



Intro to Crypto in CTFs

Zaine



In the news...



Agenda

What is Cryptography?

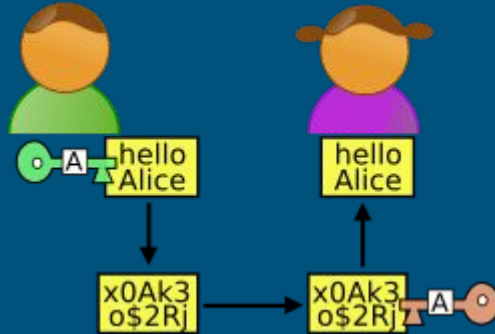
Encodings

Classical Cryptography in CTFs

Modern Crpytography in CTFs

What is Cryptography?

- Getting information from point A to point B without exposing it to anyone in the middle
- Mathematically guaranteed trust



Common Types of Cryptography in CTFs

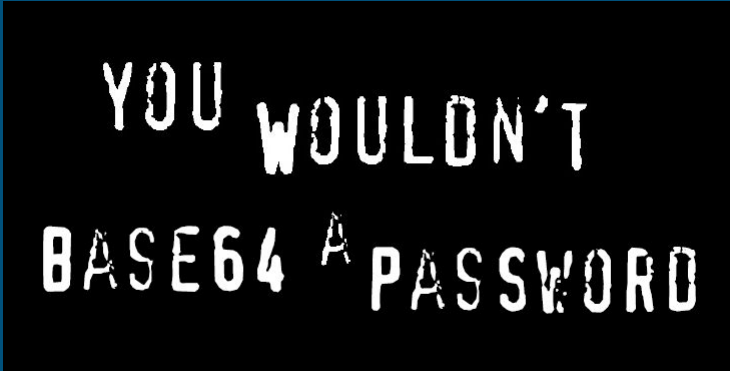
- Low-scoring crypto tends to be classical cryptography, and leverages how bad classical cryptography is
- Higher scoring problems tend to use modern cryptography, and leverage improper implementation or side channels

Easy Crypto

- Encoding
- Transposition Ciphers
- Substitution Ciphers
- Polyalphabetic Ciphers
- Simple Modern Crypto



Encoding vs Encryption



YOU WOULDN'T
BASE64 A PASSWORD

- No keys in encoding
- Encoding offers NO security
- Anyone who knows the encoding used can get the information out
- If you're encoding something and claiming it's secure, people will laugh at you

ENCRYPTION \neq ENCODING

Encoding: B64 (Base-64)

- Turns raw data into printable characters by changing its base to 64 (!)
- How to recognize:
 - Character set: English Uppercase, English Lowercase, “/”, and “+”
 - Dead giveaway: one or two trailing “=”
 - B64 is padded with “=” due to base conversion
 - If you see trailing “=”, it’s probably B64 encoded
- How to decrypt:
 - Google B64 decode

The screenshot shows the 'BASE64 Decode and Encode' web application. At the top, there's a green header with the title 'BASE64' and a subtitle 'Decode and Encode'. Below the header, there are two main buttons: 'Decode' and 'Encode', each with a folder icon. The 'Decode' button is highlighted. Below the buttons, there's a section titled 'Decode from Base64 format' with the instruction 'Simply use the form below'. The form includes a dropdown menu set to 'UTF-8' with the text 'You may also select input charset.' below it. There are two checkboxes: 'Live mode OFF' with the description 'Decodes while you type or paste.' and 'UPLOAD FILE' with the description 'Decodes an entire file (max. 10MB)'.

“TWFzb25Db21wZXRpdGI2ZUN5YmVyIQ==”→B64→“MasonCompetitiveCyber!”

Encoding: Morse

- Encodes characters to dots and dashes
- How to recognize:
 - Series of “-” and “.”, or similar looking characters
 - These tend to be fairly long, as morse code isn’t very efficient (encoding 1 character to up to 5 characters)
- How to decode:
 - Google is still your friend. Some find-and-replace may be necessary

"- - --- --- -- - .. - ---" → Morse Code → "MasonCompetitiveCyber"

Encoding: Base Conversion

- Converts characters to numbers base n from their ascii encoding
- How to recognize:
 - If encoded text has only digits, you can recognize:
 - Binary: 0,1
 - Ternary: 0,1,2
 - Quaternary: 0,1,2,3
 - Etc...
- How to decode:
 - Might just be able to jump to Google again, but a python script or similar can be used for decoding

