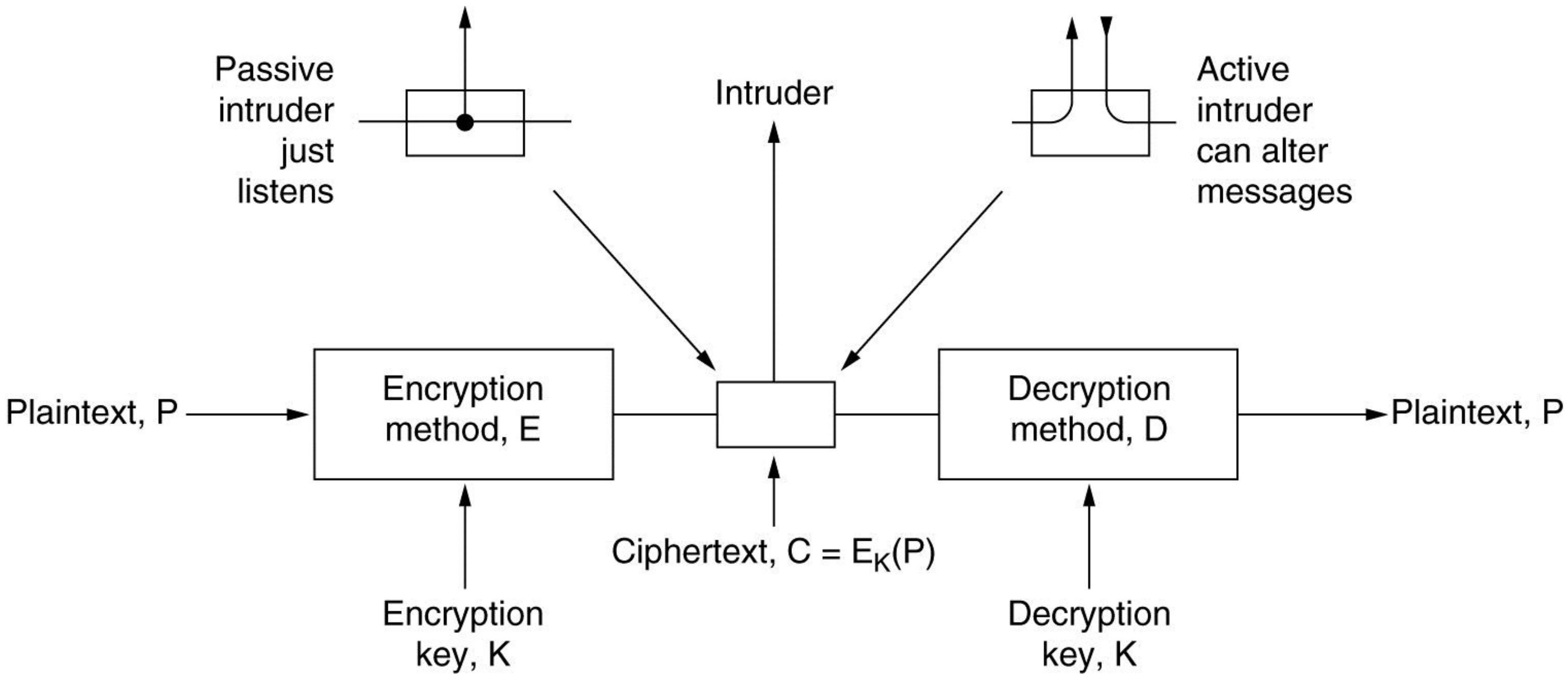
Cryptography

Types of cryptography

- Symmetric key cryptography
  - Involves the use one key
- Public key cryptography
  - Involves the use of two keys
- Hash functions
  - Involves the use of no keys

Symmetric Key Cryptography



Symmetric Encryption and Message Confidentiality

- also known as: conventional encryption, secret-key, or single-key encryption
  - only alternative before public-key crypto in 70’s
  - still most widely used alternative
  - has ingredients: plaintext, encryption algorithm, secret key, ciphertext, and decryption algorithm
- generically classified along dimensions of:
  1. type of operations used
  2. number of keys used
  3. way in which the plaintext is processed

Cryptanalysis

- attacks:
  - ciphertext only - least info, hardest
  - known plaintext - some plain/cipher pairs
  - chosen plaintext - get own plain/cipher pairs
  - chosen ciphertext - rarer
  - chosen text - rarer
- only weak algs fail a ciphertext-only attack
- usually design algs to withstand a known-plaintext attack

Computationally Secure Algs

- encryption is computationally secure if:
  - cost of breaking cipher exceeds info value
  - time required to break cipher exceeds the useful lifetime of the info
- usually very difficult to estimate the amount of effort required to break
- can estimate time/cost of a brute-force attack

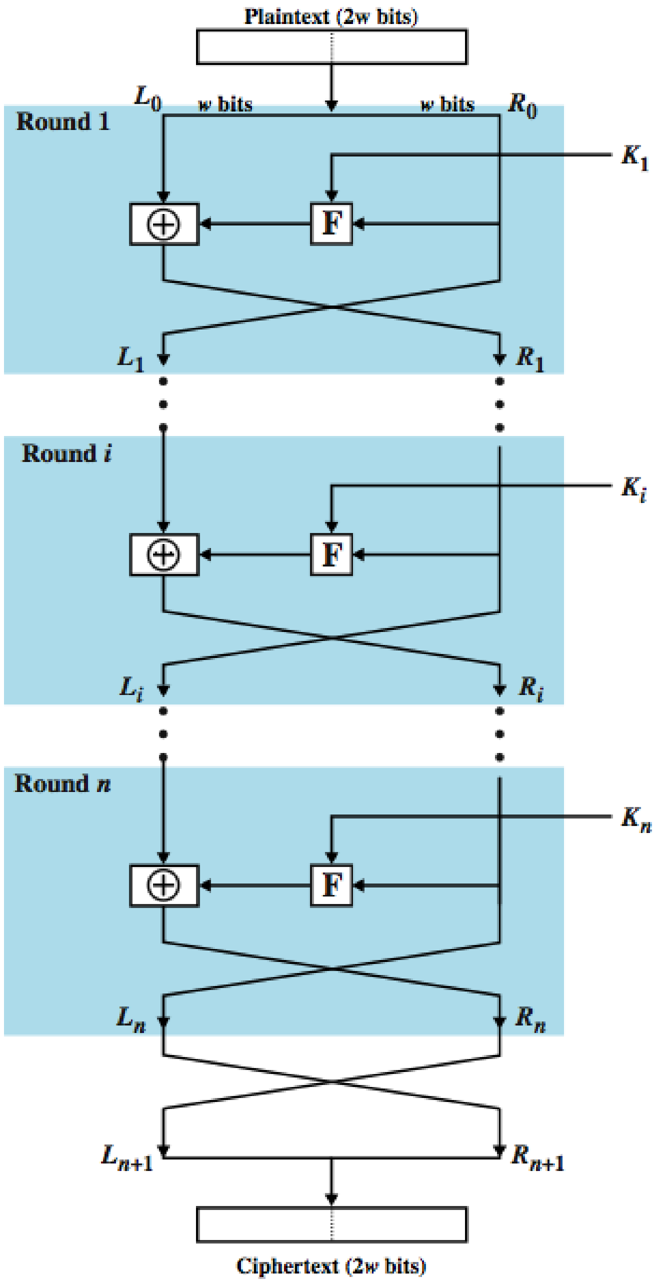
Symmetric Key Cryptography

- Algorithms:
  - DES
    - Modes: ECB, CBC, CFB, OFB, CM
  - 3DES
  - AES
  - IDEA
  - Blowfish
  - RC4
  - RC5
  - CAST
  - SAFER
  - Twofish

Block Cipher Structure

- have a general iterative block cipher structure
  - with a sequence of rounds
  - with substitutions / permutations controlled by key
- parameters and design features:
  - block size
  - key size
  - number of rounds
  - subkey generation algorithm
  - round function
  - also: fast software en/decrypt, ease of analysis

Feistel Cipher Structure



DES (Data Encryption Standard)

- Block Cipher
- Uses a combination of Substitution and Transpositions (permutations)
- Called a Product Cipher
- Goes through 16 cycles
- PlainText is organized into 64-bit Blocks
- Uses a 56-bit Key

