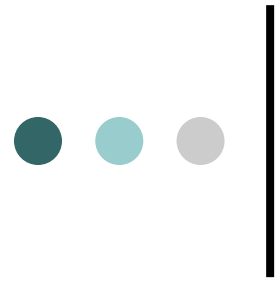


Lecture 3

Intro to Network Security



Chapter 5

Security Basics



Security Basics

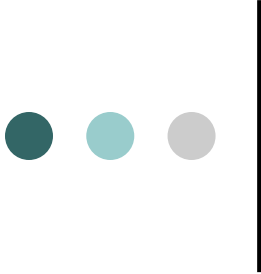
Why do you care about security?

- Identity theft!
- Bitcoin mining using your credentials?



Security Basics—Symmetric Key Cryptography and Asymmetric Key/Public Key Cryptography

- Symmetric key cryptography uses the same key to encrypt the text and to decrypt the text
 - This has the problem that both sides have to have some way to know the key (which is where public key cryptography comes in)
- With public key cryptography, two mathematically related keys are produced: a public key and a private key
 - A message that is encrypted using the public key can only be decrypted by its associated private key
 - A message that is encrypted with the private key can only be decrypted by the public key
 - So this method is also called asymmetric key cryptography since different keys are used by the encryptor and the decryptor



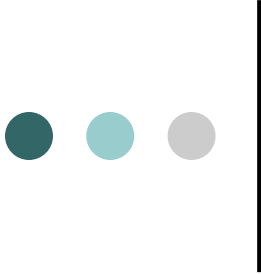
Security Basics—Symmetric Key Cryptography and Asymmetric Key/Public Key Cryptography (cont'd)

- Public keys are often used to provide a secure connection between two entities, so that the two entities can use to negotiate the use of a Symmetric key
- This is done because encryption of data using a typical symmetric key algorithm is less computationally intensive than using a public key algorithm



Security Basics—Hash (Message Digest) Functions

- A hash function creates a fixed length compressed value from an arbitrary length digital message using a mathematical function
- The hash value is much smaller than the original digital message, so a hash is often called a “digest.”
- Message Digest algorithms are used to create Digital Signatures and Message Authentication Codes.
- Well known hash functions include:
 - Message Digest (MD)
 - Secure Hash Algorithm (SHA)



Security Basics—Hash (Message Digest) Functions (Cont'd)

○ Message Digest (MD)

- MD5 has been widely used—128 bit (16 byte) hash value
 - others include: MD6, MD4, MD2
- In 2004 successful attacks on MD5 were reported, so MD5 is no longer recommended for use



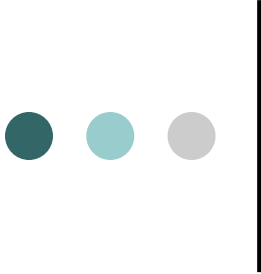
Security Basics—Hash (Message Digest) Functions (Cont'd)

- Secure Hash Algorithm (SHA)
 - SHA1 has been widely used—160 bit (20 byte) hash value
 - Successful attacks reported in 2005
 - SHA1 will be replaced in major browsers by 2017
 - SHA2 family SHA256, SHA512, etc. based on number of bits in the hash value
 - SHA3 released by National Institute of Standards in 2015



Security Basics—Hash (Message Digest) Functions (Cont'd)

- Secure Hash Algorithm (SHA)
 - SHA1 has been widely used—160 bit (20 byte) hash value
 - Successful attacks reported in 2005
 - SHA1 will be replaced in major browsers by 2017
 - SHA2 family SHA256, SHA512, etc. based on number of bits in the hash value
 - SHA3 released by National Institute of Standards in 2015



Security Basics—Digital Signatures

Digital signatures are used to tell the recipient of a digital message:

- that the message really came from the person it was supposed to come from
- that the message hasn't been tampered with since that person sent it



Security Basics—Digital Signatures (cont'd)

- To create a digital signature, the digital message is first hashed, using an algorithm such as SHA-1 or etc.
- Then the sender's private key is used to encrypt the hash
 - The reason only the hash is encrypted is that the hash is a fixed length value whereas the original message may be an arbitrary length
- Then the message is sent to the recipient along with the digital signature
- With a digital signature, it's possible to have messages verified by a receiver (with access to the public key) without having that receiver being able to later “pretend” to be the (original) sender to some third party.



