CISC 3325 - INFORMATION SECURITY

Cryptographic Tools

Topics for today

- Cryptography (cont.):
  - Problems encryption is designed to solve
  - Encryption tools categories, strengths, weaknesses
    - applications of each
  - Certificates and certificate authorities

CRYPTOGRAPHY

https://www.tripwire.com/state-of-security/security-data-protection/cryptography/ordinary-people-need-cryptography/

Motivation

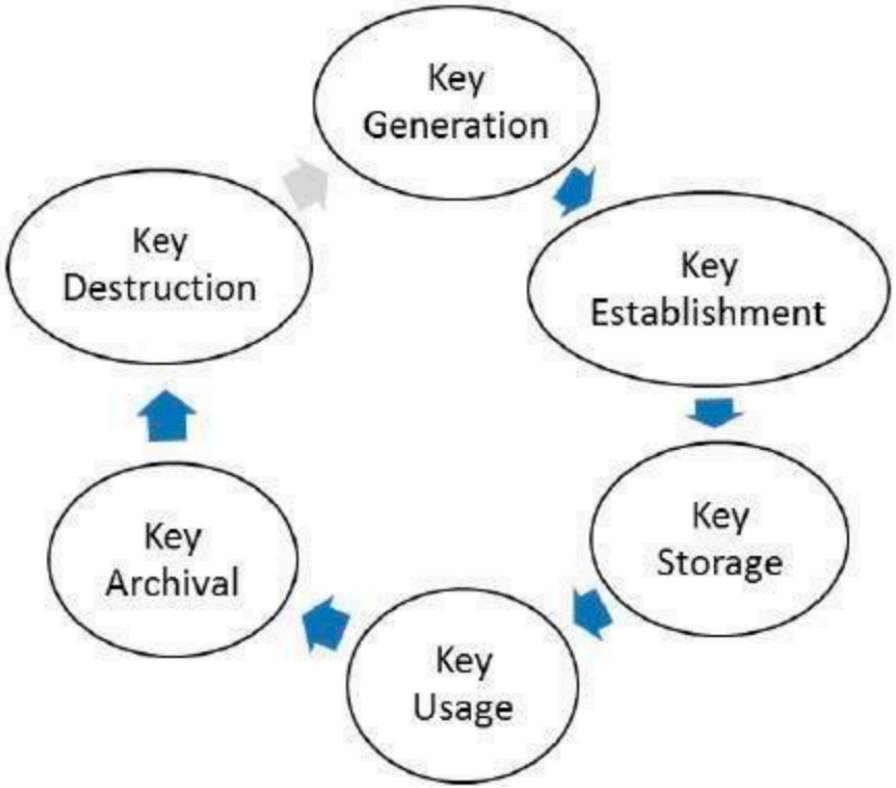
- “The basics of asymmetric cryptography are fundamental concepts that any member of society who wants to understand how the world works, or could work, needs to understand. They are as fundamental as the basics of supply and demand and monetary inflation”

Phil Libin, Evernote

Cryptography - review

- Encryption provides secrecy
  - Confidentiality
- Symmetric cryptography requires a secret key
  - Need to be shared ahead of communication
    - Stored securely
  - Keys management a critical part of cryptography
    - One of the weakest points
      - Need to be refreshed, changed upon demand, destroyed

Key Management



https://www.tutorialspoint.com/cryptography/public_key_infrastructure.htm

Key Exchange

- ***Symmetric*** algorithms use one key, which works for both encryption and decryption
  - Usually, the decryption algorithm is closely related to the encryption one
    - running the encryption in reverse
- Both parties share a secret key
  - they can both encrypt sent information as well as decrypt information from the other

Key Exchange

- As long as the key remains secret, the system also provides authenticity
  - Authenticity is ensured because only the legitimate sender can produce a message that will decrypt properly
    - with the shared key

Key Exchange

- How do two users A and B obtain their shared secret key?
  - And only A and B can use that key for their encrypted communications
  - If A wants to share encrypted communication with another user C, A and C need a different shared secret key.
- Managing keys is the major difficulty in using symmetric encryption

Key Exchange

- If we have *n* users, how many keys do we need?
  - *n* users who want to communicate in pairs need n * (n – 1)/2 keys
    - $O(N^2)$
- The number of keys needed increases at a rate proportional to the square of the number of users
- Symmetric encryption systems require a means of key distribution
- How do we solve this?

Key Exchange

- One possibility:
  - Using a Third Trusted Party (TTP)
- In this case, each party will have one shared secret key with the third trusted party
  - $K_a, K_b, k_c, etc.$

Key Exchange with TTP

- What A and B want to exchange a secret key:
  - TTP picks a new secret key $K_{a,b}$
  - TTP encrypts $K_{a,b}$ with K_a and sends it to A
  - TTP encrypts $K_{a,b}$ with K_b and sends it to B
  - A and B each decrypts their respective keys
    - Using their pre-shared secret key

Key Exchange with TTP

- Disadvantage: TTP always has to be available online to perform key exchange
- Is there another method?
  - Yes, using asymmetric encryption

Key Exchange

- ***Asymmetric or public key*** systems typically have precisely matched pairs of keys.
- The keys are produced together
  - One may be derived mathematically from the other
  - Process computes both keys as a set
- Asymmetric systems good for key management
  - public key may be emailed or post it in a public directory
- Only the corresponding private key can decrypt what has been encrypted with the public key

Key Exchange with Assymetric Cryptography

- Alice chooses a pair of public and private keys K_{pb} and k_{pr}
- Alice distributes K_{pb} public key to all parties
- Bob chooses a secret key $K_{b,a}$ and encrypts it with K_{pb}
  - Sends $E(K_{b,a}, K_{pb})$ to Alice
- Alice uses her secret key K_{pr} to decrypt $K_{b,a}$
- Alice and Bob now share a secret key $K_{b,a}$

Cryptography - review

- Encryption provides secrecy
  - Confidentiality
- Symmetric cryptography requires a secret key
  - Need to be shared ahead of communication
    - Stored securely
  - Keys management a critical part of cryptography
- Asymmetric cryptography uses one public and one private key
  - Much slower than symmetric cryptography

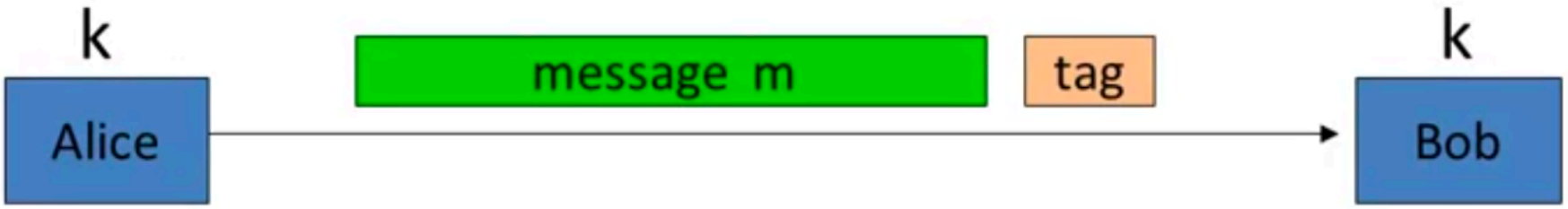
Why do we use cryptography?

