Encryption, Data Integrity, Digital Certificates, and SSL

Developed by

Jerry Scott

2002

Ideas Behind Encryption

- **When information is transmitted across intranets or the Internet, others can eavesdrop on the transmission**
- **Encryption is intended to prevent eavesdroppers from understanding the transmitted information – they may still read it, but not understand it**
  - To maintain the confidentiality of communications
  - Also helps to maintain the integrity of the data

- **Two main types of encryption exist**
  - Symmetric encryption (sometimes called secret key or 1-key)
  - Asymmetric encryption (sometimes called public key or 2-key)
- **Symmetric encryption is 1,000 times faster than asymmetric in software**

Symmetric Key Encryption

- **With symmetric encryption, the same key is used for encryption and decryption of the data – hence the simple name "1-key encryption"**
- **Symmetric key encryption is used to encrypt large amounts of data, sometimes called data streams**
  - Also known as single, 1-key, or secret key, encryption
- **Symmetric key algorithms usually have a large number of possible keys**
  - A 56-bit key size produces 2^{56} or 72,057,594,037,927,936 different keys
    - Can you name that huge number?
  - A computer that could test 8,000,000 56-bit keys/second would take 285 years to generate and test all of the 56-bit keys
- **There are two categories of symmetric encryption algorithms**
  - Block algorithms, where data is encrypted one block at a time
  - Stream algorithms, where data is encrypted one byte at a time

Symmetric Encryption Algorithms

- **Many symmetric key algorithms exist; e.g.,**

Algorithm	Category	Key length	Comment
DES	Block	56 bits	Standardized by IBM in 1970s
Triple-DES	Block	112/168 bits	Twice/three x key length of DES
IDEA	Block	128 bits	Used by PGP
RC2	Block	1–2048 bits	From RSA Data Security
RC4	Stream	1–2048 bits	From RSA Data Security
AES	Block	128, 192, 256	NIST, NSA

- **Biggest problem with using symmetric key encryption is key distribution**
  - Both parties must know the same key
    - Requires a pre-established relationship between the parties

DES = Data Encryption Standard PGP = Pretty Good Privacy NSA = National Security Agency
NIST = National Institute for Science and Technology IDEA = International Data Encryption Algorithm

Public Key Encryption

- **Public key encryption requires the use of a specially generated public key/private key pair**
  - Public key and private key are mathematically related
    - Unfortunately, the complex mathematical relationship makes the encryption and decryption process relatively slow
- **The relationship between the public and private keys is such that**
  - Data encrypted with the public key can be decrypted only with the private key
  - Data encrypted with the private key can be decrypted only with the public key
- **With public key encryption, it is possible to communicate using encryption without a pre-established relationship**
  - Can also be used as the basis for an authentication mechanism
- **Browsers have many public keys in them already**
  - In IE, go to Tools | Internet Options | Content | Trusted Root Authorities

Public Key Encryption Algorithms

- **Public key algorithms appeared in the late 1970's**
  - The two prominent algorithms are Diffie Hellmann and RSA
    - RSA is used in SSL
    - The RSA key may be any length, depending on the particular implementation
- **To send someone confidential information using public key encryption**
  - Both parties have to generate a key pair, i.e., a private and a public key
  - Somehow, exchange the public keys.
  - Encrypt the data using the recipient's public key
  - Data can be decrypted only by using recipient's private key
    - Private key must not be disclosed
- **To prove that you are the person sending the data**
  - Encrypt the data with your private key
  - Anyone with access to your public key can decrypt it
    - Proves that it was you who sent the data