Security Basics—Digital Signatures (cont'd)

- The recipient of the message and digital signature uses the sender's public key to decrypt the digital signature
- Then the message itself is hashed
- The hash from the message is compared to the hash from the decrypted digital signature
- If the two are the same, then the recipient accepts the message



Security Basics—Message Authentication Code (MAC)

- A Message Authentication code (MAC) is different from a digital signature in that it uses a secret (symmetric) key that is shared by both sender and receiver
 - A digital signature is encrypted using a private key and decrypted using a public key
- Since both sender and receiver use the same secret (symmetric) key
 - Any receiver that can verify a MAC can generate MACs of its own
 - So it is possible the receiver could later pretend to be the original sender, and send a message to a third party



Security Basics—Message Authentication Code (MAC)

One well known Message Authentication Code is the Hash-based Message Authentication Code (HMAC) that combines a hashing function with a secret key. It is calculated generally as follows:

$$\text{HMAC}(\text{key}, \text{message}) = \text{hash} \left(\begin{array}{l} (\text{secret key XOR outer-padding}) \text{ concatenated with} \\ \text{hash} \left(\begin{array}{l} (\text{secret key XOR inner padding}) \\ \text{concatenated with message} \end{array} \right) \end{array} \right)$$

where

outer padding = 0x5c0x5c...0x5c (block long hex constant)

Inner padding = 0x360x36...0x36 (block long hex constant)



Security Basics—Public Key Infrastructure and Certificate Authorities

- How to do public key infrastructure is specified in standard ITU-T X.509, also see IETF RFC 5280
- When a client wishes to connect to a server on a network, a third party that is trusted by both client and server verifies the server's credentials so the client will know that the server is who it says it is.
- The trusted third party is called a “Certification Authority”



Security Basics—Public Key Infrastructure and Certificate Authorities (Cont'd)

- The server's credentials are actually verified in advance of any desired communication
- At some point the user who owns the server issues a Certificate Signing Request including the user's credentials and the server the user wants to have verified to a Certification Authority
- The Certification Authority then verifies the user's credentials
 - it may run some outside checks about the user's business in order to do this



Security Basics—Public Key Infrastructure and Certificate Authorities (Cont'd)

- Once a user's credentials have been verified by the certificate authority, the certificate authority will issue a certificate
- The certificate contains the name of the user (as well as other information) and the user's public key
 - The Certification Authority digitally signs this certificate
- The user's public key and associated private key can:
 - have previously been generated by the user and the public key sent on to the Certification Authority, or
 - the Certification Authority can generate the public key and private key, and send both keys separately to the user (as well as including the public key in the certificate)



Security Basics—Public Key Infrastructure and Certificate Authorities (Cont'd)

- Once the user has a certificate, the user's server can present that certificate to any client that wants to connect to the user's server
- When a client wants to connect to that server, the server sends the certificate to the client
- The client then verifies the issuer of the certificate to its own list of trusted Certification Authorities
- To the client who receives this certificate, the certificate says that the certification authority vouches that the user who is named in the certificate truly owns the private key that is associated with the public key in the certificate



Security Basics—Public Key Infrastructure and Certificate Authorities (Cont'd)

- The Certification Authority is called the “root of trust”
 - That is, the Certification Authority itself is the trusted third party.
 - A Certificate issued by the root Certification Authority itself is called a “root certificate”
- Note that the root certificate itself is self-signed by the Certification Authority.
- Often there is a separate “signing certificate” that is signed by the root certificate
 - in this case the Certification Authority may take the root certificate offline and store it
 - Then the signing certificate will be used to sign server certificates



Security Basics—Public Key Infrastructure and Certificate Authorities (Cont'd)

