ECE/CS230
Computer Systems Security

Charalambos (Harrys) Konstantinou

https://sites.google.com/view/ececs230kaust/

Cryptography

News

- A1 due this Thursday 18:00
- Your task: discussions (every week)

This week’s topic

- By the end, you will be able to:
  - Understand the difference between symmetric and asymmetric cryptography
  - Understand and use one-way hash functions
  - Cryptographically suitable pseudorandom number generation

Cryptography

- Etymology: Secret (Crypt) Writing (Graphy)
- Study of mathematical techniques to achieve various goals in information security, such as confidentiality, authentication, integrity, non- repudiation, etc.
- Not the only means of providing information security, rather a subset of techniques.
- Quite an old field!

Why Cryptography?

- To put it quite simply, it is the backbone of cybersecurity, critical for protecting information (confidentiality and integrity)
- Imagine a world without Cryptography. Imagine your usernames, passwords, credit card numbers, messages, secrets, account details, personal information, emails, etc. were all transmitted over a computer network in plaintext, unencrypted

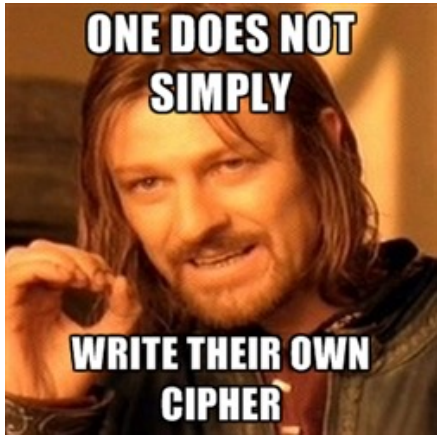
Warning

- This week is not a comprehensive study on Cryptography
- Cryptography stands as a course and field of its own
- While cryptography is critical in cybersecurity, cryptography and cybersecurity are not the same. Some academic institutions still teach cybersecurity as cryptography. There is a lot more to cybesecurity than cryptography.

Definitions

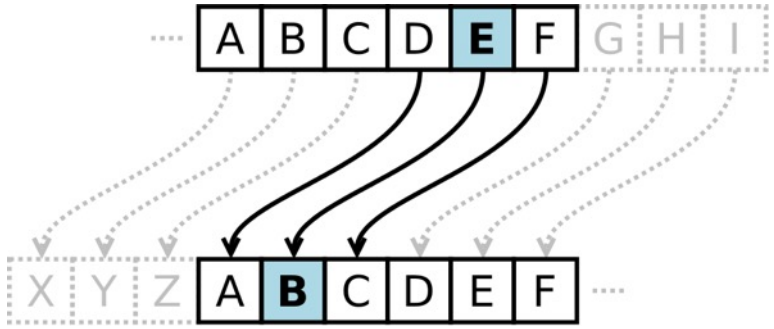
- **Cryptography** - The process of communicating secretly through the use of *cipher*
- **Cryptanalysis (codebreaking)** - The process of cracking or deciphering; code breaking
- **Cryptology** - The study of cryptography or cryptanalysis
- **Cleartext / plaintext** - What you are reading now
- **Encrypt (encipher)** - convert information or data into code to prevent unauthorized access
- **Decrypt (decipher)** – convert an encoded or unclear message into something intelligible, to plaintext
- **Cipher** - An algorithm to perform encryption and/or decryption
- **Cryptosystem** - Suite of algorithms to perform encryption and/or decryption
- **Key** - a piece of information that determines the functional output of a cryptographic algorithm. For encryption, a key specifies the transformation of plaintext into ciphertext, and vice versa for decryption

THE GOLDEN RULE



- **“Don’t roll your own crypto”**
  - The reason: https://security.stackexchange.com/questions/18197/why-shouldnt-we-roll-our-own
- “Anyone, from the most clueless amateur to the best cryptographer, can create an algorithm that he himself can't break. It's not even hard. What is hard is creating an algorithm that no one else can break, even after years of analysis. And the only way to prove that is to subject the algorithm to years of analysis by the best cryptographers around.” –Bruce Schneier
- Snake Oil: https://www.schneier.com/crypto-gram/archives/1999/0215.html#snakeoil

Historical Ciphers - Ancient History: Caesar Cipher



- A substitution cipher
- “Each letter in the original message (plaintext) is replaced with a letter corresponding to a certain number of letters up or down in the alphabet.”
- In this way, a message that initially was quite readable, ends up in a form that can not be understood at a simple glance.
- The purpose of this: what if a messenger for Julius Caesar got mugged or murdered and the message that was supposed to be delivered to another party got intercepted or stolen by enemy?

https://learncryptography.com/classical-encryption/caesar-cipher

```
Plaintext:
THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG

Ciphertext:
QEB NRFZH YOLTK CLU GRJMP LSBO QEB IXWV ALD
```

Historical Ciphers: World War II

- Enigma
  - Used by the Germans
  - Replaced letters as they were typed
  - Substitutions were computed using a key and a set of switches and rotors.

Historical Ciphers: Substitution Ciphers

Plain	abcdefghijklmnopqrstuvwxyz
Cipher	mnbvcxzasdfghjklpoiuytrewq

bob. i love you. alice

nkn. s gktc wky. mgsbc

Monoalphabetic encryption

Historical Ciphers: Transposition Ciphers

Diffusion

THIS IS A MESSAGE TO SHOW HOW
A COLUMNAR TRANSPOSITION WORKS

TSSOHOANIWHAASOLRSTOIMGHW
UTPIRSEEOAMROOKISTWCNASNS

T H I S I
S A M E S
S A G E T
O S H O W
H O W A C
O L U M N
A R T R A
N S P O S
I T I O N
W O R K S

Components (Most-likely finite)

- Set of plaintexts (P)
- Set of ciphertexts (C)
- Set of keys (K)
- Encryption functions ($c = E(k_e,p)$)
- Decryption functions ($p = D(k_d,c)$)

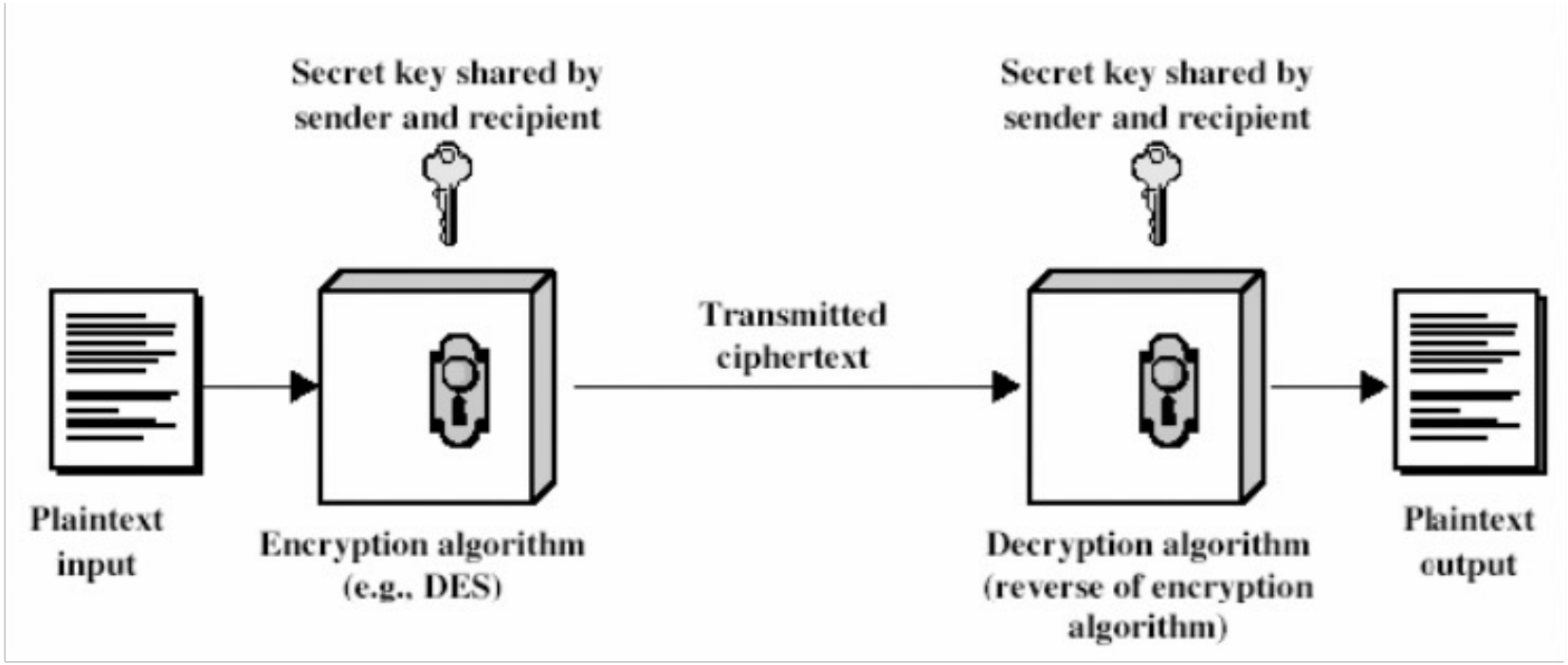
Outline

- Symmetric / Private Key cryptography
- Asymmetric / Public Key cryptography
- Hash functions
- Cryptographically suitable pseudorandom number generation
- Message integrity and digital signatures

Private Key/Public Key Cryptography

- Symmetric Key / Private Key: Sender and receiver share a common (private) key
  - Encryption and Decryption is done using the private key
  - Also called conventional/shared-key/single-key cryptography
- Asymmetric Key / Public Key: Every user has a private key and a public key
  - Encryption is done using the public key and Decryption using private key
  - Also called two-key cryptography

Symmetric / Private key model



Private Key Encryption: Main Functions

- Generate a key: KeyGen(L) -> K
  - (L is a security parameter that represents the length of the key)
- Encrypt: Enc(K,M) → C
- Decrypt: Dec(K,C) → M

Historical Ciphers: More on Caesar Cipher (or Shift Cipher)

- Substitution cipher
- Let messages be all lower case from a through z (no spaces or punctuation).
- Represent letters by numbers from 0 to 25.
- Encryption function
  - Ci = E(Pi) = Pi + K (mod 26)
  - K is secret key
- Decryption is
  - Pi = D(Ci) = Ci - K (mod 26)
- **Security of Caesar Cipher**
  - **Easy to brute force: size of key-space is 26**

Historical Ciphers: One-Time Pad (OTP)

- Invented in 1917
- Impossible to crack
- The secret key (the cipher), with random data, must be the same length as the plaintext
- Assume "A" = 0, "B" = 1, "C" = 2, etc.
- Simple to use: just XOR, modular addition
  - https://www.khanacademy.org/computing/computer-science/cryptography/modarithmetic/e/modular-addition
- Encryption: addition, mod 26
- Decryption: subtraction, if result is <0, add 26 and mod 26
- Rather impractical
  - keys are never re-used (one-time)
  - First to generate such key material and second to share it with your correspondent(s)

One Time Pad – Unconditional Security.

