北京郵電大學
BEIJING UNIVERSITY OF POSTS AND TELECOMMUNICATIONS

传输层安全协议

BUPT

一、 TLS概述

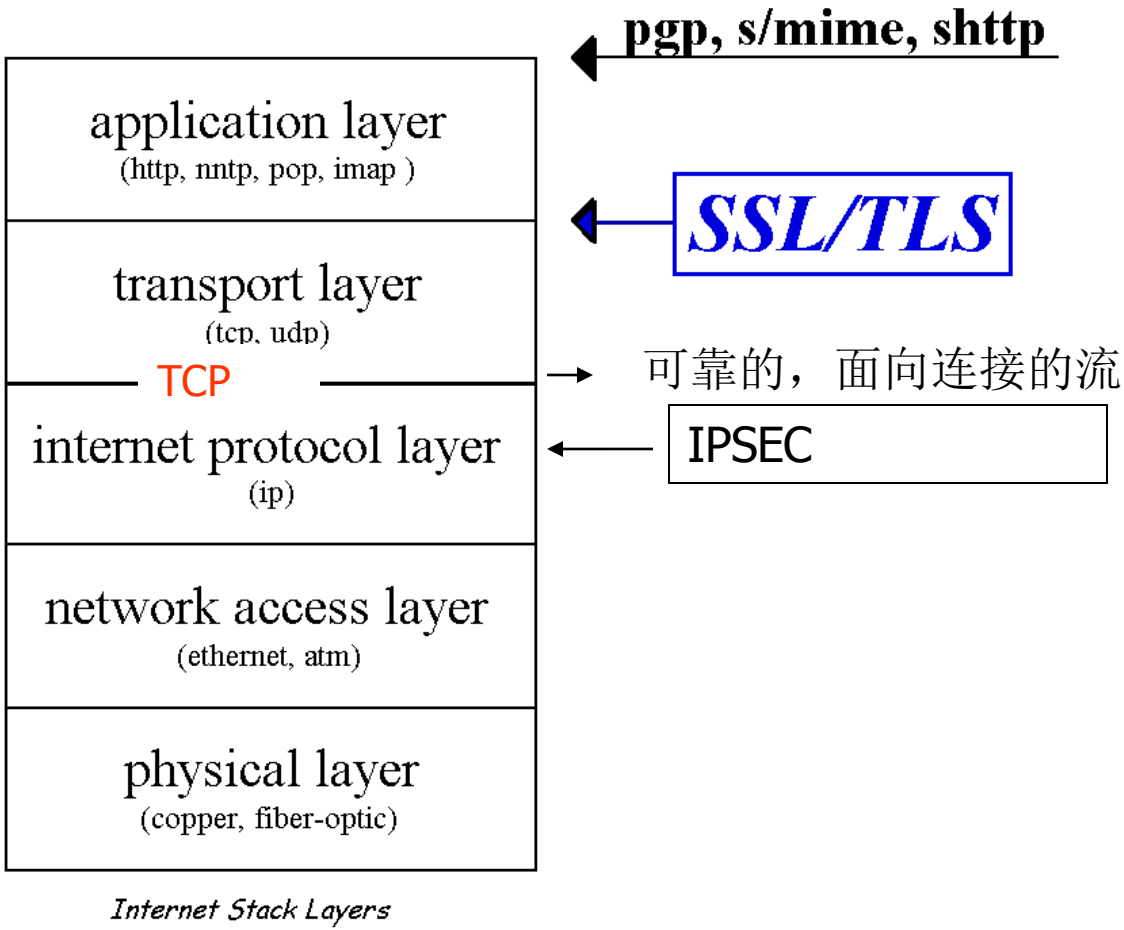
传输层的安全协议

- 传输层的安全协议可在传输层上提供保密、认证和完整性检验功能。
- 目前主要的传输层上的安全协议有
  - SSL
  - TLS
  - WTLS

SSL

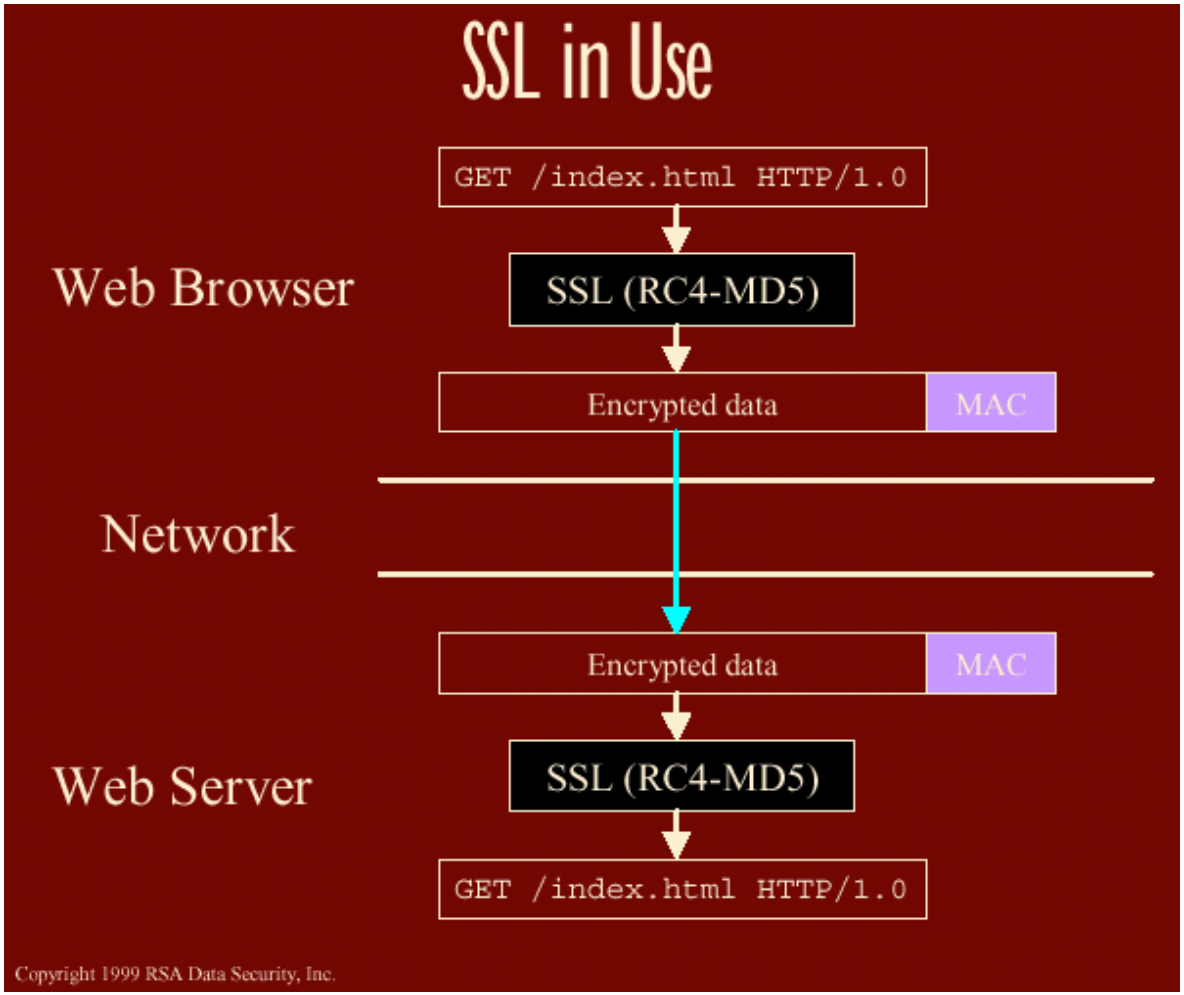
- SSL (Secure Sockets Layer)安全套接字层
  - 网景公司（Netscape）推出的基于Web应用的安全协议。
- Client/server结构, SSL 客户端嵌入 Web Browser
- SSL 采用PKI (Public Key Infrastructure), 提供 confidentiality, integrity, authentication and non-repudiation

What layer?


pgp, s/mime, shttp
application layer
(http, nntp, pop, imap)
SSL/TLS
transport layer
(tcp, udp)
TCP
可靠的，面向连接的流
internet protocol layer
(ip)
IPSEC
network access layer
(ethernet, atm)
physical layer
(copper, fiber-optic)
Internet Stack Layers