Sample base-decoding in python

```
1 #sample base decoding
2 ciphertext = "2212 10121 11021 11010 11002 2111 11010 11001 11011 10202 11022 10220 11022 10220 11101 10202 2111 11111 10122 10202 11020"
3 character_delimeter = " "
4 ciphertext_array = ciphertext.split(character_delimeter)
5
6 base = 3
7
8 plaintext = ""
9 for character in ciphertext_array:
10     plaintext+=chr(int(character,base))
11 print plaintext
```

C:\WINDOWS\system32\cmd.exe

C:\Users\Zaine Wilson\Desktop>python base_deencoder.py
MasonCompetitiveCyber

C:\Users\Zaine Wilson\Desktop>

Somewhere in between: ROT13

- Not quite a cipher, still has no key
- Rotates every character 13 places down the alphabet
- 'A' → 'N', 'B' → 'O' etc
- How to recognize:
 - No ridiculous characters, not obviously hex encoded, low-point crypto question
- How to break:
 - Gooooooooooooooooogle.
 - If you really wanted to, you could use a script. But I'm pretty sure the internet is actually 70% Javascript ROT13 crackers, so really just use someone else's.
 - ROT13 encryption is the same as ROT13 decryption



Shift Cipher: Caesar Cipher

- Like ROT13, but ROT n .
- Rotates characters through the alphabet by n
- How to recognize:
 - Like ROT13, usually a low-point crypto challenge with alphanumeric text
- How to break:
 - Systematically try every shift from 0-25
 - Can be done in a script or online
 - www.dcode.fr

“ThzvUJvtwlapapclJfily” → Decode Ceasar (Key 7) → “MasonCompetitiveCyber”

Shift Cipher: General Shift Ciphers

- Like ROT13 and Caesar, but shift over a larger alphabet or along the keyboard
- How to recognize:
 - Garbled text, may have spacing preserved
 - Not hex encoded
- Decoding:
 - Keyboard shift ciphers can be broken on www.dcode.fr
 - Shifts using the ASCII table as their alphabet may require some easy scripting



“N\aiBXinowrurucwXtvwe” → QWERTY Shift (left) → “MasonCompetitiveCyber”

ASCII Shift Example (from Metropolis)

```
C:\Users\Zaine Wilson\Desktop>python asciishift.py
B\\g:7cd+D`Zb\p
C]]h;8de,Ea[c]q
D^^i<9ef-Fb\d^r
E__j=:fg.Gc]e_s
F``k>;gh/Hd^f`t
Gaal?<hi0Ie_gau
Hbbm@=ij1Jf`hbv
IccnA>jk2Kgaicw
JddoB?kl3Lhbidx
KeepC@lm4Mickey
LffqDAmn5Njdlfz
MggrEBno6Okemg{
NhhsFCop7Plfnh|
OiitGDpq8Qmgoi}
```

Substitution Cipher: Monoalphabetic Substitution Ciphers

- Replaces each character in the plaintext with a character from a mapping dictionary
- Each character will be replaced with the same character every time
- How to recognize:
 - Same as previous two, but usually only alphabetical text and punctuation
 - Punctuation is generally preserved during enciphering
- How to break:
 - Automated solving: www.quipquip.com
 - Not always reliable for short texts, very good at long texts
 - Breaking it by hand: Using single letter or two-letter words to create a mapping
 - Breaking it by hand: Guessing mappings based on likely words in the plaintext (i.e. “flag”)

“vmwtn xtvbakykyza xpuah” → Substitution (Alphabet = ‘muxfaicoysrvntbdhwkqzelpg’) → “mason competitive cyber”

Transposition Cipher: Rail-Fence

- Transposition Cipher: scrambles around the letters in the plaintext to form the ciphertext
- How to recognize:
 - Spaces/punctuation in weird places
 - No strange characters
 - Generally not hex encoded
- How to break:
 - Systematically try every rail number and every offset, rails=2 is most common.
 - www.geocachingtoolbox.com has a good rail-fence gui for encoding/decoding
 - NOTE: Make sure the ciphertext is copied accurately, as missing only one character can result in a failed decrypt

“ovCmieMnptCraoeiyestb” → Rail-Fence(Rails=5, Offset=2) → “MasonCompetitiveCyber”

“ovCmieMnptCraoeiyest” → Rail-Fence(Rails=5, Offset=2) → “MreanComposetiveCity”