DES

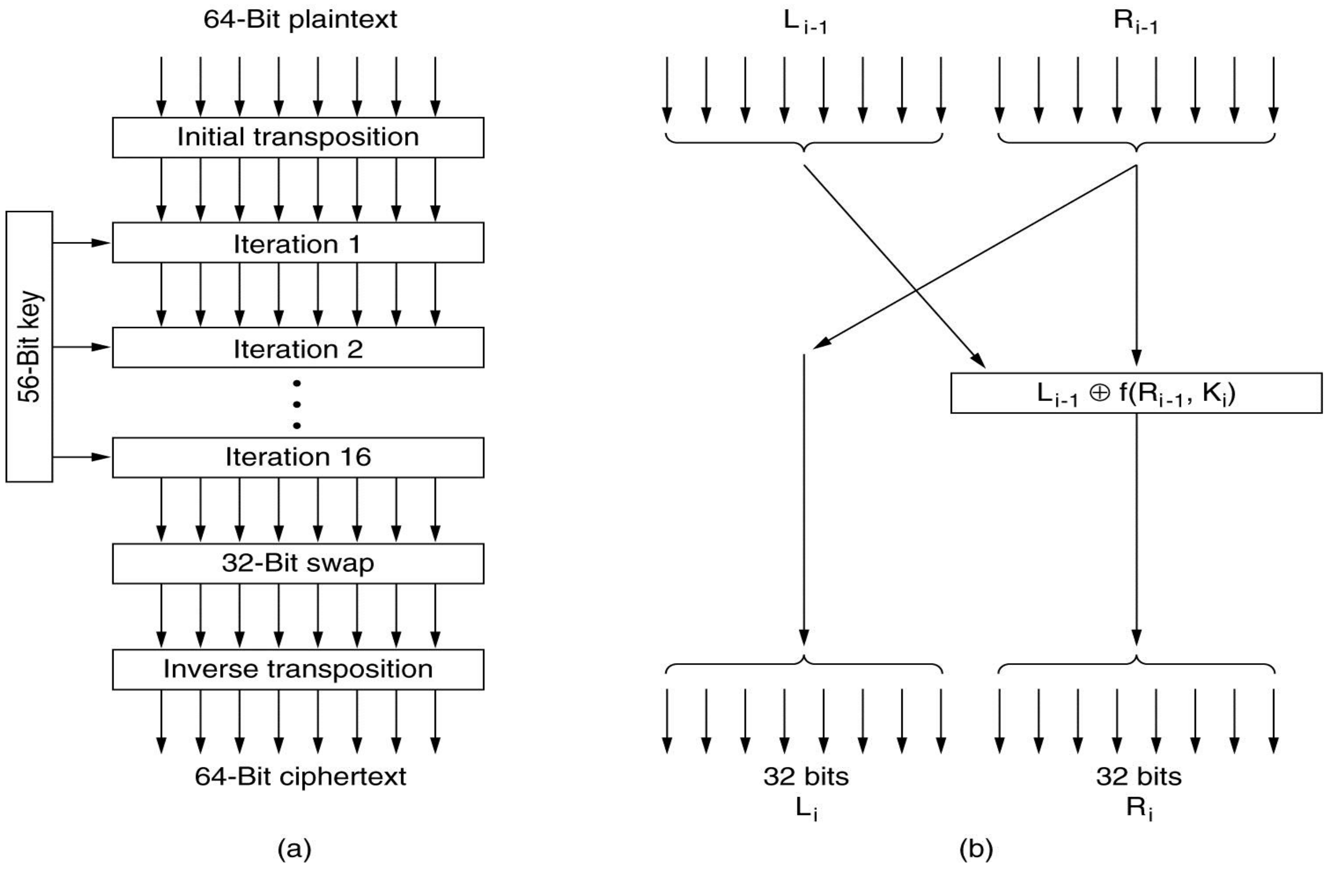
- Initial Permutation on Input Text (64-bit)
- Split into Right and Left Halves (32-bit)
- Take right half and permute it (Expansion Permutation) 48-bit
- Work on Key (shift) 56-bit, then permute key (48-bits)
- XOR resulting key with right half …result is 32-bit (S-BoX)
- Permute result
- XOR result with Left Half
- End of Cycle

DES

- The next cycle begins with:
  - The result of previous cycle as its right half
  - The old Right half (48-bit) as Its left half

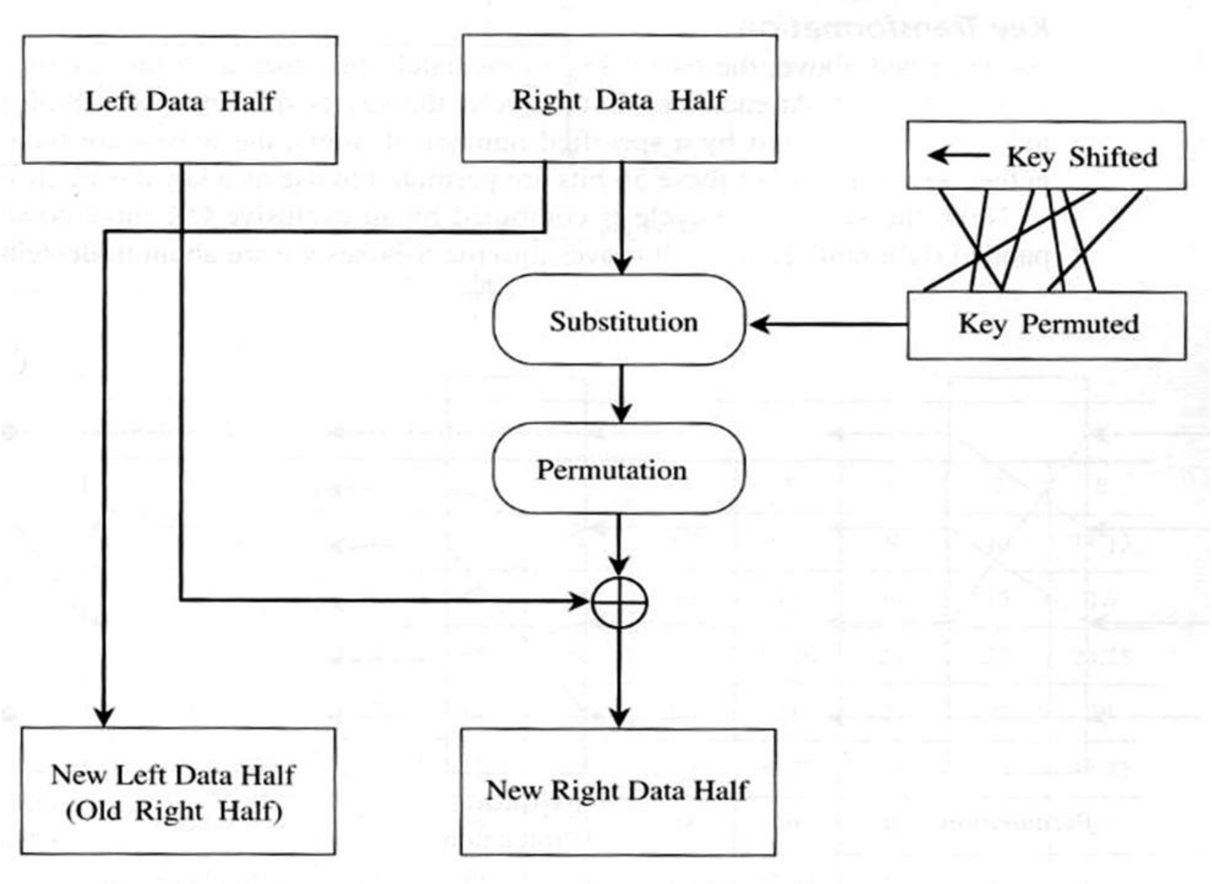
Repeat

DES



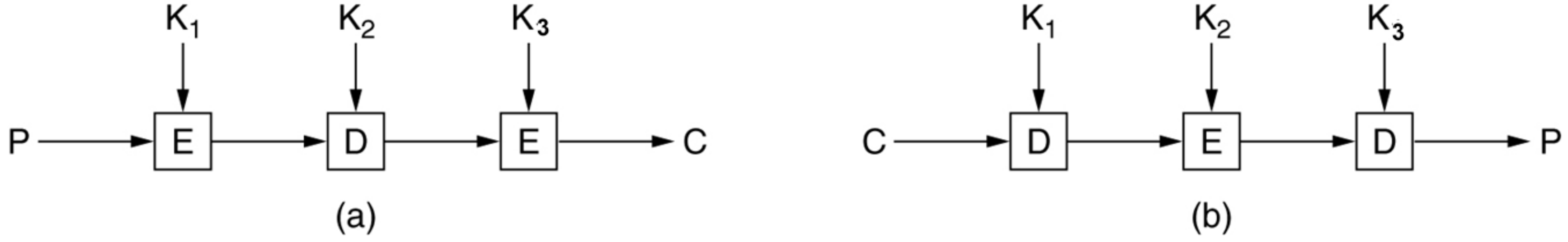
64-Bit plaintext
Initial transposition
56-Bit key
Iteration 1
Iteration 2
Iteration 16
32-Bit swap
Inverse transposition
64-Bit ciphertext
(a)
L_{i-1}
R_{i-1}
$L_{i-1} \oplus f(R_{i-1}, K_i)$
32 bits
L_i
32 bits
R_i
(b)

Details of a cycle in DES



Triple DES

- Uses three DES keys



  - (a)encryption. (b) Decryption
  - Compatible with old DES. If we use the same key instead a different K2.
  - 128 bit key→2^128=3*10^38keys;

