Network Security

Asim Rasheed

Where we are …

- Introduction to network security
- Vulnerabilities in IP
- I. CRYPTOGRAPHY
  - Symmetric Encryption and Message Confidentiality
  - Public-Key Cryptography and Message Authentication
- II. NETWORK SECURITY APPLICATIONS
  - Authentication Applications (Kerberos, X.509)
  - Electronic Mail Security (PGP, S/MIME)
  - IP Security (IPSec, AH, ESP, IKE)
  - Web Security (SSL, TLS, SET)
- III. SYSTEM SECURITY
  - Intruders and intrusion detection
  - Malicious Software (viruses)
  - Firewalls and trusted systems

E-mail Security
PGP

Outline

- PGP
  - Services
  - message format
  - key management
  - trust management
- S/MIME
  - Services
  - message formats
  - key management

Email Security

- Email is one of the most widely used and regarded network services
- Currently message contents are not secure
  - may be inspected either in transit
  - or by suitably privileged users on destination system

Email Security Enhancements

- Confidentiality
  - protection from disclosure
- Authentication
  - of sender of message
- Message integrity
  - protection from modification
- Non-repudiation of origin
  - protection from denial by sender

What is PGP?

- PGP - Pretty Good Privacy
- General purpose application to protect (encrypt and/or sign) files
- Can be used to protect e-mail messages
- Can be used by corporations as well as individuals
- Based on strong cryptographic algorithms (IDEA, RSA, SHA-1)
- Available free of charge at http://www.pgpi.org
- First version developed by Phil Zimmermann
- PGP is now on an Internet standards track (RFC 3156)

Zimmermann’s Design

- Selected the best available cryptographic algorithms
- Integrated these algos into general purpose application that is OS independent
  - Based on small set of easy to use commands

Reasons for PGP Success

- Available free worldwide with versions available for different OS
- Based on algorithms that are considered extremely secure
  - e.g., RSA, Diffie-Hellman and DSS for public key encryption
  - CAST 128, 3DES and IDEA for symmetric encryption
  - SHA-1 for hash functions
- Wide range of applicability
- Not developed by any standards organization

PGP services

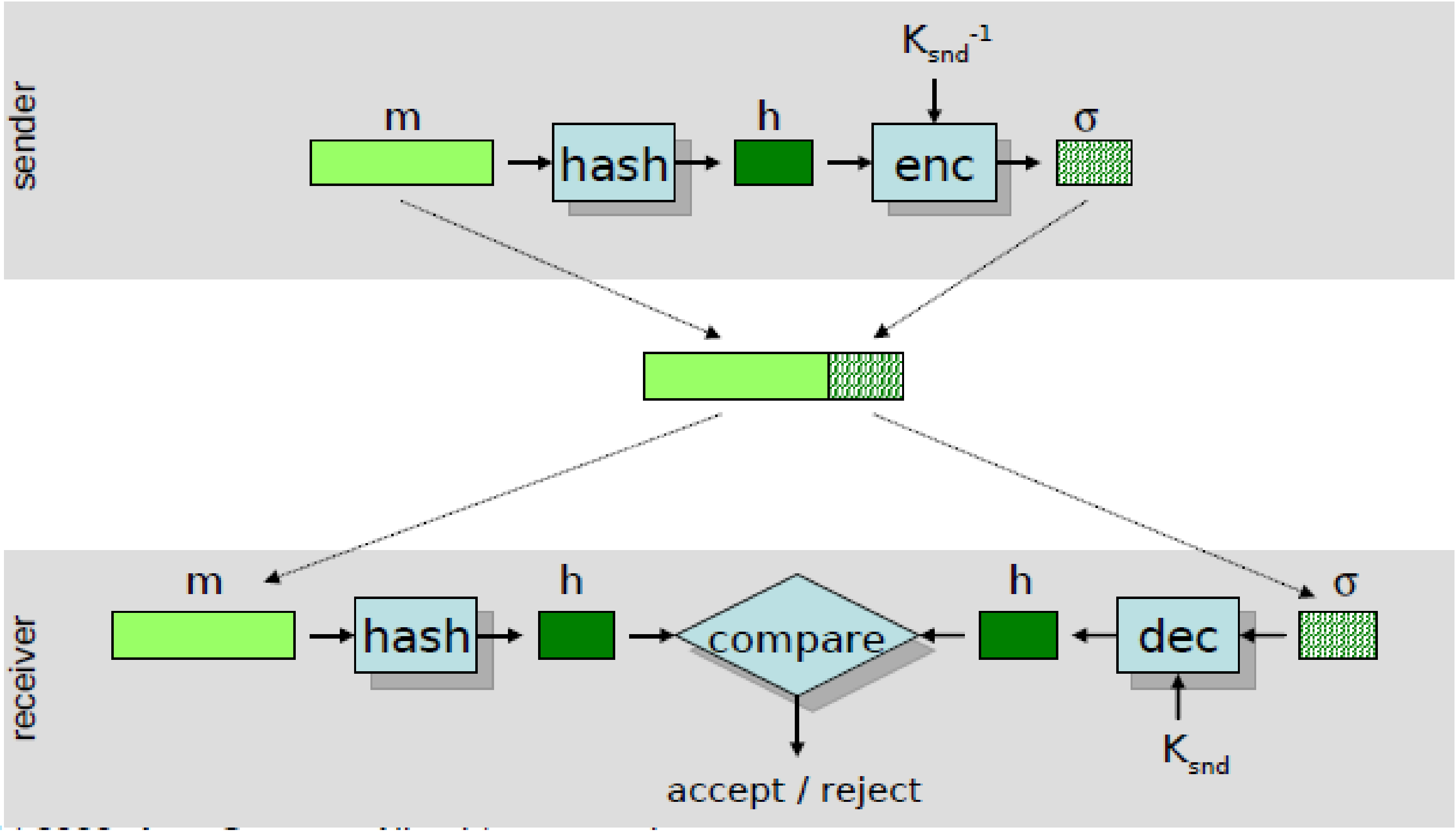
- Messages
  - Authentication
  - Confidentiality
  - Compression
  - e-mail compatibility
  - segmentation and reassembly
- Key management
  - generation, distribution, and revocation of public/private keys
  - generation and transport of session keys and IVs

