Network Security - Encryption

CME451 Tutorial 5

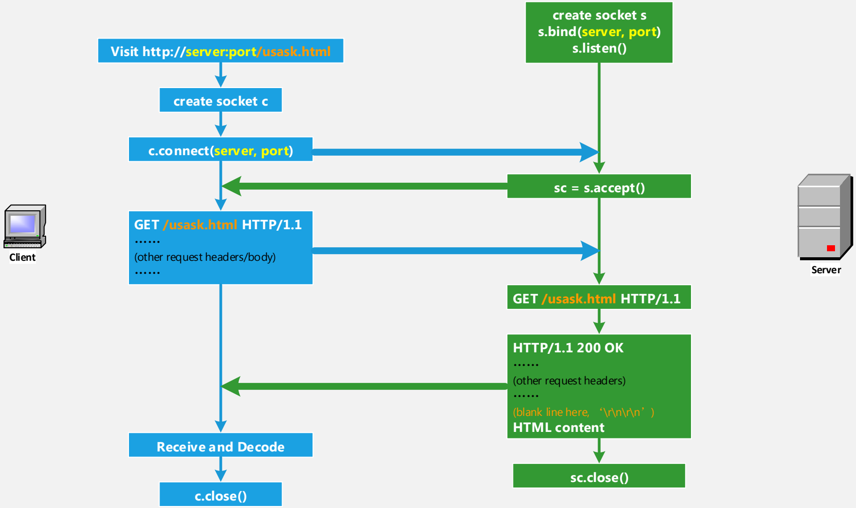
Hao Zhang
(Graduate Teaching Fellow)

Department of Electrical & Computer Engineering
University of Saskatchewan

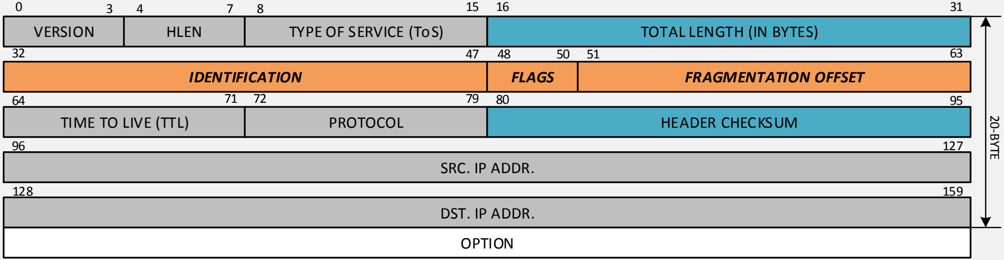
Feb 3, 2017

*Most contents are from William Stallings, *Data and Computer Communications*, 8th edition, 2007 Pearson Education Inc.

Review-Socket Web Server and Client

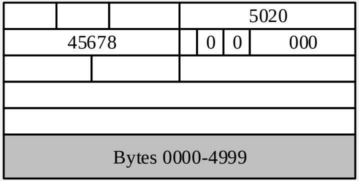


Review-IP Header and Fragmentation



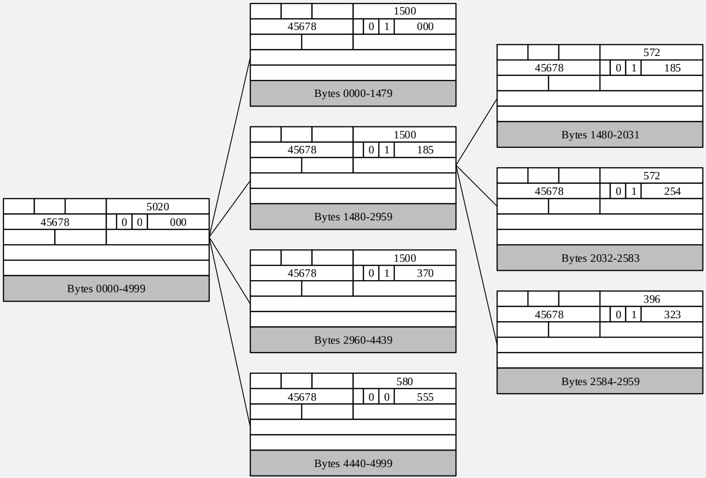
- Related component: `IDENTIFICATION`, `FLAGS`, `FRAGMENTATION OFFSET`.
- Must change: `FLAGS`, `FRAGMENTATION OFFSET`, `TOTAL LENGTH`, `HEADER CHECKSUM`.

IP Fragmentation Example



1. We want to pass this datagram through an Ethernet network. Divide the datagram into necessary fragments and change the values of the IP header fields.
2. The second fragmented datagram from previous part needs to be sent through an X.25 network. Repeat the fragmentation task.
3. Design a defragment algorithm in order to generate the complete original datagram.

