Justin Ciocoi

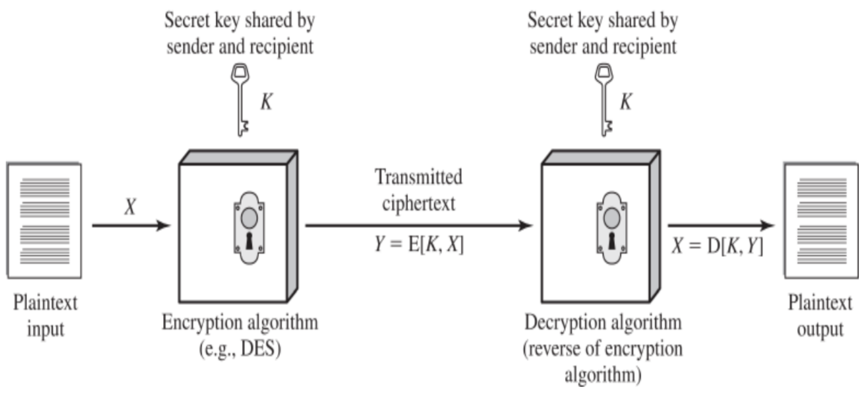
Feb. 6, 2024

CSCI 400 Textbook Notes

Chapter 2: Cryptographic Tools

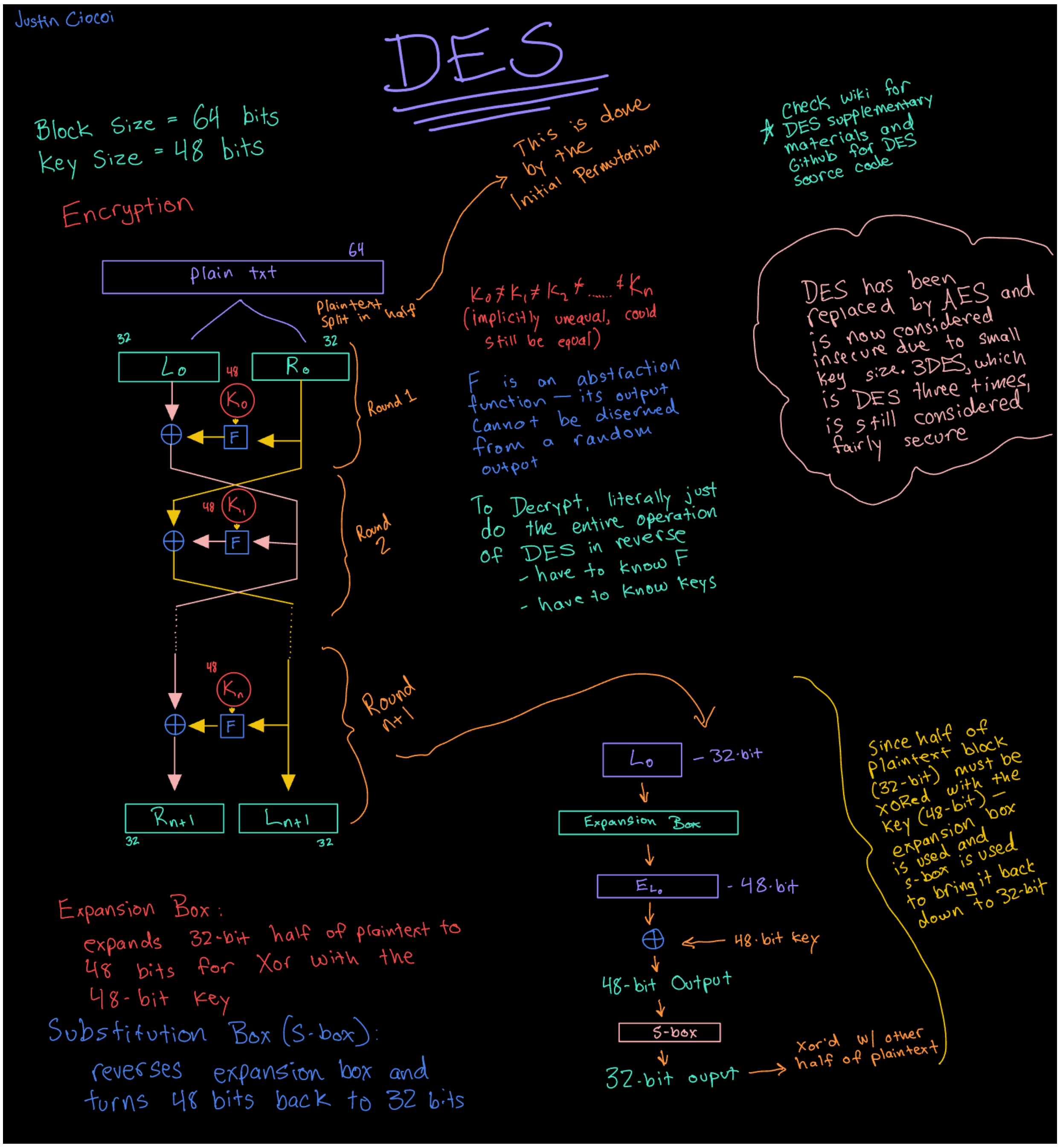
2.1: Confidentiality with Symmetric Encryption