SSL 应用

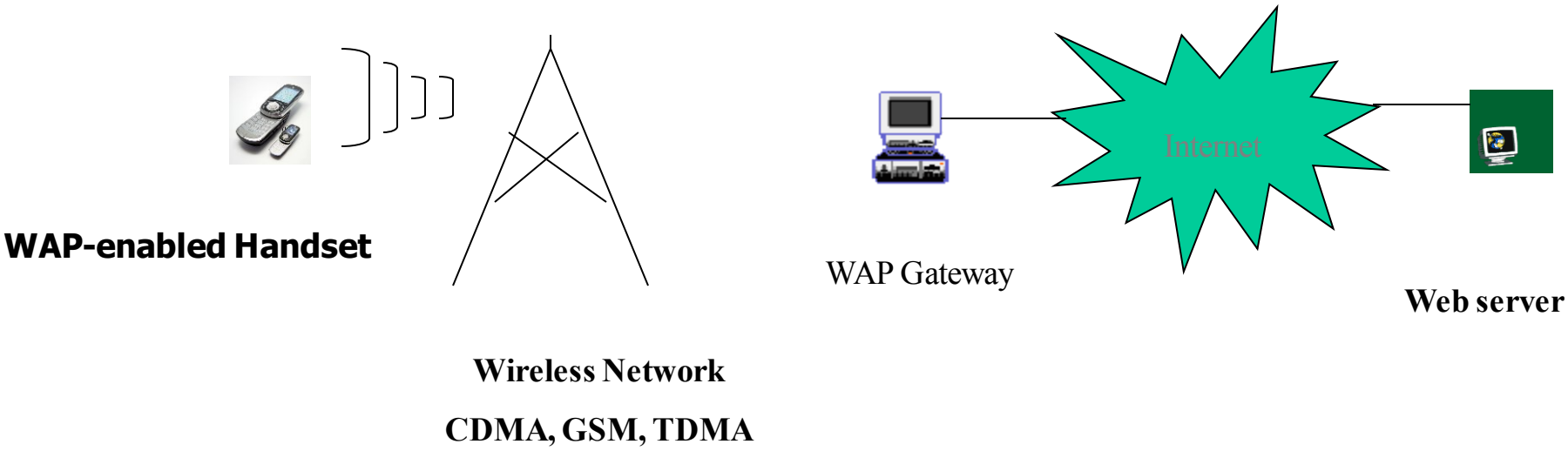
广泛用于HTTP连接



SSL演进，SSL与TLS

- SSLv2 (Secure Socket Layer)
  - Netscape, 1994
- PCT (Private Communications Technology)
  - Microsoft, 1995
  - Compatible with SSLv2
- SSLv3
  - Netscape, 1996
- TLSv1 (Transport Layer Security) RFC2246,1999
  - Minor changes with SSLv3, may be viewed as SSLv3.1
  - do not interoperate with SSL3.0
- TLSv1.1 RFC4346，2006
- TLS v1.2 RFC 5246, 2008

SSL、TLS与WTLS



wap论坛在TLS的基础上做了简化，提出了WTLS协议（Wireless Transport Layer Security）

SSL提供的安全服务

- 认证(Authentication)：
  - 客户对服务器的身份认证
    - SSL服务器允许客户的浏览器使用标准的公钥技术和一些可靠的认证中心（CA）的证书，来确认服务器的合法性。
  - 服务器对客户的身份认证
    - 也可通过公钥技术和证书进行认证，也可通过用户名，password来认证。

SSL提供的安全服务

- 建立服务器与客户之间安全的数据通道
  - 传输数据的机密性
    - DES, Triple DES, IDEA, RC2, RC4, …
  - 传输数据的完整性(integrity)
    - 基于散列的消息认证校验和（Hash-based Message Authentication Checksum (HMAC)）
    - MD5 or SHA-1
- 密钥交互

SSL 不能做什么?

- SSL只保证在Internet上的传送的安全，一旦到达对端后，内容就以明文存在。例如，以SSL传送信用卡号，server端可以知道其内容
  - SET才可以保证server端的商家无法得到卡号

TLS上的HTTP

- 原来“https”表示在443端口的SSL之上的HTTP协议。类似的，其它应用协议（例如：snews，ftps）为区分安全和不安全的使用，要求使用两个端口号
- 在1997年12月华盛顿特区IETF会议上，IESG重申不赞成使用并行的“安全”端口号。
- RFC2817：Upgrading to TLS Within HTTP/1.1 May 2000

HTTP/1.1的 Upgrade机制可以在一个打开的HTTP连接上建立传输层安全

- 客户请求升级到TLS之上的HTTP ： 发送一个包含记号“TLS/1.0”的升级包头的HTTP/1.1请求：

GET http://example.bank.com/acct_stat.html?749394889300 HTTP/1.1
Host: example.bank.com
Upgrade: TLS/1.0
Connection: Upgrade

- 服务器对升级请求的确认：

HTTP/1.1 101 Switching Protocols
Upgrade: TLS/1.0, HTTP/1.1
Connection: Upgrade

二、TLS加密

TLS中的算法

数据传输加密算法：

使用对称加密算法来加密会话消息：

① 没有加密(No Encryption)

② 流密码(Stream Ciphers)

- RC4(40比特密钥), RC4(128比特密钥)

③ CBC (Cipher Block Chaining)分组密码

- RC2(40 bit密钥)
- DES40, DES, 3DES

.

消息摘要方法：

TLS支持的HMAC，消息摘要函数包括：

- 没有消息摘要(Null Choice)
- 必须支持：
  - MD5，128比特哈希：HMAC_MD5(secret, data)
  - 安全哈希算法(SHA－1)，160比特哈希，HMAC_SHA(secret, data)
- 或双方协商的哈希算法

密钥交换方法：

TLS定义了如何得到加密客户和服务器之间传输的应用数据的对称加密密钥。

- 使用数字证书的RSA密钥交换
- 无数字证书的Diffie-Hellman密钥交换

- 必须支持：
  TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA

三、 TLS协议

TLS协议结构

- 设计目标：
  - 安全性：密码学意义上的安全
  - 兼容性：Internet上B/S的计算模型
  - 扩展性
    - 传输层：高层协议的无关性
    - 可加入新的算法：提供框架
  - 效率

TLS协议结构

- RFC2246
  - The TLS Protocol V1.0 January 1999

TLS握手协议	TLS告警协议	TLS改变密码规范协议	TLS应用数据协议
TLS记录协议			
TCP			
IP			

TLS协议结构

- TLS记录协议(record protocol)，位于可靠的传输协议（例如TCP）上面。
- 使用TLS记录协议的上层为握手协议，包括：
  - 握手协议（Handshake Protocol）：建立客户与服务器之间的安全通道，包括双方的相互认证，交换密钥参数
  - 告警协议（Alert Protocol）：向对端指示其安全错误
  - 修改密码规格协议（Change Cipher Spec Protocol）：改变密码参数
- 记录协议上还可封装应用数据协议（Application Data Protocol）（记录类型20=改变密码规格，21=告警，22=握手，23=应用层数据）
- 为了便于TLS协议的扩展，记录协议可以支持更多的上层协议。

TLS记录协议

- 建立在可靠的传输协议(如TCP)之上
- 高层应用无关：用于封装各种高层协议
- 两个基本安全特性：
  - 机密性：使用DES、RC4等对称加密算法进行数据加密。对称加密所产生的密钥对每个连接都是唯一的，且此密钥基于另一个协议（如握手协议）协商。记录协议也可以不加密使用。
  - 完整性：在信息传输过程中，使用MAC（Message authentication Code，消息认证代码）进行完整性检查。MAC计算使用了安全哈希功能（SHA、MD5等）。TLS记录协议也可以在不使用MAC的情况下操作，但这种情况一般只能用于以下模式：有另一个协议正在使用TLS记录协议传输安全参数的协商。

报头：

- 内容类型(8位)：所封装分段的高层协议类型。
- 主版本(8位)：使用SSL协议的主要版本号。（3）
- 次版本(8位)：使用SSL协议的次要版本号。（1）
- 压缩长度(16位)：分段的字节长度,小于2^14+1024字节。

TLS连接状态

- 基于状态的协议：
  - 4种状态：
    - 当前读，写状态
    - Pending 读，写状态
  - 包括：
    - compression state
    - cipher state
    - MAC secret
    - sequence number：初始为0,小于2^64-1，每个记录加1

TLS 连接安全参数

```
struct {
        ConnectionEnd          entity;
        BulkCipherAlgorithm    bulk_cipher_algorithm;
        CipherType             cipher_type;
        uint8                  key_size;
        uint8                  key_material_length;
        IsExportable           is_exportable;
        MACAlgorithm           mac_algorithm;
        uint8                  hash_size;
        CompressionMethod      compression_algorithm;
        opaque                 master_secret[48];
        opaque                 client_random[32];
        opaque                 server_random[32];
    } SecurityParameters;
```

用于生成**master secret**

其中

enum { server, client } ConnectionEnd;

enum { null, rc4, rc2, des, 3des, des40 } BulkCipherAlgorithm;

enum { stream, block } CipherType;

enum { true, false } IsExportable;

enum { null, md5, sha } MACAlgorithm;

enum { null(0), (255) } CompressionMethod;

- 问题：
  - Key如何得到
    - 例：3DES_EDE_CBC_SHA 需**104 bytes**：
      - 2 x 24 byte keys
      - 2 x 20 byte MAC secrets,
      - 2 x 8 byte IVs

定义：

P_hash(secret, seed) = HMAC_hash(secret, A(1) + seed) +
HMAC_hash(secret, A(2) + seed) +
HMAC_hash(secret, A(3) + seed) + ...

