KF School of Computing and Information Sciences
Florida International University

CNT 4403
Computing and Network Security

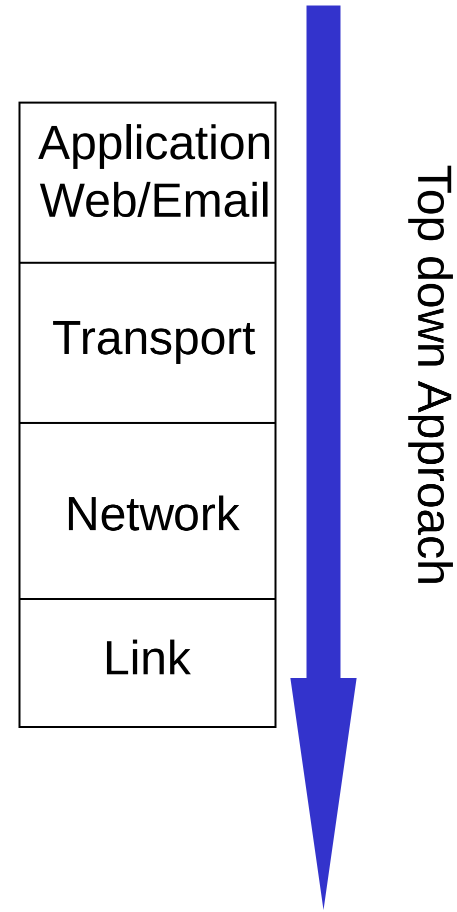
Network Security – SSL/TLS

Dr. Kemal Akkaya

E-mail: *kakkaya@fiu.edu*

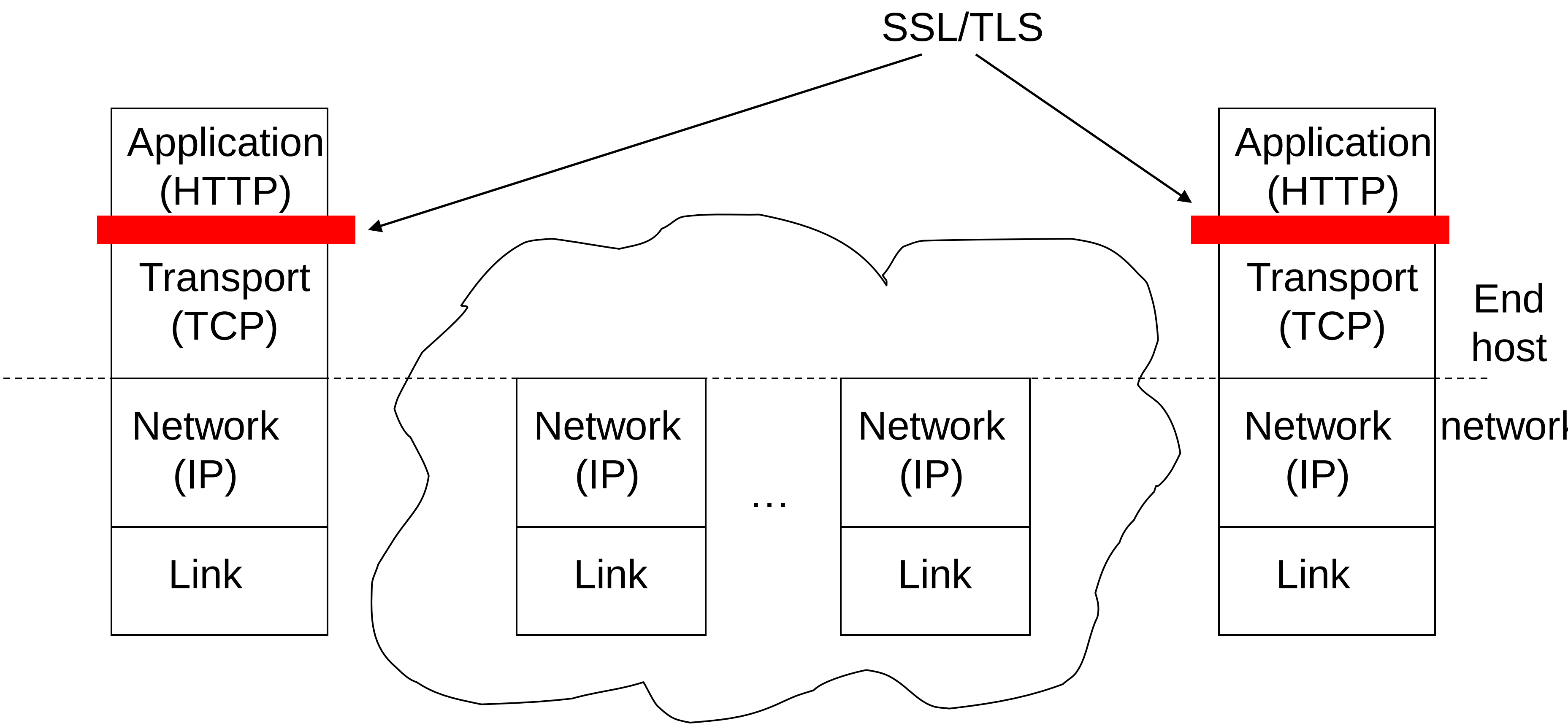
Big Picture for Protocol Security

- **Application/Transport layer based solutions**
  - Secure network-based applications
    - ✓ Web – SSL, transportation layer solution
    - ✓ Email – PGP, application layer solution
    - ✓ HTTPS, SFTP, SSH, etc.
- **Network/Link layer based solutions**
  - Secure network + support for application
    - ✓ IPSec (comes with IPv6)
      - VPN
    - ✓ Internet Security
      - BGP security
    - ✓ Wireless Security
      - IEEE 802.11 security
- **We start with SSL since it is used in other application layer protocols as well**



Security Mechanism Placement for SSL / TLS

- SSL (Secure Socket Layer) (old name)
- TLS (Transport Layer Security) (current name)



Brief History of SSL/TLS

- SSLv2
  - Released in 1995 with Netscape 1.1
  - Reverse engineered & broken by Wagner & Goldberg
- SSLv3
  - Fixed and improved, released in 1996
  - Public design process
- TLS: IETF's version; the current standard
  - First published version of TLS (1.0) is essentially very similar to SSLv3
    - ✓ TLS mandated the use of DSS instead of RSA
  - TLS 1.1 (2006)
  - TLS 1.2 (2008)
  - TLS 1.3 (2018)

SSL Design

What do we want ultimately?

- Communication between client and server
  - ✓ Confidentiality + data integrity + source authentication
- Apps: HTTPS, Secure mail

