

CSC3621 Cryptography - Exercise 3

Aim - To understand the working of a One Time Pad and practice a Two-Time Pad

Introduction:

A One Time Pad is unbreakable when used correctly. A One Time Pad is a completely random key which is used to encrypt a message of the same length.

The One Time Pad is perfectly secure, however it has practical issues.

The technique uses modular arithmetic to encrypt the bits of the message with the key.

The bits are XOR'd then the corresponding bits make the cipher text.

Implementation:

To implement such an algorithm, I first had to understand how the arithmetic worked. I then had to understand how the system reads characters and use Java to manage the process.

Encrypt

My encryption works by taking in a plain text String in ASCII form and a byte array created by placing the hexadecimal key through the `hexToByteArray()` method taken from Feng Hao's `OTPAAttack.java` file.

The encryption takes the plain text and converts it to bytes and stores in a byte array. The method then completes a check to ensure that the message length is the same as the key. If it is not then it returns an error.

```
public String encrypt(String plaintext, byte[] key) {
    /*
     * Converts plain text to bytes using getBytes() and takes key as string and converts to byte array
     * (using hexStringToByteArray) function and Xor's the bytes, then rebuilds the string using
     * bytesToHex function.
     */
    final byte[] plain = plaintext.getBytes();
    final byte[] keyByte = key;

    if (plain.length > keyByte.length) {
        System.err.println("Message length and key length not equal");
        System.exit(0);
    }

    final byte[] encrypted = new byte[plain.length];
    for (int i = 0; i < plain.length; i++) {
        encrypted[i] = (byte) ((plain[i] ^ keyByte[i]));
    }
    String e = bytesToHex(encrypted);
    return e;
}
```

A new array is created if the lengths are equal and the plain text/key is iterated through and each byte is XOR'd. The bytes are then converted to a hexadecimal form using Feng Hao's `bytesToHex()` function taken from the `OTPAAttack.java` file. It is then stored as a String and returned.

```

public String decrypt(String cText, byte[] key) {
    /*
     * Converts cipher text to bytes and takes in key bytes (using hexStringToByteArray) function
     * and Xor's the bytes, then rebuilds string of decrypted text
     */

    final byte[] cipher = hexStringToByteArray(cText);
    final byte[] keyByte = key;
    final byte[] decrypted = new byte[cipher.length];
    if (cipher.length > keyByte.length) {
        System.err.println("Message length and key length not equal");
        System.exit(0);
    }

    for (int i = 0; i < cipher.length; i++) {
        decrypted[i] = (byte) ((cipher[i] ^ keyByte[i]));
    }

    String e = new String(decrypted);

    return e;
}