A(0) = seed, A(i) = HMAC_hash(secret, A(i-1))

不同**seed**

PRF(secret, label, seed) = P_MD5(S1, label + seed)
XOR P_SHA-1(S2, label + seed);

S1 :first half of the secret, S2 : second half.

label: ASCII string.

hello中交互

master_secret = PRF(pre_master_secret, "master secret",
ClientHello.random + ServerHello.random) [0..47Byte];

Then pre_master_secret should be deleted from memory.

- 安全参数用于生成（共享密钥）：
  - client /server write MAC secret
  - client/ server write key
  - Client/ server write IV (block ciphers only)

key_block = PRF(SecurityParameters.master_secret,

"key expansion", SecurityParameters.server_random +

SecurityParameters.client_random); (non=exportable)

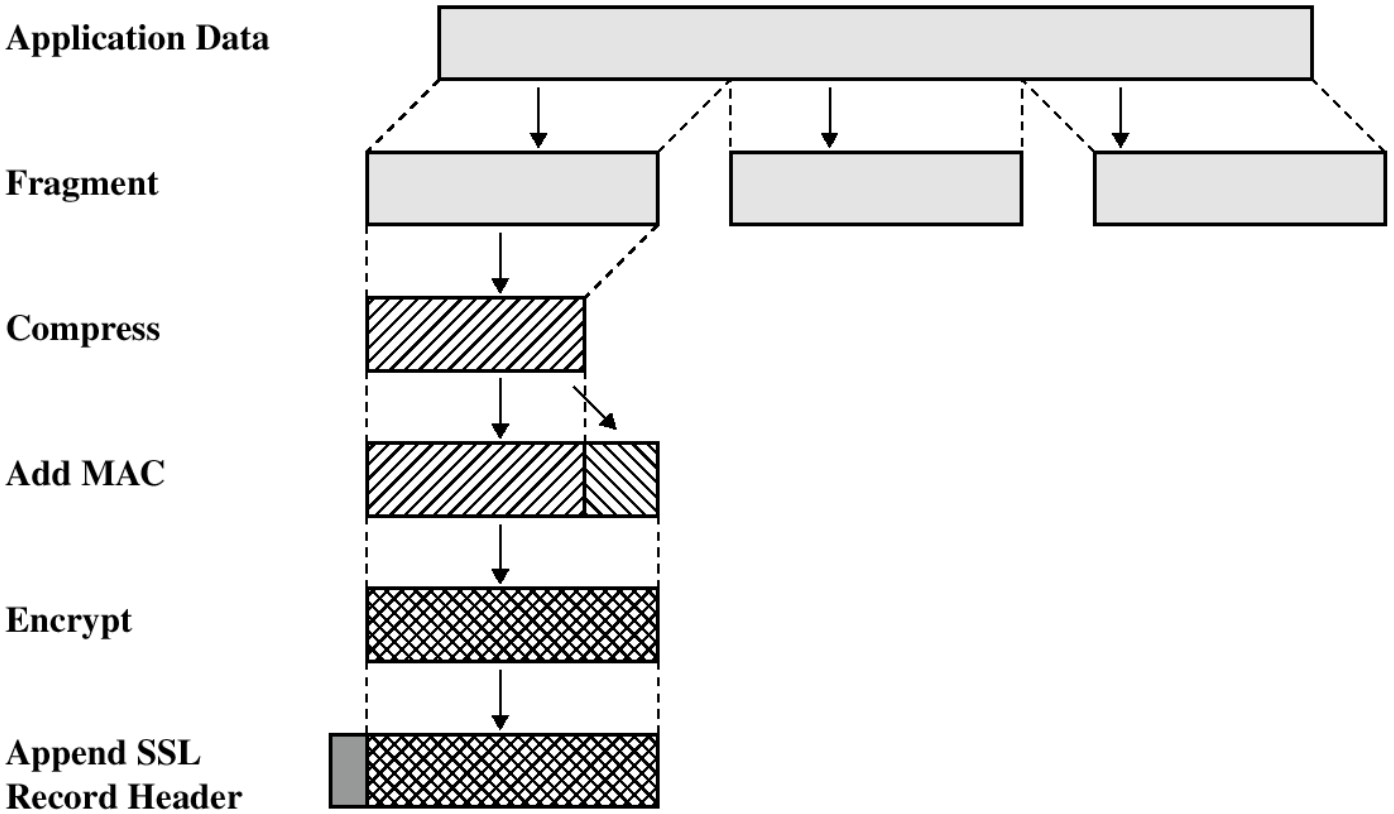
按顺序及各key长度将key_block划分成各个key

❖ Exportable: TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5 RFC中有例子

Windows TLS Cipher Suites

- TLS_RSA_WITH_RC4_128_MD5
- TLS_RSA_WITH_RC4_128_SHA
- TLS_RSA_WITH_3DES_EDE_CBC_SHA
- TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA
- TLS_RSA_WITH_DES_CBC_SHA
- TLS_DHE_DSS_WITH_DES_CBC_SHA
- TLS_RSA_EXPORT1024_WITH_RC4_56_SHA
- TLS_RSA_EXPORT1024_WITH_DES_CBC_SHA
- TLS_DHE_DSS_EXPORT1024_WITH_DES_CBC_SHA
- TLS_RSA_EXPORT_WITH_RC4_40_MD5
- TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5
- TLS_RSA_WITH_NULL_MD5
- TLS_RSA_WITH_NULL_SHA

TLS 记录协议工作过程



- 第一步 Fragment
  - Length : <2^14bytes （16384bytes）
- 第二步可选择是否进行压缩：
  - 没有指定压缩算法，标准为无压缩
  - 算法要求：必须是无损的,而且不会增加1024字节以上长度的内容。
  - 例：

RFC 3943 Transport Layer Security (TLS) Protocol Compression Using Lempel-Ziv-Stac (LZS)

第三步是给压缩后的数据计算消息验证码：

MAC使用下面公式进行计算：

HMAC_hash(MAC_write_secret, seq_num + TLSCompressed.type +

TLSCompressed.version + TLSCompressed.length + TLSCompressed.fragment));

MAC_write_secret为客户服务器之前生成的共享秘密；

seq_num为消息序列号；

hash为哈希算法；

TLSCompressed.type为处理分段的高层协议类型；

TLSCompressed.version 为 version

TLSCompressed.length为压缩分段的长度；

TLSCompressed.fragment为压缩分段(没有压缩时，就是明文分段)。

第四步，用对称加密算法给添加了MAC的压缩消息加密。

第五步，添加报头：

TLS握手协议

- 握手协议在任何应用程序数据传输之前使用。
- 协商算法
- 对客户端/服务器进行认证（可选）的协议。
- 为下一步记录协议要使用的密钥信息进行协商，使客户端和服务器建立并保持安全通信的状态信息。
- 为记录协议提供生成安全参数的必要值

TLS握手协议

- TLS握手协议提供的连接安全具有三个基本属性：
  - 使用非对称的（公钥）密钥算法（如RSA,DSS)来认证对方的身份（可选）。
  - 安全的共享加密密钥的协商，可抵御窃听和中间人攻击。
  - 可靠的协商，攻击者不可能在协商各方毫无察觉的情况下篡改协商消息。

