COMPUTER 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

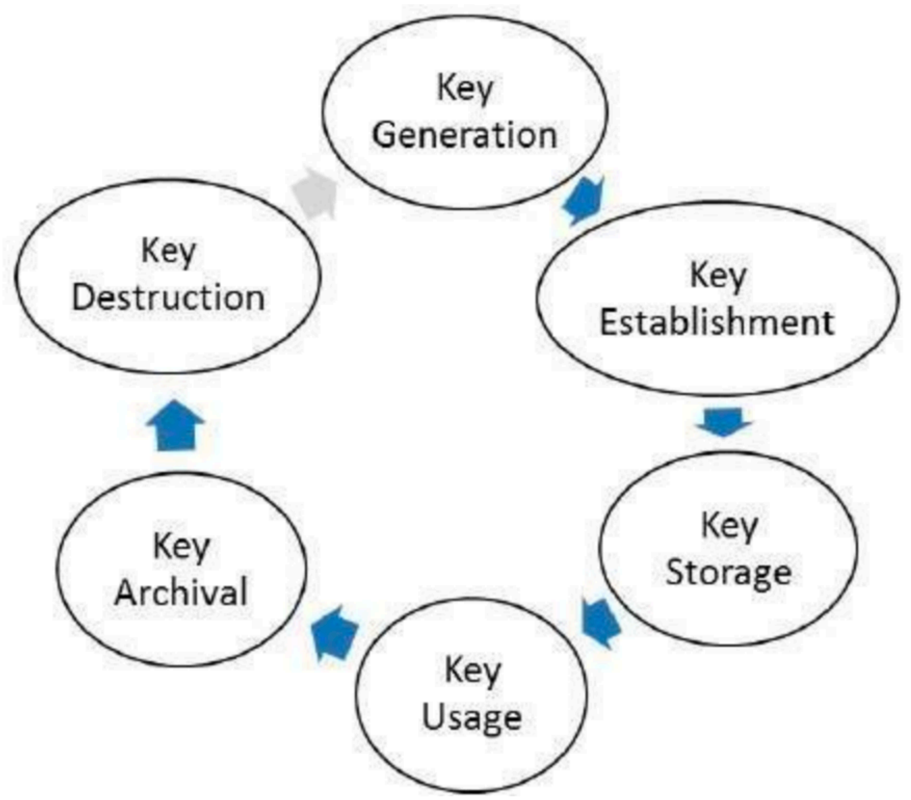
CRYPTOGRAPHY

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

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

Cryptography - review

- Asymmetric cryptography uses one public and one private key
  - Much slower than symmetric cryptography
  - Can be used for key management

Security concepts

- Concepts achieved through symmetric and asymmetric cryptography:
  - ***Confidentiality***: achieved through encryption
    - An eavesdropper can not read message back
  - ***Authenticity***: provided through the digital signature mechanism
  - ***Non-repudiation***: the author of the message can not dispute the original of the message
    - Provided through digital signature

Why do we use cryptography?

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

MESSAGE INTEGRITY

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

ERROR DETECTING CODES

Error Detecting Codes

- Communications are prone to transmission errors
- Need to have a way to verify intact transmission
  - For sensitive data
- Different error detection mechanisms exist
- Aims to indicate that a message has changed
  - Different techniques work for different errors

Error Detecting vs. Correcting Codes

- ***Error detecting codes*** detect when an error has occurred
- ***Error correcting codes*** can actually correct errors without requiring a copy of the original data
  - without requiring a copy of the original data

Error Detecting vs. Correcting Codes

- The error code is computed and stored safely on the presumed intact, original data;
- Error code can be recomputed later
    - check whether the received result matches the expected value.
    - If the values do not match, a change has occurred
        - If the values match, it is probable—but not certain—that no change has occurred

Error Detecting Codes

- Demonstrates that a block of data has been modified
- Simple error detecting codes:
    - Parity checks
        - Parity bit added to a string of binary code to ensure that the total number of 1-bits in the string is even or odd
    - Cyclic redundancy checks – used on hardware devices

Error Detecting Codes

- Cryptographic error detecting codes:
  - One-way hash functions
  - Cryptographic checksums
  - Digital signatures

Parity Check

Original Data	Parity Bit	Modified Data	Modification Detected?
0 0 0 0 0 0 0 0	1	0 0 0 0 0 0 0 1	
0 0 0 0 0 0 0 0	1	1 0 0 0 0 0 0 0	
0 0 0 0 0 0 0 0	1	1 0 0 0 0 0 0 1	
0 0 0 0 0 0 0 0	1	0 0 0 0 0 0 1 1	
0 0 0 0 0 0 0 0	1	0 0 0 0 0 1 1 1	
0 0 0 0 0 0 0 0	1	0 0 0 0 1 1 1 1	
0 0 0 0 0 0 0 0	1	0 1 0 1 0 1 0 1	

Parity Check

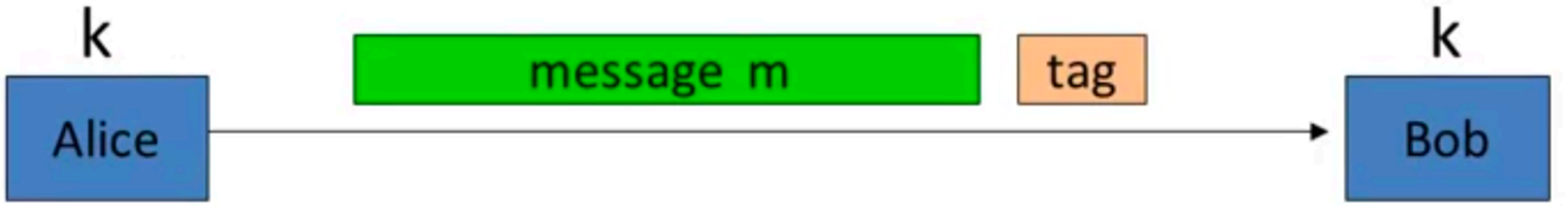
Original Data	Parity Bit	Modified Data	Modification Detected?
0 0 0 0 0 0 0 0	1	0 0 0 0 0 0 0 1	Yes
0 0 0 0 0 0 0 0	1	1 0 0 0 0 0 0 0	Yes
0 0 0 0 0 0 0 0	1	1 0 0 0 0 0 0 1	No
0 0 0 0 0 0 0 0	1	0 0 0 0 0 0 1 1	No
0 0 0 0 0 0 0 0	1	0 0 0 0 0 1 1 1	Yes
0 0 0 0 0 0 0 0	1	0 0 0 0 1 1 1 1	No
0 0 0 0 0 0 0 0	1	0 1 0 1 0 1 0 1	No

