Introduction to Public Key Infrastructure

Developed and Created by: Alex El-Jack

Cryptography

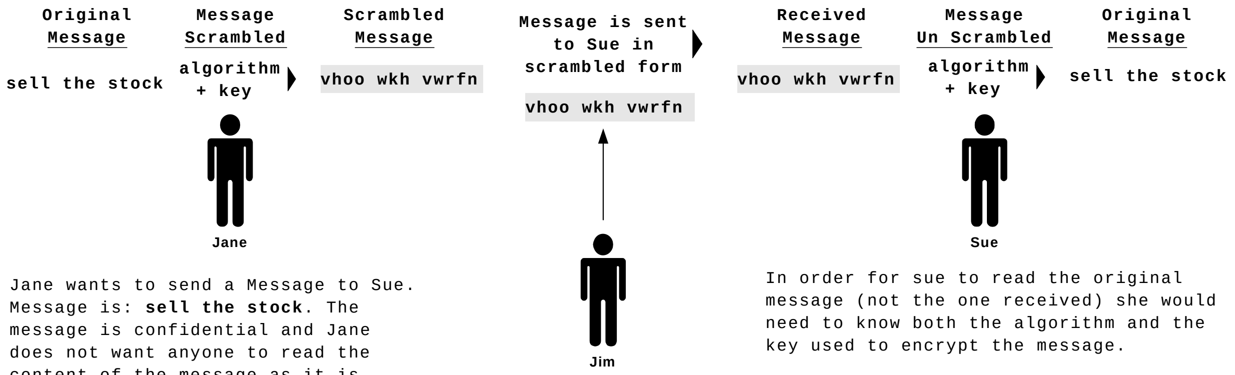
Secure communication between two entities is provided by implementing Encryption, Authentication or both.

- Encryption provides Data confidentiality
- Authentication provides Identity, Integrity and anti-replay

Encryption

Terms associated with encryption:

- **Plain Text** – The information in it's original form. This is also known as clear text.
- **Cipher Text** – The information after it has been scrambled by the encryption algorithm.
- **Algorithm** – The method of manipulation that is used to change the plain text into cipher text.
- **Session Key** – The data used with the algorithm that transforms the plain text into the cipher text or the cipher text into plain text.
- **Encryption** – The process of changing plain text into cipher text.
- **Decryption** – The process of changing cipher text into plain text.



Jane wants to send a Message to Sue. Message is: **sell the stock**. The message is confidential and Jane does not want anyone to read the content of the message as it is being sent.

Jane will need to scramble the message before she sends the message. In order to do that Jane needs an scrambling method and a key used to scramble.

Jane decides to use the method:
- Add ? characters to each letter of the alphabet.

Jane decides to use the key:
- ?=3

Jane performs the scrambling:

```
s e l l  t h e  s t o c k
| | | |  | | |  | | | | |
v h o o  w k h  v w r f n
```

If Phil intercepts the message, he will be able to read the message.
Message is: **vhoo wkh vwrfn**
Note, even though Phil is able to read the message, he is reading the scrabbled version of the message.

In order for sue to read the original message (not the one received) she would need to know both the algorithm and the key used to encrypt the message.

Jane would have had to some how inform Sue of the algorithm and the key. Jane informs Sue that the algorithm is:
- Add ? characters to each letter of the alphabet.
The key is:
- ?=3

Sue received the message:
sell the stock.
Sue now need to unscramble (decrypt) the message before she is able to identify the original message.

Sue will need to use the reverse process to decrypt the message received.
- Subtract ? characters to each letter of the alphabet.
Sue must use the key:
- ?=3
Sue performs the descrambling:

```
v h o o  w k h  v w r f n
| | | |  | | |  | | | | |
s e l l  t h e  s t o c k
```

Examples of Encryption Algorithms include – DES, 3DES and AES.

