



FA2023 Week 07 • 2023-10-12

Crypto I

Anakin and Sagnik

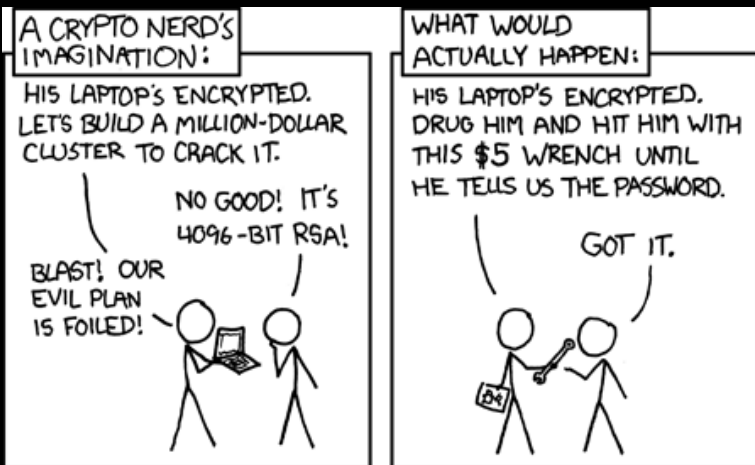
Announcements

- Lockpicking Support Group!
 - Come practice lockpicking
 - Mondays 8-9 PM



ctf.sigwny.com

sigwny{n0t_that_crypt0_but_th3_0th3r_0n3}



Outline

Basics

XOR

Diffie-Hellman



Scoreboard

Place	User		Score
1	ronanboyarski	+1.5k points	28035
2	NullPoExc		24515
3	caasher	+4k points	21290
4	CBCicada	+9k points (up 1 place)	18015
5	mgcsstywth	+.1k points	17125
6	EhWhoAml		8645
7	aaronthewinner	+.4k points	7655
8	ilegosmaster		6660
9	drizzle	+.1k points	6225
10	SHAD0WV1RUS		5970



Get Involved Callout

Looking for people to:

- run meetings
- plan events
- create challenges
- get more involved in the club :0
- Let us know if interested!



Section 1

Basics



What is Crypto Anyways?



Alice



Bob



Eve



Why Do We Care?



Why Do We Care?



Crypto in Ye Olden Days

- Relied on simple patterns
- Hard / annoying to break by hand, **easy to break by computer**



Crypto in Ye Olden Days

- Relied on simple patterns
- Hard / annoying to break by hand, **easy to break by computer**
- Examples:
 - Caesar Cipher (rot k)
 - $a \rightarrow c, b \rightarrow d, \dots, y \rightarrow a, z \rightarrow b$ (rot 2)



Crypto in Ye Olden Days

- Relied on simple patterns
- Hard / annoying to break by hand, **easy to break by computer**
- Examples:
 - Caesar Cipher (rot k)
 - $a \rightarrow c, b \rightarrow d, \dots, y \rightarrow a, z \rightarrow b$ (rot 2)
 - Substitution
 - Create a table mapping each letter to another
 - Generalization of Caesar Cipher



Crypto in Ye Olden Days

- Relied on simple patterns
- Hard / annoying to break by hand, **easy to break by computer**
- Examples:
 - Caesar Cipher (rot k)
 - $a \rightarrow c, b \rightarrow d, \dots, y \rightarrow a, z \rightarrow b$ (rot 2)
 - Substitution
 - Create a table mapping each letter to another
 - Generalization of Caesar Cipher
 - Many More
 - **All insecure!!**



Data Representation

- TL;DR: computers store things in binary (0s and 1s), and we have different ways of representing this



Data Representation

- TL;DR: computers store things in binary (0s and 1s), and we have different ways of representing this
- Look at the challenge source if given and mimic what they do



Data Representation

- TL;DR: computers store things in binary (0s and 1s), and we have different ways of representing this
- Look at the challenge source if given and mimic what they do
- Tip: In Python, always work with bytes / bytestrings, never with normal strings (Python 3.8+)



Conversion Cheatsheet

This is hard to read, download the slides!!