MESSAGE AUTHENTICATION CODES (MAC)

Message Authentication Codes (MAC)

- Generates a short piece of information
  - I.e., tag
- Allows authentication of a received message
  - Ensures message came from alleged sender
    - Not an attacker

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)

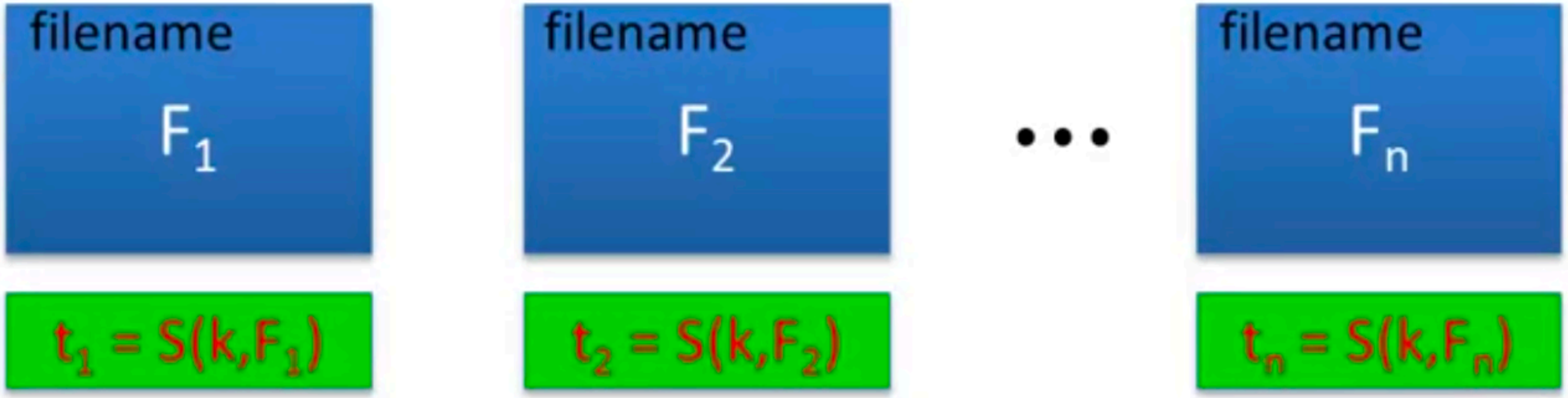
- Shared secret key is needed
  - Otherwise, an attacker may be able to modify message and generate its own tag
  - Same shared secret key used to generate and verify tag
  - Needs to be shared ahead of time

Message Integrity (MAC)

- 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 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

Message Integrity (MAC)

- How are MAC created?
  - Mac are sometimes created collision-resistant hash functions
    - Example: HMAC

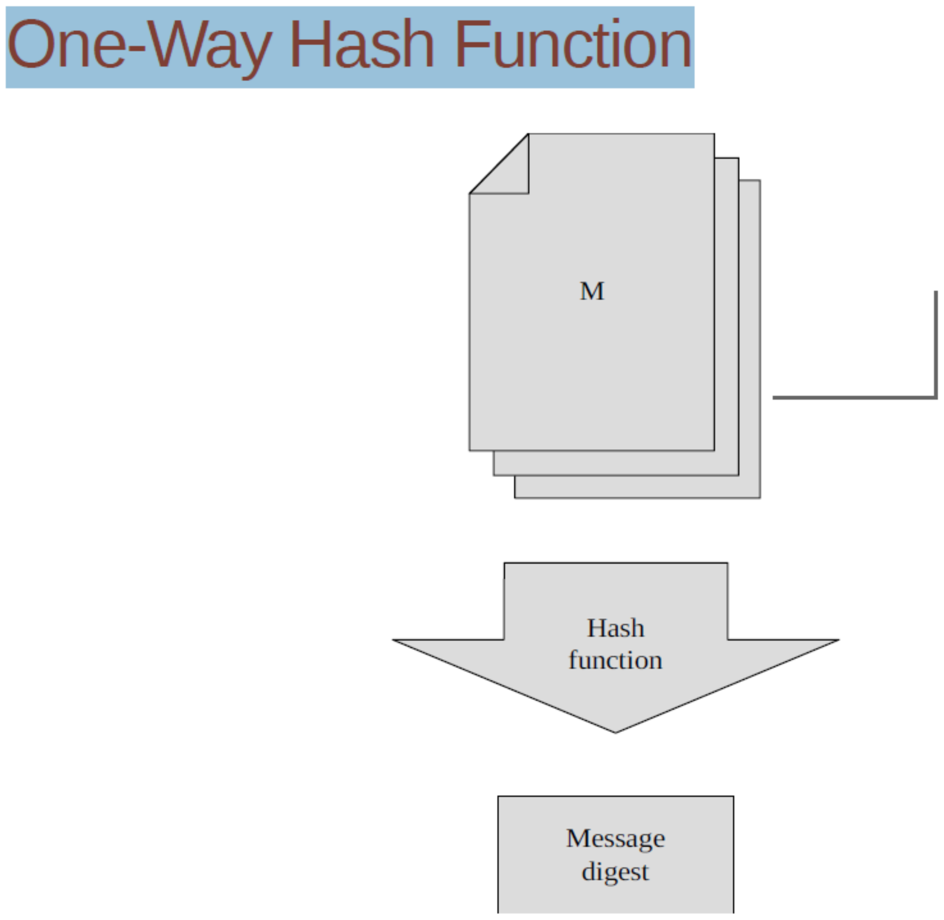
Hash Function

- 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

Checksum vs. Hash Functions

- Checksum adds up all the bits in a file or message and records the value
  - can be stored in plain text or encrypted
- A hash function applies a one-way function to the checksum or to a subset of message value
  - => it is not easily deciphered

