第十二章 密钥交换协议(II): 2-方协议

本章详细阐述带身份认证的 2-方密钥交换协议[1]。实际上我们在上一章所讨论的 *Bellovin-Merrit* 协议、*Adballa-Pointcheval* 协议和 *Bellare-Rogaway* 协议(习题 11-5)都属于这类协议，只不过那些协议基于事先共享的秘密口令来认证合法参与方的身份，而本章所讨论的这些协议基于更强有力的基本密码方案，包括公钥加密方案、数字签名方案和消息认证方案等。我们首先在 12.1 节建立这类协议的安全性概念，接下来讨论一系列的协议实例，首先是著名的 *Diffie-Hellman* 协议(12.2 节)；接下来是 *SIGMA* 协议(12.3 节)，这是许多密钥交换协议的原型，其最著名的变体就是用于因特网 *VPN/IPSec* 的 *IKE* 协议；接下来讨论用于 *Web* 安全会话的 *TLS/SSL* 协议。所有这些实例在目前都具有非常广泛的应用，不仅其安全性质有严格的理论保证，而且其实用性也经历了长期的实践检验，属于最值得掌握的网络安全协议。最后作为综合应用，较详细地解释了在因特网上实现虚拟私有网的技术解决方案。

12.1 协议安全性的概念

概要地说，带身份认证的密钥交换协议需要要同时完成两类目标：第一，在线认证协议当前参与方的身份，其中某些协议只追求一方认证另一方的身份，称为“单向认证”，另一些协议追求双方互相认证，称为“双向认证”；第二，在协议双方之间生成一个密钥 Ks，Ks 用于接下来对一切需要保密的数据进行对称加密。精确地说，这类协议的安全目标包括以下必须被同时满足的三点：

如果协议任何一方 A(B)按照协议的逻辑判定对方的身份是 B(A)，则当前实际参与协议的对方确实是 B(A)，这就是抗身份欺诈性质；

如果协议任何一方 A(B) 按照协议的逻辑判定当前与之会话的对方是 B(A)，则对方也必定判定当前与之会话的对方是 A(B)，这就是协议的一致性；

除合法参与方之外，协议所生成的会话密钥 Ks 对任何第三方(包括被动或主动攻击者)都无法(用 P.P.T.算法)有效推断出来，这就是密钥保密性。

如果读者由此简单地认为带身份认证的密钥交换协议可以由身份认证协议+密钥交换协议而构成，这就不正确了，带身份认证的密钥交换协议不能如此简单地构造，另外注意协

[1] 不少文献也称做“密钥协商协议”,今后我们有时也使用这一名称。

议的抗身份欺诈性质并不能蕴涵一致性要求，例如第 9.1 节表明确实存在这样的(设计不当的)协议，虽然攻击者不能有效冒充任何合法参与方，但却可以使合法的协议双方得出彼此不一致的判定结论。

下面几节讨论一系列的具体实例，所有这些实例都满足以上三项安全性质，这已经为精确的理论分析所证明。

12.2 *Diffie-Hellman* 协议

第一个例子是著名的 *Diffie-Hellman* 协议，发表于 1976 年[2]。为理解这一协议首先需要回顾第 8 章解释过的几类难解性问题。设 G 是循环群，例如 F_p*(p 是大素数)，g 是其公开的生成子，G 上的离散对数问题是指：任给 $U=g^x$ 求指 x；G 上的计算性 *Diffie-Hellman* 问题是指：任给 $U=g^x$ 和 $V=g^y$ 两个元素，求 g^{xy}；G 上的判定性 *Diffie-Hellman* 问题是指：任给 $U=g^x$、$V=g^y$ 和 W 三个元素，判定 $W \stackrel{?}{=} g^{xy}$。*Diffie-Hellman* 协议的安全性要求 G 上的判定性 *Diffie-Hellman* 问题难解，即不存在 P.P.T.算法 A 能从任意的输入 $U(=g^x)$、$V(=g^y)$和 W 以显著偏离 1/2 的概率判定 $W \stackrel{?}{=} g^{xy}$。

