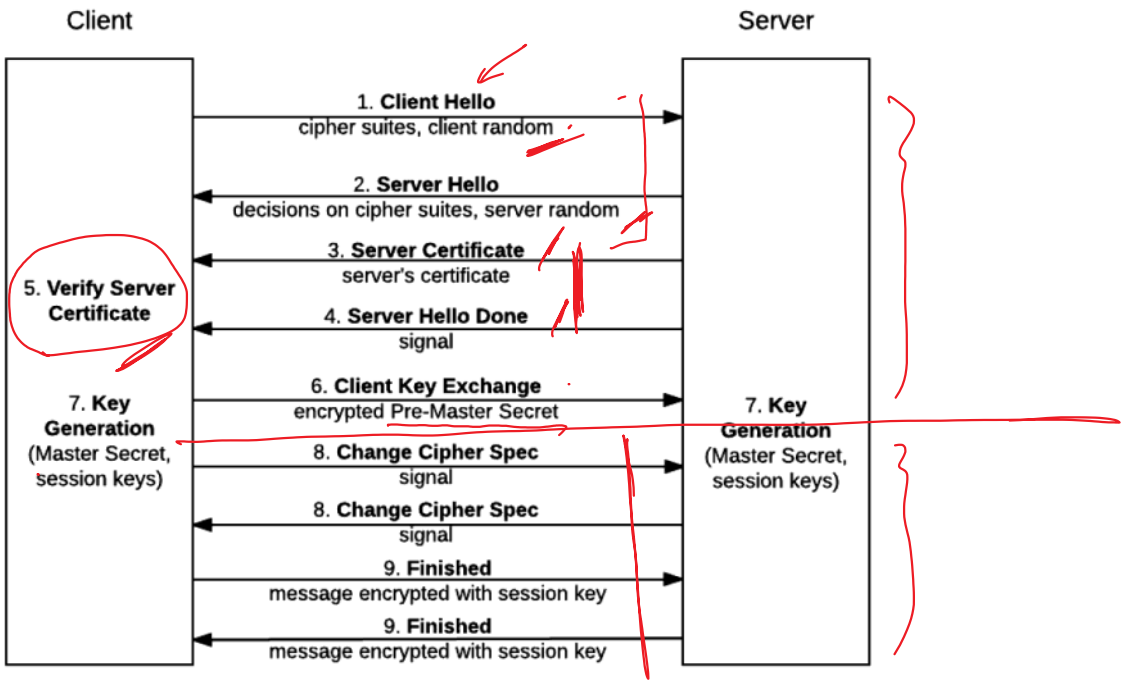
Internet Security

HTTS

HTTP
+
TLS

TLS/SSL Protocol

TLS/SSL: The Handshake Protocol



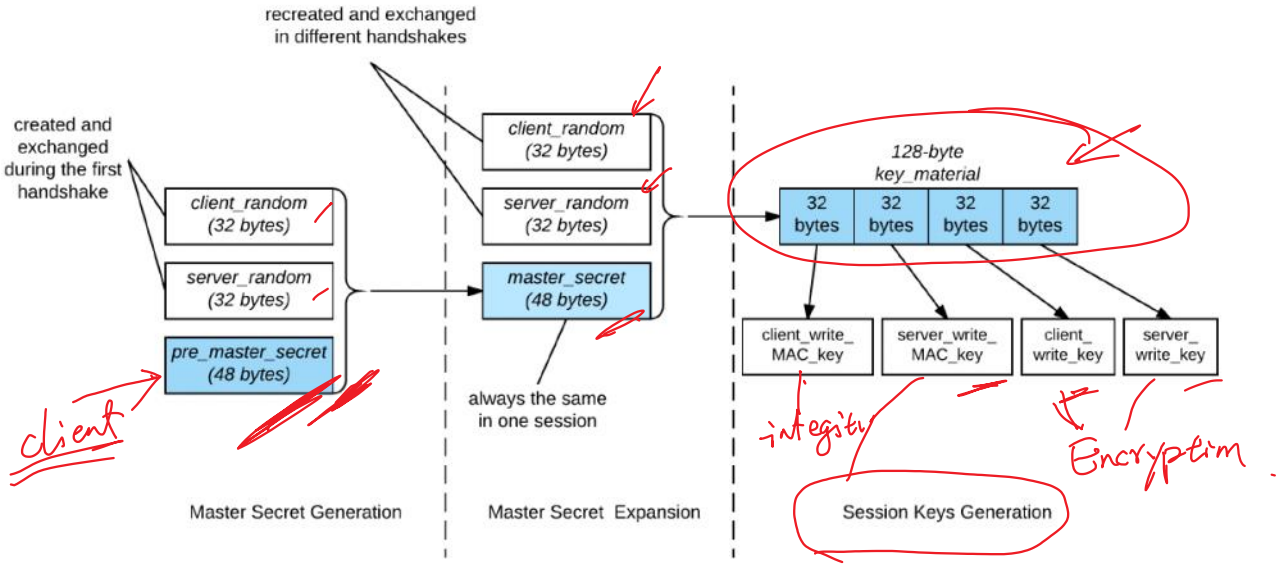
Key

TLS/SSL: Verifying Certificates

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            2c:d1:95:10:54:37:d0:de:4a:39:20:05:6a:f6:c2:7f
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=US, O=Symantec Corporation, OU=Symantec Trust Network,
                CN=Symantec Class 3 EV SSL CA - G3
        Validity
