

Characters: simple cryptographic algorithms

Lecture 09

Waterford Institute of Technology

February 11, 2016

John Fitzgerald

Presentation outline

Estimated duration presentation

Questions at end presentation

Topics discussed:

- Character encoding
 - ASCII
 - ANSI
 - Unicode
- Caesar cipher
- Vigenere cipher
- One time pad

Character encoding

Brief description

- Method to represent character in computing systems
- Character referred to as code point
- Example: letter A typically represented by decimal 65 (0x41)

```
char ch = 'A'; // print ch -> outputs A  
byte chb = (byte)ch; // print chb -> outputs 65
```

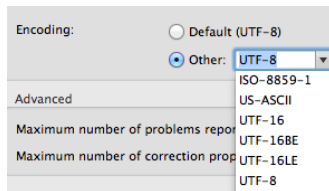
String object

Closely coupled to its encoding

String display:

- Essential to choose suitable character encoding.
- Example ISO/IEC 8859-1 does not contain € symbol.

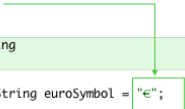
```
<?xml version="1.0" encoding="UTF-8"?>
```



String object

BlueJ editor: effect different character encodings

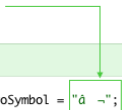
UTF-8



```
public class Encoding
{
    public static String euroSymbol = "e";

    public static void euroSymbol()
    {
        System.out.println("Euro symbol: " + euroSymbol);
    }
}
```

ISO-8859-1



```
public class Encoding
{
    public static String euroSymbol = "ä -";

    public static void euroSymbol()
    {
        System.out.println("Euro symbol: " + euroSymbol);
    }
}
```

Character encoding standards

Historical perspective

- EBCDIC (IBM)
- ASCII (128 characters)
 - May be stored in 7 bits
 - Storable in primitive **byte**
 - Contains 95 printable characters
 - Remaining 33 non-printable
- ANSI (a misnomer)
 - May be stored in 8 bits
 - First 7 same as ASCII
 - Term used for several encodings
- Unicode
 - Computing industry standard for the consistent encoding,

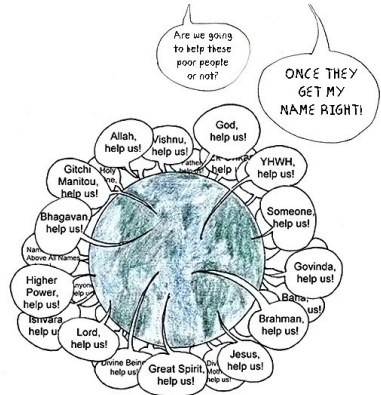
$[A - Z] \rightarrow [65 - 90]$
 $[a - z] \rightarrow [97 - 122]$

Character encoding standards

Choose your favourite term

Code page : alternative names

- codepage
- encoding
- charset
- character set
- coded character set (CCS)
- graphic character set
- character map



Unicode

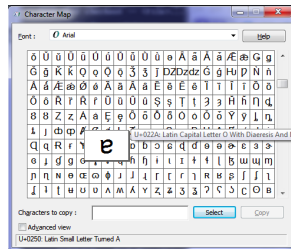
Computing industry encoding standard

Unicode 8.0 (2015)

- more than 120,000 characters
- 129 modern and historic scripts

Unicode implementations

- Unicode transformation format (UTF)
- UTF-8, UTF-16, UTF-32
- UTF-8 uses one byte for any ASCII



Character manipulation

Caesar cipher: a substitution cipher (no key)

Terminology

- Message text or plain text
- Cipher text: encrypted plain text
- Key: integral to encrypt decrypt

Encryption & Decryption

- Encrypt: shift plain text character
- Example: shift by 3 thus A becomes D
- Arithmetic operations on char valid



Caesar Cipher

Plain text comprise only upper case letters

- The fields

```
public class CaesarCipher {  
    // The number of characters to shift  
    int shift;  
    // Defines the printable range of ASCII characters  
    private int minAsciiUpper = 65;  
    private int maxAsciiUpper = 90;  
    ...  
}
```

Caesar Cipher

Plain text comprise only upper case letters

- The constructor

```
/**  
 * Constructs a CaesarCipher object  
 *  
 * @param shift A positive digit by which to shift plain text.  
 */  
public CaesarCipher(int shift) {  
    this.shift = Math.abs(shift);  
}
```

