BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Middleware Security

Srikanth Gunturu

Guest Faculty
BITS, WILP

In this segment

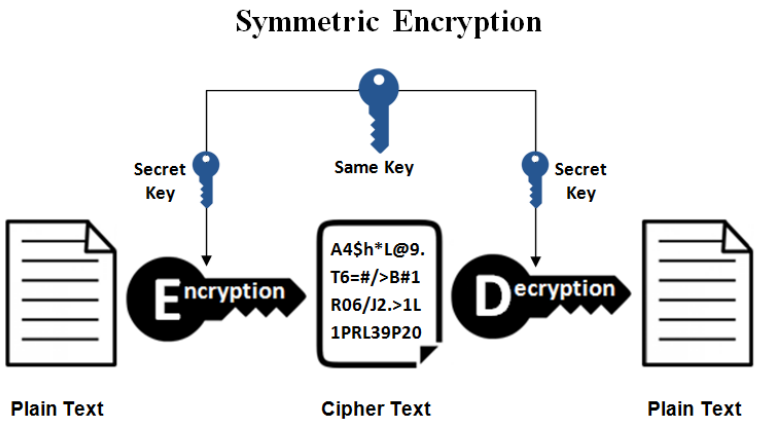
Middleware Security

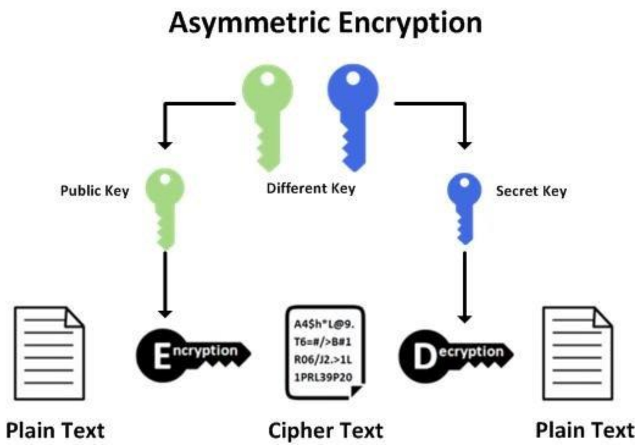
- Symmetric and Asymmetric keys
- Digital Signatures
- Message Authentication Codes
- Secure Socket Layer / Transport Layer Security

Middleware Security

Symmetric Cryptography and Asymmetric/Public Keys

- Symmetric key cryptography - uses the same key to encrypt the text and to decrypt the text
  - Both the parties need to know the key (offline sync of the keys)
- Asymmetric/Public key cryptography - uses two mathematically related keys
  - Public key
  - Private key
  - Message encrypted using either of the keys can be decrypted with the other
  - Often used to establish secure connection between two entities, to negotiate Symmetric key





