Computer and Network Security

Computer & Information Sciences & Engineering
University Of Florida
Gainesville, Florida 32611-6120
nemo@cise.ufl.edu

Pretty Good Privacy
and the
Web of Trust

What is Trust and How do you get It?

1 PGP Basics

1.1 History

- Developed by Phil Zimmerman
- Runs on Windows, Unix, Mac, etc.
- Plug-ins for mailers, and stand-alone for files
- Free, open code and commercial versions available
- Uses zip for compression before encryption
- Hides much information via layering (e.g., sender)
- Uses "Web of Trust" for key management, distribution

1.2 Crypto involved

1. Per-message symmetric cipher for encrypting messages
2. Long-term asymmetric cipher for signing and key encryption
3. Hashes for signature and fingerprint generation
4. PGP initially used RSA, IDEA, and MD5
5. PGP now uses DSS/SHA or RSA/SHA for signatures
6. PGP now uses CAST, IDEA, or 3-Key 3DES for symmetric crypto
7. PGP now uses ElGamal or RSA for asymmetric crypto

2 PGP - Authentication Only

2.1 Sender (Alice)

1. Alice creates message, Msg
2. Alice hashes message
3. Alice decrypts hash value with her private key, K_a^{-1} to produce signature, sig
4. Alice prepends sig to Msg to form message M that is sent to Bob
5.

2.2 Receiver (Bob)

1. Bob receives message M from Alice
2. Bob removes sig from M, leaving Msg
3. Bob encrypts sig with Alice's public key, K_a to form signed hash, x
4. Bob hashes Msg from M to form hash, x'
5. Bob compares x and x'; if equal, then signature is good

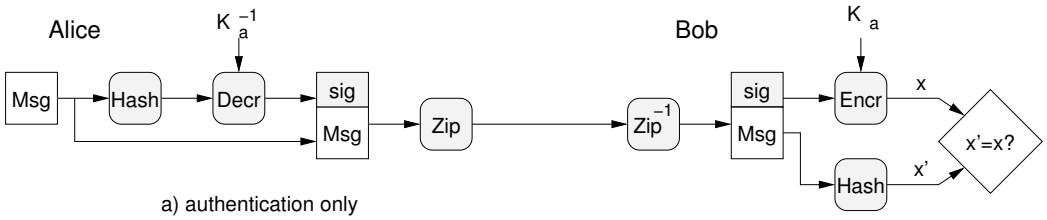

Figure 1: PGP Authentication Only Block Diagram

3 PGP - Confidentiality Only

3.1 Sender (Alice)

1. Alice creates message
2. Alice compresses message
3. Alice creates per-message symmetric cipher key, K_m
4. Alice encrypts per-message symmetric cipher key with Bob's public key K_b, creating Encrypted Message Key (EKM)
5. Alice uses K_m to encrypt the compressed message, creating EM
6. Alice prepends EKM to EM to form message M that is sent to Bob

3.2 Receiver (Bob)

1. Bob receives message M from Alice
2. Bob removes EKM from M, leaving EM
3. Bob decrypts EKM with his private key, K_b^{-1} to extract per-message key, K_m
4. Bob uses K_m to decrypt EM
5. Bob decompresses the result to obtain Msg

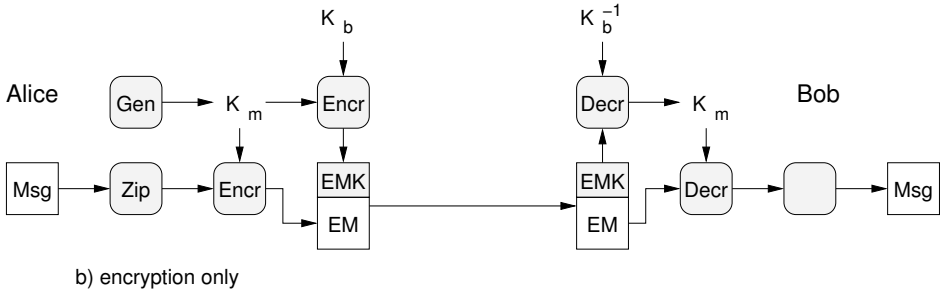

Figure 2: PGP Confidentiality Only Block Diagram

4 PGP - Authentication and Confidentiality

4.1 Sender (Alice)

1. hashes message
2. decrypts hash value with private key, K_a^{-1} to produce signature, *sig*
3. prepends *sig* to Msg
4. compresses result
5. creates per-message symmetric cipher key, K_m
6. encrypts K_m with Bob's public key K_b, to create EKM
7. encrypts compressed, signed message with K_m to create ESM
8. prepends EKM to ESM to form message M that is sent to Bob