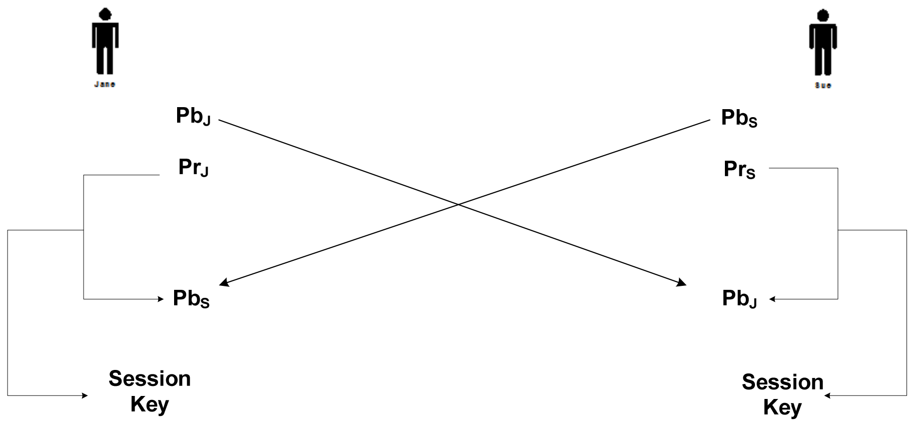
The problem with this is exchanging the keys securely. There are two ways to exchange keys:

- Manual – using an Out Of Band (OOB) method to exchange the keys.
- IKE (Internet key Exchange) method

Manual

Call the person on the telephone of send an e-mail.

IKE (Internet key Exchange)



Example of a protocol used to exchange keys is Diffie–Hellman key exchange (D–H)

Proof of the Diffie–Hellman key exchange (D–H) protocol works.

[A prime number (or a prime) is a natural number that has exactly two distinct natural number divisors: 1 and itself. The smallest twenty-five prime numbers (all the prime numbers under 100) are: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97]

p = Prime number greater than 2 (A prime number is number divisible by 1 and itself).
g = Integer smaller than p.

x_n = a random value must be less than p-1 [generated Secret key]

y_n = Public Key

y_n = g^x_n mod p [to generate the Public Key]

k = shared key / session key - used to encrypt the message.
k = y_n ^x_n mod p [to generate the shared key / Session key]

X_a = Alice's Private Key

y_a = Alice's Public Key

X_b = Bob's Private Key

y_b = Bob's Public Key

g ^ x_n - means: g is raised to the x_n power.

y_n % x_n - means: y_n is divided by y_n and the remainder is returned.

Bob and Alice agree on the values of p and g - p = 7 g = 5

<u>Alice</u>

X_a = 3 (Must be less than p-1)

y_a = g^x_a mod p
y_a = 5^3 mod 7
y_a = 125 mod 7
y_a = 6

y_b = 2

k = y_b ^x_a mod p
k = 2 ^3 mod 7
k = 8 mod 7
k = 1

<u>Bob</u>

X_b = 4 (Must be less than p-1)

