Pretty Good Privacy

Dr. Keyur Parmar

Indian Institute of Information Technology, Vadodara

Outline

Introduction

Privacy is for people who have something to hide.

Privacy is for people who have something to hide.
What is your opinion?

"Using encryption software was something I had long intended to do. ... But the program is complicated, especially for someone who had very little skill in programming and computers, like me. So it was one of those things I had never gotten around to doing."

– Glenn Greenwald (From "No Place to Hide")

GNU PG - A Tool for Secure Communication

- GNU Privacy Guard (GNU PG or GPG) - GNU PG is an alternative to Symantec's PGP software and it was implemented following the open PGP standard.

What can we do with GNU PG?

What can we do with GNU PG?

- Encrypt and decrypt documents

GNU PG - A Tool for Secure Communication

What can we do with GNU PG?

- Encrypt and decrypt documents
- Authenticate documents using digital signature

Cryptography - Basics

Have you ever used cryptography?

Cryptography - Basics

- Modern cryptography

Cryptography - Basics

- Modern cryptography
  - Symmetric-key ciphers (AES, DES, etc.)

Cryptography - Basics

- Modern cryptography
  - Symmetric-key ciphers (AES, DES, etc.)
    - Permutation and substitution

Cryptography - Basics

- Modern cryptography
  - Symmetric-key ciphers (AES, DES, etc.)
    - Permutation and substitution
    - A secret key

Cryptography - Basics

- Modern cryptography
  - Symmetric-key ciphers (AES, DES, etc.)
    - Permutation and substitution
    - A secret key
  - Public-key ciphers (RSA, ElGamal, etc.)

Cryptography - Basics

- Modern cryptography
  - Symmetric-key ciphers (AES, DES, etc.)
    - Permutation and substitution
    - A secret key
  - Public-key ciphers (RSA, ElGamal, etc.)
    - Mathematical functions

Cryptography - Basics

- Modern cryptography
  - Symmetric-key ciphers (AES, DES, etc.)
    - Permutation and substitution
    - A secret key
  - Public-key ciphers (RSA, ElGamal, etc.)
    - Mathematical functions
    - Two keys - one is public, the other is private/secret

Symmetric-key Cipher - Encryption/Decryption

- m - Plaintext (Message)
- c - Ciphertext
- k - Secret-key shared between Alice and Bob

What if an adversary finds the key?

What if an adversary finds the key and algorithm(s) used for encryption and/or decryption?

Should we keep a key secret or algorithm(s)? Why?

The Kerckhoffs' Principle

The Kerckhoffs' Principle

- "The cipher method must not be required to be secret, and it must be able to fall into the hands of the enemy without inconvenience." – Auguste Kerckhoffs (19th century)

The Kerckhoffs' Principle

- "The cipher method must not be required to be secret, and it must be able to fall into the hands of the enemy without inconvenience." – Auguste Kerckhoffs (19th century)
- Security of data should rely on the secrecy of key(s), and not algorithms.

Should we use the Kerckhoffs' Principle? Why?

Which one is relatively easy to keep it as a secret? a key(s) or algorithm(s)?

What if there are n communicating parties? Should we keep n algorithms or n keys?

In adverse circumstances (e.g., the secret information is exposed), what is relatively easy, to change a key or to change an algorithm(s)?

Should we keep an algorithm as well as a key secret?

Should we keep an algorithm as well as a key secret? No.

Should we keep an algorithm as well as a key secret? No.
Why?

- Standard encryption scheme

- Standard encryption scheme
    - Compatibility - Manufacturers can develop low-cost chip-implementation.

- Standard encryption scheme
    - Compatibility - Manufacturers can develop low-cost chip-implementation.
    - Public scrutiny ensures that weaknesses have not been found in ciphers.

A Secret Key

- As the security of data relies on the secrecy of key(s), we have to protect the secret key.

A Secret Key

- As the security of data relies on the secrecy of key(s), we have to protect the secret key.
- Which of the following is a (more) secure key? why?

A Secret Key

- As the security of data relies on the secrecy of key(s), we have to protect the secret key.
- Which of the following is a (more) secure key? why?
    1. `Password`

A Secret Key

- As the security of data relies on the secrecy of key(s), we have to protect the secret key.
- Which of the following is a (more) secure key? why?
  1. `Password`
  2. `drowssaP`

A Secret Key

- As the security of data relies on the secrecy of key(s), we have to protect the secret key.
- Which of the following is a (more) secure key? why?
  1. `Password`
  2. `drowssaP`
  3. `myfavmovieissamsara`

A Secret Key

- As the security of data relies on the secrecy of key(s), we have to protect the secret key.
- Which of the following is a (more) secure key? why?
  1. `Password`
  2. `drowssaP`
  3. `myfavmovieissamsara`
  4. `Password1!`

A Secret Key

- Ideal characteristics of a secret key

A Secret Key

- Ideal characteristics of a secret key
    1. High Entropy (randomness)

A Secret Key

- Ideal characteristics of a secret key
  1. High Entropy (randomness)
  2. Sufficiently large key-size

A Secret Key

- Ideal characteristics of a secret key
  1. High Entropy (randomness)
  2. Sufficiently large key-size
- They together produces sufficiently large key-space (i.e., the set of possible keys).

A Secret Key

- Ideal characteristics of a secret key
    1. High Entropy (randomness)
    2. Sufficiently large key-size
- They together produces sufficiently large key-space (i.e., the set of possible keys).
- Advancements in Hardware - The key-length that is considered as sufficient in 1974 (at the time of DES cipher) is no longer considered as sufficient.

Key-space - DES and AES

- Data Encryption Standard (56-bit key)

Key-space - DES and AES

- Data Encryption Standard (56-bit key)
  - 72057594037927936 (or 2^{56}) keys

Key-space - DES and AES

- Data Encryption Standard (56-bit key)
  - 72057594037927936 (or 2^{56}) keys
- Advanced Encryption Standard (128-bit key)

Key-space - DES and AES

- Data Encryption Standard (56-bit key)
  - 72057594037927936 (or 2^{56}) keys
- Advanced Encryption Standard (128-bit key)
  - 340282366920938463463374607431768211456 (or 2^{128}) keys

Key-space - DES and AES

- Data Encryption Standard (56-bit key)
  - 72057594037927936 (or 2^{56}) keys
- Advanced Encryption Standard (128-bit key)
  - 340282366920938463463374607431768211456 (or 2^{128}) keys
- Advanced Encryption Standard (256-bit key)

Key-space - DES and AES

- Data Encryption Standard (56-bit key)
  - 72057594037927936 (or 2^{56}) keys
- Advanced Encryption Standard (128-bit key)
  - 340282366920938463463374607431768211456 (or 2^{128}) keys
- Advanced Encryption Standard (256-bit key)
  - 115792089237316195423570985008687907853269984665640564039457584007913129639936 (or 2^{256}) keys

Public-key Cryptography

Why do we need the public-key cryptography?