设 $SIG=(KG_s, Sign, Vf)$是抗伪造的数字签名方案，$\Pi=(KG_e,E,D)$是保密的对称加密方案。*Diffie-Hellman* 协议的过程如图 12-1，其中(vk_A,sk_A)和(vk_B,sk_B)分别是 A 和 B 的签字公钥和私钥，并且假定 A 和 B 事先已从可信任的途径—例如公钥基础设施中的数字证书—获得了对方签字公钥[3](vk_B 和 vk_A)，f 是任何一种散列函数。

注意 *Diffie-Hellman* 协议仅指图 12-1 中的密钥交换阶段，接下来的保密会话阶段只是应用交换阶段所生成的密钥对数据进行对称加密，在这里把第二个阶段表示出来只是为了给读者一个完整的保密通讯图像，以后我们将要集中阐述的内容都属于密钥交换阶段，即图中的第一个阶段，这一阶段才真正属于密钥交换协议。

Diffie-Hellman 协议中的身份认证是双向的。A 首先向所期望的对方(B)发送自己的身份标识“A”；B 生成随机数 x、计算 $U=g^x$ 并生成数字签名 $\sigma_B=Sign(sk_B,U)$，然后向 A 发送身份标识“B”、数 U 和 σ_B；A 在接收到第二条消息后验证 σ_B 确实是 B 对 U 的数字签名，然后生成随机数 y、计算 $V=g^y$ 并生成数字签名 $\sigma_A=Sign(sk_A,V)$，A 还进一步计算出 $g^{xy}(=U^y)$并以此

[2] 这里讨论的协议比原始的 *Diffie-Hellman* 协议要完整的多，原始协议没有包含身份认证部分。实际上在 1976 年甚至还没有关于这类协议安全性的精确概念，目前我们所广泛接受的协议安全性概念出现于 80 年代末，但直到 90 年代末才基本定型。

[3] 今后凡涉及公钥方案(公钥加密、数字签名等)都做这一符合实际的假定,不再一一注明。

为密钥对字符串“0||A||B”做对称加密生成密文 $W_A=E(g^{xy}, 0||A||B)$，然后向 B 发送 V、σ_A 和 W_A；B 在接收到这一消息后验证 σ_A 确实是 A 对 V 的数字签名、计算出 $g^{xy}(=V^x)$并以此为密钥对 W_A 做解密计算、验证解密出的明文确实是“0||A||B”，由此判定当前与之会话的对方确实是 A，然后以 g^{xy} 为密钥对字符串“1||B||A”做对称加密生成密文 $W_B=E(g^{xy},1||B||A)$并将 W_B 发送回 A；最后，A 以 g^{xy} 为密钥解密消息 W_B 并验证解密出的明文确实是“1||B||A”，由此判定当前与之会话的对方确实是 B。在此之后，A、B 计算出 $f(g^{xy})$并以此为密钥对后续一切消息做对称加密，由此实现保密通讯。