Middleware Security

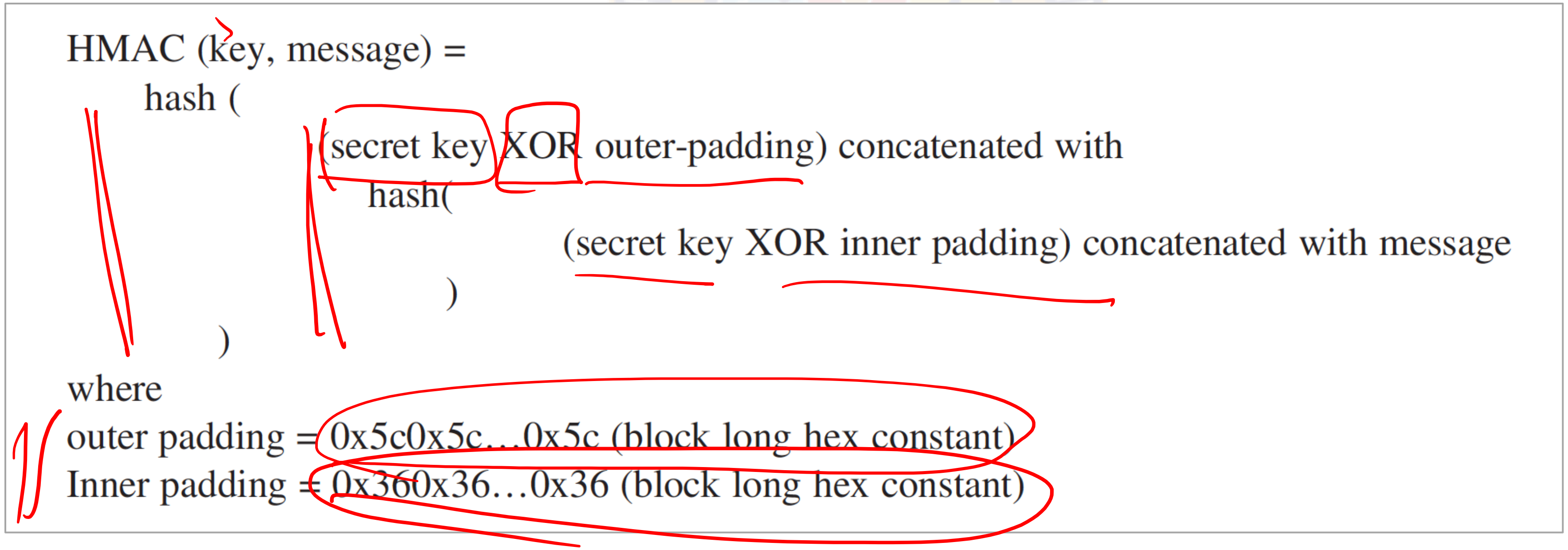
Digital Signatures

- Validates the authenticity and integrity of the digital message using asymmetric keys
- Sender side:
  - Digital message is hashed (ex: SHA-1)
  - Sender's private key is used to encrypt it
  - Message is sent along with digital signature
- Recipient side:
  - Digital signature is decrypted using sender's public key
  - Message is hashed and compared with decrypted signature
  - If they match, message is accepted as not tampered

Middleware Security

Message Authentication Codes

- Function same as digital signature, but by using symmetric keys
- Sender and receiver share same encryption key and use it for encrypting/decrypting the hash
- Caveat – A receiver can use the MAC to pretend to be the original sender to a third party
    - Workaround - Hash based MAC (HMAC) – uses hashing with secret key

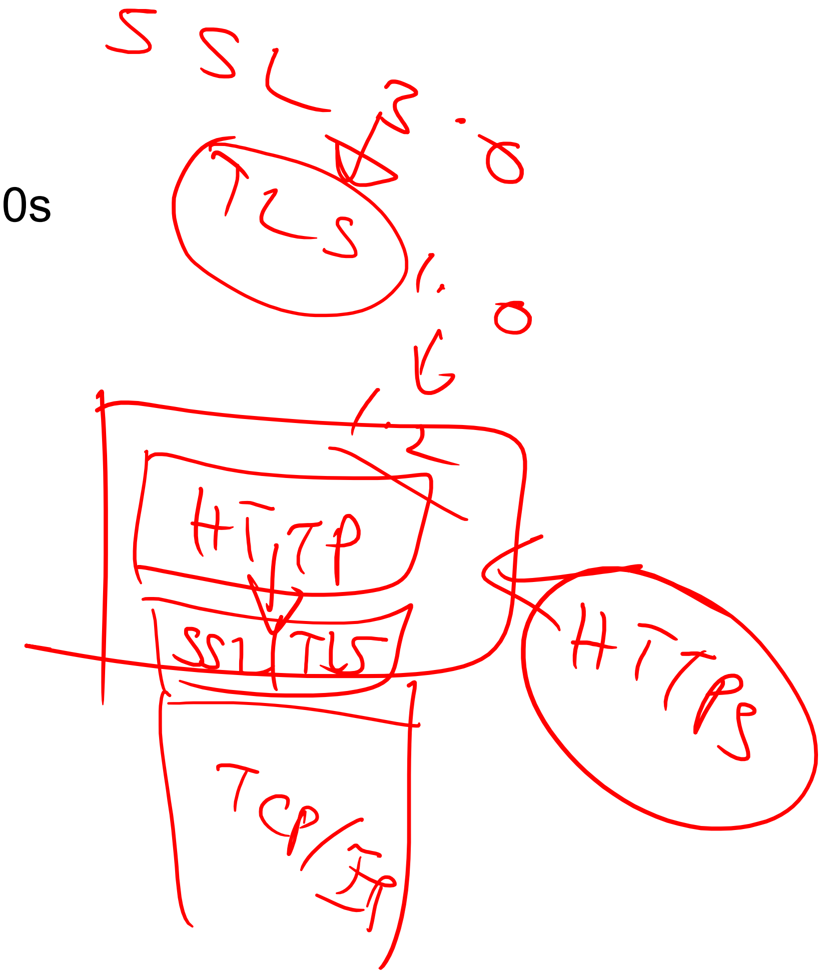


```
HMAC (key, message) =
    hash (
            (secret key XOR outer-padding) concatenated with
                hash(
                        (secret key XOR inner padding) concatenated with message
                    )
        )
where
outer padding = 0x5c0x5c…0x5c (block long hex constant)
Inner padding = 0x360x36…0x36 (block long hex constant)
```