- Plaintext is binary string and key is binary string of equal length then encryption can be done by a simple xor operation.

Plaintext: 01010000010001010011

Key: 11010101001001100111

Ciphertext: 10000101011000110100

- If the key is random and is not re-used, then such a system offers unconditional security – perfect secrecy!
- Intuitively perfect secrecy can be seen from the fact that given any plaintext and ciphertext, there is a key which maps the selected plaintext to the selected ciphertext. So given a ciphertext, we get no information whatsoever on what key or plaintext could have been used.
- How do we obtain “random” bit-strings for shared secret keys?
- System is not practical.

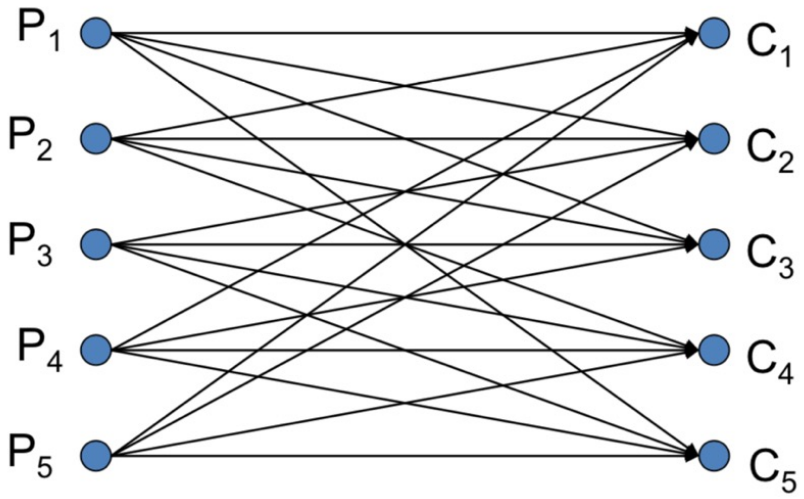
More Cryptosystem Principles:

- Open Design (**Kerckhoffs' principle**)
  - Keep everything public, except the key
  - Made assumption that attacker knows how the cryptosystem algorithms work, only the key must be protected
- Perfect secrecy
  - Cipher text conveys no information about plain text
  - Can’t do cryptanalysis or brute force attacks

Cryptosystem Principles: Perfect Secrecy

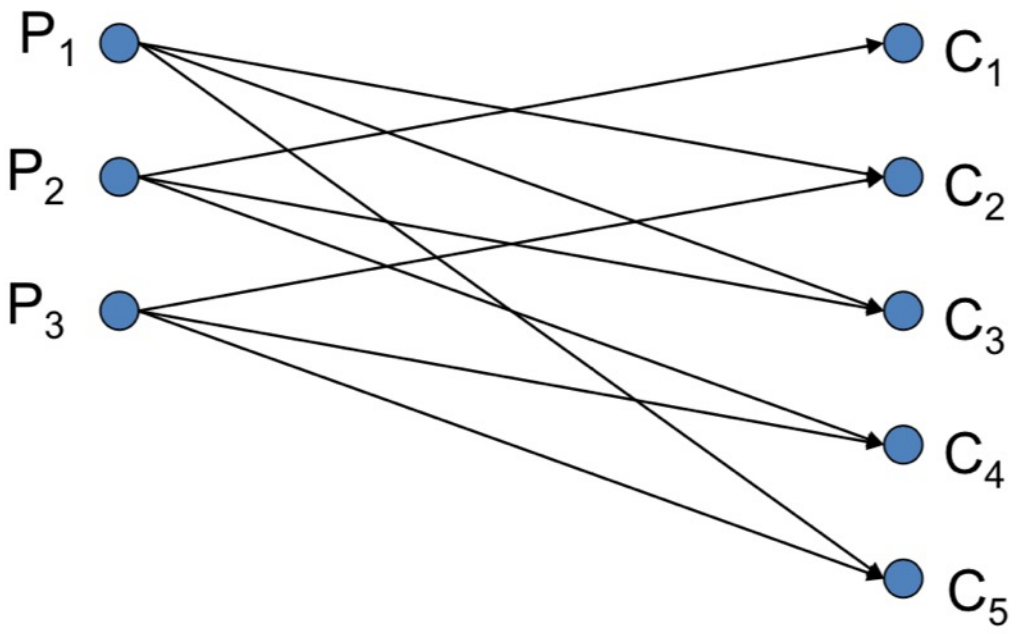
- **Given a ciphertext, cryptanalysis cannot reduce uncertainty**
- Property can be formulated in more mathematically rigorous manner.
  - e.g. Pr(p = M | c = C) = Pr(p = M) where
  - Pr(p = M) is the probability that the actual message (p) is a particular message M. Think of M as an element in a set of messages {P1,P2,P3,P4,P5} then Pr(p = “P1”) = .1, Pr(p = “P2”) = .1, Pr(p = “P3”) = .1, Pr(p = “P4”) = .1, Pr(p = “P5”) = .6
  - Pr(p = M | C = c) is the probability that the actual message (p) is M knowing that the ciphertext is a particular ciphertext – think of C as an element in the set of all ciphertexts {C1, C2, C3, C4, C5}

Four possible keys K1, K2, K3, K4



Cryptosystem Principles: Imperfect Secrecy

- Given C1 we know Message is P2!!
  - Pr(p = P2 | c = C1) = 1
- Given C5, only one bit of uncertainty.
  - Pr(p = P1 | c = C5) = .5
  - Pr(p = P3 | c = C5) = .5

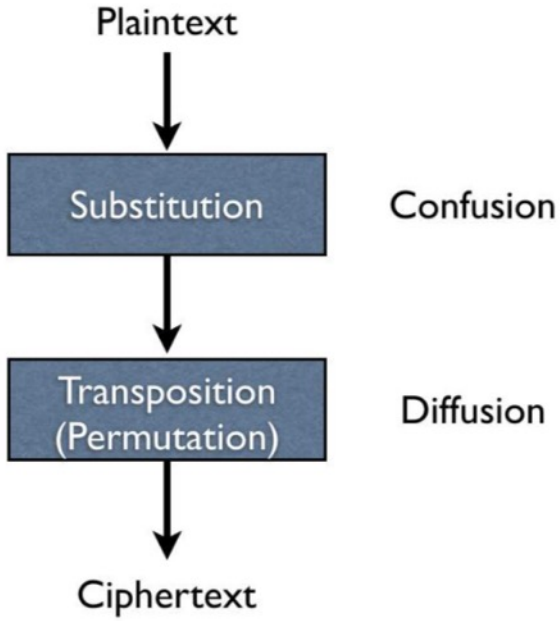


Cryptosystem Principles: Confusion and Diffusion

- Confusion: Changes in the key should affect many parts in the ciphertext.
  - “The method of confusion is to make the relation between the simple statistics of E and the simple description of K a very complex and involved one.”
- Diffusion: The cryptanalyst should not be able to predict what changing one character in the plaintext will do to the ciphertext.
  - “In the method of diffusion the statistical structure of M which leads to its redundancy is “dissipated” into long range statistics—i.e., into statistical structure involving long combinations of letters in the cryptogram. The effect here is that the enemy must intercept a tremendous amount of material to tie down this structure, since the structure is evident only in blocks of very small individual probability.”

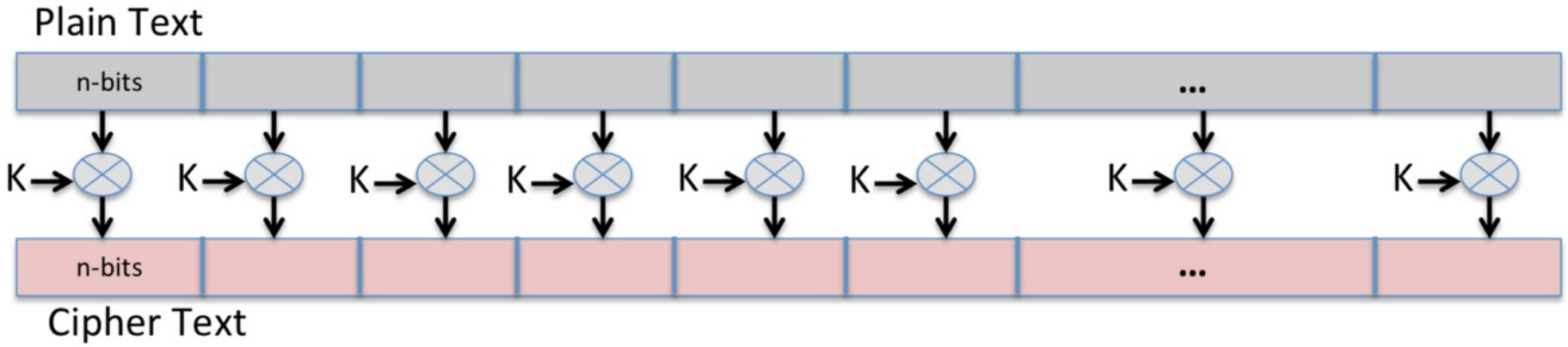
Modern Symmetric Encryption

- Symmetric – same key used for enciphering and deciphering
- Combine multiple “rounds” of Substitution and Transposition
- Approaches:
  - Block cipher
  - Stream cipher



Block Ciphers

- Breaks plaintext into *n-bit* blocks, and then encrypts each block individually



- Elements
  - Key - random string of bytes, (56 < # of bits < 256 bits)
  - Encryption algorithm - Continued rounds of:
    - Substitutions
    - Transpositions
  - Decryption algorithm – inverse process of encryption