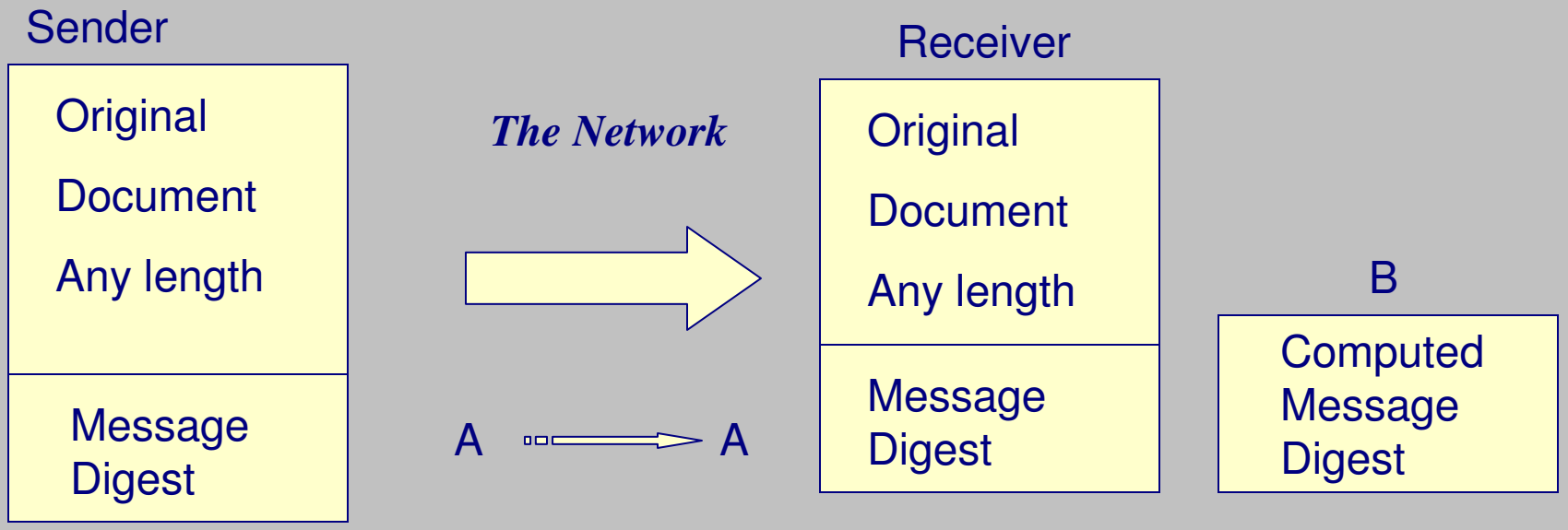
Message Digest Algorithms – MD5 and SHA1

- **Message digest algorithms**
  - Produce the size output
  - The output is called a Message Digest (MD)
  - The MD is always the same size
  - MD5 is an often used algorithm, and produces a 128-bit MD
  - SHA1 is a newer algorithm and produces a 160-bit MD
- **Message digest algorithms**
  - Always yield the same MD for the same document
  - If any portion (a single bit, character, phrase) of a document is changed, and a new MD created
    - The new MD will be different from the original MD
- **An MD (128 or 160 bits) is a "digital fingerprint" of a document**
  - Can be used to check integrity