One-Way Hash Function

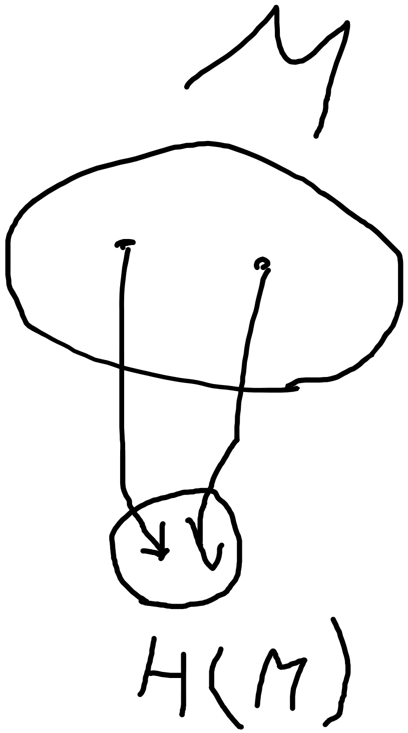


Hash Functions

- 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

Hash Functions

- What about collisions?
  - How probable are they?
- Intuition:
  - What is the likelihood of sharing a birthday?

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

$$F_k = \left(1 - \frac{1}{m}\right)\left(1 - \frac{2}{m}\right)\left(1 - \frac{3}{m}\right) \ldots \left(1 - \frac{k-1}{m}\right)$$

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

$$F_k \approx e^{-\left(\frac{1}{m}+\frac{2}{m}+\ldots+\frac{k-1}{m}\right)} = e^{\frac{-k(k-1)}{2m}}$$

The attack succeeds/fails with probability ½ when $F_k = \frac{1}{2}$,

that is,

$$e^{\frac{-k(k-1)}{2m}} = ½ \Rightarrow k \text{ ⍰ } 1.17\, m^{½}$$

Birthday Attack

- The attack succeeds/fails with probability ½ when $Fk = \frac{1}{2}$, that is,

$$e^{\frac{-k(k-1)}{2m}} = ½ \Rightarrow k \approx 1.17\, m^{½}$$

- 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 = 364/365
- The probability that the third person was not both on either of those days = 363/365
- For 24 people, we get:
  - P(no collusion) = $\frac{364*363*\cdots*241}{365^{24}}$ = 0.46 = 46%
  - $P(collusion) = 54\%$

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
    - Typically, functions designed in the 90’s should not be used!
- Only SHA-2 AND SHA-3 should be used

Cryptographic Checksum

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

HMAC - Hash Message Authentication Code

- A popular and secure type of MAC
  - Uses hash function
- Building a MAC from a cryptographic hash function is not immediate

HMAC - Hash Message Authentication Code

- 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

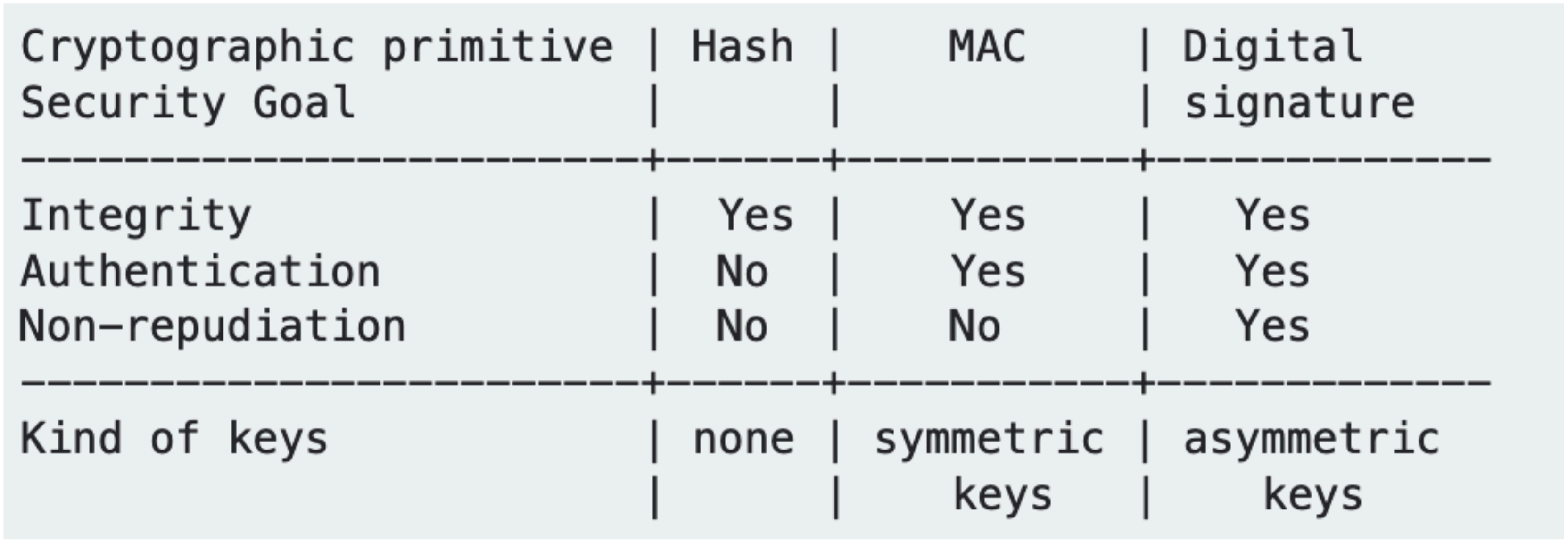
HMAC

- HMAC

CMAC (Cipher-based Message Authentication Code)