Message Authentication

- Sender creates a message
- SHA-1 used to generate 160-bit hash code of message
- Hash code is encrypted with RSA using the sender's private key, and result is attached to message
- Receiver uses RSA or DSS with sender's public key to decrypt and recover hash code
- Receiver generates new hash code for message and compares with decrypted hash code, if match, message is accepted as authentic

Message Authentication

Based on digital signatures

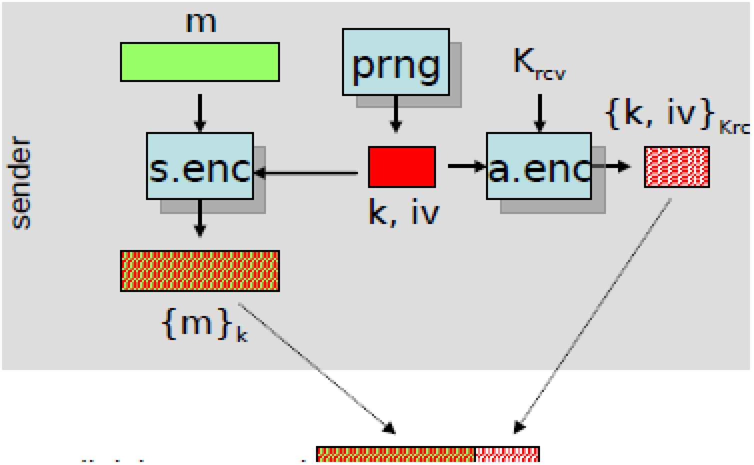


Message Confidentiality

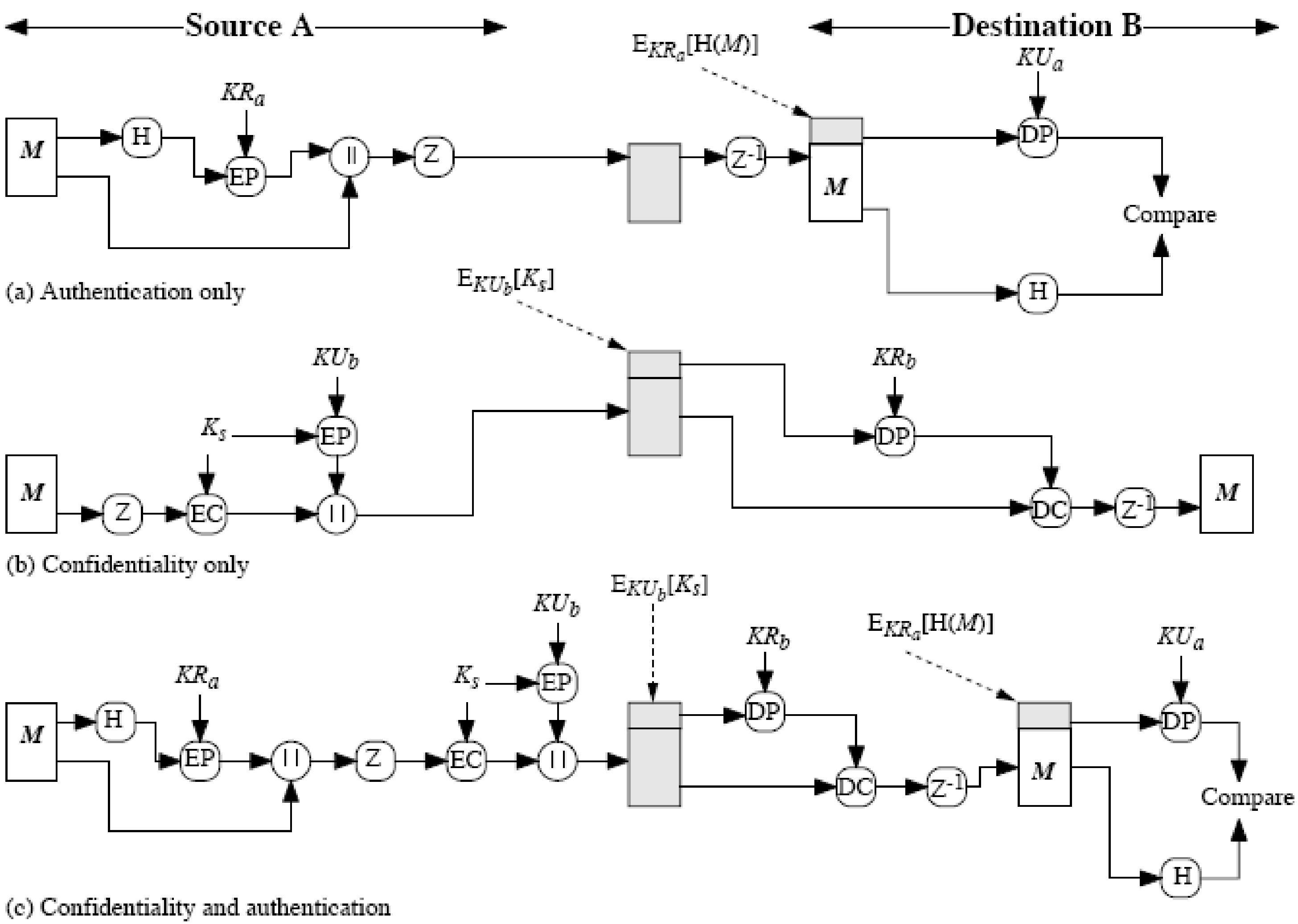
- Sender generates message and random 128-bit number to be used as session key for this message only
- Message is encrypted, using CAST-128 / IDEA/3DES with session key
- Session key is encrypted using RSA with recipient's public key, then attached to message
- Receiver uses RSA with its private key to decrypt and recover session key
- Session key is used to decrypt message