DES – Data Encryption Standard

- Encrypts by series of substitution and transpositions.
- Based on Feistel Structure
- Worldwide standard for more than 20 years.
- Has a history of controversy.
- Designed by IBM (Lucifer) with later help (interference?) from NSA.
- No longer considered secure for highly sensitive applications.
- Replacement standard AES (advanced encryption standard)

DES Security

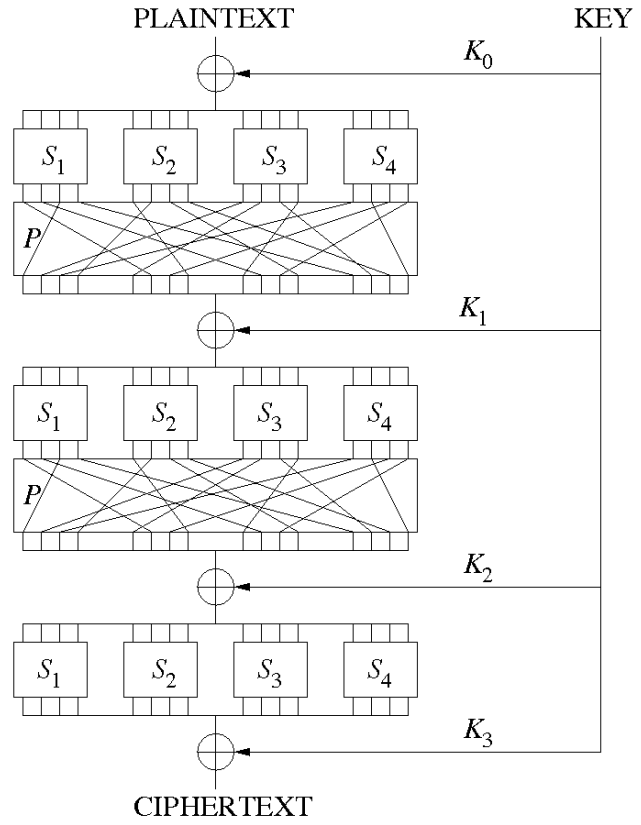
- S-Box design not well understood (secret).
- Has survived some recent sophisticated
- attacks (differential cryptanalysis)
- Key is too short. Hence is vulnerable to brute force attack.
- 1998 distributed attack took 3 months.
- $1,000,000 machine will crack DES in 35 minutes – 1997 estimate. $10,000 – 2.5 days.

Advanced Encryption Standard (AES)

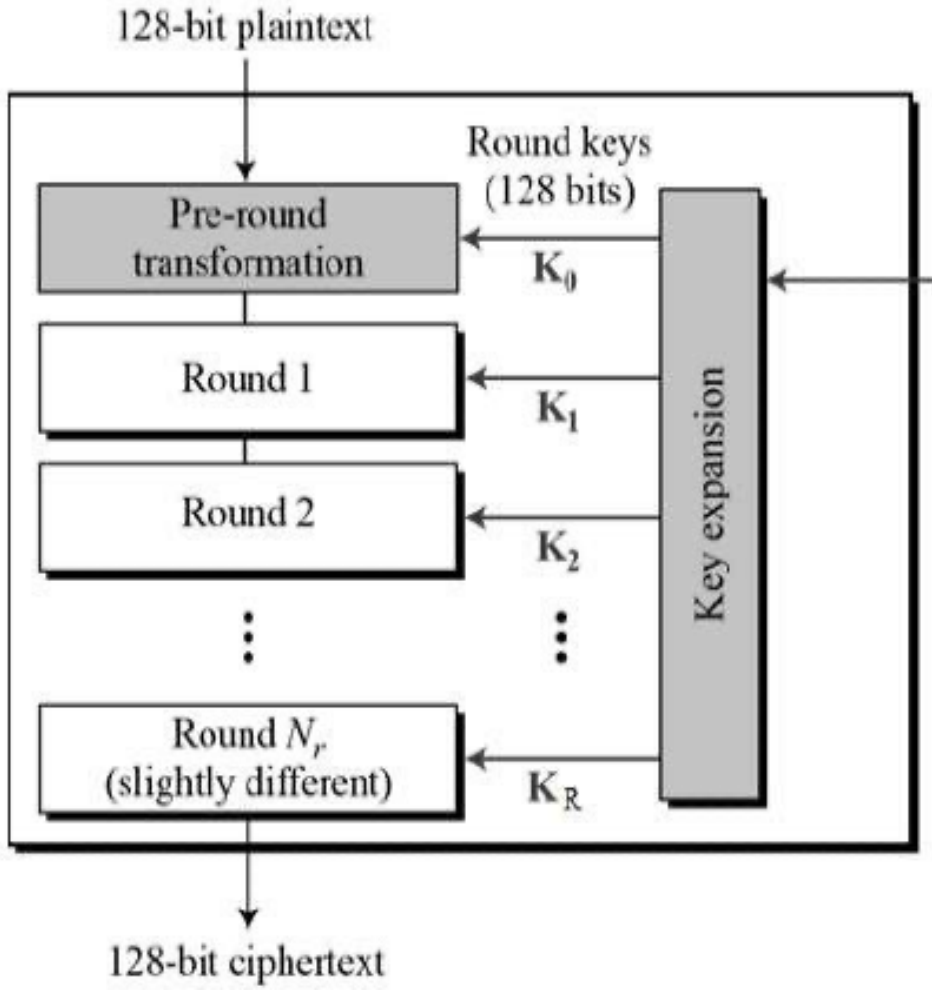
- National Institute of Science and Technology
  - DES is an aging standard that no longer addresses today's needs for strong encryption
  - Not meant to be long term solution!
- AES: The Advanced Encryption Standard
  - Finalized in 2001 - Really called "Rijndael cipher", developed by Joan Daemen and Vincent Rijmen
  - Goal – To define Federal Information Processing Standard (FIPS) by selecting a new powerful encryption algorithm suitable for encrypting government documents
  - AES candidate algorithms were required to be:
    - Symmetric-key, supporting 128, 192, and 256 bit keys
    - Royalty-Free
    - Unclassified (i.e. public domain)

AES: Substitution–permutation network

- A sketch of a substitution-permutation network with 3 rounds, encrypting a plaintext block of 16 bits into a ciphertext block of 16 bits. The S-boxes are the S_i's, the P-boxes are the same P, and the round keys are the K_i's.



AES structure

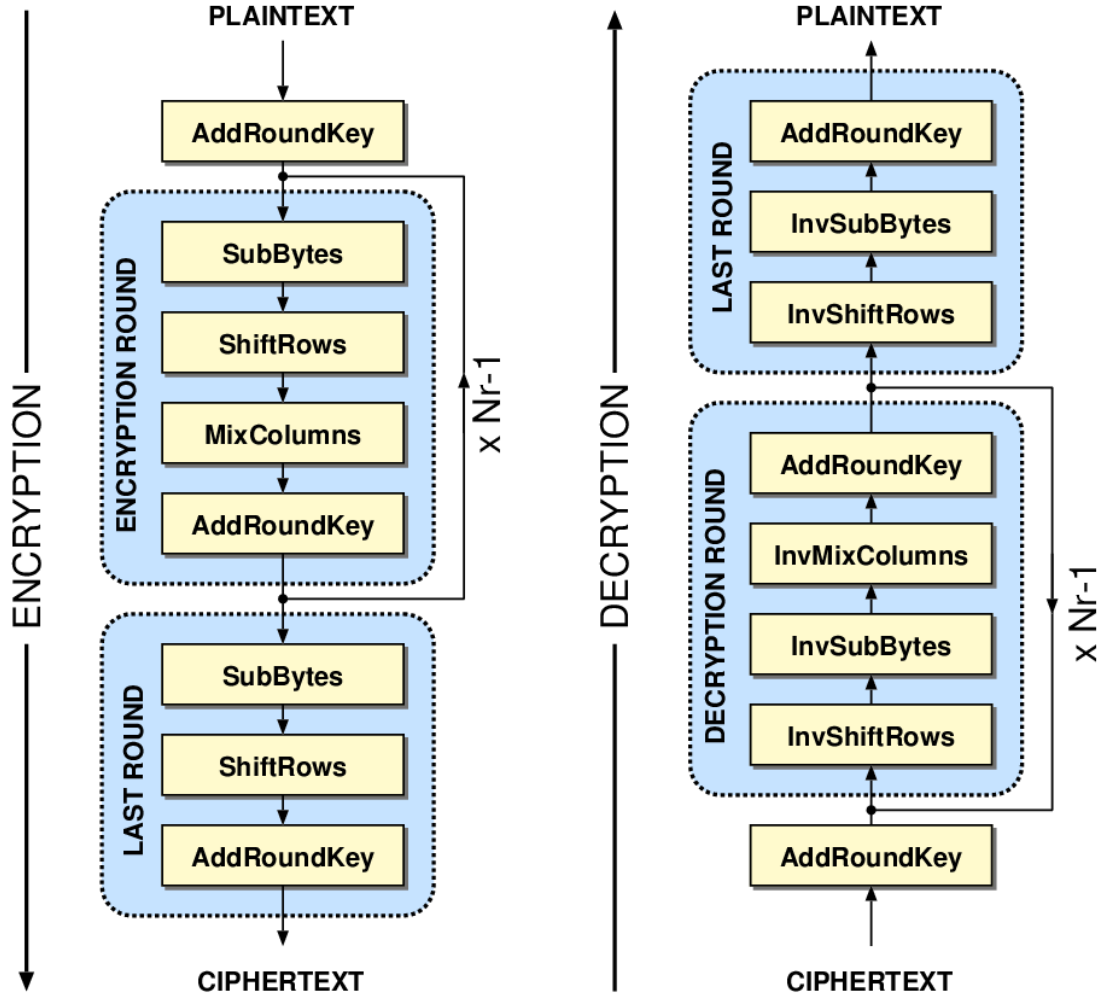


Cipher key
(128, 192, or 256 bits)

R	Key size
10	128
12	192
14	256

Relationship between number of rounds(R) and cipher key size

AES structure



How secure is DES/AES?

