UOSEC Week 1

Symmetric Cryptography

Presenters

- Adam pond2@uoregon.edu
  - IRC nick: xe0
- Frank farana@uoregon.edu
  - IRC nick: kee
- irc.freenode.net #0x4f
Agenda

• All Oregon CTF March 7th PSU
• Saturday meet up
• Welcome CIS 433 Students
• HackIM CTF ctf.nullcon.net Jan 9-11
• GITS CTF ghostintheshellcode.com Jan 16-18
• Upcoming CTFS listed on ctftime.org
Crypto Defined

What is cryptography?
Crypto Defined

What is cryptography?
Cryptography is the art and science of encryption.
Crypto Defined

What is cryptography?

Cryptography is the art and science of encryption.

What is encryption?
Crypto Defined

What is cryptography?
Cryptography is the art and science of encryption.

What is encryption?
Encryption is a way of encoding messages or information so that only authorized users can read it.
Crypto Defined

What is cryptography?
Cryptography is the art and science of encryption.

What is encryption?
Encryption is a way of encoding messages or information so that only authorized users can read it.

What is decryption?
Crypto Defined

What is cryptography?
Cryptography is the art and science of encryption.

What is encryption?
Encryption is a way of encoding messages or information so that only authorized users can read it.

What is decryption?
Decryption is the reverse of encryption. It is a way of decoding encrypted messages or information so that authorized users may read them.
Simple Example

Bob wants to send Alice a message containing his social security number (Bob is dumb).
Simple Example

Bob wants to send Alice a message containing his social security number (Bob is dumb).

Bob 111-222-333  Alice
Simple Example

Bob wants to send Alice a message containing his social security number (Bob is dumb).

Bob               111-222-333               Alice
Simple Example

Bob wants to send Alice a message containing his social security number (Bob is dumb).

Unfortunately for Bob, Eve was eavesdropping on Alice and Bobs “conversation”.

Bob  111-222-333  Alice
     |     |     
     Eve
Simple Example

Bob wants to send Alice a message containing his social security number (Bob is dumb).

Bob

111-222-333

Alice

111-222-333

Eve

111-222-333

Unfortunately for Bob, Eve was eavesdropping on Alice and Bobs “conversation”.
Simple Example

So now Eve goes on a shopping spree using Bobs information and some identity theft.
Simple Example 2

Bob wants to send Alice a message containing his social security number (Bob is dumb). This time Bob encrypts his message.

$$E(111-222-333) = C$$

Bob 111-222-333 Alice
Simple Example 2

Bob wants to send Alice a message containing his social security number (Bob is dumb). This time Bob encrypts his message.

\[ E(111-222-333) = C \]

Bob \hspace{1cm} c \hspace{1cm} Alice
Simple Example 2

Bob wants to send Alice a message containing his social security number (Bob is dumb). This time Bob encrypts his message.

\[ E(111-222-333) = C \]

Bob \hspace{2cm} C \hspace{2cm} Alice
Simple Example 2

Bob wants to send Alice a message containing his social security number (Bob is dumb). This time Bob encrypts his message.

\[ E(111-222-333) = C \]

Eve eavesdrops on the conversation

Eve
Simple Example 2

Bob wants to send Alice a message containing his social security number (Bob is dumb). This time Bob encrypts his message.

$$E(111-222-333) = C$$

Bob

Eve eavesdrops on the conversation and this time instead of receiving Bobs social security number she is only given ciphertext.
Simple Example 2

Bob wants to send Alice a message containing his social security number (Bob is dumb). This time Bob encrypts his message.

\[
E(111-222-333) = C \quad D(C) = M \text{ (message)}
\]

Bob

\[
Eve
\]

\[
111-222-333\quad Alice
\]

\[
Eve
\]

\[
c
\]

In a perfect world now Eve has only ciphertext that she cannot leverage to gain any additional information on Bobs message.

Alice decrypts Bobs encrypted message and receives the original message.
Encryption

\[ E_K(M) = C \]

E = encryption function
M = plaintext (non-encrypted message)
C = ciphertext (encrypted message)
K = key (pseudo-randomly generated usually)
Decryption

\[ D_K(C) = M \]

D = decryption function

M = plaintext (non-encrypted message)

C = ciphertext (encrypted message)

K = key (pseudo-randomly generated usually)
Kerckhoff’s Principle

• Never trust a cryptosystem where the algorithm is kept secret.
Kerckhoff’s Principle

- Never trust a cryptosystem where the algorithm is kept secret.

- “Faut qu'il n'exige pas le secret, et qu'il puisse sans inconvenient tomber entre les mains de l'ennemi.”
Kerckhoff’s Principle

• Never trust a cryptosystem where the algorithm is kept secret.