- Confidentiality:
  - Achieved through encryption
    - Block ciphers, secret key encryption, public key encryption, etc.
- Data Integrity:
  - Hash Function
  - Message Authentication Tools (MACs)
  - Digital Signatures

Message Integrity

- Goal: provide integrity
  - Not confidentiality
- Real-world examples:
  - Operating system files on disk
    - Not secret, but integrity crucial
  - Protecting public binaries/mirror repository on disk
  - Protecting ads on web pages

Message Integrity (MAC)



Message Integrity (MAC)

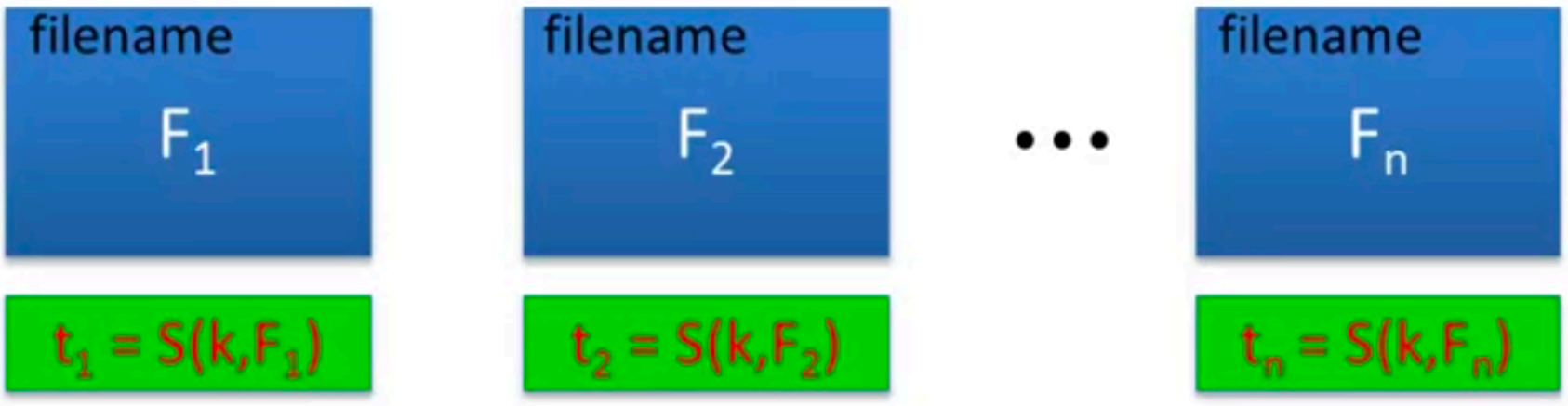
- Alice:
  - Generates tag:
    - Tag = f(K,m)
  - Appends tag to message
  - Sends to Bob
- Bob:
  - Verifies tag:
    - Verif(K,m,tag) = Yes/No
  - If verification = Yes, message accepted

Message Integrity (MAC)

- Shared secret key is needed
  - Otherwise, an attacker may be able to modify message and generate its own tag
- What if an attacker can find another message m_2 such at
  - S(k,m_2) = S(k,m_1)
  - Is the MAC secure?
    - No, the MAC is broken

Message Integrity (MAC)

- Example: protecting file systems
  - Suppose at the operating system install time, system computes tags for each system file
    - K is derived from the user's password
  - Store tag together with the file
    - K is not saved



https://www.coursera.org/learn/crypto/lecture/iVGR5/message-authentication-codes

Message Integrity (MAC)

- Later, a virus infects system and modifies system files
- User reboots into clean OS
  - Such as on a USB stick
  - User will supply his password
- Secure MAC will be computed
- All modified files on the system will be detected

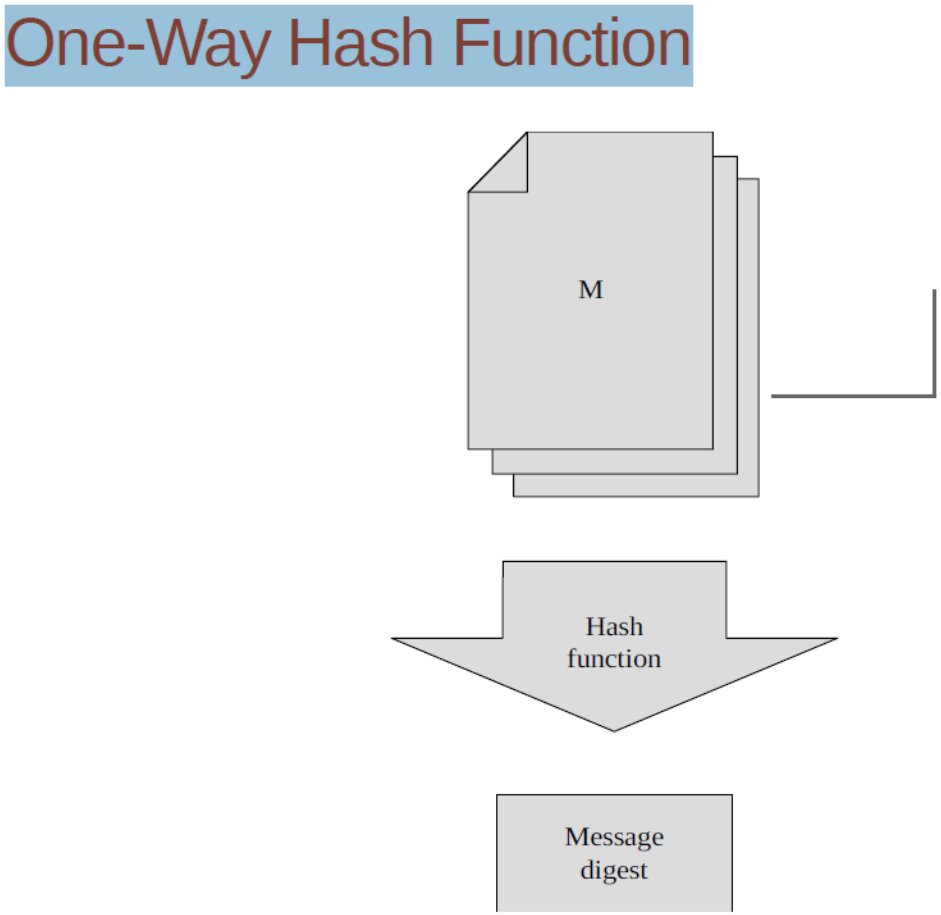
Message Integrity (MAC)

- How are MAC created?
  - From collision-resistant hash functions

Hash Codes

- We want to know that a file has not been tampered with
  - Similar to a seal on an envelope
- How can we use cryptography for that?
  - Use cryptography to create a ***hash*** of the data
    - Other options include checksums or message digests