Is it difficult to distribute a key/secret between communicating parties?

How many (symmetric) keys do we need to facilitate the communication between n parties?

$$\frac{n(n-1)}{2}$$

Public-key Cryptography

- The need for public-key cryptosystems...

Public-key Cryptography

- The need for public-key cryptosystems...
  - Key distribution problems of symmetric-key cryptosystems

Public-key Cryptography

- The need for public-key cryptosystems...
  - Key distribution problems of symmetric-key cryptosystems
  - Digital signatures

Public-key Cryptography

- History

Diffie and Hellman, New Directions in Cryptography, IEEE Transactions on Information Theory, 6, (1976).

Public-key Cryptography

- History
  - Diffie and Hellman - 1976

Diffie and Hellman, New Directions in Cryptography, IEEE Transactions on Information Theory, 6, (1976).

Public-key Cryptography

- History
  - Diffie and Hellman - 1976
  - R. Merkle - 1975 (did not publish until 1978)

Diffie and Hellman, New Directions in Cryptography, IEEE Transactions on Information Theory, 6, (1976).

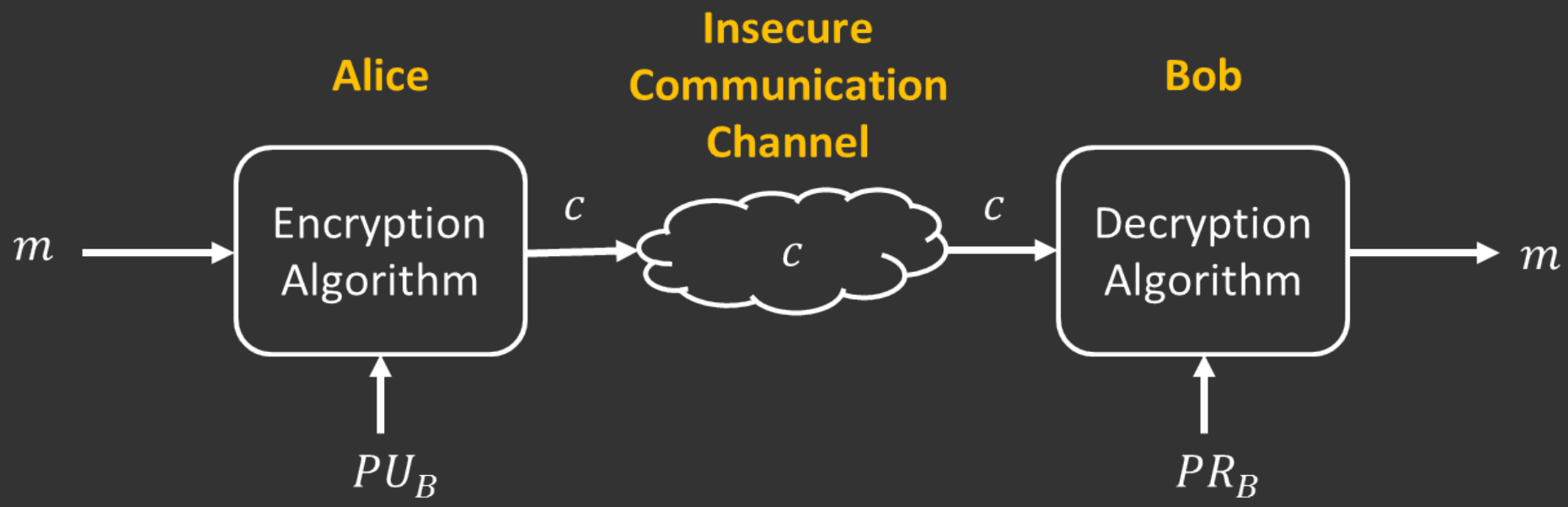
Public-key Cryptography

- History
  - Diffie and Hellman - 1976
  - R. Merkle - 1975 (did not publish until 1978)
  - Bobby Inman (Director of the NSA) - Discovered at NSA in the mid-1960s

---

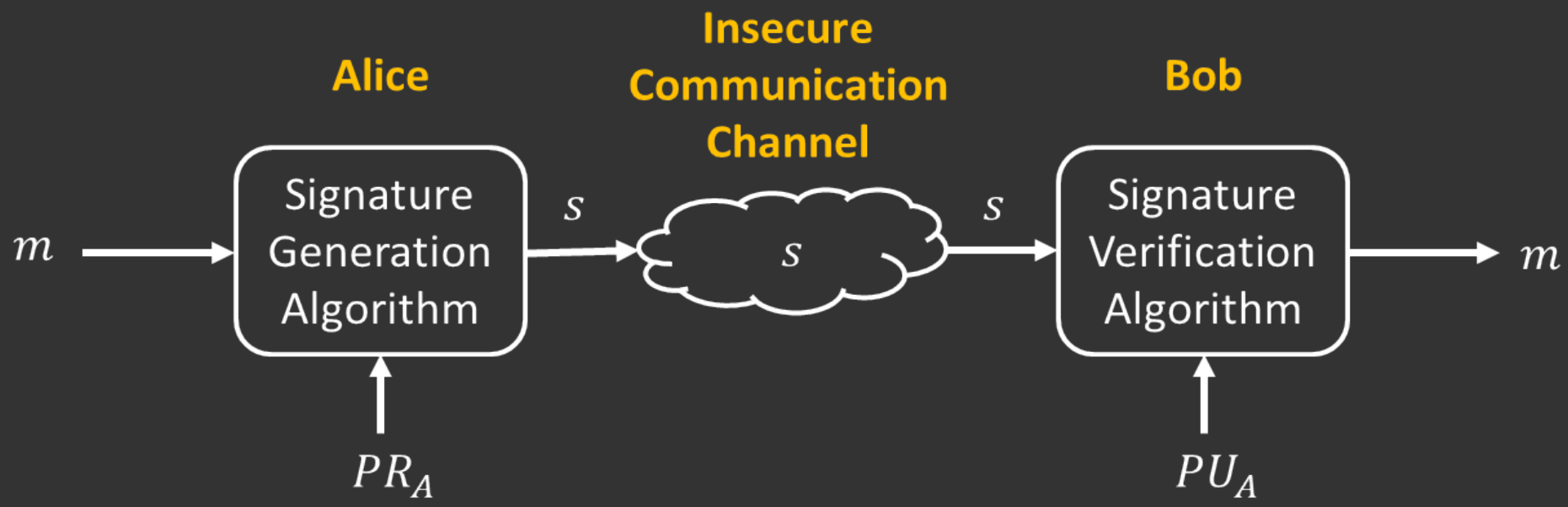
Diffie and Hellman, New Directions in Cryptography, IEEE Transactions on Information Theory, 6, (1976).