```

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec, pellentesque eu, pretium quis, sem. Nulla consequat massa quis enim. Donec pede justo, fringilla vel, aliquet nec, vulputate eget, arcu. In enim justo, rhoncus ut, imperdiet a, venenatis vitae, justo. Nullam dictum felis eu pede mollis pretium. Integer tincidunt. Cras dapibus. Vivamus elementum semper nisi. Aenean vulputate eleifend tellus. Aenean leo ligula, porttitor eu, consequat vitae, eleifend ac, enim. Aliquam lorem ante, dapibus in, viverra quis, feugiat a, tellus. Phasellus viverra nulla ut metus varius laoreet. Quisque rutrum. Aenean imperdiet. Etiam ultricies nisi vel augue. Curabitur ullamcorper ultricies nisi. Nam eget dui. Etiam rhoncus. Maecenas tempus, tellus eget condimentum rhoncus, sem quam semper libero, sit amet adipiscing sem neque sed ipsum. Nam quam nunc, blandit vel, luctus pulvinar, hendrerit id, lorem. Maecenas nec odio et ante tincidunt tempus. Donec vitae sapien ut libero venenatis faucibus. Nullam quis ante. Etiam sit amet orci eget eros faucibus tincidunt. Duis leo. Sed fringilla mauris sit amet nibh. Donec sodales sagittis magna. Sed consequat, leo eget bibendum sodales, augue velit cursus nunc, quis gravida magna mi a libero. Fusce vulputate eleifend sapien. Vestibulum purus quam, scelerisque ut, mollis sed, nonummy id, metus. Nullam accumsan lorem in dui. Cras ultricies mi eu turpis hendrerit fringilla. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; In ac dui quis mi consectetur lacinia. Nam pretium turpis et arcu. Duis arcu tortor, suscipit eget, imperdiet nec, imperdiet iaculis, ipsum. Sed aliquam ultrices mauris. Integer ante arcu, accumsan a, consectetur eget, posuere ut, mauris. Praesent adipiscing. Phasellus ullamcorper ipsum rutrum nunc. Nunc nonummy metus. Vestibulum volutpat pretium libero. Cras id dui. Aenean ut eros et nisl sagittis vestibulum. Nullam nulla eros, ultricies sit amet, nonummy id, imperdiet feugiat, pede. Sed lectus. Donec mollis hendrerit risus. Phasellus nec sem in justo pellentesque facilisis. Etiam imperdiet imperdiet orci. Nunc nec neque. Phasellus leo dolor, tempus non, auctor et, hendrerit quis, nisi. Curabitur ligula sapien, tincidunt non, euismod vitae, posuere imperdiet, leo. Maecenas malesuada. Praesent congue erat at massa. Sed cursus turpis vitae tortor. Donec posuere vulputate arcu. Phasellus accumsan cursus velit. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Sed aliquam, nisi quis porttitor congue, elit erat euismod orci, ac placerat dolor lectus quis orci. Phasellus consectetur vestibulum elit. Aenean tellus metus, bibendum sed, posuere ac, mattis non, nunc. Vestibulum fringilla pede sit amet augue. In turpis. Pellentesque posuere. Praesent turpis. Aenean posuere, tortor sed cursus feugiat, nunc augue blandit nunc, eu sollicitudin urna dolor sagittis lacus. Donec elit libero, sodales nec, volutpat a, suscipit non, turpis. Nullam sagittis. Suspendisse pulvinar, augue ac venenatis condimentum, sem libero volutpat nibh, nec pellentesque velit pede quis nunc. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Fusce id purus. Ut varius tincidunt libero. Phasellus dolor. Maecenas vestibulum mollis diam. Pellentesque ut neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. In dui magna, posuere eget, vestibulum et, tempor auctor, justo. In ac felis quis tortor malesuada pretium. Pellentesque auctor neque nec urna. Proin sapien ipsum, porta a, auctor quis, euismod ut, mi. Aenean viverra rhoncus pede. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Ut non enim eleifend felis pretium feugiat. Vivamus quis mi. Phasellus a est. Phasellus magna. In hac habitasse platea dictumst. Curabitur at lacus ac velit ornare lobortis. Curabitur a felis in nunc fringilla tristique. Morbi mattis ullamcorper velit. Phasellus gravida semper nisi. Nullam vel sem. Pellentesque libero tortor, tincidunt et, tincidunt eget, semper nec, quam. Sed hendrerit. Morbi ac felis. Nunc egestas, augue at pellentesque laoreet, felis eros vehicula leo, at malesuada velit leo quis pede. Donec interdum, metus et hendrerit aliquet, dolor diam sagittis ligula, eget egestas libero turpis vel mi. Nunc nulla. Fusce risus nisi, viverra et, tempor et, pretium in, sapien. Donec venenatis vulputate lorem. Morbi nec metus. Phasellus blandit leo ut odio. Maecenas ullamcorper, dui et placerat feugiat, eros pede varius nisi, condimentum viverra felis nunc et lorem. Sed magna purus, fermentum eu, tincidunt eu, varius ut, felis. In auctor lobortis lacus. Quisque libero metus, condimentum nec, tempor a, commodo mollis, magna. Vestibulum ullamcorper mauris at ligula. Fusce fermentum. Nullam cursus lacinia erat. Praesent blandit laoreet nibh. Fusce convallis metus id felis luctus adipiscing. Pellentesque egestas, neque sit amet convallis pulvinar, justo nulla eleifend augue, ac auctor orci leo non est. Quisque id mi. Ut tincidunt tincidunt erat. Etiam feugiat lorem non metus. Vestibulum dapibus nunc ac augue. Curabitur vestibulum aliquam leo. Praesent egestas neque eu enim. In hac habitasse platea dictumst. Fusce a quam. Etiam ut purus mattis mauris sodales aliquam. Curabitur nisi. Quisque malesuada placerat nisl. Nam ipsum risus, rutrum vitae, vestibulum eu, molestie vel, lacus. Sed augue ipsum, egestas nec, vestibulum et, malesuada adipiscing, dui. Vestibulum facilisis, purus nec pulvinar iaculis, ligula mi congue nunc, vitae euismod ligula urna in dolor. Mauris sollicitudin fermentum libero. Praesent nonummy mi in odio. Nunc interdum lacus sit amet orci. Vestibulum rutrum, mi nec elementum vehicula, eros quam gravida nisl, id fringilla neque ante vel mi. Morbi mollis tellus ac sapien. Phasellus volutpat, metus eget aenean mollis. lacus lacus blandit dui. id aenean quam mauris ut lacus. Fusce vel dui. Sed in libero ut nibh placerat arcuam. Proin faucibus arcu nunc ante. In consectetur turpis ut velit. Nulla elit