- Universally, the technique which is used to provide confidentiality for transmitted or stored data is symmetric encryption
- The two most important algorithms, DES and AES, are block encryption algorithms and encrypt data in blocks of predefined size, whereas stream encryption algorithms encrypt information in a ore bit-by-bit manner
- **Symmetric Encryption**
    - Symmetric encryption is the most widely used category of encryption scheme and is illustrated in the below diagram

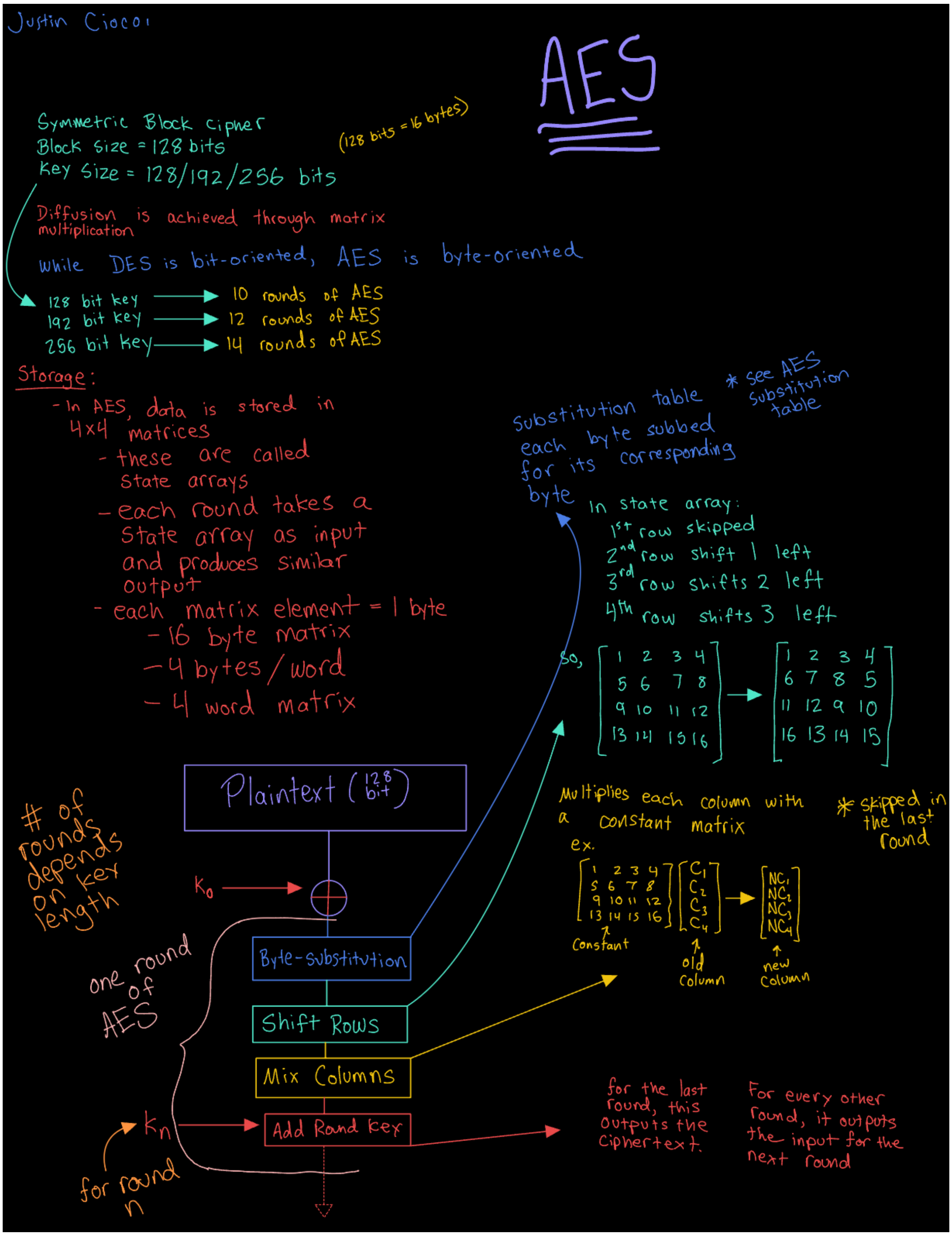

Secret key shared by sender and recipient
K
Plaintext input
X
Encryption algorithm (e.g., DES)
Transmitted ciphertext
Y = E[K, X]
Secret key shared by sender and recipient
K
Decryption algorithm (reverse of encryption algorithm)
X = D[K, Y]
Plaintext output

    - A symmetric encryption scheme has five ingredients
        - **Plaintext**, which is the original message or data that is fed into the algorithm as input
        - **Encryption Algorithm** which performs transformations and substitutions on the inputted plain text
        - **Secret Key** which is also input into the encryption algorithm and dictates how transformations and substitutions will happen

        - **Ciphertext**, which is the encrypted message which will be transmitted as is over communication lines
        - **Decryption Algorithm**, which allows only the intended recipient to decode the ciphertext back into the original plaintext
    - Generally, two requirements exist for utilizing a symmetric encryption scheme securely
        - A strong encryption algorithm is needed
        - Both the sender and receiver must have copies of the secret key and must keep the key secure
    - Two broad categories of attack exist for symmetric algorithms
        - The first of these is **Cryptanalysis**
            - Cryptanalysis focuses on patterns in ciphertext or in the logic used to generate ciphertext
            - It can also attack the nature of the security mechanism by exploiting certain mathematical properties, such as in a man-in-the-middle attack
        - The second is **Brute Force**
            - Brute-Forcing involves trying every possible copy of the secret key until you find a key that encrypts plaintext into ciphertext in the same way as the cipher