y_b = g^x_b mod p
y_b = 5^4 mod 7
y_b = 625 mod 7
y_b = 2

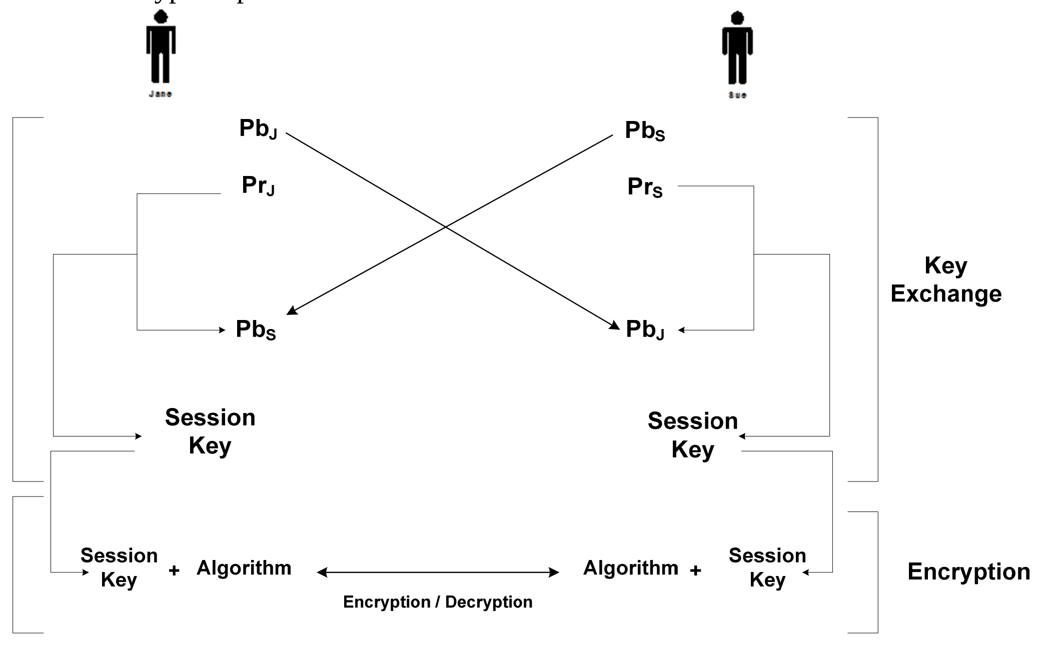
y_a = 6

k = y_a ^x_b mod p
k = 6 ^4 mod 7
k = 1296 mod 7
k = 1

---

y_a = 125 mod 7	(125 divided by 7 = 17 and reminder 6)
y_b = 625 mod 7	(625 divided by 7 = 89 and remainder 2)
k = 8 mod 7	(8 divided by 7 = 1 and remainder 1)
k = 1296 mod 7	(1296 divided by 7 = 185 and remainder 1)

http://calculator.sdsu.edu/calculator.php
Note: Keep in mind that the key exchange process is only used to exchange / generate matching session keys which can then be used in the encryption process.

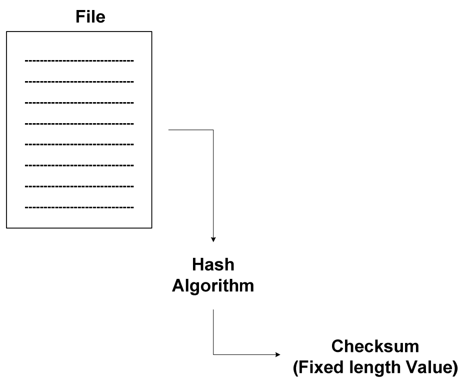


Protocols

Encryption only protocol(s): DES, 3DES and AES
Key Exchange only protocol(s): D-H
Protocol(s) that support both encryption and Key Exchange: RSA

Authentication

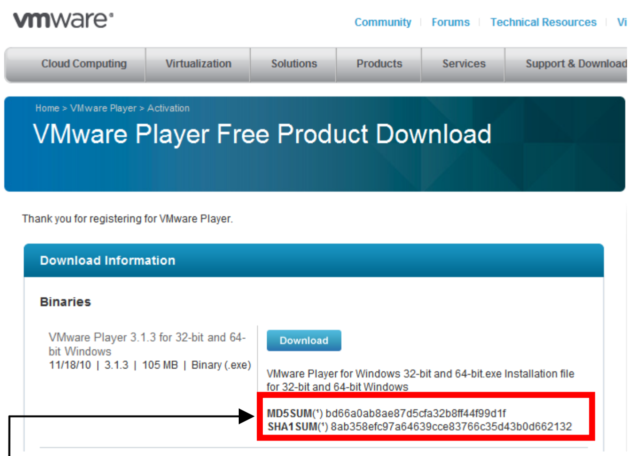
In computing environments authentication is used to validate the identity and authenticity of an entity.
An example of the use of the hashing process is to validate the content of a file.



Running the same algorithm against the same file will always produce the same checksum; any changes to the content of the file will change the checksum.

Examples of hashing algorithms: MD5 (Message-Digest algorithm 5) and SHA
- MD5 (Message-Digest algorithm 5): produces a 128-bit (16-byte) checksum / hash / message digest.
- SHA-1: produces a 160-bit checksum / hash / message digest.

Example: Download the VMware player from http://downloads.vmware.com/d/ (this requires a login) you can create an account to get to this file.



VMware Player Free Product Download

Thank you for registering for VMware Player.