Transposition Cipher: General Transposition

- Mixes all the letters around based on some rule
- How to recognize:
 - Similar to rail-fence, there might be weird spacing or punctuation, but most of the time only the letters are transposed and spacing and punctuation is preserved
- How to break:
 - www.dcode.fr has a good tool for decoding Transposition Ciphers
 - Allows for the use of cribbed text, known key length, and brute forcing

“OMSON MACIPTVTEEI ECBRY”→Transposition(“FUNCRYPT”)→“MASON COMPETITIVE CYBER”

This example uses www.dcode.fr rules to create a transposition from a string-based key

Polyalphabetic Cipher: Vigenere

- Very commonly used polyalphabetic substitution cipher
- Unlike all of the last ciphers, the alphabet used for each letter in the ciphertext may be different
- Vigenere uses an alphabetic key to map plaintext into ciphertext

Vigenere Encryption

Key= "CRYPT"

Plain= "MASONCOMPETITIVECYBER"

CRYPTCRYPTCRYPTCRYPTC

MASONCOMPETITIVECYBER

ORQDGEFKEXVZR XO GTWQXT

⊠	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	PLAIN TEXT							
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T								
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U							V	W
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	KEYWORD				E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X		
Z		F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z				

Polyalphabetic Cipher: Vigenere

- How to recognize:
 - Alphanumeric plaintext, may have spacing preserved. Single-character words map to something other than two characters (In monoalphabetic substitution “I” and “A” map to a unique character each, while in polyalphabetic, they might map to a different character every time they appear)
- How to break:
 - Automated tools: www.dcode.fr, crib “flag” or “CTF”
 - Sometimes you’re just given the key
 - Sometimes the key or a partial key may be in the hints
 - Attacks on Index of Coincidence to find probable key lengths
 - This would take a long time to explain, check out www.practicalcryptography.com for more

Polyalphabetic Cipher: Autokey

- Autokey is very similar to vigenere, but the way the key repeats is different
- Instead of repeating the same key until the message length is reached, after the end of the key is reached the beginning of the plaintext is appended to the key

CRYPTMASONCOMPETITIVE

MASONCOMPETITIVECYBER

ORQDGOOEDRVWFXZXKRJZV

- The full key is revealed as decryption progresses

Classical Ciphers: The Rest

<http://practicalcryptography.com/ciphers/classical-era/>

Very complete list of classical ciphers and how to break them

Modern-ish Cryptography: One-Time pads

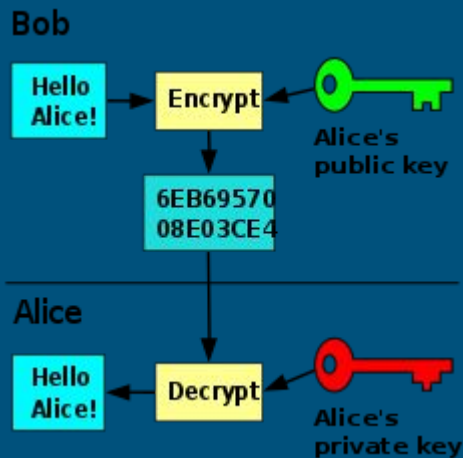
- Like vigenere, but we're using a random key which is the length of the string and using XOR in order to combine the key and the plaintext
- Theoretically unbreakable, *IF THE KEY IS NEVER RE-USED*
- If you've never heard of XOR:
 - $1 \text{ XOR } 1 = 0$
 - $0 \text{ XOR } 0 = 0$
 - $1 \text{ XOR } 0 = 1$
 - $0 \text{ XOR } 1 = 1$
- Using “^” as XOR, A as plaintext, B as the key and C as ciphertext:
 - $A \wedge B = C$
- Reusing the same key for message A' results in ciphertext C':
 - $A' \wedge B = C'$
- $C' \wedge C = (A' \wedge B) \wedge (A \wedge B) = (A' \wedge A) \wedge (B \wedge B) = (A' \wedge A) \leftarrow$ That's a lot of information revealed!

Modern-ish Cryptography: Single-Byte XOR

- TCTF Problem directly related to this
- Take every byte of the plaintext and XOR it by some byte to get the ciphertext
- How to recognize:
 - Hinted at in title or in challenge hints
 - Ciphertext is hex encoded or garbled after decode
- How to break:
 - Try every byte from 0x0 to 0xff. One of them will give you something that's not unprintable garbage, which will probably be your plaintext.