            - You must have at least one piece of ciphertext and its corresponding plaintext to conduct this attack
                - The longer the corresponding string of text that you have, the less likely it is that you will find an incorrect key that correctly encrypts your string
- **Symmetric Block Encryption Algorithms**
    - Block ciphers are the most commonly used symmetric encryption algorithms, and they encrypt information in blocks of predetermined size
    - **DES**, is an older block encryption algorithm that has since become deprecated due to its small key space


Justin Ciocoi
DES
Block Size = 64 bits
Key Size = 48 bits
Encryption
This is done by the Initial Permutation
★ Check wiki for DES supplementary materials and Github for DES source code
64
plain txt
Plaintext split in half
32
L₀
48
R₀
32
K₀
F
Round 1
48
K₁
F
Round 2
48
Kₙ
F
Round n+1
Rₙ₊₁
Lₙ₊₁
32
32
K₀ ≠ K₁ ≠ K₂ ≠ ...... ≠ Kₙ
(implicitly unequal, could still be equal)
F is an abstraction function — its output cannot be diserned from a random output
To Decrypt, literally just do the entire operation of DES in reverse
- have to know F
- have to know keys
DES has been replaced by AES and is now considered insecure due to small key size. 3DES, which is DES three times, is still considered fairly secure
L₀
- 32-bit
Expansion Box
E_L₀
- 48-bit
48-bit key
48-bit Output
S-box
32-bit ouput
Xor'd w/ other half of plaintext
Since half of plaintext block (32-bit) must be XORed with the key (48-bit) — expansion box is used and s-box is used to bring it back down to 32-bit
Expansion Box:
expands 32-bit half of plaintext to 48 bits for Xor with the 48-bit key
Substitution Box (S-box):
reverses expansion box and turns 48 bits back to 32 bits

- **Triple DES, or 3DES,** is three consecutive applications of DES and is still considered secure, but it leaves a lot to be desired the field of efficiency
- **AES** is a newer encryption algorithm by the name of Rijndael which arose as an algorithm to replace DES



Justin Ciocoi

AES

Symmetric Block cipher
Block size = 128 bits (128 bits = 16 bytes)
Key size = 128/192/256 bits

Diffusion is achieved through matrix multiplication

While DES is bit-oriented, AES is byte-oriented

- 128 bit key → 10 rounds of AES
- 192 bit key → 12 rounds of AES
- 256 bit key → 14 rounds of AES

Storage:

- In AES, data is stored in 4x4 matrices
  - these are called state arrays
  - each round takes a state array as input and produces similar output
- each matrix element = 1 byte
  - 16 byte matrix
  - 4 bytes / word
  - 4 word matrix

Substitution table each byte subbed for its corresponding byte

* see AES substitution table

In state array:
1st row skipped
2nd row shift 1 left
3rd row shifts 2 left
4th row shifts 3 left

so,

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 2 & 3 & 4 \\ 6 & 7 & 8 & 5 \\ 11 & 12 & 9 & 10 \\ 16 & 13 & 14 & 15 \end{bmatrix}$$

Multiplies each column with a constant matrix

* skipped in the last round

ex.

$$\underbrace{\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}}_{\text{Constant}} \underbrace{\begin{bmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \end{bmatrix}}_{\text{old Column}} \rightarrow \underbrace{\begin{bmatrix} NC_1 \\ NC_2 \\ NC_3 \\ NC_4 \end{bmatrix}}_{\text{new Column}}$$

# of rounds depends on key length

Plaintext (128 bit)

K_0

one round of AES

Byte-Substitution

Shift Rows

Mix Columns

Add Round Key

K_n

for round n

for the last round, this outputs the ciphertext.

For every other round, it outputs the input for the next round

2.2: Message Authentications and Hash Functions

- Encryptions main function is to prevent passive attacks by making eavesdropping useless
- However, active attacks require different measures to prevent and protection against such attacks is known as message or data authentication

- A message or file is said to be *authentic* when it is genuine and came from the source that it claims
- This has two important parts
  - Verification that the message's contents have not been altered
  - Verification that the source of the message is authentic
- **Authentication Using Symmetric Encryption**
  - Through the use of the secret key only known by the sender and recipient, symmetric encryption can assure us of the authenticity of a message's origin
  - Additionally, if error-checking is implemented, symmetric algorithms can also verify whether a message has been altered in transit
  - However, symmetric algorithms alone are not enough for authentication functions since reordering of transmitted blocks could not be detected
- **Message Authentication Without Message Encryption**
  - There exist a number of scenarios in which it would be desirable to verify the source of a message, but not to encrypt the message itself
    - If the same message is broadcast to a number of destinations
    - If one side has a heavy load and is incapable of spending time decrypting all incoming messages
    - Authenticating a computer program without needing to decrypt it upon each use
  - **Message Authentication Code (MAC)**
    - One technique used to authenticate messages is the use of a secret key to generate a small piece of data known as a message authentication code, or MAC, which will be appended to the message
    - This technique assumes that between the sender and receiver a common shared key, K_{AB}, exists
    - A use will calculate the message authentication code as a complex function where the inputs are the message itself and the shared secret key

$$MAC_M = F(K_{AB}, M)$$

        - The receiver will then conduct the same calculations and compare their generated MAC with the MAC_M appended to the message
        - If the codes are matching,
            - The receiver can be assured that the message has not been altered in transit since the message itself is part of the input for MAC generation
            - The receiver can be assured that the message is coming from the stated sender since nobody else is in possession of the secret key used in MAC generation
            - If the message includes a sequence number, then the receiver can be assured that the message is in its proper sequence
    - **One-Way Hash Function**
        - The one-way hash function serves as an alternative for message authentication codes
        - A hash function accepts a variable-size message, M, as input and produces a fixed-size digest, $H(M)$ as output
        - Unlike the MAC a hash function does not use a secret key as part of its input
- **Secure Hash Functions**
    - In order to be useful in the field of message authentication, a hash function must have the following properties
        - The function can be applied to a block of data of any size
        - The function produces a fixed-length output
        - The function is relatively easy to compute for any given input parameter
        - For any given digest, h, it is computationally infeasible to find an x such that $H(x) = h$
            - This property is referred to as *pre-image resistance*
        - For any given block x it is computationally infeasible to find $x \neq y$ with $H(y) = H(x)$
            - This property is referred to as *second pre-image resistance*

        - It is computationally infeasible to find any pair (x, y) such that $H(x) = H(y)$
            - This property is referred to as *collision resistance*
    - SHA is the most widely used family of secure hash functions and has many functions the most recent of which is SHA-3
- **Other Applications of Hash Functions**
    - Passwords can be stored as hashes rather than plaintext as it minimizes the damage that will be done in the event of a breach including user password data
    - By computing the hash of a file at closing and recomputing before opening, intrusion can be detected by ensuring that the contents of the file have not changed unexpectedly