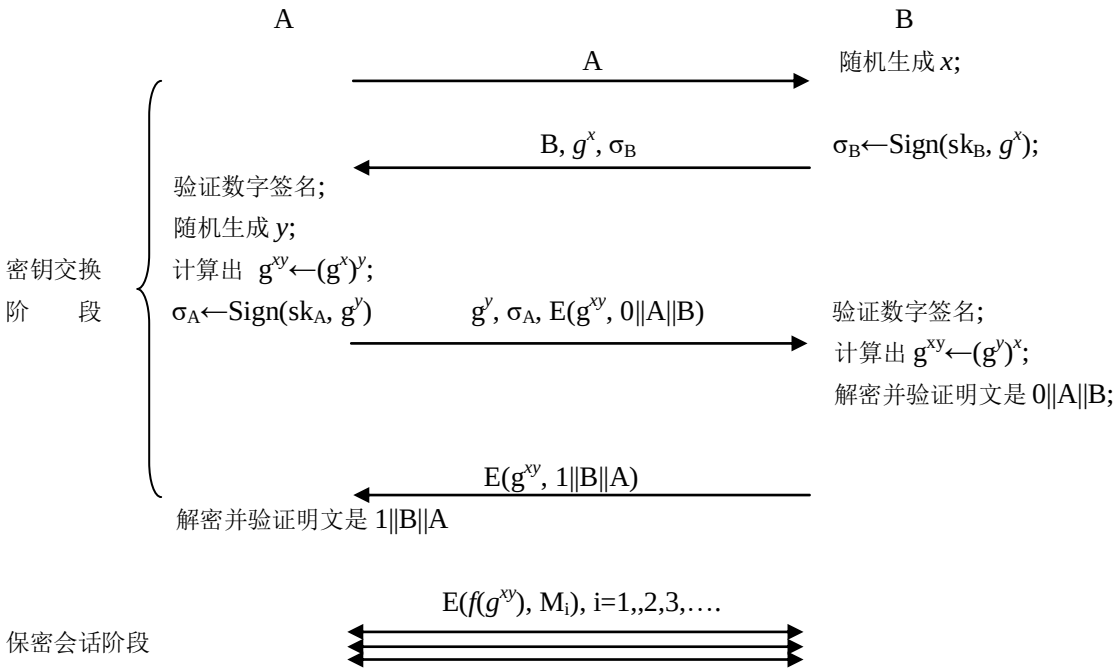


图 12-1 *Diffie-Hellman* 协议

细心的读者可能会注意到，以上协议的第三条消息和第四条消息中的密文分量 W_A 和 W_B(分别等于 $E(g^{xy}, 0||A||B)$和 $E(g^{xy}, 1||B||A)$)似乎“多余”，因为第二条消息和第三条消息中的数字签名分量 σ_B 和 σ_A 已经足以证明数 U 和 V(分别等于 g^x 和 g^y)确实源于 B 和 A，为什么还需要验证 W_A 和 W_B 被解密后的明文？答案在于：σ_B 和 σ_A 确实能够证明数 U 和 V 源于 B 和 A，但不能证明 U 和 V 是 B 和 A 当前所生成！通过验证 W_B 和 W_A 被解密后的明文确实具有所规定的内容，才能向 A 和 B 证实 W_B 和 W_A 的发送方持有正确的密钥 g^{xy}；注意这一密钥既与 A 当前生成的随机数 *y* 有关、也与 B 当前生成的随机数 *x* 有关，而且由于所选择的群上的计算性 *Diffie-Hellman* 问题难解(这是判定性 *Diffie-Hellman* 问题难解性的推论，应用一点反证法就能立刻明白)，一个盗用旧消息分量(例如旧的 g^x)的攻击者(未知 *x*)无法仅仅

根据当前观测到的 g^y 有效计算出正确的 g^{xy} 从而使 A 能够顺利通过这一检验。换句话说，两个密文分量 W_A 和 W_B 的目的是抑制第 9 章解释过的重放攻击，因此是不可或缺的。

12.3 *SIGMA* 协议

SIGMA[4]协议是一大类密钥交换协议的原型，最早发表于 1996 年，主要出于开发因特网安全解决方案的需要而设计，目前广泛应用的因特网安全协议族 IPSec(RFC#2401-2412)中的 IKE 协议就是以此为蓝本的设计。

12.3.1 主体协议

SIGMA 协议也是一类基于判定性 *Diffie-Hellman* 问题难解性的密钥交换协议。记号 prf 表示拟随机函数(在目前阶段读者将其想象为散列函数即可)，SIG=(KG_s,Sign,Vf)是抗伪造的数字签名方案，MAC=(KG_m, MAC, MVf)是抗伪造的消息认证码方案，循环群 G 以 g、为公开的生成子，G 上的判定性 *Diffie-Hellman* 问题难解。*SIGMA* 协议过程如图 12-2，其中 CA(A||vk_A)和 CA(B||vk_B)表示 A 和 B 的公钥证书（回顾第 8.3.3 节，公钥证书 CA(A||vk_A)本质上是可信任的公钥证书服务器 CA 对“A||vk_A”的一个数字签名，用来证实 A 确实持有与公钥 vk_A 对偶的私钥 sk_A；公钥证书 CA(B||vk_B)的解释类似)。