RFC 2246中的交互过程

```
Client                                   Server

   ClientHello          -------->
                                         ServerHello
                                        Certificate*
                                    ServerKeyExchange*
                                   CertificateRequest*
                        <--------      ServerHelloDone
   Certificate*
   ClientKeyExchange
   CertificateVerify*
   [ChangeCipherSpec]
   Finished             -------->
                                   [ChangeCipherSpec]
                        <--------          Finished
```

*为可选

实际实现也可能分成几个包交互

A：我想和你安全的通话，我这里的对称加密算法有DES,RC5,密钥交换算法有RSA和DH，摘要算法有MD5和SHA。

B：我们用DES－RSA－SHA这对组合好了。

这是我的证书，里面有我的名字和公钥，你拿去验证一下我的身份（把证书发给A）。

A：（查看证书上B的名字是否无误，并通过手头早已有的CA的证书验证了B的证书的真实性，如果其中一项有误， 发出警告并断开连接，这一步保证了B的公钥的真实性）

（产生一份秘密消息，这份秘密消息处理后将用作加密密钥，加密初始化向量和hmac的密钥。将这份秘密消息

-称为pre_master_secret-用B的公钥加密，封装成称作ClientKeyExchange的消息。

我生成了一份秘密消息，并用你的公钥加密了，给你（把ClientKeyExchange发给B）

注意，下面我就要用加密的办法给你发消息了！

（将秘密消息进行处理，生成加密密钥，加密初始化向量和hmac的密钥）

B：（用自己的私钥将ClientKeyExchange中的秘密消息解密出来，然后将秘密消息进行处理，生成加密密钥，

加密初始化向量和hmac的密钥，这时双方已经安全的协商出一套加密办法了）

注意，我也要开始用加密的办法给你发消息了！

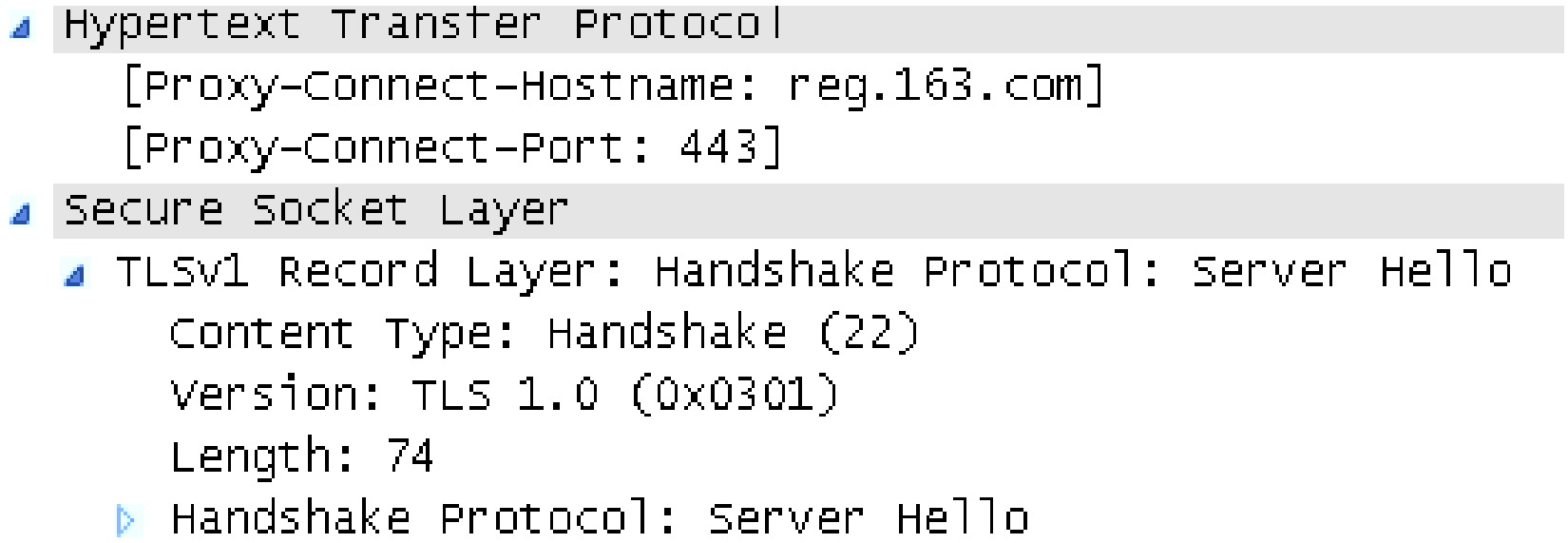
北京郵電大學
BEIJING UNIVERSITY OF POSTS AND TELECOMMUNICATIONS

实例：TLS握手这程

- 第一个包（client→server）： client hello。
- 客户端向服务器端的443端口发送client hello,请求建立TLS连接。

```
Hypertext Transfer Protocol
    [Proxy-Connect-Hostname: reg.163.com]
    [Proxy-Connect-Port: 443]
Secure Socket Layer
  TLSv1 Record Layer: Handshake Protocol: Client Hello
      Content Type: Handshake (22)
      Version: TLS 1.0 (0x0301)
      Length: 65
    Handshake Protocol: Client Hello
```

- 第二个包（server→client）：server hello



北京郵電大學
BEIJING UNIVERSITY OF POSTS AND TELECOMMUNICATIONS

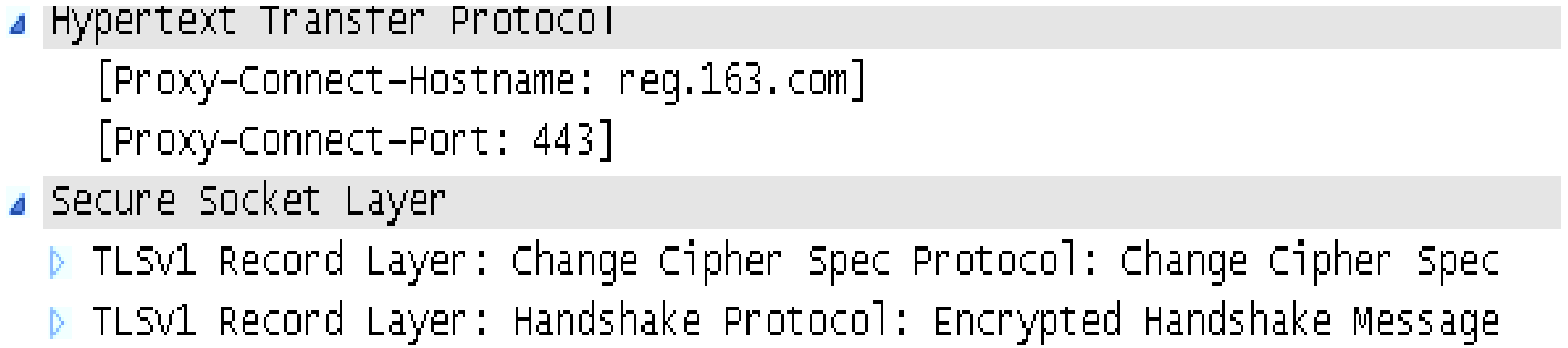
- 第三个包（server→client）certificate

```
Hypertext Transfer Protocol
    [Proxy-Connect-Hostname: reg.163.com]
    [Proxy-Connect-Port: 443]
Secure Socket Layer
  TLSv1 Record Layer: Handshake Protocol: Certificate
      Content Type: Handshake (22)
      Version: TLS 1.0 (0x0301)
      Length: 3172
    Handshake Protocol: Certificate
Hypertext Transfer Protocol
    [Proxy-Connect-Hostname: reg.163.com]
    [Proxy-Connect-Port: 443]
Secure Socket Layer
  TLSv1 Record Layer: Handshake Protocol: Server Hello Done
      Content Type: Handshake (22)
      Version: TLS 1.0 (0x0301)
      Length: 4
    Handshake Protocol: Server Hello Done
```