            Not Before: Feb  2 00:00:00 2016 GMT
            Not After : Oct 30 23:59:59 2017 GMT
        Subject: 1.3.6.1.4.1.311.60.2.1.3=US/
                 1.3.6.1.4.1.311.60.2.1.2=Delaware/
                 businessCategory=Private Organization/
                 serialNumber=3014267, C=US/
                 postalCode=95131-2021, ST=California, L=San Jose/
                 street=2211 N 1st St, O=PayPal, Inc., OU=CDN Support,
                 CN=www.paypal.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:da:43:c8:b3:a6:33:5d:83:c0:63:14:47:fd:6b:
                    22:bd:bf:4e:a7:43:11:55:eb:20:8b:e4:61:13:ee:
                    ......
                    00:c5:01:69:b5:10:16:a5:85:f8:fd:07:84:9a:c9:
                    14:91
                Exponent: 65537 (0x10001)
    Signature Algorithm: sha256WithRSAEncryption
         4b:a9:64:20:cc:77:0b:30:ab:69:50:d3:7f:de:dc:7c:e2:fb:
         93:84:fd:78:a7:06:e8:14:03:99:c0:e4:4a:ef:c3:5d:15:2a:
         ...
         7d:6a:de:cb:9f:ff:ef:8c:65:35:e4:22:b5:88:b2:48:32:1e:
         a4:71:a7:9e
```

- CN = user's intent (not automatic)
- Signature. auto
- Expiration date auto

Key Exchange

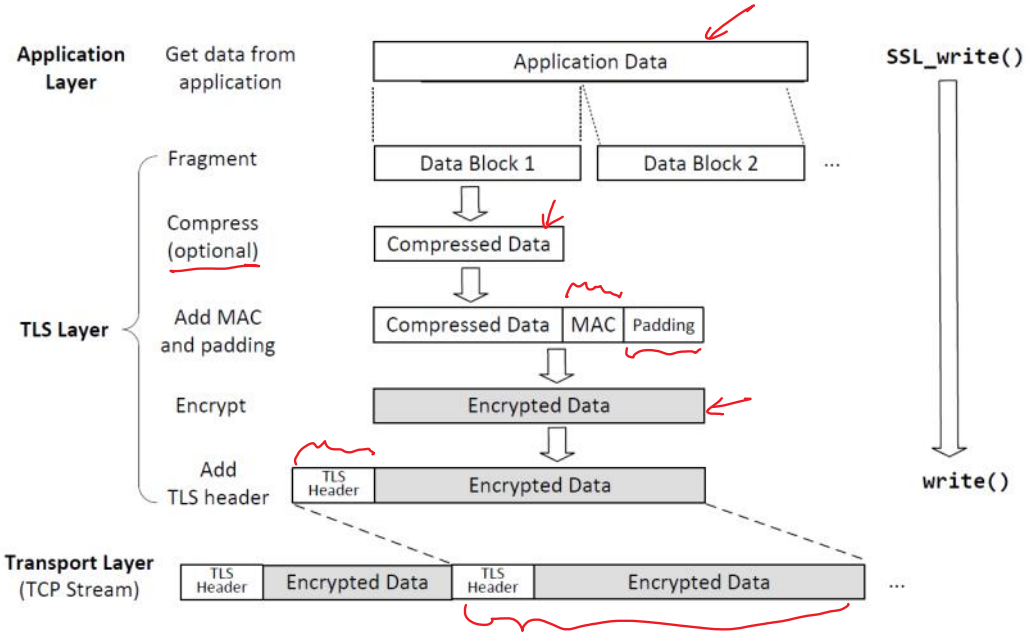


MAC

MAC

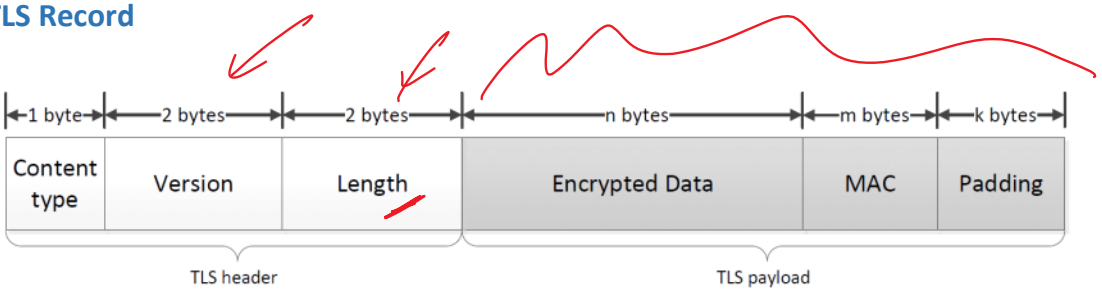
HMAC

TLS/SSL: Data Sending



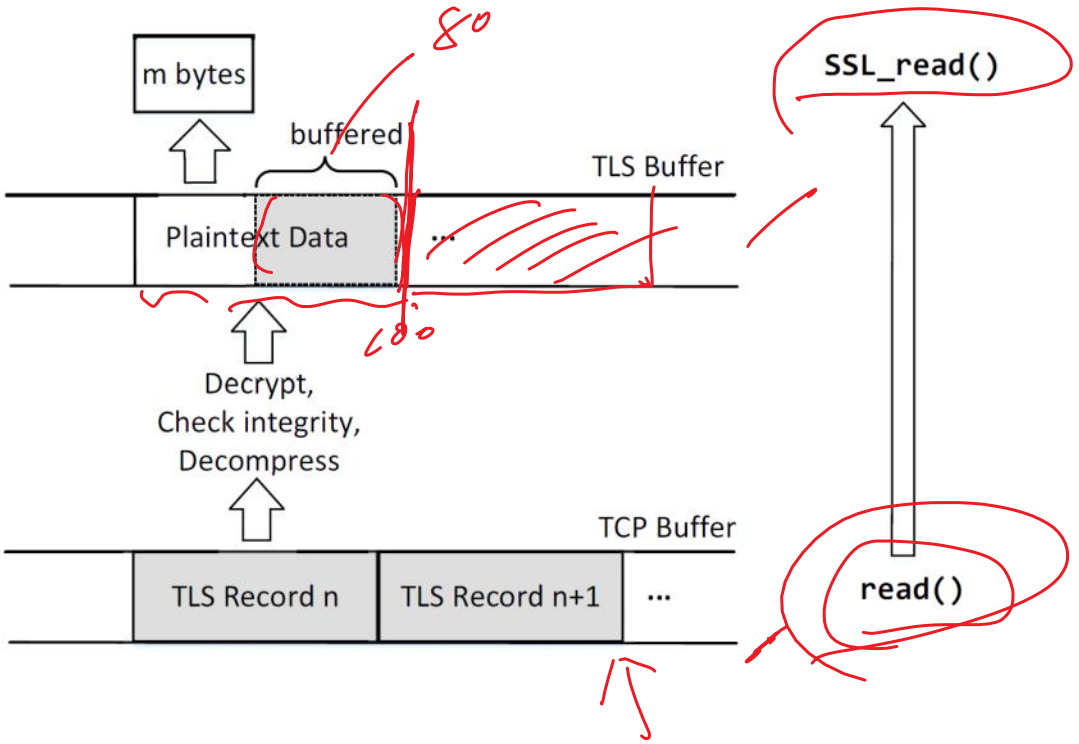
TLS
|
TCP

TLS Record



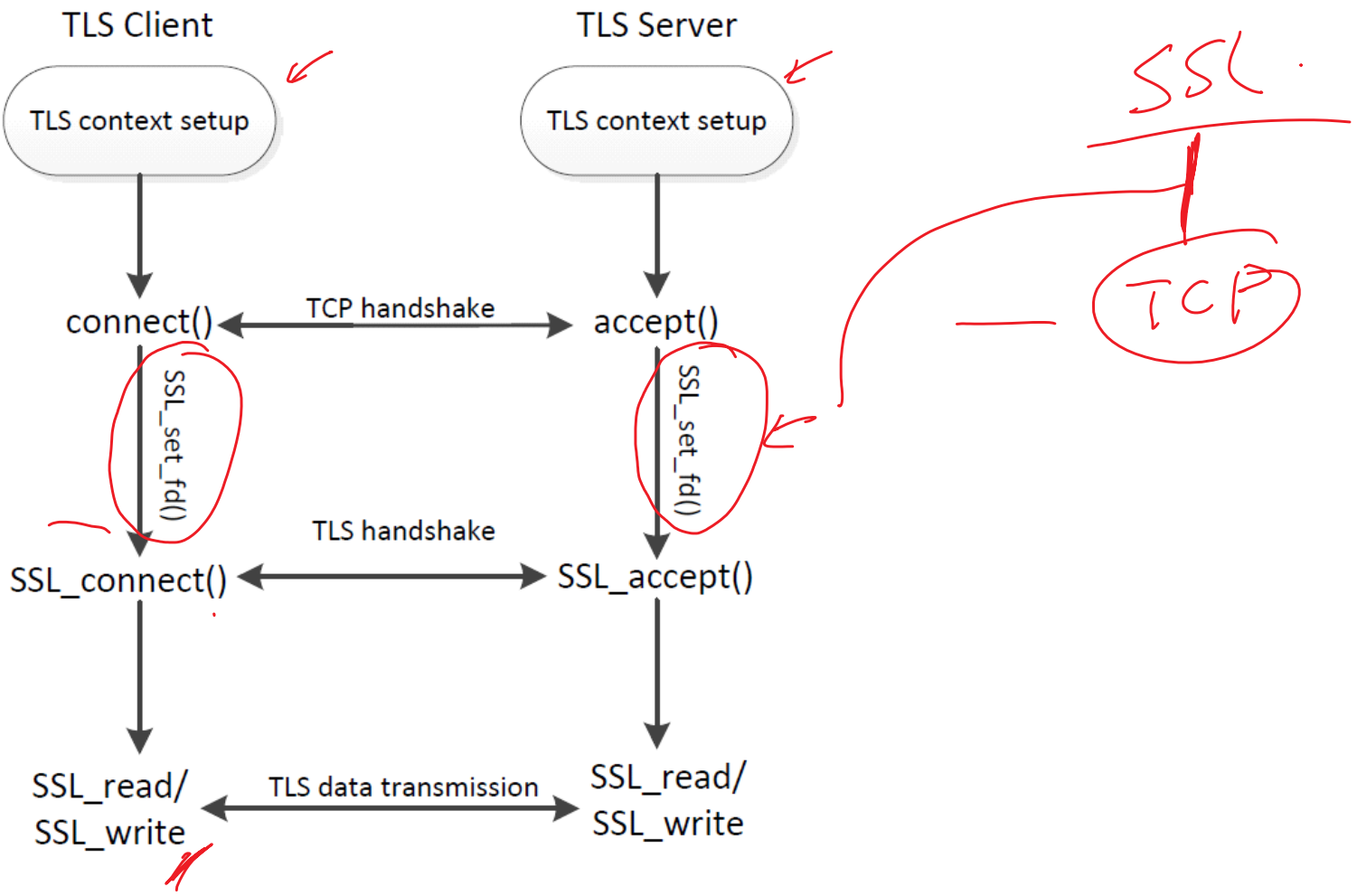
TLS/SSL: Data Receiving

100



TLS
↓
TCP

TLS Programming: Overview



TLS Setup

❖ Client