Format	Description	From Bytes	To Bytes
base64	uses printable letters to encode more complex binary	<code>base64.b64encode</code>	<code>base64.b64decode</code>
hex	uses symbols 0-9, A-F	<code>bytearray.hex()</code> , <code>binascii.hexlify()</code>	<code>bytes.fromhex()</code> , <code>binascii.unhexlify()</code>
integer	normal integers	<code>Crypto.Util.number.bytes_to_long</code> (PyCryptoDome), <code>int.from_bytes</code>	<code>Crypto.Util.number.long_to_bytes</code> (PyCryptoDome), <code>int.to_bytes</code>

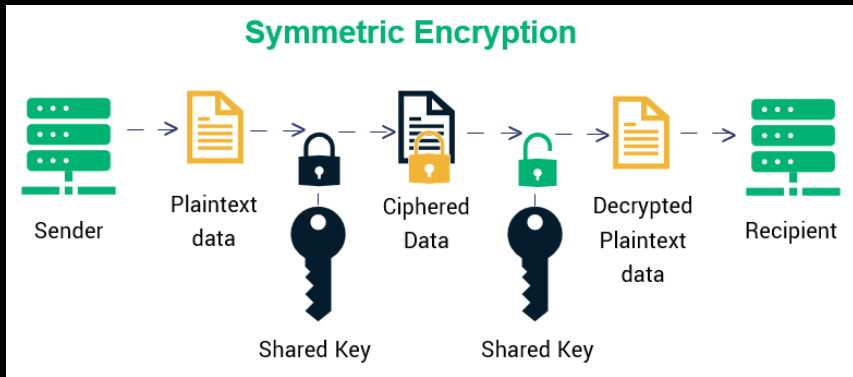


Section 2

XOR



Symmetric Encryption



XOR

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0



XOR

- XOR has some really nice properties that make it perfect for symmetric encryption
- Say M is some message as a bitstring, K is some key
- Then let $C = M \oplus K$ be a ciphertext



XOR

- XOR has some really nice properties that make it perfect for symmetric encryption
- Say M is some message as a bitstring, K is some key
- Then let $C = M \oplus K$ be a ciphertext
- Properties:
 - **Order doesn't matter:** $M \oplus K = K \oplus M$



XOR

- XOR has some really nice properties that make it perfect for symmetric encryption
- Say M is some message as a bitstring, K is some key
- Then let $C = M \oplus K$ be a ciphertext
- Properties:
 - **Order doesn't matter:** $M \oplus K = K \oplus M$
 - **Group as needed:** $M \oplus (K \oplus K) = (M \oplus K) \oplus K$



XOR

- XOR has some really nice properties that make it perfect for symmetric encryption
- Say M is some message as a bitstring, K is some key
- Then let $C = M \oplus K$ be a ciphertext
- Properties:
 - **Order doesn't matter:** $M \oplus K = K \oplus M$
 - **Group as needed:** $M \oplus (K \oplus K) = (M \oplus K) \oplus K$
 - **\emptyset is the identity:** $M \oplus \emptyset = M$



XOR

- XOR has some really nice properties that make it perfect for symmetric encryption
- Say M is some message as a bitstring, K is some key
- Then let $C = M \oplus K$ be a ciphertext
- Properties:
 - **Order doesn't matter:** $M \oplus K = K \oplus M$
 - **Group as needed:** $M \oplus (K \oplus K) = (M \oplus K) \oplus K$
 - **0 is the identity:** $M \oplus 0 = M$
 - **Self Inverse:** $K \oplus K = 0$



XOR

- XOR has some really nice properties that make it perfect for symmetric encryption
- Say M is some message as a bitstring, K is some key
- Then let $C = M \oplus K$ be a ciphertext
- Properties:
 - **Order doesn't matter:** $M \oplus K = K \oplus M$
 - **Group as needed:** $M \oplus (K \oplus K) = (M \oplus K) \oplus K$
 - **0 is the identity:** $M \oplus 0 = M$
 - **Self Inverse:** $K \oplus K = 0$
- All of this means $C \oplus K = M \oplus K \oplus K = M \oplus 0 = M$