4.2 Receiver (Bob)

1. removes EKM from M, leaving ESM
2. decrypts EKM with private key, K_b^{-1} to extract per-message key, K_m
3. uses K_m to decrypt ESM
4. decompresses the result to obtain signed message
5. removes *sig* from signed message, leaving Msg
6. encrypts *sig* with Alice's public key, K_a to form signed hash, x
7. hashes Msg from M to form hash, x'
8. compares x and x'; if equal, then signature is good

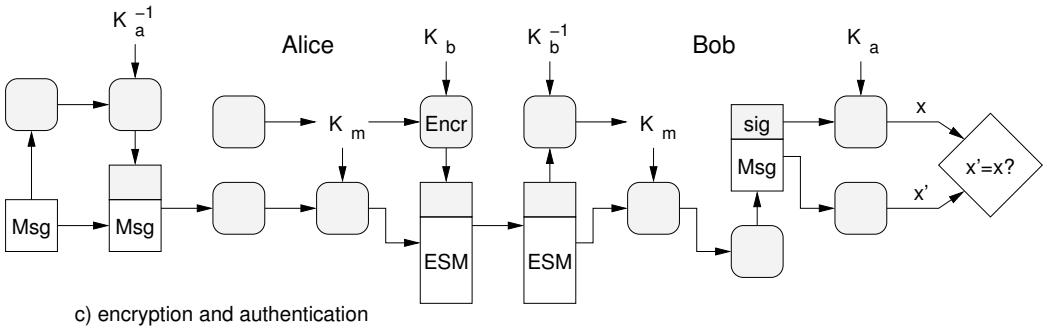

Figure 3: PGP Confidentiality and Authentication Block Diagram

5 PGP Message Format

1. Encryption is done after signing - keeps sender anonymous
2. Encryption is done after signing - sender knows what she signs
3. Encryption is done after compression - more secure
4. Encryption is done after compression - more efficient
5. Hash is done before compression - compression flexible

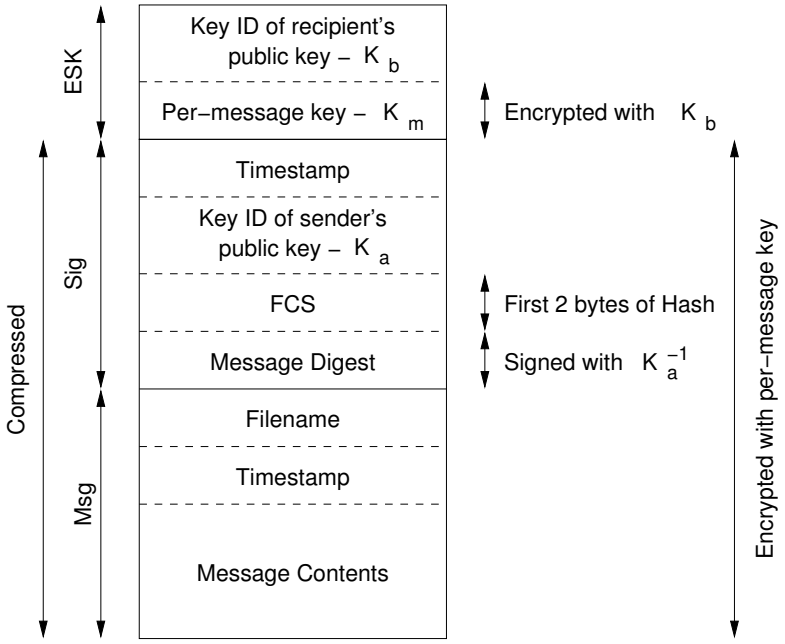

Figure 4: PGP Message Format

6 PGP Key Rings

6.0.1 General

1. Allows multiple UserIDs for user
2. Allows multiple keys per UserID
3. Uses (UserID, KeyID) as primary key in table
4. KeyID is 8 LS bytes of public key

6.0.2 Private-Key Ring

1. Keeps private-public key pairs for user
2. Timestamp of when public-private key pair generated
3. Private key encrypted using a symmetric key generated from the user password using SHA-1
4. UserID is usually the user's email address
5. Defaults to first key for signing
6. Uses KeyID from ESK to find entry for decryption

Private–Key Ring

Timestamp	Key ID*	Public Key	Encrypted Private Key	User ID*
⋮	⋮	⋮	⋮	⋮
TS	LSB–64(K_a)	K_a	$E(K_a^{-1}, H(P))$	A
⋮	⋮	⋮	⋮	⋮