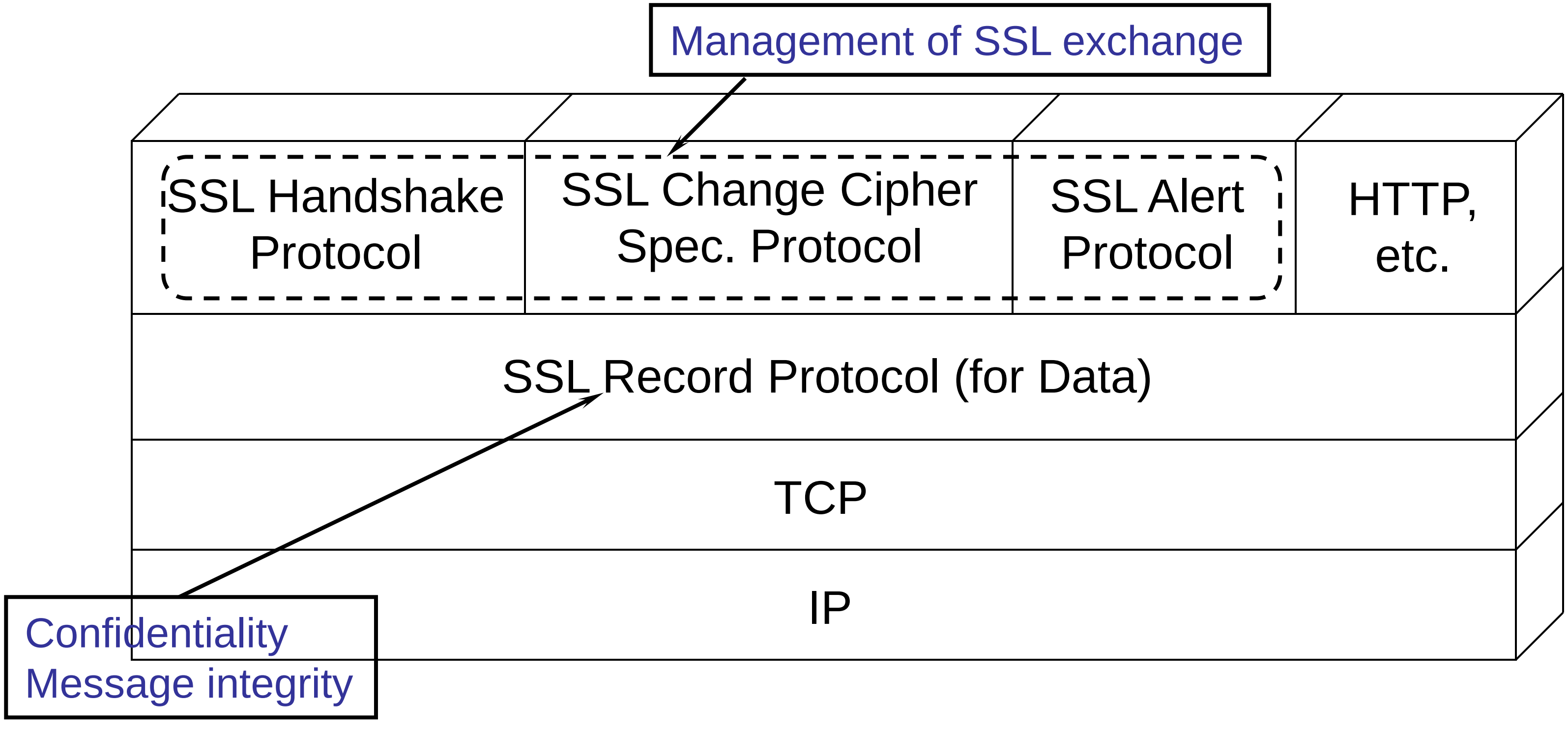
How?

- Authentication → public-key based authentication
- Confidentiality → Symmetric encryption
- Integrity → MAC

What do we need?

- Certificate for authentication
- Shared secret key 1 for encryption
- Shared secret key 2 for MAC
- Initialization vector for mode of operation

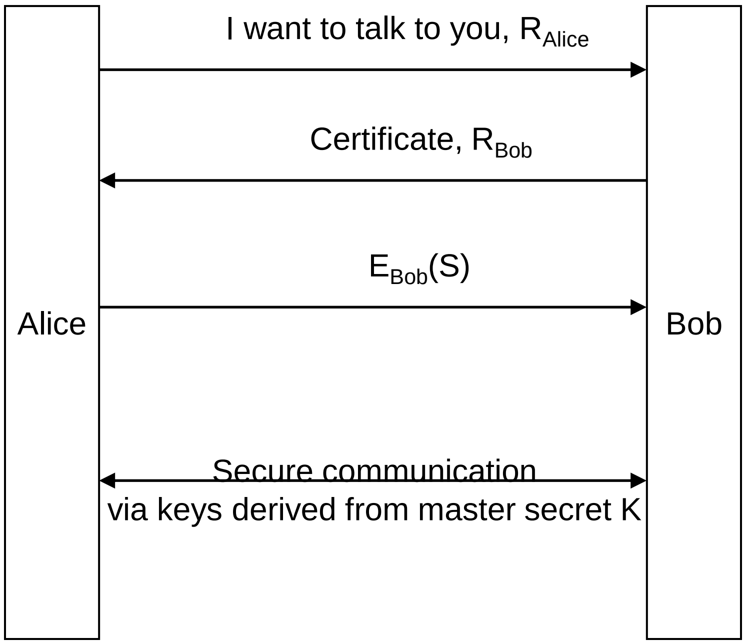
SSL/TLS Architecture



- **Record Protocol: Message encryption/integrity**
- **Management Protocols:**
  - Handshake Protocol: Identity authentication & key agreement
    - ✓ This is done for every TLS connection initially
  - Alert Protocol: Error notification (cryptographic or otherwise)
  - Change Cipher Protocol: Activate the pending crypto suite

