第5章 应用层安全协议

5.1 引言

在实际的网络系统中，通常将ISO OSI参考模型的5～7层，即会话层、表示层和应用层都归结为应用层，用一个层次来实现，例如，TCP/IP协议集就采用了这种简化的网络层次结构。

应用层为特定的网络应用提供数据传输服务，根据应用程序的需要，数据传输服务可以是面向连接或无连接的。应用层协议仍属于网络体系结构的一部分，并不是应用程序。在实现方式上，可以由网络系统实现，也可以在应用程序中实现，主要取决于具体的网络应用。例如，互联网中的Web系统采用客户/服务器模式，客户端是Web浏览器，服务器端是Web服务器，它们之间采用HTTP协议进行通信。HTTP协议就是一种应用层协议，由应用程序来实现，在客户端由Web浏览器来实现，在服务器端由Web服务器来实现。

由于网络应用是多种多样的，每一种网络应用都可能对应一种应用层协议。例如，在互联网中，除了上面提到的Web系统外，还有电子邮件(E-mail)、远程登录(Telnet)、文件传输(FTP)以及域名系统(DNS)等，它们都属于开放的网络应用系统，都需要通过相应的应用层协议来支持通信。

应用层安全性主要是解决面向应用的信息安全问题，涉及信息交换的保密性和完整性，防止在信息交换过程中数据被非法窃听和篡改。

有些应用层安全协议是对应用层协议的安全性增强，即在应用层协议的基础上增加了安全算法协商和数据加密/解密等安全机制，如S-HTTP(Secure HTTP)协议、S/MIME(Secure/MIME)协议等；还有些应用层安全协议是为解决特定应用的安全问题而开发的，如PGP(Pretty Good Privacy)协议等。本章对这些应用层安全协议作简要的介绍。

5.2 S-HTTP协议

5.2.1 HTTP协议

Web系统是互联网中应用最为广泛的应用系统，它基于客户/服务器模式，整个系统由Web服务器、浏览器和通信协议三部分组成。其中，通信协议为超文本传输协议

(HTTP),它是为分布式超媒体信息系统设计的一种应用层协议,能够传送任意类型的数据对象,以满足 Web 服务器与客户之间多媒体通信的需要。

HTTP 协议是一种面向 TCP 连接的协议,客户与服务器之间的 TCP 连接是一次性连接。它规定每次连接只处理一个请求,服务器返回本次请求的应答后便立即关闭连接,在下次请求时再重新建立连接。这种一次性连接主要考虑到 Web 服务器将面向互联网中的成千上万个用户,只能提供有限个连接,及时地释放连接可以提高服务器的执行效率,避免服务器连接的等待状态。同时,服务器不保留与客户交易时的任何状态,以减轻服务器的存储负担,从而保持较快的响应速度。HTTP 协议允许传送任意类型的数据对象,通过数据类型和长度来标识所传送的数据内容和大小,并允许对数据进行压缩传送。

用户在浏览器或 HTML 文档中定义了一个超文本链后,浏览器将通过 HTTP 协议请求与指定的服务器建立连接。如果该服务器一直在 HTTP 端口上侦听连接请求,该连接便会建立起来。然后客户通过该连接发送一个包含请求方法的请求消息块。HTTP 协议定义了 7 种请求方法,每种请求方法规定了客户和服务器之间不同的信息交换方式,常用的请求方法是 GET 和 POST。服务器将根据客户请求完成相应的操作,并以应答消息块的形式返回给客户,最后关闭连接。在 HTTP 协议中,客户与服务器之间的信息交换采用了下列两种消息块结构。

1. 请求消息块

在 HTTP 协议中,客户的请求消息将按下列结构来组织。

description: method URL HTTP version

General Header

Request Header

Entity Header

Entity Body

请求消息中的各个字段是根据不同的请求方法任选的。

(1) 描述行(description)字段定义了请求方法(method)、URL 和 HTTP 协议版本号。由于这时已经建立了连接,故这里的 URL 不再包含协议名、服务器名和端口号。请求方法定义了在该资源上应执行的操作,HTTP 协议定义了 7 种请求方法,如 GET, HEAD, PUT 和 POST 等,而最常用的方法是 GET 和 POST。