Overview of Easy Some Attacks

- For certain reasons, in general XOR is really really hard to break
 - Without more information, you need to try 2^λ guesses to break a bitstring of length λ



Overview of Easy Some Attacks

- For certain reasons, in general XOR is really really hard to break
 - Without more information, you need to try 2^λ guesses to break a bitstring of length λ
- Usually you need to know some information about the plaintext
 - Known plaintext + ciphertext pair
 - Properties like language (common letters / words)



Overview of Easy Some Attacks

- For certain reasons, in general XOR is really really hard to break
 - Without more information, you need to try 2^λ guesses to break a bitstring of length λ
- Usually you need to know some information about the plaintext
 - Known plaintext + ciphertext pair
 - Properties like language (common letters / words)
- You may need to know some information about the key
 - Really short keys are able to be brute forced



Overview of Easy Some Attacks

- For certain reasons, in general XOR is really really hard to break
 - Without more information, you need to try 2^λ guesses to break a bitstring of length λ
- Usually you need to know some information about the plaintext
 - Known plaintext + ciphertext pair
 - Properties like language (common letters / words)
- You may need to know some information about the key
 - Really short keys are able to be brute forced
 - Flag Formats: `sigpwny{`



Section 3

Diffie-Hellman



Modular Arithmetic

- Numbers can get really big really fast



Modular Arithmetic

- Numbers can get really big really fast
- We use **modular arithmetic** to deal with this



Modular Arithmetic

- Numbers can get really big really fast
- We use **modular arithmetic** to deal with this
- Modular arithmetic is **arithmetic with remainders after division**
 - Keep taking remainders as you do arithmetic



Modular Arithmetic

- Numbers can get really big really fast
- We use **modular arithmetic** to deal with this
- Modular arithmetic is **arithmetic with remainders after division**
 - Keep taking remainders as you do arithmetic
- If we do computation **mod** n that means we will take remainders after division by n



Remainders

- Assume we have some number n . We are going to do some computation **mod** n



Remainders

- Assume we have some number n . We are going to do some computation **mod** n
- For now, say $n = 101$

$$131 + 140 * (102)^{2000} \equiv 131 + 39 * (102)^{2000} \pmod{101}$$



Remainders

- Assume we have some number n . We are going to do some computation **mod** n
- For now, say $n = 101$

$$\begin{aligned} 131 + 140 * (102)^{2000} &\equiv 131 + 39 * (102)^{2000} && (\text{mod } 101) \\ &\equiv 30 + 39 * (102)^{2000} && (\text{mod } 101) \end{aligned}$$



Remainders

- Assume we have some number n . We are going to do some computation **mod** n
- For now, say $n = 101$

$$\begin{aligned} 131 + 140 * (102)^{2000} &\equiv 131 + 39 * (102)^{2000} && (\text{mod } 101) \\ &\equiv 30 + 39 * (102)^{2000} && (\text{mod } 101) \\ &\equiv 30 + 39 * (1)^{2000} && (\text{mod } 101) \end{aligned}$$



Remainders

- Assume we have some number n . We are going to do some computation **mod** n
- For now, say $n = 101$

$$\begin{aligned} 131 + 140 * (102)^{2000} &\equiv 131 + 39 * (102)^{2000} && (\text{mod } 101) \\ &\equiv 30 + 39 * (102)^{2000} && (\text{mod } 101) \\ &\equiv 30 + 39 * (1)^{2000} && (\text{mod } 101) \\ &\equiv 30 + 39 && (\text{mod } 101) \end{aligned}$$