Public-key Cryptosystem - Encryption/Decryption



- m - Plaintext (Message)
- c - Ciphertext
- PU_B - Bob's public-key
- PR_B - Bob's private-key

Public-key Cryptosystem - Digital Signature



- m - Plaintext (Message)
- c - Ciphertext
- PU_A - Alice's public-key
- PR_A - Alice's private-key

Public-key Cryptosystems

- Classification of public-key cryptosystems

Public-key Cryptosystems

- Classification of public-key cryptosystems
  - Encryption/decryption - Uses recipient's public key to encrypt the message.

Public-key Cryptosystems

- Classification of public-key cryptosystems
  - Encryption/decryption - Uses recipient's public key to encrypt the message.
  - Digital signature - Uses sender's private key to sign the message.

Public-key Cryptosystems

- Classification of public-key cryptosystems
  - Encryption/decryption - Uses recipient's public key to encrypt the message.
  - Digital signature - Uses sender's private key to sign the message.
  - Key exchange - Uses the private key(s) of sender and/or recipient.

Is it difficult to design a public-key cryptosystem? If we have to design a public-key cryptosystem, what do we need?

Trapdoor One-way Function

- Trap-door one-way function
  - Easy to compute in one direction.
  - Infeasible to compute in the other direction without the trapdoor information.
- Trapdoor one-way function

$$
\begin{aligned}
C &= f_k(M) && \text{easy, if } k \text{ and } M \text{ are known} \\
M &= f_k^{-1}(C) && \text{easy, if } k \text{ and } C \text{ are known} \\
M &= f_k^{-1}(C) && \text{infeasible, if } C \text{ is known but } k \text{ is unknown}
\end{aligned}
$$

Here, "easy" means possible in polynomial time.

Public-key Cryptosystems - Attacks

- Brute-force attack

Public-key Cryptosystems - Attacks

- Brute-force attack
  - Use large keys. How large (keys)?

Public-key Cryptosystems - Attacks

- Brute-force attack
  - Use large keys. How large (keys)? Large enough to make brute-force attack impractical and small enough to keep the encryption and decryption feasible.

Public-key Cryptosystems - Attacks

- Brute-force attack
  - Use large keys. How large (keys)? Large enough to make brute-force attack impractical and small enough to keep the encryption and decryption feasible.
- Find ways to compute the private key given the public key.

Public-key Cryptosystems - Attacks

- Brute-force attack
  - Use large keys. How large (keys)? Large enough to make brute-force attack impractical and small enough to keep the encryption and decryption feasible.
- Find ways to compute the private key given the public key.
- Known/Chosen plaintext/ciphertext attacks to derive the private key.

Cryptographic Hash Functions

What is a cryptographic hash function?

Cryptographic Hash Function

- Input:

Cryptographic Hash Function

- Input:
  - A variable-length data block M

Cryptographic Hash Function

- Input:
  - A variable-length data block M
- Output:

Cryptographic Hash Function

- Input:
  - A variable-length data block M
- Output:
  - A fixed-size hash code $h = H(M)$

How to design a cryptographically secure hash function?

Cryptographic Hash Functions

Cryptographic Hash Functions

1. One-way property (preimage resistant)

Cryptographic Hash Functions

1. One-way property (preimage resistant)
2. Second preimage resistant (weak collision resistant)

Cryptographic Hash Functions

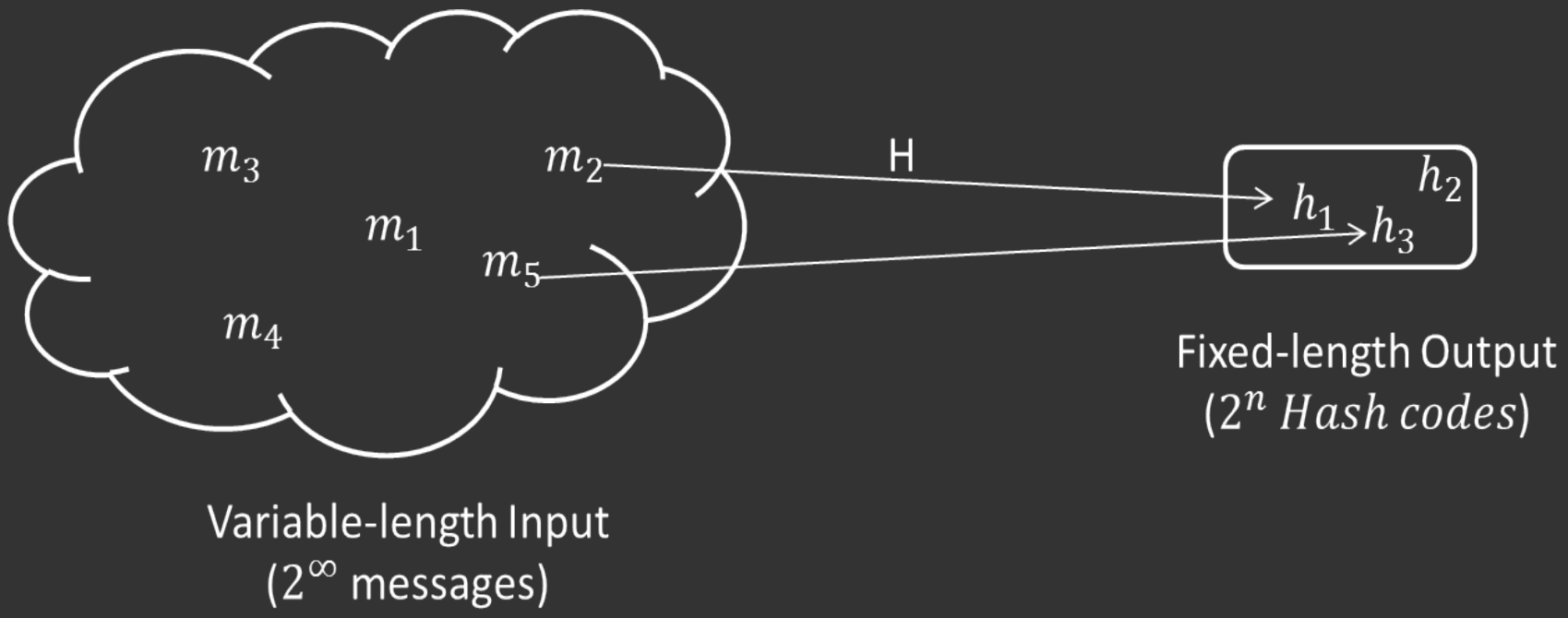
1. One-way property (preimage resistant)
2. Second preimage resistant (weak collision resistant)
3. Collision resistant (strong collision resistant)

