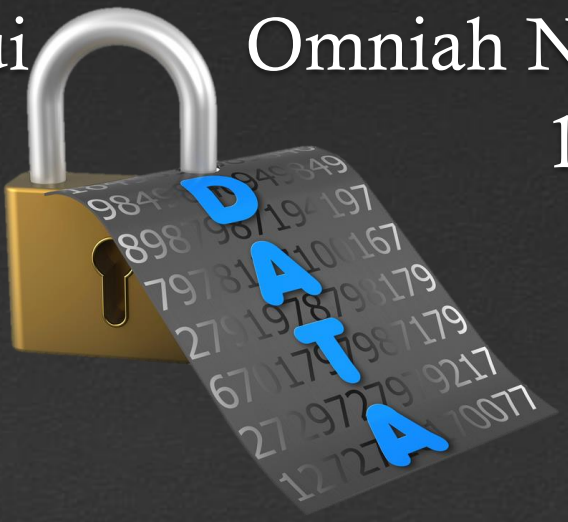
Cryptography in D

Shahzeb Siddiqui
124211

Omniah Nagoor
123923



Spring 2013 - KAUST

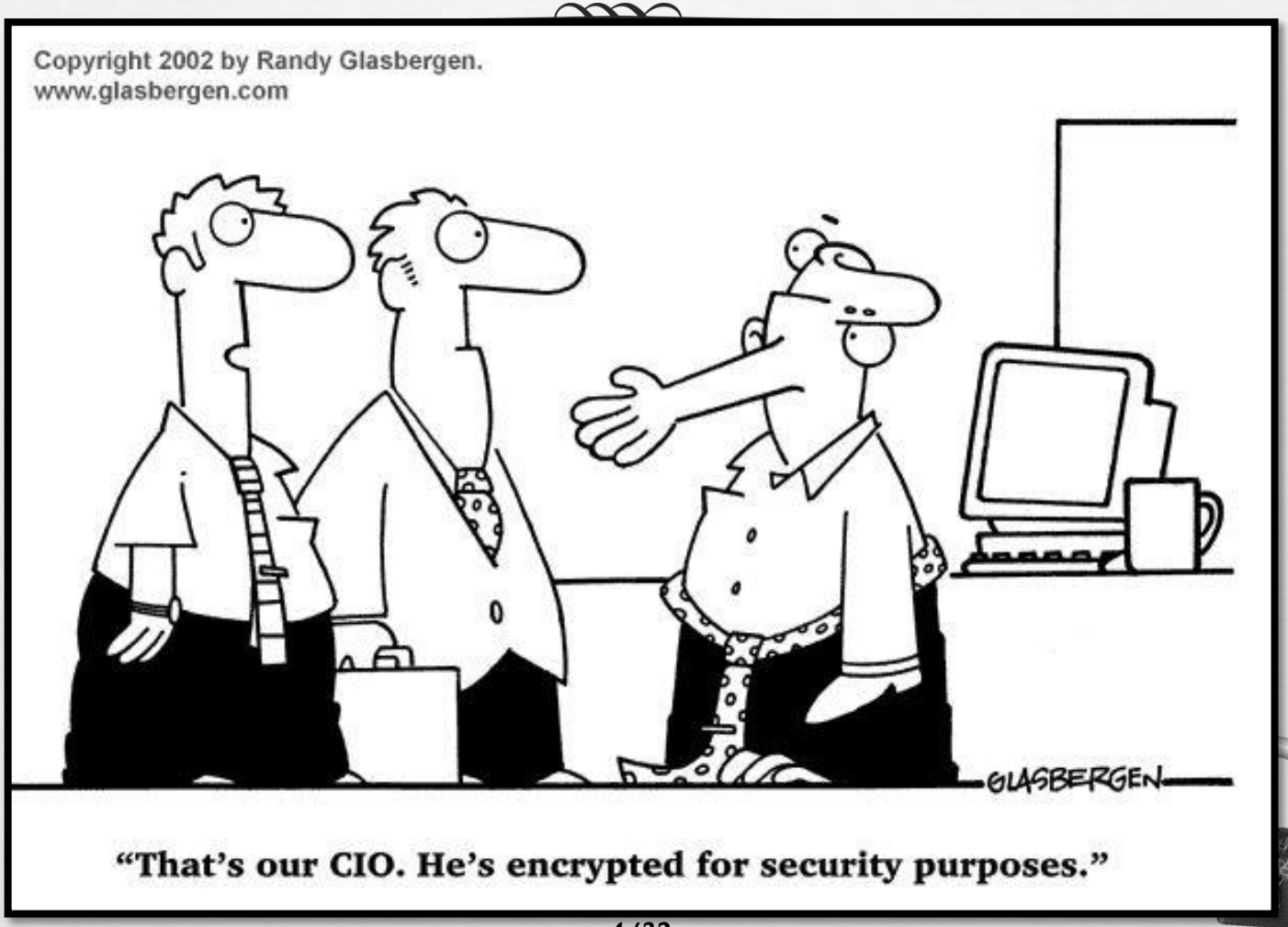
AGENDA

- Introduction to Cryptography
    1. Caesar Cipher
    2. Stream Cipher
    3. Block Cipher – Hill Cipher
- D Language Features
- Feature Implementation
- Language Features Comparison
- Demo
- Conclusion

INRODUCTION

- Cryptography is a technique for distorting the message to protect against others from reading it through the use of encryption
- The goal of cryptography is to ensure confidentiality, data integrity, and authentication
- Modern cryptography is based on mathematical theory and computational complexity

Security Purposes



Cryptography Terminology

- **Plaintext:** Original message
- **Cipher text:** The encoded message
- **Cipher:** Algorithm to transform plaintext to cipher text
- **Key:** Info to grant sender/receiver to recover message from cipher
- **Encrypt:** Method to transform plaintext to cipher text
- **Decrypt:** Method to transform cipher text to plaintext

Caesar Cipher 1/2

- Caesar Cipher was used by the Romans named after Julius Caesar to secretly communicate messages