① GET: 该方法是取回由 URL 指定的资源,主要用于取回由一个超文本链所定义的对象。如果对象是文件,则 GET 取回的是文件内容;如果对象是程序或描述,则 GET 取回的是该程序执行的结果或该描述的输出;如果对象是数据库查询,则 GET 取回的是本次查询的结果。

② POST: 当客户向服务器传送大块数据并要求服务器和公共网关接口(CGI)程序做进一步处理时,要使用 POST 方法。例如,发送 HTML FORM 内容,让 CGI 程序进行处理。这时,FORM 内容的 URL 编码将随请求消息一起发出。

(2) 普通头(General Header)字段是对请求消息的一般说明。

(3) 请求头(Request Header)字段给出了所传送数据对象的类型、长度、压缩方法以

及编程语言等。服务器将根据请求头来解释和处理本次请求。例如，服务器将根据数据的类型和长度动态地分配空间，以存放 FORM 的内容。

(4) 实体头(Entity Header)字段提供了对实体的进一步描述。

(5) 实体体(Entity Body)字段是客户进行 POST 请求时的 FORM 内容，提供给服务器的 CGI 程序做进一步处理。

2. 应答消息块

在 HTTP 协议中，服务器的应答消息将按下列结构来组织：

description: HTTP version Status code Reason phrase

General Header

Response Header

Entity Header

Entity Body

同样，应答消息块中的各个字段也是根据不同的应答内容任选的。

(1) 描述行(description)字段给出了 HTTP 协议版本号、状态码(Status Code)和原因短语(Reason Phrase)。主要说明了服务器是否成功地执行了客户请求及其原因。

(2) 普通头(General Header)字段是对返回消息的一般说明。

(3) 应答头(Response Header)字段主要给出了服务器程序名、通知客户所请求的 URL 需要认证、所请求的资源需要经过多长时间才能使用等信息。

(4) 实体头(Entity Header)字段提供了有关本次所返回对象的信息。浏览器将根据这些信息来解释所返回的对象。它主要有实体信息的类型、长度、压缩方法、最后一次修改的时间及数据有效期等。

(5) 实体体(Entity Body)是服务器应答对象本身，可以是任何格式的超媒体文件。

由于 HTTP 协议是为开放的互联网设计的，因此没有涉及信息安全问题。随着 Web 技术的发展，B/S(浏览器/服务器)应用模式日益受到人们的青睐，不仅互联网中的信息服务系统，而且很多企业网、商务网、政务网以及金融网等内部网中的信息系统也都采用 Web 技术来构建。因此，Web 通信安全问题变得十分突出。

5.2.2　S-HTTP 协议

解决 Web 通信安全问题的基本方法是通过 HTTP 安全协议来增强 Web 通信的安全性。目前，HTTP 安全协议主要有两种：HTTPS 和 S-HTTP。

HTTPS 协议是基于 SSL 的 HTTP 安全协议，通常工作在标准的 443 端口上。在实际应用中，HTTPS 协议使用比较简便。如果一个 Web 服务器提供基于 HTTPS 协议的安全服务，并在客户机上安装该服务器认可的数字证书，则用户便可以使用支持 SSL 协议的浏览器(通常浏览器都支持 SSL 协议，如 IE 浏览器等)，并通过“https://www.服务器名.com”域名来访问该 Web 服务器，Web 服务器与浏览器之间通过 SSL 协议进行安全通信，提供身份鉴别、数据加密和数据认证等安全服务。由于 HTTPS 协议的安全机制是通过 SSL 协议实现的，在第 4 章中已对 SSL 协议做了详细的介绍，因此本节主要介绍

S-HTTP 协议。

S-HTTP 协议最初是由 Terisa 公司开发的，它是在 HTTP 协议的基础上扩充了安全功能，提供了 HTTP 客户和服务器之间的安全通信机制，以增强 Web 通信的安全性。RFC 2660 文档公布了 S-HTTP 协议的技术规范。