The distribution unlike the previous two algorithms has very little affect as a frequency is not really important.

Tests

As part of the analysis I wanted to test another piece of text. I used the generator to create a one time pad to be the same length as “Supercalifragilisticexpialidocious”

The One time pad is

“360a4129135a53415310777c642d46ea64377a1f464c7402027f5a34564c6a636e44”

Using this one time pad I was able to encrypt the Plain text above and outputted the following.

This result is the encryption and decryption of the plain text.

```
360a4129135a53415310777c642d46ea64377a1f464c7402027f5a34564c6a636e44
Supercalifragilisticexpialidocious
```

I also tested other plain text and cipher texts into the functions as outputted below:

```
Encryption of plainText: 28b14ab7ecc33ea157b539ea426c5e9def0d81627eed498809c17ef9404cc5
Encryption of plainText3: 360a4129135a53415310777c642d46ea64377a1f464c7402027f5a34564c6a636e44
Decryption of cipherText3: Supercalifragilisticexpialidocious
Decryption of cipherText: Every cloud has a silver lining
```

XOR Spaces

As part of the analysis is was important to test the effects of XORing a space with a character.

I wrote a function to test this and then tried with different cases.

I found that XORing a space with a character of lowercase results in the character in uppercase and vice versa.

The affect is reversible.

XORing spaces tests:

A
a
B
b

```
System.out.println("XORing spaces tests: ");  
otp.xorSpaceTest("a");  
otp.xorSpaceTest("A");  
otp.xorSpaceTest("b");  
otp.xorSpaceTest("B");  
System.out.println("\n");
```

These are the results from XORing the characters with a space. This shows that the character case is inverted when XORed with the space.

Two time Pad

Using this knowledge I was able to run the OTPAttack.java and from the result I could see where spaces were in the resulting output because of the common similarities. This meant that I could deduce where some letter were. This has been achieved by XORing each cipher text.

“ - o u - h a v - - d - n - - y - u r - w - - k

The text shown above is what I deduced and from that I can see that the message is most likely “ you have done your work”. This is because where the spaces are assumed the characters are all uppercase and are similar at various points. Therefore XORing the space would make them lower case of the same character leaving us with the message above.

Conclusion

I found this task extremely challenging to program as I am still quite an inexperienced programmer. Because of this I chose to manually do the two time pad attack using Feng Hao's OTPAttack.java. This allowed me to see the result and piece together manually the result. I understand this is not programmed however I understand the theory behind the algorithm and understand that it is not possible to retrieve the whole message as there are some spaces where there are too many possibilities which would need to be brute force attacked and on a large scale this would be extremely difficult to complete if not “impossible”.

References

OTPAAttack.java Feng Hao