- Larger key length
  - 56 bit key = 2^{56} (7.2 x 1016) possible keys
  - 128 bit key = 2^{128} (3.4 x 1038) possible keys
  - 256 bit key = 2^{256} (1.2 x 1077) possible keys
- How big is 2^{128}???
  - 340,282,366,920,938,463,463,374,607,431,768,211,456

    Or
  - “three hundred forty undecillion, two hundred eighty-two decillion, three hundred sixty-six nonillion, nine hundred twenty octillion, nine hundred thirty-eight septillion, four hundred sixty-three sextillion, four hundred sixty-three quintillion, three hundred seventy-four quadrillion, six hundred seven trillion, four hundred thirty-one billion, seven hundred sixty-eight million, two hundred eleven thousand, four hundred fifty-six”

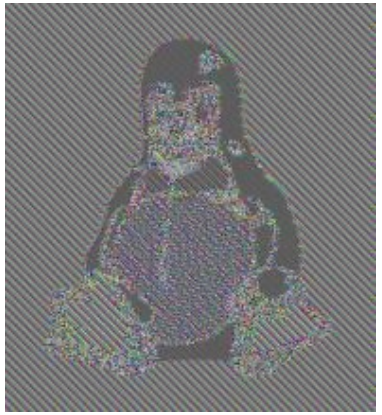
Brute force decryption time

- Currently 33,862,700 GFlops/s or 3.4×10^{16}
  - Assume 1000 operations per decryption
  - Decryptions /second: 3.4×10^{13}
  - Seconds in year: 3.15×10^{7}
  - Decryptions/year: ~1.1×10^{21}
    - 3.13×10^{13} microseconds/year
- Decryption times:
  - DES (56 bit) = $2^{56}/(3.4 \times 10^{13})$ = ~35 minutes
  - AES (128 bit) = $2^{128}/(1.1 \times 10^{21})$ = 3.1×10^{17} years
  - AES (256 bit) = $2^{256}/(1.1 \times 10^{21})$ = 1.1×10^{56} years

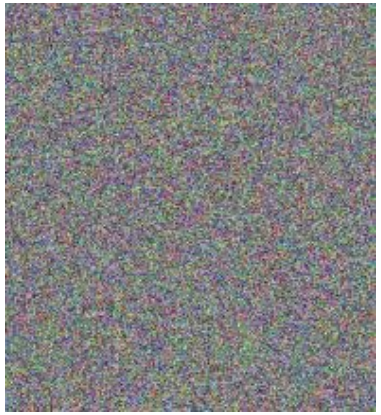
Block Chaining

- How do we handle messages longer than the block size
- Example
  - Electronic Code Book (ECB) – each block encrypted independently from previous previous block
    - Similar block encrypt to same output
  - Cipher Block Chaining (CBC) – each block XORs the output of the previous block
    - Similar block encrypt to different output

Plaintext

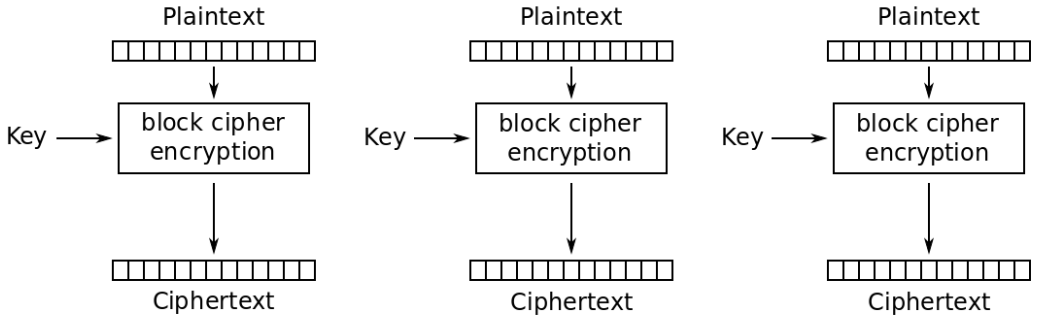


128bit AES-ECB

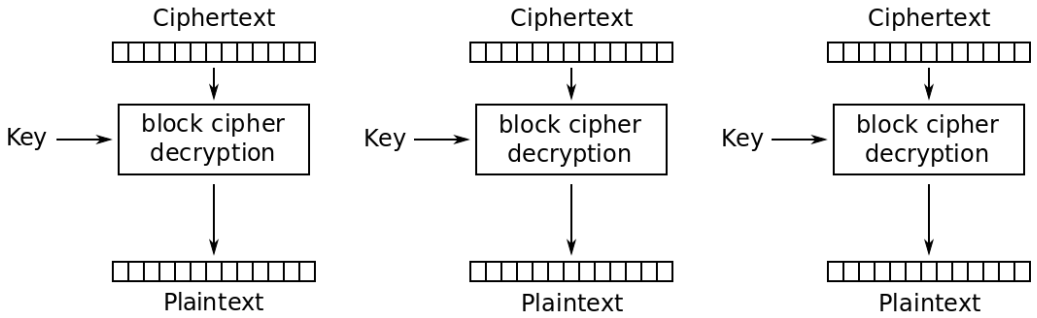


128bit AES-CBC

Electronic Codebook (ECB)

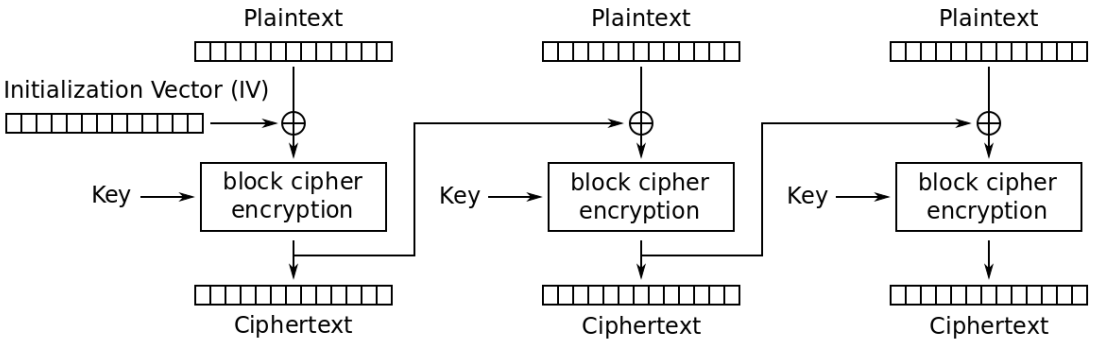


Electronic Codebook (ECB) mode encryption

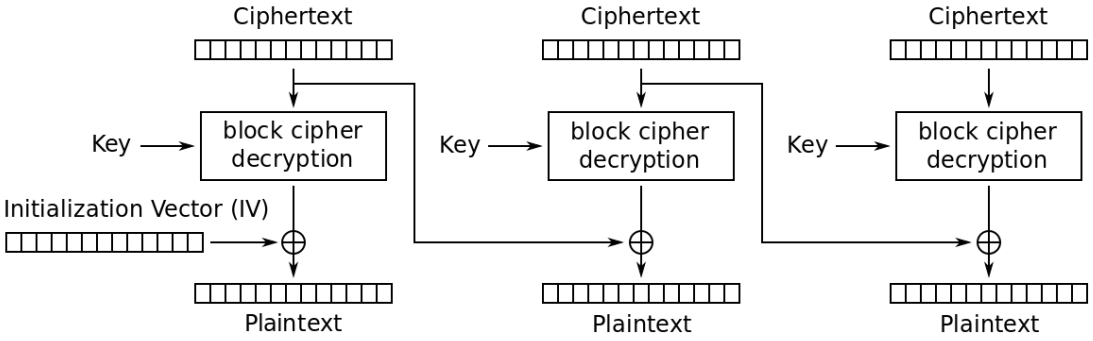


Electronic Codebook (ECB) mode decryption

Cipher Block Chaining (CBC)



Cipher Block Chaining (CBC) mode encryption



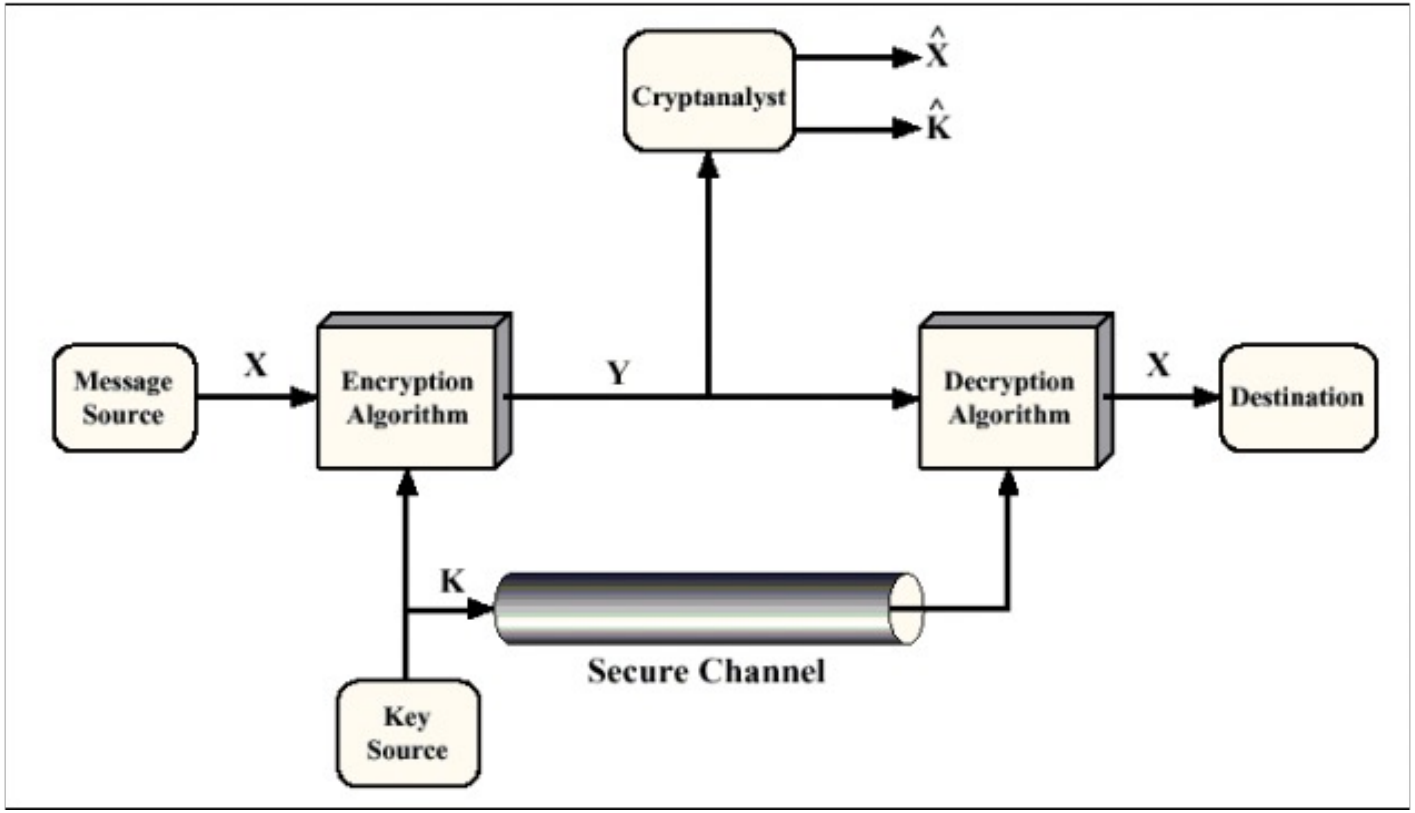
Cipher Block Chaining (CBC) mode decryption