One-Way Hash Function

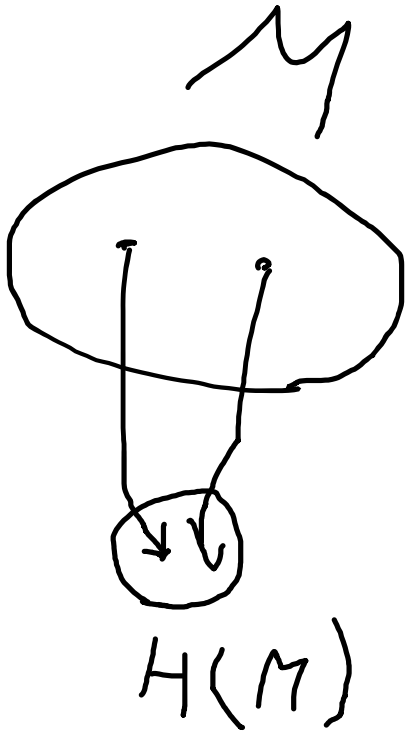


Hash Functions

- A hash function h maps a plaintext x to a fixed-length value $x = h(P)$ called hash value or digest of P
  - A collision is a pair of plaintexts P and Q that map to the same hash value
    - $h(P) = h(Q)$ and $P \neq Q$

Hash Functions

- Hash is smaller than the original message
- Message space significantly larger than hash space
  - Therefore, collisions are unavoidable



Hash Functions

- A hash function h maps a plaintext x to a fixed-length value x = h(P) called hash value or digest of P
  - A collision is a pair of plaintexts P and Q that map to the same hash value
    - $h(P) = h(Q)$ and $P \neq Q$
    - Collisions are unavoidable
    - However, it should be hard for an attacker to find a collision
      - In this case, the function is “collision resistant”

Hash Functions

- The computation of the hash function should take time proportional to the length of the input plaintext
  - For efficiency

Birthday Attack

- The brute-force birthday attack aims at finding a collision for a hash function h
  - Randomly generate a sequence of plaintexts $X_1, X_2, X_3, \ldots$
  - For each X_i compute $y_i = h(X_i)$ and test whether $y_i = y_j$ for some $j < i$
  - Stop as soon as a collision has been found

Birthday Attack

- If there are m possible hash values, the probability that the i-th plaintext does not collide with any of the previous i -1 plaintexts is

$$1 - \frac{i-1}{m}$$

Birthday Attack

- Probability F_k that the attack fails (no collisions) after k plaintexts is

$$F_k = (1 - 1/m)\,(1 - 2/m)\,(1 - 3/m)\,\ldots\,(1 - (k - 1)/m)$$

- Using the standard approximation $1 - x \approx e^{-x}$

$$F_k \approx e^{-(1/m + 2/m + 3/m + \ldots + (k-1)/m)} = e^{-k(k-1)/2m}$$

- The attack succeeds/fails with probability ½ when $F_k = ½$, that is,

$$e^{-k(k-1)/2m} = ½$$

$$k \approx 1.17\, m^{½}$$

  - We conclude that a hash function with b-bit values provides about b/2 bits of security

Birthday Attack

- Probability that 2 people were not born on same day of the year = 30/31
- The probability that the third person was not both on either of those days = 29/31
- For 11 people, we get:
  - P(no collusion) = $\frac{30*29*28*\cdots*21}{31^{10}}$ = 0.13 = 13%
  - P(collusion) = 87%

Birthday Attack

- Notice: Once we know that the first ten do not collide, the probability that the eleventh one will collide is 10/31
  - 10 times larger than the first, which was 1/31
- So each person we add has significantly larger probability of collusion

Birthday Paradox

- Birthday Paradox

Secure Hash Algorithm (SHA)

- Developed by NSA and approved as a federal standard by NIST
- SHA-0 and SHA-1 (1993)
  - 160-bits
  - Considered insecure
  - Still found in legacy applications

Secure Hash Algorithm (SHA)

- SHA-2 family (2002)
  - 256 bits (SHA-256) or 512 bits (SHA-512)
  - Still considered secure despite published attack techniques
- SHA-3 (2015)
  - More complex and secure hash algorithm
  - Faster than SHA-1 and SHA=2

Secure Hash Algorithm (SHA)

- Older hash functions include MD4, MD5
  - Significant vulnerabilities found
  - Should never be used
- Only SHA-2 AND SHA-3 should be used

Message Authentication Code (MAC)

- A.K.A. Cryptographic Checksum
  - Cryptographic function that produces checksum.
- A digest function using a cryptographic key
  - Key is presumably known only to the originator and the proper recipient of the data
- The attacker does not have a key with which to recompute the checksum
  - => Checksum important for data modification detection

Cryptographic Checksum

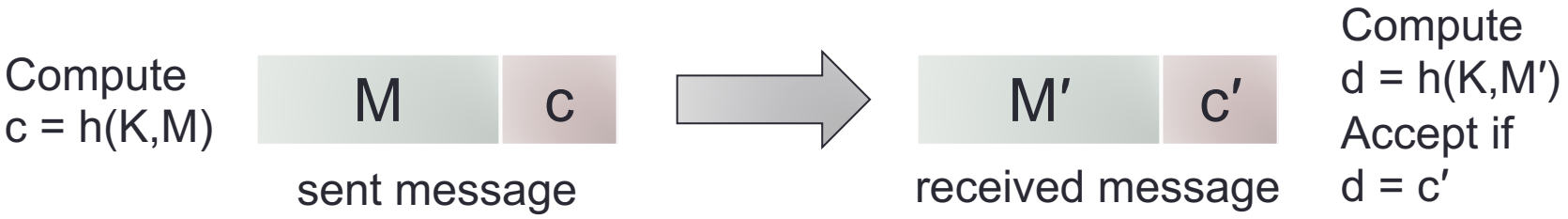
- Major uses of cryptographic checksums:
  - Code-tamper protection
    - Detect changes in system files
  - Message integrity protection in transit
    - Calculate checksum on received data and compare to sent values

Cryptographic Checksum

- AES can be used as a cryptographic checksum algorithm
  - However, simpler algorithms can be used for less sensitive data
  - SHA (Secure Hash Algorithm) defined by U.S. gov.