Modern Cryptography: Bad RSA keys

- Thanx to Metropolis CTF
- RSA is a symmetric key cryptosystem



Shameless Wikipedia Screenshot

1. Choose two distinct prime numbers, such as

$$p = 61 \text{ and } q = 53$$

2. Compute $n = pq$ giving

$$n = 61 \times 53 = 3233$$

3. Compute the Carmichael's totient function of the product as $\lambda(n) = \text{lcm}(p - 1, q - 1)$ giving

$$\lambda(3233) = \text{lcm}(60, 52) = 780$$

4. Choose any number $1 < e < 780$ that is coprime to 780. Choosing a prime number for e leaves us only to check that e is not a divisor of 780.

$$\text{Let } e = 17$$

5. Compute d , the modular multiplicative inverse of $e \pmod{\lambda(n)}$ yielding,

$$d = 413$$

Worked example for the modular multiplicative inverse:

$$d \times e = 1 \pmod{\lambda(n)}$$

$$413 \times 17 = 1 \pmod{780}$$

The **public key** is $(n = 3233, e = 17)$. For a padded plaintext message m , the encryption function is

$$c(m) = m^{17} \pmod{3233}$$

The **private key** is $(n = 3233, d = 413)$. For an encrypted ciphertext c , the decryption function is

$$m(c) = c^{413} \pmod{3233}$$

Modern Cryptography: Bad RSA

- RSA's security is provided by the difficulty of factoring large numbers into two primes
- Wow, that was nice and boring, wasn't it?

$$c \equiv m^e \pmod{n}$$

$$y = \frac{\sqrt{\text{I don't give a shit.}}}{x + 1}$$

$$c^d \equiv (m^e)^d \equiv m \pmod{n}$$

Basically:

- If you get the prime factors (p and q) of the public key, you win.

Modern Cryptography: Bad RSA

- Bad vs Good RSA keys

```
n=102029
e=13223
c=38574
```

```
-----BEGIN RSA PUBLIC KEY-----
MIIBCAQEA2E+hpIWDxrExdXmcuZiyHOM8rJ7u+OVRkkmxXEJPPu5P04IqQtSp
bYKIfk75+DNYSpa5L2SBwgbE1R+VEer5414a5SFY4RFpBwPfYV3R4uex7t+zyp1l
O/xCOVvp36Nm3mb84TTIFFGRpDh7Z8by58uunPLHUpDVzbqSIgbohRREYtkCyZMb
VTTRTSaESTHU6ih9NM88A/7ekStCmHNPyDBFWepwUOXMBuJbKdipBtIAPzz1xYS
WpVGJOV67OWFOMAoUaOnknQOLjfv8C2Pk7/KtMyfayHt59HTqRerzoKsmQumGeVA
8R162GQzoSsubTGmbegereojIoXBL4XfwIDAQAB
-----END RSA PUBLIC KEY-----
```

```
n=628821961583746936616084742316067948568308593760913447085877346
e=495858693331169811177329490125296887954995875786287931147804921
c=602 673 325 419 300 071 060 487 538 369 145 051 751 973 672 167
```

Mod

- The
- The
- Rea

62882196158374693661608474231606794856830859
37609134470858773469966884377135674386207998
52028989602673325419300071060487538369145051
75197367216789196038200936248912027185828591
69587227885465655543802567783983217024851725
60148407183025888152205749010740401024116528
38185694483639150459855925895943780475061354
95557919000695765114293601881416342686556452
03359275492506097523797726833661721244502743
66338908436438485202591088133248468403348033
74216290029210385949094969112131366577955728
60071003932367119838191764919398856675386406
00891748712451289865204909386070186245888494
01002298969620117686522018893220706525378171
33811175464627049585869333116981117732949012
52968879549958757862879311478049211590933479
76785597805108075622734878917355289917505198

Modern Cryptography: Bad RSA

- If you use small numbers, though...
- Like $n=102029$
- Which, let me check...

Modern Cryptography: Bad RSA

- Is WAY not big enough...

Prime Factors Calculator

Enter the Number to Factor

☐ *create a factorization tree*

☐ *also show me all factors*

Answer:

The Prime Factorization is:
257 x 397

Bad RSA

- We can then use Drexel's RSA calculator to find our private key from our primes
- And, using the private key, obtain the message!

$38574 \rightarrow \text{RSA}(\text{PrivKey} = 23, N=102029) \rightarrow \text{Base 26 Conversion} \rightarrow \text{"MCC"}$

If the numbers are too large for online factoring to handle, use YAFU, Yet Another Factoring Utility, which can factor up to 300 bit keys relatively quickly

End

Questions about:

Crypto?

Creating Hash Functions with Genetic Algos?

Sneaky beaky file exfiltration/C2?