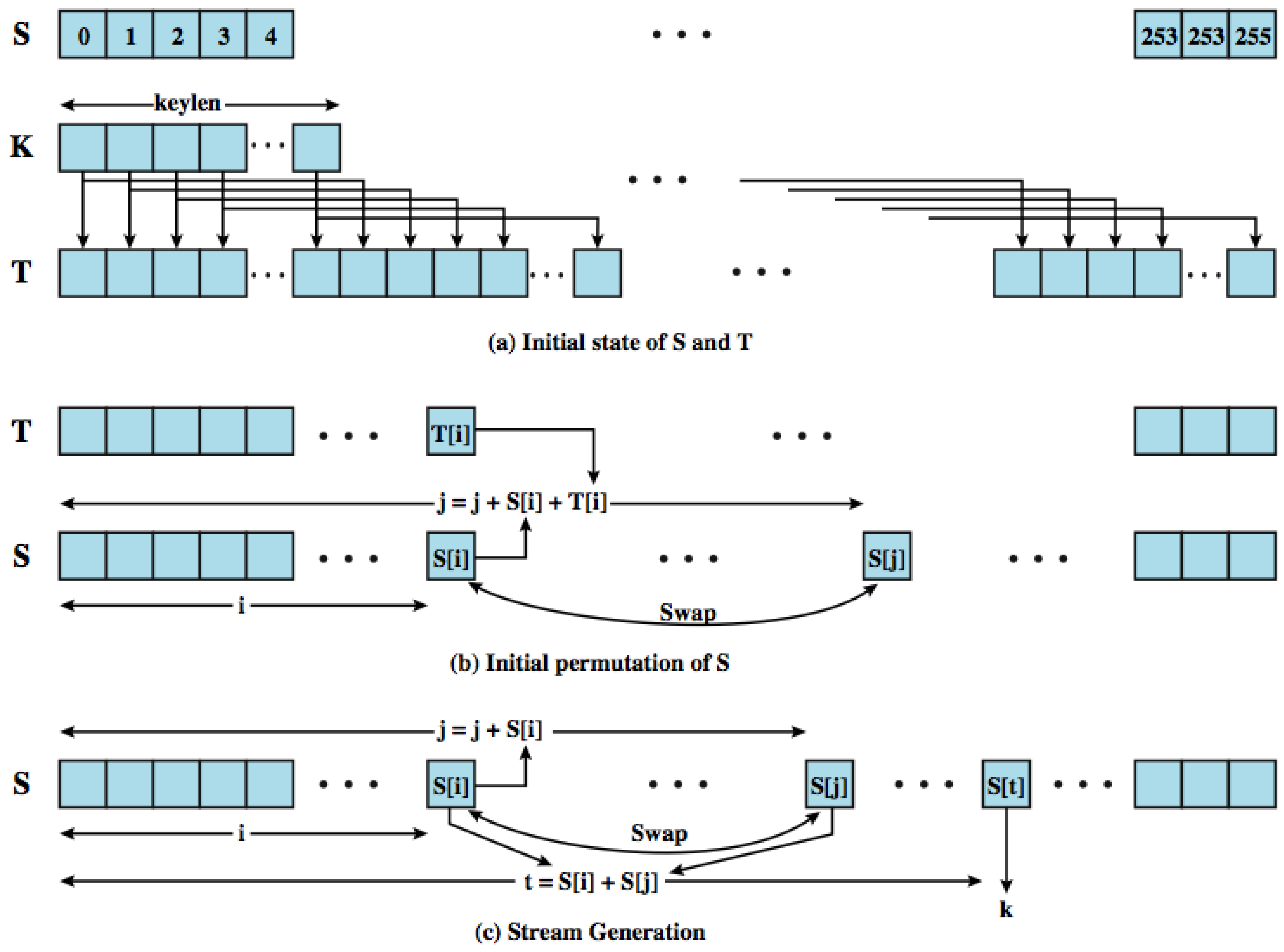
Triple DES

- Keying options
  - Option 1: All the 3 keys are independent
    - Strongest with 3 x 56 bits = 168 independent key bits
  - K_1 and K_2 are independent, $K_3 = K_1$
    - Less security. 2 x 56 = 112 independent key bits
  - All three keys are Identical K1 = K2 = K3
    - Weakest.
    - Equivalent to DES with 56 independent key bits

Stream Ciphers

- processes input elements continuously
- key input to a pseudorandom bit generator
  - produces stream of random like numbers
  - unpredictable without knowing input key
  - XOR keystream output with plaintext bytes
- are faster and use far less code
- design considerations:
  - encryption sequence should have a large period
  - keystream approximates random number properties
  - uses a sufficiently long key

RC4



S
0
1
2
3
4
253
253
255
keylen
K
T
(a) Initial state of S and T
T
T[i]
j = j + S[i] + T[i]
S
S[i]
S[j]
i
Swap
(b) Initial permutation of S
j = j + S[i]
S
S[i]
S[j]
S[t]
i
Swap
t = S[i] + S[j]
k
(c) Stream Generation

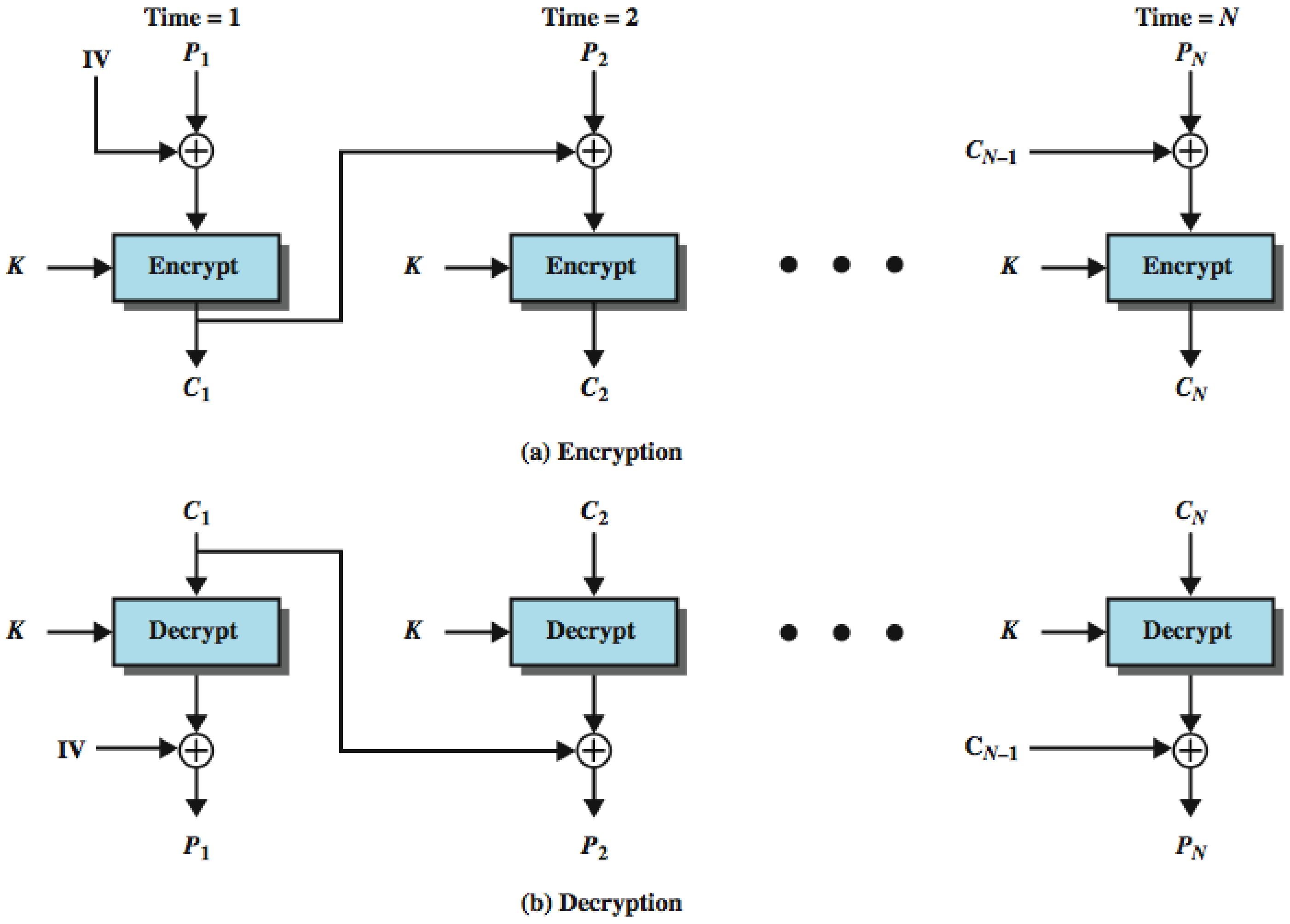
Modes of Operation

- block ciphers process data in blocks
  - e.g. 64-bits (DES, 3DES) or 128-bits (AES)
- for longer messages must break up
  - and possibly pad end to blocksize multiple
- have 5 five **modes of operation** for this
  - defined in NIST SP 800-38A
  - modes are: ECB, CBC, CFB, OFB, CTR

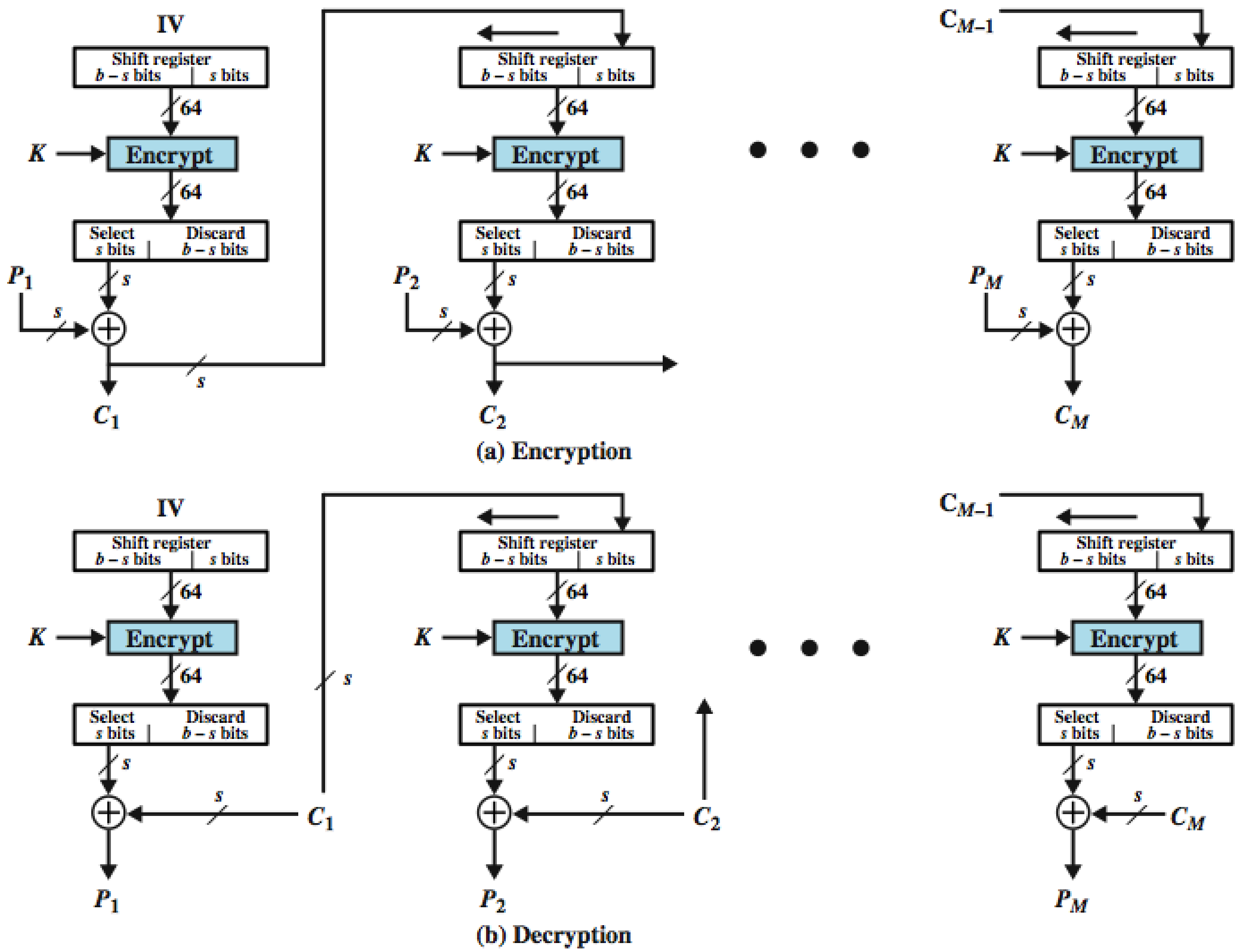
Electronic Codebook (ECB)