One-way Property (Preimage Resistant)

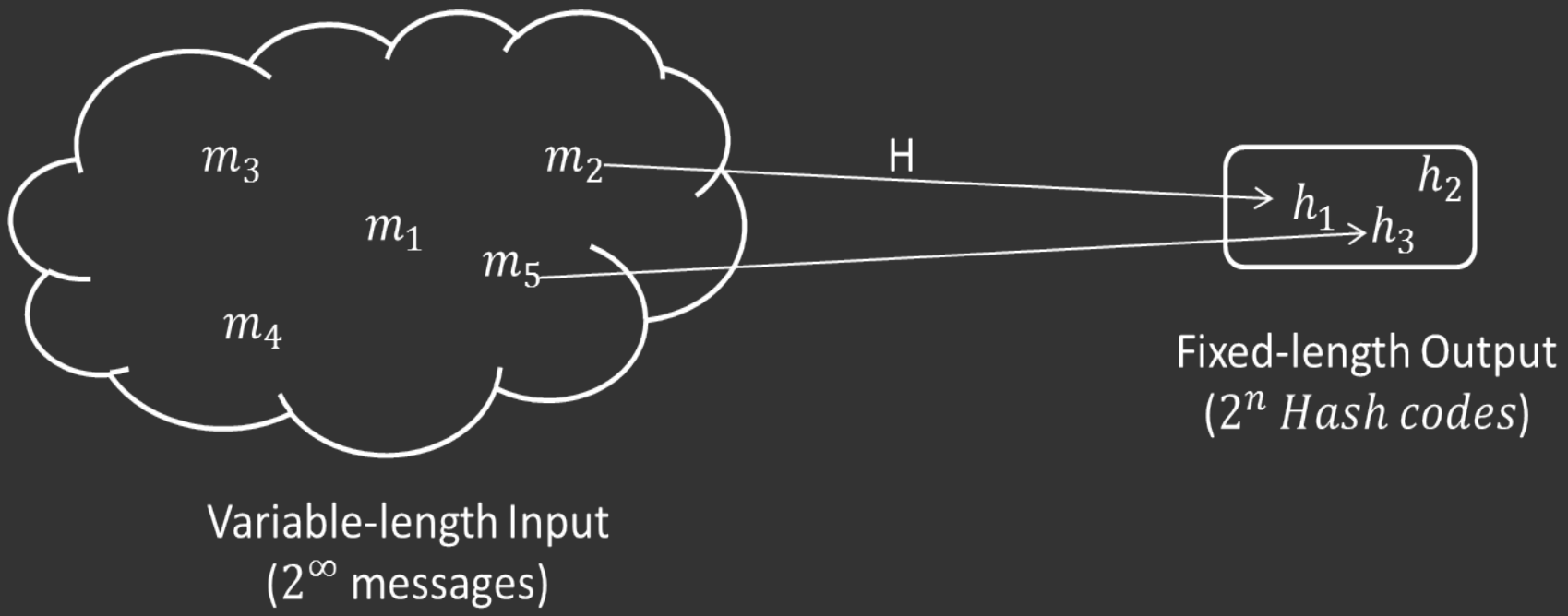
Given a hash code h where $h = H(m)$, it must be computationally infeasible to find the input message m.

Note: Preimage means input to the hash function.

Requirements for a Cryptographic Hash Functions



Requirements for a Cryptographic Hash Functions



- As there is a many-to-one mapping, for any given hash code h, there will be multiple preimages.

Second Preimage Resistant (Weak Collision Resistant)

For any given message m_1, it must be computationally infeasible to find another message m_2 such that $m_1 \neq m_2$ and $H(m_1) = H(m_2)$.

Collision Resistant (Strong Collision Resistant)

It must be computationally infeasible to find a pair of messages (m_1, m_2) such that $m_1 \neq m_2$ and $H(m_1) = H(m_2)$.

Security Attacks

- Attacks on hash functions
  - Brute-force attacks
  - Cryptanalysis

What should be the length of the hash code h to prevent the preimage/second preimage/collision-resistant attack?

Cryptographic Hash Functions - Examples

- Secure Hash Algorithm (SHA)-2 (Year: 2002) and SHA-3 (Year: 2015)

Cryptographic Hash Functions - Examples

- Secure Hash Algorithm (SHA)-2 (Year: 2002) and SHA-3 (Year: 2015)
    - SHA-224

Cryptographic Hash Functions - Examples

- Secure Hash Algorithm (SHA)-2 (Year: 2002) and SHA-3 (Year: 2015)
  - SHA-224
  - SHA-256

Cryptographic Hash Functions - Examples

- Secure Hash Algorithm (SHA)-2 (Year: 2002) and SHA-3 (Year: 2015)
  - SHA-224
  - SHA-256
  - SHA-384

Cryptographic Hash Functions - Examples

- Secure Hash Algorithm (SHA)-2 (Year: 2002) and SHA-3 (Year: 2015)
  - SHA-224
  - SHA-256
  - SHA-384
  - SHA-512

Digital Signature

What is a signature? Is it useful? How?

Goal: To generate a signature for digital data, e.g., sign a text message or a video.

Goal: To generate a signature for digital data, e.g., sign a text message or a video.

Is it difficult to sign digital data?

Why Do We Use Digital Signature?

- Masquerade

Why Do We Use Digital Signature?

- Masquerade
- Content modification

Why Do We Use Digital Signature?

