Class: Cybersecurity Technologies - Fall 2021 CYSM 3000 SEC501

Date: Oct. 9, 2021

Author: Blake Lawall

Discussion Question 3:

Suppose that someone suggests the following way to confirm that the two of you are both in possession of the same secret key. You create a random bit string the length of the key, XOR it with the key, and send the result over the channel. Your partner XORs the incoming block with the key (which should be the same as your key) and sends it back. You check, and if what you receive is your original random string, you have verified that your partner has the same secret key, yet neither of you has ever transmitted the key. Is there a flaw in this scheme?

Answer:

Yes, there is a flaw in this scheme! Here is a senerio that will show how. Here I will be sending a secret random message to Yolanda. Dr. Hicks in his patient and creative way will sniff the encrypted message that I send Yolanda and the unencrypted message which Yolanda sends back to me.

In [1]:
```
# imported libraries
import string
import random
```

In [2]:
```
# List of characters that are used in the random message including some control chars \n, \r, \t,
print(string.printable)
```

```
0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!"#$%&'()**+,-./:;<=>?@[\]^_`{|}~
```

In [3]:
```
key = "Too many secrets" # Sneakers ref (Robinson, 1992)
# Random message created as same length as the key
message_to_be_sent = random.sample(string.printable, len(key))
```

In [4]:
```
# Converting from a list to a string
message_to_be_sent_tostring = "".join(message_to_be_sent)
print(message_to_be_sent_tostring)
```

```
f_M6*E{XWRTSuC5
```

In [5]:
```
# Converting secret key chars to numbers
ascii_key = list(map(ord, key))
```

In [6]:
```
# Converting message chars to numbers
ascii_message_to_be_sent = list(map(ord, message_to_be_sent_tostring))
```

In [7]:
```
# Making the key and message into pairs to be able to XOR them
pairs = list(zip(ascii_key, ascii_message_to_be_sent))
```

In [8]:
```
# Function that is XORing the secret key and random message together for encryption
def xor_pair(items):
    return items[0] ^ items[1] # using builtin XOR operator (BitwiseOperators - Python Wiki, n.d
```

In [9]:
```
# Calling the function to encrypt the random message
encrypted_message = list(map(xor_pair, zip(ascii_key, ascii_message_to_be_sent)))
```

In [10]:
```
# Showing the number values of the key, message, and the encrypted message
print(ascii_key)
print(ascii_message_to_be_sent)
print(encrypted_message)
```

```
[84, 111, 111, 32, 109, 97, 110, 121, 32, 115, 101, 99, 114, 101, 116, 115]
[102, 95, 77, 54, 42, 69, 11, 123, 88, 87, 82, 84, 83, 117, 67, 53]
[50, 48, 34, 22, 71, 36, 101, 2, 120, 36, 55, 55, 33, 16, 55, 70]
```

In [11]:
```
84 ^ ascii_message_to_be_sent[0] # example 1 showing that the XORing works
```

Out[11]:
```
50
```

In [12]:
```
111 ^ ascii_message_to_be_sent[1]  # example 2 showing that the XORing works
```

Out[12]:
```
48
```

In [13]:
```
111 ^ ascii_message_to_be_sent[2]  # example 3 showing that the XORing works
```

Out[13]:
```
34
```

In [14]:
```
# Converting the secert message back to characters to be sent to Yolanda
encrypted_message_sent = "".join(map(chr, encrypted_message))
print(encrypted_message_sent) # Showing the encrypted message
```

```
20"G$ex$77!7F
```

Dr. Hicks-in-the-middle attack sniffs and intercepts the encrypted_message_sent and waits.

Yolanda receives the encrypted message and proceeds to decrypt with her secert key which should be the same as mine.

In [15]:
```
# Yolanda converting the encrypted message to numbers
ascii_rcvd_msg = list(map(ord, encrypted_message_sent))
print(ascii_rcvd_msg)
```

```
[50, 48, 34, 22, 71, 36, 101, 2, 120, 36, 55, 55, 33, 16, 55, 70]
```

In [16]:
```
# Yolanda creating the pairs list to run the XOR on
yolanda_pairs = list(zip(ascii_key, ascii_rcvd_msg))
print(yolanda_pairs)
```

```
[(84, 50), (111, 48), (111, 34), (32, 22), (109, 71), (97, 36), (110, 101), (121, 2), (32, 120),
(115, 36), (101, 55), (99, 55), (114, 33), (101, 16), (116, 55), (115, 70)]
```

In [17]:
```
# Yolanda running her secret key to decrypt the message
unencrypted_message = list(map(xor_pair, yolanda_pairs))
```

In [18]:
```
# the numerical values of the enencrypted message
print(unencrypted_message)
```

```
[102, 95, 77, 54, 42, 69, 11, 123, 88, 87, 82, 84, 83, 117, 67, 53]
```

In [19]:
```
# Taking the message back to characters
unencrypted_message_sent = "".join(map(chr, unencrypted_message))
print(unencrypted_message_sent)
```

```
f_M6*E{XWRTSuC5
```

Dr. Hicks-in-the-middle attack sniffs and intercepts the unencrypted_message_sent and reverse XOR and get the key.

In [20]:
```
# Dr. Hicks inspecting the encrypted message and the unencryped message
print(encrypted_message_sent)
print(unencrypted_message_sent)
```

```
20"G$ex$77!7F
f_M6*E{XWRTSuC5
```

In [21]:
```
# Dr. Hicks converting the messages to numbers
ascii_encrypted_message_sent = list(map(ord, encrypted_message_sent))
ascii_unencrypted_message_sent = list(map(ord, unencrypted_message_sent))
```

In [22]:
```
# Dr. Hicks looking at those numbers
print(ascii_encrypted_message_sent)
print(ascii_unencrypted_message_sent)
```

```
[50, 48, 34, 22, 71, 36, 101, 2, 120, 36, 55, 55, 33, 16, 55, 70]
[102, 95, 77, 54, 42, 69, 11, 123, 88, 87, 82, 84, 83, 117, 67, 53]
```

In [23]:
```
# Dr. Hicks creating a pair list to run XOR on to see if he can get the secert key
drhicks_pairs = list(zip(ascii_encrypted_message_sent, ascii_unencrypted_message_sent))
```

In [24]:
```
# Dr. Hicks running the XOR
drhicks_gets_key = list(map(xor_pair, drhicks_pairs))
```

In [25]:
```
# Dr. Hicks looking at the numbers he got from XORing
print(drhicks_gets_key)
```

```
[84, 111, 111, 32, 109, 97, 110, 121, 32, 115, 101, 99, 114, 101, 116, 115]
```

In [26]:
```
# Dr. Hicks changing the numbers to characters to get the key
stollen_key = "".join(map(chr, drhicks_gets_key))
```

In [27]:
```
# Dr. Hicks printing the secret key
print(stollen_key)
```

```
Too many secrets
```

Now Dr. Hicks knows the secret key.

References

BitwiseOperators - Python Wiki. (n.d.). Python. Retrieved October 9, 2021, from https://wiki.python.org/moin/BitwiseOperators

Robinson, P. A. (Director). (1992). Sneakers [Film]. Universal Pictures.

Symmetric Key Cryptography: The XOR Cipher. (2019, March 3). YouTube. https://www.youtube.com/watch?v=pvIl6_O6KAc

In [ ]:
```
```