Overview of TLS

Rough Explanation



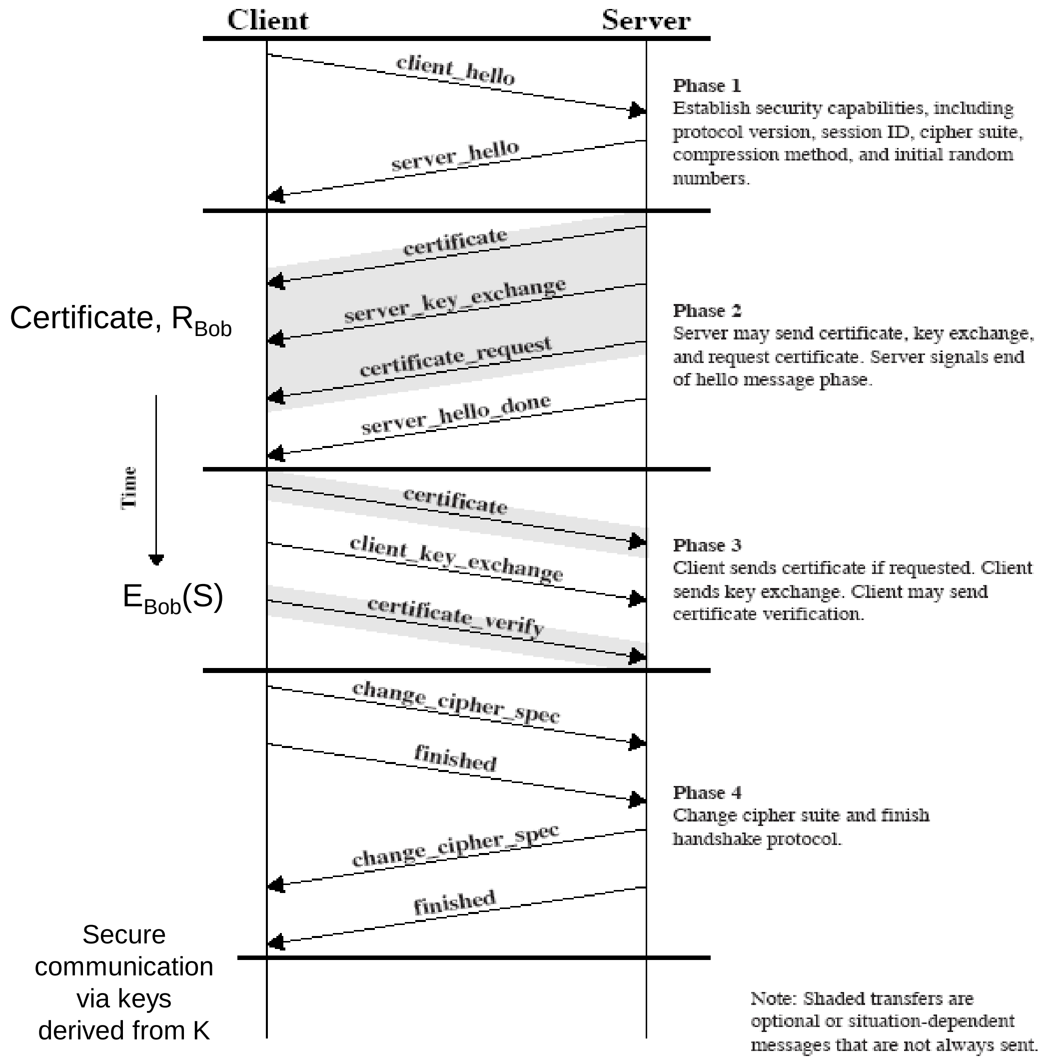


Figure 17.6 Handshake Protocol Action

Cool explanation of TLS: https://tls.ulfheim.net

Client Hello – Phase 1

- **Protocol version**
  - SSLv3(major=3, minor=0)
  - TLS (major=3, minor=1)
- **Client Random Number**
  - 32 bytes
  - First 4 bytes, time of the day in seconds, other 28 bytes random
  - Prevents replay attack
- **Optional Session ID**
  - 32 bytes – indicates the use of previous cryptographic material
- **A list of Compression algorithm**
- **A list of Cipher suites**

Client Hello - Cipher Suites

- *Crypto suite*: A complete package specifying the crypto to be used. (encryption algorithm, key length, integrity algorithm, etc.)
- ~30 predefined standard cipher suites.

SSL_NULL_WITH_NULL_NULL = { 0, 0 } INITIAL (NULL) CIPHER SUITE

PUBLIC-KEY ALGORITHM

SYMMETRIC ALGORITHM

HASH ALGORITHM

CIPHER SUITE CODES USED IN SSL MESSAGES

SSL_RSA_WITH_NULL_MD5 = { 0, 1 }

SSL_RSA_WITH_NULL_SHA = { 0, 2 }

SSL_RSA_EXPORT_WITH_RC4_40_MD5 = { 0, 3 }

SSL_RSA_WITH_RC4_128_MD5 = { 0, 4 }

SSL_RSA_WITH_RC4_128_SHA = { 0, 5 }

SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5 = { 0, 6 }

SSL_RSA_WITH_IDEA_CBC_SHA = { 0, 7 }

SSL_RSA_EXPORT_WITH_DES40_CBC_SHA = { 0, 8 }

SSL_RSA_WITH_DES_CBC_SHA = { 0, 9 }

SSL_RSA_WITH_3DES_EDE_CBC_SHA = { 0, 10 }

Server Hello – Phase 1

- **Protocol Version**
- **Server Random Number**
  - Protects against handshake replay
- **Session ID**
  - Provided to the client for later resumption of the session
- **Selected Cipher suite**
  - Usually picks client's best preference – No obligation
- **Selected Compression method**
  - Optional

Certificates – Phase 2

- **Server sends its X.509 certificate to Client**
- **X.509 Certificate associates public key with identity**
  - Certification Authority (CA) creates certificate
    - ✓ Adheres to policies and verifies identity
    - ✓ Signs certificate
  - User of Certificate must ensure it is valid
    - ✓ Must recognize accepted CA in certificate chain
      - One CA may issue certificate for another CA
    - ✓ Must verify that certificate has not been revoked
      - CA publishes Certificate Revocation List (CRL)
- **Client does not use the public keys in this X.509 certificate (it is for authentication only)**
- **Server still needs to generate a public/private key and send to client this *ephemeral* public key**
  - The public and private keys are used to agree on a shared key (later)
- **Server sends Hello done message**

Phase 3

- **Client Key Exchange**
  - Client also calculates an *ephemeral* public/private key and sends to server the public key
  - Elliptic Curve Diffie-Hellman is used for calculating the shared encryption key (more to come)
- **Client may send a certificate if requested by the server (rare)**
- **Client may also send certificate verification**