Middleware Security

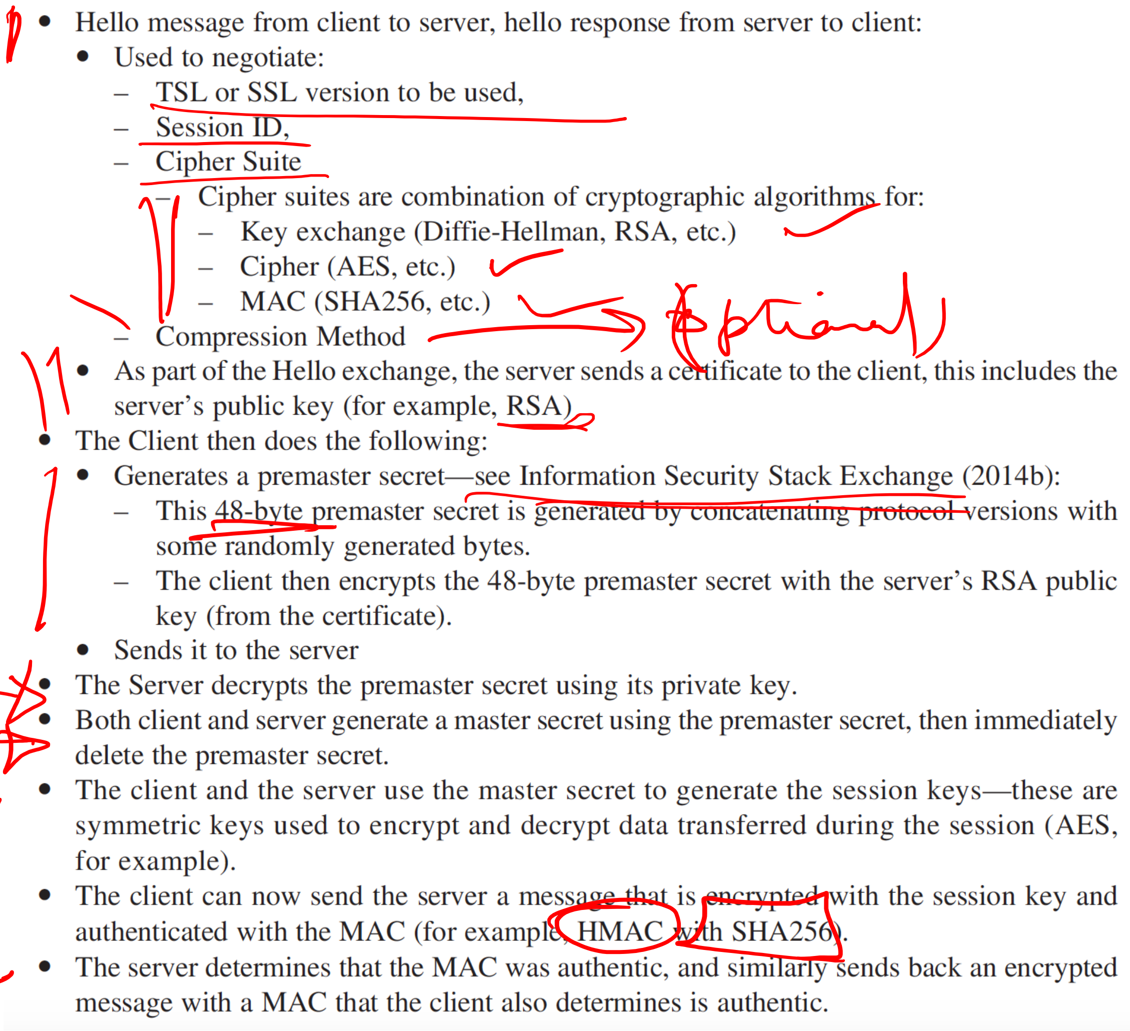
Secure Socket Layer and Transport Layer Security