* – used as primary key

Figure 5: PGP Private Key Ring

6.0.3 Public-Key Ring

1. Keeps public keys of other users known to user
2. Timestamp of when this entry was added to table
3. Owner Trust indicates level of trust user has for key owner to vouch for other keys
4. Key Legitimacy indicates whether user believes key is legitimate based on thresholds
5. Signature(s) contains signatures from other key owners vouching for this key-UserID
6. Signature Trust(s) caches the Owner Trust user has for key owners who signed this key
7. Owner Trust, Key Legitmacy, Signature Trust(s) are contained in trust flag byte.
8. Key Legitimacy derived by adding Signature Trusts - if ¿= 1, then key is legit.

Public–Key Ring

Trust Flag Byte

Timestamp	Key ID*	Public Key	User ID*	Owner Trust	Key Legit	Sig Trust(s)	Signature(s)
TS_{ab}	LSB–64(K_b)	K_b	B	OT_{ab}	L_{ab}	ST_{ac}	S_{cb}

* – used as primary key

Figure 6: PGP Public Key Ring

7 PGP Web of Trust

1. Each user assigns degree of trust to others
2. Each user assigns key validity parameters
3. Systems computes key validity from trust levels

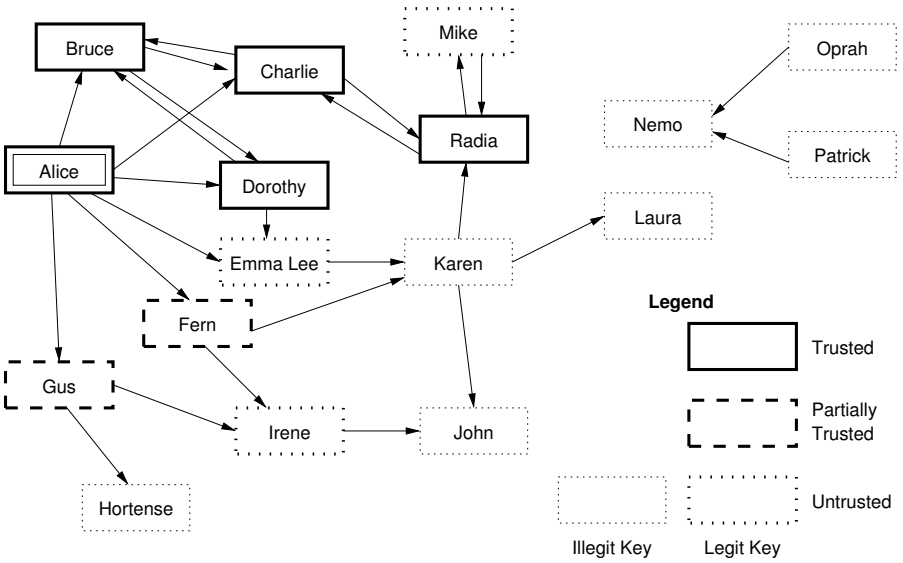

Figure 7: PGP Web of Trust

In Figure 7, Alice is the user of interest, with the rule that two marginally trusted signatures are required to vouch for the legitimacy of another key.

- Alice considers the keys of Bruce, Charlie, Dorothy, Emma Lee, Fern, and Gus to be legitimate because she has signed them herself.
- Alice considers Radia's key to be legit, because Charlie signed it with a legit key and she trusts Charlie
- Alice considers Mike's key to be legit because Radia signed it with a legit key and she trusts Radia (though she has not yet signed Radia's key)
- Alice does not trust Emma Lee, and only marginally trusts Fern, so though both of their keys are legit, Alice does not consider Karen's key to be legit.
- Alice only marginally trusts Fern and Gus, and considers their keys to be legit, so together they can vouch for Irene's key, but Gus alone cannot cause Alice to consider Hortense's key to be legit.
- Karen and Irene's keys are not considered legit, so it does not matter to Alice that they have signed John or Laura's key
- Oprah and Patrick's keys are not considered legit, so it does not matter to Alice that they have signed Nemo's key

8 Insurance Models of Trust

1. User B applies for 'bonding' by insurors
2. Insuror is liable for up to bond value for a bonded key
3. Only certificate is bonded - does not vouch for behavior of user whose key it is
4. Other user A who risks some value V based on B's authenticity can verify certificate chains from insurors to B
5. A must have confidence that insuror will pay if B's certificate has been compromised/forged
6. Transitivity issues - path minimum, overlapping paths, length of path, path constraints