Outline

- Symmetric / Private Key cryptography
- Asymmetric / Public Key cryptography
- Hash functions
- Cryptographically suitable pseudorandom number generation
- Message integrity and digital signatures

Private key cryptography revisited

- Good: Efficient
- Bad: Key distribution and management is a serious problem

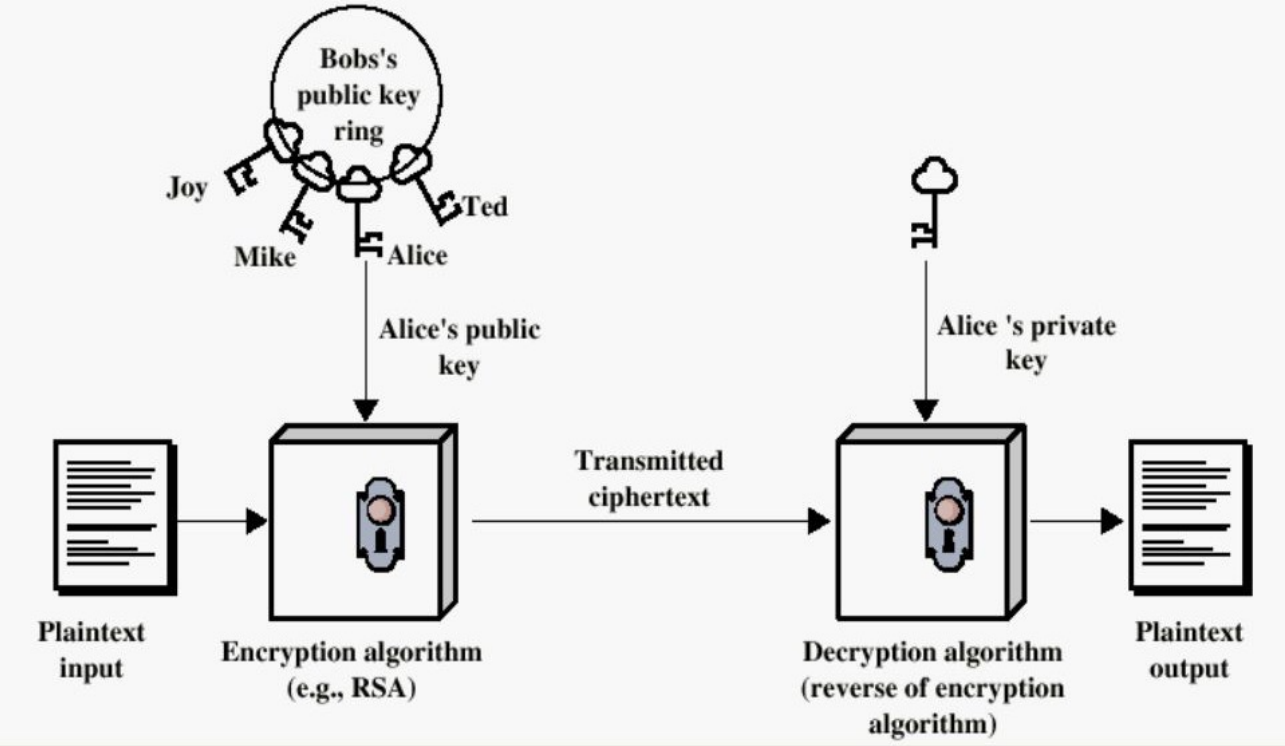


Asymmetric / Public Key cryptography

- Encryption and Decryption use different keys
  - Also called “public key” crypto
- Invented in mid 1970s
- A completely different viewpoint on messages
  - Instead of consisting of symbols that could be substituted or rearranged, look at messages as numbers that can added, subtracted, multiplied, divided, exponentiated, etc.
- Based on one-way functions
  - Discrete logarithm
  - Integer factoring
  - Elliptic curve arithmetic

Asymmetric / Public Key cryptography

- Good: Key management problem potentially simpler
- Bad: Much slower than symmetric/private key crypto



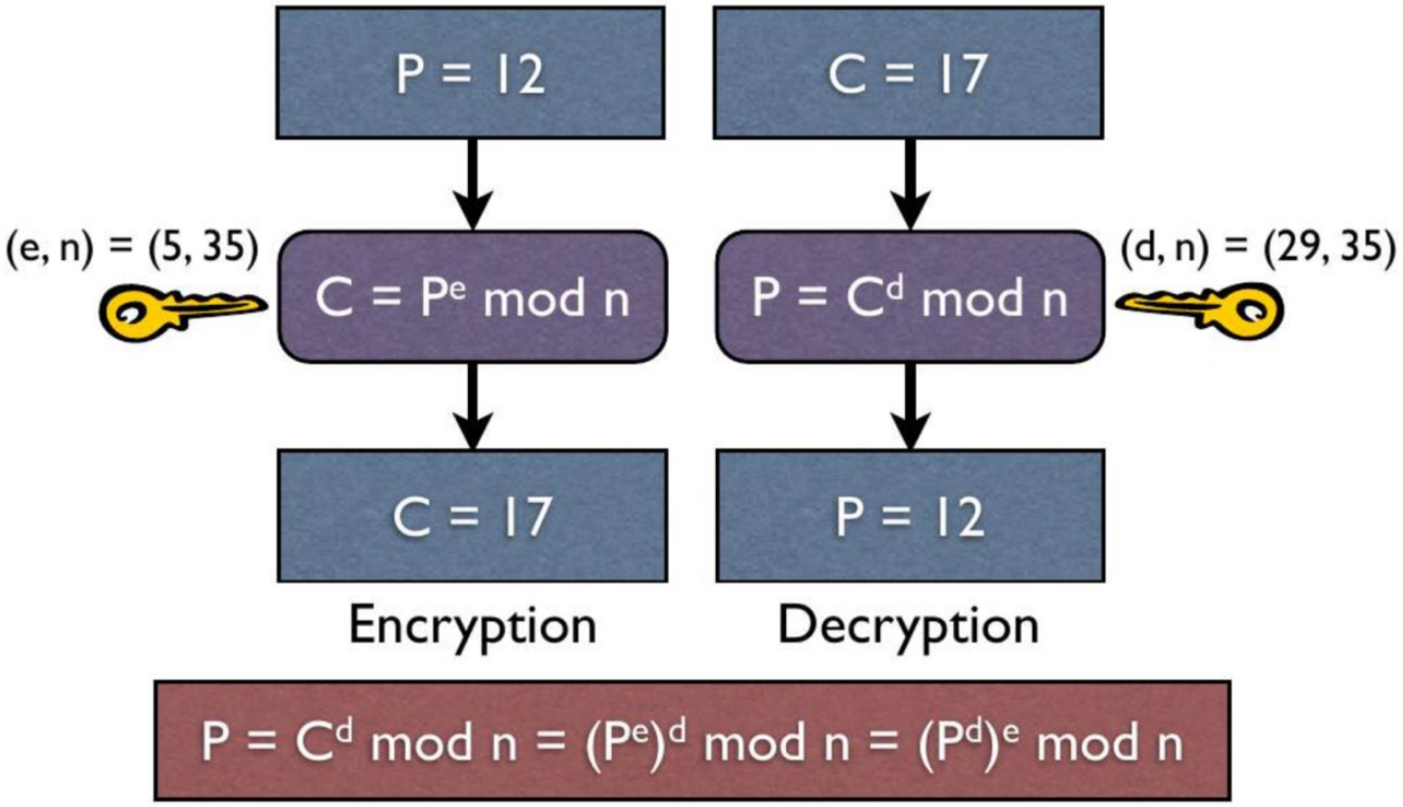
Encryption and 1-way trap doors

- Two keys:
  - public encryption key e
  - private decryption key d
- Encryption easy when e is known
- Decryption hard when d is not known
- d provides “trap door”: decryption easy when d is known
- We’ll study the RSA public key encryption scheme, but skip the theory.

One-way functions and trapdoors

- A function f() is said to be one-way if given x it is “easy” to compute y = f (x), but given y it is “hard” to compute $x = f^{-1}(y)$.
- A trap-door one-way function $f_K()$ is such that to compute
  - $y = f_K(x)$ is easy if K and x are known.
  - $x = f^{-1}{}_K(y)$ is easy if K and y are known.
  - $x = f^{-1}{}_K(y)$ is hard if y is known but K is unknown.
- Given a trap-door one-way function one can design a public key cryptosystem.