• “Faut qu'il n'exige pas le secret, et qu'il puisse sans inconvenient tomber entre les mains de l’ennemi.”

• “the cipher must not depend on the secrecy of the mechanism, it must not matter if it falls into the hands of the enemy”
Word of Warning

• Cryptography is really cool but you should never try to write your own crypto (unless you’re name is Leonard Adleman).

• There are many reasons why you shouldn’t roll your own crypto but these reasons require a high knowledge of number theory, physics, and electrical engineering to understand.

• Instead, here is what Phil Zimmerman the creator of PGP(pretty good privacy) has to say on the issue
Word of Warning

Also, I recognize this is an appeal to authority fallacy
Word of Warning

“When I was in college in the early 70s, I devised what I believed was a brilliant encryption scheme. Years later, I discovered this same scheme in several introductory cryptography texts and tutorial papers. Unfortunately, the scheme was presented as a simple homework assignment on how to use elementary cryptanalytic techniques to trivially crack it. So much for my brilliant scheme.”

-Phil Zimmerman
Perfect Cipher

• With that said, it is surprising to find that there exists a perfect cipher called a one time pad or OTP.
Perfect Cipher

• With that said, it is surprising to find that there exists a perfect cipher called a one time pad or OTP.

• We say a cipher is perfect if and only if it reveals no information about the key or message.
Perfect Cipher

• With that said, it is surprising to find that there exists a perfect cipher called a one time pad or OTP.

• We say a cipher is perfect if and only if it reveals no information about the key or message.

• Let's examine the OTP
OTP

• The OTP is a modular arithmetic cipher.

  o The general form is
    • \( E_K(M) = (K+M) \mod L \)
      
    = C

E = encryption function
K = random one time use key
M = message to encrypt
L = value to mod. varies based on language.
C = ciphertext
OTP Simple Example

Bob encrypts a secret message and sends it to Alice.

Bob HELLO Alice
OTP Simple Example

Bob encrypts a secret message and sends it to Alice.

Bob     HELLO     Alice

We will assign a numeric value to each letter (although you could simply xor the binary values).
OTP Simple Example

Bob encrypts a secret message and sends it to Alice. He uses the secret “random” key XMCKL.

\[ E_K(M) = C \]

Bob | HELLO | Alice
OTP Simple Example

Bob encrypts a secret message and sends it to Alice. He uses the secret “random” key XMCKL.

\[ E_K(M) = C \]

Bob  EQNVZ  Alice
OTP Simple Example

Bob encrypts a secret message and sends it to Alice. He uses the secret “random” key XMCKL.

\[ E_K(M) = C \]
OTP Simple Example

Bob encrypts a secret message and sends it to Alice. He uses the secret “random” key XMCKL.

\[ E_K(M) = C \]

Eve intercepts the message but has no way of learning anything from the ciphertext.
OTP Summary

- Unfortunately OTP isn’t used due to practicality reasons such as
  - Key size is the same as the message length
  - Vulnerable to MITM modification
  - Keys cannot be reused
Exercise 1

Decrypt the following message

5130394e5253425554794244564559675530465556564a4551566c540a

hint: all about that base
Exercise 1

Decrypt the following message

5130394e5253425554794244564559675530465556564a4551566c540a

hint: all about that base

hint 2: 16 and 64 sure are interesting numbers
Exercise 1

Answer: COME TO CTF SATURDAYS

```python
>>> print(msg)
5130394e5253425554794244564559675530465556564a4551566c540a
>>> d = msg.decode("hex").decode("base64")
>>> print(d)
COME TO CTF SATURDAYS
```
Symmetric Crypto Defined

- **Symmetric cryptography** is a type of cryptosystem where the same key is used for encryption and decryption.

- This differs from other forms of cryptography such as **asymmetric cryptography** commonly referred to as **public key cryptography** where the encryption and decryption keys are different.

- We will talk about public key crypto next meeting.
Symmetric Crypto Defined

- There are two main types of symmetric key algorithms stream and block cipher.
Symmetric Crypto Defined

- There are two main types of symmetric key algorithms **stream** and **block** cipher.

- Stream ciphers typically encrypt a message one byte at a time.
Symmetric Crypto Defined

• There are two main types of symmetric key algorithms **stream** and **block** cipher.

• Stream ciphers typically encrypt a message one byte at a time.