Download Information

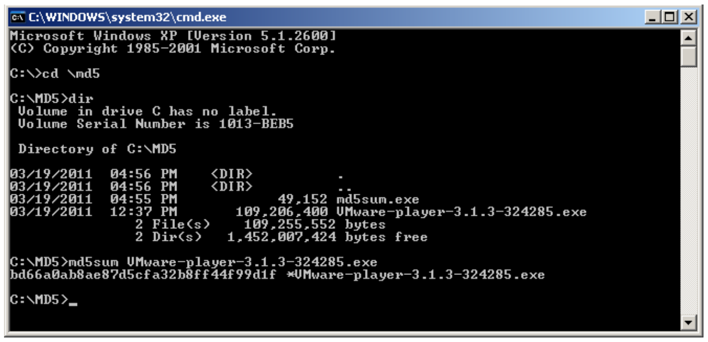
Binaries

VMware Player 3.1.3 for 32-bit and 64-bit Windows
11/18/10 | 3.1.3 | 105 MB | Binary (.exe)

VMware Player for Windows 32-bit and 64-bit.exe Installation file for 32-bit and 64-bit Windows

MD5SUM(*) bd66a0ab8ae87d5cfa32b8ff44f99d1f
SHA1SUM(*) 8ab358efc97a64639cce83766c35d43b0d662132

After downloading the file compare the checksums



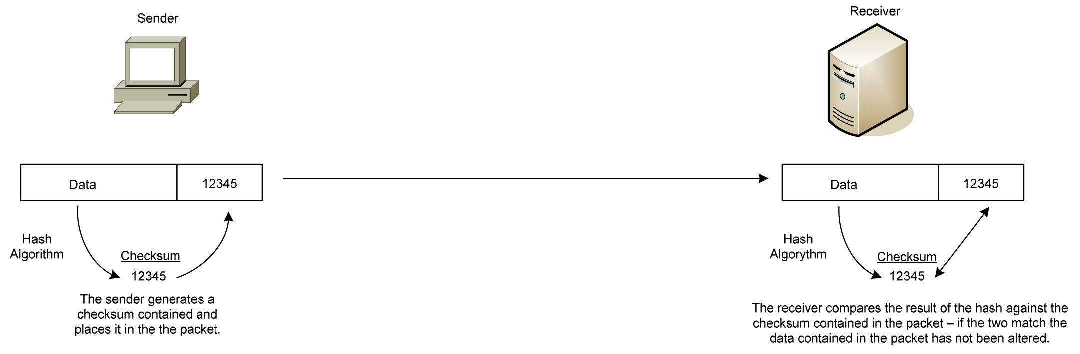
```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\>cd \md5

C:\MD5>dir
 Volume in drive C has no label.
 Volume Serial Number is 1013-BEB5

 Directory of C:\MD5

03/19/2011  04:56 PM    <DIR>          .
03/19/2011  04:56 PM    <DIR>          ..
03/19/2011  04:55 PM            49,152 md5sum.exe
03/19/2011  12:37 PM       109,206,400 VMware-player-3.1.3-324285.exe
               2 File(s)    109,255,552 bytes
               2 Dir(s)   1,452,007,424 bytes free

C:\MD5>md5sum VMware-player-3.1.3-324285.exe
bd66a0ab8ae87d5cfa32b8ff44f99d1f *VMware-player-3.1.3-324285.exe

C:\MD5>_
```

Note: The object of the hashing algorithm is to verify or authenticate the content or identity of an entity (not encrypt).

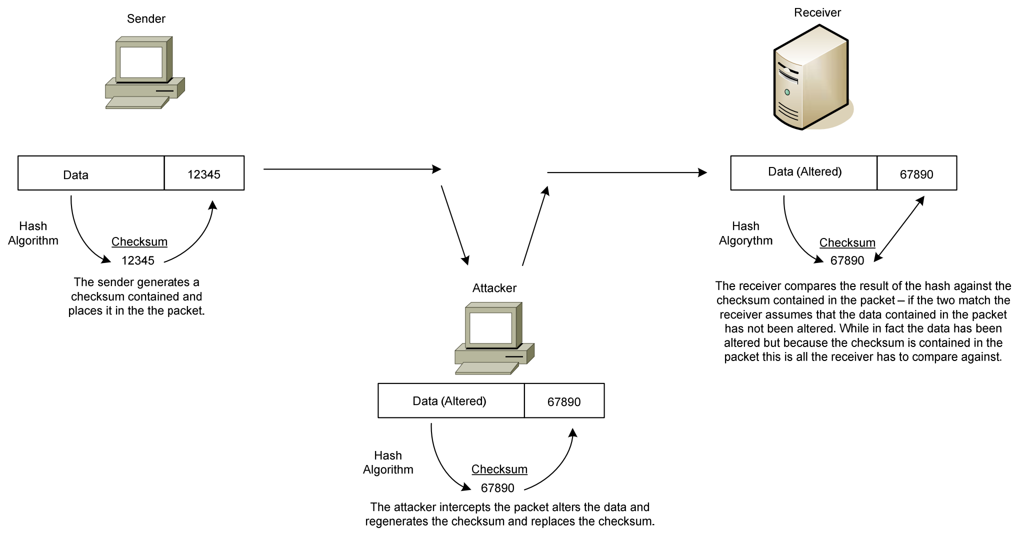
The process of verifying integrity shown above is practical only if the content can be verified manually / visually. In some cases the integrity of data being transferred between two entities is required. In such a case the method shown above would be ineffective. A variation of the MD5 and SHA-1 protocols were developed to address this situation. This variation uses the HMAC (Hash-based Message Authentication Code) which is a specific construct for calculating the checksum in combination with a secret key.

