IMAGE ENCRYPTION AND SHARING

By

Debadrita Roy (001910501025) & Ratul Chakraborty (001910501088)

BCSE - IV

BACKGROUND: WHAT IS ENCRYPTION?

Encryption is the process of encoding information. This process converts the original representation of the information, known as plaintext, into an alternative form known as ciphertext. Ideally, only authorized parties can decipher a ciphertext back to plaintext and access the original information. Encryption does not itself prevent interference but denies the intelligible content to a would-be interceptor. Since data may be visible on the Internet, sensitive information such as passwords and personal communication may be exposed to potential interceptors. If the encryption method is effective, it should completely protect data from unauthorized access. The process of encrypting and decrypting messages involves keys. The two main types of keys in cryptographic systems are symmetric-key and public-key (also known as asymmetric-key).

NEED FOR ENCRYPTION

According to The Software Alliance, cybercriminals stole 423 million identities in 2015. In 2018, data breaches exposed five billion records, which was a drop from the 7.9 billion records that were compromised in 2017, but is still no small number. The threats to data security continue to multiply: in 2019, the top cybersecurity concerns include relatively new types of threats, such as formjacking, cross-site scripting XSS attacks, and AI botnets. Consumers aren't the only losers when data is compromised — companies often lose employee data as well. Businesses must be well-versed in encryption methods and communications to help protect their own and their customers' sensitive data. The cyberworld is on pace to create 44 zettabytes (ZB) of data by 2020, which, according to the World Economic Forum, is "40 times more bytes than there are stars in the observable universe." With that amount of data in play, encryption is an absolute necessity for communicating online with privacy and security. According to Internet World Stats, as of July 2022, 69% of the world population are using the Internet.

BACKGROUND: WHAT IS SECRET SHARING?

Secret sharing (also called secret splitting) refers to methods for distributing a secret among a group, in such a way that no individual holds any intelligible information about the secret, but when a sufficient number of individuals combine their 'shares', the secret may be reconstructed. Whereas insecure secret sharing allows an attacker to gain more information with each share, secure secret sharing is 'all or nothing' , 'all' meaning the necessary number of shares.

In one type of secret sharing scheme there is one dealer and n players. The dealer gives a share of the secret to the players, but only when specific conditions are fulfilled will the players be able to reconstruct the secret from their shares. The dealer accomplishes this by giving each player a share in such a way that any group of t (for threshold) or more players can together reconstruct the secret but no group of fewer than t players can. Such a system is called a (t, n)-threshold scheme.

Secret sharing was invented independently by Adi Shamir and George Blakley in 1979.

EXISTING SECRET SHARING SCHEMES

- SHAMIR'S SCHEME

In this scheme, any t out of n shares may be used to recover the secret. The system relies on the idea that you can fit a unique polynomial of degree $t - 1$ to any set of t points that lie on the polynomial. The method is to create a polynomial of degree $t - 1$ with the secret as the first coefficient and the remaining coefficients picked at random. Next find n points on the curve and give one to each of the players. When at least t out of the n players reveal their points, there is sufficient information to fit a $(t - 1)$th degree polynomial to them, the first coefficient being the secret.

EXISTING SECRET SHARING SCHEMES

- BLAKLEY'S SCHEME

Any *n* nonparallel (*n* − 1)-dimensional hyperplanes intersect at a specific point. The secret may be encoded as any single coordinate of the point of intersection. If the secret is encoded using all the coordinates, even if they are random, then an insider gains information about the secret since he knows it must lie on his plane. If an insider can gain any more knowledge about the secret than an outsider can, then the system no longer has information theoretic security. If only one of the *n* coordinates is used, then the insider knows no more than an outsider. Each player is given enough information to define a hyperplane; the secret is recovered by calculating the planes' point of intersection and then taking a specified coordinate of that intersection. Blakley's scheme is less space-efficient than Shamir's; while Shamir's shares are each only as large as the original secret, Blakley's shares are *t* times larger, where *t* is the threshold number of players. Blakley's scheme can be tightened by adding restrictions on which planes are usable as shares. The resulting scheme is equivalent to Shamir's polynomial system.

