \to 100, \; \ldots$$