- Masquerade
- Content modification
- Sequence modification

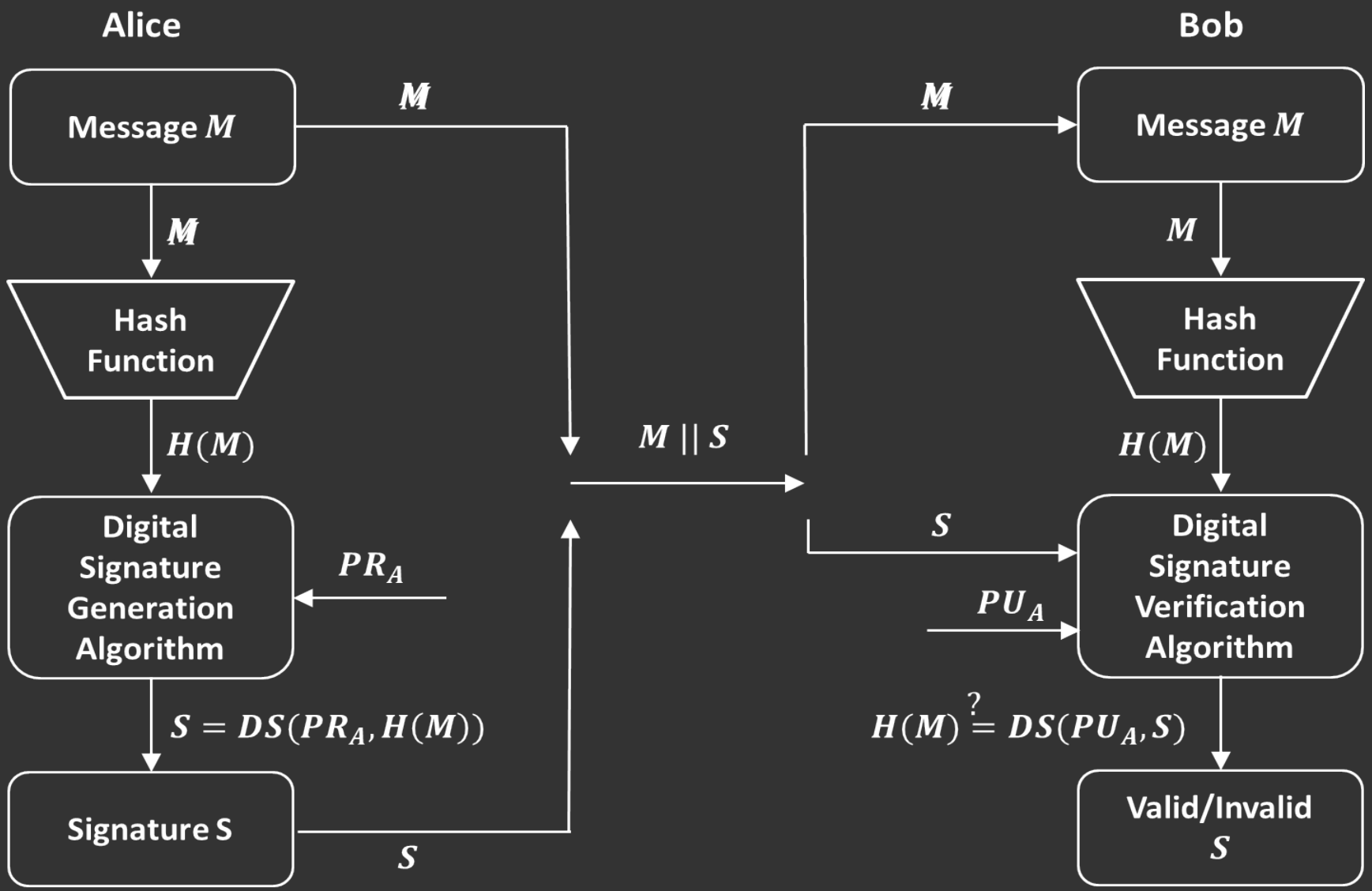
Why Do We Use Digital Signature?

- Masquerade
- Content modification
- Sequence modification
- Timing modification

Why Do We Use Digital Signature?

- Masquerade
- Content modification
- Sequence modification
- Timing modification
- Source repudiation

Digital Signature



PR_A - Alice's Private Key PU_A - Alice's Public Key

Digital Signature - Algorithms

- Elgamal Digital Signature Scheme

Digital Signature - Algorithms

- Elgamal Digital Signature Scheme
- NIST Digital Signature Algorithm (DSA)

Digital Signature - Algorithms

- Elgamal Digital Signature Scheme
- NIST Digital Signature Algorithm (DSA)
- Elliptic Curve Digital Signature Algorithm (ECDSA)

Digital Signature - Algorithms

- Elgamal Digital Signature Scheme
- NIST Digital Signature Algorithm (DSA)
- Elliptic Curve Digital Signature Algorithm (ECDSA)
- Schnorr Digital Signature Scheme

Digital Signature - Algorithms

- Elgamal Digital Signature Scheme
- NIST Digital Signature Algorithm (DSA)
- Elliptic Curve Digital Signature Algorithm (ECDSA)
- Schnorr Digital Signature Scheme
- RSA-PSS Digital Signature Scheme

The GNU Privacy Guard

The GNU Privacy Guard

- Symmetric ciphers

The GNU Privacy Guard

- Symmetric ciphers
- Asymmetric ciphers

Symmetric-key Ciphers

Supported Cryptosystems

Encryption Algorithms:

- IDEA, 3DES, CAST5, BLOWFISH, AES, AES192, AES256, TWOFISH, CAMELLIA128, CAMELLIA192, CAMELLIA256

Hash Functions:

- MD5, SHA1, RIPEMD160, SHA256, SHA384, SHA512, SHA224

Symmetric-key Ciphers - Encryption

```
gpg -c --cipher-algo AES <file1.txt>
```

- For encryption, use `--symmetric` (or `-c`) option.

Symmetric-key Ciphers - Encryption

```
gpg -c --cipher-algo AES <file1.txt>
```

- For encryption, use `--symmetric` (or `-c`) option.
- Output: An encrypted file - `<file1.txt.gpg>`.

Symmetric-key Ciphers - Encryption

```
gpg -c --cipher-algo AES <file1.txt>
```

- For encryption, use `--symmetric` (or `-c`) option.
- Output: An encrypted file - `<file1.txt.gpg>`.
- What is missing in the above command?

Symmetric-key Ciphers - Encryption

```
gpg -c --cipher-algo AES <file1.txt>
```

- For encryption, use `--symmetric` (or `-c`) option.
- Output: An encrypted file - `<file1.txt.gpg>`.
- What is missing in the above command? A secret key. You have to provide a passphrase that can be used as a secret key.

Symmetric-key Ciphers - Encryption

- To specify the output file name, during encryption, use `-o` (or `--output`) option.

```
gpg -o <ofile1> -c --cipher-algo AES <ifile1.txt>
```

Symmetric-key Ciphers - Encryption

- To generate an ASCII version of output having `.asc` extension (Default: Binary file with `.gpg` extension), use the option `--armor`.

```
gpg --armor -o <ofile1> -c --cipher-algo AES
                    <ifile1.txt>
```

Symmetric-key Ciphers - Decryption

- For decryption, use `--decrypt` (or `-d`) option.
- Output: The output is printed on the "standard output" (e.g., screen).

```
gpg -d <file1.txt.gpg>
```

Symmetric-key Ciphers - Decryption

- To specify the output file name, during decryption, use `-o` (or `--output`) option.

```
gpg -o <ofile1> -d <file1.txt.gpg>
```

Symmetric-key Ciphers - Message Authentication

- To generate a hashcode

```
gpg --print-md sha512 <file1.txt>
```

- Output: Hashcode (will be displayed on the standard output, e.g., screen).