- simplest mode
- split plaintext into blocks
- encrypt each block using the same key
- “codebook” because have unique ciphertext value for each plaintext block
  - not secure for long messages since repeated plaintext is seen in repeated ciphertext

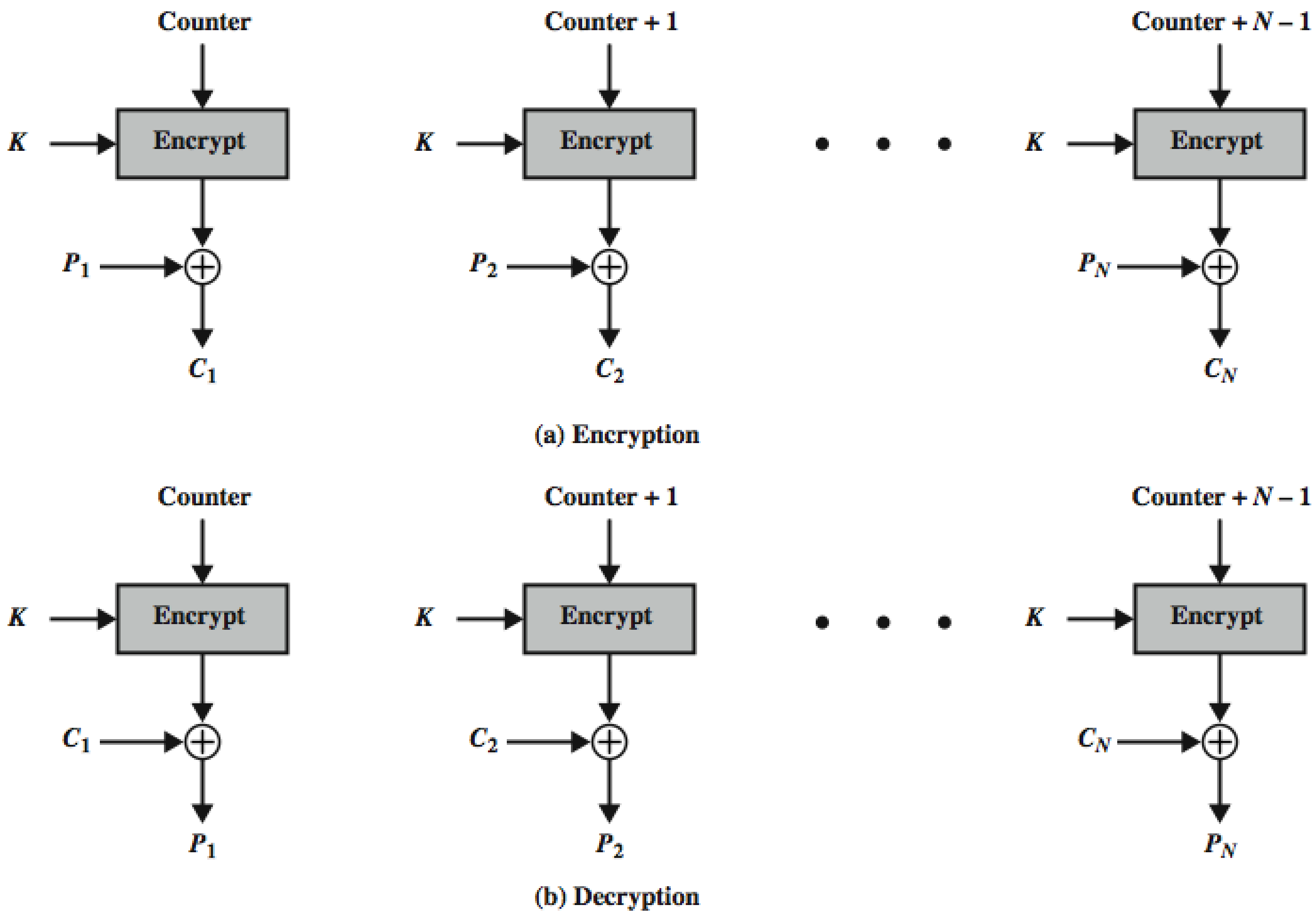
Cipher Block Chaining (CBC)



Cipher Feedback (CFB)

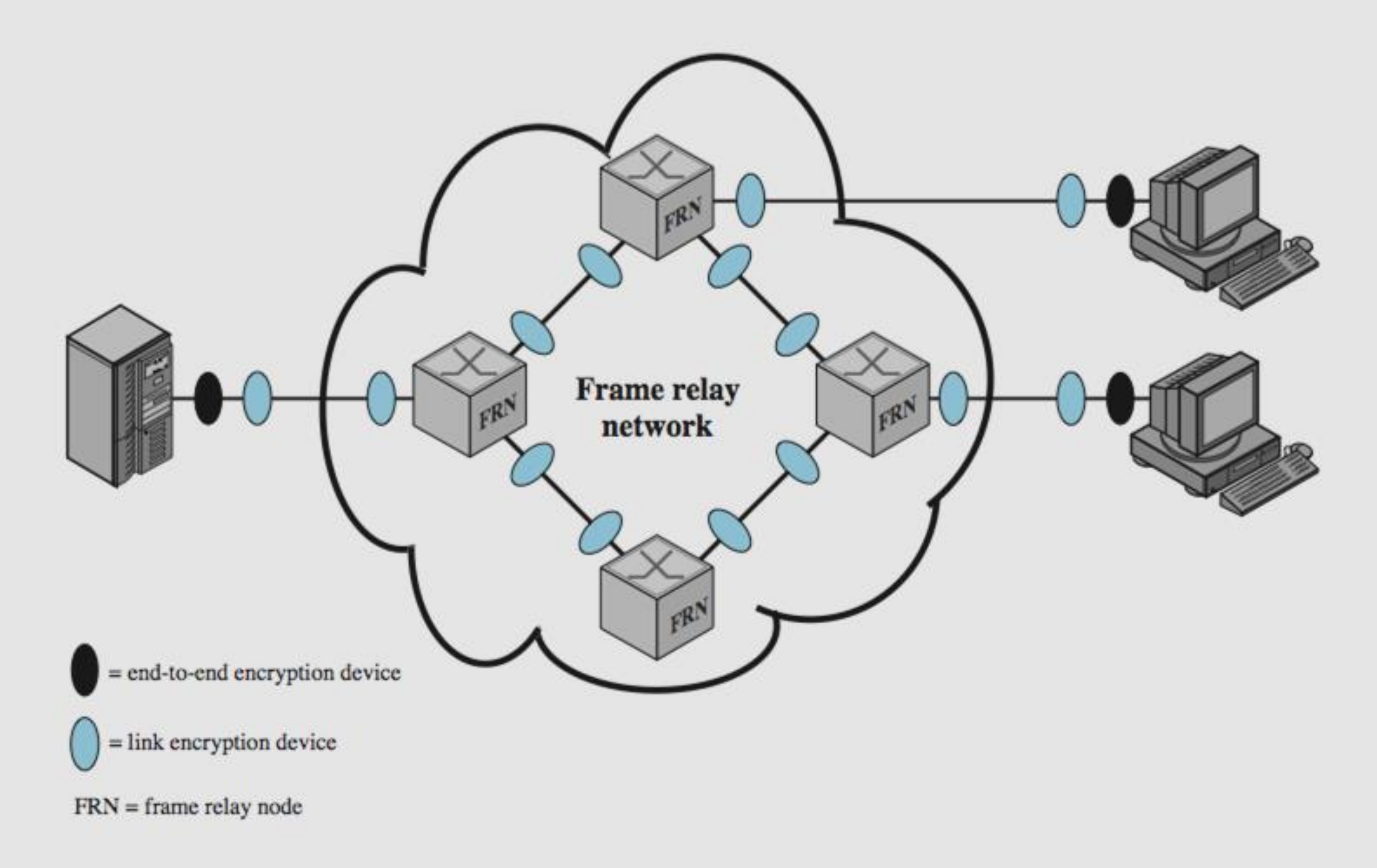


Counter (CTR)



Counter
Counter + 1
Counter + N – 1
K
Encrypt
P1
P2
PN
C1
C2
CN
(a) Encryption
Counter
Counter + 1
Counter + N – 1
K
Encrypt
C1
C2
CN
P1
P2
PN
(b) Decryption

