Pretty Good Privacy

Cryptography & Network Security

Jibi Abraham

COEP TECHNOLOGICAL UNIVERSITY

Shivajinagar, Pune-411 005

(A Unitary Technological University of Govt. of Maharashtra)

Pretty Good Privacy (PGP)

- First version developed by Phil Zimmermann in 2002
- 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)
- It is available free on a variety of platforms
- Not developed or controlled by governmental or standards organizations

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

Order of Encryption and Compression

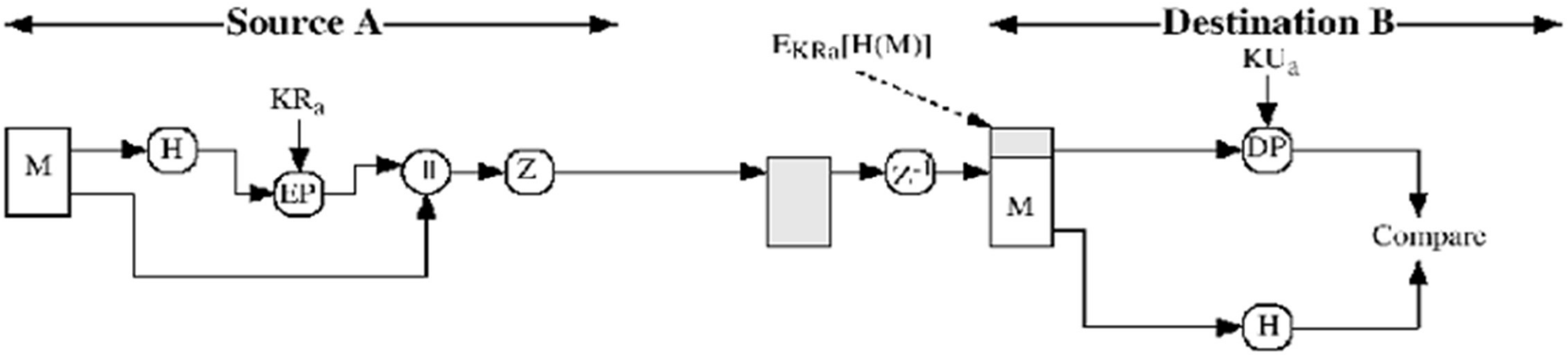
- What order to perform Encryption and Compression so that the effect is of performing these operations is best?
- Compression identifies patterns and replaces them with other patterns
- Encryption of data results in as random as possible. ie. ciphertext will have as few patterns as possible in it
- Encrypted content will be especially difficult to compress
- Compression will get a smaller result, then encryption will totally remove all patterns

Order of Signing and Compression

- What order to perform Encryption and Compression so that the effect is of performing these operations is best?
- Compression algorithms are lossy or lossless
- Signing needs a hash to get generated from input
- If sign and then compression, and if the compression is lossy, the result will have a corrupted signature, thus leaving them useless
- Sign the compressed content, don't have the risk of corruption

PGP Authentication only

- H: SHA-1 is used to generate 160bit hash
- Z –Compression using ZIP (lossless)
- EP – Public Key Encryption with RSA/DSS



PGP Authentication

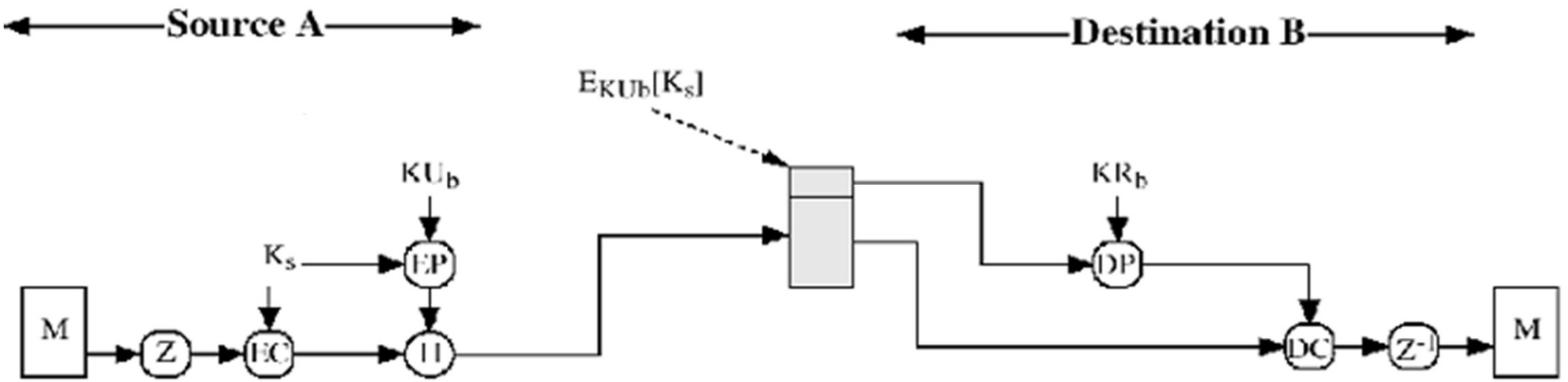
- Sender creates the message
- SHA-1 is used to generate 160bit hash code of the message
- Hash code is encrypted with RSA/DSS using sender's private key
- Receiver uses sender's public key to recover the hash
- Receiver verifies the hash, and if matches, accepts the message.