2.3: Public-Key Encryption

- The biggest difference that public-key encryption has when compared to aforementioned encryption mechanisms is the fact that it is *asymmetric*, involving the use of two separate keys
- A public-key encryption algorithms has 6 key parts
    - **Plaintext**, which is the input into the encryption function
    - **Encryption Algorithm** which performs various substitutions and transformations on the plaintext
    - **Public/Private Keys**, which influence the transformations made by the encryption algorithm as well as one being used for encryption and *the other* being used for decryption
    - **Ciphertext**, which is the encrypted text
    - **Decryption Algorithm** which allows only the intended recipient to decode the received ciphertext back into the original plaintext
- The essential steps for setting up such a scheme are as follows
    - Each user will generate a pair of keys to be used for encryption and decryption of messages
    - Each user places one of two keys on a public register and keeps the other key secret

        - Each user also maintains a collection of public keys which have been received from others
    - Each user will use the intended recipient's public key when wanting to encrypt a message that will be sent to that person
    - A user will then use their private key in order to decrypt their message, providing confidentiality since only the intended recipient can decode the ciphertext back into the original plaintext
- **Requirements for Public-Key Cryptosystems**
    - It is computationally easy for a party, B, to generate a key pair $(pub_B, priv_B)$
    - It is computationally easy for a sender, A, knowing the public key and message, M, to be encrypted, to generate the corresponding ciphertext using

      $$C = E(pub_B, M)$$

    - It is computationally easy for the receiver, B, to decrypt the ciphertext using the private key in order to recover the original message using

      $$M = D(priv_B, C) = D[priv_B, E(pub_B, M)]$$

    - It is computationally infeasible for an opponent who knows the public key pub_B to determine the private key $priv_B$
    - It is computationally infeasible for an opponent who knows the public key, pub_B and ciphertext, C, to recover the original message M
    - Either related key can be used for encryption provided that the other will be used for decryption
- **Asymmetric Encryption Algorithms**
    - **RSA**
        - RSA has been the most widely used and popular public-key encryption scheme since its invention in 1977
        - RSA currently requires a key size of 1024-bit to be considered secure in the modern age
    - **Diffie-Hellman Key Exchange**

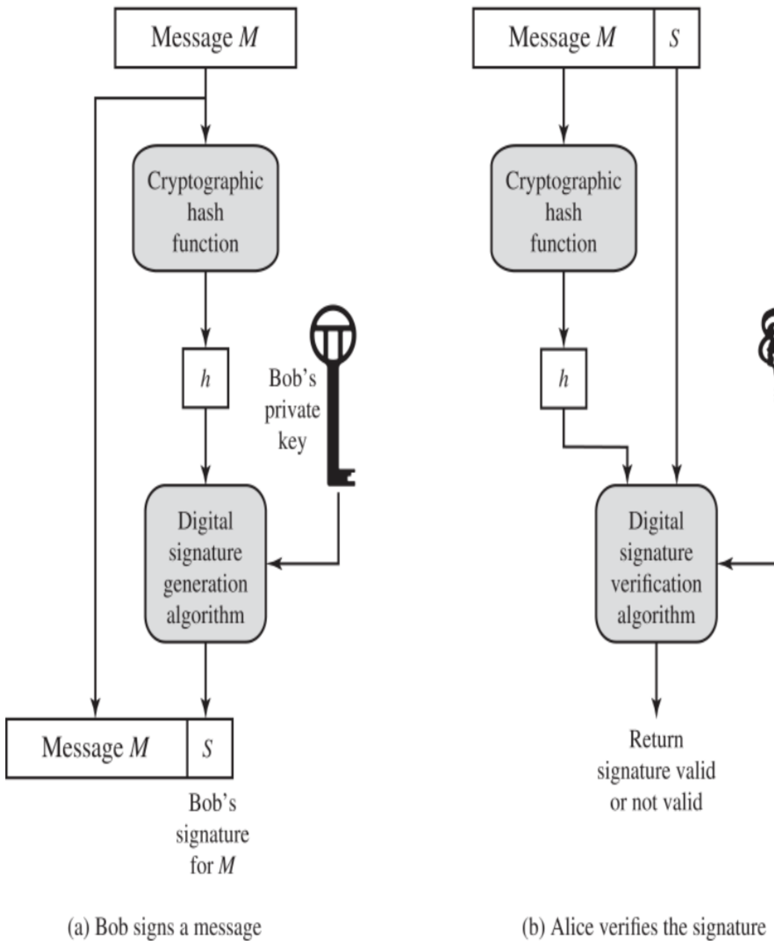
        - This algorithm exists primarily to allow pairs of users to securely share a secret key over a secure channel using public-key cryptography in order to establish more efficient symmetric cryptography between the two of them
    - **Digital Signature Standard**
        - This standard makes use of SHA-1 in order to provide message authentication mechanisms while not providing any form of encryption
    - **Elliptic Curve Cryptography**
        - Elliptic Curve Cryptography reduces the key space needed for RSA by being as secure with a smaller key size due to the lack of an index calculus attack on elliptic curve groups used in ECC
        - It is still somewhat new, however, and confidence in ECC is not yet as high as it is in RSA