```
SSL* setupTLSClient(const char* hostname)
{
  // Step 0: OpenSSL library initialization
  // This step is no longer needed as of version 1.1.0.
  SSL_library_init();
  SSL_load_error_strings();

  // Step 1: SSL context initialization
  SSL_METHOD *meth = (SSL_METHOD *)TLSv1_2_method();
  SSL_CTX* ctx = SSL_CTX_new(meth);
  SSL_CTX_set_verify(ctx, SSL_VERIFY_PEER, NULL);
  SSL_CTX_load_verify_locations(ctx, NULL, "./cert");

  // Step 2: Create a new SSL structure for a connection
  SSL* ssl = SSL_new (ctx);

  // Step 3: Enable the hostname check
  X509_VERIFY_PARAM *vpm = SSL_get0_param(ssl);
  X509_VERIFY_PARAM_set1_host(vpm, hostname, 0);

  return ssl;
}
```

verify server cert

Trusted CA certificates

❖ Server

```
SSL* setupTLSServer()
{
    SSL_METHOD *meth;
    SSL_CTX* ctx;
    SSL* ssl;
    int err;

    // Step 1: SSL context initialization
    meth = (SSL_METHOD *)TLSv1_2_method();
    ctx = SSL_CTX_new(meth);
    SSL_CTX_set_verify(ctx, SSL_VERIFY_NONE, NULL);

    // Step 2: Set up the server certificate and private key
    SSL_CTX_use_certificate_file(ctx, "./bank_cert.pem",
                                 SSL_FILETYPE_PEM);
    /* SSL_CTX_use_certificate_chain_file(ctx,
                                 "./bank_chain_cert.pem"); */
    SSL_CTX_use_PrivateKey_file(ctx, "./bank_key.pem",
                                 SSL_FILETYPE_PEM);

   // Step 3: Create a new SSL structure for a connection
    ssl = SSL_new (ctx);

    return ssl;
}
```

server cert

private

TLS Handshake and Data Transmission

❖ Handshake

Client

```
SSL_set_fd(ssl, sockfd);
int err = SSL_connect(ssl);
```

Server

```
int sock = accept(listen_sock, (struct sockaddr*)&sa_client,
&client_len);
if (fork() == 0) { // The child process
   close (listen_sock);

   SSL_set_fd (ssl, sock);
   int err = SSL_accept (ssl);
```

❖ Sending/Receiving Data

```
char buf[9000];
char sendBuf[200];

sprintf(sendBuf, "GET / HTTP/1.1\nHost: %s\n\n", hostname);
SSL_write(ssl, sendBuf, strlen(sendBuf));

int len;
do {
     len = SSL_read (ssl, buf, sizeof(buf) - 1);
     buf[len] = '\0';
     printf("%s\n",buf);
} while (len > 0);
```

SSL_write → write( )
TCP

Man-In-The-Middle Attack

What is the problem of the following code?