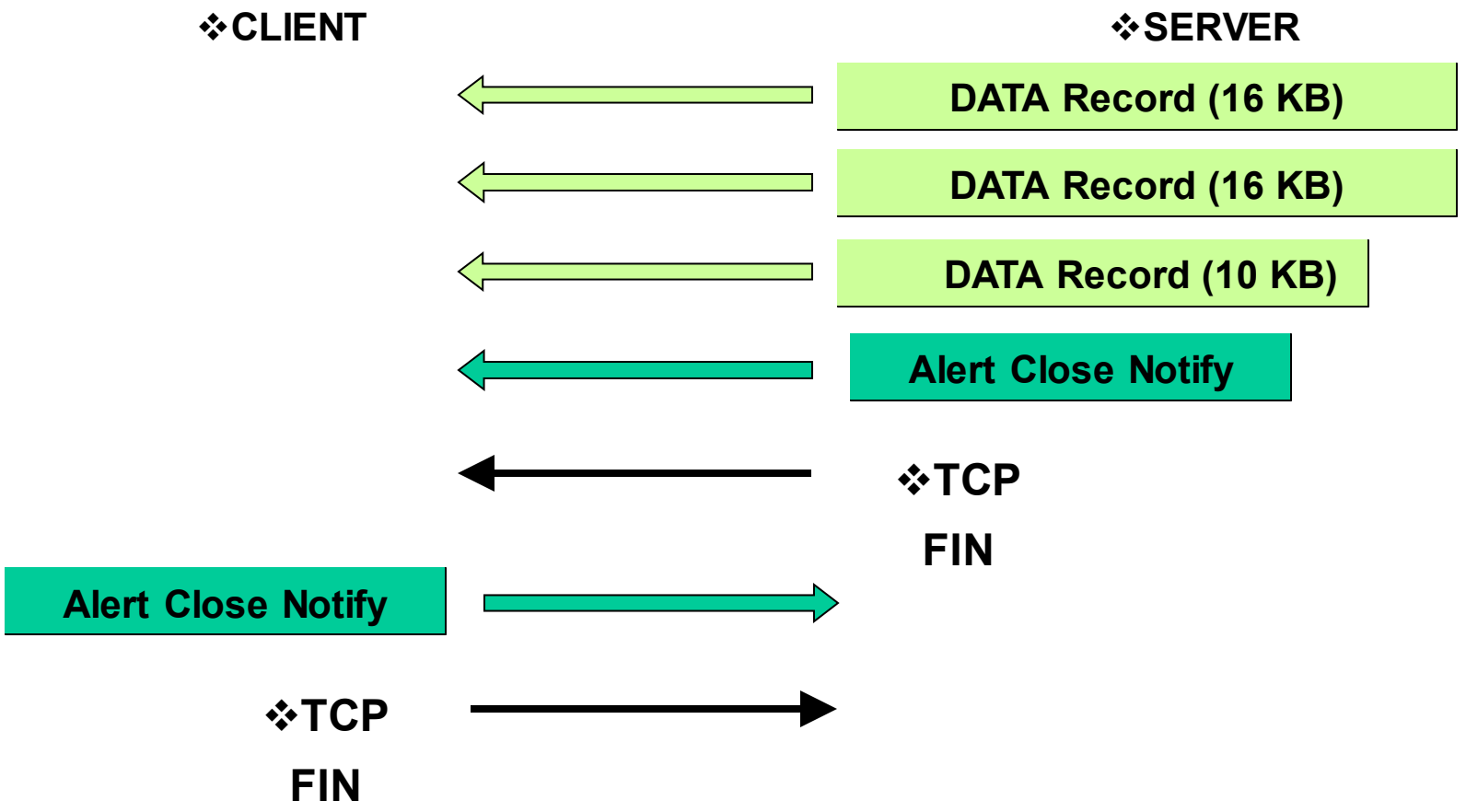
- 第四个包（client→server）：client key exchange，change cipher spec，encrypted handshake message.

```
Hypertext Transfer Protocol
  [Proxy-Connect-Hostname: reg.163.com]
  [Proxy-Connect-Port: 443]
Secure Socket Layer
  TLSv1 Record Layer: Handshake Protocol: Client Key Exchange
  TLSv1 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
  TLSv1 Record Layer: Handshake Protocol: Encrypted Handshake Message
```

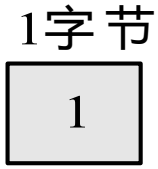
- 第五个包（server→client）：change cipher spec，encrypted handshake message。



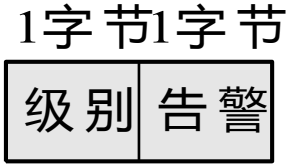
SSL/TLS 连接拆除


❖CLIENT
❖SERVER
DATA Record (16 KB)
DATA Record (16 KB)
DATA Record (10 KB)
Alert Close Notify
❖TCP
FIN
Alert Close Notify
❖TCP
FIN

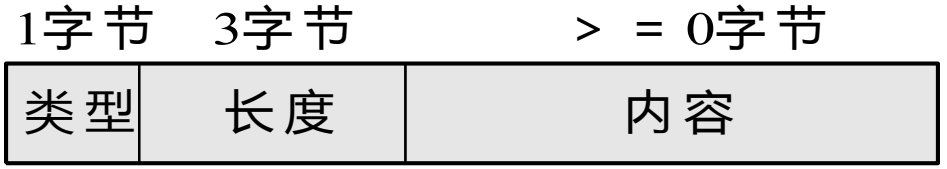
TLS握手协议格式



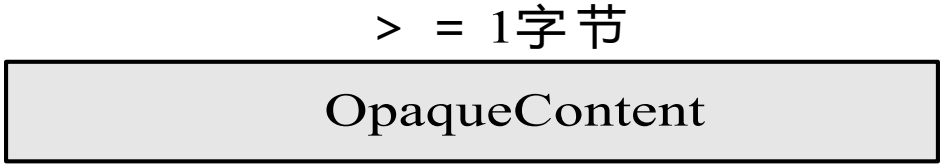
(a) 改变密码规范协议



(b) 告警协议



(c) 握手协议



(d) 其它上层协议

TLS告警协议

- TLS告警协议是用来为对等实体传递TLS的相关警告。如果在通信过程中某一方发现任何异常，就需要给对方发送一条警示消息通告。
- 警示消息有两种：
  - 一种是致命（Fatal）错误，如传递数据过程中，发现错误的MAC，双方就需要立即中断连接，同时消除自己缓冲区相应的连接记录；
  - 第二种是警告（Warning）消息，通信双方只是记录日志，而对通信过程不造成任何影响。

告警号	告警名称	含　义
0	Close_notify	通知接收方发送方在本连接中不会再发送任何消息
10	Unexpected_message	接收到不适当的消息
20	Bad_record_mac	接收到的记录MAC错误(致命)
30	Decompression_failure	解压缩失败(致命)
40	Handshake_failure	发送方无法成功协调一组满意的安全参数设置(致命)
42	Bad_certificate	证书已经破坏
43	Unsupported_certificate	不支持接收的证书类型
44	Certificate_revoked	证书已经撤销
45	Certificate_expired	证书过期
46	Certificate_unknown	在实现证书时产生了一些不确定问题
47	Illegal_parameter	握手过程某个字段超出范围或者与其它字段不符

TLS改变密码规范协议

- TLS修改密文协议由单个消息组成，该消息只包含一个值为1的单个字节。
- 作用:使未决(pending)状态变换为当前(current)状态，更新用于当前连接的密码组。为了保障SSL传输过程的安全性，双方应该每隔一段时间改变加密规范。