RSA overview – Mechanics



RSA Keys

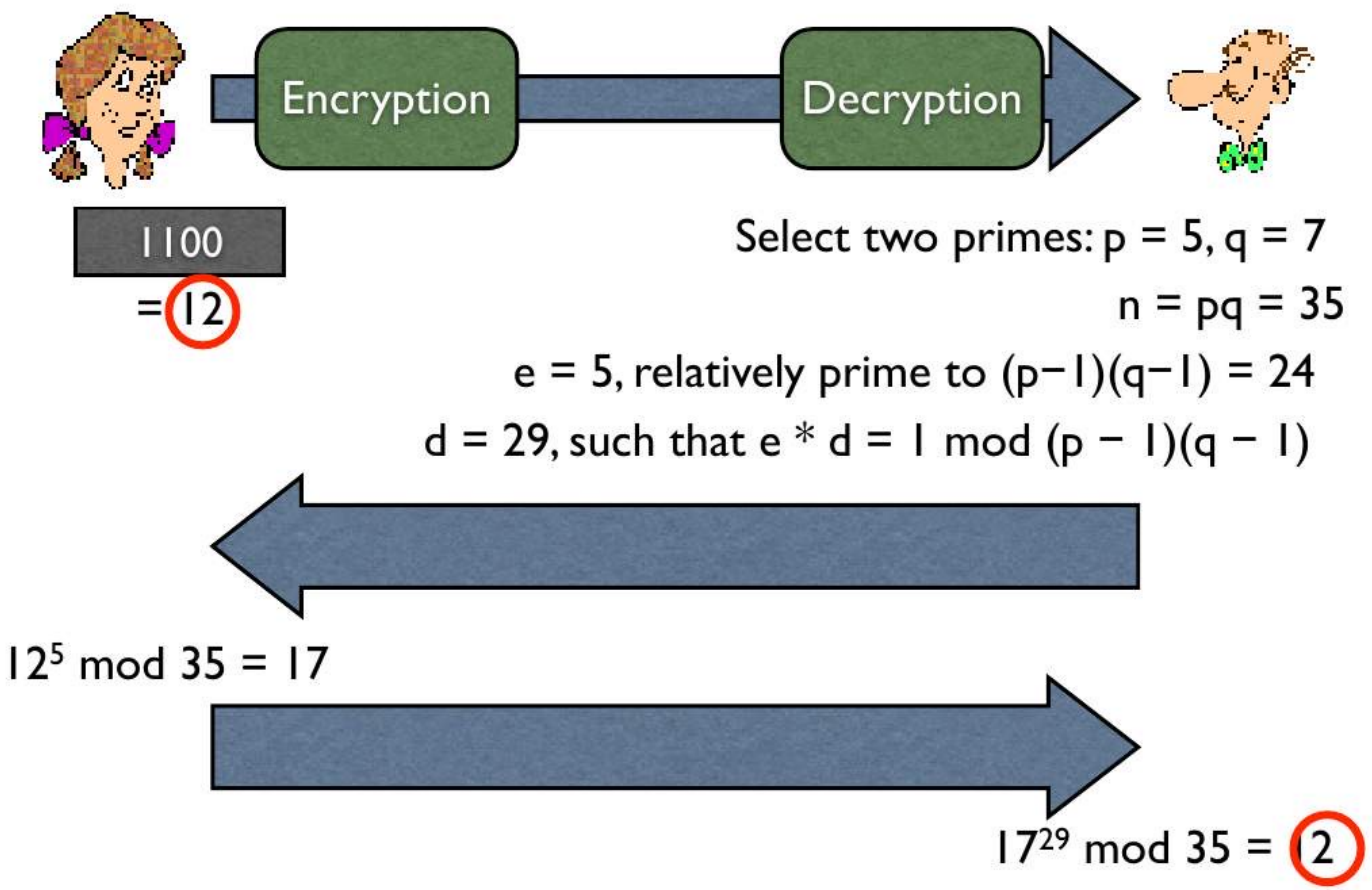
(e, n) = (5, 35)

(d, n) = (29, 35)

- n: a product of two primes, p and q
  - >200 digits long (512 bits)
- e: relatively prime to $(p - 1)(q - 1)$
- d: $e * d = 1 \bmod (p - 1)(q - 1)$

$$5 * d = 1 \bmod 24$$

RSA Example



(5,35) is a ***public*** key; anyone, can encrypt with it

(29,35) is Bob's ***private*** key; only he is able to decrypt messages created with the public key

Security of RSA: RSA assumption

- Suppose Eve intercepts the encrypted message y that Bob has sent to Alice.
- Eve can look up (e,n) in the public directory (just as Bob did when he encrypted the message)
- If Eve can compute d = e-1 mod n then she can use it to recover the plaintext x.
- If Eve can compute factor n into p and q, she can compute d (the same way Alice did).
  - Factoring large numbers is hard.

Security and Performance

How big should n be?

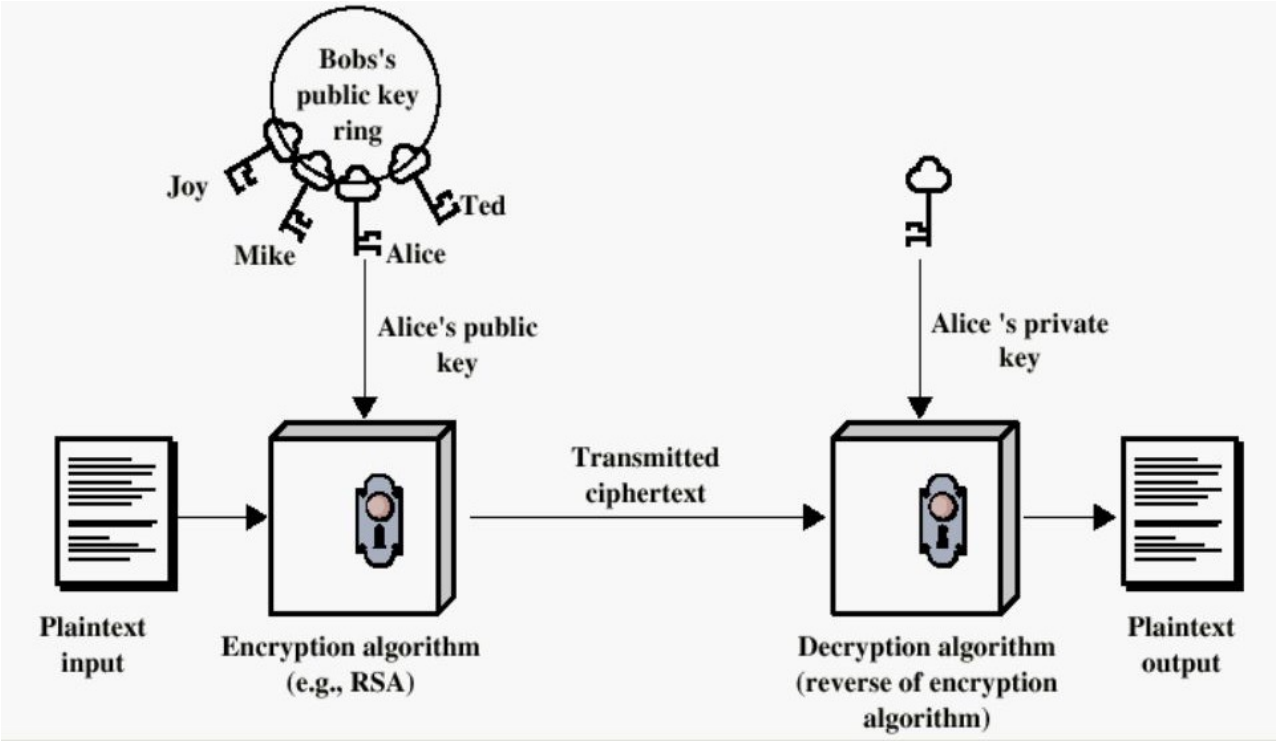
- Today we need n to be at least 1024-bits
  - This is equivalent to security provided by 80-bit long keys in private-key crypto
- RSA (and other asymmetric crypto*) is comparatively slow
  - Asymmetric cryptosystems are many times more computationally expensive than good symmetric systems

*DSA, ElGamal, Pailliar (Homomorphic) , Cramer-Shoup

Asymmetric / Public key crypto summary

Key Points

- Keygen(L) → K_{pub}, K_{priv}
- E(B_{pub}, M) → C
- D(B_{priv}, C) → M
- Can't compute B_{priv} given B_{pub}!!!



Outline

- Symmetric / Private Key cryptography
- Asymmetric / Public Key cryptography
- Hash functions
- Cryptographically suitable pseudorandom number generation
- Message integrity and digital signatures

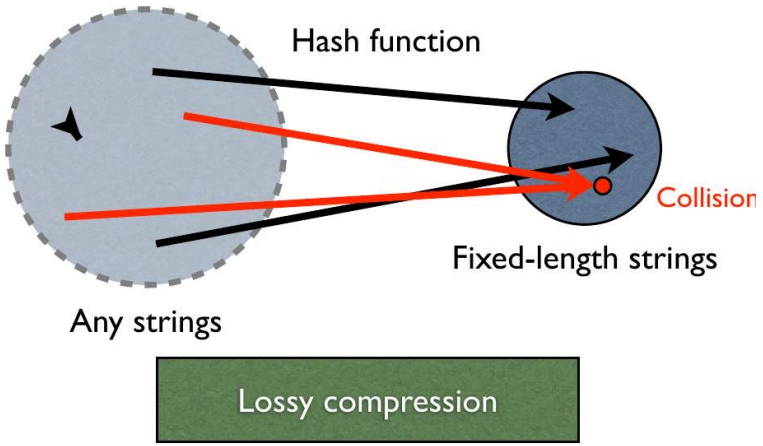
Cryptographic Hash Functions

Requirements of cryptographic hash functions:

- Can be applied to data of any length.
- Output is fixed length
- Relatively easy to compute h(x), given x and deterministic
- Infeasible to get x, given h(x). **One-way property**
- Given x, infeasible to find y such that h(x) = h(y). **Weak-collision resistance property**
- Infeasible to find any pair x and y such that h(x) = h(y). **Strong-collision resistance property**

Cryptographic Hash Functions

- Hash function: one way function mapping a variable length string to fixed length digest
- Common functions: SHA(1,2,3), MD5
- Used for: Digital Signatures, HMACs, Password storage
- Three key requirements