Message confidentiality

- Symmetric key encryption in CFB mode with a random session key and IV
- Supported algorithms:
  - Symmetric: CAST, IDEA, 3DES
  - Asymmetric: RSA, ElGamal



PGP Cryptographic Functions



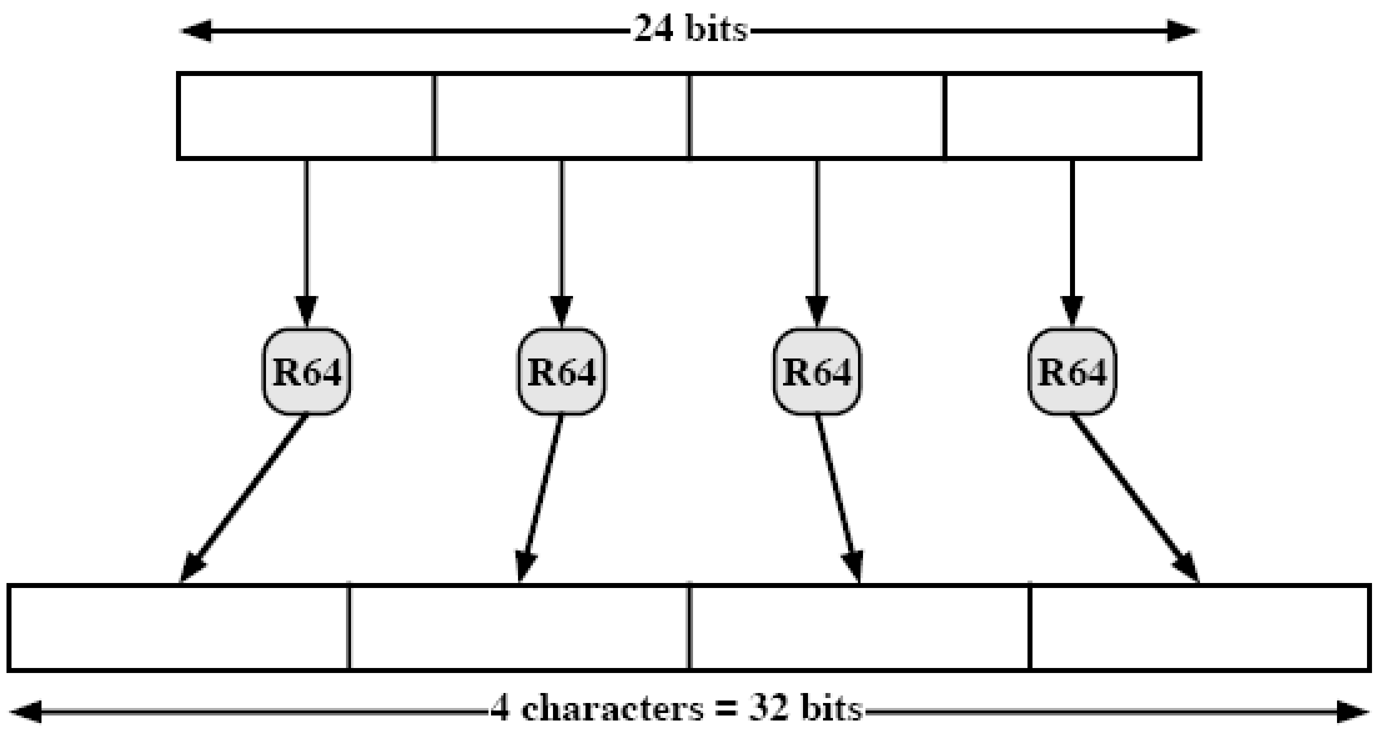
Compression

- Applied after the signature
  - enough to store clear message and signature for later verification
  - it would be possible to dynamically compress messages before signature verification, but …
  - then all PGP implementations should use the same compression algorithm
  - however, different PGP versions use slightly different compression algorithms
- Applied before encryption
  - compression reduces redundancy ® makes cryptanalysis harder
- Supported algorithm: ZIP

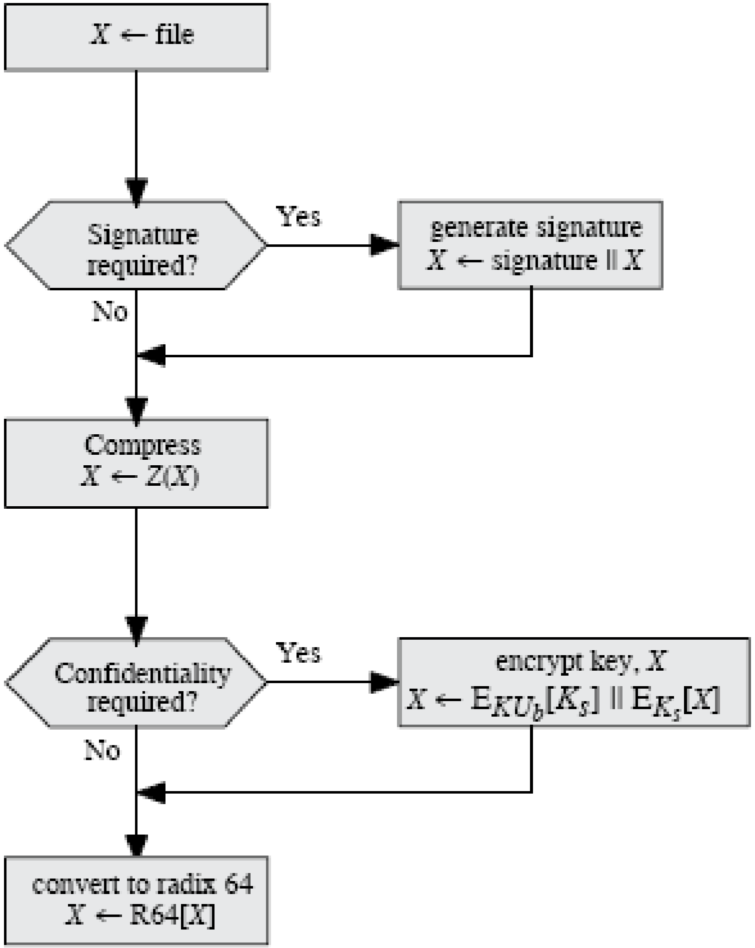
E-mail compatibility

- Encrypted messages and signatures may contain arbitrary octets
- Most e-mail systems support only ASCII characters
- PGP converts an arbitrary binary stream into a stream of printable ASCII characters
- Radix 64 conversion: 3 8-bit blocks => 4 6-bit blocks