S-HTTP 协议的目标是提供一种面向消息的可伸缩安全协议，以便广泛地应用于商业事务处理。因此，它支持多种安全操作模式、密钥管理机制、信任模型、密码算法和封装格式。在使用 S-HTTP 协议通信之前，通信双方可以协商加密、认证和签名等算法以及密钥管理机制、信任模型、消息封装格式等相关参数。在通信过程中，双方可以使用 RSA、DSS 等密码算法进行数字签名和身份鉴别，以保证用户身份的真实性；使用 DES、3DES、RC2、RC4 等密码算法来加密数据，以保证数据的保密性；使用 MD2、MD5、SHA 等单向散列函数来验证数据和签名，以保证数据的完整性和签名的有效性，从而增强了 Web 应用系统中客户和服务器之间通信的安全性。

S-HTTP 是一种面向安全消息的通信协议，它与 HTTP 消息模型共存，很容易实现与 HTTP 应用的集成。S-HTTP 为 HTTP 客户和服务器提供了多种安全机制，进而为用户提供安全的 Web 服务。

在 S-HTTP 客户和服务器中，主要采用 CMS（Cryptographic Message Syntax）和 MOSS（MIME Object Security Services）消息格式，但并不限于 CMS 和 MOSS，它还可以融合其他多种加密消息格式及其标准，并且支持多种与 HTTP 相兼容的系统实现。S-HTTP 只支持对称密码操作模式，不需要客户提供公钥证书或公钥，这意味着客户能够自主地产生个人事务，并不要求具有确定的公钥。

S-HTTP 支持端到端的安全事务，客户可以事先初始化一个安全事务。S-HTTP 中的密码算法、模式和参数是可伸缩的，客户和服务器之间可以协商事务模式（如请求/响应是否加密和签名）、密码算法（RSA 或 DSA 签名算法，DES 或 RC2 加密算法）以及证书选择等。

1. 消息处理

1）创建 S-HTTP 消息

一个 S-HTTP 消息可以通过下列方法来创建。

（1）Clear-text 消息：这是一个 HTTP 消息或者一些其他的数据对象，Clear-text 消息被封装在一个 S-HTTP 消息中进行传送；

（2）接收者的密码参数选择和密钥材料：这是由接收者或者一些默认参数集明确指定的；

（3）发送者的密码参数选择和密钥材料：这是由发送者输入的，只存在于发送者的内存中。

为了创建一个 S-HTTP 消息，发送者需要将发送者参数和接收者参数集成在一起，产生一个密码和密钥材料的列表。然后发送者使用列表中的数据来增强 Clear-text 消息的安全性，再通过发送者和接收者参数组合将 Clear-text 消息转换成 S-HTTP 消息。

2）恢复 S-HTTP 消息

接收者可以采用下列 4 种方法之一来恢复一个输入的 S-HTTP 消息：

(1) S-HTTP 消息；

(2) 接收者规定的密码参数选择和密钥材料；

(3) 接收者当前的密码参数选择和密钥材料；

(4) 发送者事先规定的密码选项。

发送者可以规定在一个消息中所执行的加密操作。为了恢复一个 S-HTTP 消息，接收者需要读取消息头信息，以发现在该消息中的密码变换，并使用某种发送者和接收者参数组合来去除该变换。接收者也可以选择校验，增强发送者和接收者之间的匹配。

3) 操作模式

任何消息都可以采用签名、认证和加密来保护，这三种保护方法可以单独使用，也可以组合起来使用。并支持多种密钥管理机制，包括基于口令的人工共享私密和基于公钥的密钥交换。在交换密钥时，要事先建立一个会话密钥，以便将机密消息传递给没有公钥对的用户。

(1) 签名：如果使用了数字签名，则可以将一个适当的证书与消息联系起来(可以沿着一个证书链)，或者发送者可以认为接收者独立地获得了所需的证书。

(2) 密钥交换和加密：为了支持对称密码算法，S-HTTP 定义了两种密钥传递机制：一是使用被公钥密封的密钥交换；二是使用预先安排(Prearranged)的密钥。对于前者，在传送对称密码系统的密钥时要使用接收者公钥来加密。对于后者，使用预先安排的会话密钥来加密内容。密钥认证信息是由消息头指定的。