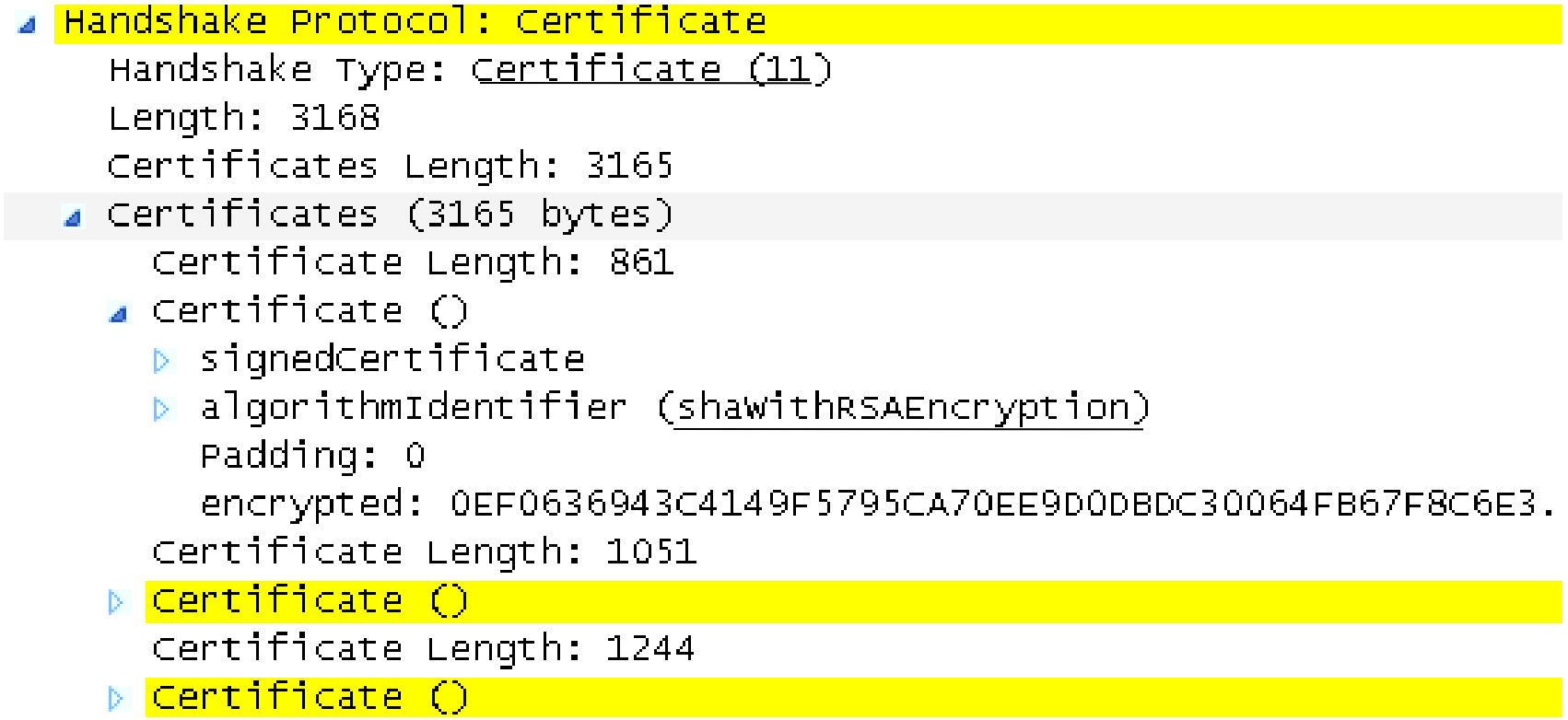
实例：包格式

```
TLSv1 Record Layer: Handshake Protocol: Client Hello
  Content Type: Handshake (22)
  Version: TLS 1.0 (0x0301)
  Length: 65
  Handshake Protocol: Client Hello
    Handshake Type: Client Hello (1)
    Length: 61
    Version: TLS 1.0 (0x0301)
    Random
      gmt_unix_time: Oct 16, 2008 16:59:22.000000000
      random_bytes: 239564C39AD1034521FC747CEE6C0046EA888F
    Session ID Length: 0
    Cipher Suites Length: 22
    Cipher Suites (11 suites)
    Compression Methods Length: 1
    Compression Methods (1 method)
      Compression Method: null (0)
```

必须支持的：TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA

```
TLSv1 Record Layer: Handshake Protocol: Server Hello
  Content Type: Handshake (22)
  Version: TLS 1.0 (0x0301)
  Length: 74
  Handshake Protocol: Server Hello
    Handshake Type: Server Hello (2)
    Length: 70
    Version: TLS 1.0 (0x0301)
    Random
      gmt_unix_time: Oct 16, 2008 16:59:21.000000000
      random_bytes: D43AD17FE589E8BB7B51ABF22DAF870ED96DF0E1DB712FA9
    Session ID Length: 32
    Session ID: 2D7620AAD05A2322CDED39D35752FEFDA1DE1508E0B599C7...
    Cipher Suite: TLS_RSA_WITH_RC4_128_SHA (0x0005)
    Compression Method: null (0)
[Malformed Packet: SSL]
```

- Server为会话确定编号：2D7620….


Handshake Protocol: Certificate
Handshake Type: Certificate (11)
Length: 3168
Certificates Length: 3165
Certificates (3165 bytes)
Certificate Length: 861
Certificate ()
signedCertificate
algorithmIdentifier (shaWithRSAEncryption)
Padding: 0
encrypted: 0EF0636943C4149F5795CA70EE9D0DBDC30064FB67F8C6E3.
Certificate Length: 1051
Certificate ()
Certificate Length: 1244
Certificate ()

- TLSv1 Record Layer: Handshake Protocol: Client Key Exchange
  - Content Type: Handshake (22)
  - Version: TLS 1.0 (0x0301)
  - Length: 134
  - Handshake Protocol: Client Key Exchange
    - Handshake Type: Client Key Exchange (16)
    - Length: 130
- TLSv1 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
  - Content Type: Change Cipher Spec (20)
  - Version: TLS 1.0 (0x0301)
  - Length: 1
  - Change Cipher Spec Message
- TLSv1 Record Layer: Handshake Protocol: Encrypted Handshake Message
  - Content Type: Handshake (22)
  - Version: TLS 1.0 (0x0301)
  - Length: 36
  - Handshake Protocol: Encrypted Handshake Message

四、 TLS性能及应用

Transport Layer Security

- TLS与SSL不同处：
  - version number
  - message authentication code
  - pseudorandom function
  - alert codes
  - cipher suites
  - client certificate types
  - certificate_verify and finished message
  - cryptographic computations
  - Padding

SSL/TLS优缺点

- TLS
  - 优点
    - TLS设计具有可扩展性
    - TLS与Application Layer相互独立
    - TLS支持更多的key size和提供不同的安全等级
    - TLS的hash function使用上更谨慎,同时用了MD5，SHA，一种算法的问题不致引起严重后果
- SSL
  - 优点
    - 很多的browser已支持SSL
  - 缺点
    - SSL需要修改proxy protocol