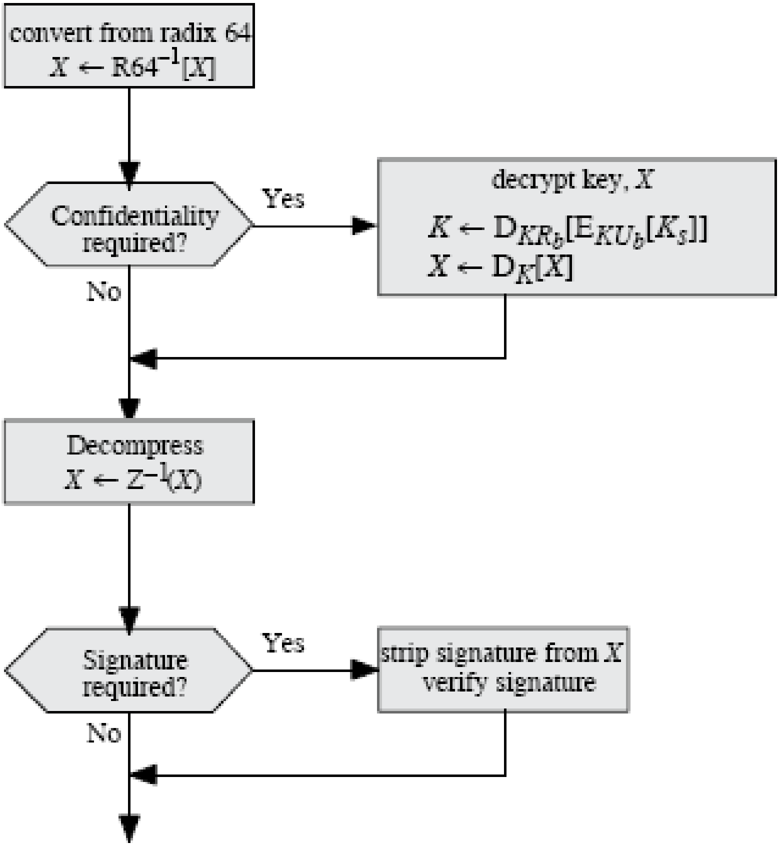
Printable encoding of binary data in to Radix 64 (R-64)



Transmission of PGP Message



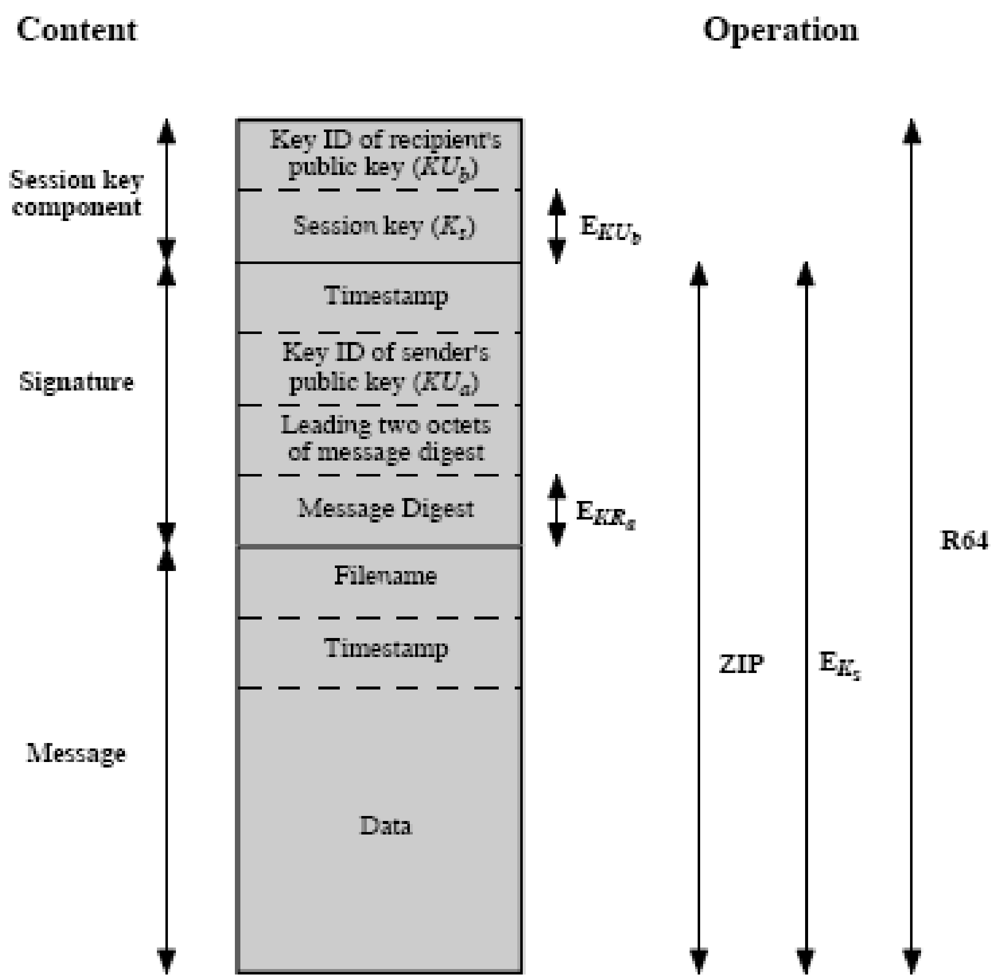
Reception of PGP Message



Segmentation and Reassembly

- E-mails facilities are often restricted to a maximum length
- Longer message needs to be broken down into segments
- PGP automatically does segmentation
- Segmentation is done after all other processing
- At reception, first reassembles the entire original block then further processing