(3) 消息完整性和发送者认证：S-HTTP 通过计算 MAC 码来校验消息的完整性，并对消息的发送者进行认证。它使用一个共享密钥对关键的内容进行散列计算，共享密钥可以通过多种方法预先协商好，不必使用公钥密码系统，也不需要加密。

2. 消息头

从语句上看，S-HTTP 消息与 HTTP 消息相类似，都是由消息头和消息体组成的。然而，S-HTTP 消息头范围不同于 HTTP，消息体通常是加密保护的。

1) 请求头

为了将 S-HTTP 消息与 HTTP 消息区分开，并允许特定的处理，S-HTTP 将请求头中的 method 定义为 Secure；version 定义为 Secure-HTTP/1.4；URL 应设置为 *，以防止潜在的敏感信息泄露。例如，一个 S-HTTP 请求头可以描述如下：

```
Secure * Secure-HTTP/1.4
```

这样，S-HTTP 与 HTTP 进程就可以混合使用相同的 TCP 端口(如 80 端口)了。

2) 响应头

对于 S-HTTP 响应头，同样使用 Secure-HTTP/1.4 来标识该协议。例如，一个 S-HTTP 响应头可以描述如下：

```
Secure-HTTP/1.4 200 OK
```

在 S-HTTP 响应头中，状态始终为 200 OK，它并不表示 HTTP 请求成功或失败的状态，主要为防止通过对 HTTP 请求成功与否状态的分析来推测数据的接收者。

3) S-HTTP 头

在 S-HTTP 头中，除了 Content-Type 和 Content-Privacy-Domain 外都是可选的，消息体与头之间用两个连续的 CRLF 符分隔开。

(1) Content-Type：内容类型(Content-Type)行描述了一个消息的内容类型，主要有两种内容类型：CMS 和 MOSS。

在一般情况下，由端点封装的内容应是一个 HTTP 消息，内容类型是 CMS，这里用一个内容类型行来说明：Content-Type：message/http。如果内部消息是 S-HTTP 消息，则内容类型是 application/s-http。

MOSS 内容类型是一种可接受的 MIME 内容，它描述了对密文所做的处理，如加密、签名等。在内容类型行上描述了内部内容的类型，对于 HTTP 消息，内容类型是 message/http。

(2) Prearranged-Key-Info：这个描述行给出了有关密钥信息，这个密钥是针对预先已约定好的内部加密格式，主要用于支持会话密钥的 Inband 通信，以返回加密方法。在这种情况下，通信的任何一方都不需要拥有一个密钥对。

在 S-HTTP 中，定义了两种交换密钥的方法：Inband 和 Outband。Inband 方法表明会话密钥是预先交换的，它使用了一个适当方法(method)的 Key-Assign 头。Outband 方法表明通过一个确定的名字从外部访问密钥材料，名字可以通过访问数据库或者利用键盘输入来获得。

(3) MAC-Info：在消息头中，定义了一个 MAC 行，用于提供消息认证和完整性检查，它定义了散列算法、认证数据和密钥空间。散列计算可以采用 MD2、MD5 和 SHA 等算法，认证数据包含消息文本散列值、时间值以及客户与服务器之间的共享秘密信息等。时间参数是可选的，不做散列计算，主要为防止重播攻击。消息文本应当是被封装的 S-HTTP 消息内容。MAC-Info 允许快速的消息完整性认证，双方共享一个密钥(可以在前面的消息中使用 Key-Assign 参数)。

3. 消息内容

消息内容主要由 Content-Privacy-Domain 和 Content-Transfer-Encoding 字段来确定。对于一个 CMS 消息，使用 8 位 Content-Transfer-Encoding，其内容就是 CMS 消息本身。如果 Content-Privacy-Domain 是 MOSS，则内容是由 MOSS 多个安全部分组成的。下面是消息封装格式选项。