- The Certification Authority can also certify other entities, called Registration Authorities (or subordinate Certification Authorities or intermediate Certification Authorities), to issue certificates
 - or the Certification Authority can issue certificates itself
- So if a given Certification Authority issued the certificate, and the client finds that particular Certification Authority on its own list of trusted Certification Authorities,
 - then all is well, the client believes that the server is who it says it is



Security Basics—Public Key Infrastructure and Certificate Authorities (Cont'd)

- However, if the given Certification Authority is not on the client's own list of trusted Certification Authorities:
 - then the client must look at the issuing Certification Authority's own certificate and see if that was issued by a trusted Certification Authority
 - if this is the case, then the issuing Certification Authority was a subordinate Certification Authority to the original trusted third party Certification Authority
 - If so, then all is well
 - If not, then the certificate is not accepted



Security Basics—Public Key Infrastructure and Certificate Authorities (Cont'd)

- Note that web browsers come with the signing certificates (or root certificates) of many Certificate Authorities already installed inside them
- To make this procedure work, a Certificate Database is needed on the Certificate Authority:
 - it saves the certificate requests that have been issued
 - it saves whether or not they were later revoked
- on the user's local machine a Certificate Store is necessary
 - to save certificates that were issued to the user



Security Basics—Public Key Infrastructure and Certificate Authorities (Cont'd)

Contents of a typical digital certificate are as follows:

- Serial Number: identifier number that identifies the certificate (can later be used for revocation)
- Subject: The entity (for ex., server belonging to a user) that is being identified
- Algorithm used to create the signature
- Signature itself
- Issuer: trusted third party that issued the certificate
- Validity period
- Public Key: entity that is being identified's public key
- Thumbprint Algorithm: Algorithm used to hash the public key certificate
- Thumbprint (fingerprint): The hash of the public key certificate itself



Security Basics—Public Key Infrastructure and Certificate Authorities (Cont'd)

- If a certificate must be revoked, then the Certification Authority creates a Certification Revocation List (CRL):
 - that is digitally signed by the Certification Authority
 - that identifies revoked certificates by certification identification number
- Certification Revocation Lists are:
 - typically issued periodically
 - can become quite large



Security Basics—Transport Layer Security (TLS) and Secure Sockets Layer (SSL)

According to Information Security Stack Exchange (2011):

“HTTPS is HTTP-within-SSL/TLS. SSL (TLS) establishes a secured, bidirectional tunnel for arbitrary binary data between two hosts. HTTP is a protocol for sending requests and receiving answers, each request and answer consisting of detailed headers and (possibly) some content. HTTP is meant to run over a bidirectional tunnel for arbitrary binary data; when that tunnel is an SSL/TLS connection, then the whole is called ‘HTTPS’.”



Security Basics—Transport Layer Security (TLS) and Secure Sockets Layer (SSL) (cont'd)

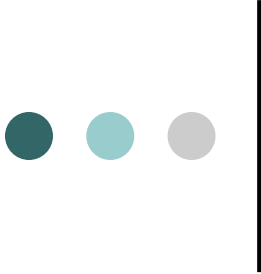
- TLS and SSL provide:
 - Confidentiality
 - Message authentication
- Both TSL and SSL run on top of TCP/IP but below HTTP
- TLS is a descendant/follow on of SSL
 - SSL v. 3.0 dates from 1996
 - TLS v. 1.0 dates from 1999
 - TLS v 1.0 cannot talk directly to SSL v 3.0, but can negotiate to use SSL v 3.0
 - TLS 1.2 dates from 2008
 - RFC 6175 prohibits TLS 1.2 from negotiating to use SSL 2.0



Security Basics—Transport Layer Security (TLS) and Secure Sockets Layer (SSL) (cont'd)

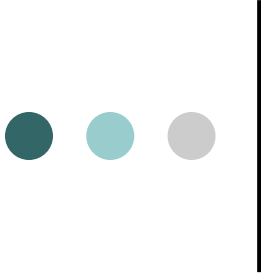
Deficiencies of SSL 2.0, according to RFC 6175:

- It uses MD5 which is no longer considered secure
- It is subject to man in the middle attacks
 - that select weak cryptography algorithm choices
 - because it does not protect handshake messages
 - that can terminate a session
 - because the communicating entities cannot determine whether the session was ended legitimately
- The same key is used for message integrity and message encryption
 - a problem if the cryptography algorithm choice is not strong



Security Basics—Transport Layer Security (TLS) and Secure Sockets Layer (SSL)—But How does TLS Work?