PGP message format



Cryptographic Keys

- PGP uses four type of keys
  - Session keys (symmetric)
  - Public keys
  - Private keys
  - Pass phrase based keys

Requirements of keys

- Means of generating unpredictable session keys
- Means for identifying particular keys.
  - User may have multiple public-private key pairs
- Each entity must maintain a file of its own key pairs as well as a file of public key correspondents

PGP Session Key Generation

- Need a session key for each message
- Used only for encrypting and decrypting that message
- Generated using ANSI X12.17 mode
- Uses random inputs taken from previous uses and from keystroke timing of user to generate random session keys

Key IDs

- A user may have several public key – private key pairs
  - which private key to use to decrypt the session key?
  - which public key to use to verify a signature?
- Transmitting the whole public key would be wasteful
- Associating a random ID to a public key would result in management burden
- PGP key ID: least significant 64 bits of the public key
  - unique within a user with very high probability

Random number generation

- True random numbers
  - Used to generate public key – private key pairs
  - Provide the initial seed for the pseudo-random number generator (PRNG)
  - Provide additional input during pseudo-random number generation
- Pseudo-random numbers
  - Used to generate session keys and IVs

True random numbers

- PGP maintains a 256-byte buffer of random bits
- Each time PGP expects a keystroke from the user, it records
  - the time when it starts waiting (32 bits)
  - the time when the key was pressed (32 bits)
  - the value of the key stroke (8 bits)
- The recorded information is used to generate a key
- The generated key is used to encrypt the current value of the random-bit buffer

Private-Key Ring

- Used to store the public key – private key pairs owned by a given user
- Keys are encrypted using passphrase
- Essentially a table, where each row contains the following entries:
  - Timestamp
  - key ID (indexed)
  - public key
  - encrypted private key
  - user ID (indexed)

Private-Key Ring

- KeyID: least significant 64 bits of public key
- Can be indexed by either userID or keyID

Private Key Ring

Timestamp	Key ID*	Public Key	Encrypted Private Key	User ID*
• • •	• • •	• • •	• • •	• • •
T_i	$KU_i \bmod 2^{64}$	KU_i	$E_{H(P_i)}[KR_i]$	User i
• • •	• • •	• • •	• • •	• • •

Public-Key Ring

- Used to store public keys of other users
- A table, where each row contains the following entries:
  - Timestamp
  - key ID (indexed)
  - public key
  - user ID (indexed)
  - owner trust
  - signature(s)
  - signature trust(s)
  - key legitimacy