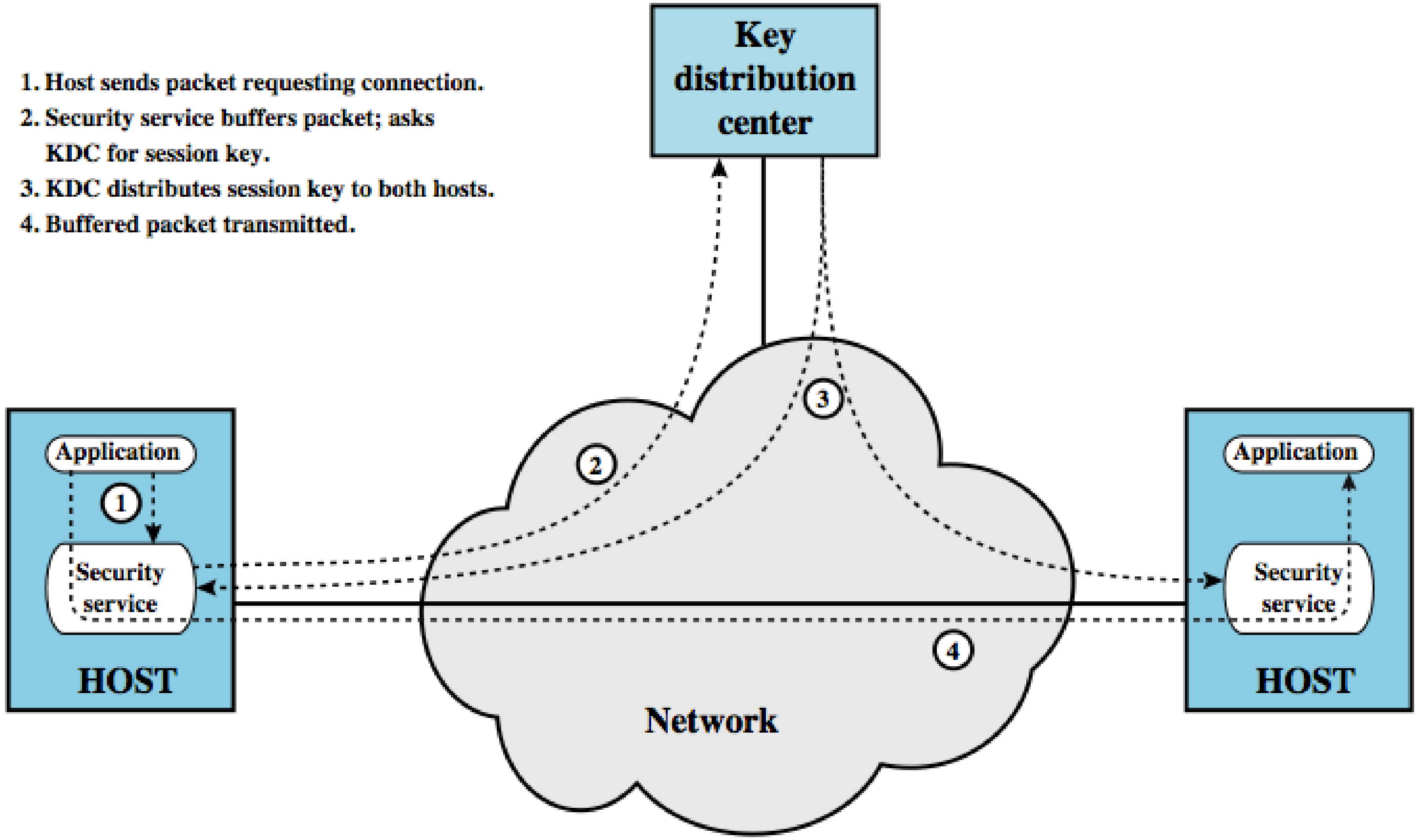
Location of Encryption



Key Distribution

- symmetric crypto needs a shared key:
- two parties A & B can achieve this by:
  - A selects key, physically delivers to B
  - 3rd party select keys, physically delivers to A, B
    - reasonable for link crypto, bad for large no’s users
  - A selects new key, sends encrypted using previous old key to B
    - good for either, but security fails if any key discovered
  - 3rd party C selects key, sends encrypted to each of A & B using existing key with each
    - best for end-to-end encryption

Key Distribution



Symmetric Key Cryptography

- Problems:
  - The key must be pre-shared between the sender and the recipient. Implies impossibility if sender and receiver have never interacted before.
  - Key maintenance challenges
    - Need to keep so many keys for each person one wishes to communicate with
- Birth of public key cryptography in 1976

Public Key Cryptography

- Uses two keys
  - Private key:
    - Known only to the sender
    - Used for decryption
  - Public key:
    - Known to everybody
    - Used for encryption
- Sender and receiver do not share key

Public Key Cryptography

- Alice {Pr_a, Pu_a}
- Bob {Pr_b, Pu_b}

E_Pr_a(M) ==C_a { D_Pu_a(C_a) == M}

- E_Pu_a(M) ==C_a { D_Pr_a(C_a) == M}

Main difference from symmetric key cryptography

- The public encryption key is different from the secret decryption key.
- Infeasible for an attacker to find out the secret decryption key from the public encryption key.
- No need for sender and receiver to distribute a shared secret key beforehand
- only one pair of public and secret keys is required for each user

How public key cryptography works

- Bob:
  - Receives the ciphertext from Alice
  - Decrypts the ciphertext using his secret decryption key, together with the decryption algorithm

Public Key Cryptography Algorithms

- 1976 Dillfie and Hellman proposed crypto scheme with two keys; public key and private key.
- Requirements:
  - Must be computationally easy to encipher/decipher messages using these keys
  - Must be computationally infeasible to derive the private key from public key
  - Must be computationally infeasible to determine the private key from a chosen plaintext attack
- Symmetric key exchange protocol
- RSA
- Other Public-Key Algorithms

Prime and Composite numbers

- Prime and composite numbers
  - a prime number is an integer that can divided only by 1 and itself
  - all other integers are composite

Public Key Cryptography Algorithms

- 1976 Dillfie and Hellman proposed crypto scheme with two keys; public key and private key.
- Requirements:
  - Must be computationally easy to encipher/decipher messages using these keys
  - Must be computationally infeasible to derive the private key from public key
  - Must be computationally infeasible to determine the private key from a chosen plaintext attack
- Symmetric key exchange protocol
- RSA
- Other Public-Key Algorithms

Prime and Composite numbers

- Prime and composite numbers
  - a prime number is an integer that can divided only by 1 and itself
  - all other integers are composite

Modular operation

- “remainder”
  - (13 mod 5) = 3, (1 mod 7) = 1
  - (20 mod 5)= 0, (32 mod 7) = 4
- modular exponentiation
  - (2^2 mod 3)= 1, (3^2 mod 3) = 0
  - (2^2 mod 5)= 4, (10^2 mod 92) = 8
  - (4^6 mod 10)= 6, (3^{11} mod 10) = 7

Diffe-Hellman’s Symmetric Key Exchange Protocol

- It is based on discrete logarithm problem.
- Alice and Bob chooses a prime p=53 and g=17 which is not 0, 1, or p-1=52.
- Alice chooses private key=5, public key=17^5 mod 53 =40.
- Bob choose private key=7, public key=17^7 mod 53=6.
- Bob would like to send message to Alice.

Diffe-Hellman's Symmetric Key Exchange Protocol

- Bob compute the shared secret key by enciphering Alice's public key using his private key:
  $40^7 \bmod 53 = 38$
- Encipher the message with key=38.
- Alice computes the shared secret key as
  $6^5 \bmod 53=38$.
- Then decipher the message with key=38.

Realising public key ciphers

- The most famous system that implements Diffie & Hellman’s ideas is RSA.
- Derived from names of inventers:
  - Ronald Rivest
  - Adi Shamir
  - Leonard Adleman

RSA

- An exponentiation cipher.
- Choose two prime numbers p and q.
- Let $n = pq$. The totient $\phi(n)$ of n is the number of numbers less than n with no factors in common with n.
- $\phi(n)=(p-1)(q-1)$?
- E.g., $\phi(10) = 4$; since 1,3, 7, 9 are relative prime of 10.

RSA

- Choose e <n; e be relative prime to $\phi(n)$.
- Find d such that ed mod $\phi(n) = 1$.
- The public key is (e,n), private key is d.
- $C=m^e \bmod n$
- $M=c^d \bmod n$

RSA

- Actual RSA primes should be at least 512 bits → modulus at least 1024 bits.
- An example of the RSA algorithm. Here p=3, q=11, n=33, z=(p-1)*(q-1)=20, choose d=7, which is relative prime of z. choose e=3 where e*d mod 20 = 1.
- Here (3, 20) is public key. (7,20) is private key.
- $C=P^e$ mod n; $P=C^d$ mod n;

RSA

Plaintext (P)			Ciphertext (C)		After decryption	
Symbolic	Numeric	P^3	P^3 (mod 33)	C^7	C^7 (mod 33)	Symbolic
S	19	6859	28	13492928512	19	S
U	21	9261	21	1801088541	21	U
Z	26	17576	20	1280000000	26	Z
A	01	1	1	1	01	A
N	14	2744	5	78125	14	N
N	14	2744	5	78125	14	N
E	05	125	26	8031810176	05	E

Sender's computation

Receiver's computation

Private Vs public key ciphers

- Public key cipher strengths
  - No key distribution challenges
- Public key cipher weaknesses
  - Relatively expensive to use
  - Relatively slow
  - VLSI chips not available or relatively high cost

Private Vs Public key ciphers

- Usage:
  - Use a public key cipher (such as RSA) to distribute keys
  - Use a private key cipher (such as DES) to encrypt and decrypt messages

Computer Systems Security

Advanced Cryptography

(Hashing Algorithms and Digital Signatures)

Hashing

- Possible message alteration scenarios
  - Insertion of messages from fraudulent sources
  - Changing of message content
  - Modification of message sequence by insertion, deletion or re-ordering of message chunks
  - Modification of message timings by replaying valid sessions

Hashing

- Definition:
  - Condensing of arbitrary messages to fixed size messages
    - h = H(M), H – hash function
    - Condensed representation is known as a hash.
    - Also commonly referred to as message digest
- Purpose
  - Authentication, not encryption