```
SSL* setupTLSClient(const char* hostname)
{
   // Step 0: OpenSSL library initialization
   // This step is no longer needed as of version 1.1.0.
   SSL_library_init();
   SSL_load_error_strings();

   // Step 1: SSL context initialization
   SSL_METHOD *meth = (SSL_METHOD *)TLSv1_2_method();
   SSL_CTX* ctx = SSL_CTX_new(meth);
   SSL_CTX_set_verify(ctx, SSL_VERIFY_PEER, NULL);
   SSL_CTX_load_verify_locations(ctx, NULL, "./cert");

   // Step 2: Create a new SSL structure for a connection
   SSL* ssl = SSL_new (ctx);

   return ssl;
}
```

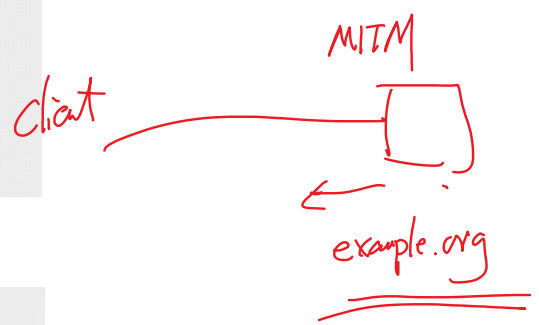
MITM Experiment

❖ The TLS Setup

```
SSL* setupTLSClient(const char* hostname)
{
   SSL_METHOD *meth;
   SSL_CTX* ctx;
   SSL* ssl;

   meth = (SSL_METHOD *)TLSv1_2_method();
   ctx = SSL_CTX_new(meth);

   SSL_CTX_set_verify(ctx, SSL_VERIFY_PEER, verify_callback); ❶
   SSL_CTX_load_verify_locations(ctx, NULL, "./cert");
   ssl = SSL_new (ctx);

   // Enable the hostname check
   X509_VERIFY_PARAM *vpm = SSL_get0_param(ssl);  ❷
   X509_VERIFY_PARAM_set1_host(vpm, hostname, 0); ❸

   return ssl;
}
```

❖ The Callback function

```
int verify_callback(int preverify_ok, X509_STORE_CTX *x509_ctx)
{
    char  buf[300];

    X509* cert = X509_STORE_CTX_get_current_cert(x509_ctx); ❹
    X509_NAME_oneline(X509_get_subject_name(cert), buf, 300);
    printf("subject= %s\n", buf);      ❺

    if (preverify_ok == 1) {
       printf("Verification passed.\n");
    } else {
       int err = X509_STORE_CTX_get_error(x509_ctx);
       printf("Verification failed: %s.\n",
          X509_verify_cert_error_string(err)); ❻
    }
}
```



❖ Launch the Attack (no hostname check)

```
$ client www.facebook.com 443
subject= ... /CN=DigiCert High Assurance EV Root CA
Verification passed.
subject= ... /CN=DigiCert SHA2 High Assurance Server CA
Verification passed.
subject= ... /CN=www.example.org
Verification passed.
SSL connection is successful
SSL connection using ECDHE-RSA-AES128-GCM-SHA256
```

❖ Launch the Attack Again (with hostname check)

```
Assigned Names and
Numbers/OU=Technology/CN=www.example.org
Verification failed:  Hostname mismatch.  ❶
subject= ... /CN=DigiCert High Assurance EV Root CA
Verification passed.
```

```
Verification failed:  Hostname mismatch.  ❶
subject= ... /CN=DigiCert High Assurance EV Root CA
Verification passed.
subject= ... /CN=DigiCert SHA2 High Assurance Server CA
Verification passed.
subject= ... /CN=www.example.org
Verification passed.                                  ❷
SSL connection is successful                          ❸
SSL connection using ECDHE-RSA-AES128-GCM-SHA256
```

Summary

- TLS Handshake
- Key Exchange
- TLS Data Transmission
- TLS Programming
- TLS Client and Server Programming
- MITM attacks and hostname checks