Remainders

- Assume we have some number n . We are going to do some computation **mod** n
- For now, say $n = 101$

$$\begin{aligned} 131 + 140 * (102)^{2000} &\equiv 131 + 39 * (102)^{2000} && (\text{mod } 101) \\ &\equiv 30 + 39 * (102)^{2000} && (\text{mod } 101) \\ &\equiv 30 + 39 * (1)^{2000} && (\text{mod } 101) \\ &\equiv 30 + 39 && (\text{mod } 101) \\ &\equiv 69 && (\text{mod } 101) \end{aligned}$$



Discrete Log

- If $a^b \equiv X \pmod{p}$, $b =$ the **discrete log of X with base a**



Discrete Log

- If $a^b \equiv X \pmod{p}$, $b =$ the **discrete log of X with base a**
- Given some random X and a , finding b is really hard to compute for large primes p
- This **Discrete Log Problem (DLP)** is the basis for many modern cryptography standards



Trapdoors

- Think of the DLP as a trapdoor



Trapdoors

- Think of the DLP as a trapdoor
 - Easy to enter, hard to exit



Trapdoors

- Think of the DLP as a trapdoor
 - Easy to enter, hard to exit
- If $2^n \equiv 79 \pmod{97}$, what is n ?



Trapdoors

- Think of the DLP as a trapdoor
 - Easy to enter, hard to exit
- If $2^n \equiv 79 \pmod{97}$, what is n ?
- $n = 15$
- Imagine this with larger primes. Multiplication is easy, logs are hard

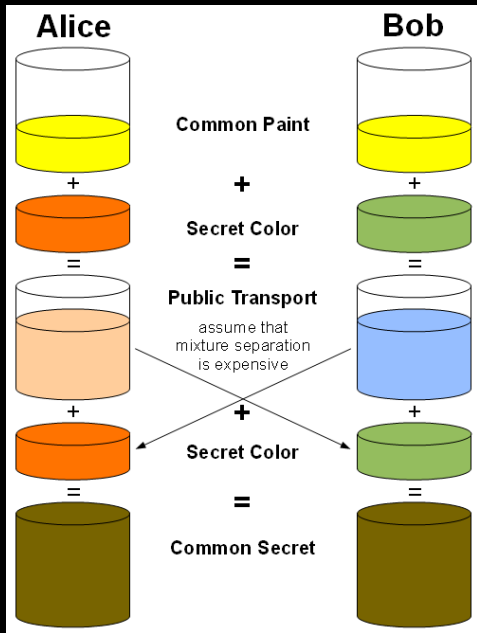


Trapdoors

- Think of the DLP as a trapdoor
 - Easy to enter, hard to exit
- If $2^n \equiv 79 \pmod{97}$, what is n ?
- $n = 15$
- Imagine this with larger primes. Multiplication is easy, logs are hard

Diffie-Hellman takes advantage of this!





Painting with Numbers

- Let g be a public number we call a generator and p be some public prime



Painting with Numbers

- Let g be a public number we call a generator and p be some public prime
- Alice generates secret a and computes $A \equiv g^a \pmod{p}$
- Bob generates secret b and computes $B \equiv g^b \pmod{p}$



Painting with Numbers

- Let g be a public number we call a generator and p be some public prime
- Alice generates secret a and computes $A \equiv g^a \pmod{p}$
- Bob generates secret b and computes $B \equiv g^b \pmod{p}$
- Alice sends Bob A and Bob sends Alice B



Painting with Numbers

- Let g be a public number we call a generator and p be some public prime
- Alice generates secret a and computes $A \equiv g^a \pmod{p}$
- Bob generates secret b and computes $B \equiv g^b \pmod{p}$
- Alice sends Bob A and Bob sends Alice B
- Alice computes $B^a \pmod{p}$
- Bob computes $A^b \pmod{p}$



Painting with Numbers

- Let g be a public number we call a generator and p be some public prime
- Alice generates secret a and computes $A \equiv g^a \pmod{p}$
- Bob generates secret b and computes $B \equiv g^b \pmod{p}$
- Alice sends Bob A and Bob sends Alice B
- Alice computes $B^a \pmod{p}$
- Bob computes $A^b \pmod{p}$

Alice and Bob now have the same key!

$$A^b \equiv (g^a)^b \equiv g^{ab} \equiv (g^b)^a \equiv B^a \pmod{p}$$



Overview of Some Attacks

Remember, discrete log in general is **hard**



Overview of Some Attacks

Remember, discrete log in general is **hard**

- Small Primes are easy to brute force
 - You have a computer, use it!