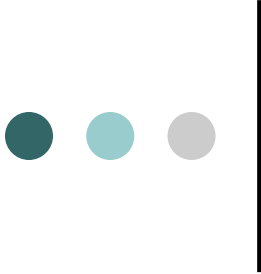
- TLS provides:
 - Confidentiality through use of encryption
 - Authentication through use of public key certificates
 - Integrity through use of message authentication codes



Security Basics—Transport Layer Security (TLS) and Secure Sockets Layer (SSL)—But How does TLS Work?

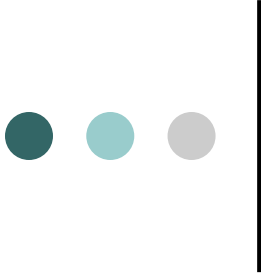
TLS uses several sub-protocols, see RFC 5246:

- Record protocol
 - Breaks a message into blocks
 - (Optionally) compresses the data
 - adds a MAC
 - Then encrypts the whole and transmits it
- Other protocols on top of the Record layer:
 - Handshake protocol Cipher spec protocol—changes cipher strategies/signals the beginning of secure communication
 - Application data protocol



Security Basics—Transport Layer Security (TLS) and Secure Sockets Layer (SSL)—But How does TLS Work? (cont'd)

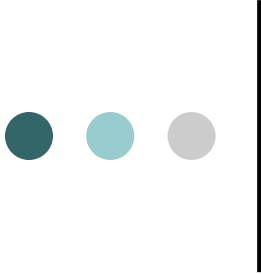
- Handshake protocol
 - negotiates what algorithms, etc. will be used during a session
- Alert protocol
 - sends error messages, can be warnings or else fatal errors (with connection terminated)
 - Some possible errors are:
 - bad certificate
 - certificate revoked
 - decryption error—unable to verify a signature or validate a message
 - record overflow
- Cipher spec protocol
 - changes cipher strategies/signals the beginning of secure communication
- Application data protocol
 - data message from the application are carried by the record layer and treated as transparent data



Security Basics—Transport Layer Security (TLS) and Secure Sockets Layer (SSL)—But How does TLS Work? (cont'd)

At the beginning of a connection, the handshake protocol does the following:

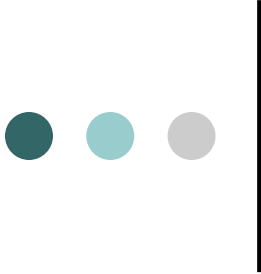
- Hello message from client to server, hello response from server to client:
 - used to negotiate:
 - TSL or SSL version to be used
 - Session ID
 - Cipher Suite
 - Cipher suites are combination of cryptographic algorithms for:
 - Key exchange (Diffie-Hellman, RSA, etc.)
 - Cipher (AES, etc.)
 - MAC (SHA256, etc.)
 - Compression Method
 - as part of the Hello exchange, the server sends a certificate to the client, this includes the server's public key (for example, RSA)



Security Basics—Transport Layer Security (TLS) and Secure Sockets Layer (SSL)—But How does TLS Work? (cont'd)

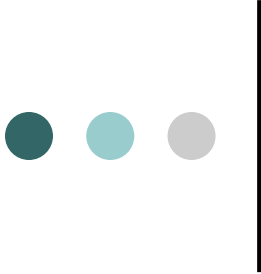
The Client then does the following:

- generates a pre-master secret key:
 - 48 bytes
 - generated by concatenating protocol version with some randomly generated bytes
 - Encrypted with the server's RSA public key (from the certificate)
- sends it to the server