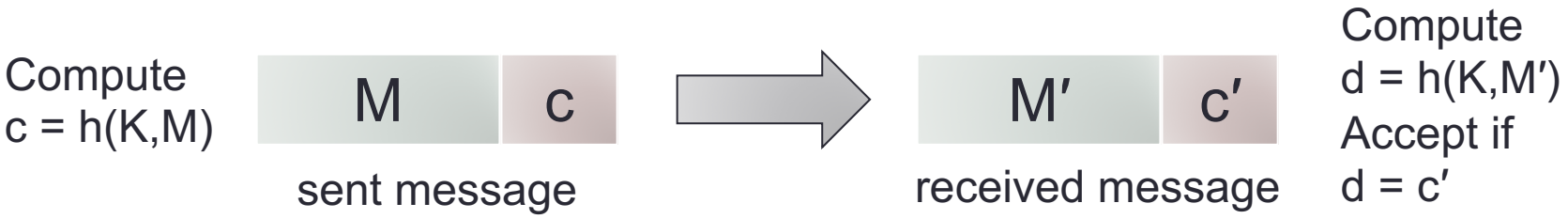
Message Authentication Code (MAC)

- Cryptographic hash function h(K,M) with two inputs:
  - Secret key K
  - Message M



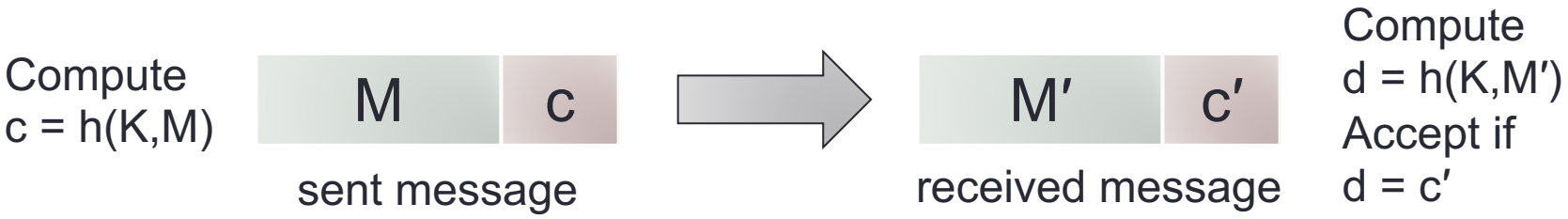
Message Authentication Code (MAC)

- Message integrity with MAC
  - Sequence of messages transmitted over insecure channel
  - Secret key K shared by sender and recipient
  - Sender computes MAC c = h(K,M) and transmits it along with message M
  - Receiver recomputes MAC from received message and compares it with received MAC



Message Authentication Code (MAC)

- Message integrity with MAC (cont.)
  - Attacker cannot compute correct MAC for a forged message
  - More efficient than signing each message
  - Secret key can be sent in a separate encrypted and signed message



HMAC

- Building a MAC from a cryptographic hash function is not immediate
- The following MAC constructions are insecure:
  - h(K ‖ M)
  - h(M ‖ K)
  - h(K ‖ M ‖ K)
    - Because of the iterative construction of standard hash functions,

HMAC

- HMAC provides a secure construction:
  - $h(K \oplus A \| h(K \oplus B \| M))$
  - A and B are constants
  - Internet standard used, e.g., in IPSEC
  - HMAC security is the same as that of the underlying cryptographic hash function

HMAC

- HMAC

Digital Signatures

- Three purposes:
  - ***Authentication***: message was created and sent by claimed sender.
  - ***Non-repudiation***: sender can not deny having sent the message later
  - ***Integrity***: ensures message not altered in transit

Digital Signatures

- Commonly used in:
  - Financial transactions
  - Software distribution
  - Other cases where forgery detection is needed
  - Popular in email applications

Digital Signatures

- Tool to demonstrate authenticity
  - similar to a paper signature
- A way by which a person or organization can affix a bit pattern to a file
  - implies confirmation,
  - pertains to that file only
  - cannot be forged

Digital Signatures

- A digital signature often uses asymmetric or public key cryptography
  - public key cryptographic protocols involve several sequences of messages and replies
    - Time consuming if both parties are not immediately available
  - In this situation, a technique that could authenticate a party even if it is inactive would be useful
    - Similar to a paper signature

Paper Signatures

- Traditional properties of a check signature:
  - A check is a *tangible object*
    - authorizing a financial transaction.
  - The check signature confirms authenticity
    - (presumably) only legitimate signer can produce that signature.
  - In the case of an alleged forgery, a third party can be called in to judge authenticity.
  - Once a check is cashed, it is canceled
    - it cannot be reused.
  - The paper check is not alterable.
    - most forms of alteration are easily detected.

Digital Signatures

- Properties required of digital signatures:
  - It must be unforgeable
    - If person S signs message M with signature Sig(S,M), no one else can produce the pair [M,Sig(S,M)].
  - It must be authentic
    - If a person R receives the pair [M, Sig(S,M)] purportedly from S, R can check that the signature is really from S.
    - Only S could have created this signature
      - the signature is firmly attached to M.

Digital Signatures

- Desired Properties:
  - It is not alterable
    - After being transmitted, M cannot be changed
      - By either S, R, or an interceptor.
  - It is not reusable
    - A previous message presented again will be instantly detected by R