Message Digest Algorithm -- MD5 -- Demonstration

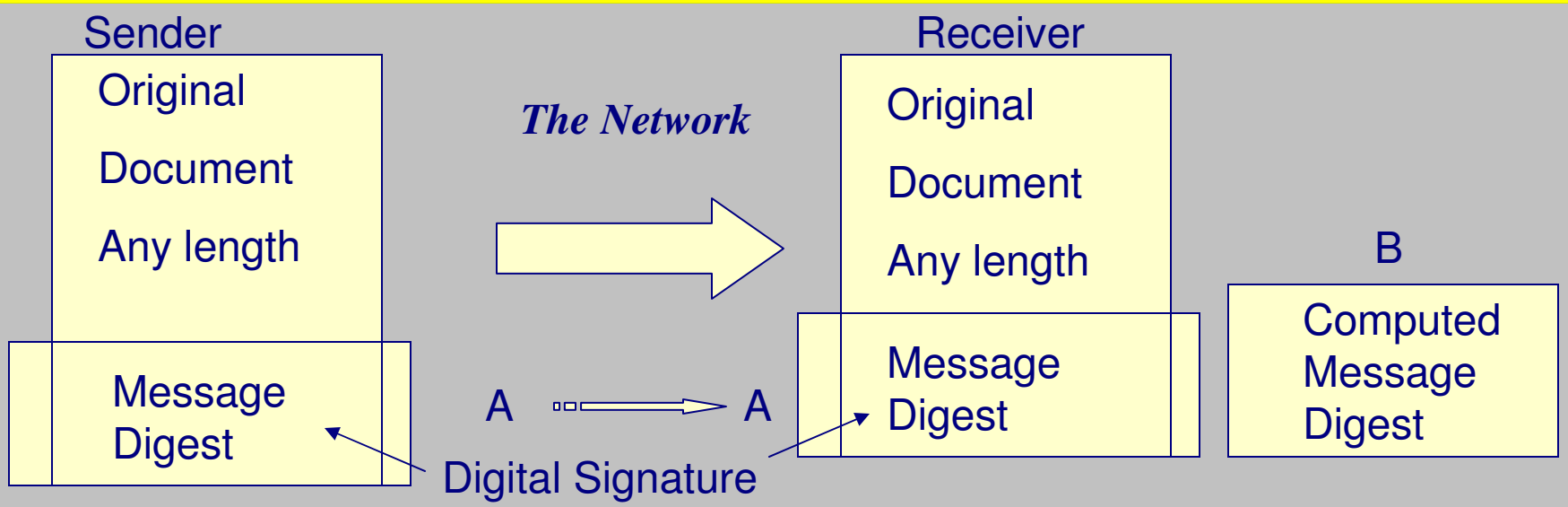
- **In this demo, we will change 1 bit of a document and observe the effect on the MD5 message digest**
- **Using the publicly available MD5.exe program**
  - First create a text file with just your first name a space and then your last name. Capitalize the first letters of your two names
    - For your instructor, this would be Jerry Scott
    - Name the file yourfirstname1.txt, as in jerry1.txt
  - Use the md5.exe program to produce and save a MD, as in
    - Md5 < jerry1.txt >md1.txt
  - Change the first letter to a lowercase, as in jerry Scott and save the file as jerry2.txt
  - Use the md5.exe program to produce a new MD, as in
    - Md5 <jerry2.txt > md2.txt
  - Notice how different the two MD's are.
    - You only changed 1 bit!

Achieving Data Integrity



- Sender calculates a MD (A)
  - Sends it along with the original document.
- Receiver calculates its own MD (B) from the document it received
  - Compares B to the received MD (A) the sender calculated.
- If A is the same as B, receiver thinks the packet is unaltered.

Better Data Integrity – Using a Digital Signature



Sender encrypts the MD (A) with his/her private key. **This is called a digital signature.**

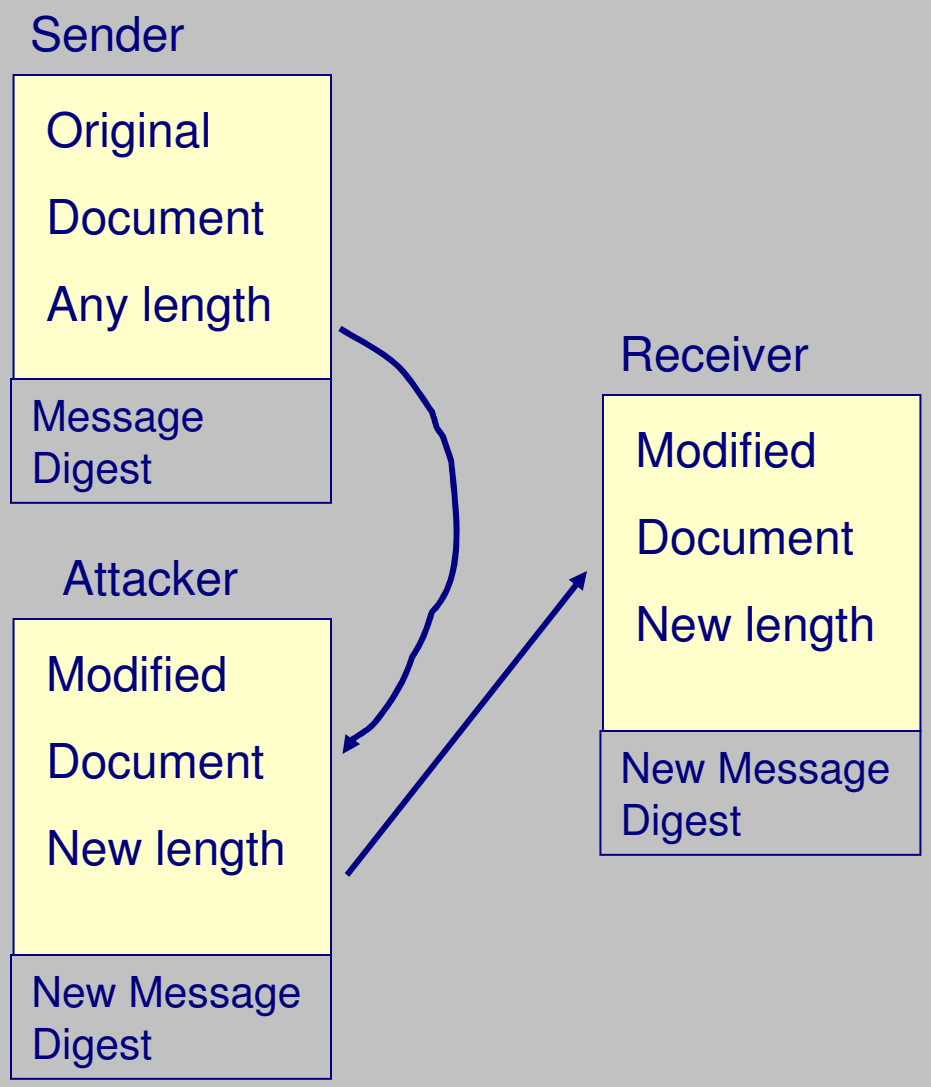
Receiver uses his/her copy of the sender's public key to decrypt the digital signature. This yields an MD (hopefully the same as the sender sent).