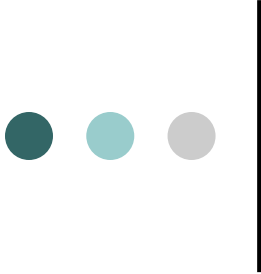
Security Basics—Transport Layer Security (TLS) and Secure Sockets Layer (SSL)—But How does TLS Work? (cont'd)

- The Server decrypts the pre-master secret using its private key
- Both client and server generate a master secret using the pre-master secret, then immediately delete the pre-master secret
- The client and the server use the master secret to generate the session keys
 - symmetric keys used to encrypt and decrypt data transferred during the session (AES, for example)



Security Basics—Transport Layer Security (TLS) and Secure Sockets Layer (SSL)—But How does TLS Work? (cont'd)

- The client can now send the server a message that is encrypted with the session key and authenticated with the MAC
 - for example, HMAC with SHA256
- The server determines:
 - that the MAC was authentic
 - sends back an encrypted message with a MAC
 - that the client also determines is authentic



Security Basics—Transport Layer Security (TLS) and Secure Sockets Layer (SSL)—Cryptographic Message Syntax (CMS)

Cryptographic Message Syntax (CMS):

- used to support certificate-based key management in X.509 Public Key Infrastructure
- supports various architectures for certificate-based key management
 - by providing an encapsulation mechanism that supports encryption and digital signatures
 - allows one encapsulation envelope to be nested inside another
 - allows encapsulated data to be digitally signed



Security Basics—Transport Layer Security (TLS) and Secure Sockets Layer (SSL)—Cryptographic Message Syntax (CMS) (cont'd)

In CMS:

- the content
 - that is, the data that is encapsulated
- is associated with an identifier that describes its type
 - this is called the content-type
-



Security Basics—Transport Layer Security (TLS) and Secure Sockets Layer (SSL)—Cryptographic Message Syntax (CMS) (cont'd)

- In CMS, normally the content-type is specified using Abstract Syntax Notation One (ASN.1)
 - although many different specialized content-types are supported
 - ASN.1 is a formal notation that describes data that is transmitted by communications protocols



Security Basics—Transport Layer Security (TLS) and Secure Sockets Layer (SSL)—Cryptographic Message Syntax (CMS) (cont'd)

Basic types supported in ASN.1 include (among others):

- BIT STRING—an arbitrary string of 0s and 1s
- BOOLEAN
- INTEGER
- NULL
- OBJECT IDENTIFIER—a sequence of integers that uniquely identify an object
- OCTET STRING—an arbitrary string of 8 bit values



Security Basics—Transport Layer Security (TLS) and Secure Sockets Layer (SSL)—Cryptographic Message Syntax (CMS) (cont'd)

String types supported in ASN.1 include (among others):

- IA5String—an arbitrary string of International Alphabet 5 (ASCII) characters
- PrintableString—an arbitrary character string that contains only printable characters
- UTF8String—a string with characters in UTF-8 format
 - uses Unicode Transformation Format 8 (UTF-8)
 - UTF-8 is the widest used character encoding on the internet
 - uses 8 bit blocks to represent a character
 - UTF-8 encoding is recommended by the W3C for XML and HTML



Security Basics—Transport Layer Security (TLS) and
Secure Sockets Layer (SSL)—Cryptographic Message
Syntax (CMS) (cont'd)

The CMS values are encoded using
Basic Encoding Rules (BER-
encoding)

- original rules specified in the ASN.1 standard to encode data into a data stream
- a binary encoding



Security Basics—Transport Layer Security (TLS) and Secure Sockets Layer (SSL)—Cryptographic Message Syntax (CMS) (cont'd)

Using BER, information is encoded using Type-Length-Value (TLV) encodings, which consist of:

- a Type identifier
- a Length field
- the data value itself
- an end of data marker (where needed)



Security Basics—Transport Layer Security (TLS) and Secure Sockets Layer (SSL)—Cryptographic Message Syntax (CMS) (cont'd)

cn_equals_directory_manager (2006) gives the following example of ASN.1 BER encoding:

- the OCTET STRING "Hello" is represented as:
 - 04 05 48 65 6C 6C 6F
 - 0x04 is the Type identifier for OCTET_STRING
 - 0x05 is the length field (specifies 5 bytes)
 - 48 65 6C 6C 6F are the IA5 (ASCII) characters for "Hello"



Security Basics—Transport Layer Security (TLS) and Secure Sockets Layer (SSL)—Cryptographic Message Syntax (CMS) (cont'd)

A digital envelope uses two layers of encryption:

- the message itself is encoded using symmetric encryption
- the symmetric key to use for decryption is sent over the network
 - but it is itself encoded (and decoded on the other end) using public key encryption
 - thus, no plain text communication of the symmetric key takes place



Security Basics—Transport Layer Security (TLS) and Secure Sockets Layer (SSL)—Cryptographic Message Syntax (CMS) (cont'd)

The six basic types provided by CMS are as follows:

1. Data—arbitrary strings of bytes, usually encapsulated in one of the other types
2. Signed-Data—includes the data plus zero or more digital signatures
3. Enveloped-Data—a digital envelope,
4. Digested-Data – includes the content plus a hash (message digest) of the content
5. Encrypted-Data—the encrypted data itself, keys are sent in some other way
6. A method for key management **MUST** be used! CMS does not define any particular key management method
7. Authenticated-Data – this consists of the content, message authentication code (MAC), and encrypted authentication keys for one or more recipients



Security Basics—Transport Layer Security (TLS) and Secure Sockets Layer (SSL)—Cryptographic Message Syntax (CMS) (cont'd)

Signed-Data—includes the data plus zero or more digital signatures

- More than one digital signature is possible, from different signers
 - Any number of signers can sign any type of content
- A hash (message digest) is computed on the content
- The hash (message digest) is digitally signed using the signer's private key



Security Basics—Transport Layer Security (TLS) and Secure Sockets Layer (SSL)—Cryptographic Message Syntax (CMS) (cont'd)

Enveloped-Data—a digital envelope, includes:

- Data encrypted using a symmetric key
- The symmetric key to decrypt the content is separately encrypted for each recipient using the recipient's public key



Security Basics—Transport Layer Security (TLS) and Secure Sockets Layer (SSL)—Cryptographic Message Syntax (CMS) (cont'd)

The Data format looks like:

- Content-type=DATA
- the data itself



Security Basics—Transport Layer Security (TLS) and Secure Sockets Layer (SSL)—Cryptographic Message Syntax (CMS) (cont'd)

The Signed-Data format looks like:

- Content-type=SIGNATURE
- Hash (Message Digest) algorithm (can be more than one)
- the data itself
- Digital signatures (can be more than one)



Security Basics—Transport Layer Security (TLS) and Secure Sockets Layer (SSL)—Cryptographic Message Syntax (CMS) (cont'd)

The Authenticated-data format looks like:

- Content-type=AUTHENTICATED-DATA
- (optional) information about the recipient—may contain digital certificates, etc.
 - present only if needed by key management algorithm
- Message Authentication Code (MAC) algorithm used by the originator
- (optional) Message Digest Algorithm
 - Author attributes field, must be included if Message Digest Algorithm is included. Contains:
 - Content-type of information being authenticated
 - Message digest of the content
- Message Authentication Code (MAC) itself
- Encrypted Authentication Key for one or more recipients



Security Basics—Transport Layer Security (TLS) and Secure Sockets Layer (SSL)—Cryptographic Message Syntax (CMS) (cont'd)

The Enveloped-data format looks like:

- Encrypted data



Security Basics—Transport Layer Security (TLS) and Secure Sockets Layer (SSL)—Cryptographic Message Syntax (CMS) (cont'd)

The Authenticated-data format looks like:

- Content-type=AUTHENTICATED-DATA
- (optional) information about the recipient—may contain digital certificates, etc.
 - Present only if needed by key management algorithm
- Message Authentication Code (MAC) algorithm used by the originator
- (optional) Message Digest Algorithm
 - according to RFC 5652, if any information is being authenticated in addition to the content
 - then a message digest is calculated on the content,
 - the message digest of the content plus the other information are authenticated together using the authentication key
 - the result is used as the MAC