2.4: Digital Signatures and Key Management

- With regard to the management and distribution of keys in a public-key environment, three distinct properties of public-key encryption are used
    - The secure distribution of public keys
    - The use of public-key encryption to distribute secret keys
    - The use of public-key encryption to create temporary keys for message encryption
- **Digital Signatures**
    - NIST defines a digital signature as follows:

      The result of a cryptographic transformation of data that, when properly implemented, provides a mechanism for verifying origin authentication, data integrity, and signatory non-repudiation
    - FIPS 186-4 specifies the use of one of three digital signature algorithms
        - **Digital Signature Algorithm (DSA)** which is the original NIST-approved digital signature algorithm
        - **RSA Digital Signature Algorithm** which is based on the public-key cryptography of RSA

        - **Elliptic Curve Digital Signature Algorithm (ECDSA)** which is based on the public-key cryptography of elliptic curves
    - Here, we can see an example of Bob sending a message to Alice along with a digital signature used to verify message origin


Message M
Cryptographic hash function
h
Bob's private key
Digital signature generation algorithm
Message M
S
Bob's signature for M
(a) Bob signs a message
Message M
S
Cryptographic hash function
h
Bob's public key
Digital signature verification algorithm
Return signature valid or not valid
(b) Alice verifies the signature

    - The digital signature does not provide confidentiality, and is thus still susceptible to eavesdropping attacks

- **Public-Key Certificates**
    - One issue that exists in public key cryptography stems from the inherent property that there exists a publicly available key used for encryption
    - An adversary could easily forge a public announcement by using another person's publicly broadcast public key
    - The solution to this problem comes in the use of public-key certificates
    - Generally, public-key certificates consist of a public key and user ID of the key owner, with the whole block being signed by a trusted third party such as a reputable

certificate authority (CA)

  - The key steps of using public-key certificates can be summed up as follows
    - Client creates a pair of keys, one public and one private
    - Client prepares an unsigned certificate that includes userID and the user's public key
    - Client provides the unsigned certificate to a CA in a secure manner
    - CA creates a signature as follows
      - CA uses a hash function to calculate the hash code of the unsigned certificate
      - CA generates digital signature using the CA's private key and a signature generation algorithm
    - CA attaches the signature to the unsigned certificate to create a signed certificate
    - CA returns the signed certificate to the client
    - Client may now provide the signed certificate to any other user
    - Any other user may verify the validity of that certificate as follows
      - User calculates hash code of the certificate without the signature and then uses the CA's public key as well as a signature verification algorithm to determine the validity of the included digital signature

2.5: Random and Pseudorandom Numbers

- A number of different network security algorithms make use of random numbers, such as the following
  - Generation of keys for RSA
  - Generation of a stream key for symmetric stream cipher
  - Generation of a symmetric key for use as a temporary session key or in creating a digital envelope
  - In a number of key distribution scenarios, random numbers are used in handshake scenarios to prevent replay attacks
  - Session key generation

- A sequence of numbers has two properties that make it compatible for use in cryptography; namely *randomness* and *unpredictability*
- Randomness can be assessed using the following criteria
    - Uniform distribution of numbers, meaning that the frequency of all digits should be about the same
    - Independence, meaning that no one value in the random sequence can be inferred from any of the others
- While true randomness is sometimes desired, the bar for achieving this on a computer system is far higher than it is to achieve a pseudorandom number generator whose unpredictability makes it suitable for use in cryptographic endeavors

2.6: Practical Application - Encryption of Stored Data

- One of the most important aspects of securing a computer system is the protection of stored data
- Such security mechanisms include access control, intrusion detection, and intrusion prevention schemes
- Because data is at risk not only during transmission, it is also desirable to encrypt data at rest
- A simple approach would be to use a commercially available encryption package such as Pretty Good Privacy (PGP)