Hash functions examples

- Unix commands

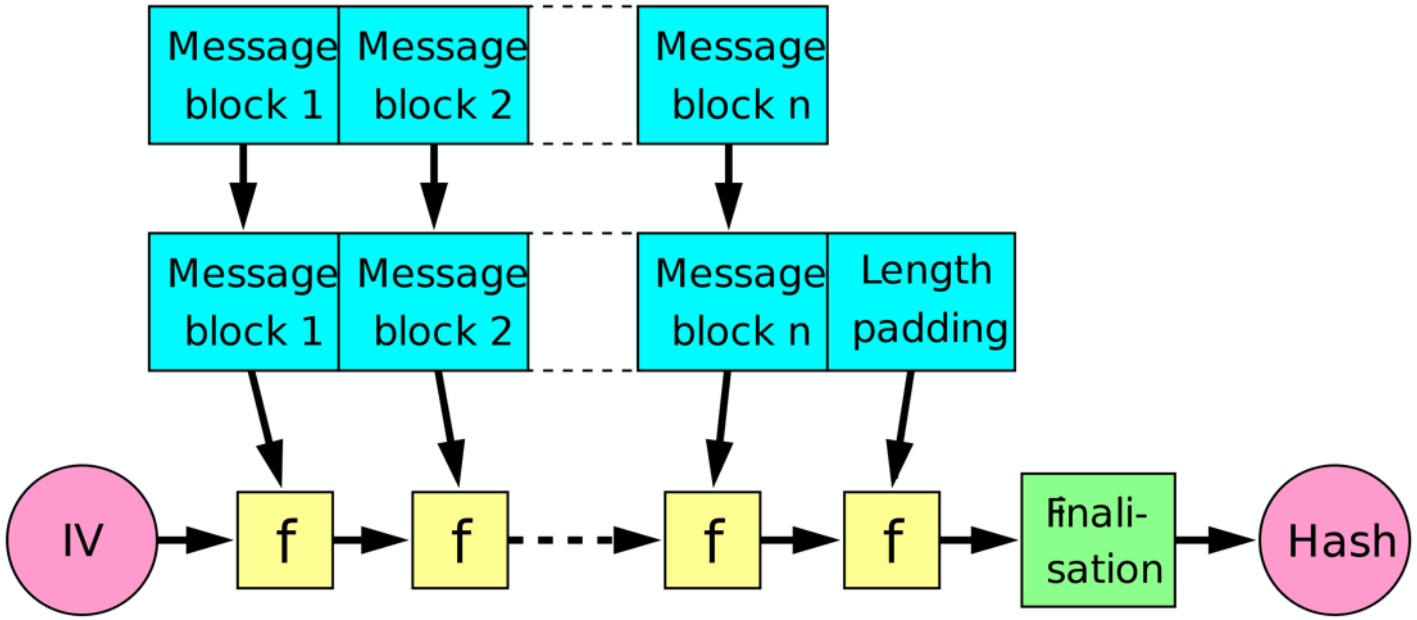
echo "This is a test message to demonstrate hash functions" | sha1sum
> 799772e0410af26e43ee279af9f630150c7bb8bb

echo "This is a test message to demonstrate hash functions." | sha1sum
> 5bd891f72d006b2eaff8b0ad36457fe7cce6f21c

echo "This is a tset message to demonstrate hash functions" | sha1sum >
d3e266656a24a986e6303eccab4474447b887def

echo "This is a test message to demonstrate hash functions" | sha512sum
>72a26cb6e857cecbaf6b2b88bfaebd680de0551e091e87e7f822c10933ce58c696356d88124a114ff1
eec089a3a2e8f2cecb0916c567f8 20c12d07c598d790c5

Hash function design - The Merkle–Damgård construction



Cryptographic Hash Functions

Key Points

- h(x) → hash
  - hash is fixed length (16,20,32 bytes)
  - h(x) is fast / cheap
- Given hash, cannot compute x
- Given x, cannot find y so that h(x) == h(y)
- Cannot find any x,y so that h(x) == h(y)

Outline

- Symmetric / Private Key cryptography
- Asymmetric / Public Key cryptography
- Hash functions
- Cryptographically suitable pseudorandom number generation
- Message integrity and digital signatures

Random Number Generation

- Could be as simple as dice, coin flipping, the shuffling of playing cards etc.
- A sequence of random numbers, R1, R2, R3, must have two important properties:
  - Uniformity, i.e. they are equally probable every where
  - Independence, i.e. the current value of a random variable has no relation with the previous values
- Problem - Most methods fail to produce true random numbers
- Solution- Pseudorandom number generator

Pseudorandom number generator

- An algorithm for generating a sequence of numbers that approximates the properties of random numbers.
- The sequence is not truly random in that it is completely determined by a relatively small set of initial values, called the PRNG's state, which includes a truly random seed
- A PRNG suitable for cryptographic applications is called a cryptographically secure PRNG (CSPRNG).
- A requirement for a CSPRNG is that an adversary not knowing the seed has only negligible advantage in distinguishing the generator's output sequence from a random sequence.

See: https://cdn.atraining.ru/docs/Intel_TRNG_Report_20120312.pdf

Random Number Generation

Key Point:

- Given a sequence of values from the CSRNG, cannot guess the next value!

Outline

- Symmetric / Private Key cryptography
- Asymmetric / Public Key cryptography
- Hash functions
- Cryptographically suitable pseudorandom number generation
- Message integrity and digital signatures

Authentication

- Message authentication code
- Digital signature

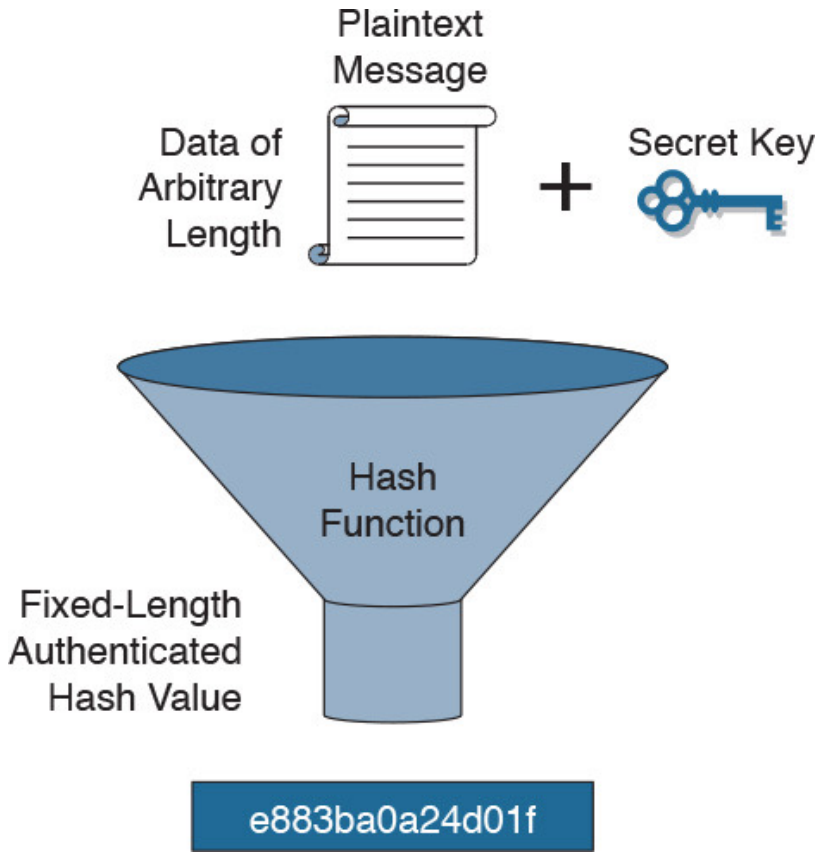
Message authentication code: HMAC

Hash Message Authentication Code (HMAC)

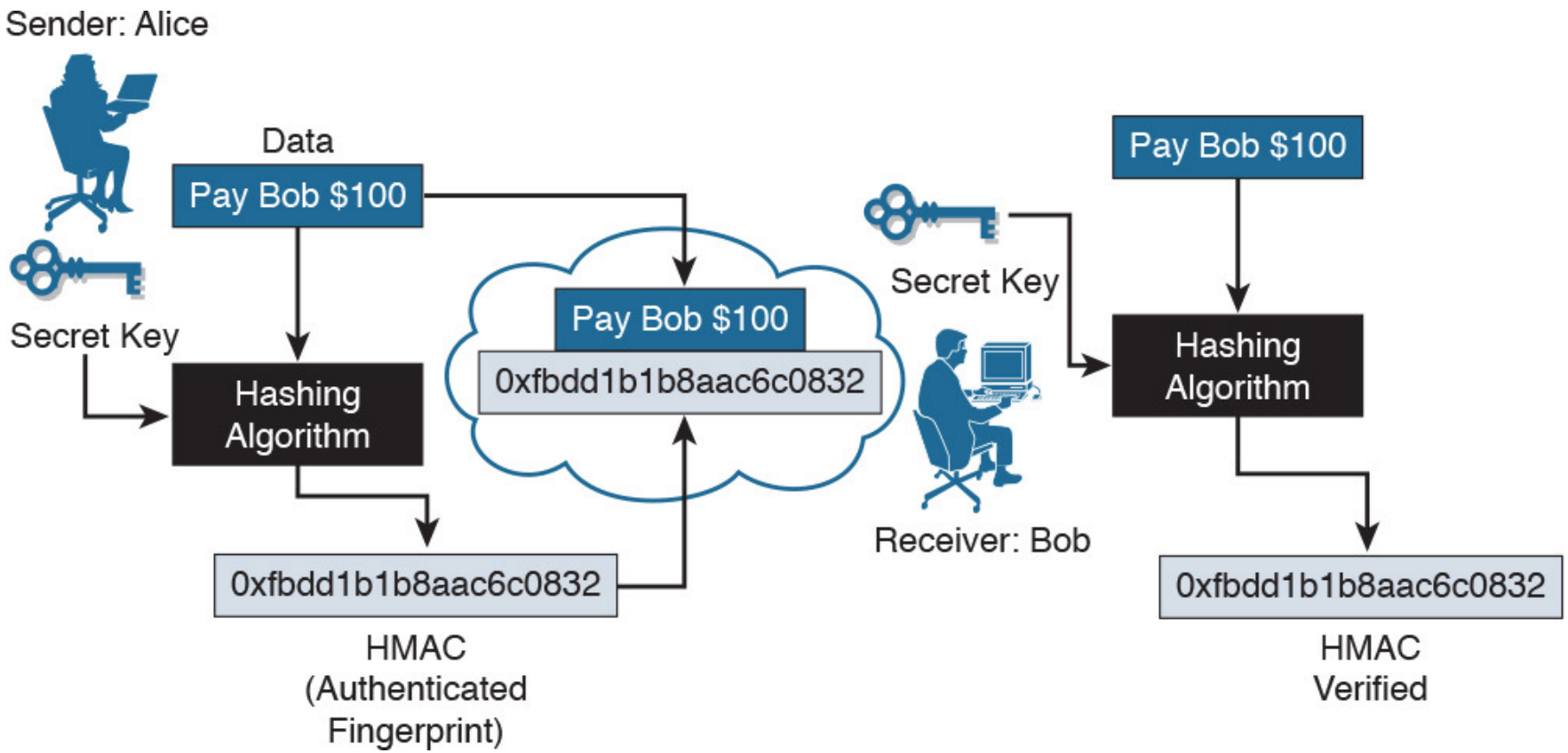
- MAC – cryptographic digest appended to a communication
  - Cannot be manipulated by attack without the key used to create it
- HMAC - MAC based on hash functions
  - Stronger security properties
    - Usually faster than traditional symmetric crypto algs (e.g., DES)
    - Less vulnerable to collisions than just hash functions
      - Used with modern hashes (MD5, SHA, etc)
- HMAC output is appended to message
  - Receiver verifies message by computing the HMAC and compare with the appended HMAC
- HMAC basic example:
  - HMAC(k,m)=H(K||m)
    - Length extension attack
- Better example
  - HMAC(k,m)=H(K||H(K||m))