- Another class of MACS
- Instead of hash function, a symmetric cipher function is used to encrypt data
  - Using the output as the MAC
- Example: CBC-MAC uses block-ciphers to produce MACS

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

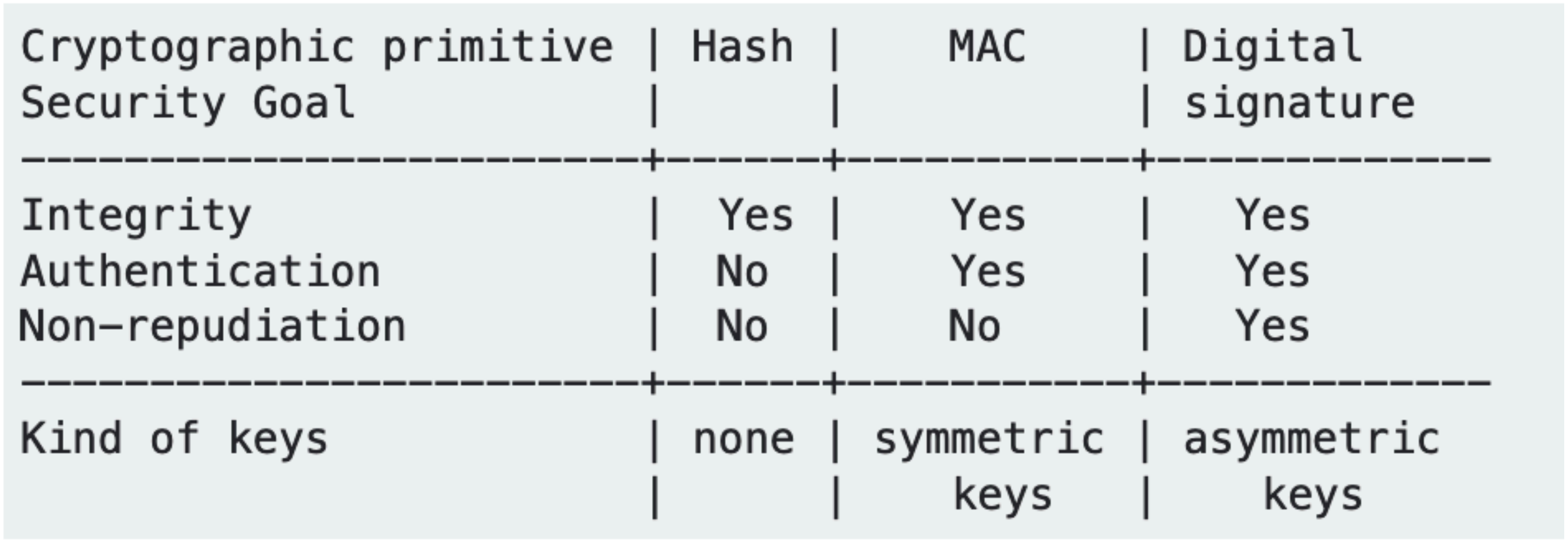
- How do we achieve integrity?
  - Data can not be changed
- How do we achieve authentication?
  - Tool depends both on the content and the signer
- How do we achieve non-repudiation?
  - Only original signer could sign data, everyone can verify this

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

PUBLIC KEY INFRASTRUCTURE (PKI)

PKI

- A technology for authenticating users and devices in the digital world
- Have trusted parties digitally sign documents
  - certifying that a particular cryptographic key belongs to a particular user or device
- The key can then be used as an **identity for the user** in digital networks
- **Certificate Authority (CA)** is the most common implementation