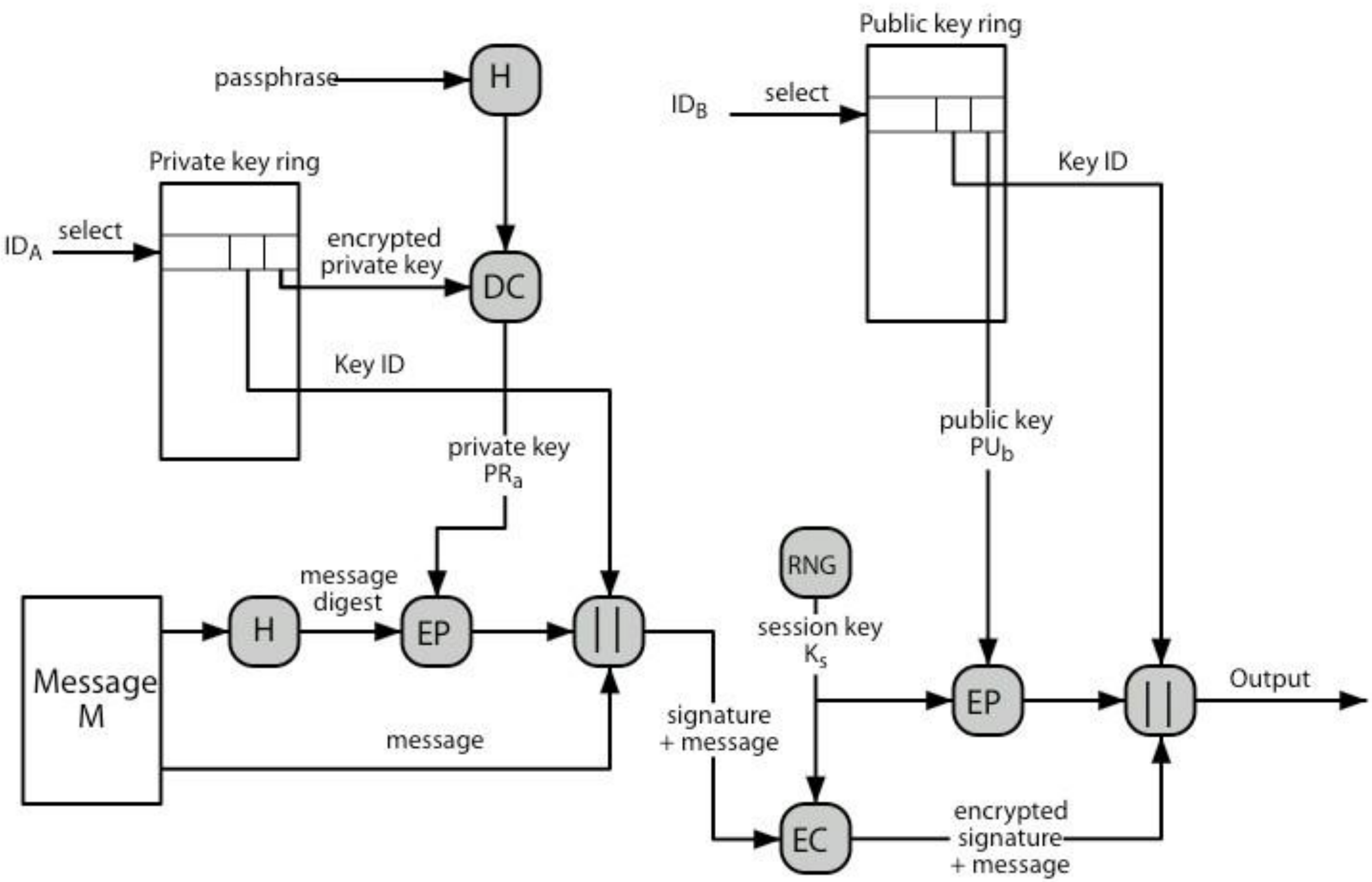
Public-Key Ring

- Used to store public keys of others are that known to this user

Public Key Ring

Timestamp	Key ID*	Public Key	Owner Trust	User ID*	Key Legitimacy	Signature(s)	Signature Trust(s)
• • •	• • •	• • •	• • •	• • •	• • •	• • •	• • •
T_i	$KU_i \bmod 2^{64}$	Ku_i	$trust\_flag_i$	User i	$trust\_flag_i$		
• • •	• • •	• • •	• • •	• • •	• • •	• • •	• • •

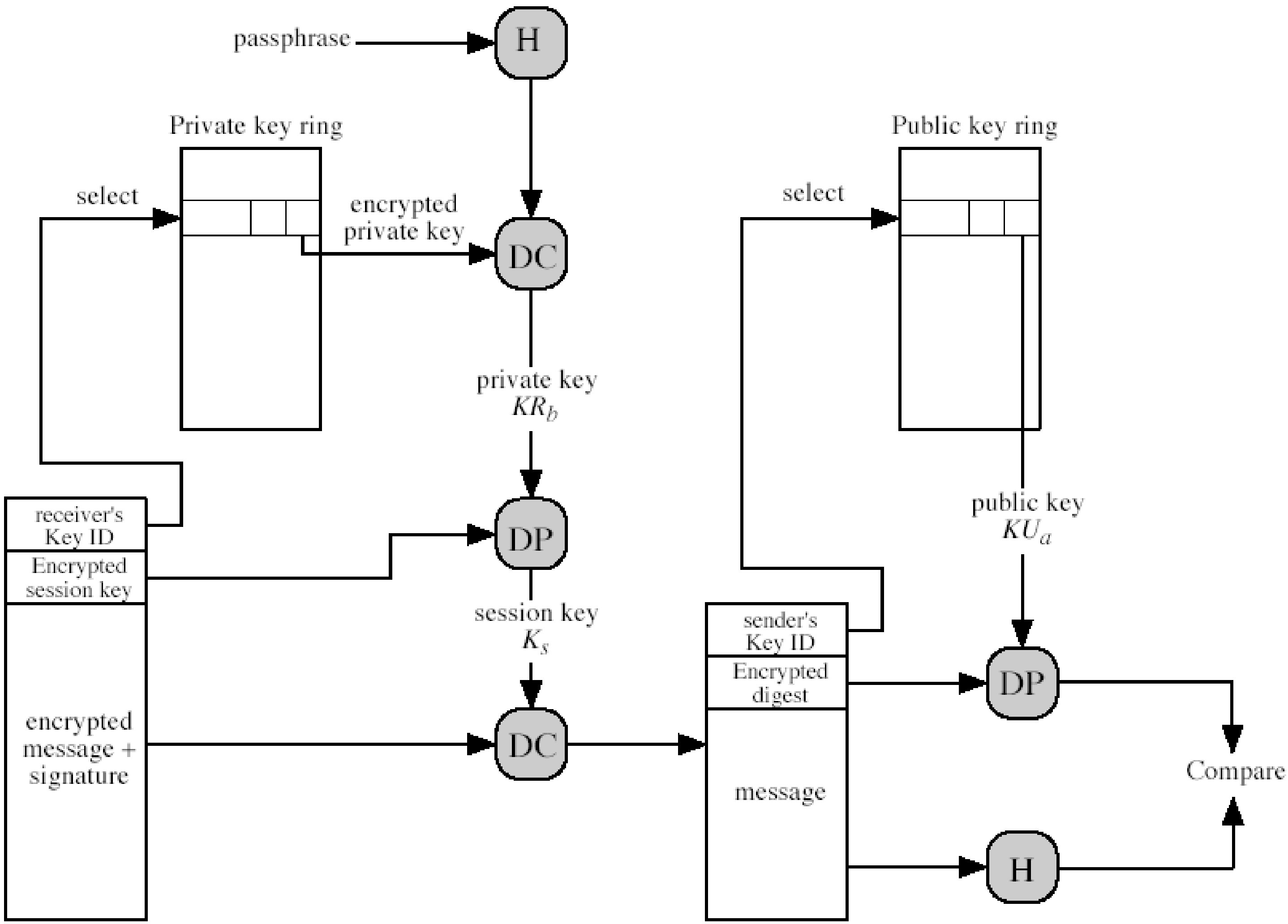
PGP Message Generation



Message Generation

- Signing the message
  - PGP retrieves sender's private key from private key ring using your_userid.
  - PGP prompts user for passphrase to recover unencrypted key
  - Signature is constructed
- Encrypting the message
  - Generates a session key and encrypts message
  - Retrieves public key of recipient from public key
  - Session key component of message is constructed

PGP Reception



PGP Reception

- Decrypting the message
  - Retrieves receiver's private key from private key ring
  - PGP asks for passphrase to recover unencrypted private key
  - Recovers the session key and decrypt message
- Authenticating the message
  - Retrieves the sender’s public key from public key ring, using key id field in signature component
  - Recovers transmitted message digest
  - Computes message digest and compares to authenticate

Trust management

- Owner trust
  - Assigned by the user
  - Possible values:
    - *unknown user*
    - *usually not trusted to sign*
    - *usually trusted to sign*
    - *always trusted to sign*
    - *ultimately trusted (own key, present in private key ring)*