PGP Compression

- PGP compresses the message after applying the signature, but before encryption
- You can store clear message and signature for later verification
- Encryption after compression to reduce redundancy and strengthen security
- ZIP compression is deterministic
- Average compression ratio of 2.0

PGP only Confidentiality

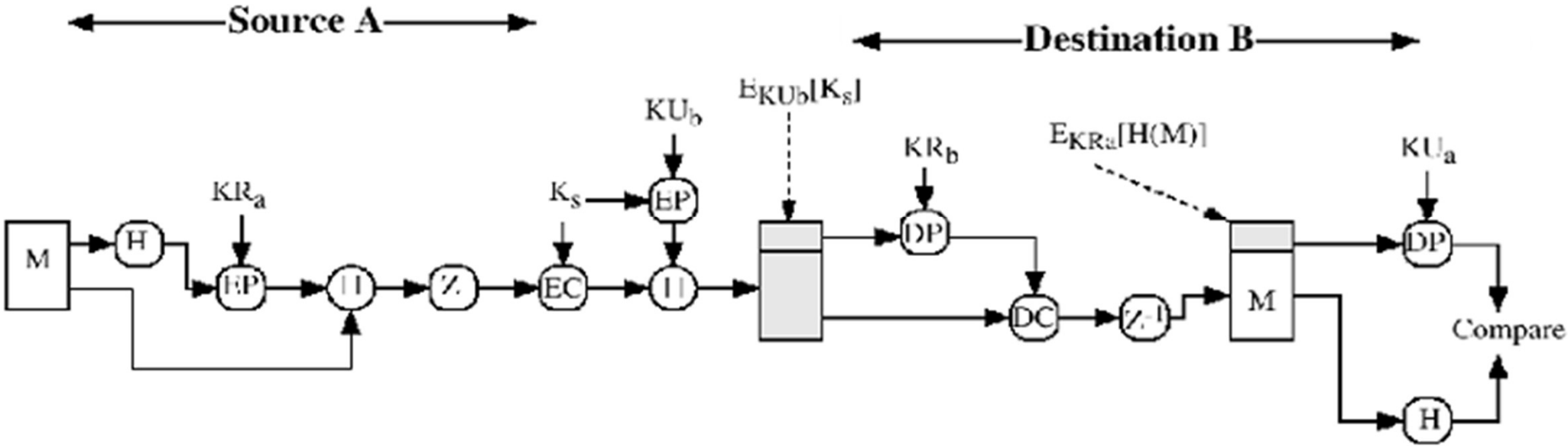
- Z -Compression
- EC – Symmetric Key encryption
- EP – Public Key Encryption



PGP Confidentiality

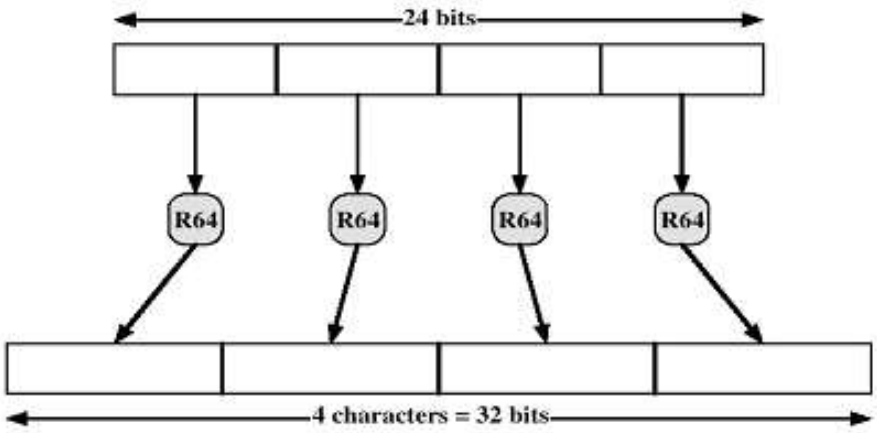
- Sender creates the message and a random 128 bit number as the session key
- The message is encrypted with CAST-128 (or IDEA or 3-DES) with the session key
- Session key is encrypted using RSA (or ElGamal) with receiver's public key and appended to the message
- Receiver uses it's private key to recover session key.
- Message is decrypted using the session key