Caesar Cipher

Plain text comprise only upper case letters

- Encrypt string

```
/**
 * Encrypts the plain text string using helper method
 * Helper method: char encrypt(char ch)
 * @param The plaintext string
 * @return The ciphertext: encrypted plaintext
 */
String encrypt(String string)
{
    char[] chars = string.toCharArray();
    char[] encryptedChars = new char[chars.length];
    for (int i = 0; i < chars.length; i += 1) {
        encryptedChars[i] = encrypt(chars[i]);
    }
    return new String(encryptedChars);
}
```

Caesar Cipher

Plain text comprise only upper case letters

- Encrypt single character

```
/**
 * Encrypts a single character
 * @param ch The plaintext character
 * @return The encrypted character
 */
private char encrypt(char ch) {
    int encryptedChar = ch + shift;
    if (encryptedChar > maxAsciiUpper) {
        encryptedChar = minAsciiUpper + encryptedChar % maxAsciiUpper -
            1;
    }
    return (char)encryptedChar;
}
```

Caesar Cipher

Plain text comprise only upper case letters

- Decrypt string

```
/**
 * Decrypts the plain text string using helper method
 * Helper method: char decrypt(char ch)
 * @param The ciphertext string
 * @return The plaintext: decrypted plaintext
 */
String decrypt(String string)
{
    char[] chars = string.toCharArray();
    char[] encryptedChars = new char[chars.length];
    for (int i = 0; i < chars.length; i += 1) {
        decryptedChars[i] = decrypt(chars[i]);
    }
    return new String(decryptedChars);
}
```

Caesar Cipher

Plain text comprise only upper case letters

- Decrypt single character

```
/**
 * Decrypts a single character
 * @param ch The ciphertext character
 * @return The decrypted character
 */
char decrypt(char ch) {
    int range = maxAsciiUpper - minAsciiUpper;
    int decryptedChar = ch + range - shift + 1;
    if (decryptedChar > maxAsciiUpper) {
        decryptedChar = minAsciiUpper + decryptedChar % maxAsciiUpper - 1;
    }
    return (char)decryptedChar;
}
```

Caesar Cipher

Plain text comprise only upper case letters

- A unit test

```
public static void test {  
    // Unit test: The full alphabet range of shifts and then some.  
    for (int shift = 0; shift < 30; shift += 1) {  
        CaesarCipher caesarCipher = new CaesarCipher(shift);  
        String cipherTxt = caesarCipher.encrypt(plainText);  
        String dcryptTxt = caesarCipher.decrypt(cipherTxt);  
        if (!dcryptTxt.equals(plainText)) {  
            System.out.println("Oh oh: problem at GCHQ");  
        }  
        else {  
            System.out.println("Successful text " + (shift + 1));  
        }  
    }  
}
```


Caesar Cipher

Decryption without knowledge of shift

- Cipher-text only attack
- Very simple this case since only 26 possible values
- Brute-force attack tests all possible shift values
- Easy to refactor existing code

```
// Example brute force attack on ciphertext
// Invoke encrypt to obtain sample ciphertext.
// We then have plaintext against which to check outputs below.
String ciphertext = "GUVFVFNVCYNVAGRKGFGFGEVATVGFHERYLVF";
for (int shift = 1; shift <= 26; shift += 1)
{
    System.out.println(cc.decrypt(ciphertext, shift));
}
```

Vigenere Cipher

Vigenere Table

		PLAIN TEXT																									
		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
KEY	A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
	B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
	C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
	D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
	E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
	F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
	G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
	H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
	I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
	J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
	K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
	L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
	M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
	N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
	O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
	P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
	Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
	R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
	S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
	T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
	U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
	V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
	W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
	X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
	Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
	Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Vigenere Cipher

Key length matches plain text

- Plain text
 - VIGENERECIPHEREXAMPLE
- Key same length plaintext
 - ATTACKATDAWNATTACKATD