Trust management

- Signature trust
    - assigned by the PGP system
    - if the corresponding public key is already in the public key ring, then its owner trust entry is copied into signature trust
    - otherwise, signature trust is set to *unknown user*

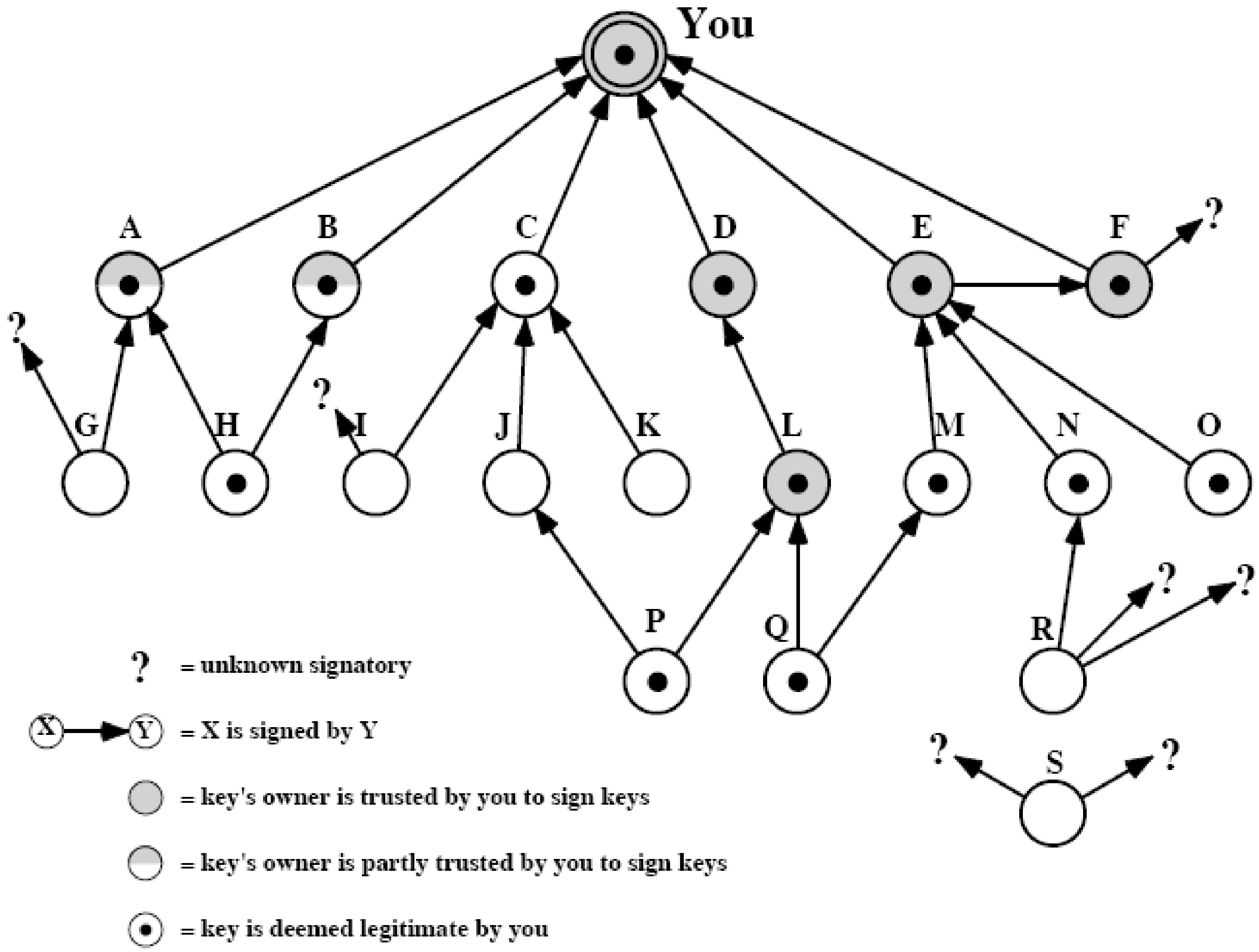
Trust management

- Key legitimacy
  - computed by the PGP system
  - if at least one signature trust is ultimate, then the key legitimacy is 1 (complete)
  - otherwise, a weighted sum of the signature trust values is computed
    - always trusted signatures has a weight of 1/X
    - usually trusted signatures has a weight of 1/Y
    - X, Y are user-configurable parameters