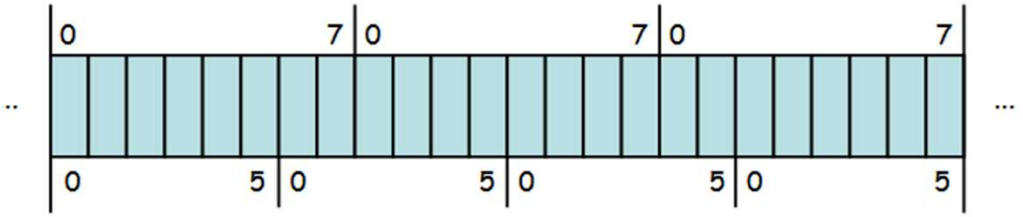
PGP Confidentiality and Authentication



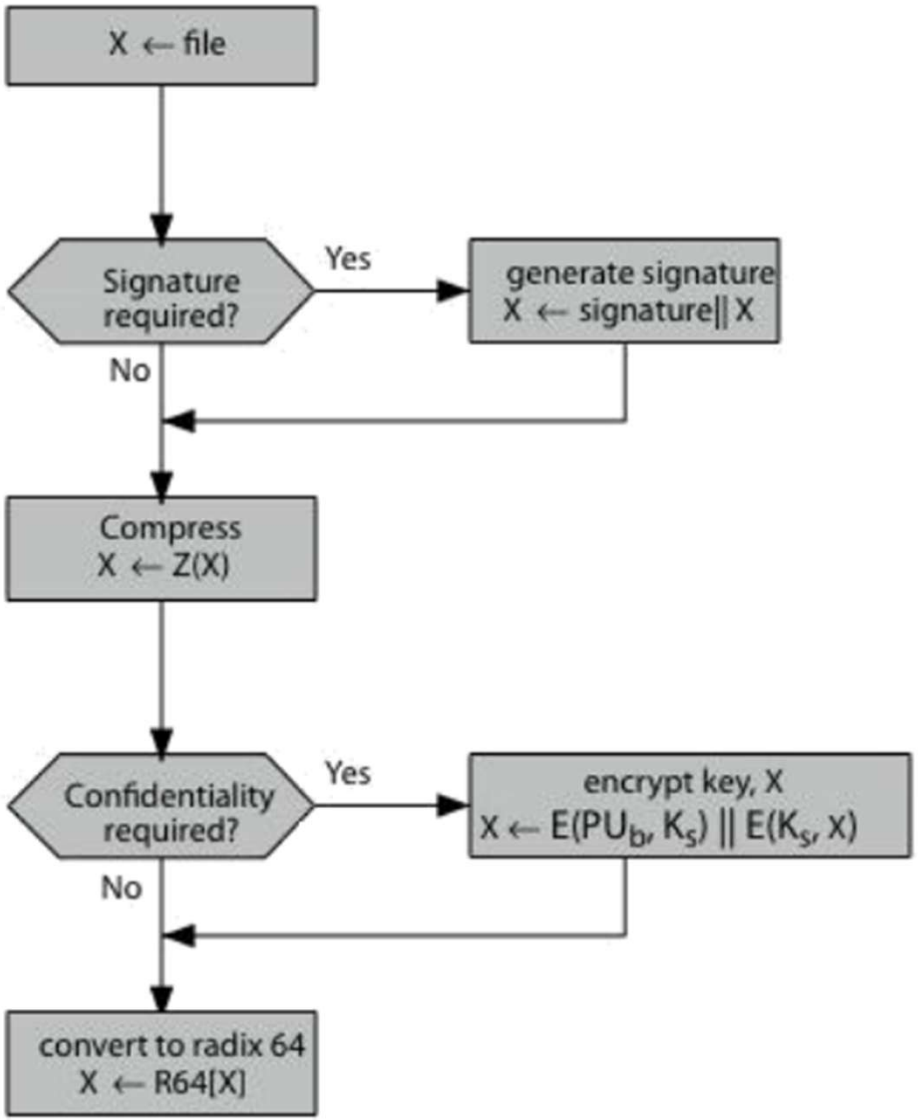
Email Compatibility

- Electronic mail systems permit only ASCII text.
- Converts raw 8bit binary stream to printable ASCII characters
- Radix 64 conversion:
  - 3 8-bit blocks -> 4 7-bit blocks (4bytes)
- Expansion of 133%
  - Net compression = 1.33 * 0.5 = 66.5%