Digital Signatures

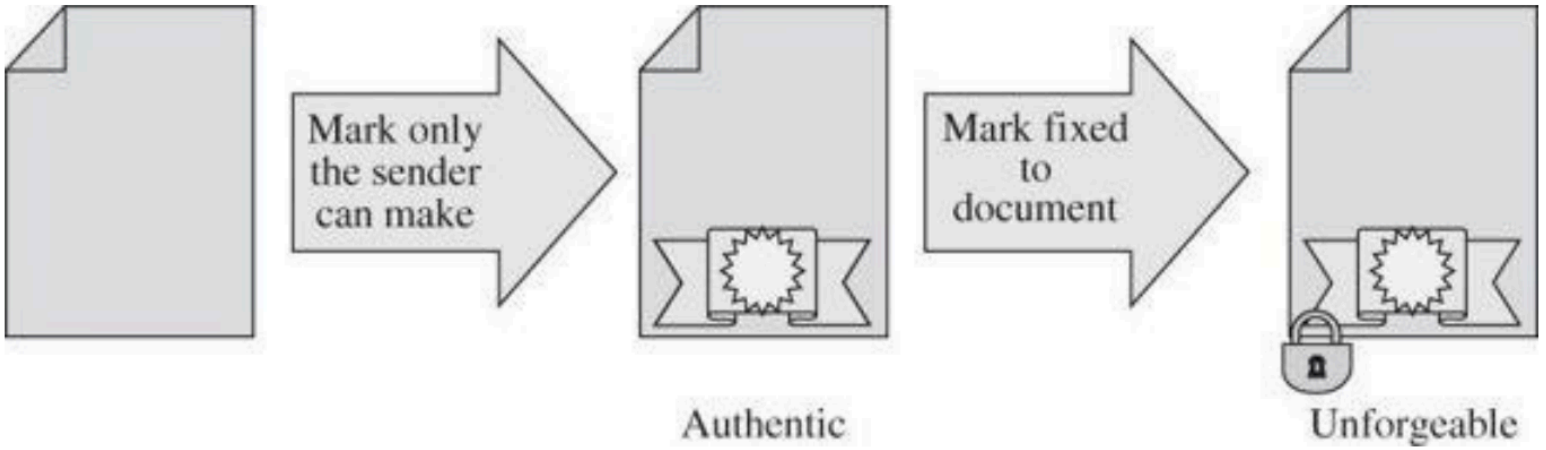
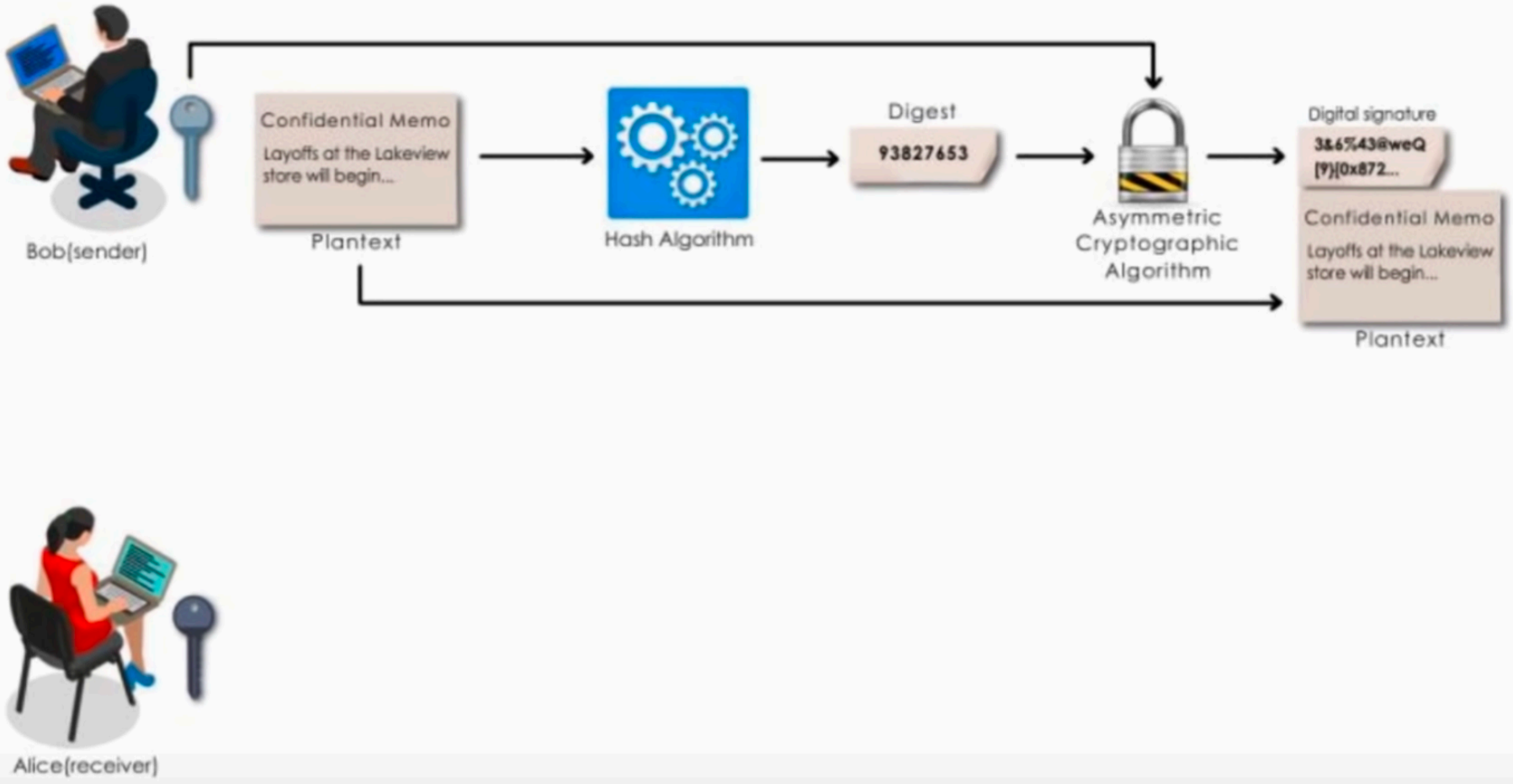


FIGURE 2-26 Digital Signature Requirements

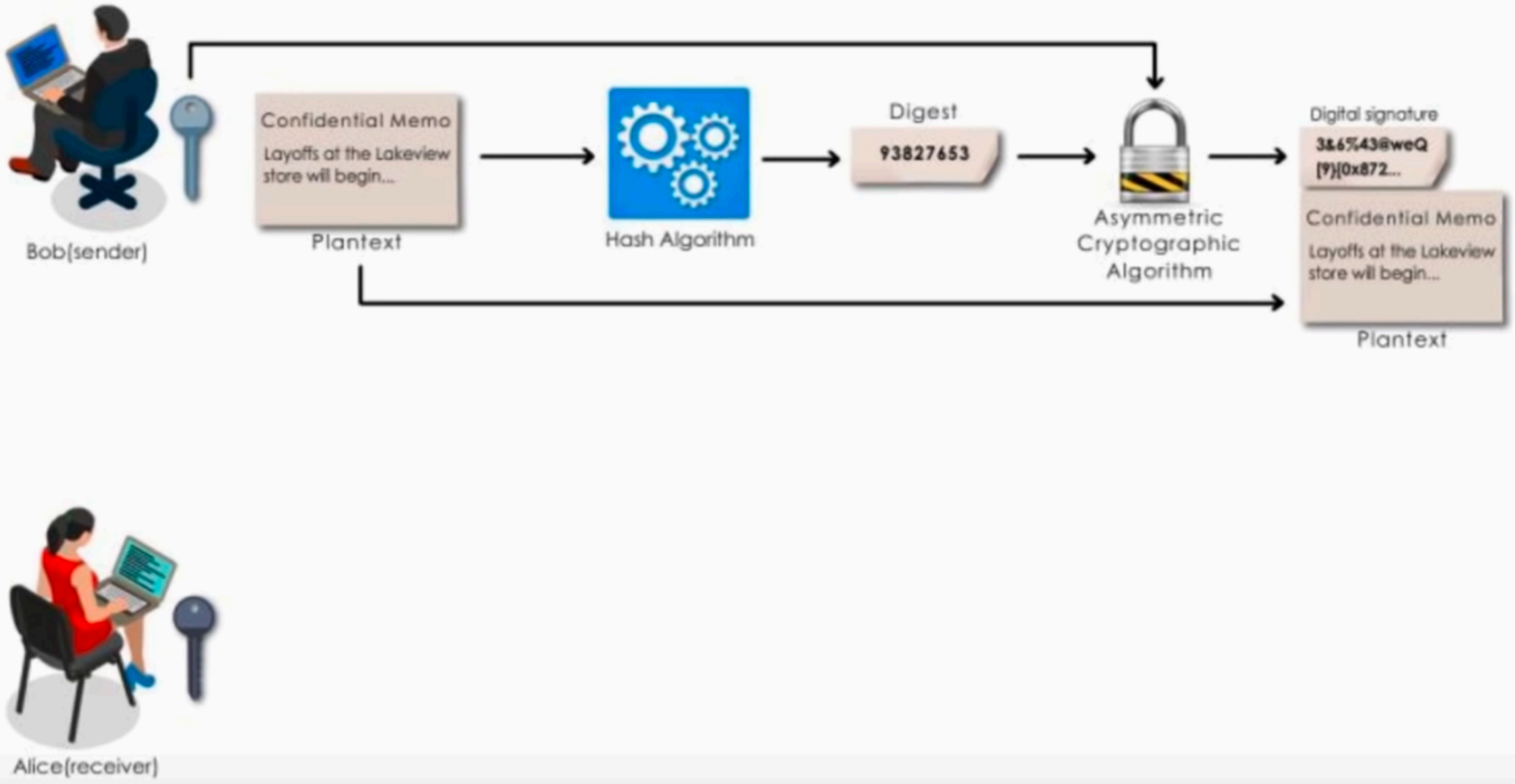
Digital Signature



Digital Signature

- Bob wants to sign a document
  - Bob generates a public and private key
    - Sends public key to Alice
  - Bob hashes the message to get a digest
  - Bob encrypts the digest using his private key
    - The encrypted digest is the digital signature for the memo
  - Bpb sends the digital signature and memo to Alice