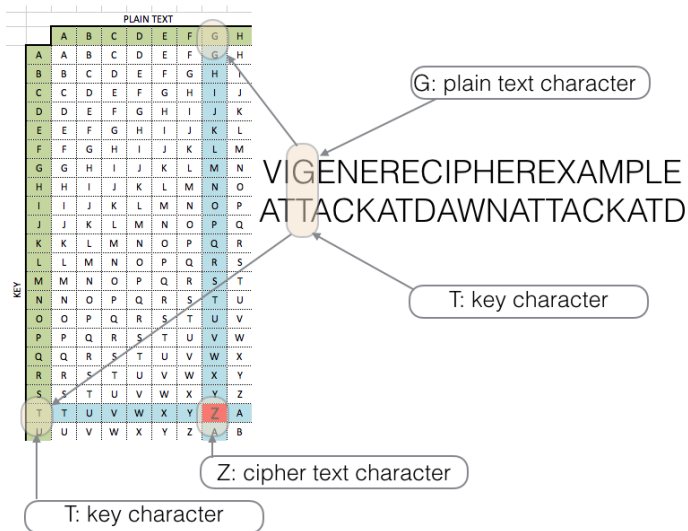
		PLAIN TEXT																										
KEY	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z		
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z		
	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A		
	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B		
	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C		
	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D		
	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E		
	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F		
	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G		
	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H		
	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I		
	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J		
	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K		
	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L		
	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M		
	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N		
	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O		
	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P		
	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q		
	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R		
	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S		
	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T		
	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U		
	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V		
	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W		
	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X		
	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y		

Encryption - Decryption

		PLAIN TEXT																									
		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
KEY	A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
	B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
	C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
	D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
	E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
	F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
	G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
	H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
	I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
	J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
	K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
	L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
	M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
	N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
	O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
	P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
	Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
	R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
	S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
	T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
	U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
	V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
	W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
	X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
	Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
	Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Vigenere Cipher

Encryption - Decryption



One Time Pad

Key same length as plaintext

- m denotes plaintext or message text
- k denotes key
- c denotes the cipher text or encrypted message
- $c = m \text{ xor } k$

a	b	a XOR b
0	0	0
0	1	1
1	0	1
1	1	0

m	0	1	1	0	1	1
k	1	0	1	1	0	0
c	1	1	0	1	1	1

One Time Pad

Key same length as plaintext

Observe from table:

- $c = m \text{ xor } k$
- $m = c \text{ xor } k$

m	0	1	1	0	1	1
k	1	0	1	1	0	0
c	1	1	0	1	1	1
c xor k	0	1	1	0	1	1

One Time Pad

Instantiation

```
public class OneTimePad
{
    byte[] m; // plain text message
    byte[] k; // key
    byte[] c; // cipher text — encrypted message under key
    /**
     * Constructs OneTimePad object
     * Converts message to byte array
     * Generates a secure key
     * Initializes the cipher text byte array
     * @param m The plaintext message
     */
    public OneTimePad(String m)
    {
        this.m = m.getBytes();
        k = new byte[m.length()];
        new SecureRandom().nextBytes(k);
        c = new byte[m.length()];
    }
}
```


One Time Pad

Encryption

```
/**
 * Encrypts the message
 * @return The cipher text as a byte array
 */
public byte[] encrypt()
{
    for (int i = 0; i < c.length; i++)
    {
        c[i] = (byte) (m[i] ^ k[i]);
    }
    return c;
}
```

One Time Pad

Decryption

```
/**
 * Decrypts the cipher text byte array
 * @return The decrypted byte array
 */
public byte[] decrypt(byte[] ciphertext)
{
    byte[] decrypted = new byte[ciphertext.length];
    for (int i = 0; i < decrypted.length; i += 1)
    {
        decrypted[i] = (byte) (ciphertext[i] ^ k[i]);
    }
    return decrypted;
}
```

Presentation summary

- Character encoding: UTF-8 a superset of ASCII
- Caesar cipher: gained experience in simple character manipulation
- Vigenere cipher: A more complex character manipulation exercise
- One time pad: Of no practical use but provides insight into gist of encryption using **Exclusive OR** logical operator

Referenced Material

1. Character Sets And Code Pages At The Push Of A Button

<http://www.i18nguy.com/unicode/codepages.html>

[Accessed 2016-02-06]

2. Unicode 8.0 Character Code Charts

<http://www.unicode.org/charts>

[Accessed 2016-02-06]

3. Unicode & Character Sets

<http://www.joelonsoftware.com/articles/Unicode.html>

[Accessed 2016-02-06]

4. Vigenere Cipher

<http://www.cs.mtu.edu/~shene/NSF-4/Tutorial/VIG/Vig-Base.html>

[Accessed 2016-02-10]