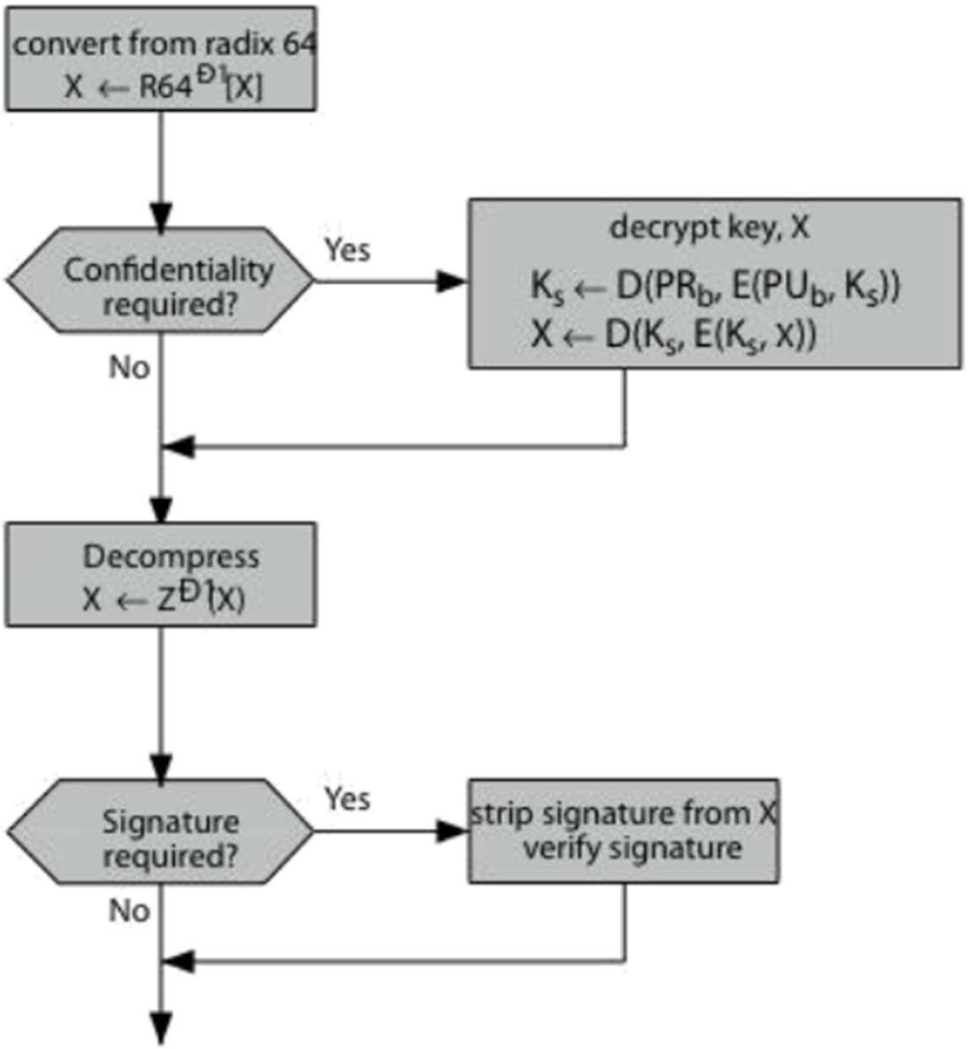




PGP Massage Transmission and Reception



(a) Generic Transmission Diagram (from A)