- Originally each letter was substituted three positions where A -> D, B-> E, … and it wrapped around on the ends of alphabet.

Shift (caesar) cipher. (n.d.). Retrieved from http://cryptoclub.math.uic.edu/shiftcipher/shiftcipher.htm

Caesar Cipher 2/2

- Caesar cipher was used on rotating wheel where plaintext is on the outer ring, and cipher text is in the inner wheel.

Stream Cipher 1/2

- Stream Cipher is a symmetric key cipher where plaintext is XOR with a pseudo random key. The key should be at least the size of the plaintext to make it hard to break.
- In symmetric key cipher, only sender and receiver know the key which makes stream cipher secure against outside attack
- Stream cipher can be broken using brute force but that's not efficient nowadays as key lengths have increased to 128,192, or 256 bit keys.

Stream Cipher 2/2

Plaintext	1	1	0	1	0	0	0	1	1	0	1	0	1	0	0	1
Keystream	0	1	1	1	1	0	0	0	0	1	1	0	1	1	1	0
Ciphertext	1	0	1	0	1	0	0	1	1	1	0	0	0	1	1	1

(a) Encryption

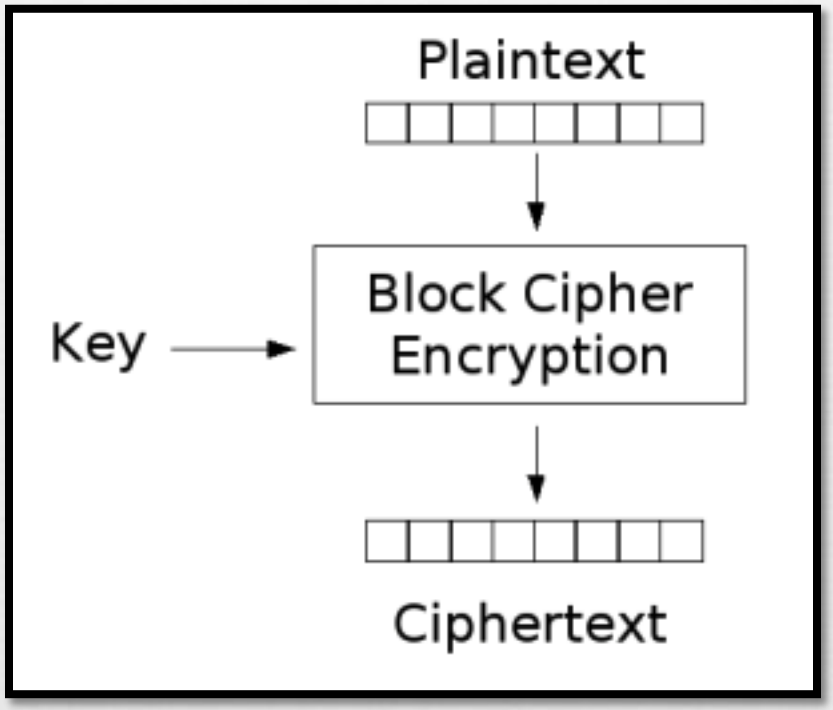
Ciphertext	1	0	1	0	1	0	0	1	1	1	0	0	0	1	1	1
Keystream	0	1	1	1	1	0	0	0	0	1	1	0	1	1	1	0
Plaintext	1	1	0	1	0	0	0	1	1	0	1	0	1	0	0	1