Hashing

  - Cryptographic hash function is usually public
  - Hashing is used to detect modifications on messages
- Cryptographic hash functions, properties:
  - One-way
    - Computationally infeasible to find recreate the message given a hash
  - Collision-free
    - Computationally infeasible to map two or more data with the same hash

Hashing - Applications

- Public Key Algorithms
  - Password logins
  - Encryption key management
  - Digital signatures
- Integrity checking
  - Virus and malware scanning
- Authentication
  - Secure web
    - (SSH, SSL, S/MIME) etc
- Pseudorandom functions (PRFs) /pseudorandom number generators (PRNGS)

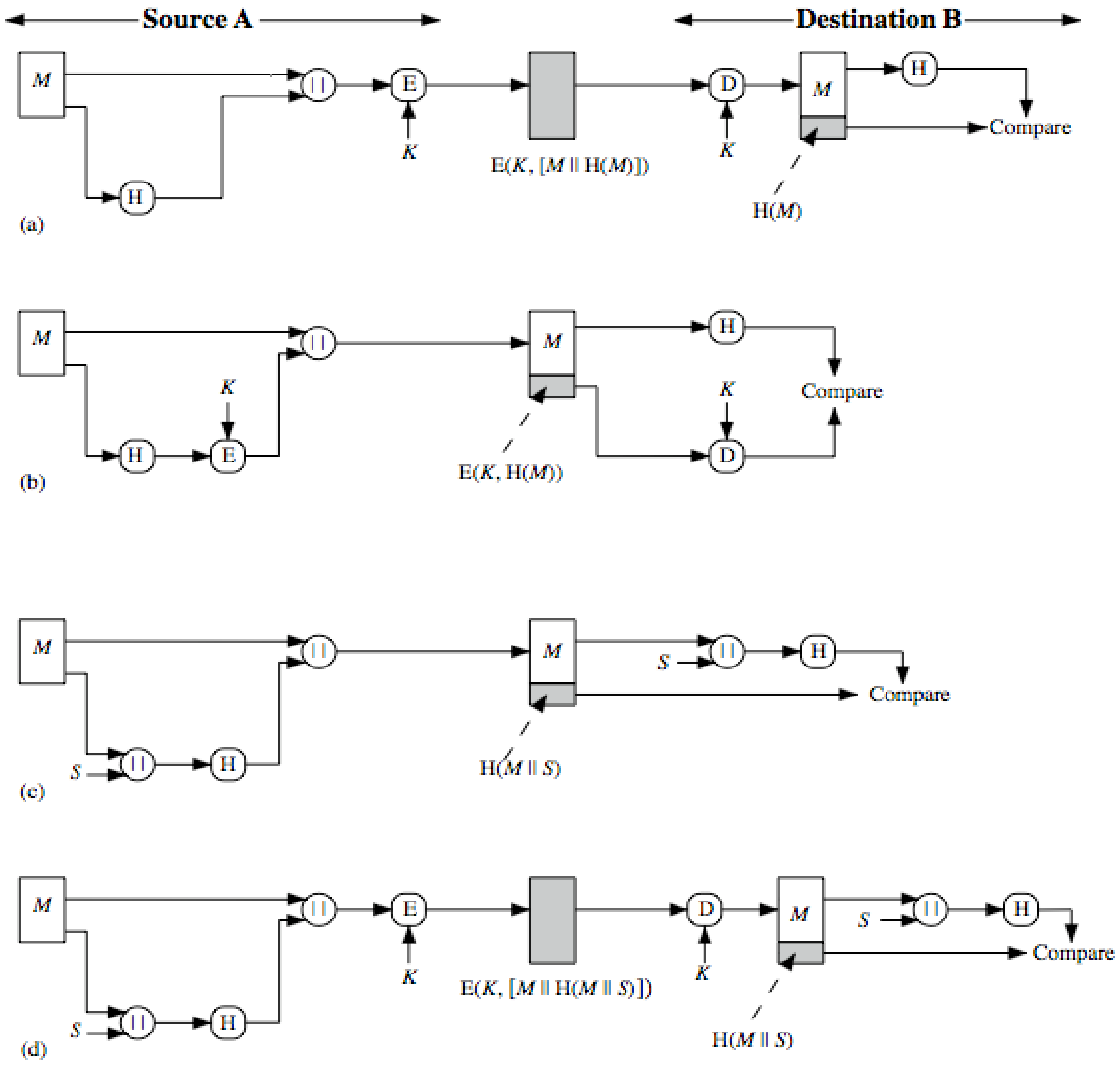
Simple Hash functions

- Simple and insecure
  - bit-by-bit XOR of every block
  - $C_i = b_{i1}\ xor\ b_{i2}\ xor\ .\ .\ .\ xor\ b_{im}$
  - reasonably effective as data integrity check
- One bit circular shift on hash value
  - For each successive n bit block
  - Shift current hash value to the left by 1 bit and XOR block

Hashing and Message Authentication

- Message Authentication
  - A mechanism or service used to verify the integrity of a message, by assuring that the data received are exactly as sent.

Hashing and Message Authentication



Hashing Algorithms

- MD4 and MD5 (by Ron Rivest)
- SHA-0, SHA-1 by NSA
- RIPEMD-160
- SHA-2 (2002 – 224, 256, 385, 512)
- SHA-3 (Selected in 2012)
- etc

Attacks on Hash functions

- Brute force attack and cryptanalysis
  - find m such tha. $H(m)$ equals a given hash value
- Collision resistance attack
  - Find two messages x & y with same hash so $H(x) = H(y)$

Secure Hashing Algorithm (SHA)

- Originally designed by NIST & NSA in 1993
- Revised in 1995 as SHA-1
- Based on design of MD4 with key differences
- produces 160-bit hash values
- In 2005 results on security of SHA-1 raised concerns on its use in future applications
- In 2012, SHA-3 was defined.

Secure Hashing Algorithm (SHA-1)

- Step 1: Append Padding Bits
  - Message is “padded” with a 1 and as many 0’s as necessary to bring the message length to 64 bits fewer than an even multiple of 512.
- Step 2: Append Length
  - 64 bits are appended to the end of the padded message. These bits hold the binary format of 64 bits indicating the length of the original message.

Secure Hashing Algorithm

- Step 3: Prepare Processing Functions
  - SHA1 requires 80 processing functions defined as:
    - f(t;B,C,D) = (B AND C) OR ((NOT B) AND D) (0 <= t <= 19)
    - f(t;B,C,D) = B XOR C XOR D (20 <= t <= 39)
    - f(t;B,C,D) = (B AND C) OR (B AND D) OR (C AND D) (40 <= t <=59)
    - f(t;B,C,D) = B XOR C XOR D   (60 <= t <= 79)
- Step 4: Prepare Processing Constants
  - SHA1 requires 80 processing constant words defined as:
    - K(t) = 0x5A827999 (0 <= t <= 19)
    - K(t) = 0x6ED9EBA1 (20 <= t <= 39)
    - K(t) = 0x8F1BBCDC (40 <= t <= 59)
    - K(t) = 0xCA62C1D6 (60 <= t <= 79)

Secure Hashing Algorithm

- Step 5: Initialize Buffers
  - SHA1 requires 160 bits or 5 buffers of words (32 bits):
    - H0 = 0x67452301
    - H1 = 0xEFCDAB89
    - H2 = 0x98BADCFE
    - H3 = 0x10325476
    - H4 = 0xC3D2E1F0

Secure Hashing Algorithm

- Step 6: Processing Message in 512-bit blocks (L blocks in total message)
  - Loops through the padded and appended message in 512-bit blocks
  - Input and predefined functions:
    - M[1, 2, ..., L]: Blocks of the padded and appended message
    - f(0;B,C,D), f(1,B,C,D), ..., f(79,B,C,D): 80 Processing Functions
    - K(0), K(1), ..., K(79): 80 Processing Constant Words
    - H0, H1, H2, H3, H4, H5: 5 Word buffers with initial values

SHA - Pseudocode

```
For loop on k = 1 to L
        /* Divide M[k] into 16 words */
        (W(0),W(1),...,W(15)) = M[k]
        For t = 16 to 79 do:
            W(t) = (W(t-3) XOR W(t-8) XOR W(t-14) XOR W(t-16)) <<< 1
         A = H0, B = H1, C = H2, D = H3, E = H4
         For t = 0 to 79 do:
              TEMP = A<<<5 + f(t;B,C,D) + E + W(t) + K(t) E = D, D = C,
                             C = B<<<30, B = A, A = TEMP
         End of for loop
         H0 = H0 + A, H1 = H1 + B, H2 = H2 + C, H3 = H3 + D, H4 = H4 + E
    End of for loop

Output:
        H0, H1, H2, H3, H4, H5: Word buffers with final message digest
```

SHA versions

	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512
Message digest size	160	224	256	384	512
Message size	$< 2^{64}$	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Block size	512	512	512	1024	1024
Word size	32	32	32	64	64
Number of steps	80	64	64	80	80

Digital Signatures

- An **electronic signature** that can be used to authenticate the identity of the sender of a message or the signer of a document.
  - Can be used to identify when the contents of the original document has been changed
- A type of **asymmetric cryptography**
  - Gives two algorithms,
    - one for signing : Involves the user's secret or **private key**, and
    - one for verifying signatures: Involves the user's **public key**.
    - The output of the signature process is called the "digital signature.