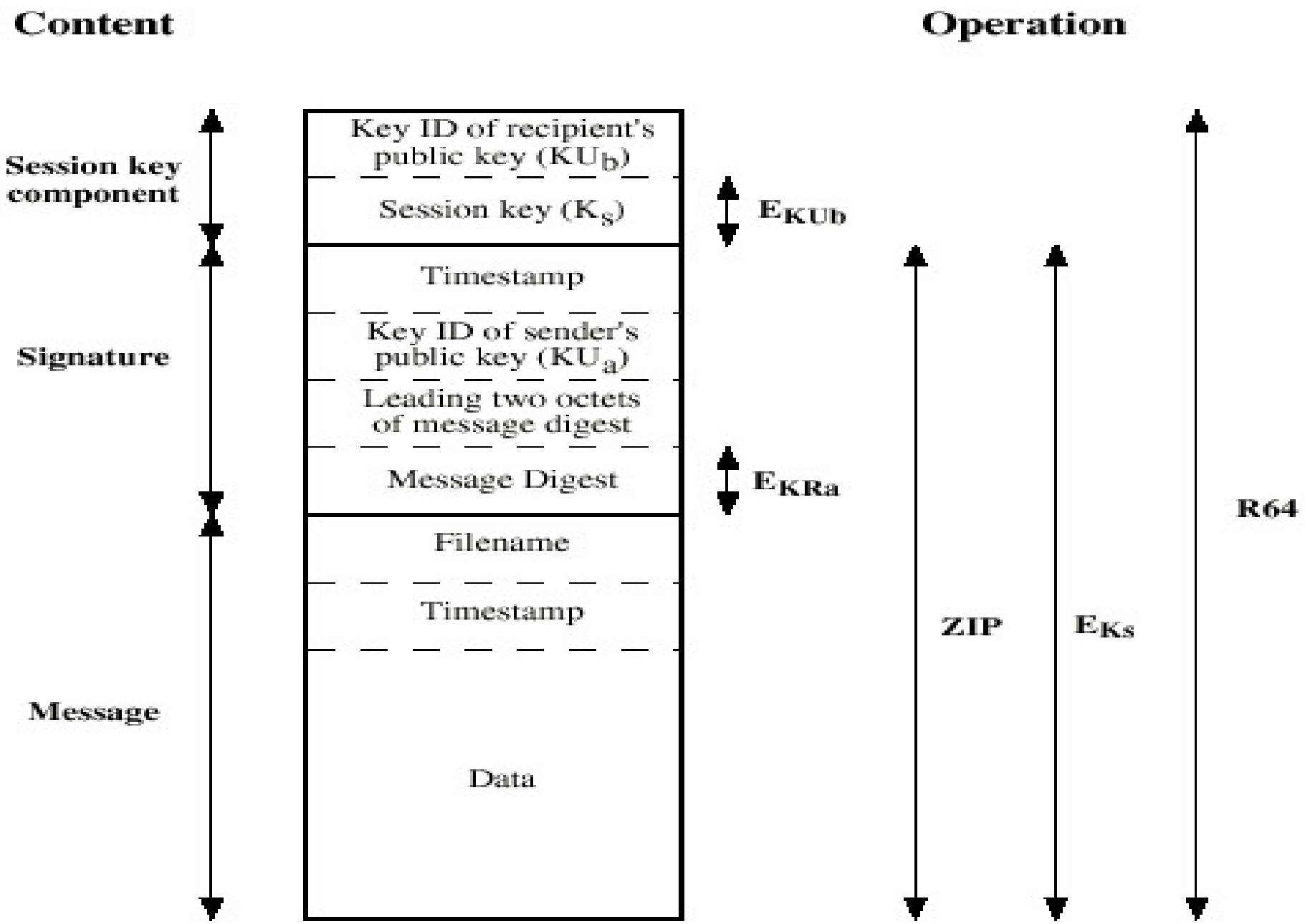


(b) Generic Reception Diagram (to B)

Segmentation and Reassembly

- Email facilities are restricted to a maximum message length of 50,000 octets.
- Longer messages must be broken up into segments.
- Segmentation is done after all processing including compression
- The receiver strips of all e-mail headers and reassembles the block

PGP Message Format



Cryptographic keys

- Session Keys (based on ANSI X12.17)
  - A one-time session key for each message
  - CAST-128 uses 128 bit key (generated by random movements of the user's mouse, and keyboard strokes) and two 64 bits plaintext. Cipher text is the session key
  - session key is then encrypted with the recipient's public key
- Key identifiers
  - Multiple public keys for a single user
  - Assign key ID to each public key of the user (least significant 64bits of the public key)

Managing Key Pairs

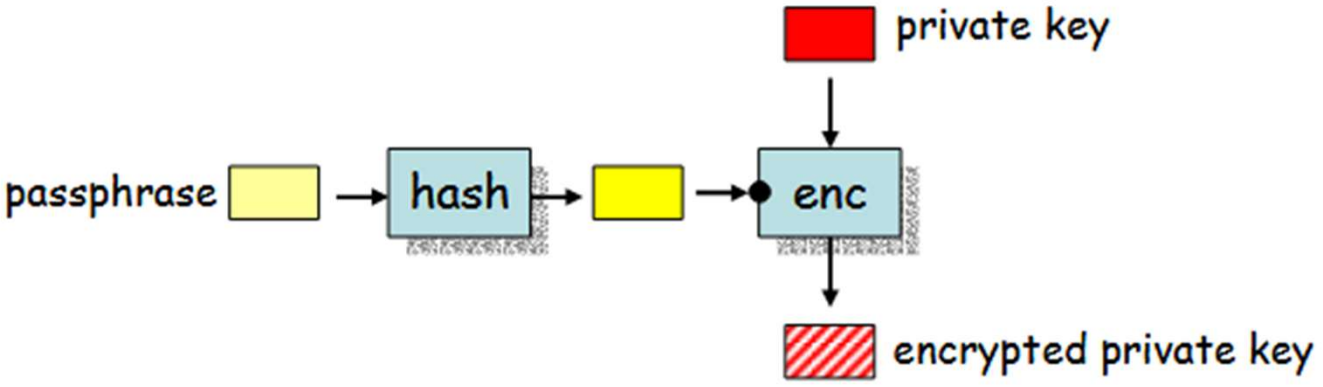
- Given that a user may have multiple public/private key pairs, how do we know which public key was used to encrypt a message
  - Send the public key along with the message. Inefficient, since the key might be thousands of bits.
  - Associate a unique ID with each key pair and send that with the message.
  - Would require that all senders know that mapping of keys to ID's for all recipients
  - Generate an ID likely to be unique for a given user. This is PGP's solution. Use the least significant 64-bits of the key as the ID.
  - This is used by the receiver to verify that he has such a key on his "key ring." The associated private key is used for the decryption.

Key Rings

- A data structure to store keys (for authentication and confidentiality) in a systematic way
- Private key ring
  - Public/private key pairs owned by that node
- Public key ring
  - Public keys of other users known at this node

Private Key Ring

- **Passphrase**: A longer, more memorable phrase (such as a random sequence of words) that is more resistant to brute-force attacks
- **Encryption:** When you create a PGP key pair, you set a passphrase to encrypt your private key
  - Generate passphrasekey = 160bit hash of pass phrase using SHA-1 and pass phrase is discarded
  - Encrypt the private key using CAST-128 with 128 bits of the passphrasekey and passphrasekey is discarded.