1) Content-Privacy-Domain：CMS

Content-Privacy-Domain 的 CMS 符合 CMS 标准格式，任何消息都可以采用保护和无保护模式，其中保护模式有三种：加密、签名和加密加签名。S-HTTP 的认证保护模式是由 MAC-Info 头中的 CMS 编码独立提供的，因为 CMS 只支持 DigestedData 类型，而不支持 KeyDigestedData 类型。

(1) 签名：签名使用了 CMS SignedData 类型。当使用数字签名时，可以将一个适当的证书和 CMS 所指定的消息(可以沿着一个证书链)联系起来，或者接收者独立地获取该证书。

(2) 加密：加密使用了两种 CMS 数据类型，EnvelopedData 和 EncryptedData 。当使用公钥加密一个消息时，采用 EnvelopedData 类型。当使用预先安排的密钥来加密一个消息时，采用 EncryptedData 类型。在这个模式中，使用了预先安排的会话密钥来加密内容，而会话密钥是通过消息头中的密钥验证信息来验证的。

当需要同时使用加密和签名来保护一个消息时，必须创建一个 CMS SignedData 过程来支持签名，并用 EncryptedData 类型来封装消息。

2) Content-Privacy-Domain：MOSS

MOSS 的消息体是一个 MIME 消息，其内容类型与 S-HTTP 头中的 Content-Type 行相匹配。在加密和签名消息时，应当分别使用 Multipart/encrypted 和 Multipart/signed 类型。然而，Multipart/signed 并不能传输密钥材料，它可以使用 Multipart/mixed 消息，以便传输验证签名所使用的证书。当同时使用加密和签名时，通常签名先于加密。

3) 允许的 HTTP 头

为安全起见，HTTP 头通常应当出现在一个 S-HTTP 消息的内部内容中，而不能出现在该 S-HTTP 消息的外包装上。然而，有些消息头必须是代理(Agent)可见的，它们并不需要访问被封装的数据，这些头可以出现在 S-HTTP 头中。

4. 密码参数

每个 S-HTTP 请求通过接收者所提供的密码参数选项进行预处理。这些选项位于两个地方：

(1) 在一个 HTTP 请求/响应头中；

(2) 在包含废弃锚(Anchor)的 HTML 中。

这里可以提供两种密码选项：协商选项和密钥选项。协商选项给出了一个消息接收者的密码参数选择；密钥选项提供了密钥材料，发送者可以用它来增强一个消息。

1) 协商选项

双方可以通过 permit/require 形式来协商各自的密码强度需求和参数选择，协商选项的选取依赖于实现的能力和特定应用的需求。协商是通过一个协商头实现的，协商头位于被封装的 HTTP 头中，而不在 S-HTTP 头中。一个协商头是由 4 部分组成的。

(1) 属性(Property)：被协商的选项，如分组密码算法。

(2) 值(Value)：属性值，如 DES-CBC 等。

(3) 方向(Direction)：从源点观察的协商源或目的。

(4) 强度(Strength)：参数选择强度，即必需、可选和拒绝。

例如，一个协商头定义为 SHTTP-Symmetric-Content-Algorithms：recv-optional＝DES-CBC，RC2。其含义是可以任意使用 DES-CBC 或 RC2 算法加密消息。

S-HTTP 定义了以下的协商头。

(1) SHTTP-Privacy-Domains：这个头涉及 Content-Privacy-Domain 类型。

(2) SHTTP-Certificate-Types：这个头指定了代理认可的公钥证书类型，当前定义的值是 X.509 和 X.509v3。

(3) SHTTP-Key-Exchange-Algorithms：这个头指定了可用于密钥交换的算法，定

义的值是 DH、RSA、Outband 和 Inband，DH 为 Diffie-Hellman X9.42 样式的信封，RSA 为 RSA 信封，Outband 为某些扩展密钥协议类型，Inband 表明会话密钥是预先交换的。推荐的配置是客户无证书而服务器有证书。

(4) SHTTP-Signature-Algorithms：这个头指定了可用于数字签名的算法，定义的值是 RSA 和 NIST-DSS，RSA 和 NIST-DSS 的密钥长度是指定的，密钥长度与一种给定的证书相互作用，因为密钥及其长度是在公钥证书中指定的。