IP Fragmentation Example



IP Fragmentation Example

Algorithm 1 Pseudocode for IP fragmentation

```
mtu_d = mtu − 20, frag_num = 0
while data_length > 0 do
    frag_offset = mut_d × frag_num ÷ 8
    data_length = data_length − mtu_d
    if data_length > 0 then
        flag = 001
        frag_total_length = mtu_d + header_length × 4
        frag_data_length = mtu_d
    else
        flag = 000
        frag_total_length = mtu_d + data_length + header_length × 4
        frag_data_length = mtu_d + data_length
    end if
    frag_num = frag_num + 1
end while
```

IP Fragmentation Example

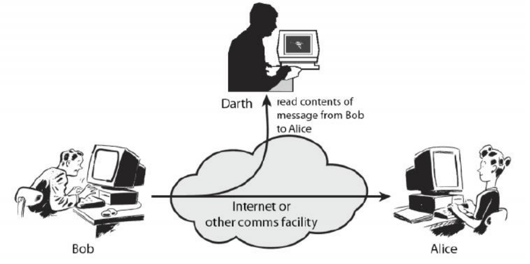
Algorithm 2 Pseudocode for IP defragmentation

```
frag_list.append()
frag_id = frag_list[0]['ID']
for frag in frag_list do
    if frag[ID]! = frag_id then
        del frag
    end if
end for
frag_list.sort(key = offset)
defrag_pkt[total_length] = 20, defrag_pkt[data_length] = 0
for frag in frag_list do
    defrag_pkt[total_length]+ = frag[data_length]
end for
defrag_pkt[data_length] = defrag_pkt[total_length] − 20
```

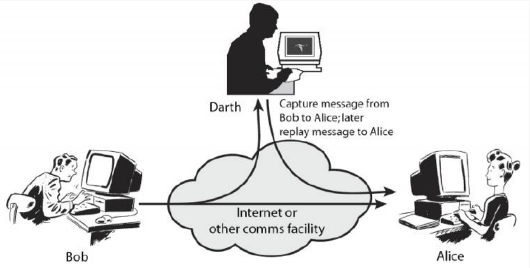
- Network security is important:
  - Data are transmitted through networks.
  - Protect the privacy.
  - Avoid data to be stolen or copied.
  - Protect the data from being modified.
  - Verify we receive the data from the correct source.

- Network security requirements:
    - **Confidentiality:** the data can only be accessed by the authorized parties. This type of access includes printing, displaying, and other forms of disclosure, including simple revealing the existence of an object.
    - **Integrity:** only the authorized parties can modify data. Modification includes writing, changing, changing status, deleting, and creating.
    - **Availability:** data are available to authorized parties.
    - **Authenticity:** a host or a service should be able to verify the identity of users.

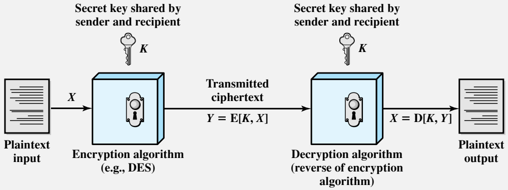
- Passive attacks:
  - Eavesdropping on or monitoring of transmission.
  - Release of message contents: phone conversion, email content, file content, ...
  - Traffic analysis: guessing the nature of the communication, ...
  - Hard to detect: no alteration of data.
  - Solution: encryption.



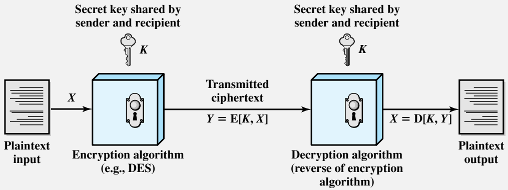
- Active attacks:
  - Modify data stream or create a false stream.
  - Masquerade: one entity pretends to be a different entity.
  - Replay: capture and retransmit of a data.
  - Modification of Message: data is altered, delayed or reordered.
  - Denial of service: prevents or inhibits the normal use or management.
  - Solution: authentication.



- Five ingredients:
  - **Plaintext:** the original message or data.
  - **Encryption algorithm:** the algorithms perform substitutions or transmissions on the plaintext.
  - **Secret key:** the encryption depends on the secret keys.
  - **Ciphertext:** the scrambled/encrypted message.
  - **Decryption algorithm:** algorithm to decrypt the scrambled message.



- Two requirements:
  - A strong encryption algorithm: an opponent who knows the algorithm and has access to several ciphertexts would be unable to decipher the text or figure out the key.
  - Sender and receiver must obtain the secret key in a secure fashion and keep the key secure.