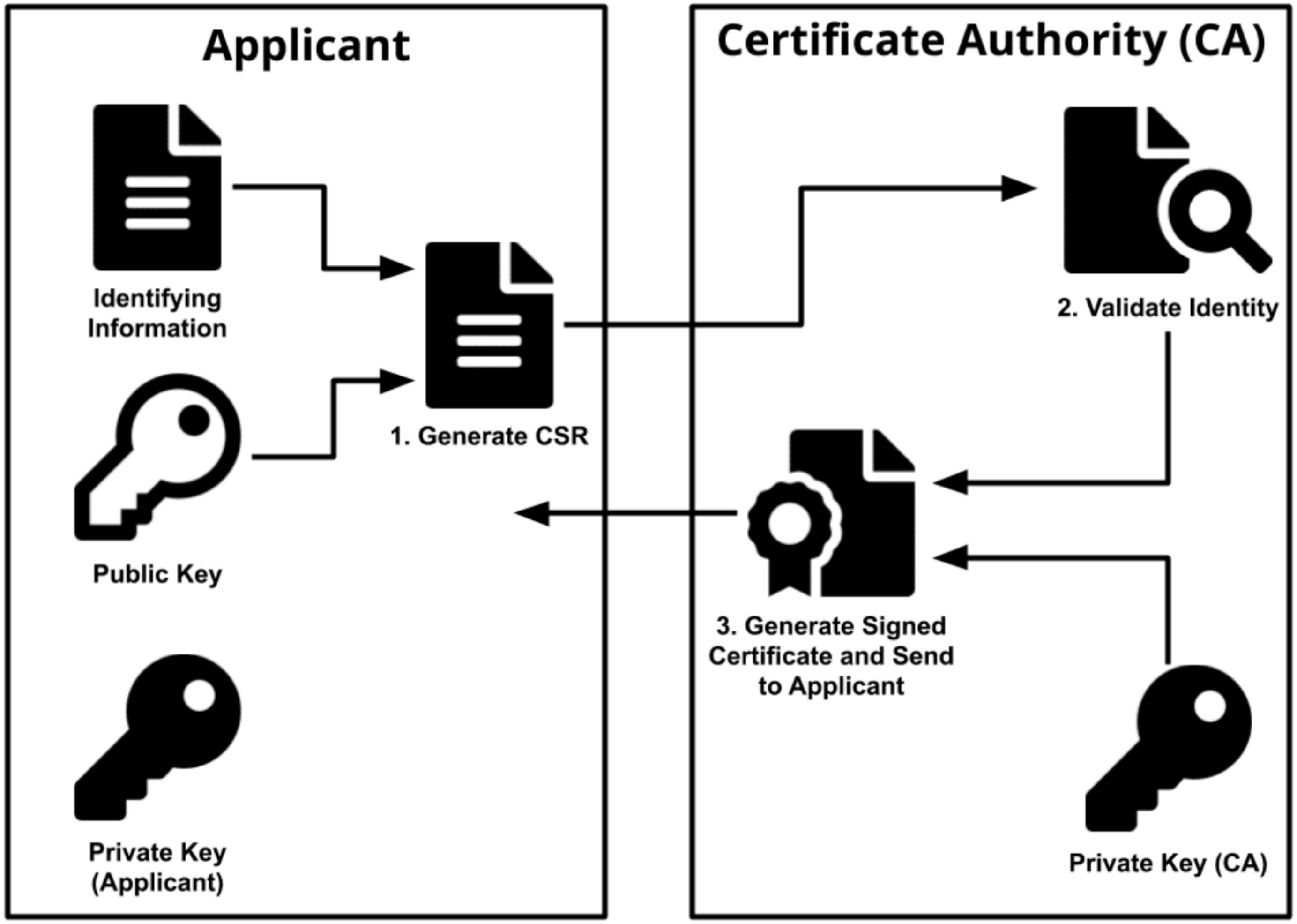
Trust

- 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***

Certificate Authority and Digital Signature

- **Digital certificate** is an electronic document issued by **a Certificate Authority (CA)**
- Contains a ***public key*** and an **identity** bound together and signed by the **CA**
  - Key can later be used for digital signatures
  - Specifies the identity associated with the key
    - such as the name of an organization