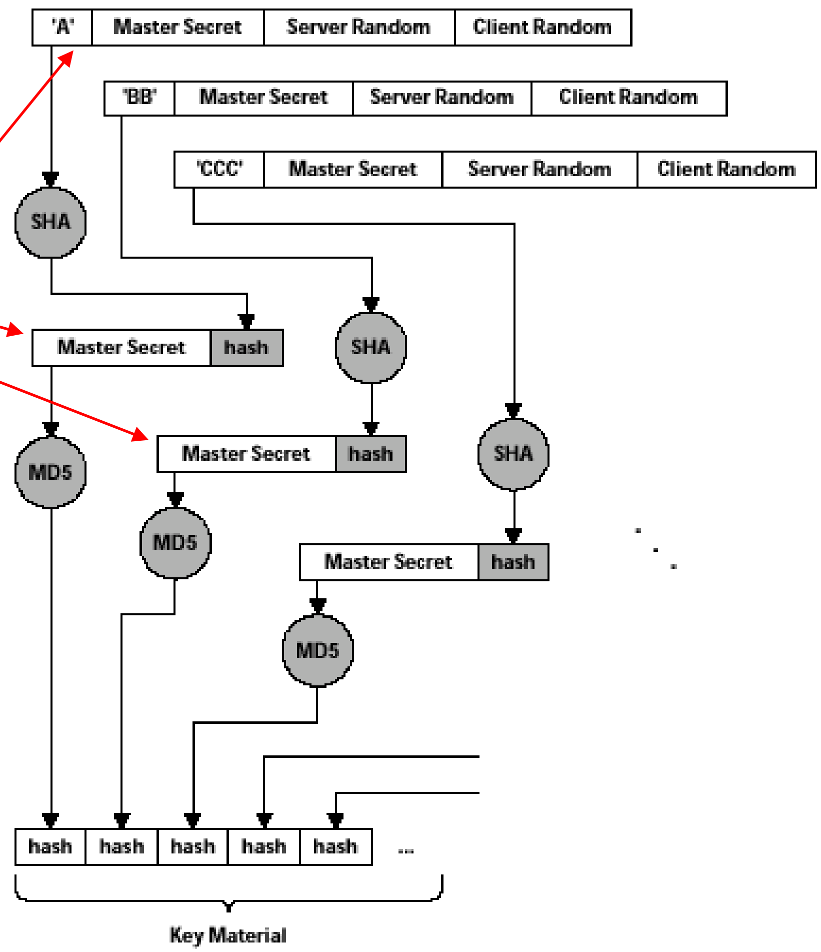
Phase 4

- **Client send Change Cipher Spec**
  - This message is removed in TLS 1.3
- **Client sends handshake finished**
- **Server send Change Cipher Spec**
  - Again this message is removed in TLS 1.3
- **Server Handshake Finished – First encryption**
  - Server now has the shared key using Elliptic Curve Diffie Hellman
  - First message encrypted with new crypto parameters
    - ✓ Digest of negotiated master secret, the ensemble of handshake messages, sender constant

SSL/TLS Keys Generation

- **Three steps performed at both client and server:**
- **1- Generate a Master secret, K**
  - Generated by both parties
    - ✓ server random RA (from Server Hello)
    - ✓ client random RB (from Client Hello)
    - ✓ Server/client public/private keys
  - This Master secret K will be a function of these.
- **2- Generate Key material from Master Secret**
  - Generated from the master secret K and shared random values RA and RB
    - ✓ Similar process as in the generation of K
- **3- Generate various keys from the Key Material**
  - For each connection, 6 keys are generated.
  - 3 keys for each direction: encryption, authentication/integrity, IV
    - ✓ Client encrypt vs server encrypt keys

