EPFL

SPRING

Computer Security (COM-301)

Applied cryptography I

Interactive Exercises

Carmela Troncoso
SPRING Lab
carmela.troncoso@epfl.ch

OTP is the best?

Agree or disagree and justify :

"A One Time Pad is the best choice to transmit a secret document of 10Mb because we know it provides perfect secrecy"

2

Reasons why OTP is **not** the best choice:

- You need a key *as long as the message*. When the message is 10Mb, this is not so easy to do random without modem crypto
- You need the means to transmit such a long key to the receiver. This must be done on a channel different than the one where the message is sent (otherwise the adversary can access the key!)

One could say that for the application at hand secrecy is so important that OTP must be used, but then you have to solve the two problems above (i.e., explicitly say how the key would be created and transmitted)

Secure streaming

Is the following pseudocode for a function to encrypt movies a secure use of a stream cipher? Justify your answer.

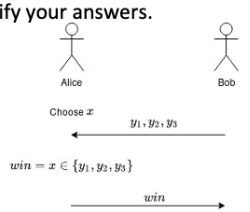
```
var iv = "bestidever"

def EncryptMovie(movie):
      key = GenerateSecureRandomKey() // generates a truly random key
      secureKeyTransfer(key)  // securely sends the key to the
                              // recipient
      EncryptedMovie = Salsa20(key,iv,movie)    // uses Salsa20 stream
                                                // cipher
      return(EncryptedMovie)
```

Yes, it is secure because, even if the IV is constant, a new random key is generated for encryption in every loop -> combination (key, iv) is fresh -> there will be no two messages encrypted with the same pair.

Guess my number

Alice and Bob have decided to play a game called "Guess my number". In this game, Alice chooses one number between 0 and 100. Then, Bob has 3 chances to guess the number. Bob wins if he guesses the number correctly. How can Alice cheat? Bob thinks that using a Message Authentication Code (MAC) can prevent Alice from cheating. Can you integrate a MAC into this game in a way that allows Bob to publicly show that Alice has cheated? What if Alice and Bob have a trusted friend called Charlie and Bob only cares about Charlie's opinion? Justify your answers.



Alice can cheat lying about her number and always deny that Bob has guessed correctly.

Introducing a MAC cannot prevent cheating since Bob cannot prove publicly who is the author of the message. Thus, Alice can deny having sent the MAC, and still cheat. Also, given that there are only 100 numbers, Bob could compute the MAC of all numbers and learn Alice's choice.

Having a trusted third party (Charlie) would solve the problem. If Alice sends to Charlie her choice and the MAC, and Bob sends to Charlie his guesses. Charlie can say if Bob has won, and there is no doubt because Charlie knows the origin of the messages.

Encrypt full speed

You are designing a high-speed encrypted link between two buildings at your company. What symmetric encryption scheme would you use if

- There is only one core in the receiver and transmitter

- There are several cores in the receiver and the transmitter

1 Core: stream cipher would be a nice solution. Fast and compact. You don't gain anything from other ciphers as encryption and decryption have to be sequential anyway.

N cores: We can take advantage of parallelization, e.g., using a block cipher in CTR, or CBC (only allows for parallel decryption).

OTP or AES

Alice knows that she will want to send a single 128-bit message to Bob at some point in the future. To prepare, Alice and Bob first select a 128-bit key $k \in \{0, 1\}^{128}$ uniformly at random.

When the time comes to send a message $x \in \{0, 1\}^{128}$ to Bob, Alice considers two ways of doing so:

1) She can use the key as a one time pad, sending Bob $k \oplus x$.

1) She can use AES to encrypt x. Recall that AES is a 128-bit block cipher which can use a 128-bit key, so in this case she would encrypt x as a single block and send Bob $AES_k(x)$.

Assume the adversary Eve will see either $k \oplus x$ or $AES_k(x)$, that Eve knows an initial portion of x (a standard header), and that she wishes to recover the remaining portion of x. If Eve is an all powerful adversary and has time to try out every possible key $k \in \{0, 1\}^{128}$, which scheme would be more secure?

They would be equally secure. Either way, Eve would not be able to learn the unknown portion of x.

Even after trying every possible key (including the actual one), Eve will have no way of recognizing the correct plaintext or even narrowing down the possibilities in any way. Why is this? Well, since AES is a permutation of 128 bits under each possible key, and the key was selected uniformly at random, given any plaintext, each possible ciphertext is equally likely. So when AES is used for a single block with a random key of the same length, the effect is exactly the same as using a one time pad: the ciphertext reveals no information about the plaintext.