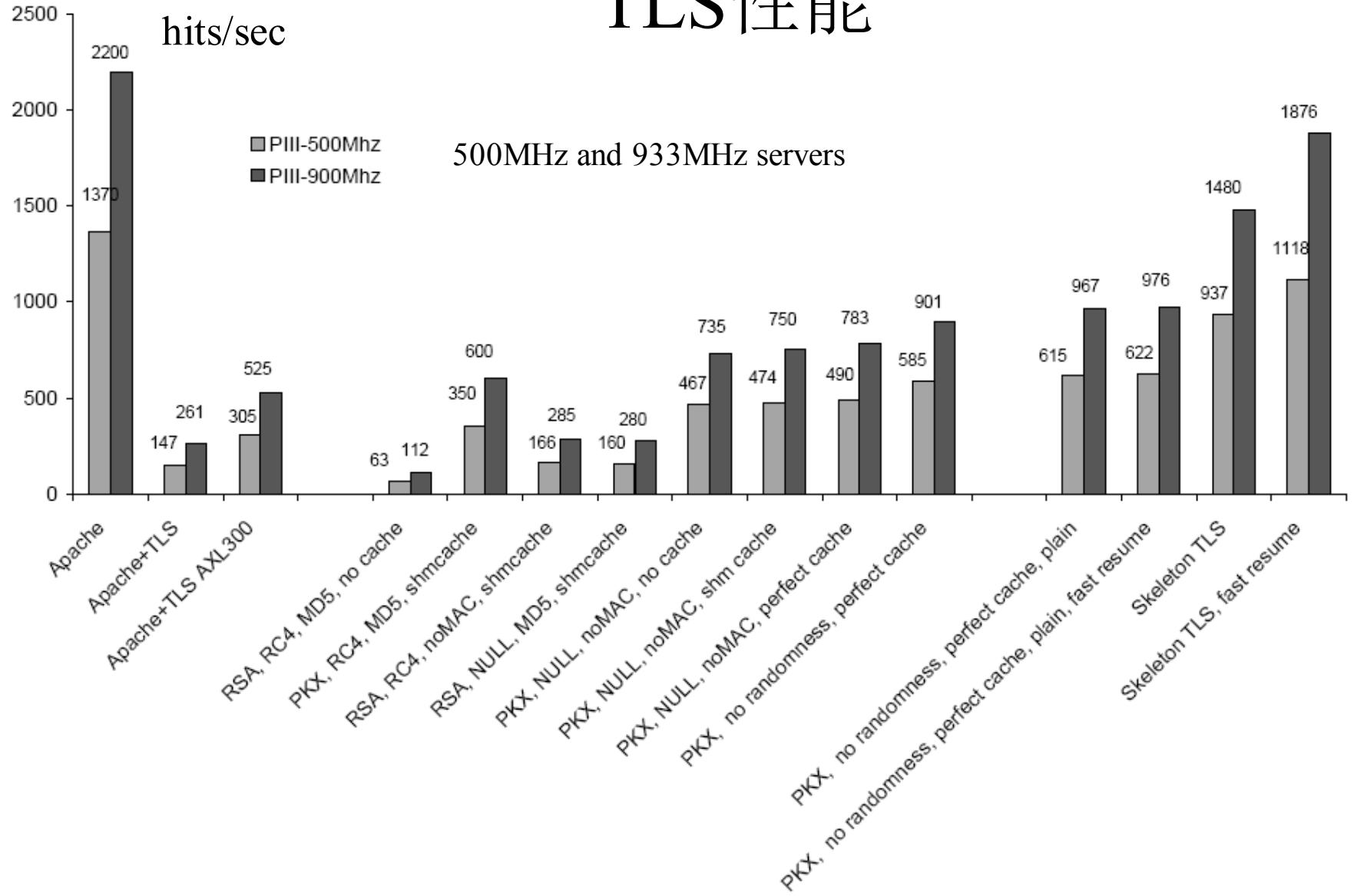
TLS/SSL协议的安全性分析

- 采用的加密和认证算法
  - 加密算法与会话密钥：算法有RC4，RC2，IDEA和DES；密钥由消息散列函数MD5产生
  - 认证算法：采用X.509证书
- 安全
  - 中间人攻击
  - 易受DOS攻击
  - 流量数据分析攻击…

存在的问题

- 密钥管理问题
  - 许多实现，服务器的证书不是基于可信的CA颁发
- 加密强度问题
- 不严谨的实现：man in the middle攻击
- **gotofail：苹果 SSL/TLS**（对签名的认证不会失败http://blog.jobbole.com/60425/）
- Heartbleed安全漏洞：获得服务器私钥

TLS性能



Throughput for Amazon trace and different server configurations

Label	Description of server configuration
Apache	Apache server
Apache+TLS	Apache server with TLS
Apache+TLS AXL300	Apache server with TLS and AXL300
RSA	RSA protected key exchange
PKX	plain key exchange
NULL	no bulk cipher (plaintext)
RC4	RC4 bulk cipher
noMAC	no MAC integrity check
MD5	MD5 MAC integrity check
no cache	no session cache
shmcache	shared-memory based session cache
perfect cache	idealized session cache (always hits)
no randomness	no pseudo-random number generation (also: NULL, noMAC)
plain	no bulk data marshaling (plaintext written directly to the network)
fast resume	simplified TLS session resume (eliminates one round-trip)
Skeleton TLS	all messages of correct size, but zero data

TLS性能

- 参考资料：
  - **Performance Analysis of TLS Web Servers**，Cristian Coarfa, Peter Druschel and Dan S. Wallach，Rice University， In Proceedings of NDSS '02, 2002
  - TLS incurs a substantial cost and reduces the throughput by 70 to 89% relative to the insecure Apache.

- 更多资料：
  - http://www.net.informatik.uni-goettingen.de/research_projects/ptls/sec_perf_papers.htm

代理

实现：

在浏览器里设置https的代理（如127.0.0.1），在浏览器访问网络时：

浏览器与proxy建立tcp链接，发出：

CONNECT server.example.com:443 HTTP/1.1

Host: server.example.com:443

然后proxy会向webserver端建立tcp连接；

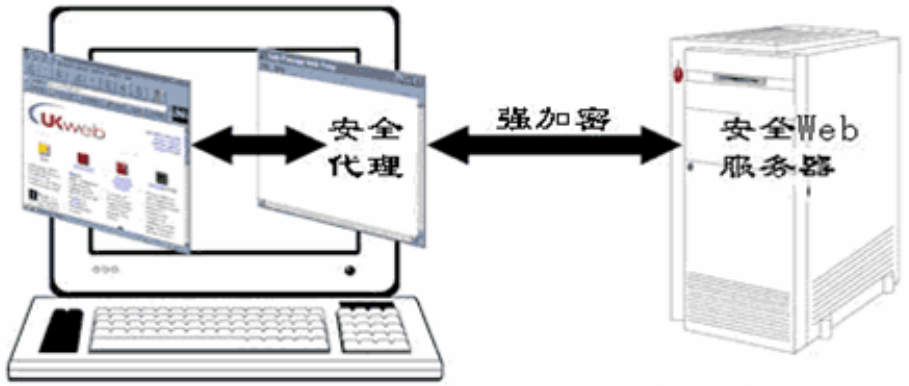
之后这个代理便成了内容转发装置；

浏览器与目标机建立一个安全通道（端到端）

用途：

在无法提高浏览器的加密级别时，

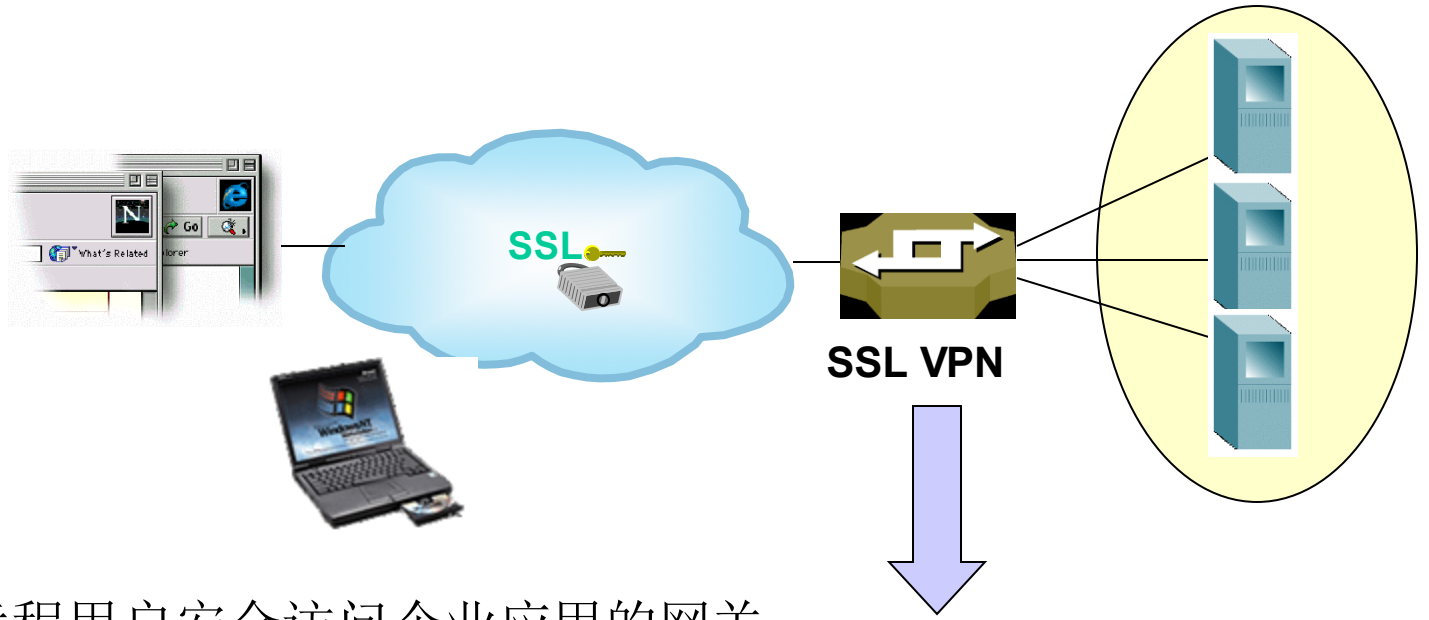
用代理实现更强的加密。



应用

- 匿名连接
  - 使用基本模式——仅对服务器验证，用于用户注册等信息的保护
- VPN
  - 服务器端需要验证证书，客户端可以验证证书或用户名加密码
- 电子商务
  - 商家和银行必须有证书
  - 客户和商家之间的通信用匿名方式，兼顾了安全和易用性
  - 商家与银行之间传送的是顾客数据，通信双方必须先互相验证证书

