CTF Crypto

Kanav

Feeling of Randomness

- Plaintext is always something we might know about, some pattern. i.e - for two messages m1, m2 - P(m1) != P(m2). For example, if we toss a coin - P(“heads”) = ½, P(“tails”) = ½ P(“hello world”) = 0
- But Ciphertext is generally random.
- But this probabilistic disparity in plaintext sometimes leads to breaking of complete cipher.

Must have tools

1. Python - both 2 and 3
    a. 2 is used when you know you have supported software and need simplicity
    b. 3 when need to integrate shit code from internet
    c. Pro tip: Always refer internet when converting *bytearray, bytes* and *string*.
    d. Install pwntools in both
    e. Install pycrypto
2. Sage Math
    a. Optional, never used personally
    b. Scar uses and solving group theory and ecc things is very easy using sagemath
3. A habit of backing up multiple times
    a. Things take time to generate in crypto, good to print everything out or even better write to file as well.

Terminology

1. Symmetric Encryption

   E(m, k) = c D(c, k) = m

2. Asymmetric Encryption

   Keygen() = pub, priv E(m, pub) = c D(c, priv) = m

3. Cryptographic Hashing

   H(m) = h h -> m not possible

4. Signing

   Keygen() = pub, priv Sign(m, priv) = s Verify(m, s, pub) = t/f

pwntools

- Interacting with programs hosted with nc (nc is just like running terminal programs, running on remote computer)

```
>>> conn = remote('ftp.ubuntu.com',21)
>>> conn.recvline() # doctest: +ELLIPSIS
b'220 ...'
>>> conn.send(b'USER anonymous\r\n')
>>> conn.recvuntil(b' ', drop=True)
b'331'
>>> conn.recvline()
b'Please specify the password.\r\n'
>>> conn.close()
```

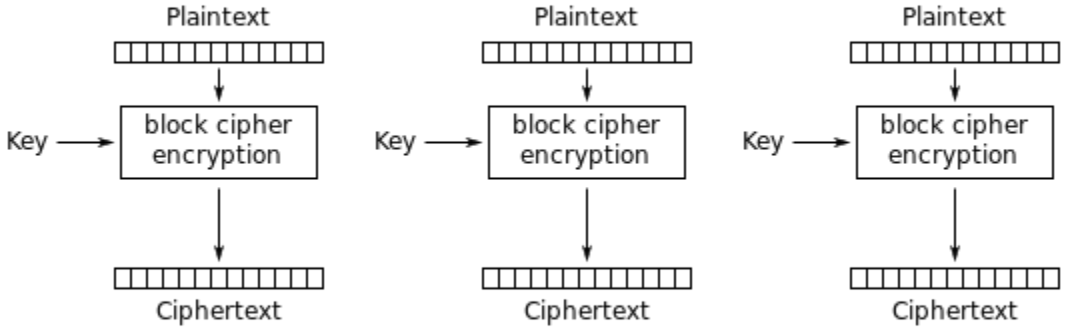
Classical Ciphers

- Substitution Cipher
  - divide the data into blocks, run a deterministic function on it, concatenate all result blocks
  - Attack - Frequency analysis attack
- Single byte Xor Cipher
  - ct[i] = p[i] ^ k
  - Attack - Frequency analysis attack
- Multibyte Xor Cipher
  - k = m-byte key, ct[i] = p[i] ^ k[i % m]
  - Take 1st, (m + 1)th, (2m+1)th…. Byte, concatenate, run single byte xor cipher breaker on it
  - Xortool

AES

- AES is a block cipher. It converts 128 bit (16 bytes) of plaintext into 128 bit or ciphertext using a key. It has three variants depending on key size. Bigger keysize means better security (debatable). AES-128, AES-196 and AES-256
- No need to know what’s inside it. For CTF purposes, assume that without key, you cannot encrypt/decrypt.
- As it only encrypts 128 bits, we developed some tricks to encrypt arbitrary length plaintexts called block modes.
- Note: Block modes are not limited to AES, but for any block cipher.

ECB Mode



Electronic Codebook (ECB) mode encryption

ECB Mode Security Problem - Non Diffusion

Assume block size is 8 bit

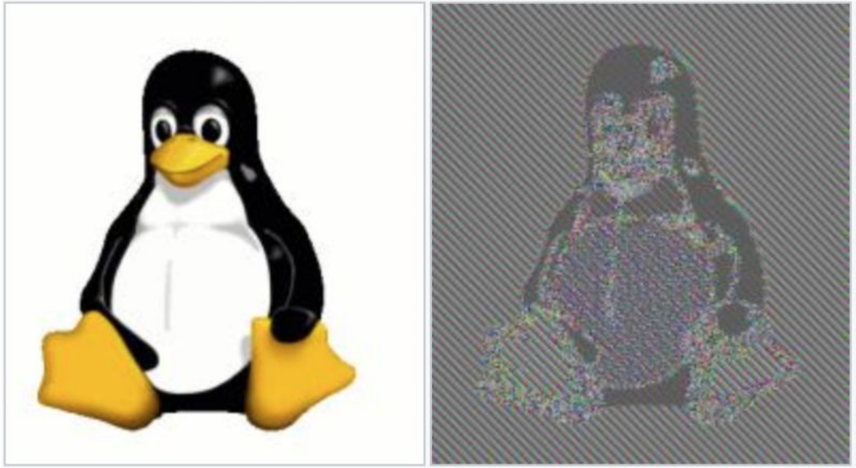
ECB(“A”) = “X”