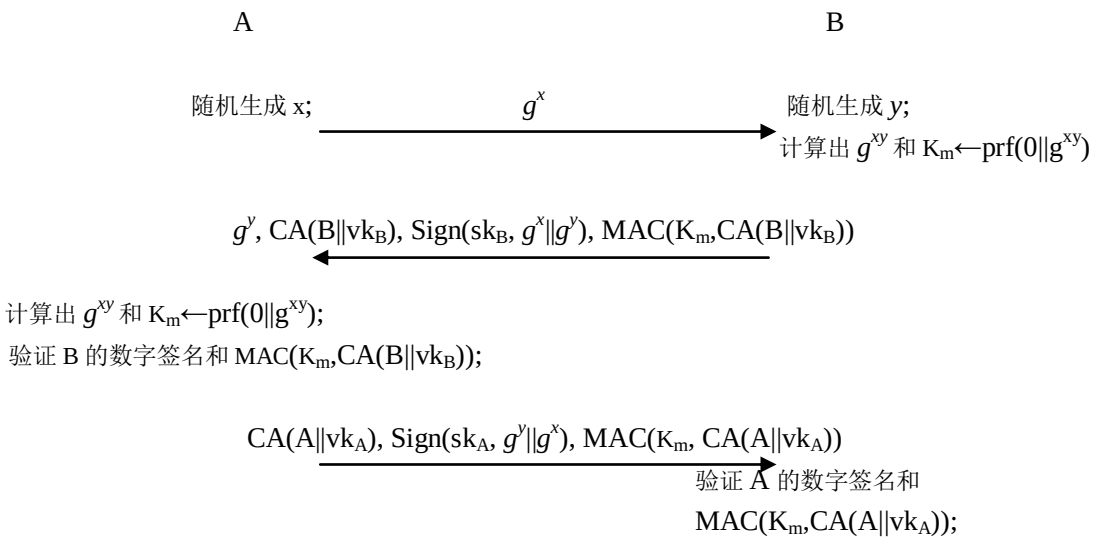


图 12-2 *SIGMA* 协议

[4] *SIGMA* 是 SIGn-and-MAc 的缩写，因为其中实质性地应用了数字签名方案(sign)和消息认证码方案(MAC)。

SIGMA 协议看上去较之 *Diffie-Hellman* 协议结构更为简洁。作为一个很好的练习，请读者仿上一节的方式自行详细解释协议过程。*SIGMA* 协议最终生成的会话密钥 Ks 是 $\text{prf}(1\|g^{xy})$，图中的 $K_m(=\text{prf}(0\|g^{xy}))$仅仅用做密钥交换阶段的临时 MAC-密钥。*SIGMA* 协议的计算效率和通讯效率都非常高，协议仅需要三条消息，而且除了数字签名算法需要比较大的计算量之外，所有其他计算一般都有很高效的实现方案。

与上一节的协议不同，这里公证书直接出现于消息之中，目的是针对一类特殊应用模式，即协议参与方之一事先未必总能明确当前需要与之对话的对方。例如，在大规模分布式计算系统中，一个客户进程 A 需要请求特定的服务时，未必总按照事先的安排访问特定的服务器(B)，而是在网络环境中广播请求，任何能够提供服务的、可信任的服务器都可以成为协议的对方；又例如在无线移动网络环境中，A 需要与访问接入点或基站建立协议会话，这时 A 尤其不可能事先明确究竟将和哪个对方(B)进行协议会话，只能在线认证对方的身份。既然 A 不能事先明确 B 究竟是“谁”，因此只能在第 2 条消息中待明确看到身份标识“B”之后根据 B 提供的公钥证书 $\text{CA}(\text{B}\|\text{vk}_\text{B})$来验证 B 的数字签名 $\text{Sign}(\text{sk}_\text{B}, g^x\|g^y)$。对 B 也存在同样的例子。

这里需要注意的是两个消息认证码 $\text{MAC}(K_m,\text{CA}(\text{A}\|\text{vk}_\text{A}))$和 $\text{MAC}(K_m,\text{CA}(\text{B}\|\text{vk}_\text{B}))$的作用：为什么需要对协议双方的公钥证书用与特定会话有关的密钥 K_m 进行完整性保护？