Private Key Ring

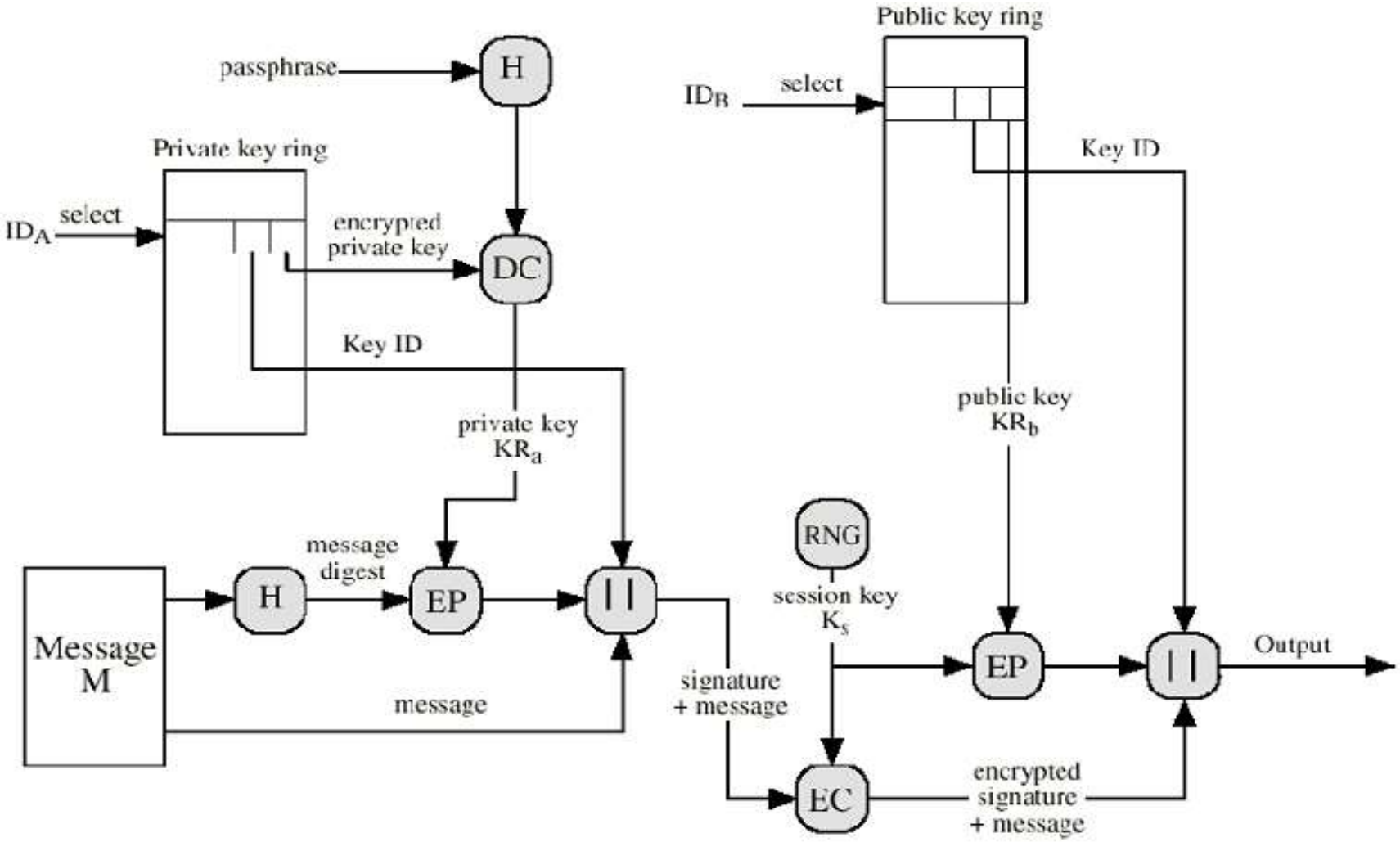
- **Storage:** encrypted private key is stored in private-key ring
- **Accessing the Private Key:** (e.g., to decrypt an email), PGP prompts you for the passphrase
- **Decryption:** PGP uses the passphrase to generate a hash, which is then used to decrypt the private key
- **Loss of Passphrase:** you will not be able to access your private key, leading to a critical security incident