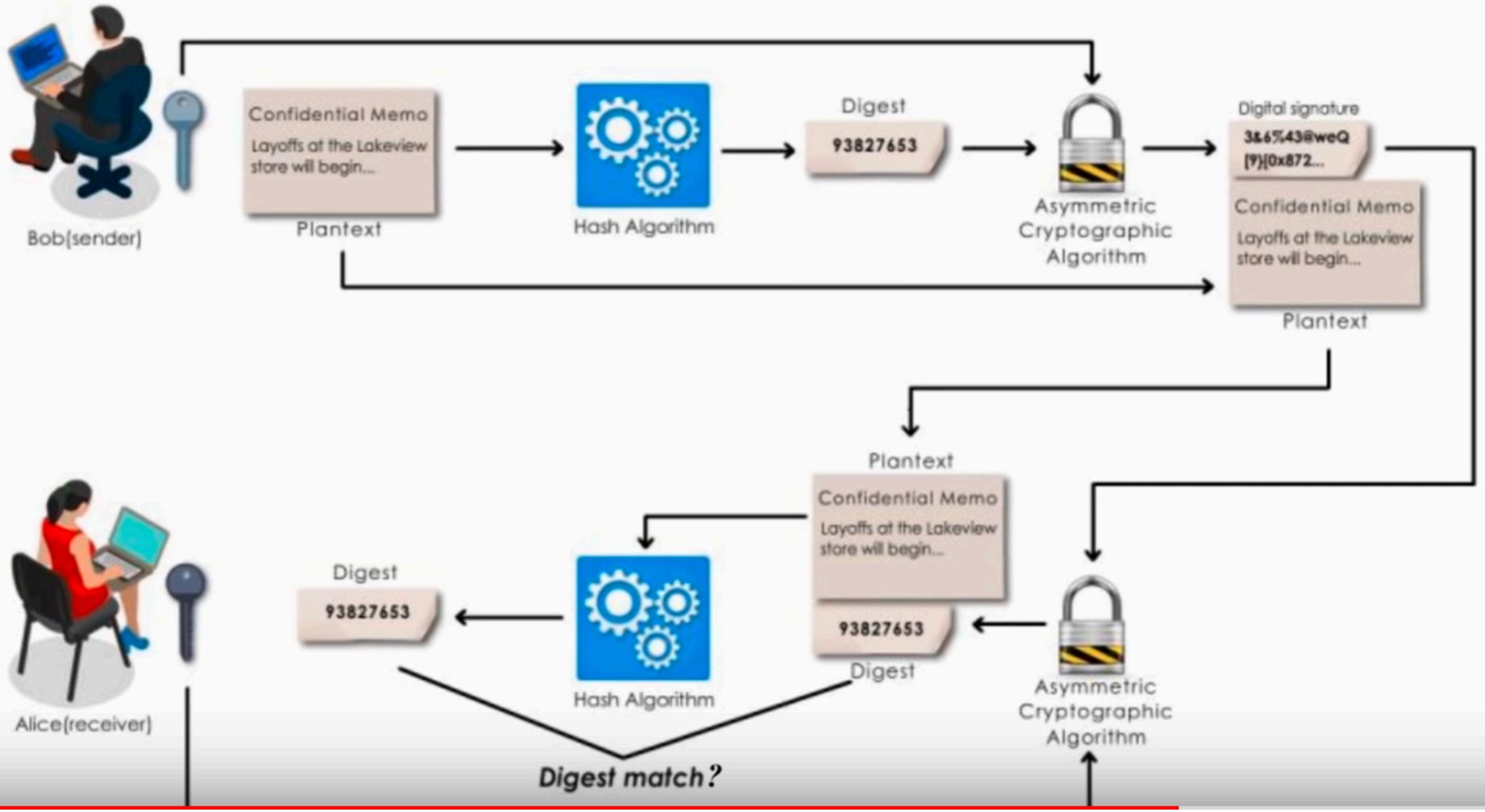
Digital Signature



Digital Signature

- Alice decrypts the digital signature using the public key
  - If she can not decrypt it, she knows it did not come from Bob
    - Only Bob can generate this digest
- Alice hashes the plaintext and compares the digests
  - If they are equal, she knows message did not change