(b) Decryption using an identical keystream

Block Cipher 1/2

- Block ciphers encrypting a block of letters simultaneously.
- Many of the modern (symmetric) cryptosystems are block ciphers. DES operates on 64 bits of blocks while AES uses 128 bits of blocks(192 and 256 are also possible).

Block Cipher 2/2



Hill Cipher 1/2

- Hill Cipher is a polygraphic substitution cipher invented by Lester Hill in 1929
- The cipher utilizes concepts of linear algebra
- Each letter is represented by a number modulo 26

Hill Cipher 2/2

- To encrypt message, each block of n letters is selected as a n-component vector that is multiplied by a invertible n x n matrices modulo 26
- The n x n matrices contain the cipher key and it must be chosen randomly

Hill Cipher Example

$$C1 = 9*p1 + 18*p2 + 10*p3 \pmod{26}$$

$$C2 = 16*p1 + 21*p2 + 1*p3 \pmod{26}$$

$$C3 = 5*p1 + 12*p2 + 23*p3 \pmod{26}$$

$$\begin{pmatrix} C1 \\ C2 \\ C3 \end{pmatrix} = \begin{pmatrix} 9 & 18 & 10 \\ 16 & 21 & 1 \\ 5 & 12 & 23 \end{pmatrix} \begin{pmatrix} p1 \\ p2 \\ p3 \end{pmatrix} \pmod{26}$$

D Language Features 1/2

D Language Feature Comparison Table

Feature	D
Garbage Collection	Yes
Functions	
Function delegates	Yes
Function overloading	Yes
Out function parameters	Yes
Nested functions	Yes
Function literals	Yes
Closures	Yes
Typesafe variadic arguments	Yes
Lazy function argument evaluation	Yes
Compile time function evaluation	Yes
Arrays	
Lightweight arrays	Yes
Resizeable arrays	Yes
Built-in strings	Yes
Array slicing	Yes
Array bounds checking	Yes
Array literals	Yes
Associative arrays	Yes
String switches	Yes
Aliases	Yes
Generic Programming	
Class Templates	Yes
Function Templates	Yes
Implicit Function Template Instantiation	Yes
Partial and Explicit Specialization	Yes
Value Template Parameters	Yes
Template Template Parameters	Yes
Variadic Template Parameters	Yes
Template Constraints	Yes
Mixins	Yes
static if	Yes
is expressions	Yes
typeof	Yes
foreach	Yes
Implicit Type Inference	Yes
Reliability	
Contract Programming	Yes
Unit testing	Yes
Static construction order	Yes
Guaranteed initialization	Yes
RAII (automatic destructors)	Yes
Exception handling	Yes
Scope guards	Yes
try-catch-finally blocks	Yes
Thread synchronization primitives	Yes

D Language Features 2/2

OOP	
Object Oriented	Yes
Multiple Inheritance	No
Interfaces	Yes
Operator overloading	Yes
Modules	Yes
Dynamic class loading	No
Nested classes	Yes
Inner (adaptor) classes	Yes
Covariant return types	Yes
Properties	Yes
Performance	
Inline assembler	Yes
Direct access to hardware	Yes
Lightweight objects	Yes
Explicit memory allocation control	Yes
Independent of VM	Yes
Direct native code gen	Yes

Compatibility	
C-style syntax	Yes
Enumerated types	Yes
Support all C types	Yes
80 bit floating point	Yes
Complex and Imaginary	Yes
Direct access to C	Yes
Use existing debuggers	Yes
Struct member alignment control	Yes
Generates standard object files	Yes
Macro text preprocessor	No
Other	
Conditional compilation	Yes
Unicode source text	Yes
Documentation comments	Yes