Private Key Ring

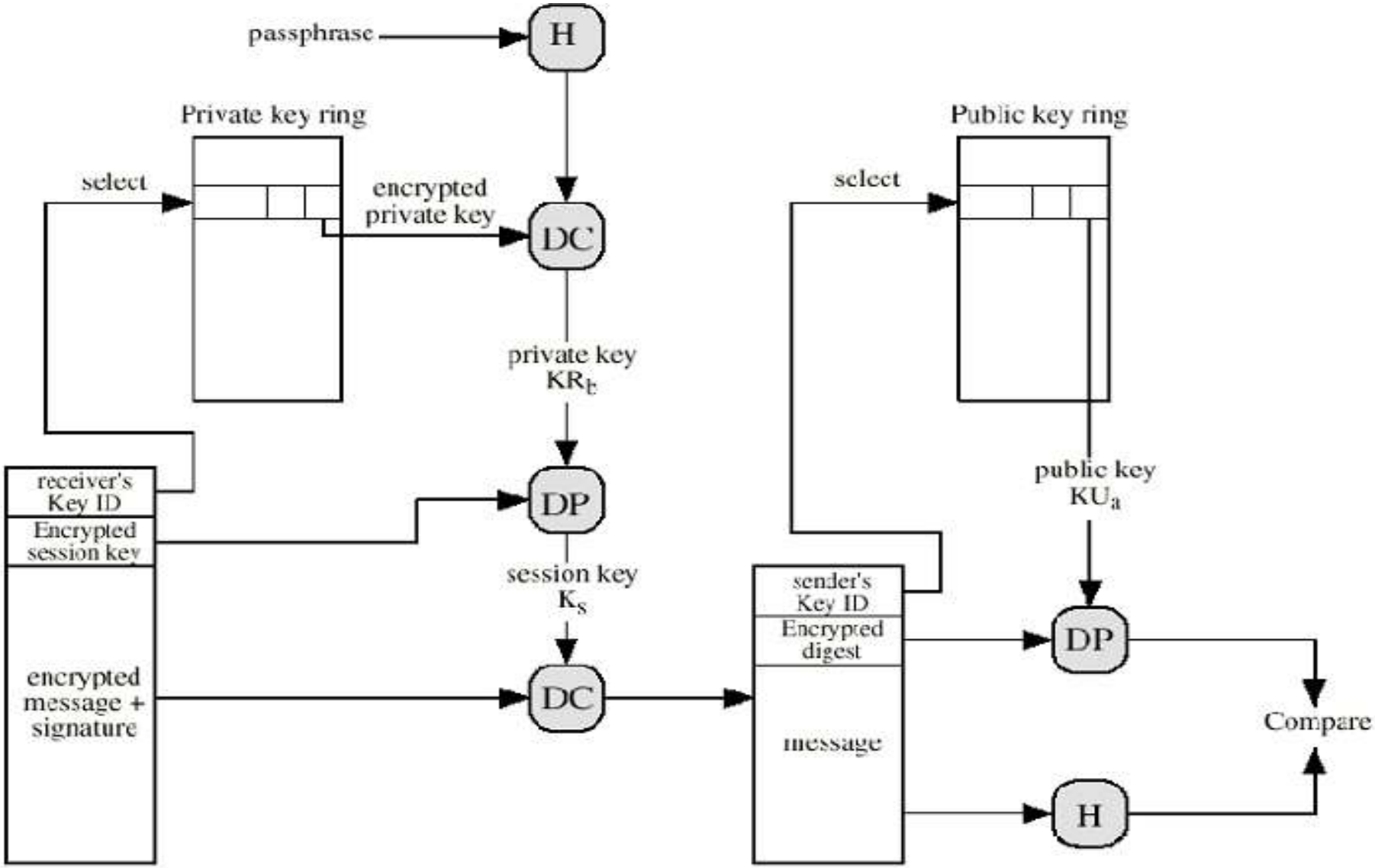
- **Timestamp**: date and time the specific key pair was generated
- **Key ID**: A unique identifier for the key, typically the least significant 64 bits of the public key
- **Public Key**: public portion of the key pair
- **Private Key**: user's secret key, which is encrypted with a passphrase
- **User ID**: Identifies the user associated with the key pair

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
•	•	•	•	•
•	•	•	•	•
•	•	•	•	•

PGP Message Generation



PGP Message Reception



Public Key Ring: Management

- Approach to avoid storage of false keys in the public key ring
- Owner trust field
  - assigned by the user
  - possible values:
    - unknown user
    - usually not trusted to sign
    - usually trusted to sign
    - always trusted to sign

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$_i$		
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.

Public Key Certificates

- A digital certificate is basically a collection of identifying information bound together with a public key and signed by a trusted third party to prove its authenticity
- PGP recognizes two different certificate formats:
  - PGP certificates
  - X.509 certificates
- you can create your own PGP certificate; you must request and be issued an X.509 certificate from a Certification Authority (CA)
- A certificate requires someone to validate that a public key and the name of the key's owner go together
- With PGP certificates, anyone can play the role of validator
- With X.509 certificates, the validator is always a Certification Authority or someone designated by a CA
- Validity is essential in a public key environment where you must constantly establish whether or not a particular certificate is authentic.

Public Key Ring

- PGP assigns a personal trust level to each public key
- Key legitimacy indicates the extent to which PGP trusts that this is a valid public key for this user
- Legitimacy is determined from certificates and chains of certificates, the user's assessment of the trust to be assigned to the key, and various heuristics for computing trust

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$_i$		
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.

Public Key Ring: 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 the signature trust
  - otherwise, signature trust is set to unknown user

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 Trust Levels

- **Trust levels:** You can assign different levels of trust to other people's public keys in your public keyring
- **Ultimate Trust:** You have total trust that the key belongs to its owner. This is reserved for your own keys.
- **Full Trust:** You have verified that a public key belongs to a specific person and you trust that person's judgment in signing other keys.
- **Marginal Trust:** You have some confidence that a key is valid, but you don't fully trust that person's judgment in signing other keys.
- **Untrusted:** The key has not been verified.