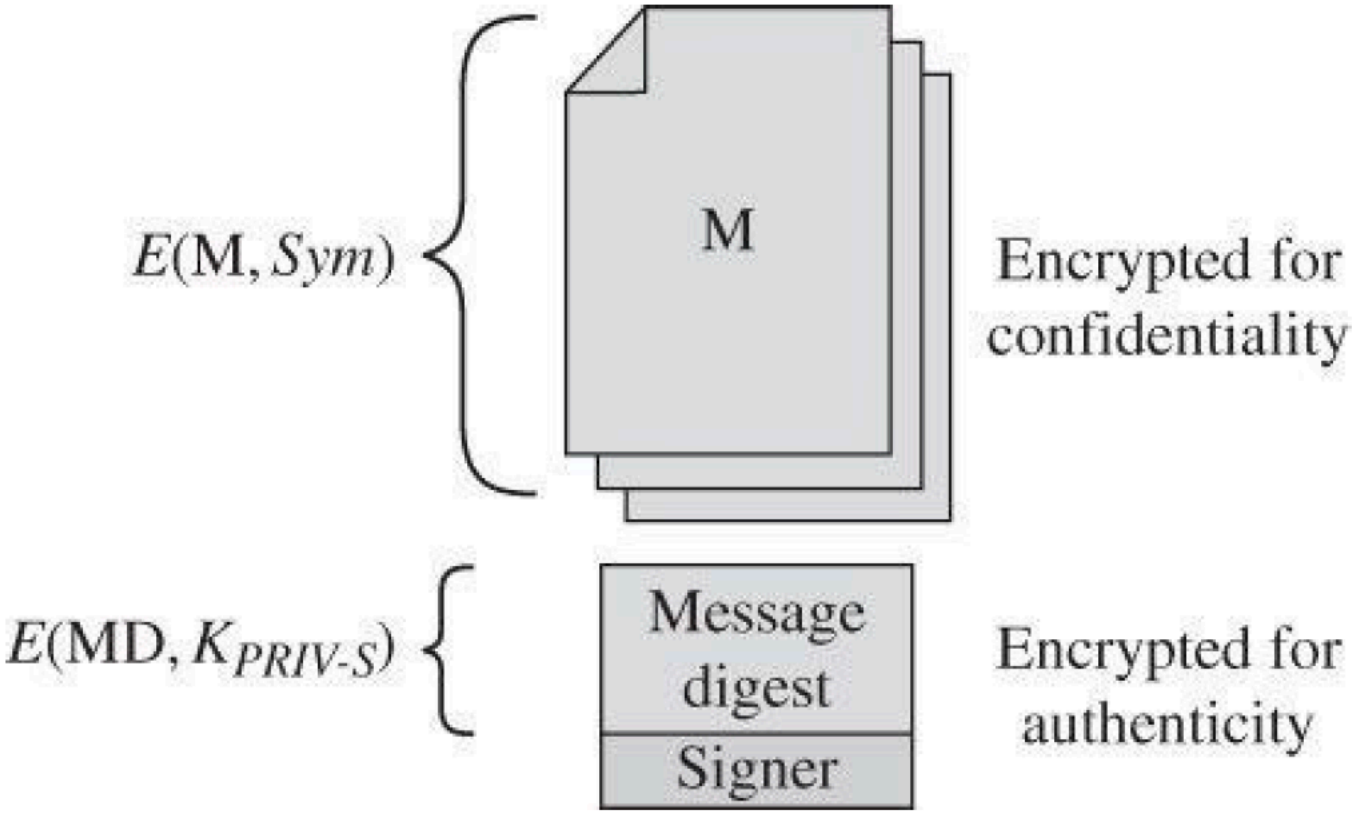
Digital Signature



Complete Digital Signature

- Confidentiality:
  - Digital signature does not provide confidentiality
    - In the previous example, message sent in plaintext
  - If confidentiality is required, signer can encrypt the file
    - Using symmetric key encryption and the user key

Complete Digital Signature



Digital Signatures

- A digital signature can indicate the authenticity of a file or a piece of code
- When you attempt to install a piece of signed code, the operating system will inspect the certificate and file
  - notify you if the certificate and hash are not acceptable
- Digital signatures provide an effective tool
  - coupled with strong hash functions and symmetric encryption, help ensure that a file is precisely what the originator stored for download

Digital Signature

- Digital Signature

Cryptographic Tools

Cryptographic primitive Security Goal	Hash	MAC	Digital signature
Integrity	Yes	Yes	Yes
Authentication	No	Yes	Yes
Non-repudiation	No	No	Yes
Kind of keys	none	symmetric keys	asymmetric keys

https://crypto.stackexchange.com/questions/5646/what-are-the-differences-between-a-digital-signature-a-mac-and-a-hash

Cryptographic Tools

- To achieve integrity, the hash needs to be kept separate from the file we are authenticating
  - When storing the file
    - Otherwise, adversary can change it too
  - When the file is read, the hash is recalculated
    - And compared to the saved hash
- Hash does not provide authentication
  - Keyless, so anyone can compute the hash

https://crypto.stackexchange.com/questions/5646/what-are-the-differences-between-a-digital-signature-a-mac-and-a-hash

Cryptographic Tools

Cryptographic primitive Security Goal	Hash	MAC	Digital signature
Integrity	Yes	Yes	Yes
Authentication	No	Yes	Yes
Non-repudiation	No	No	Yes
Kind of keys	none	symmetric keys	asymmetric keys

https://crypto.stackexchange.com/questions/5646/what-are-the-differences-between-a-digital-signature-a-mac-and-a-hash

Trust

- How do we know that a webpage indeed belongs to the listed company?
  - Web pages can be replaced and faked
    - No warning to the user
- We can establish trust based on a common and trusted entity
  - The ***certificate authority***

Certificates: Trustable Identities and Public Keys

- A certificate is a public key and an identity bound together and signed by a ***certificate authority***.
- A ***certificate authority*** is an authority that users trust
  - accurately verifies identities before generating certificates that bind those identities to keys.
- Certificates have a lifetime and must be maintained.

Certificate Authority and Digital Signature

- **Digital** certificate is **an electronic** document issued by **a Certificate Authority** (CA)
- It contains the public key for a **digital signature**
  - specifies the identity associated with the key, such as the name of an organization

Digital Certificate Authorities, 2015 - Wikipedia

Rank	Issuer	Usage	Market Share
1	Comodo	8.1%	40.6%
2	Symantec	5.2%	26.0%
3	GoDaddy	2.4%	11.8%
4	GlobalSign	1.9%	9.7%
5	IdenTrust	0.7%	3.5%
6	DigiCert	0.6%	3.0%
7	StartCom	0.4%	2.1%
8	Entrust	0.1%	0.7%
9	Trustwave	0.1%	0.5%
10	Verizon	0.1%	0.5%

Digital Certificate Authorities, May, 2018 - Wikipedia

Rank	Issuer	Usage	Market share
1	IdenTrust	20.4%	39.7%
2	Comodo	17.9%	34.9%
3	DigiCert	6.3%	12.3%
4	GoDaddy	3.7%	7.2%
5	GlobalSign	1.8%	3.5%
6	Certum	0.4%	0.7%
7	Actalis	0.2%	0.3%
8	Entrust	0.2%	0.3%
9	Secom	0.1%	0.3%
10	Let's Encrypt	0.1%	0.2%

What happened to Symantec?

- in 2016, users noticed Symantec issuing certificates against certain guidelines
  - posted this information to a Mozilla mailing list
- Other major CA's discussed the issue
  - Decided to distrust Symantec
  - Google also announced it is distrusting their certificate

Certificate Signing and Hierarchy

- It is possible to create a certificate that includes another certificate
  - Creating a chain of trust

Certificate Signing and Hierarchy

To create Diana's certificate:

Diana creates and delivers to Edward:

Name: Diana Position: Division Manager Public key: 17EF83CA ...

Edward adds:

Name: Diana Position: Division Manager Public key: 17EF83CA ...	hash value 128C4

Edward signs with his private key:

Name: Diana Position: Division Manager Public key: 17EF83CA ...	hash value 128C4

Which is Diana's certificate.

To create Delwyn's certificate:

Delwyn creates and delivers to Diana:

Name: Delwyn Position: Dept Manager Public key: 3AB3882C ...

Diana adds:

Name: Delwyn Position: Dept Manager Public key: 3AB3882C ...	hash value 48CFA

Diana signs with her private key:

Name: Delwyn Position: Dept Manager Public key: 3AB3882C ...	hash value 48CFA

And appends her certificate:

Name: Delwyn Position: Dept Manager Public key: 3AB3882C ...	hash value 48CFA
Name: Diana Position: Division Manager Public key: 17EF83CA ...	hash value 128C4

Which is Delwyn's certificate.

Certificate Signing and Hierarchy - Example

- Diana's certificate is made using Edward's signature.
- Delwyn's certificate includes Diana's certificate so that it can effectively be tied back to Edward, creating a chain of trust.

Certificate Signing and Hierarchy

To create Diana's certificate:

Diana creates and delivers to Edward:

Name: Diana Position: Division Manager Public key: 17EF83CA ...