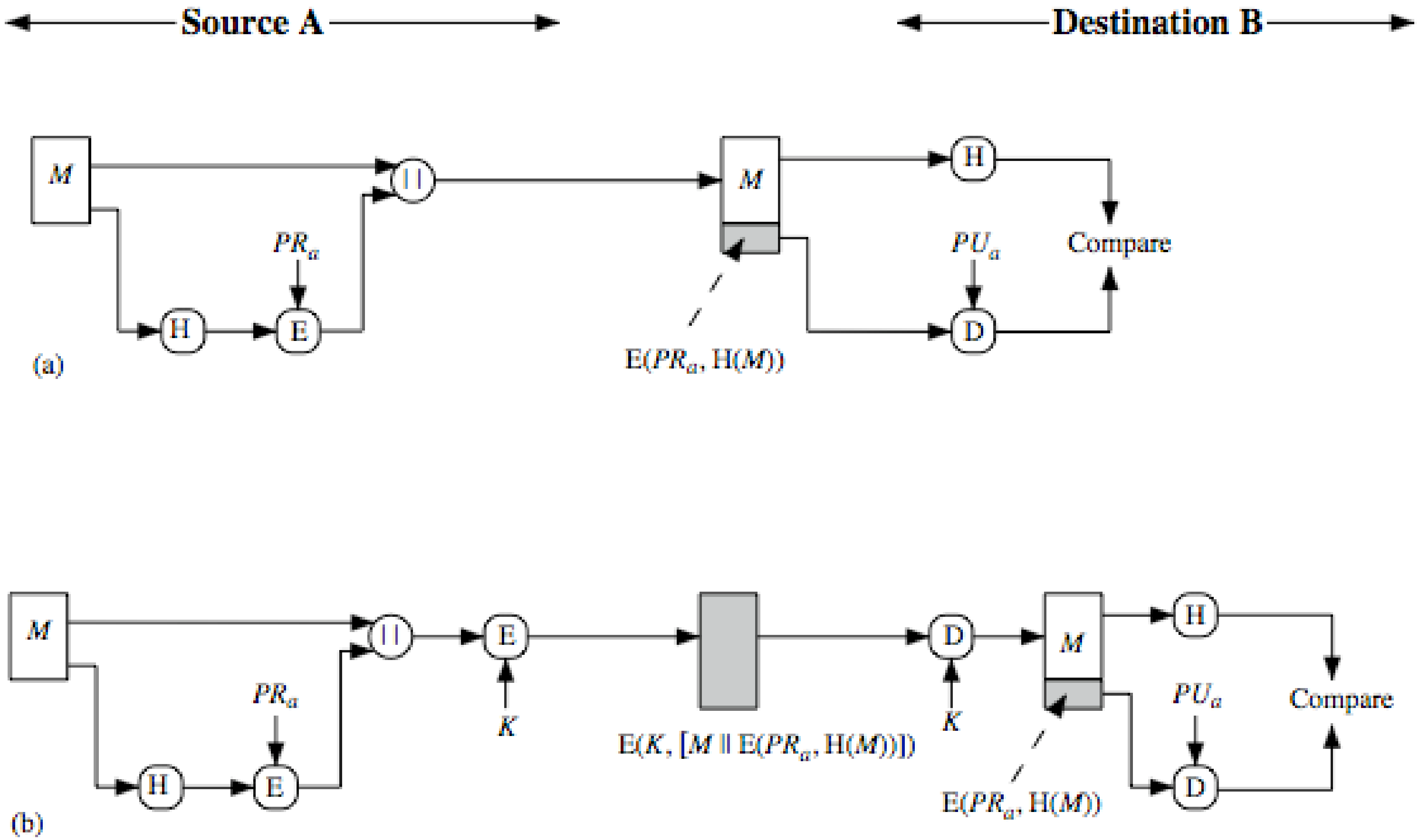
DS Creation

- Performed by the signer
  - Uses a hash result derived from and unique to both the signed message and a given private key.
  - There must be only a negligible possibility that the same digital signature could be created by the combination of any other message or private key.

DS Verification

- Process of checking the digital signature by reference to the original message and a given public key
  - Helps to determine whether the digital signature was created for that same message using the private key that corresponds to the referenced public key.

Hashing and Digital Signatures



Computer Systems Security

Advanced Cryptography

(Digital Certificates and PKI)

Digital Signature

- Services
  - Authentication
  - Integrity
  - Non-repudiation

Digital Signatures

- Problem
  - “REAL” identity of the signer
  - Why should the receiver trust the sender to be whom he/she claims to be?

Digital Certificates

- Digital Signature:
  - Self signed by the sender
- Digital Certificate:
  - A binding between an entity's public key and one or more attributes relating to the entity.
  - Entity:
    - Person, Browser, Hardware
- A Digital Certificate is issued and signed by someone.
  - A trusted third party (TTP)

Digital Certificates

- Specifies:
  - Issuer (usually a CA)
  - Serial Number (Allocated by the CA)
  - Validity period, (usually a year)
  - Subject (the user)
  - User’s public key parameters e.g. RSA
  - Signing algorithm
  - Issuer’s Digital signature (CA signature)

Digital Certificates

- Additional features
  - Tamper-evident
  - The TTP is usually a CA
  - Complete user identification
  - Have fixed expiration

Attribute vs Identity Certificates

- A certificate may bind:
  - A public key with a name or
  - A set of attributes with a name and public key.
- Concerns:
  - More likely to be revoked since attributes are more likely to be changed than the name/key.
  - Certificate becomes too long with description of all attributes

Digital Certificates - Standards

- ITU-T X.509 PKI standard
  - Defined and standardized to be a general, flexible certificate format.
- Three nested components in an X.509 certificate
  - Tamper evident envelop (outer most)
  - Basic certificate contents
  - Certificate extensions (options)

Tamper-evident Envelop

- “Encloses” the certificate (contents)
- Provides:
  - The signature algorithm (e.g. DSA with SHA-1)
  - The signature value

Basic Certificate Contents

- X.509 certificate contents:
  - Version number
  - Serial number
  - Issuer
  - Subject issued certificate
  - Validity period
  - Public key algorithm information of the certificate
  - Digital Signature of the issuing authority
  - Public key of the subject

Certificate Extensions

- Subject type (CA or end entity)
- Names and identity information
- Key attributes
- Policy information
- Additional information
  - CRL distribution points
  - Freshest CRL
  - Authority (CA) information access

X.509 Certificate Usage Model

- To verify a Digital Certificate, the user:
  - Fetches the certificate
  - Fetches certificate revocation list (CRL)
  - Checks certificate against CRL
  - Checks signature using the certificate

Digital Certificates

- Challenges
  - Who issues them?
  - How are they Issued?
  - Why should I Trust the Certificate Issuer?
  - How can I check if a Certificate is valid?
  - How can I revoke a Certificate?
  - Who is revoking Certificates?

Public Key Infrastructure (PKI)

- The goal is to ensure scalable security services
- PKI supports scalable security services using public key cryptography
- PKI: An Infrastructure to support and manage Public Key-based Digital Certificates

Public Key Infrastructure (PKI)

- A PKI is a set of:
  - Agreed-upon standards,
  - Certification Authorities (CA),
  - Structure between multiple CAs,
  - Methods to discover and validate Certification Paths, Operational Protocols, Management Protocols, Interoperable Tools and supporting Legislation

Public Key Infrastructure (PKI)

- Full PKI includes:
    - Certification authority
    - Certificate repository
    - Certificate revocation
    - Key backup and recovery
    - Automatic key update
    - Key history management
    - Cross-certification
    - Support for non-repudiation
    - Time stamping
    - Client software

PKI Architectures

- Single CA
- Hierarchical PKI
- Mesh PKI
- Trust lists (Browser model)
- Bridge CAs

A single CA

- A CA that issues certificates to users and systems, but not to other CAs
- Advantages:
  - Easy to build
  - Easy to maintain
  - All users trust this CA
  - Paths have one certificate and one CRL
- Disadvantages:
  - Doesn’t scale particularly well

Hierarchical PKI

- CAs have a hierarchical relationship
- All CAs trust the root CA
- Root CA certifies its child CAs, and they in turn certify their child CAs, and so on.
- Advantages:
    - Easy to establish/verify trust relationship between any two CAs

Mesh PKI

- CAs have peer-to-peer relationships
- Users trust the CA that issued their certificates

Trusted Lists

- User trusts more than one CA
- Each CA could be a single CA or part of a PKI
  - For hierarchies, should be the root
  - For mesh PKIs, could be any CA

Bridge CA

- Addresses the shortcomings of the trust lists and cross-certified enterprise architecture
- Unify many PKIs into a single PKI
  - Acts as a sort of trust arbitrator
  - If the trust domain is implemented as a hierarchical PKI, the bridge CA will establish a relationship with the root CA
  - If the domain is implemented as a mesh, the bridge will establish a relationship with one of its CAs.

Cross-certification

- Peer-to-peer relationships among Cas
  - CA of one organization being certified by another CA of a different organization
  - Appropriate when a small number of enterprise PKIs intend to establish trust relationships

PKI Design Guidelines

- Identity implementation:
  - Use a locally meaningful identifier
  - User name, email address, Account number
- If possible, design your PKI so that revocation isn't required
- Alternately, use a mechanism which provides freshness guarantees

PKI Design Guidelines

- Application-specific PKIs
- Simple PKI (SPKI) Binds a key to an authorization
- X.509 binds a key to an identity which must then somehow be mapped to an authorization
- PGP is designed to secure email