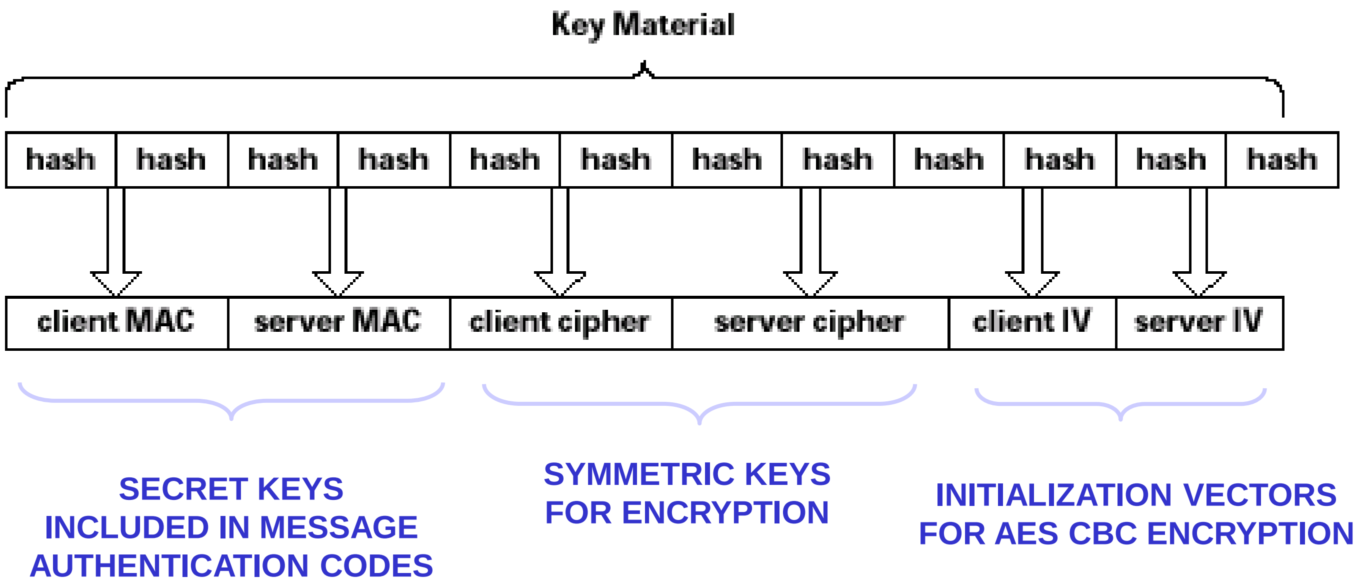
Generation of Key Material



THE MASTER SECRET K IS USED HERE

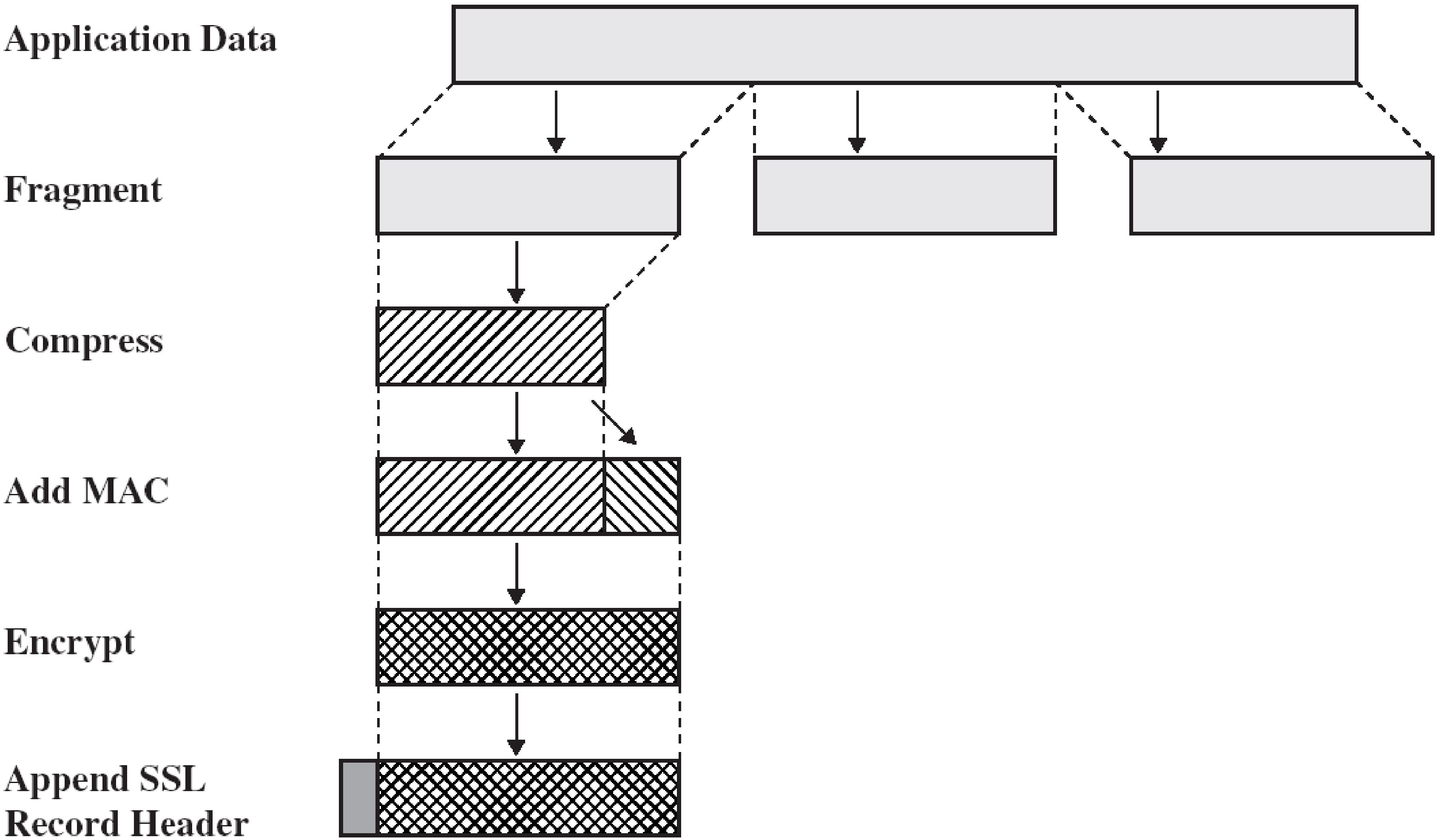
SOURCE: THOMAS, *SSL AND TLS ESSENTIALS*