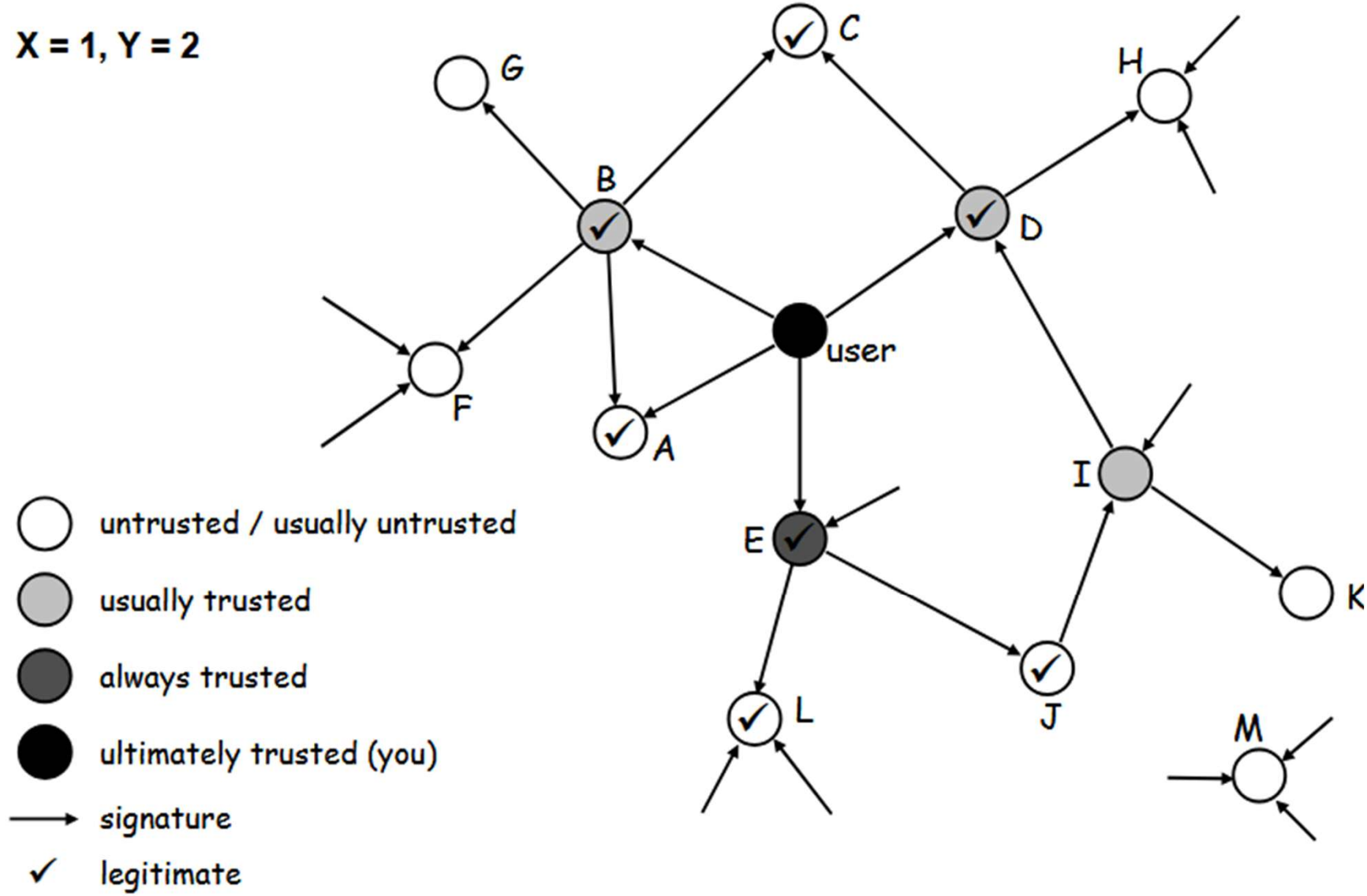
Public Key Ring: Key Legitimacy

- Key legitimacy is an automatically calculated value that determines how much confidence you have that a public key genuinely belongs to the person who claims to own it
- To determine the legitimacy of a new public key, PGP calculates a weighted sum of the signature trust values from all keys that have signed it.
- For example, if you want to verify Bob's key:
  - Import Bob's public key into your key ring
  - You see that Carol has signed Bob's key
  - You have previously assigned a "full trust" level to Carol because you know her well and have verified her key personally
  - Based on your high trust in Carol's signature, your PGP software automatically marks Bob's key as legitimate

Key Legitamacy Calculation

- 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
  - X, Y are user-configurable parameters
  - Always trusted signatures has a weight of 1/X
  - Usually trusted signatures has a weight of 1/Y
- 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

Web of Trust


X = 1, Y = 2
C
G
H
B
D
user
F
A
I
E
K
L
J
M
untrusted / usually untrusted
usually trusted
always trusted
ultimately trusted (you)
signature
legitimate

Public Key Revocation

- 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
- Disseminates it as widely and quickly as possible

PGP Advantages

- Uses best available cryptographic algorithms
- Available on variety of platforms
- Package including the source code is freely available
- Not controlled by Govt. or standards organizations