EXISTING SECRET SHARING SCHEMES

- USING THE CHINESE REMAINDER THEOREM

Mignotte's scheme and Asmuth-Bloom's scheme use the CRT for secret sharing. *n* relatively prime integers m1,m2,...,mN are chosen such that the secret *S* is smaller than the product of any choice of *k* of these integers, but at the same time is greater than any choice of *k-1* of them. Then the shares s1,s2,..,sN are defined by s(i)=S(mod m(i)) for i=1,2,3,..N. In this manner, thanks to the CRT, we can uniquely determine *S* from any set of *k* or more shares, but not from less than *k*. Since *S* is smaller than the smallest product of *k* of the integers, it will be smaller than the product of any *k* of them. Also, being greater than the product of the greatest $k - 1$ integers, it will be greater than the product of any $k - 1$ of them.

WHAT WE SET OUT TO ACHIEVE?
~ PROBLEM STATEMENT

We aimed to design a secret sharing scheme to be used to distribute shares of images to a number of people, such that when a predefined number of people (say, k) come together with their shares, the original image can be reconstructed. Any combination of <k people will not be able to reconstruct the original image.

HOW WE WENT ABOUT IT?

We first used AES (Output Feedback Mode) to encrypt the given image, which is to be securely sent, and which is henceforth referred to as the secret S. Then we reshape the encrypted secret into a square matrix (adding padding if necessary). We then utilize the idea of linear combination. We generate a list of random coefficients (length=order of the square matrix) and perform dot product with each column of the square matrix. We repeat this a few times and generate a number of [p,c] combinations, where p stands for the random coefficients and c stands for the resultant vectors. Each person is given a number of [p,c] combinations such that if any k people bring their sets, they can construct square matrices P and C from them and get the secret S as $P^{-1}C$.

BACKGROUND: ADVANCED ENCRYPTION STANDARD (AES)

AES is a specification for the encryption of electronic data established by the U.S National Institute of Standards and Technology (NIST) in 2001. AES is widely used today as it is a much stronger than DES and triple DES. It processes 128 bits of input data at a time and outputs 128 bits of encrypted cipher text. AES relies on substitution-permutation network principle which means it is performed using a series of linked operations which involves replacing and shuffling of the input data. The number of rounds depends on the key length as follows :

- 128 bit key – 10 rounds
- 192 bit key – 12 rounds
- 256 bit key – 14 rounds

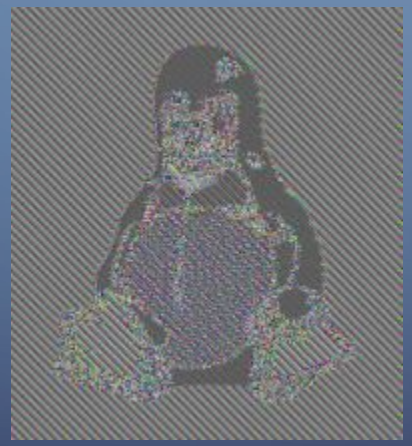
RATIONALE BEHIND USING AES (OFB MODE)

AES is very fast and secure, and is the de facto standard for symmetric encryption. As we were encrypting images, there were numerous blocks of 16 bytes to be encrypted, as such ECB was not used (its security properties are weak and it is prone to cryptanalysis since there is a direct relationship between plaintext and ciphertext). Rather we used Output Feedback Mode.

IMAGE WHEN ENCRYPTED USING ECB MODE AND WHEN ENCRYPTED USING OTHER MODES

ORIGINAL IMAGE

ENCRYPTED USING ECB



ENCRYPTED USING OTHER MODES

SECRET SHARING SCHEME: MOTIVATION

Our motivation came from traditional symmetric ciphers, in particular the Hill Cipher. The Hill Cipher is vulnerable to known-plaintext attack. Given some plaintext-ciphertext pairs, if Eve knows the number of rows/columns of the key matrix, they can get the key matrix by inverting the plaintext matrix and multiplying with the ciphertext matrix. Once they have the key, they can break any ciphertext encrypted using that key matrix.

SECRET SHARING SCHEME: LOGIC

We have considered the threshold number of people to be k (k<=n, where n is the total number of people having shares). We then reshape the encrypted secret into a square matrix having say, m number of rows. Then we generate m coefficients (p_1,p_2,p_3, $p_{4...}p_m$) randomly and multiply them with the respective elements of each column. One combination consists of the m p(i)'s along with the m resultant coefficients(c(i)). Each person receives m/k such combinations if m is divisible by k and [m/k] +1 if m is not divisible by k (where [ ] is the box function). When k people come together, they get at least m rows between themselves and so can construct square matrices P and C from m p(i)'s and m c(i)'s respectively and get the encrypted secret as $P^{-1}C$.

EXAMPLE

Suppose, n=10 and k=5. We have a 500X500 square matrix (encrypted secret). So we give each person (500/5) = 100 combinations. So, when 5 people come together, they can construct 500X500 square matrices and get the encrypted secret.