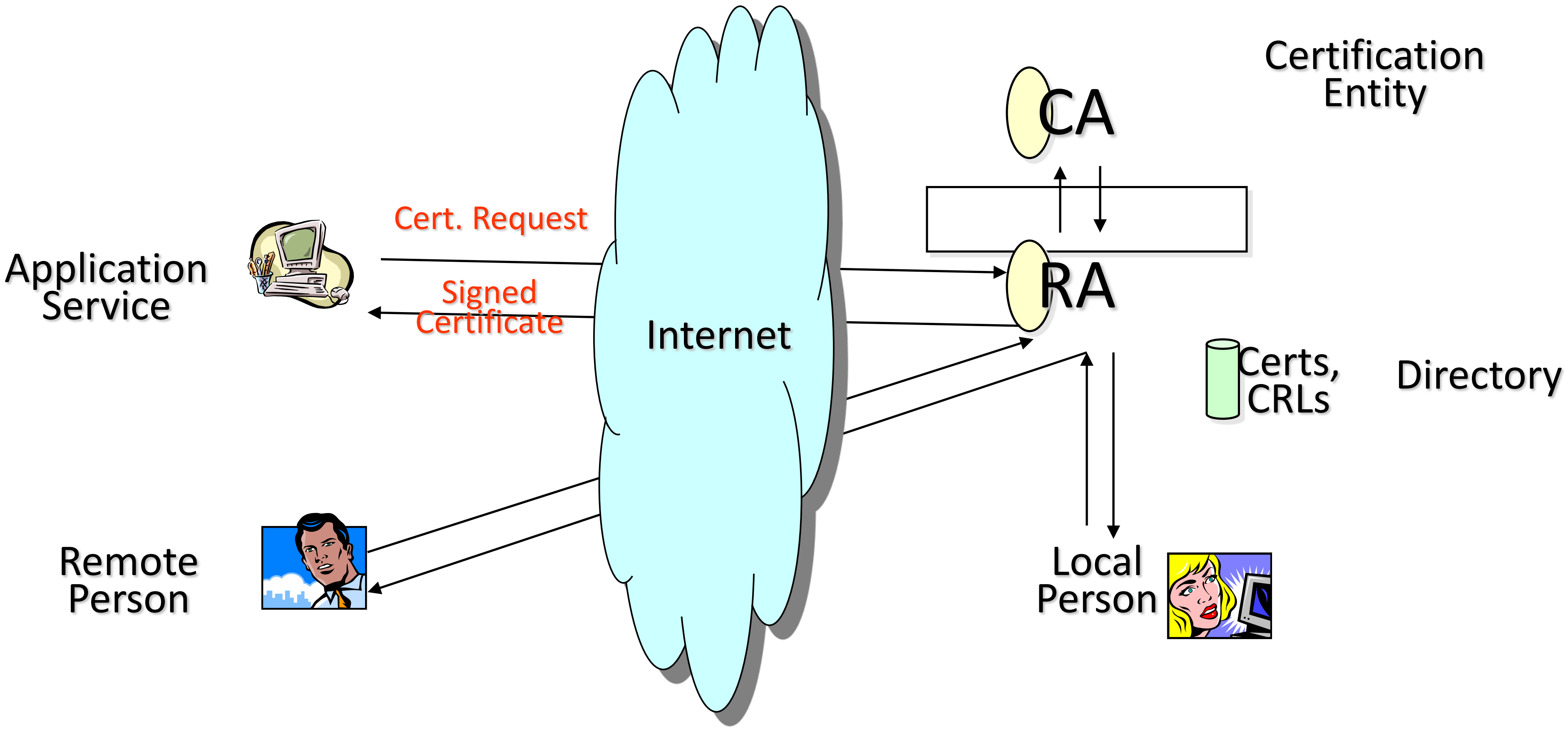
Public Key Infrastructure (PKI)

- Approaches:
  - SPKI
  - X509 PKI
- Focus on IETF defined standards:
  - X509 PKI
  - X509 Digital Certificates

X.509 PKI Overview

- Basic components
  - Provider Side
    - Certificate Authority (CA)
    - Registration Authority (RA)
    - Certificate Distribution System
  - Consumer Side
    - PKI enabled applications

X509 PKI Overview



X509 PKI Certificate Authority (CA)

- Tasks and responsibilities
  - Key Generation
  - Digital Certificate Generation
  - Certificate Issuance and Distribution
  - Revocation
  - Key Backup and Recovery System
  - Cross-Certification

X509 PKI Registration Authority (RA)

- Tasks and Responsibilities
  - Registration of Certificate Information
  - Face-to-Face Registration
  - Remote Registration
  - Automatic Registration
  - Revocation

X.509 Certificate Distribution System (CDS)

- Typically implemented as:
  - Special Purposes Databases
  - Lightweight Directory Acces Protocol (LDAP) directories
- Provides a repository for:
  - Digital Certificates
  - Certificate Revocation Lists (CRLs)

Certificate Revocation List (CRL)

- Revoked certificates remain in a CRL until they expire
- Published by CAs at well defined time intervals
- User application must deal with the revocation processes
- “Users” of certificates “download” a CRL and verify if a certificate has been revoked

CRL Problems

- Similar to Credit card black list problems
  - Not issued frequently enough to be effective against an attacker
  - Expensive to distribute
  - Vulnerable to simple DOS attacks
    - Attacker can prevent revocation by blocking CRL delivery

CRL Problems

- Can contain retroactive invalidity dates:
  - CRL issued right now can indicate that a cert was invalid last week
  - Checking that something was valid at time *t* isn’t sufficient to establish validity
  - Back-dated CRL can appear at any point in the future
  - Destroys the entire concept of non-repudiation

CRL Problems

- CRLs have a fixed validity period
  - Valid from *issue date* to *expiry date*
  - At *expiry date*, all relying parties connect to the CA to fetch the new CRL
  - Massive peak loads when a CRL expires (DDOS attack)
  - The problem worsens if CRLs have to be issued to provide timely revocations

Certificate Revocation List (CRL)

- A CRL typically contains
  - Certificate Issuer (Usually CA)
  - Time of this update
  - Time of the next update
  - List of revoked certificate serial numbers with dates and reasons

Online Certificate Status Protocol (OCSP)

- An alternative to CRLs
- IETF/PKIX standard for a real-time check if a certificate has been revoked/suspended
- Requires a high availability OCSP Server
- Inquires of the issuing CA whether a given certificate is still valid
  - Acts as a simple responder for querying CRLs
  - Still requires the use of a CA to check validity

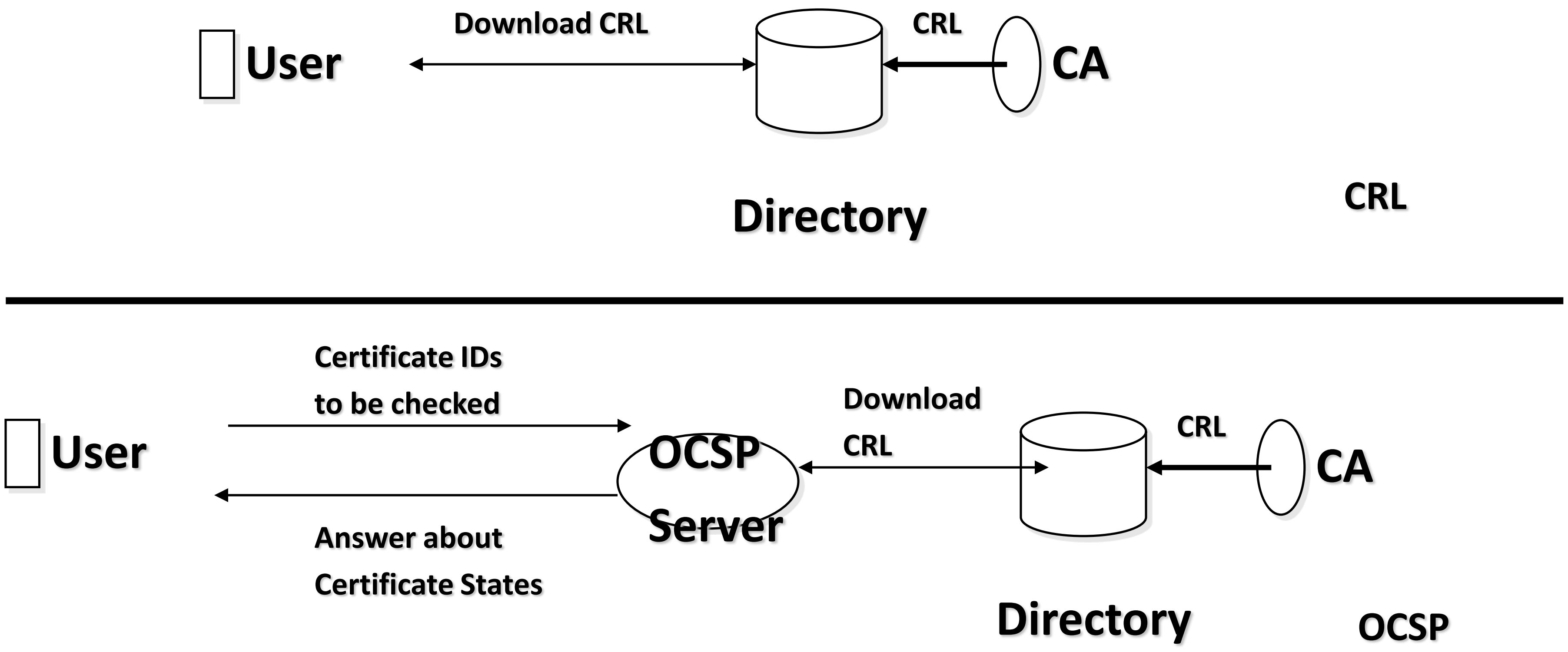
Online Certificate Status Protocol (OCSP)

- Acts as a selective CRL protocol
  - Standard CRL process: “Send me a CRL for everything you’ve got”
  - OCSP process: “Send me a pseudo-CRL/OCSP response for only the specified certificates”
  - Lightweight pseudo-CRL avoids CRL size problems

Online Certificate Status Protocol (OCSP)

- Reply is created on the spot in response to the request
- Ephemeral pseudo-CRL avoids CRL validity period problems
- Requires a signing operation for every query

CRLs Vs OCSP Server



X.509 PKI – User Applications

- PKI enabled applications need:
  - Cryptographic functionality
  - Secure storage of Personal Information
  - Digital Certificate Handling
  - Directory Access
  - Communication Facilities

Running an Enterprise PKI

- Factors to consider:
  - PKI functionality
    - Modular design that is reliable and has high performance in certificate issuance
  - Ease of integration
  - Availability and scalability
  - Security and risk management
  - Expertise

Running an Enterprise PKI

- Deployment models
    - In-house deployment of standalone PKI software
    - Outsourced deployment to an integrated PKI platform E.g. Verisign’s managed PKI