Receiver calculates its own MD (B) from the document it received over the network and compares it to the decrypted MD (A)

If the MD A is the same as MD B, the receiver thinks the packet is unaltered.

What Does a Digital Signature Add

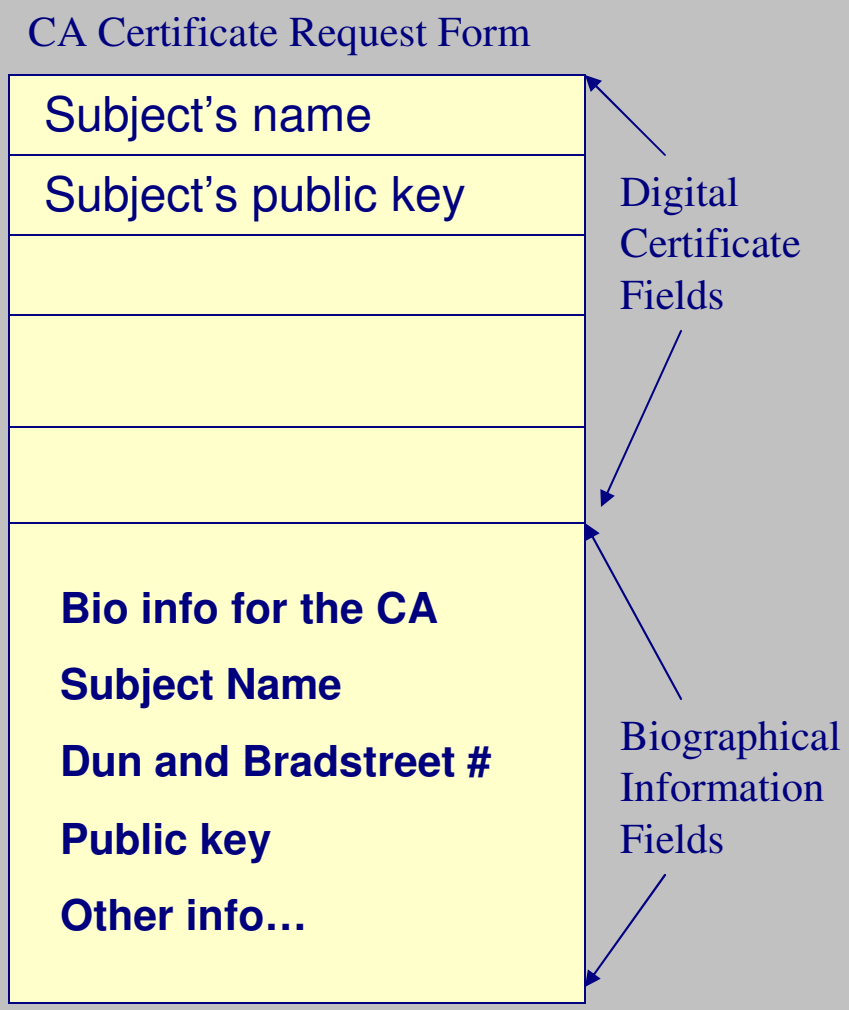
- **In the classic interception or person-in-the-middle attack**
  - An attacker who knows the MD algorithm can add data, re-compute a new MD, and send the document to the ultimate receiver
  - The receiver would get the new MD, calculate its own MD on the data received, and the two MD's would be the same
  - The receiver will not know the data has been modified
- **When the sender encrypts the MD with his/her private key, the attacker cannot add data and encrypt it with the sender's private key**