Overview of Some Attacks

Remember, discrete log in general is **hard**

- Small Primes are easy to brute force
 - You have a computer, use it!
- “Oracle” attacks: access to a special machine that leaks information
 - Write out what do and don't know as equations
 - Do not be afraid of pen and paper



Overview of Some Attacks

Remember, discrete log in general is **hard**

- Small Primes are easy to bruteforce
 - You have a computer, use it!
- “[Oracle](#)” attacks: access to a special machine that leaks information
 - Write out what do and don’t know as equations
 - Do not be afraid of pen and paper
- Primes are generated in specific ways
 - “Smooth Primes” p where $p-1$ has many factors
 - [Pohlig-Hellman](#), [Pollard’s Rho](#)
 - More on this next week with [advanced factoring](#)!



Misc Chals

- A lot of Crypto people like cute little math / puzzles, many challenges are just "reverse the math"



Misc Chals

- A lot of Crypto people like cute little math / puzzles, many challenges are just "reverse the math"
 - Solve some polynomial equations
 - Linear algebra
 - Undo Randomness
 - Is it really random? Does the randomness really have an effect on anything?



Misc Chals

- A lot of Crypto people like cute little math / puzzles, many challenges are just "reverse the math"
 - Solve some polynomial equations
 - Linear algebra
 - Undo Randomness
 - Is it really random? Does the randomness really have an effect on anything?
- **Strategy:** Just try things, look for patterns, more like math-y reverse engineering. Don't be afraid to just start.



Tools!

- Python + [SageMath](#) is your friend



Tools!

- Python + [SageMath](#) is your friend
- [PyCryptodome](#) is an extremely useful Python crypto library



Tools!

- Python + [SageMath](#) is your friend
- [PyCryptodome](#) is an extremely useful Python crypto library
- [PwnTools](#) will allow you to automate parts of your attacks



Tools!

- Python + [SageMath](#) is your friend
- [PyCryptodome](#) is an extremely useful Python crypto library
- [PwnTools](#) will allow you to automate parts of your attacks
- Google + StackOverflow (“how to crack DH with ...”)

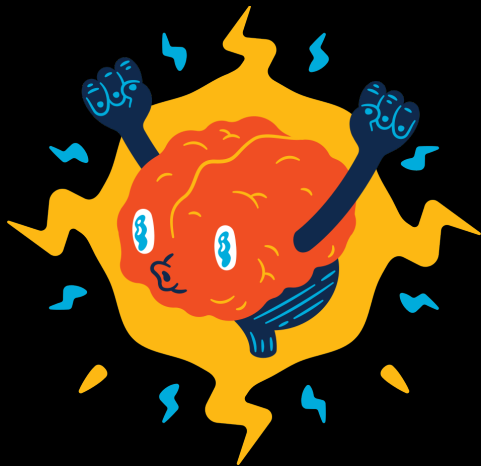


Tools!

- Python + [SageMath](#) is your friend
- [PyCryptodome](#) is an extremely useful Python crypto library
- [PwnTools](#) will allow you to automate parts of your attacks
- Google + StackOverflow (“how to crack DH with ...”)
- Installation is **annoying**, use the [CryptoHack Docker](#)



Practice @ CryptoHack



Next Meetings

2023-10-15 – This Sunday

- Crypto II
- More Diffie-Hellman + RSA

2023-10-19 – Next Thursday

- PWN I with Sam

2022-10-22 – Next Sunday

- Pwn II with Kevin



ctf.sigpwny.com

sigpwny{n0t_that_crypt0_but_th3_0th3r_0n3}

Thanks for listening!



SIGPwny