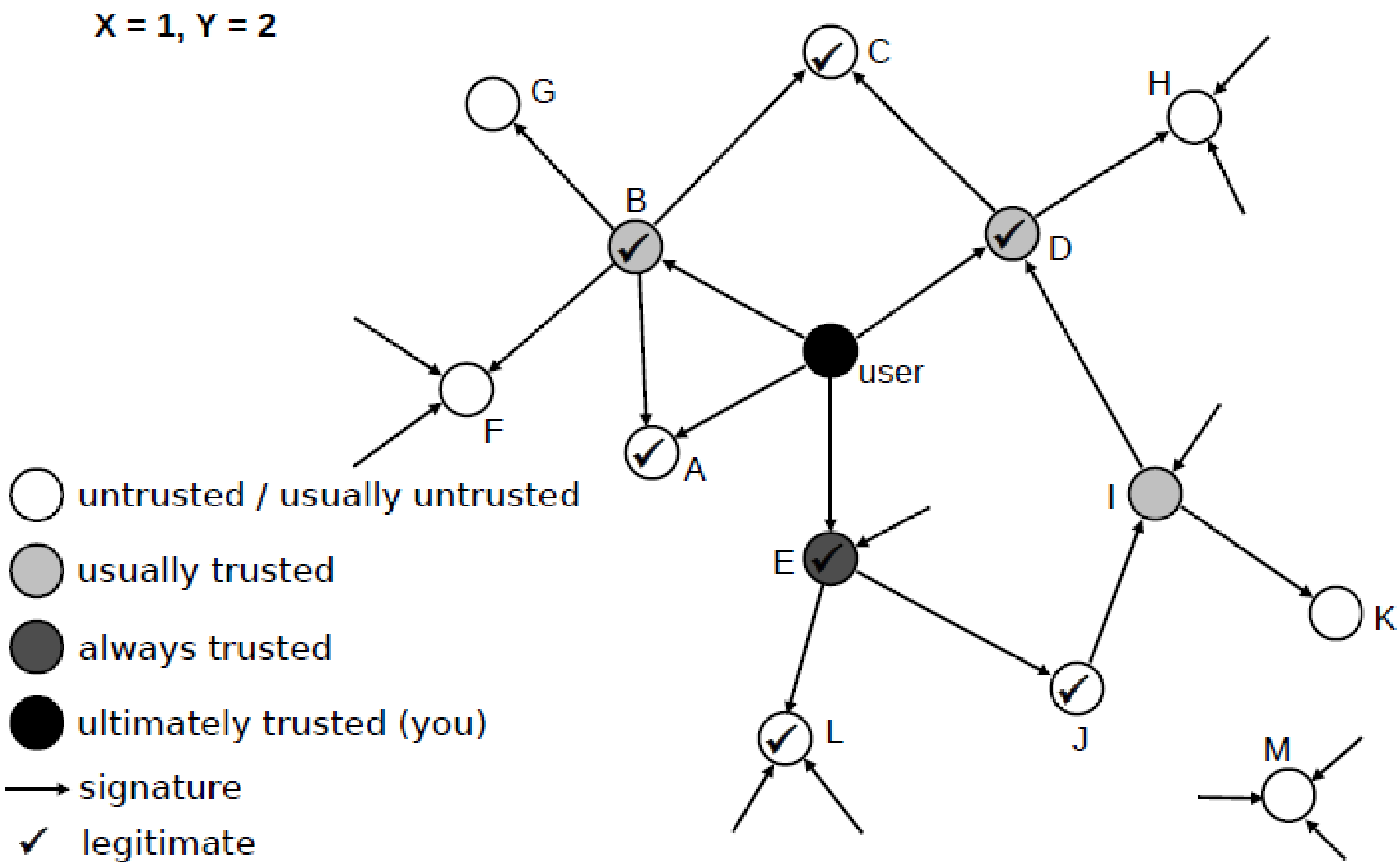
Trust management

- Key legitimacy …
  - example: X=2, Y=4
    - 1 ultimately trusted, or
    - 2 always trusted, or
    - 1 always trusted and 2 usually trusted, or
    - 4 usually trusted signatures are needed to obtain full legitimacy

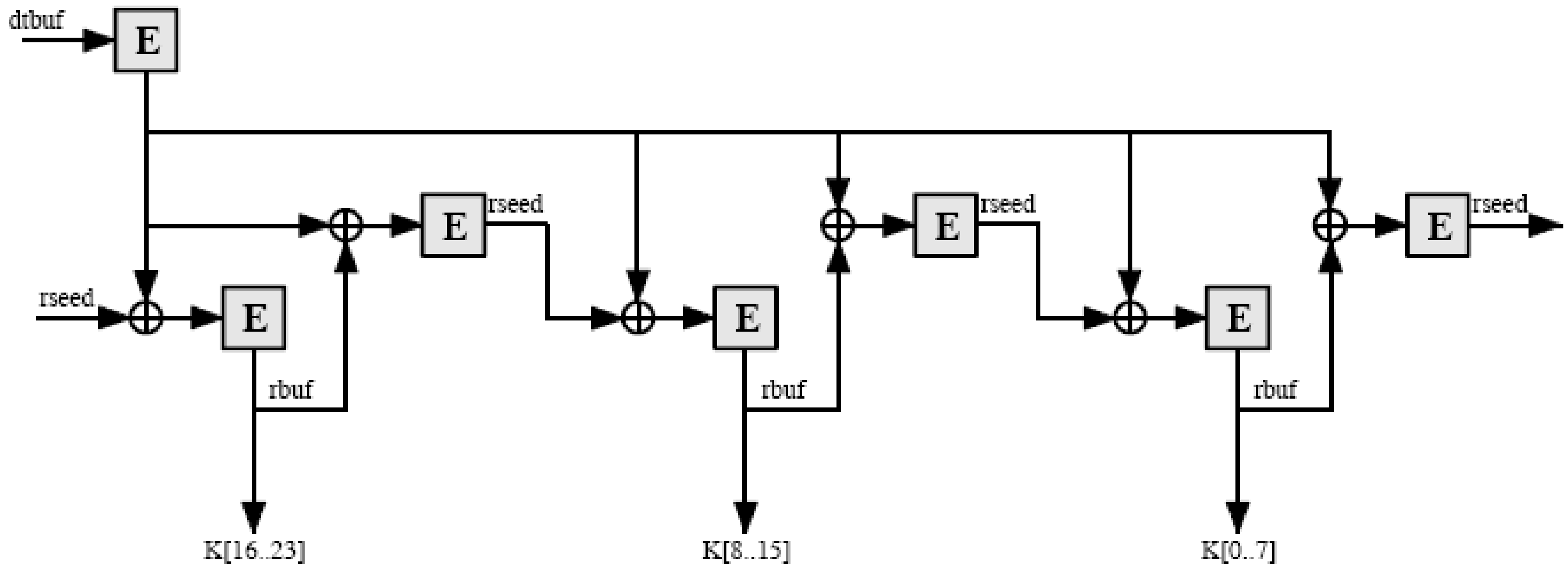
PGP Trust Model



Example – Key Legitimacy



PGP Session Key and IV Generation



Public-key revocation

- Why to revoke a public key?
  - suspected to be compromised (private key got known by someone)
  - re-keying
- The owner issues a revocation certificate …
  - has a similar format to normal public-key certificates
  - contains the public key to be revoked
  - signed with the corresponding private key

Public-key revocation

- And disseminates it as widely and quickly as possible
- If a key is compromised:
  - e.g., Bob knows the private key of Alice
  - Bob can issue a revocation certificate to revoke the public key of Alice
  - even better for Alice

Any question ?