ECB(“B”) = “Y”

ECB(“ABBA”) = “XYYX”

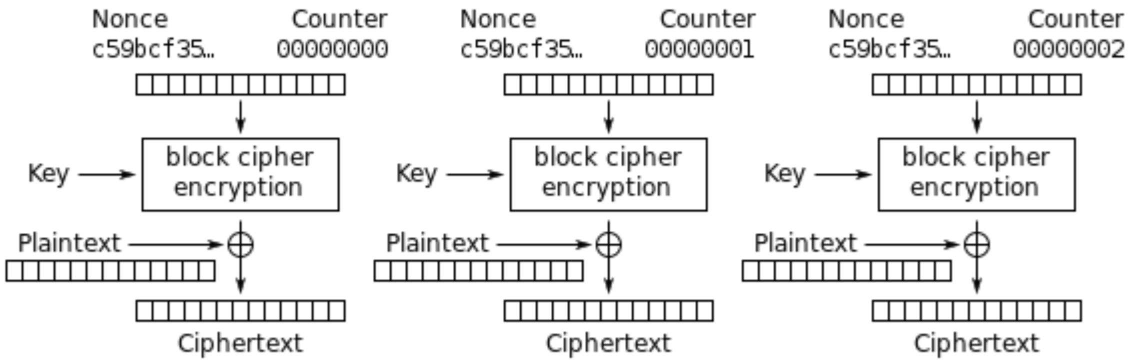
Simple Substitution Cipher if the key is fixed.

Assignment: Make the penguin on the right using PIL and AES.



Original image — Encrypted using ECB mode

CTR Mode

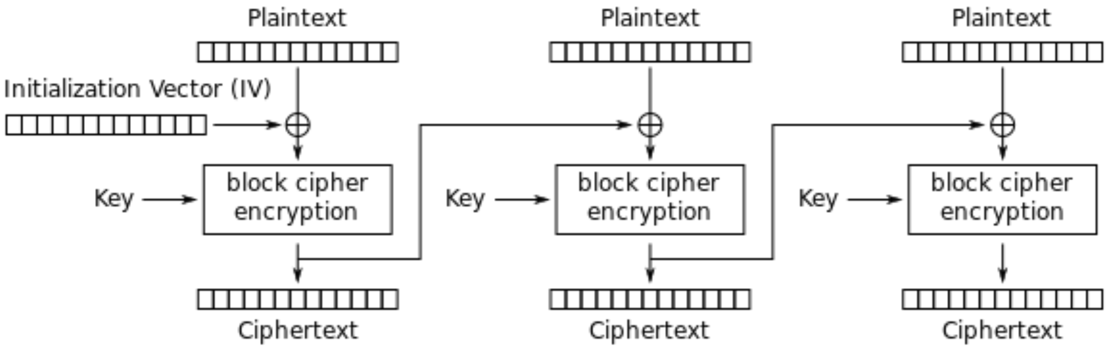


Counter (CTR) mode encryption

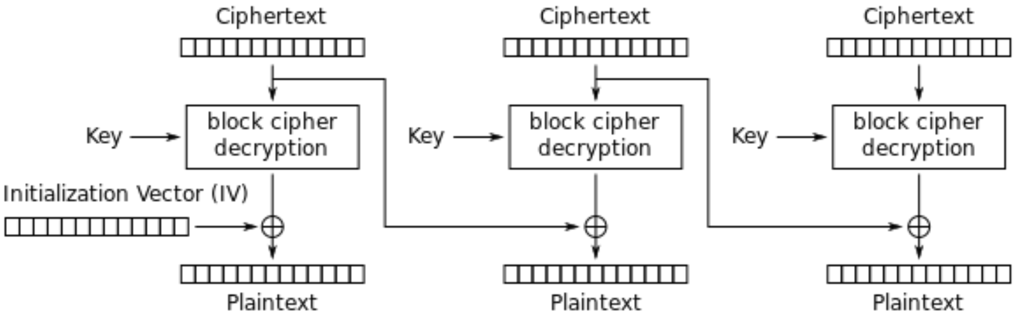
CTR Mode Security Problems

1. 1 p/c pair means game over
    - CTR Mode is like XOR cipher
    - X = CTR(key)
    - C = P ^ X
    - If you have P1, C1, X = C1 ^ P1
    - If you get C2, P2 = C2 ^ X
2. Bit flipping
    - P = C ^ X => P ^ 1 = (C ^ 1) ^ X