Feature Implementation

Associative Array

- Associative array have an index that is not necessarily an integer. It can be other type such as string.

```
ulong [string] keylog;

int [string] table;
```

Dynamic Array

- Dynamic Array have variable number of elements at run time.

```
int [] key;
int [] src;
int [] sum;
```

Variadic Function

- Variadic Function is a function that takes variable number of arguments.

```
void variable_keys_ceaser_cipher(string plaintext, int[] numkeys ...)
```

Array Slicing

- Array slicing specifies subarray and create new pointer reference to it.

```
char[] newtable = table[key .. charCount] ~ table[0 .. key];
```

Array Operations

- **.length**: returns the number of values in the array.

```
sum[i] = sum[i] % s.length;
```

- **.dup:** create a dynamic array of the same size and copy the contents of the array into it.

```
char EncMssg[] = charMessage.dup;
```

- **.reverse :** reverses in place the order of the elements in the array and returns the array.

```
binaryreverse = binary.reverse;
```

Nested Function

- Function inside another function

```
void variable_keys_ceaser_cipher(string plaintext, int[] numkeys ...)
{
   int i, index;

   char table[charCount] = "abcdefghijklmnopqrstuvxyz ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@#$%^&*(),./;'[]{}:";

   char [][charCount] newtable;
   char [] plaintext_inchar = plaintext.dup;
   char [] ciphertext;

   void ceasercipher(char letter, int key, int keyindex)
   {
       int j;

       for (j = 0; j < table.length; j++)
       {
           if (letter == table[j])
               ciphertext ~= newtable[keyindex][j];
       }

       writeln("letter = ",letter,"\tciphertext = ",ciphertext, "\t key = ", key);
   }
```

Contract Programming 1/2

- **Assert(expr):** evaluates expression expr if it is nonzero there no effect otherwise assert throws an exception of type AssertError.
- **Contract Programming :** preconditioner check for the Expression and if it is true then execute the body.

Contract Programming 2/2

```
ulong gen_key(string keyname)
{
  ulong check_key_if_valid (ulong key)
    in { assert (key >= 0 && key <= long_max);}
    body
    {
      bool keymatch = false;

      if (keylog.length == 0)
          keylog[keyname] = key;
      else
      {
          do
          {
              for (int i = 0; i < numkeys_assign - 1; i++)
              {
                  if (keylog[person_with_key[i]] == key)
                      keymatch = true;
              }

              if (keymatch == true)
              {
                  key = uniform (0,long_max);
              }

          }
          while (keymatch == true);
          keylog[keyname] = key;
      }
      return key;
    }
```

Other Functions 1/2

- **to!string():** converts a value from type Source to type target.
- **readln():** read line from stream fp. It return a string data type.
- **write():** prints the given arg to the standard output. A call without any arguments will fail to compile.
- **writeln():** Similar to write function but with a newline. Calling writeln without arguments is valid and just prints a newline to the standard output.

Other Functions 2/2

- **Foreach:** iterator that start by index 0 to the end of the array

```
foreach(i; 0 .. s.length)
    table[to!string(s[i])] = i;
```

- ~: This binary operator is used to concatenate array elements

```
ciphertext ~= newtable[keyindex][j];
```

- **uniform(min,max):** give a random number between [min,max]

```
key ~= uniform(0,s.length);
```

Language Features Comparison

Features	C	C++	D	Java
OO	No	Yes	Yes	Yes
Multiple Inheritance	No	Yes	No	No
Operator overloading	No	Yes	Yes	No
Garbage collection	No	No	Yes	Yes
Standard documentation mechanism	No	No	Yes	Yes
Function pointers	Yes	Yes	Yes	No
Interfaces	No	Yes	Yes	Yes
Contract Programming	No	Yes	Yes	Yes
Nested Functions	No	No	Yes	No

DEMO

Conclusion

- Our project is implementation of three different encryption algorithms which are:
    1. Caesar Cipher
    2. Stream Cipher
    3. Block Cipher – Hill Cipher
- D features and functions gives the programmers flexibility to implement programs in easy manner.
- C/C++ and Java programmer have short learn curve of D language.

References

- *D programming language*. (n.d.). Retrieved from http://dlang.org/
- Andrei Alexandrescu, “The D Programming Language”.
- *Shift (caesar) cipher*. (n.d.). Retrieved from http://cryptoclub.math.uic.edu/shiftcipher/shiftcipher.htm

Thank You!

Any Questions?