Certificate Authority



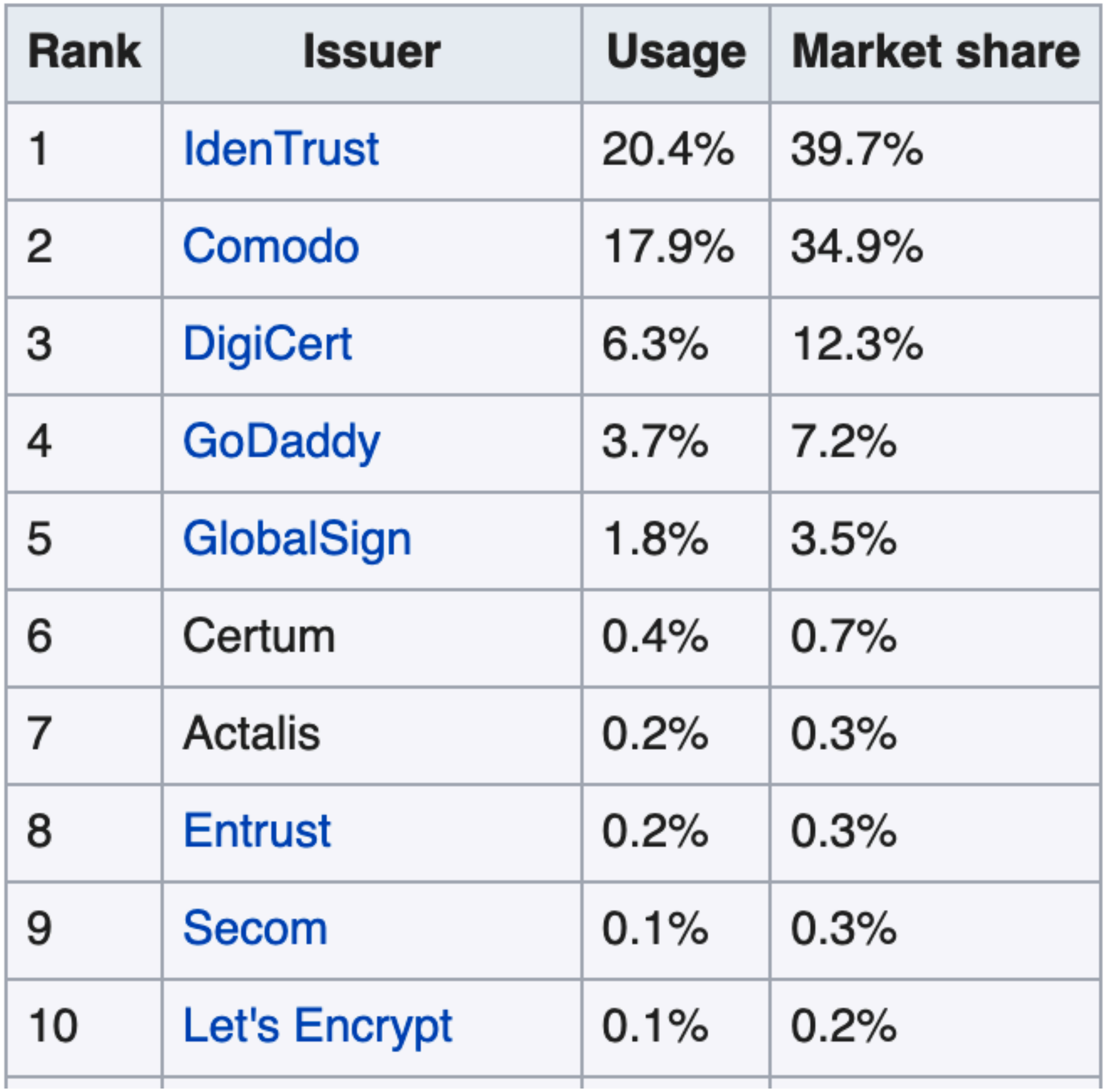
Digital Certificate

- Scenario:
  - Alice wants to communicate with Bob, but does not know Bob’s ***public key***
  - Bob can send her his public key
    - How can Alice Verify Bob’s public key?
  - Bob gets a **digital certificate** from a certificate authority
    - Sends it to Alice
    - Alice can verify Bob’s public key using the CA public key

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

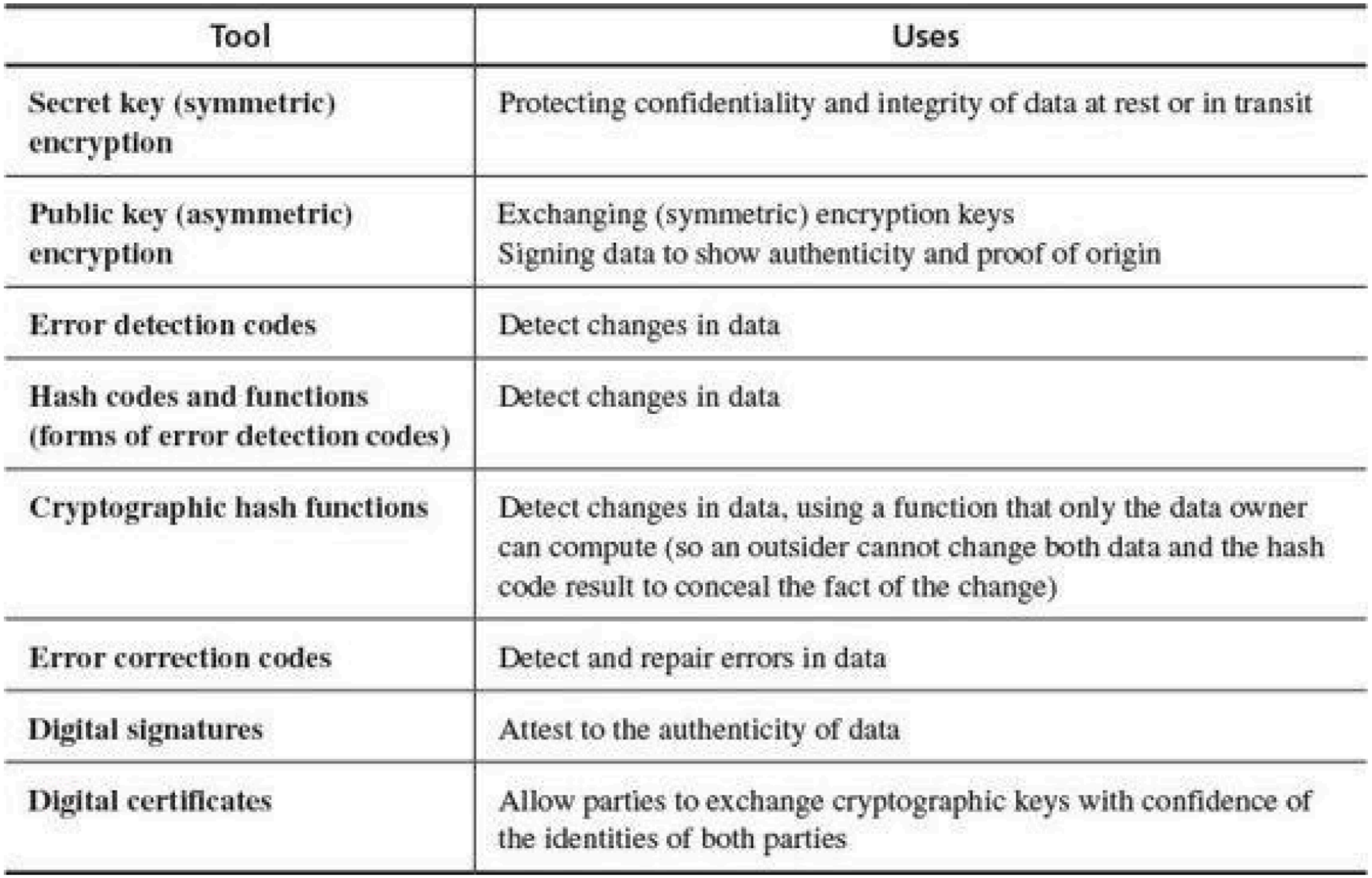


Rank	Issuer	Usage	Market share
1	IdenTrust	20.4%	39.7%
2	Comodo	17.9%	34.9%
3	DigiCert	6.3%	12.3%
4	GoDaddy	3.7%	7.2%
5	GlobalSign	1.8%	3.5%
6	Certum	0.4%	0.7%
7	Actalis	0.2%	0.3%
8	Entrust	0.2%	0.3%
9	Secom	0.1%	0.3%
10	Let's Encrypt	0.1%	0.2%

What happened to Symantec?