Symmetric-key Ciphers - Message Authentication

- To specify the output file name, use redirection operator (>).

```
gpg --print-md sha512 <ifile1.txt> > <ofile1.txt>
```

Public-key Ciphers

Public-key Ciphers

- A key-pair generation
- Encryption and decryption
- Signature generation and verification

A Key-pair Generation

```
gpg --full-generate-key
```

```
Please select what kind of key you want:
(1) RSA and RSA (default)
(2) DSA and Elgamal
(3) DSA (sign only)
(4) RSA (sign only)
Your selection?  1
```

- Digital signature algorithm and encryption algorithm
- A primary key must be capable of making signatures.

A Key-pair Generation

```
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want?  (2048) 2048
Requested keysize is 2048 bits
```

- Large keys increase the security against brute force. However, they increase computation and communication overhead.
- Once selected, the key-size cannot be changed.

A Key-pair Generation

```
Please specify how long the key should be valid.
0 = key does not expire
<n> = key expires in n days
<n>w = key expires in n weeks
<n>m = key expires in n months
<n>y = key expires in n years
Key is valid for?  (0) 1w
Key expires at Thursday 04 July 2019 06:56:33 AM IST
Is this correct?  (y/N) y
```

A Key-pair Generation

```
You need a user ID to identify your key; the
software constructs the user ID from the Real Name,
Comment and Email Address in this form:
‘‘Heinrich Heine (Der Dichter)
<heinrichh@duesseldorf.de>"
Real name:  Keyur
Email address:  keyur@iiitvadodara.ac.in
Comment:  STTP at SVNIT
You selected this USER-ID:
"Keyur (STTP at SVNIT) <keyur@iiitvadodara.ac.in>"
Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit?
O
You need a Passphrase to protect your secret key.
```

A Key-pair Generation

- Real name (Mandatory)

A Key-pair Generation

- Real name (Mandatory)
- Comment (Optional)

A Key-pair Generation

- Real name (Mandatory)
- Comment (Optional)
- Email ID (Optional)

A Key-pair Generation

- Real name (Mandatory)
- Comment (Optional)
- Email ID (Optional)
- It is possible to create additional user IDs (after key generation) for a key to be used in different contexts, but it is not possible to modify existing user IDs.

A Key-pair Generation

- Real name (Mandatory)
- Comment (Optional)
- Email ID (Optional)
- It is possible to create additional user IDs (after key generation) for a key to be used in different contexts, but it is not possible to modify existing user IDs.
- Passphrase is optional and can be omitted (not a good practice).

A Key-pair Generation

- Generate a lot of random bytes (e.g., type on the keyboard, move the mouse, etc.) to gain enough entropy for the PRNG.

```
gpg:  key D84FBEC8 marked as ultimately trusted
public and secret key created and signed.
...
pub 2048R/D84FBEC8 2019-06-27 [expires:  2019-07-04]
Key fingerprint = 8F0D 4EE4 F76B 0B9D 5634 D20E 2A31
2F9C D84F BEC8
uid Keyur (STTP at SVNIT) <keyur@iiitvadodara.ac.in>
sub 2048R/86EDA8DC 2019-06-27 [expires:  2019-07-04]
```

What to do when you forget your pass-phrase?

Generate a Revocation Certificate

```
gpg --output revoke.asc --gen-revoke <Key ID>
```

To List the Keys of the Key-ring

```
gpg --list-keys
```

```
pub 2048R/D84FBEC8 2019-06-27 [expires:  2019-07-04]
uid Keyur (STTP at SVNIT) <keyur@iiitvadodara.ac.in>
sub 2048R/86EDA8DC 2019-06-27 [expires:  2019-07-04]
```

Exchanging Keys - To Export a Key

```
gpg --output Keyur.gpg --armor --export <Key ID>
```

- <Key ID> - Information about the public-key that is exported.
- By deafult, the key is exported in a binary format (inconvenient for email, webpage, etc.).
- To export a key in ASCII armored format, use `--armor`.

A Public Key - ASCII Format

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version:  GnuPG v1
mQENBFOUG5UBCAChN4WGcs9wP9TzKiEj6KlxhM7
...
qw9JgEaI+2veFOXlEVBrwRSejvMuvPnO49uCo8X
-----END PGP PUBLIC KEY BLOCK-----
```

Exchanging Keys - To Import a Key

```
gpg --import <Key ID>
```

- A public-key that is imported will be added to your key-ring.

Exchanging Keys - To Import a Key

```
gpg --import <Key ID>
```

- A public-key that is imported will be added to your key-ring.
- An imported key must be validated by verifying the key's fingerprint (ask the owner to get the original fingerprint).

Exchanging Keys - To Import a Key

```
gpg --import <Key ID>
```

- A public-key that is imported will be added to your key-ring.
- An imported key must be validated by verifying the key's fingerprint (ask the owner to get the original fingerprint).
- If it is a valid key, then sign it (and export it to public-key servers) to certify that it is a valid key.

Exchanging Keys - To Import a Key

```
gpg --edit-key <Key ID>
```

- fpr: To verify the fingerprint of an imported public-key

Exchanging Keys - To Import a Key

```
gpg --edit-key <Key ID>
```

- fpr: To verify the fingerprint of an imported public-key
- sign: To sign an imported public-key (available in your public-key ring)

Exchanging Keys - To Import a Key

```
gpg --edit-key <Key ID>
```

- fpr: To verify the fingerprint of an imported public-key
- sign: To sign an imported public-key (available in your public-key ring)
- save: To save the changes

Exchanging Keys - To Import a Key

```
gpg --edit-key <Key ID>
```

- fpr: To verify the fingerprint of an imported public-key
- sign: To sign an imported public-key (available in your public-key ring)
- save: To save the changes
- quit: To quit the `edit-key` menu

Delete a Key From a Key-ring

- To delete a private-key

Delete a Key From a Key-ring

- To delete a private-key

```
gpg --delete-secret-key <Key ID>
```

Delete a Key From a Key-ring

- To delete a private-key

```
gpg --delete-secret-key <Key ID>
```

- To delete a public-key

Delete a Key From a Key-ring

- To delete a private-key

```
gpg --delete-secret-key <Key ID>
```

- To delete a public-key

```
gpg --delete-key <Key ID>
```

Delete a Key From a Key-ring

- To delete a private-key

```
gpg --delete-secret-key <Key ID>
```

- To delete a public-key

```
gpg --delete-key <Key ID>
```

- Note: A private-key (if available) should be deleted before deleting a corresponding public-key.

Asymmetric Cipher - Encryption

```
gpg --output filename.gpg --encrypt
  --recipient <Key ID> <filename>
```

- To encrypt a document, a public-key of the recipient is required.

Asymmetric Cipher - Encryption