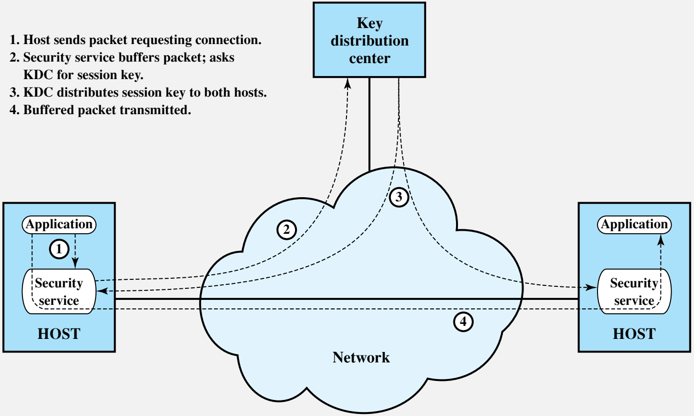
- **DES:** Data Encryption Standard
  - DES uses 56-bit key.
  - 3DES: repeat DES three times, using two or three unique keys for a key size of 112-bit or 168-bit.
  - Use only 64-bit block size: a larger block size is desirable.
- **AES:** Advanced Encryption Standard
  - Block length of 128-bit.
  - Key length of 128-bit, 192-bit, or 256-bit.
  - Improved efficiency and security strength.

- Deliver a key to two parties (A and B) without allowing others to see the key.
  1. A key could be selected by A and physically delivered to B.
  2. A third party could select the key and physically delivered to A and B.
  3. If A and B have previously used a key, one party could transmit the new key to the other, encrypted using the old key.
  4. If A and B each have an encrypted connection to a third party C, C could deliver a key on the encrypted links to A and B.

Network Security - Encryption

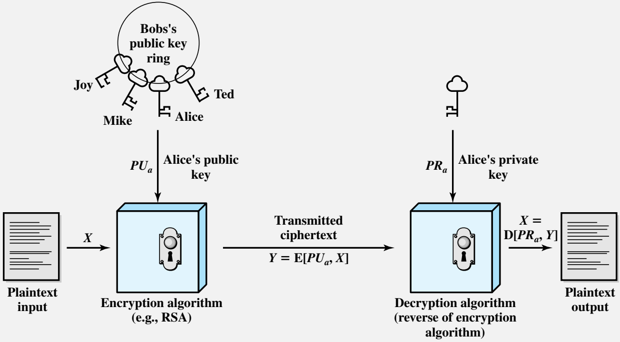
Symmetric Encryption - Key Distribution

- Detailed process of option 4:



- Asymmetric encryption, also called public-key encryption, involving two separate keys.
  - **Plaintext:** the original message or data.
  - **Encryption algorithm:** the algorithms perform substitutions or transmissions on the plaintext.
  - **Public and private key:** a pair of keys, one for encryption and one for decryption.
  - **Ciphertext:** the scrambled/encrypted message.
  - **Decryption algorithm:** algorithm to decrypt the scrambled message.

- Asymmetric encryption algorithm:



- A pair of keys: public key and private key
  - Public key is made for others to use, all participants have access to it.
  - Private key is known only to its owner, need never be distributed.
  - One for encryption and the other for decryption.
- Infeasible to determine the decryption key given the knowledge of cryptographic algorithm and the encryption key.
- `A` want to send message to `B`:
  - `A` encrypt the message using `B`'s public key.
  - Only `B` can decrypt using own private key.

- RSA
  - Developed in 1977 by Ron Rivest, Adi Shamir, and Len Adleman.
  - For plaintext block M and ciphertext block C:

$$\begin{aligned} C &= M^e \ mod \ n \\ M &= C^d \ mod \ n = (M^e)^d \ mod \ n = M^{ed} \ mod \ n \end{aligned} \tag{1}$$

  - Both sender and receiver must know n and e, and only the receiver knows d. Public key $PU = \{e, n\}$ and private key $PR = \{d, n\}$.
  - Infeasible to determine d given e and n for large values of e and n.

- Asymmetric encryption can help the key management of symmetric encryption.
  - A encrypts a message using symmetric encryption with symmetric key k_s.
  - A encrypts the k_s using public-key encryption with B's public key.
  - Attached the encrypted k_s to the message and send to B.
  - Then only B is able to decrypt the k_s and thus to recover the original message.

Summary

- Network Encryption:
  - Symmetric Encryption
  - Asymmetric encryption/Public-Key encryption
- Encryption protects against passive attack.
- To protect against active attack: message authentication.
- `Pycrypto` can be used to implemented encryption and authentication.