(5) SHTTP-Message-Digest-Algorithms：这个头指定了可用于消息摘要的算法，定义的值是 RSA-MD2、RSA-MD5 和 NIST-SHS。

(6) SHTTP-Symmetric-Content-Algorithms：这个头指定了用于加密消息内容的对称密码算法，定义的值有 DES-CBC、DES-EDE-CBC、DESX-CBC、RC2-CBC、IDEA-CBC 和 CDMF-CBC，其中 RC2 密钥的长度是可变的。

(7) SHTTP-Symmetric-Header-Algorithms：这个头指定了用于加密消息头的对称密码算法，定义的值有 DES-ECB、DES-EDE-ECB、DES-EDE3-ECB、DESX-ECB、IDEA-ECB、RC2-ECB 和 CDMF-ECB，其中 RC2 密钥的长度是可变的。

(8) SHTTP-MAC-Algorithms：这个头指定了一个可接受的 MAC 算法，定义的值有 RSA-MD2-HMAC、RSA-MD5-HMAC 和 NIST-SHS-HMAC。

(9) SHTTP-Privacy-Enhancements：这个头指定了应用的安全增强，定义的值有 sign、encrypt 和 auth，分别指示对消息的签名、加密和认证。

(10) Your-Key-Pattern：这是一个通用的模式匹配语法，在大量密钥材料类型情况下用作描述标识符。

下面的例子是一个服务器典型的头块配置。

```
SHTTP-Privacy-Domains: recv-optional=MOSS, CMS;orig-required=CMS
SHTTP-Certificate-Types: recv-optional=X.509; orig-required=X.509
SHTTP-Key-Exchange-Algorithms: recv-required=DH; orig-optional=Inband, DH
SHTTP-Signature-Algorithms: orig-required=NIST-DSS; recv-required=NIST-DSS
SHTTP-Privacy-Enhancements: orig-required=sign; orig-optional=encrypt
```

在协商选项中还使用了默认值，这些默认值为

```
SHTTP-Privacy-Domains: orig-optional=CMS; recv-optional=CMS
SHTTP-Certificate-Types: orig-optional=X.509; recv-optional=X.509
SHTTP-Key-Exchange-Algorithms: orig-optional=DH, Inband, Outband; recv-
optional=DH, Inband, Outband
SHTTP-Signature-Algorithms: orig-optional=NIST-DSS; recv-optional=NIST-DSS
SHTTP-Message-Digest-Algorithms: orig-optional=RSA-MD5; recv-optional=RSA
-MD5
SHTTP-Symmetric-Content-Algorithms: orig-optional=DES-CBC; recv-optional=
DES-CBC
SHTTP-Symmetric-Header-Algorithms: orig-optional=DES-ECB; recv-optional=
DES-ECB
SHTTP-Privacy-Enhancements: orig-optional=sign, encrypt,auth; recv-required
```

```
=encrypt;recv-optional=sign, auth
```

2）密钥选项

这里是一组用于通信或标识接收者密钥材料的选项。

（1）Encryption-Identity：加密标识信息，其中有一个用 ASCII 字符串表示的名字类型(name-class)，它采用两种名字格式：DN(Domain Name)和 MOSS，前者在 RFC-1779 中描述，后者在 RFC 1848 中描述。

（2）Certificate-Info：为了支持在 DN(由 Encryption-Identity 头所指定)上的公钥操作，发送者可以在这个选项中包含证书信息，它定义了两种证书组：PEM 和 CMS。

（3）Key-Assign：将一个密钥捆绑到符号名上，可选的参数有 Key-Name、Lifetime、Method、Ciphers 和 Method-args 等，其中，

① Key-Name 是该密钥捆绑后的符号名，用一个字符串表示。

② Lifetime 是密钥的生存期，表示在此期间该消息接收者允许发送者接收密钥。如果没有指定生存期，则说明这个密钥可以重复使用于若干事务中。

③ Method 是若干密钥交换方法中的一种，当前定义的值只有 Inband。

④ Ciphers 是一个密码算法列表，这些密码算法都是该密钥能够适用的。如果是 null 值，则表示该密钥不适合与任何一种密码算法一起使用，这对于交换和计算 MAC 密钥是有用的。