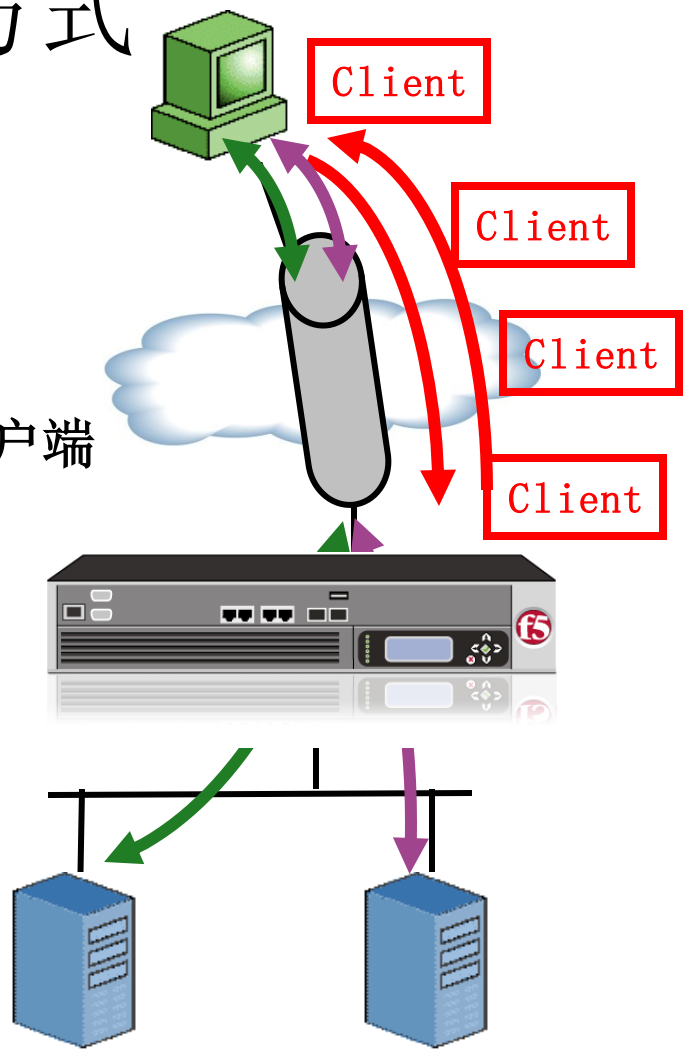
SSL VPN: what ?



- 远程用户安全访问企业应用的网关
- 作为用户访问企业应用的门户
- 作为用户和企业应用之间的代理,用户无法直接访问企业私网
- …..
  - 为用户与企业应用之间建立安全会话

SSL VPN工作方式

- 用户使用443端口与VPN网关建立SSL连接
- VPN网关下载 Java或 ActiveX 代码到客户端
- VPN网关与浏览器建立加密安全隧道
- 用户通过SSL VPN隧道访问资源

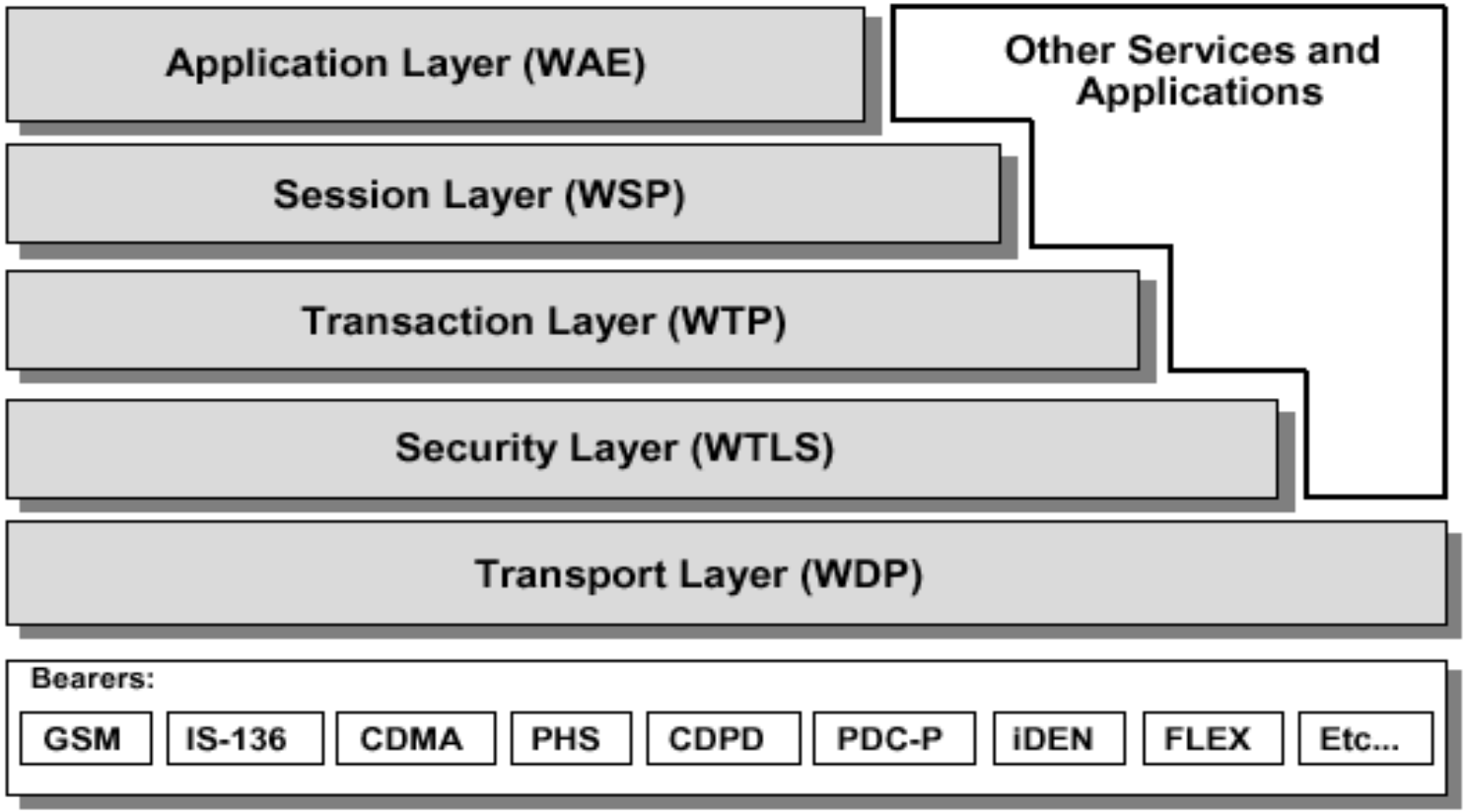


五、WTLS简介

SSL/TLS与WTLS的差别

- WTLS在一般udp这类不可靠信道上工作，因此每个消息里要有序列号，协议里也要靠它来处理丢包，重复等情况。此外，拒绝服务攻击也因此变得更加容易。
- WTLS建立的安全连接是在wap网关和手持设备之间，wap网关和web server之间如果也要保密，便要采用SSL，即在这种模型中无法实现端到端的加密。
- WTLS协议里加了一种称为key_refresh的机制,当传递了一定数量数据包后，双方通过同样的算法将自己的密钥做一下更新。

Wireless Transport Layer Security (WTLS)



❖若低层承载为IP则传输层为UDP

WTLS

- WTLS is an optional layer
- WTLS is a variant of TLS optimized for use in wireless applications
  - Authentication: Asymmetric Key Crypto
    - Class 1: No Authentication
    - Class 2: Server Authentication
    - Class 3: Mutual Authentication
  - Privacy: Symmetric Key Crypto
  - Data Integrity: MACs

WTLS Class 1 Authentication

ClientHello ---------->

ServerHello

<---------- ServerHelloDone

ClientKeyExchange
ChangeCipherSpec
Finished ---------->

<---------- Finished

Application Data <----------> Application Data

WTLS Class 2 Authentication

Server Authentication Only

ClientHello	---------->	
		ServerHello
		Certificate
	<----------	ServerHelloDone
ClientKeyExchange		
ChangeCipherSpec		
Finished	---------->	
	<----------	Finished
Application Data	<---------->	Application Data

WTLS Class 3 Authentication

Mutual Authentication

Client Hello	---------->	
		ServerHello
		Certificate
		CertificateRequest
	<----------	ServerHelloDone
Certificate		
ClientKeyExchange *(only for RSA)*		
CertificateVerify		
ChangeCipherSpec		
Finished	---------->	
	<----------	Finished
Application Data	<---------->	Application Data

总结

二、TLS概述

- SSL、TLS与WTLS
- TLS(SSL）功能与作用
- 基于TLS的HTTP

二、**TLS**加密/密码学基础

- 对称加密、非对称加密及单向散列函数
- TLS的密钥协商处理

总结

三、TLS协议

- TLS体系结构, RFC2246标准
- Handshake Protocol
- Record Protocol

四、 TLS性能及应用

五、 WTLS介绍