- SSL Evolved in 1990s as security mechanism for HTTP (i.e. HTTPS)
- Transport Layer Security (TLS) evolved as follow-on for SSL in late ‘90s
- SSL/TLS run on TCP/IP, above HTTP – making it HTTPS
- SSL 2.0 Major Limitations (RFC 6175)
  - MD5 is no longer considered secure
  - Subject to man-in-the-middle attacks and forced session terminations
  - Same key used for message integrity and encryption
- TLS Sub protocols (RFC 5246)
  - Record protocol – Encrypts message using MAC
  - Handshake protocol – Agrees on the algorithm to use in the session
  - Alert protocol – Error handling
  - Cipher spec protocol – Changes cipher strategies/signals
  - Application data protocol – Provides data transparency to applications

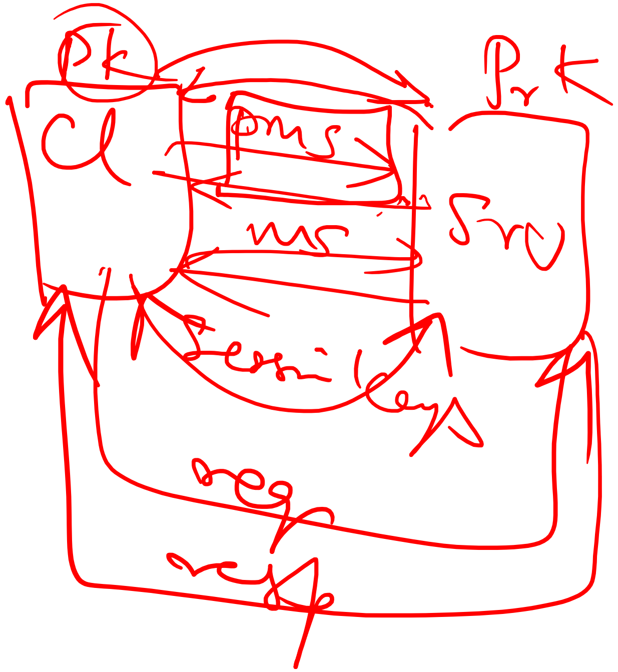


Middleware Security

TLS Handshake Protocol flow



- 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 (optional)
  - As part of the Hello exchange, the server sends a certificate to the client, this includes the server's public key (for example, RSA)
- The Client then does the following:
  - Generates a premaster secret—see Information Security Stack Exchange (2014b):
    - This 48-byte premaster secret is generated by concatenating protocol versions with some randomly generated bytes.
    - The client then encrypts the 48-byte premaster secret with the server's RSA public key (from the certificate).
  - Sends it to the server
- The Server decrypts the premaster secret using its private key.
- Both client and server generate a master secret using the premaster secret, then immediately delete the premaster secret.
- The client and the server use the master secret to generate the session keys—these are symmetric keys used to encrypt and decrypt data transferred during the session (AES, for example).
- 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, and similarly sends back an encrypted message with a MAC that the client also determines is authentic.



Thank You!

In our next session:

JMS Server setup and configuration