⑤ Method-args 是所希望的会话密钥。

这个头行可以出现在一个非封装的头中或者在一个封装的消息中。当一个未经密封的密钥被直接分配时，这个头行只能出现在一个加密封装的内容中。

在 Inband 密钥分配中，允许将一个未经密封的密钥直接分配给一个符号名。Inband 密钥分配非常重要，因为它允许代理之间秘密地进行通信，并且只要任何一方(并非双方)拥有密钥对即可。这种机制还允许在不计算公钥的情况下去改变密钥。在这个头行中所传送的密钥信息必须是在被保护的 HTTP 请求内部，不能在未加密的消息中使用。

（4）SHTTP-Cryptopts：它允许服务器将若干个头组合起来，捆绑到一个 HTML 锚上，这些头的锚名是用 scope 参数来指示的。如果一个消息包含了 S-HTTP 协商头和 SHTTP-Cryptopts 行上的组合头，则其他头应当用于所有没有被捆绑在 SHTTP-Cryptopts 行上的锚。

5.2.3　S-HTTP 协议的应用

1. 对 HTTP 消息加密保护

示例 1：假设一个支持 S-HTTP 协议的客户通过 URL 来访问一个 Web 服务器，要求服务器对返回的 HTML 页面内容进行加密保护，使用 Inband 方式分配和交换密钥。下面是客户生成的 HTTP 请求所需的相关信息。

```
200 OK HTTP/1.0
Server-Name: Linux-Server-1
Certificate-Info: CMS, MIAGCSqG … (省略的证书信息)
Encryption - Identity: DN - 1779, null, CN = NPU Computer college , OU = NISI
```

```
Certificate, O="NPU Information Security, Inc.", C=CN;
SHTTP-Privacy-Enhancements: recv-required=encrypt
< A name=tag1 HREF="shttp://www.secage.com/secret">  Don't read this. < /A>
```

客户按指定的 URL 创建以下的 HTTP 请求：

```
GET/secret HTTP/1.0
Security-Scheme: S-HTTP/1.4
User-Agent: Web-O-Vision 1.2
Accept: * . *
Key-Assign: Inband, 1, reply, des-ecb; 7878787878787878
```

在这个 HTTP 请求中加入了 Key-Assign 行，表示要求服务器对返回的应答消息进行加密。由于采用了 Inband 密钥分配方式，客户可以和服务器共享一个密钥，而不必拥有公钥。客户将该 HTTP 请求封装成以下的 S-HTTP 消息：

```
Secure * Secure-HTTP/1.4
Content-Type: message/http
Content-Privacy-Domain: CMS
MIAGCSqG …（被 RSA 封装的消息）
```

当服务器收到该请求后，首先解释和执行该请求，查询相应的文档。然后生成一个 HTTP 响应，向客户返回该文档。该 HTTP 响应为

```
HTTP/1.0 200 OK
Security-Scheme: S-HTTP/1.4
Content-Type: text/html
<A href="/prize.html"
CRYPTOPTS="Key-Assign: Inband, chenning 1, reply, des-ecb; 020406080a0c0e0f;
SHTTP-Privacy-Enhancements: recv-required=auth">Click here to claim your prize</
A>
```

服务器将这个 HTTP 响应封装成以下的 S-HTTP 消息：

```
Secure *  Secure-HTTP/1.4
Content-Type: message/http
Prearranged-Key-Info: des-ecb, 697fa820df8a6e53, inband:1
Content-Privacy-Domain: CMS
MIAGCSqG …(被加密的 CMS 消息)
```

其中，被加密的 CMS 消息可以通过下列计算公式解密成原文：

```
DES-DECRYPT(inband:1,697fa820df8a6e53)
```

2. 对 HTTP 消息认证保护

示例 2：假设一个支持 S-HTTP 协议的客户通过 URL 来访问一个 Web 服务器，要求服务器在返回 HTML 页面时对客户身份进行认证。首先，客户创建以下的 HTTP 消息：