What is a Digital Certificate?

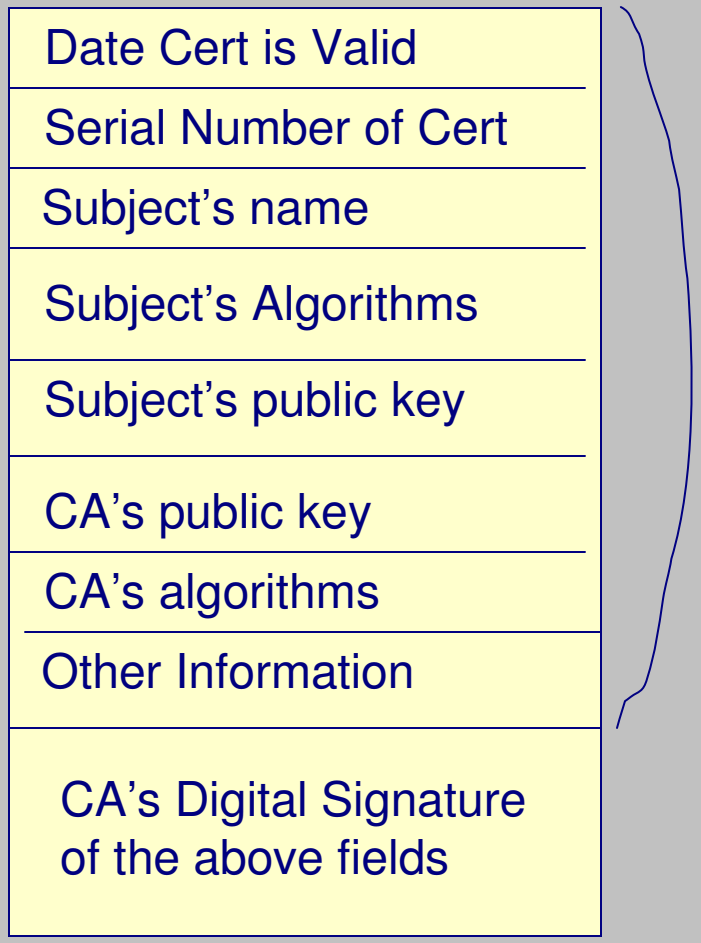
- **To get a digital certificate, a server or user or company contacts a certificate authority, CA.**
- **The CA runs software that issues certificates**
- **There are different kinds of digital certificates. The difference is in the fields in the certificate**
  - User, server, software publisher, financial
- **CA's have different issuance policies for different kinds of digital certificate**

How Do You Get A Digital Certificate?



- **User fills out the CA's certificate request form**
- **CA uses Bio information to verify the identify of the user or service**
- **CA puts the required fields into the certificate**
- **The CA calculates a MD of the certificate fields**
- **The CA encrypts the MD with its private key**
  - This is the certificate's digital signature

Verifying A Digital Certificate



- **To verify a digital certificate**
  - Client uses its copy of the CA's public key
  - Stored in your browser
- **Decrypts the CA's Digital Signature**
  - Gets CA's MD
- **Calculates "received" MD on the received fields in the Cert (minus the Digital Signature**
- **If the CA's MD equals the received MD, the cert is verified**
- **All fields in the Cert are now valid, including the subject's name and public key**
- **After verification, the receiver knows it has a correct public key and a correct Subject name**