Edward adds:

Name: Diana Position: Division Manager Public key: 17EF83CA ...	hash value 128C4

Edward signs with his private key:

Name: Diana Position: Division Manager Public key: 17EF83CA ...	hash value 128C4

Which is Diana's certificate.

To create Delwyn's certificate:

Delwyn creates and delivers to Diana:

Name: Delwyn Position: Dept Manager Public key: 3AB3882C ...

Diana adds:

Name: Delwyn Position: Dept Manager Public key: 3AB3882C ...	hash value 48CFA

Diana signs with her private key:

Name: Delwyn Position: Dept Manager Public key: 3AB3882C ...	hash value 48CFA

And appends her certificate:

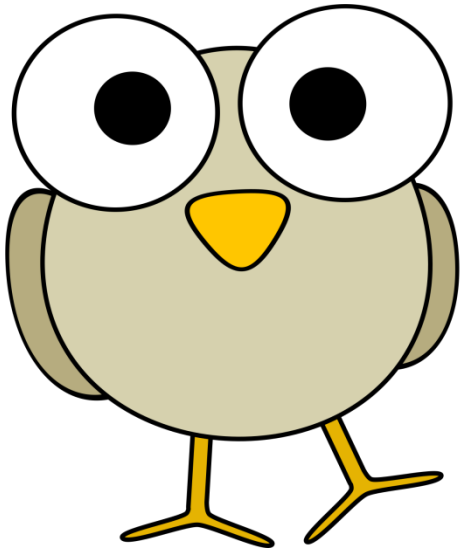
Name: Delwyn Position: Dept Manager Public key: 3AB3882C ...	hash value 48CFA
Name: Diana Position: Division Manager Public key: 17EF83CA ...	hash value 128C4

Which is Delwyn's certificate.

TOOLS DERIVED FROM CRYPTOGRAPHY

Tool	Uses
Secret key (symmetric) encryption	Protecting confidentiality and integrity of data at rest or in transit
Public key (asymmetric) encryption	Exchanging (symmetric) encryption keys Signing data to show authenticity and proof of origin
Error detection codes	Detect changes in data
Hash codes and functions (forms of error detection codes)	Detect changes in data
Cryptographic hash functions	Detect changes in data, using a function that only the data owner can compute (so an outsider cannot change both data and the hash code result to conceal the fact of the change)
Error correction codes	Detect and repair errors in data
Digital signatures	Attest to the authenticity of data
Digital certificates	Allow parties to exchange cryptographic keys with confidence of the identities of both parties

- Questions?



ADDITIONAL TOPICS

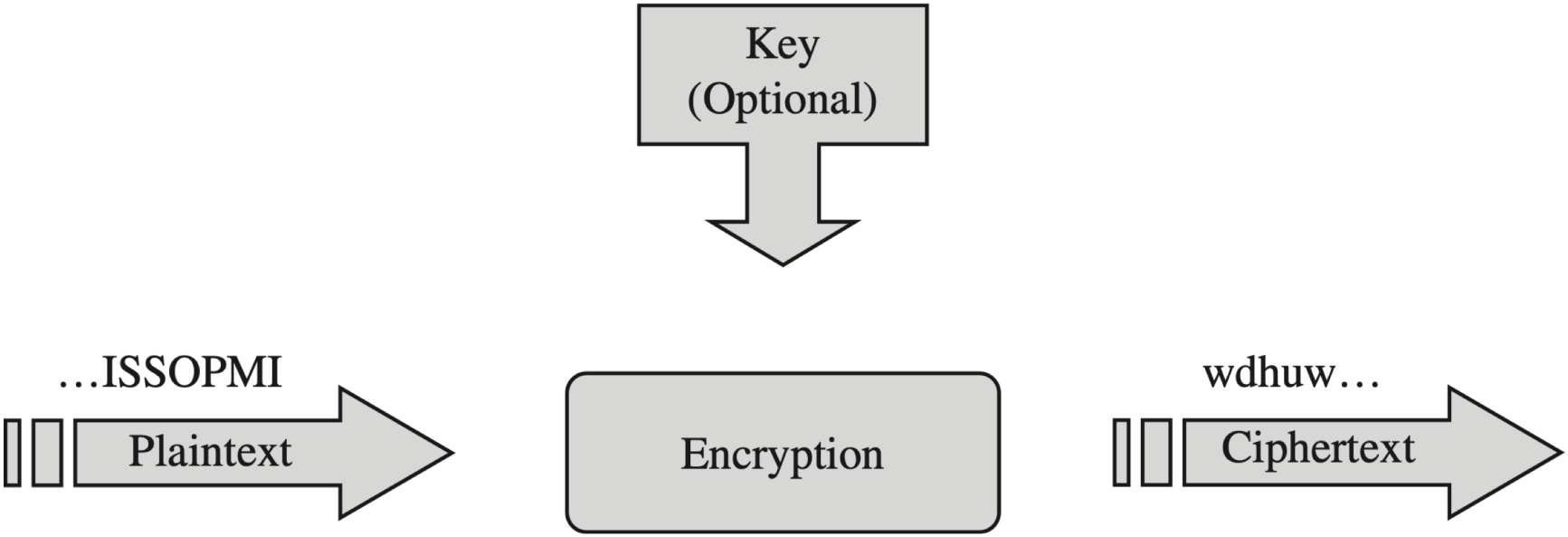
Stream Ciphers

- An alternative approach to using block cipher
- In stream ciphers, each byte of the data stream is encrypted separately
  - This is as opposed to block ciphers

Stream Ciphers

- Key stream
  - Pseudo-random sequence of bits S = S[0], S[1], S[2], …
  - Can be generated on-line one bit (or byte) at the time
  - Successive elements of the keystream generated based on an internal state
- Stream cipher
  - XOR the plaintext with the key stream C[i] = S[i] ⊕ P[i]
  - Suitable for plaintext of arbitrary length generated on the fly, e.g., media stream

Stream Ciphers



Key Stream Generation

- RC4
  - Designed in 1987 by Ron Rivest for RSA Security
  - Trade secret until 1994
  - Uses keys with up to 2,048 bits
  - Simple algorithm

Key Stream Generation

- Block cipher in counter mode (CTR)
  - Use a block cipher with block size b
  - The secret key is a pair (K,t), where K a is key and t (counter) is a b-bit value
  - The key stream is the concatenation of ciphertexts
  - EK (t), EK (t + 1), EK (t + 2), …
  - Can use a shorter counter concatenated with a random value
  - Synchronous stream cipher

Questions?

CRYPTOGRAPHIC LIBRARIES

NaCl

- A Networking and Cryptography library that has a symmetric library (secretbox) and an asymmetric library (box)
  - designed by Daniel J. Bernstein
  - Can be found at:
    NACL Library

Cryptographic Libraries

- Many other exist, see:
  - Cryptographic Libraries
- Adding cryptographic functionality to your product:
  - Start by choosing a secure protocol
    - Protocols continuously get updated, make sure you have current information
    - Choose a known secure library
      - Do not write your own!

Questions?