This example will demonstrate the use of the HMAC variation of the MD5 and SHA-1 hashing algorithms, referred to as the MD5-HMAC and SHA-1-HMAC respectively.

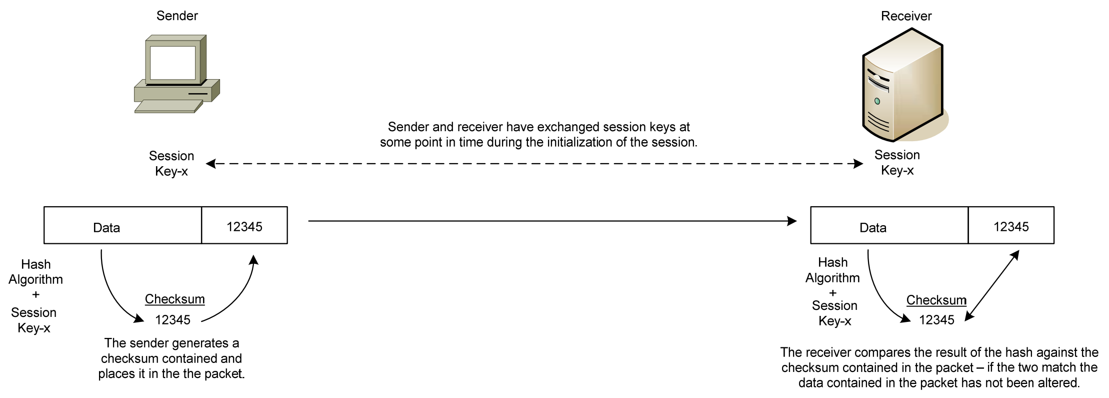
Looking at the process without the HMAC variation the example demonstrates two entities exchanging information over the wire and using hashes to assure that the integrity of the data (Data has not been altered in transit).


Sender
Receiver
Data
12345
Data
12345
Hash Algorithm
Checksum
12345
The sender generates a checksum contained and places it in the the packet.
Hash Algorythm
Checksum
12345
The receiver compares the result of the hash against the checksum contained in the packet – if the two match the data contained in the packet has not been altered.

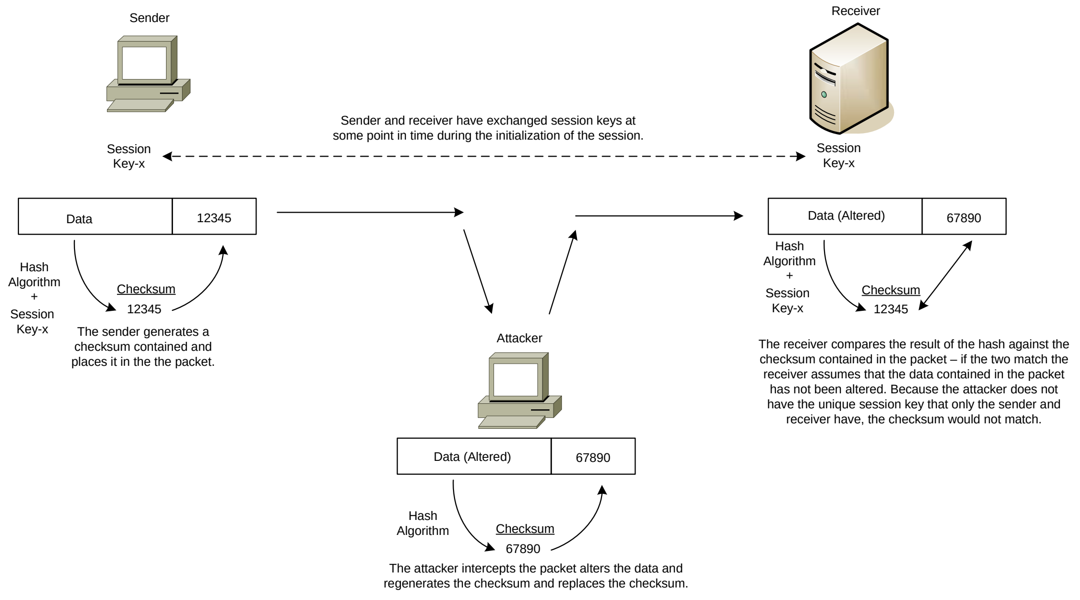
In the next example an intruder is introduced into the picture intercepting the data packet and altering the data and the hash.