```
gpg --output filename.gpg --encrypt
  --recipient <Key ID> <filename>
```

- To encrypt a document, a public-key of the recipient is required.
- If `<filename>` is omitted, it encrypts the standard input.

Asymmetric Cipher - Encryption

```
gpg --output filename.gpg --encrypt
  --recipient <Key ID> <filename>
```

- To encrypt a document, a public-key of the recipient is required.
- If `<filename>` is omitted, it encrypts the standard input.
- For multiple recipients, use `--recipient` option for each recipient.

Asymmetric Cipher - Decryption

```
gpg --output <filename> --decrypt
<Encrypted-filename>
```

- To decrypt a document, use a private-key.

Asymmetric Cipher - Decryption

```
gpg --output <filename> --decrypt
    <Encrypted-filename>
```

- To decrypt a document, use a private-key.
- To access the private-key (if available in the key-ring), a pass-phrase is required.

Asymmetric Cipher - Generate a Signature

```
gpg --armor --output <ofilename.sig> --sign
                    <ifilename>
```

- Output - Default: A binary format. To produce the output in ASCII format, use `--armor`.

Asymmetric Cipher - Verify a Signature

```
gpg --verify <ifilename>
```

- Above command verifies the signature on a file.

Asymmetric Cipher - Verify a Signature

```
gpg --verify <ifilename>
```

- Above command verifies the signature on a file.

```
gpg --output <ofilename> --decrypt <ifilename>
```

Asymmetric Cipher - Verify a Signature

```
gpg --verify <ifilename>
```

- Above command verifies the signature on a file.

```
gpg --output <ofilename> --decrypt <ifilename>
```

- Above command verifies the signature and recovers the document.

Asymmetric Ciphers - A Detached Signature

```
gpg --output <ofilename> --detach-sig
                    <ifilename>
```

- The command produces a detached signature.

Asymmetric Ciphers - A Detached Signature

```
gpg --output <ofilename> --detach-sig
<ifilename> gpg --verify <signature-file>
<original-file>
```

- A signature and the original document are required to verify the signature.

Key Management

Key Management

- A key-pair comprises of a public-key and a private-key.

Key Management

- A key-pair comprises of a public-key and a private-key.
- A public-key

Key Management

- A key-pair comprises of a public-key and a private-key.
- A public-key
  - Public portion of the master singing-key (pub)

Key Management

- A key-pair comprises of a public-key and a private-key.
- A public-key
  - Public portion of the master singing-key (pub)
  - Public portion of the sub-ordinate signing and encryption keys (sub)

Key Management

- A key-pair comprises of a public-key and a private-key.
- A public-key
  - Public portion of the master singing-key (pub)
  - Public portion of the sub-ordinate signing and encryption keys (sub)
  - Set of user-IDs to associate the pubic-key with a real person.

Key Management

- A key-pair comprises of a public-key and a private-key.
- A public-key
  - Public portion of the master singing-key (pub)
  - Public portion of the sub-ordinate signing and encryption keys (sub)
  - Set of user-IDs to associate the pubic-key with a real person.
- The structure of private key is nearly identical.

Key Management

- Each key comprises of

Key Management

- Each key comprises of
  - Key-ID

Key Management

- Each key comprises of
  - Key-ID
  - Key

Key Management

- Each key comprises of
  - Key-ID
  - Key
  - Expiry date, etc.

Key Management

- Each key comprises of
  - Key-ID
  - Key
  - Expiry date, etc.
- Each user-ID comprises of

Key Management

- Each key comprises of
  - Key-ID
  - Key
  - Expiry date, etc.
- Each user-ID comprises of
  - User-name

Key Management

- Each key comprises of
  - Key-ID
  - Key
  - Expiry date, etc.
- Each user-ID comprises of
  - User-name
  - Comment (optional)

Key Management

- Each key comprises of
  - Key-ID
  - Key
  - Expiry date, etc.
- Each user-ID comprises of
  - User-name
  - Comment (optional)
  - Email address (optional)

Key Management

```
gpg --edit-key <Key-ID or User ID>
```

- The command is used to view a key-pair.

Key Management

```
gpg --edit-key <Key-ID or User ID>
```

- The command is used to view a key-pair.
- To switch between a public and a private key (if available), use “toggle”.

Key Management

```
gpg --edit-key <Key-ID or User ID>
```

- The command is used to view a key-pair.
- To switch between a public and a private key (if available), use "toggle".
- To exit from the edit-key menu, use `quit` command and save the changes.

Key Management – Edit-key - Output

1. Type of the key
    - pub - Public master singing-key, sub - Public sub-ordinate keys, sec - Private master singing-key, sbb - Private sub-ordinate keys
2. Key's bit-length, type, and ID
    - Type - `R` - RSA, `D` - DSA, `G` - Elgamal
    - Capital letters (R, D, G) - to represent signing and encryption keys
    - Small letters (r, d, g) - to represent encryption keys
3. Date of creation and expiration
4. User-IDs

The public-key can be modified by adding/deleting keys, user IDs, etc. What is its impact on the security?

What is “a man in the middle attack”?

How do you protect against a man in the middle attack?

How to prevent an adversary from modifying and publishing your public-key?

How to prevent an adversary from modifying and publishing your public-key? Sign the key(s).

Signing the Keys

- Signing our public-key enables us to detect the key tampering.

Signing the Keys

- Signing our public-key enables us to detect the key tampering.
- Signing other people's public-key enables us to certify the key and associate the key with the owner of the key (User ID).

Signing the Keys

- Signing our public-key enables us to detect the key tampering.
- Signing other people's public-key enables us to certify the key and associate the key with the owner of the key (User ID).
- Key signatures are used in "web of trust" (discussed later).

Key Integrity

- To protect the public-keys, the GNU PG uses self-signatures.

Key Integrity

- To protect the public-keys, the GNU PG uses self-signatures.
- To view the self signature, use the following command.

Key Integrity

- To protect the public-keys, the GNU PG uses self-signatures.
- To view the self signature, use the following command.