Verifying Integrity of Software

What the Software Publisher Sends You

- **To verify a signed applet or signed Active X download**
- **The client's browser first verifies Software publisher's digital certificate**
  - Gets Software publisher's name and public key
- **Using the Software Publisher's public key**
  - Client browser then decrypts Software publisher's Digital Signature to get the Software publisher's MD
- **Client browser then calculates the "Received MD" on the downloaded software**
- **If the software publisher's MD = "Received MD", the software is verified**
- **Client now knows who the applet is from and that is exactly what was sent**

SSL History

- **Netscape developed the SSL prototype SSL in 1994 – never released it**
- **Then in 1994, SSLV2 was released by Netscape**
  - SSLV2 was weak cryptographically and a hack against it was done
- **In 1995, Microsoft began developing its version, which it named PCT**
  - They also developed the Secure Transport Layer Protocol in 1996
- **Also in 1995, Netscape released SSLV3**
- **The Internet Engineering Task Force (IETF) chartered a working group to develop an internet standard for this type of protocol in 1996**
  - It wanted to standardize a new protocol around SSLV3, PCT, and STLP
  - This effort failed. Netscape went their own way with SSLV3
  - IETF's internal policies caused its version, Transport Layer Security (TLS) to be delivered in January, 1999.
    - TLS uses different algorithms and data structures than SSLV3
- **Today, most browsers support SSLV2, SSLV3, and TLS**

How Does SSL Work -- 1

- **The first stage in SSL is the handshake C → S**
  - Client issues an https in the address line to start SSL handshake
  - Client tells server which SSL version and algorithms it can use
  - Client sends a 32-byte random number it will use later in the key generation process
- **The second stage is the Server's initial answer S → C**
  - Server sends the client its server digital certificate so the client can verify its Subject name and get its public key
  - Server also sends client a 32-byte random number which it will use later as part of its key generation process and a 32-byte session-id.
  - Server chooses an algorithm set from what the client just sent
- **In the third stage, the client computer verifies the server's digital certificate**
  - Client uses CA's public key, which is in its browser, to decrypt the digital signature on the Server's certificate.
  - This produces the CA's MD of the server's digital certificate fields
  - Then the client uses the same algorithm to create a message digest of the certificate fields it has just received
  - Client compares the two MD's. If they are the same, the digital certificate is verified.

How Does SSL Work -- 2

- **In step four, the client extracts Server's name and public key from the certificate**
- **In the fifth step, the client creates a 48-byte pre-master secret, which it encrypts with the Server's public key. Only the server has the private key to encrypt this message**
  - The encrypted pre-master secret is 64 bytes long.
  - The client calculates an MD of this information and encrypts it with the public key
- **In the sixth step, the server**
  - Uses its private key to decrypt the encrypted message, getting the pre-master secret and the client MD of it
  - Calculates its own MD of the received pre-master secret
  - When the two MD's agree, the server has verified the pre-master secret
- **In the seventh step, the client and the server use a Key Derivation Function to independently compute their own symmetric keys from the pre-master secret**
  - Their random numbers are input to the KDF
  - There are four keys to be created
    - Two for encryption: one for C to encrypt to S, and one for S to encrypt to C
    - Two for signing: one for C to sign to S, and one for S to sign to C

How Does SSL Work -- 3

- **Now that both C and S have candidates for their symmetric keys, they need to test them for validity**
- **To do this, each side sends its own MD calculated from the following fields**
  - The negotiated master secret, NMS
  - The concatenated handshake messages that side received
  - The MD is actually an MD within an MD. It's format is
    - MD(NMS + padding1 + MD(Concatenated Messages + NMS + padding2))
- **Each side decrypts its received MD and compares it to the same messages it has received and sent.**
  - If these are the same, the keys are valid
  - If the two MD's are not the same, the SSL connection fails
- **Once SSL has been setup, each side now uses the four session keys to encrypt and sign each frame sent to the other**
- **SSL is very conservative: Each frame is encrypted and an MD for each frame is calculated**
  - Each side receives a frame, decrypts it, and then uses the two MD technique to verify its integrity