Sender
Receiver
Data
12345
Data (Altered)
67890
Hash Algorithm
Checksum
12345
The sender generates a checksum contained and places it in the the packet.
Hash Algorythm
Checksum
67890
The receiver compares the result of the hash against the checksum contained in the packet – if the two match the receiver assumes that the data contained in the packet has not been altered. While in fact the data has been altered but because the checksum is contained in the packet this is all the receiver has to compare against.
Attacker
Data (Altered)
67890
Hash Algorithm
Checksum
67890
The attacker intercepts the packet alters the data and regenerates the checksum and replaces the checksum.

To solve this issue the next example demonstrates the HMAC variation of the hashing algorithms.


Sender
Receiver
Sender and receiver have exchanged session keys at some point in time during the initialization of the session.
Session Key-x
Session Key-x
Data
12345
Data
12345
Hash Algorithm + Session Key-x
Checksum
12345
The sender generates a checksum contained and places it in the the packet.
Hash Algorithm + Session Key-x
Checksum
12345
The receiver compares the result of the hash against the checksum contained in the packet – if the two match the data contained in the packet has not been altered.

In the next example an intruder is introduced into the picture intercepting the data packet and altering the data and the hash. Because the original hash that was generated by the sender is unique as it is derived using the session key (since the HMAC variation is used here) and only the sender and receiver have the session key, the attacker is unable to produce a valid checksum that can be verified by the receiver.