HMAC usage

- Adds authentication to integrity assurance



HMAC usage


Sender: Alice
Data
Pay Bob $100
Secret Key
Hashing Algorithm
0xfbdd1b1b8aac6c0832
HMAC (Authenticated Fingerprint)
Pay Bob $100
0xfbdd1b1b8aac6c0832
Secret Key
Receiver: Bob
Pay Bob $100
Hashing Algorithm
0xfbdd1b1b8aac6c0832
HMAC Verified

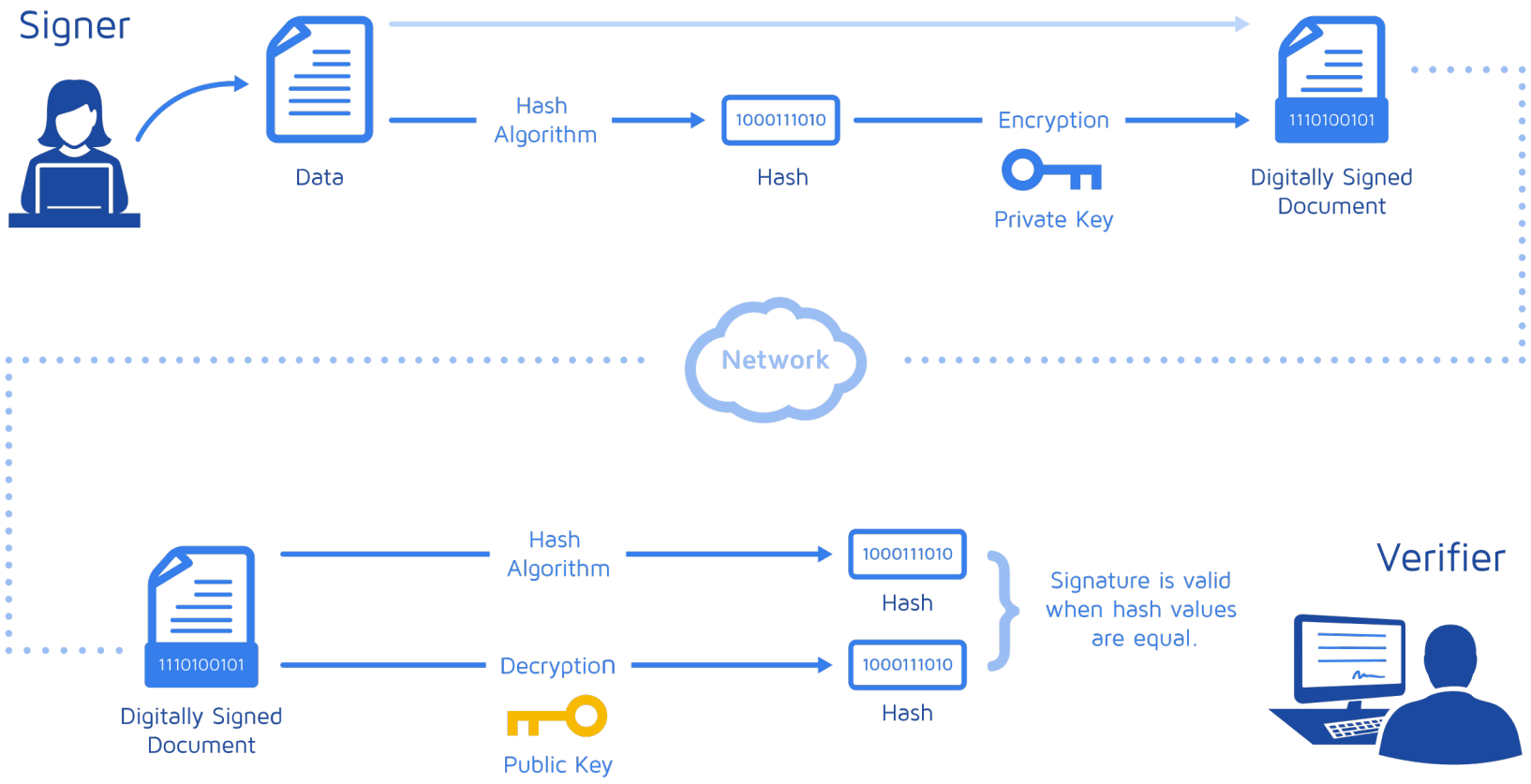
Authentication

- Message authentication code
- Digital signature

Digital Signatures

- Utilize public key crypto in reverse, to provide authenticity
  - Sender encrypts (signs) with private key (usually a message hash)
  - Receive decrypts (verifies) with public key
- Upon successful decryption (*verify*), recipient knows message came from someone holding the corresponding private key
- Anyone can verify (because verifying uses the ***public*** key)

Digital Signatures: Simplified Digital Signature Example



Digital Signatures: Simplified Digital Signature Example

- Alice wants to send a message M to Bob.
- Alice will compute the secure hash of M
  - hash(M) → h
- Alice encrypts this hash with her private key
  - sign(h, A_{priv}) → s
- Alice sends M and s to Bob
- Bob computes the secure hash of M
  - hash(M) → h
- Bob verifies the signature
  - verify(M, s, A_{pub})
  - assert dec(s, A_{pub}) == h

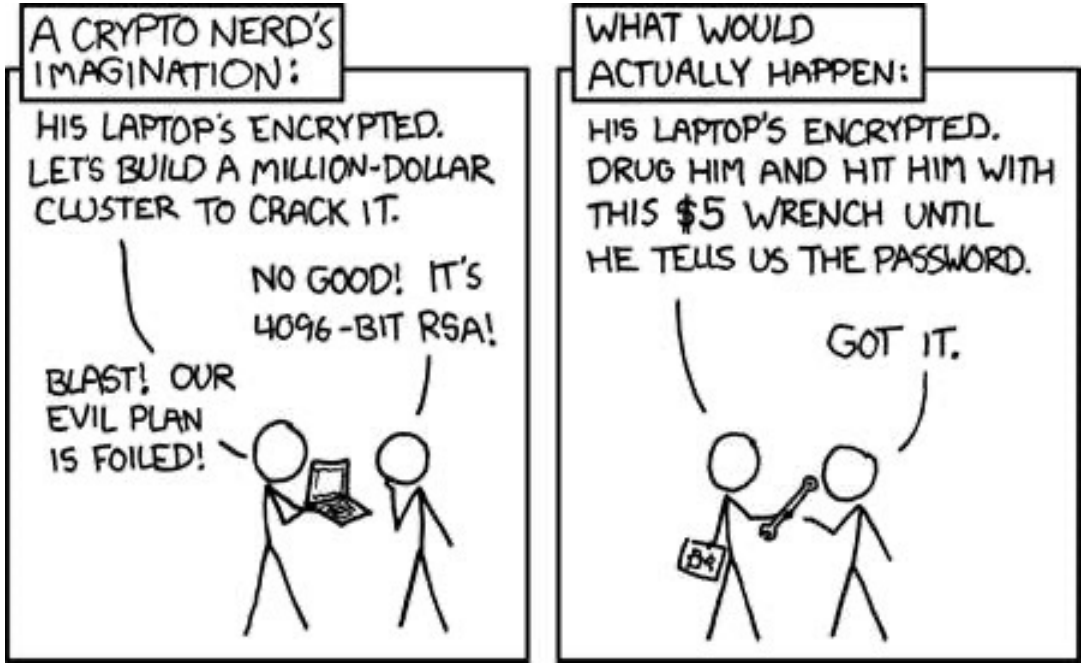
Digital Signatures

Ensuring Integrity+Authentication through Digital Signatures

- Message Integrity
  - Detect if message is tampered with while in the transit
- Source/Sender Authentication
  - No forgery possible
- Non-repudiation
  - If I sign something, I can not deny later
  - A trusted third party (court) can resolve dispute

Crypto Warnings

- Strong crypto != strong security
  - Crypto usually bypassed not broken
- Vulnerabilities in
  - Design
  - Implementation
  - Installation



To Learn More

- https://www.garykessler.net/library/crypto.html
- Birthday paradox (hash functions collisions)
- Cryptography: Theory and Practice – D. Stinson. CRC Press.
- Handbook of Applied Cryptography – Menezes et. al. CRC Press.
- Cryptography and Network Security – William Stallings.
- Applied Cryptography – B. Schneier. John Wiley.
- North American Crypto archive http://cryptography.org/
- Ron Rivest's crypto page http://theory.lcs.mit.edu/~rivest/crypto- security.html
- Cryptography Research Inc. Resource page http://www.cryptography.com/resources/index.html
- Cryptography archive: http://www.austinlinks.com/Crypto/
- AES home page https://csrc.nist.gov/projects/cryptographic-standards-and-guidelines/archived-crypto-projects/aes-development