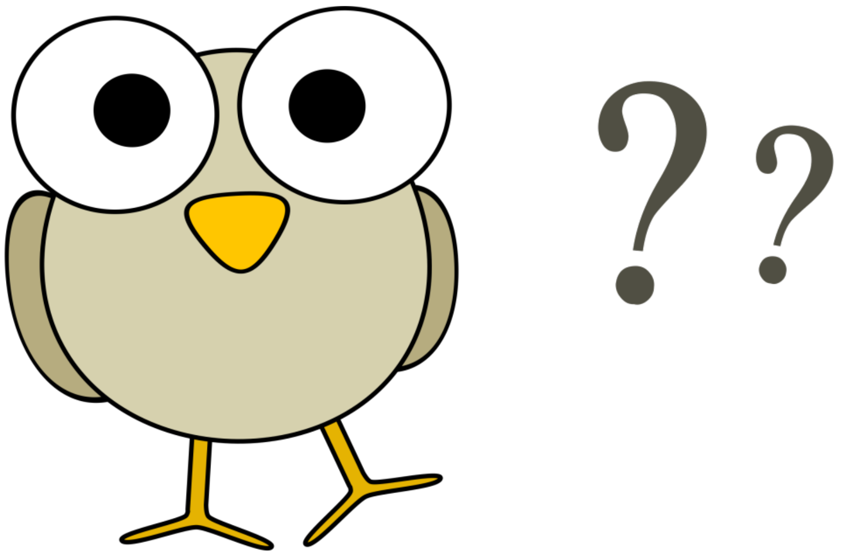
- in 2016, users noticed Symantec issuing certificates against certain guidelines
  - posted this information to a Mozilla mailing list
- Other major CA’s discussed the issue
  - Decided to distrust Symantec
  - Google also announced it is distrusting their certificate

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

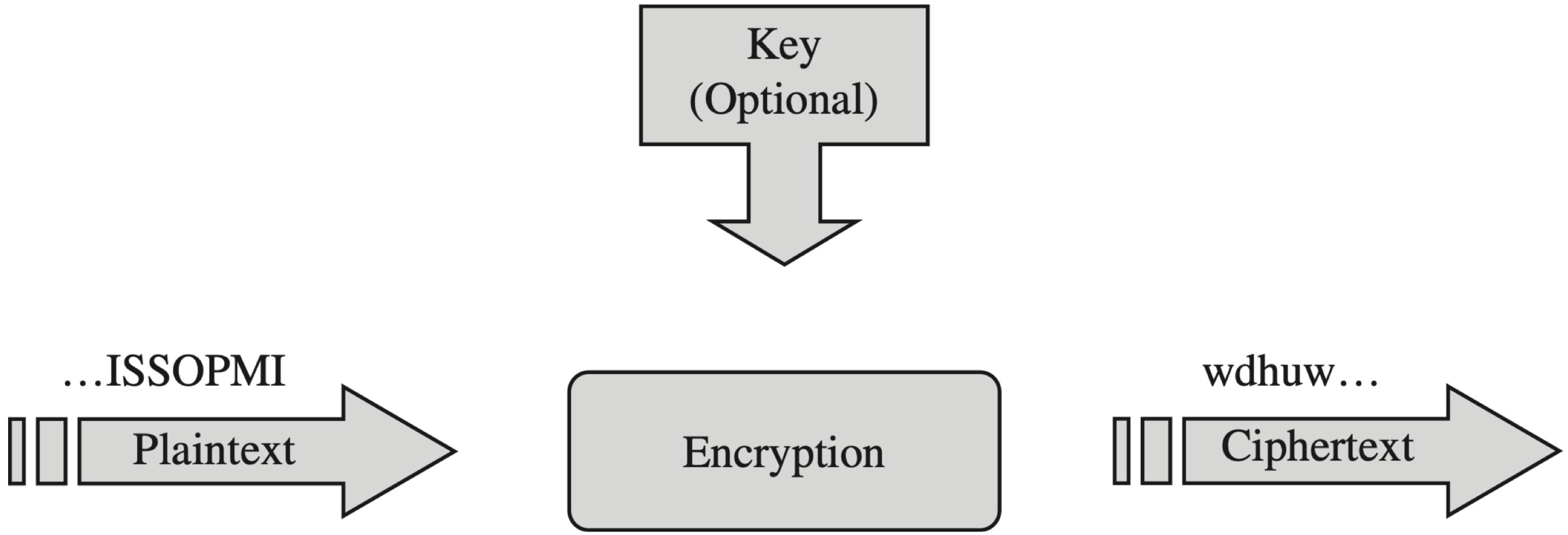
Stream Ciphers

- 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 $C[i] = S[i] \oplus P[i]$
  - Suitable for plaintext of arbitrary length generated on the fly, e.g., media stream