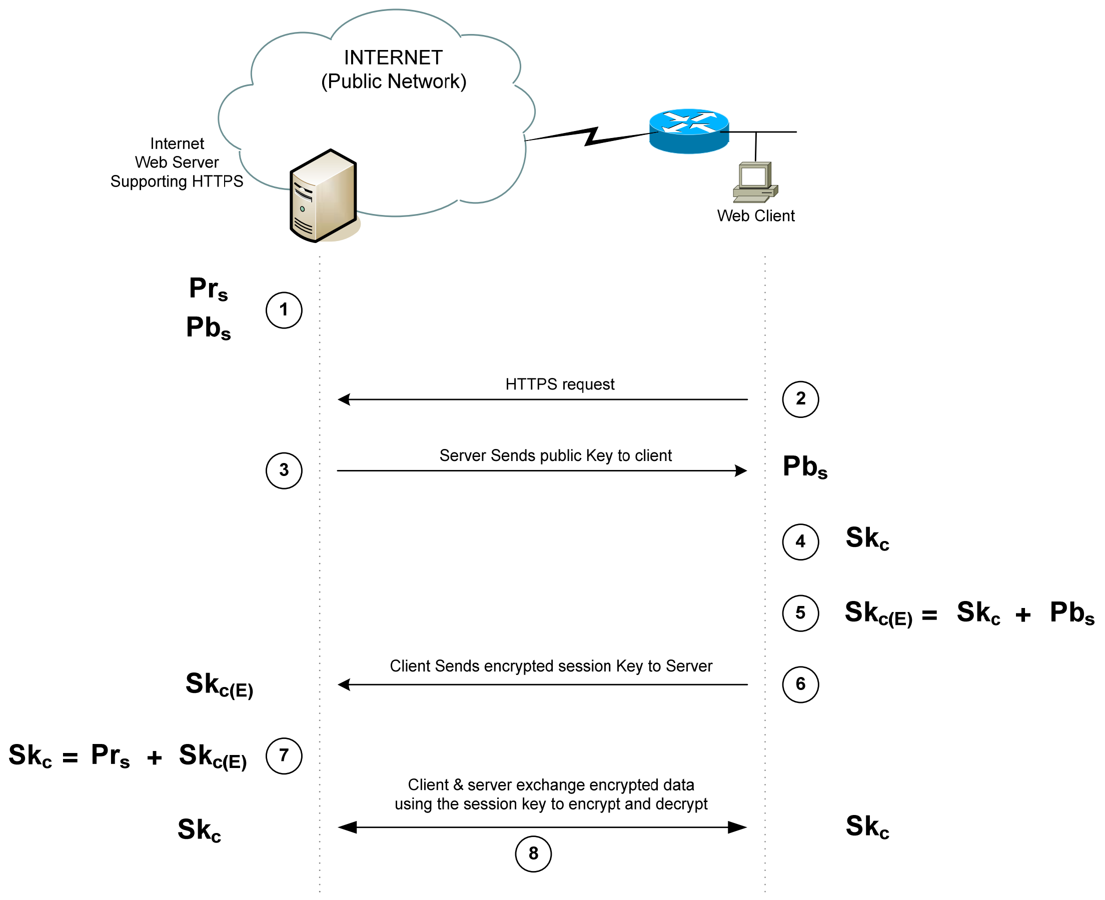

Sender
Receiver
Sender and receiver have exchanged session keys at some point in time during the initialization of the session.
Session Key-x
Session Key-x
Data
12345
Data (Altered)
67890
Hash Algorithm + Session Key-x
Checksum
12345
The sender generates a checksum contained and places it in the the packet.
Hash Algorithm + Session Key-x
Checksum
12345
The receiver compares the result of the hash against the checksum contained in the packet – if the two match the receiver assumes that the data contained in the packet has not been altered. Because the attacker does not have the unique session key that only the sender and receiver have, the checksum would not match.
Attacker
Data (Altered)
67890
Hash Algorithm
Checksum
67890
The attacker intercepts the packet alters the data and regenerates the checksum and replaces the checksum.

Because the TCP/IP version 4 does not have any built in security mechanisms such as the ability to encrypt the information being exchanged between devices or the ability to authenticate who is communicating, the protocol is vulnerable to misuse.

Some protocols which use TCP/IP as the transport protocol have security integrated into the protocol.

An example of such a protocol is SSH or the secure shell protocol which is a replacement for the Telnet protocol which is not secure and is vulnerable to attack. (See the Telnet hijack document)

Another example would be the HTTPS protocol which is a secure alternative to HTTP. HTTPS has built in security mechanisms using the SSL/TLS protocol.