CBC Mode

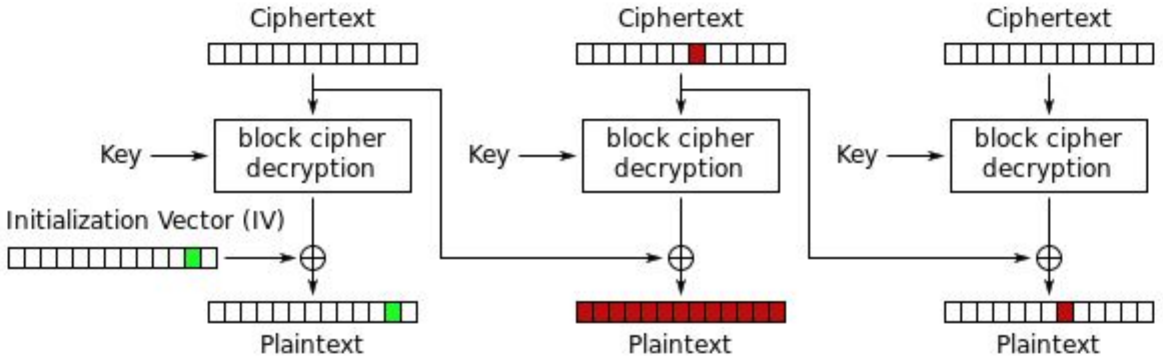


Cipher Block Chaining (CBC) mode encryption



Cipher Block Chaining (CBC) mode decryption

CBC Mode Security Problem - Bitflipping



Cipher Block Chaining (CBC) mode decryption

Padding

- Not all data is of size of multiple of 16 bytes.
- CTR Mode doesn’t need padding. (Why?)
- CBC and ECB need padding to be multiple of 16.
- Padding should be such that it is reversible.
- Pad(m) => mp, Unpad(mp) => m, Size(mp) = 16k
- Standard Algo -
    - Size = 16k + r, pad the byte (16-r), (16-r) times

Padding Security Problems - Padding Oracle Attack

- Programs might behave unexpectedly when they receive an invalidly padded data and pass them to unpad.
-

RSA Primer

N = p * q => public key

phi(N) = (p - 1) * (q - 1) => cannot calculate without p and q individually

phi - euler totient function

a ^ phi(N) mod N = a mod N   => fermat’s theorem

(m ^ e) ^ d mod N = m mod N   => e*d = 1 mod phi

Enc(m) = m ^ e mod N   Dec(c) = c ^ d mod N

Popular RSA Attacks

Observation	Attack
p and q too close (or any linear relation between the two)	p = sqrt(N) then start increasing until p | N (or solve the equation)
size(m) * e < size(N)	m = c ^ (1/e) in normal arithmetic (use binary search)
same message m, same e, but e public keys Ni	Use chinese remainder theorem to calculate c = M^e (mod N1*N2*....Ne), then m = c^(1/e) in normal arithmetic
multiple Ns	Try pairwise GCD on all Ni
blinding	
Server decrypts and calculates LSB	LSB Attack

Always try to limit the private key as much as possible!

Shamir Secret Sharing

(n, t) - secret sharing techniques - (Split, Reconstruct)

Split(secret) = s1, s2, s3….sn

Reconstruct(s1, s2, … st) = secret (or any t of the n splits)

Shamir - make a random polynomial (p) with constant = secret, of degree (t - 1)

s1 = (1, p(1)), s2 = (2, p(2)) …, we can reconstruct a polynomial with any of degree + 1 = t points

Discrete Logarithm Problem

a^x = b   => x = loga(b)

In normal arithmetic (i.e. Z or N) x is easy to calculate given a and b - as log is an increasing function. We can run binary search.

But in discrete settings, things are not easy. (as we saw in RSA)

Diffie Hellman Key Exchange - two people on internet, want to generate a common number only known to these two people.

Party 1 - make a random a, random g, send (g, g^a mod N) to second person

Party 2 - make a random b, key = (g^a mod N) ^ b mod N, send g^b mod N to 1

Party 1 - key = (g^b mod N) ^ a mod N

ECC - Elliptic Curve Group in Discrete Logarithm

- $y^2 = x^3 + ax + b$
- Points on this curve make an element on group, zero point on infinity
- Point Addition (P + Q) - connect two points, extend line, where line intersects again, take reflection along x axis, that point is your result
- Point Addition with same point (P + P) - tangent instead of line
- Scalar Multiplication (sP) = add s times
- given P and Q, finding s such that sP = Q is hard

Challenges -
https://github.com/kanav99/csaw19-quals-writeup/tree/master/brillouin-crypto-500

Name contains hints

sss - shamir secret sharing

lowe - small e

Further Reading

1. CRIME Attack
2. Smart’s Attack
3. Learn to use openssl (or maybe during ctf?)