Stream Ciphers



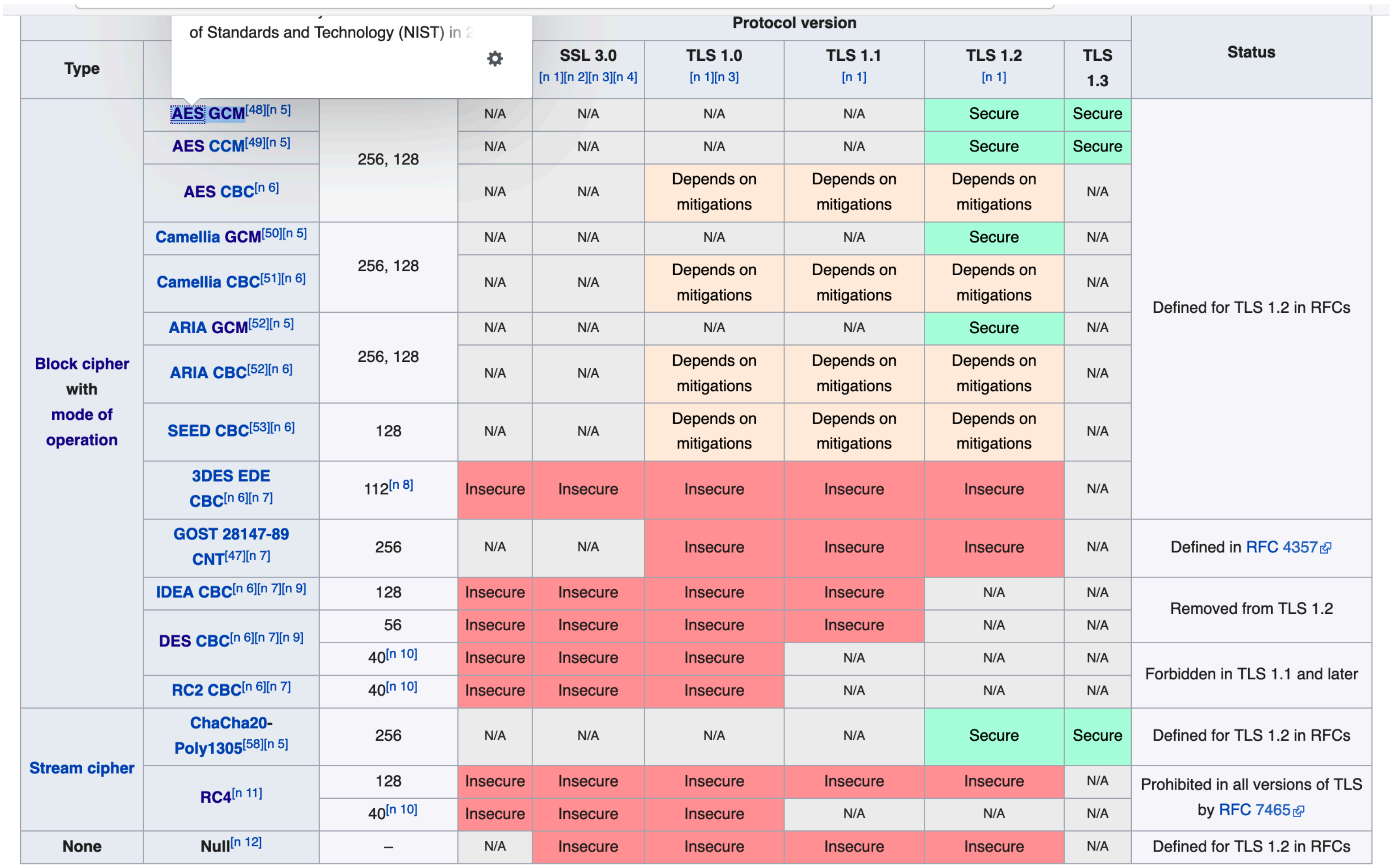
Key Stream Generation

- RC4
    - Symmetric stream cipher algorithm
    - Designed in 1987 by Ron Rivest for RSA Security
    - Trade secret until 1994
    - Uses keys with up to 2,048 bits
    - Widely adopted due to its simplicity and speed

Key Stream Generation

- RC4:
  - Recent attacks show cipher is not secure anymore
  - Any protocol that uses this cipher is vulnerable to attacks
  - Was used in WEP,WPA
    - SSL AND TLS used it until 2015 because of inherent weaknesses

TLS



of Standards and Technology (NIST) in

				Protocol version					
Type				SSL 3.0 [n 1][n 2][n 3][n 4]	TLS 1.0 [n 1][n 3]	TLS 1.1 [n 1]	TLS 1.2 [n 1]	TLS 1.3	Status
Block cipher with mode of operation	AES GCM[48][n 5]	256, 128	N/A	N/A	N/A	N/A	Secure	Secure	Defined for TLS 1.2 in RFCs
	AES CCM[49][n 5]		N/A	N/A	N/A	N/A	Secure	Secure	
	AES CBC[n 6]		N/A	N/A	Depends on mitigations	Depends on mitigations	Depends on mitigations	N/A	
	Camellia GCM[50][n 5]	256, 128	N/A	N/A	N/A	N/A	Secure	N/A	
	Camellia CBC[51][n 6]		N/A	N/A	Depends on mitigations	Depends on mitigations	Depends on mitigations	N/A	
	ARIA GCM[52][n 5]	256, 128	N/A	N/A	N/A	N/A	Secure	N/A	
	ARIA CBC[52][n 6]		N/A	N/A	Depends on mitigations	Depends on mitigations	Depends on mitigations	N/A	
	SEED CBC[53][n 6]	128	N/A	N/A	Depends on mitigations	Depends on mitigations	Depends on mitigations	N/A	
	3DES EDE CBC[n 6][n 7]	112[n 8]	Insecure	Insecure	Insecure	Insecure	Insecure	N/A	
	GOST 28147-89 CNT[47][n 7]	256	N/A	N/A	Insecure	Insecure	Insecure	N/A	Defined in RFC 4357
	IDEA CBC[n 6][n 7][n 9]	128	Insecure	Insecure	Insecure	Insecure	N/A	N/A	Removed from TLS 1.2
	DES CBC[n 6][n 7][n 9]	56	Insecure	Insecure	Insecure	Insecure	N/A	N/A	
		40[n 10]	Insecure	Insecure	Insecure	N/A	N/A	N/A	Forbidden in TLS 1.1 and later
	RC2 CBC[n 6][n 7]	40[n 10]	Insecure	Insecure	Insecure	N/A	N/A	N/A	
Stream cipher	ChaCha20-Poly1305[58][n 5]	256	N/A	N/A	N/A	N/A	Secure	Secure	Defined for TLS 1.2 in RFCs
	RC4[n 11]	128	Insecure	Insecure	Insecure	Insecure	Insecure	N/A	Prohibited in all versions of TLS by RFC 7465
		40[n 10]	Insecure	Insecure	Insecure	N/A	N/A	N/A	
None	Null[n 12]	–	N/A	Insecure	Insecure	Insecure	Insecure	N/A	Defined for TLS 1.2 in RFCs

https://en.wikipedia.org/wiki/Transport_Layer_Security

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?

Security Engineering

- Rob: what a security engineering does

Questions?