Suppose, we have a 506X506 square matrix (encrypted secret). So we give each person [506/5]+1=102 combinations. So, when 5 people come together, they have 510 combinations and can choose 506 from them. If 4 people come together, they have 408 combinations, i.e., they still need 98 more combinations.

ASSUMPTIONS TO TAKE CARE OF

- **The encrypted secret key matrix should be invertible.**

  AES results in almost random bytes of ciphertext, so minimal probability of the determinant being zero (either row/column has all 0's or one row/column is a linear combination of some other rows/columns. We observed random square matrices of order 2X2, 3X3, and so on. For a 2x2 matrix, the probability of getting a non-invertible matrix was 0.011% which steadily decreased as the order increased, so much so that for 5X5 it was close to 0. As we are working with images, i.e., the order of the matrices is greater than 500 in almost every scenario, we can confidently say that the encrypted secret key matrix is invertible.

- **The matrix P should be invertible.**

  P consists of random coefficients and the order is same as that of the key matrix, so the above argument holds here as well.

```
import random
from scipy import linalg
import sys

wrong = 0
net = 0
n=100
while net<50000:
  net+=1
  data__ = []
  for i in range(n):
    eq=[int(256*random.random()) for _ in range(n)]
    data__.append(eq)

  try:
    linalg.inv(data__)
  except:
    wrong += 1

  sys.stdout.write(
                    "\r[%d wrong][%d net]"
                    % (
                        wrong,
                        net
                    )
                )
```

```
[0 wrong][50000 net]
```

```
import random
from scipy import linalg
import sys

wrong = 0
net = 0
n=500
while net<10000:
  net+=1
  data__ = []
  for i in range(n):
    eq=[int(256*random.random()) for _ in range(n)]
    data__.append(eq)

  try:
    linalg.inv(data__)
  except:
    wrong += 1

  sys.stdout.write(
                    "\r[%d wrong][%d net]"
                    % (
                        wrong,
                        net
                    )
                )
```

```
[0 wrong][10000 net]
```

WHY USE THIS?

1. **Secure:** The scheme has Information theoretic security as having less than the requisite number of shares of the secret provides no information about the secret.
2. **Minimal:** The size of each piece does not exceed the size of the original data.
3. **Extensible:** When k is kept fixed, shares can be dynamically added or deleted without affecting the other pieces, because computing new equations does not affect the currently computed equations. So, there is no strict limit that one has to know the value of n beforehand.
4. **Dynamic:** Security can be easily enhanced without changing the secret, but by changing the coefficient matrix and constructing new shares for the participants.

WHY USE THIS? (CONTINUED)

5. **Flexible:** In organizations where hierarchy is important, each participant can be assigned different numbers of shares according to their importance inside the organization. For instance, the president could unlock the safe alone, whereas 3 secretaries would be required to combine their shares to unlock the safe.
6. **Space efficient:** Each share is roughly twice the size of the secret divided by k. So, when k>2, the size of each share is less than that of the secret. As k increases, the size of each share decreases.
7. **Lazy computation of shares:** Shares can be generated as and when required, one does not need to store computed shares if for instance a person in the group is unavailable at the time or in case of cloud storage across different servers, a server which would need a share is down at the time.
8. One cannot gain information about the dimensions of the original image by looking at the shares, thus negating any attack which would use that.
9. **Anonymous:** The identities of the participants are not required for reconstruction.

HOW TO RECONSTRUCT ORIGINAL IMAGE?

The encrypted secret is obtained from the steps in the previous slides. We remove the padding of 1's which were added to make the square matrix. So, we make the list of bytes a multiple of 16 again. Then we use the decryption algorithm using the same key used for encryption (this key had been agreed upon beforehand, as was the initialization vector, IV) to get the original image.