• Block ciphers typically encrypt a certain number of bits at a certain time (e.g. 64bit) and pad the message so that it is a multiple of the block size.
Stream Ciphers

- Exactly like OTP except not truly random numbers can be used and key size is often less than the message size
- In comparison to block ciphers, stream ciphers are usually faster
- Stream ciphers are vulnerable to MITM attack
- Examples of widely used stream ciphers RC4, A5/1, A5/2, FISH
Block Ciphers

- Block ciphers work by encrypting groups of plaintext into blocks (8, 16, 32, 64, 128, 256) of bits.
- Since block ciphers have fixed size blocks, we must pad any group of plaintext which isn’t exactly a multiple of the bit size.
- Block ciphers have various modes such as ecb, cbc, ctr, cfb.
- DES, AES, Blowfish are widely used block ciphers.
Modification Problem

- One of the issues that all the ciphers have we have talked about is that a malicious user can intercept and modify the ciphertext.
Modification Problem

- One of the issues that all the ciphers have we have talked about is that a malicious user can intercept and modify the ciphertext.

Bob HI Alice
Modification Problem

- One of the issues that all the ciphers have we have talked about is that a malicious user can intercept and modify the ciphertext.

\[ E(HI) = @_ \]

Bob \_ @ Alice
Modification Problem

- One of the issues that all the ciphers have we have talked about is that a malicious user can intercept and modify the ciphertext.
Modification Problem

• One of the issues that all the ciphers have we have talked about is that a malicious user can intercept and modify the ciphertext.
Modification Problem

• One of the issues that all the ciphers have we have talked about is that a malicious user can intercept and modify the ciphertext.

\[ D(_\_\_\_) = PI \]

Bob

Eve

$\_$

Alice
Modification Problem

- One of the issues that all the ciphers have we have talked about is that a malicious user can intercept and modify the ciphertext.

- Eve was able to successfully modify the message without Alice ever knowing.

- How do we prevent Eve from modifying the message?
Modification Problem

• One of the issues that all the ciphers have we have talked about is that a malicious user can intercept and modify the ciphertext.

• Eve was able to successfully modify the message without Alice ever knowing.

• How do we prevent Eve from modifying the message? You can’t. We can however add hash to the message to make it so that Alice has a good idea the message was tampered with.
MAC Defined

- A **MAC** or Message Authentication Code is a way of adding a checksum to a message.

\[ A = H_K(M) \]  (authentication code)
\[ H = \text{Hash function (note: not all MACS use hash)} \]
\[ K = \text{Key for hash function} \]
\[ M = \text{Message} \]
MAC Defined

• MAC works because both Bob and Alice know the hash function and a secret key.

• The message is then hashed and this creates a “unique” MAC which will be sent in addition with the message
MAC Example

• Bob wants to send a message to Alice but he wants to make sure it will resist MITM modification attacks.

Bob important Alice
MAC Example

• Bob wants to send a message to Alice but he wants to make sure it will resist MITM modification attacks.

\[ H_K(M) = A \]
MAC Example

- Bob wants to send a message to Alice but he wants to make sure it will resist MITM modification attacks.

\[ H_K(M) = A \]
MAC Example

• Bob wants to send a message to Alice but he wants to make sure it will resist MITM modification attacks.

\[ H_K(M) = A \]
MAC Example

• Bob wants to send a message to Alice but he wants to make sure it will resist MITM modification attacks.

\[ H_K(M) = A \]

\[ H_K(M') \neq A \]

Bob

Alice

impotent

Eve
MAC Example

• Alice knows the message isn’t legit and puts it where it belongs
Exercise 2

I have devised the ultimate cryptosystem. I’m so confident in its security I will give you the key and you still will never find out my secret message.

message = DwcTBwAUDhtsBmVzDQAZAhnAdaQoI
key = WHAT WOULD SNOWDEN DO

hint = do you think this cryptosystem is exclusive or what
Exercise 2

I have devised the ultimate cryptosystem. I’m so confident in its security I will give you the key and you still will never find out my secret message.

message = DwcTBwAUDhtsBmVzDQAZAhAdaQoI
key = WHAT WOULD SNOWDEN DO

hint = do you think this cryptosystem is exclusive or what

hint 2 = ord() and chr() are your friends
Exercise 2

Answer: XORS CAN BE CONFUSING

Step 1. Decode the message encoded in base64
Step 2. xor the Decoded message with the key

```python
def str_xor(s1, s2):
    return "".join([chr(ord(c1) ^ ord(c2)) for (c1, c2) in zip(s1, s2)])

key = "WHAT WOULD SNOWDEN DO"
msg = "XORS CAN BE CONFUSING"

encoded = str_xor(msg, key).encode("base64").strip()
print(encoded)

decoded = str_xor(encoded.decode("base64").strip(), key)
print(decoded)
```
Summary

- Don’t roll your own crypto
- Remember Kerckhoff’s principle (don’t trust secret algorithms)
- Symmetric cryptosystems use the same key for encryption as they do for decryption
- MITM attacks can be deterred using MACS but cannot be prevented
Where to learn more

• Cryptopals.com

• Microcorruption.com (warning all assembly)

• Mystertwisterc3.org (website is slow and awful)

• CTFS (like on Saturdays…..see what I’m getting at)