Obtaining Keys from the Key Material



SOURCE: THOMAS, *SSL AND TLS ESSENTIALS*

SSL/TLS Record Protocol



Record Header

- **Three pieces of information**
  - Content type: Application data, Alert, Handshake, Change_cipher_spec
  - Content length
    - Suggests when to start processing
  - SSL/TLS version
    - Redundant check for version agreement
- **Max. record length: $2^{14} - 1$**
- **MAC**
  - Data
  - Headers
  - Sequence number
    - To prevent replay and reordering attack
    - Not included in the record

SSL/TLS Overhead

- **2-10 times slower than a TCP session**
- **Where do we lose time?**
  - Handshake phase
    - Client and servers use public-key cryptography for signature verification as well as key computation
    - Usually clients have to wait on servers to finish
  - Data Transfer phase
    - MAC computation
    - Symmetric key encryption

TLS Advantages

- **TLS is the recommended security mechanism specified in RFC 3261 by the IETF.**
- **End-to-end security, best for connection-oriented sessions**
- **TLS is implemented at the application level instead of the kernel level**
  - OS does not have to be changed
  - It is deployed via a browser and does not require specialized client software.
- **Easy to modify applications to use TLS**
  - Used to secure http sessions (HTTPS)
- **Provides user authentication instead of data-origin authentication**
  - If user sends own Client Certificates
- **Preferred in e-commerce solutions such as online banking**
- **De facto standard for WPA2 wireless security authentication (EAP-TLS)**

TLS Disadvantages

- **Both of the TLS models require the server and client to support PKI features, such as certificate validation and certificate management.**
  - Not all clients and solutions support PKI.
- **PKI is computationally expensive since it uses public key cryptography**
- **TCP and TLS pose significant memory consumption and scaling issues when you have tens of thousands of TCP connections.**
  - Runs on top of TCP only (connection-oriented).
  - There is a subset version of TLS that is supported for use with UDP called DTLS (RFC 4347)
- **In Server-Side Authentication, only one end is authenticated**

When to Use TLS

- **Every web browser has the TLS protocol built into it.**
  - There is no configuration required to utilize it.
  - Therefore to use a secure HTTP session or an TLS VPN requires no additional client software to install.
- **Ease of use with little maintenance required is what makes TLS extremely popular.**
- **One of the benefits of an TLS VPN is that it allows for granular controls.**
  - If I wanted to provide a VPN for a particular application and not the entire network, I can achieve this using a TLS VPN.