IMPLEMENTATION: ENCRYPTION AND CONSTRUCTING THE KEY MATRIX

We use the Crypto library available in Python to encrypt the image.

```
cipher = AES.new(key, AES.MODE_OFB, iv)
ciphertext = cipher.encrypt(byte)
int_list = [x for x in bytearray(ciphertext)]
order = int(np.sqrt(len(int_list))) + 1
if len(int_list) < (order * order):
    int_list += [1] * (order * order - len(int_list))
key_matrix = np.array(int_list).reshape((order, order))
tra_key_mat=np.transpose(key_matrix)
```

```
if order%k==0:
    shares=int(order/k)
else:
    shares=int(order/k)+1
```

IMPLEMENTATION: CONSTRUCTING SHARES FOR EACH PERSON

```
def get_share(num_shares,matrix):
  shares_list=[]
  for _ in range(num_shares):
        eq=[int(256*random.random()) for _ in range(len(matrix))]
        shares_list.append(eq)
  val=[]
  for j in range(num_shares):
      p=shares_list[j]
      c= np.dot(p,matrix)
      val.append([c,p])
  return val
```

IMPLEMENTATION: RECONSTRUCTION END

```
def get_encoded_image(p_c_array):
  # assumption p_c_array is a multidimentional list of the format [[c1,p1],[c2,p2],[c3,p3]....]

  # print(numpy.shape(p_c_array))
  if(numpy.shape(p_c_array)[0] != numpy.shape(p_c_array)[2]):
    if(numpy.shape(p_c_array)[0] > numpy.shape(p_c_array)[2]):
      p_c_array= p_c_array[:numpy.shape(p_c_array)[2]]
    else:
      print("square array needed")
      return 0

  plain_text = []
  cipher_text = []
  for c_p in p_c_array:
    cipher_text.append(c_p[0])
    plain_text.append(c_p[1])

  plain_text = numpy.array(plain_text)
  cipher_text = numpy.array(cipher_text)

  try:
    p_inv = linalg.inv(plain_text)
  except:
    print("singular")
    return 0

  return numpy.dot(p_inv, cipher_text)# ,plain_text
```

IMPLEMENTATION: RECONSTRUCTION END (2)

Below, all_shr contains a list of the shares from k people.

```
key_=get_encoded_image(all_shr)
key_matrix = numpy.transpose(numpy.around(key_)).astype(int)
new_list = key_matrix.tolist()
if not len(new_list) % 16 == 0:
    length = 16 * (len(new_list) // 16)
    new_list = new_list[0:length]
plain = AES.new(key, AES.MODE_OFB, iv)
bb = bytes()
for num in new_list:
    bb = bb + bytes(num)
plaintext = plain.decrypt(bb)
```

LIMITATIONS AND SCOPE FOR IMPROVEMENT

- ★ **Verification of correctness of the retrieved shares during the reconstruction process:** A person may lie about their shares in order to gain access to the other shares. So, scheme is vulnerable to malicious insiders.
- ★ Can take time for very large images
- ★ In the share generating part modular arithmetic can be used for added diffusion.

APPLICATIONS

Traditional methods for encryption are ill-suited for simultaneously achieving high levels of confidentiality and reliability. This is because when storing the encryption key, one must choose between keeping a single copy of the key in one location for maximum secrecy, or keeping multiple copies of the key in different locations for greater reliability. Secret sharing schemes address this problem, and allow arbitrarily high levels of confidentiality and reliability to be achieved. It also allows the distributor of the secret to trust a group 'in aggregate'. Traditionally, giving a secret to a group for safekeeping would require that the distributor completely trust all members of the group. Secret sharing schemes allow the distributor to securely store the secret with the group even if not all members can be trusted all the time. So long as the number of traitors is never more than the critical number needed to reconstruct the secret, the secret is safe.

Secret sharing schemes are important in cloud computing environments. Thus a key can be distributed over many servers by a threshold secret sharing mechanism. The key is then reconstructed when needed. Secret sharing has also been suggested for sensor networks where the links are liable to be tapped by sending the data in shares which makes the task of the eavesdropper harder.

APPLICATIONS (2)

A secret sharing scheme can secure a secret over multiple servers and remain recoverable despite multiple server failures. Each share may be stored on a different server, but one can recover the secret even if several servers break down as long as they can recover at least *t* shares; however, crackers that break into one server would still not know the secret as long as fewer than *t* shares are stored on each server.

A dealer could send *t* shares, all of which are necessary to recover the original secret, to a single recipient. An attacker would have to intercept all *t* shares to recover the secret, a task which is more difficult than intercepting a single file, especially if the shares are sent using different media (e.g. some over the Internet, some mailed on CDs).

For large secrets, it may be more efficient to encrypt the secret and then distribute the key using secret sharing.

CONCLUSION

We set out to find a way to distribute shares of an image to a number of people such that the original image can be reconstructed using the shares of a threshold (k<=n) number of people. We used our scheme on multiple image files, making a number of shares and reconstructing the image from the shares of k people (k given as user input at the start). Thus, our scheme is secure (shares from <k people will not reveal information) and reconstructs the original image without any alterations.


Thank You!