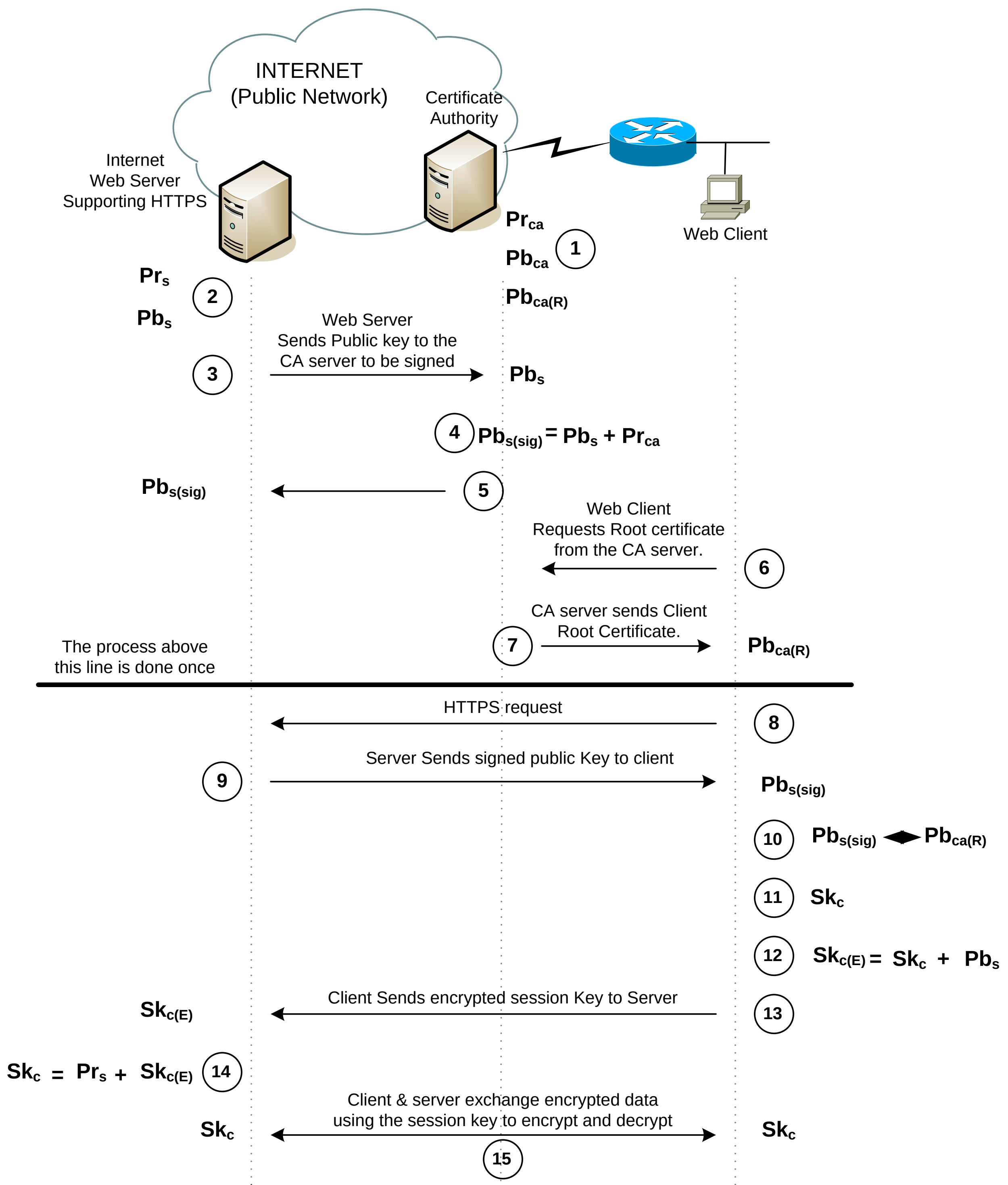

INTERNET
(Public Network)
Internet
Web Server
Supporting HTTPS
Web Client
Pr_s
Pb_s
1
HTTPS request
2
Server Sends public Key to client
3
Pb_s
4
Sk_c
5
$Sk_{c(E)} = Sk_c + Pb_s$
Client Sends encrypted session Key to Server
$Sk_{c(E)}$
6
$Sk_c = Pr_s + Sk_{c(E)}$
7
Client & server exchange encrypted data
using the session key to encrypt and decrypt
Sk_c
8
Sk_c

The HTTPS protocol described above uses the RSA protocol to perform the Key Exchange (exchange of the session key) and Encryption.

The SSH ver 1 protocol uses a similar process as the HTTPS protocol shown above.

One of the issues related to HTTPS, in the example above, is the fact that the client cannot assure that they are communicating with the correct server.

Certificates can be used to authenticate / validate the identity of the server that the client is connecting to.


INTERNET
(Public Network)
Certificate Authority
Internet Web Server Supporting HTTPS
Web Client
Pr_{ca}
Pb_{ca}
1
$Pb_{ca(R)}$
Pr_s
2
Pb_s
Web Server Sends Public key to the CA server to be signed
3
Pb_s
4
$Pb_{s(sig)} = Pb_s + Pr_{ca}$
$Pb_{s(sig)}$
5
Web Client Requests Root certificate from the CA server.
6
CA server sends Client Root Certificate.
7
$Pb_{ca(R)}$
The process above this line is done once
HTTPS request
8
Server Sends signed public Key to client
9
$Pb_{s(sig)}$
10
$Pb_{s(sig)} \longleftrightarrow Pb_{ca(R)}$
11
Sk_c
12
$Sk_{c(E)} = Sk_c + Pb_s$
Client Sends encrypted session Key to Server
$Sk_{c(E)}$
13
$Sk_c = Pr_s + Sk_{c(E)}$
14
Client & server exchange encrypted data using the session key to encrypt and decrypt
Sk_c
Sk_c
15