```
gpg --check-sigs
```

Key Components - Add and Delete

- To add a new sub-key, use `addkey` command.

Note: To delete a key or user ID, first select it using the `key <number>` or `uid <number>` commands. To de-select the key, use the same command.

Key Components - Add and Delete

- To add a new sub-key, use `addkey` command.
- To add a new user ID, use `adduid` command.

Note: To delete a key or user ID, first select it using the `key <number>` or `uid <number>` commands. To de-select the key, use the same command.

Key Components - Add and Delete

- To add a new sub-key, use `addkey` command.
- To add a new user ID, use `adduid` command.
- To delete a sub-key, use `delkey` command.

Note: To delete a key or user ID, first select it using the `key <number>` or `uid <number>` commands. To de-select the key, use the same command.

Key Components - Add and Delete

- To add a new sub-key, use `addkey` command.
- To add a new user ID, use `adduid` command.
- To delete a sub-key, use `delkey` command.
- To delete a user ID, use `deluid` command.

Note: To delete a key or user ID, first select it using the `key <number>` or `uid <number>` commands. To de-select the key, use the same command.

Revoke a Key

- To revoke a key

Revoke a Key

- To revoke a key
  - Select the key using `key <number>` command.

Revoke a Key

- To revoke a key
  - Select the key using `key <number>` command.
  - Use `revkey` command.

Revoke a Key

- To revoke a key
  - Select the key using `key <number>` command.
  - Use `revkey` command.
- The revocation does not delete the key. It only adds revocation self-signature to the key.

Update a Key's Expiration Time

- To update a key's expiration time

Update a Key's Expiration Time

- To update a key's expiration time
  - Select the key using `key <number>` command.

Update a Key's Expiration Time

- To update a key's expiration time
  - Select the key using `key <number>` command.
  - Use `expire` command.

Web of Trust

How do you verify the public-keys of others?

A Public-key Verification

- Contact the owner of the public-key, and verify the `fingerprint`.

A Public-key Verification

- Contact the owner of the public-key, and verify the `fingerprint`.
- If the public-key is valid, then sign it using the private-key (to detect the key tampering in the future).

How do you verify a public-key of the person whom you do not know personally?

How do you verify a large number of public-keys when communicating with a large number of people?

Web of Trust

- Instead of verifying all the public-keys, web of trust delegates the task to the people whom the user trust.

Web of Trust

- Instead of verifying all the public-keys, web of trust delegates the task to the people whom the user trust.
- If the people, a user trusts, validate the keys properly, a user accepts the public-keys signed by them as valid keys (rather than verifying them personally).

Trust in a Key's Owner

- Trust is subjective.

Trust in a Key's Owner

- Trust is subjective.
- Each public-key includes the trust a user has in a key's owner.

Trust in a Key's Owner

- Trust is subjective.
- Each public-key includes the trust a user has in a key's owner.
    - unknown or q - No information about the owner's judgment in key signing.

Trust in a Key's Owner

- Trust is subjective.
- Each public-key includes the trust a user has in a key's owner.
  - unknown or q - No information about the owner's judgment in key signing.
  - none or n - Not trusted

Trust in a Key's Owner

- Trust is subjective.
- Each public-key includes the trust a user has in a key's owner.
  - unknown or q - No information about the owner's judgment in key signing.
  - none or n - Not trusted
  - marginal or m - Trusted

Trust in a Key's Owner

- Trust is subjective.
- Each public-key includes the trust a user has in a key's owner.
  - unknown or q - No information about the owner's judgment in key signing.
  - none or n - Not trusted
  - marginal or m - Trusted
  - full or f - Fully trusted

Trust in a Key's Owner

- Trust is subjective.
- Each public-key includes the trust a user has in a key's owner.
  - unknown or q - No information about the owner's judgment in key signing.
  - none or n - Not trusted
  - marginal or m - Trusted
  - full or f - Fully trusted
- Trust levels of the public-keys remain private to the user.

Trust in a Key's Owner

- Trust is subjective.
- Each public-key includes the trust a user has in a key's owner.
  - unknown or q - No information about the owner's judgment in key signing.
  - none or n - Not trusted
  - marginal or m - Trusted
  - full or f - Fully trusted
- Trust levels of the public-keys remain private to the user.
- To adjust the trust in a key's owner, use trust command.

Trust in a Key's Owner

- A key is considered valid if it meets two conditions.

Trust in a Key's Owner

- A key is considered valid if it meets two conditions.
    1. It has been signed by enough valid keys.

Trust in a Key's Owner

- A key is considered valid if it meets two conditions.
    1. It has been signed by enough valid keys.
        - A user has signed it personally.

Trust in a Key's Owner

- A key is considered valid if it meets two conditions.
    1. It has been signed by enough valid keys.
        - A user has signed it personally.
        - It has been signed by one fully trusted key.

Trust in a Key's Owner

- A key is considered valid if it meets two conditions.
  1. It has been signed by enough valid keys.
     - A user has signed it personally.
     - It has been signed by one fully trusted key.
     - It has been signed by three marginally trusted keys.

Trust in a Key's Owner

- A key is considered valid if it meets two conditions.
  1. It has been signed by enough valid keys.
     - A user has signed it personally.
     - It has been signed by one fully trusted key.
     - It has been signed by three marginally trusted keys.
  2. The path of signed keys, leading from the key back to the user's own key, is 5 steps or shorter.

Trust in a Key's Owner

- A key is considered valid if it meets two conditions.
  1. It has been signed by enough valid keys.
     - A user has signed it personally.
     - It has been signed by one fully trusted key.
     - It has been signed by three marginally trusted keys.
  2. The path of signed keys, leading from the key back to the user's own key, is 5 steps or shorter.
- Note: The above numbers are "defaults" and they can be changed.

How to distribute a public-key?

Distribute Keys to the Key Server

- Personal webpage, email, etc.

Distribute Keys to the Key Server

- Personal webpage, email, etc.
- Public-key servers

Distribute Keys to the Key Server

- Personal webpage, email, etc.
- Public-key servers
- After signing the public-keys of others, people often sent them to the key-servers. The signed public-keys are then used in the web of trust.

Distribute Keys to the Key Server

```
gpg --keyserver Certserver.pgp.com --recv-key
                        <Key ID>
```

- The command retrieves the latest version of the requested key from the key server.

Distributing Keys to the Key Server

```
gpg --keyserver Certserver.pgp.com --send-key
                        <Key ID>
```

- The command sends the newly generated/signed key to the key server (to update the server's version of the key).

Mistakes to be Avoided...

- Forgetting passphrase

Mistakes to be Avoided...

- Forgetting passphrase
- Accidentally deleting the private key

Mistakes to be Avoided...

- Forgetting passphrase
- Accidentally deleting the private key
- Accidentally publishing the private key

Mistakes to be Avoided...

- Forgetting passphrase
- Accidentally deleting the private key
- Accidentally publishing the private key
- Accidentally revoking a key

Mistakes to be Avoided...

- Forgetting passphrase
- Accidentally deleting the private key
- Accidentally publishing the private key
- Accidentally revoking a key
- No backup of the key rings

References

References

- The GnuPG Project, URL: https://www.gnupg.org/index.html
- The GNU Privacy Handbook, URL: https://www.gnupg.org/gph/en/manual.pdf
- Using the GNU Privacy Guard, URL: https://www.gnupg.org/documentation/manuals/gnupg.pdf
- Philip R. Zimmermann - Homepage, URL: https://philzimmermann.com/EN/background/index.html

Thank You.