首先解释 $\text{MAC}(K_m,\text{CA}(\text{A}\|\text{vk}_\text{A}))$ 的作用。假如第 3 条消息不包含分量 $\text{MAC}(K_m,\text{CA}(\text{A}\|\text{vk}_\text{A}))$，一个攻击者 P 在 A 接收到第 2 条消息后向 B 发送第 3 条消息 $\text{CA}(\text{P}\|\text{vk}_\text{P})$, $\text{Sign}(\text{sk}_\text{p}, g^y\|g^x)$，于是 B 据此判定当前与之对话的对方是 P(而非 A)，但同时 A 在接收到第 2 条消息后却判定当前对方是 B，A、B 双方对对方身份的判别不一致，破坏了协议应该满足的安全性质之一(一致性)。或许有读者觉得该攻击似乎无害，毕竟密钥 Ks 并未泄露，P 并不能破译 A 或 B 接下来将要发送的明文消息。但实际上请读者想象以下应用情形：B 是服务器、A 是客户，A 与 B 协商密钥的目的是为了向 B 转帐。在以上攻击发生后，A 开始向 B 发送以刚刚生成的会话密钥 Ks 加密的帐务数据用以支付某种款项，而 B 却判定该支付行为来自 P(而非 A！)，尽管 P 对消息内容无从得知，但实际效果如何？

接下来解释消息分量 $\text{MAC}(K_m,\text{CA}(\text{B}\|\text{vk}_\text{B}))$的作用，这里涉及一类特殊的攻击，即所谓*证书替换攻击*。假如第 2 条消息不包含分量 $\text{MAC}(K_m,\text{CA}(\text{B}\|\text{vk}_\text{B}))$，这时一个攻击者 P 在看到来自 B 的消息 g^y, $\text{CA}(\text{B}\|\text{vk}_\text{B})$, $\text{Sign}(\text{sk}_\text{B}, g^x\|g^y)$后可以向公钥证书服务器 CA 请求一个包含自己的名字 P 和 B 的签字公钥 vk_B 的证书 $\text{CA}(\text{P}\|\text{vk}_\text{B})$，P 将这一(虽然并非伪造但却不正确的)公钥证书替换原来消息中的 $\text{CA}(\text{B}\|\text{vk}_\text{B})$，然后将 g^y, $\text{CA}(\text{P}\|\text{vk}_\text{B})$, $\text{Sign}(\text{sk}_\text{B}, g^x\|g^y)$作为第 2 条消

息发送到 A；A 在接收到这一消息后，因为能够正确验证“P”的数字签名 $\mathrm{Sign}(sk_B, g^x||g^y)$从而判定当前的对方“确实是 P”，而实际上对方并非 P 而是 B，破坏了协议应该满足的一致性！

就 P 在线欺骗证书服务器而言，这一攻击显得有些理想化，读者有理由怀疑在实际中是否能够行得通。但无论如何以上分析表明这是一个应该避免的安全漏洞。读者现在可以分析，如果第 2 条消息带上分量 $\mathrm{MAC}(K_m,\mathrm{CA}(B||vk_B))$，P 的以上攻击还能凑效吗？因为 P 无法有效计算出 K_m，所以不能有效伪造出 $\mathrm{MAC}(K_m,\mathrm{CA}(P||vk_B))$，这样一来以上证书替换攻击就不再有效。

从这些分析可以看出，$\mathrm{MAC}(K_m,\mathrm{CA}(A||vk_A))$和 $\mathrm{MAC}(K_m,\mathrm{CA}(B||vk_B))$起着对公钥证书的在线签字作用，用以向对方证实自己确实是证书的持有者，为此两个分量都是不可或缺的。

12.3.2 匿名的变体

SIGMA 协议有许多变体，这里概要阐述其中的匿名变体，这类协议在新型网络环境，例如无线/移动网络中具有十分广泛的应用价值。

上一小节曾经提到，*SIGMA* 协议有能力针对一类特殊应用模式，即协议参与方之一事先未必总能明确当前需要与之对话的对方。这时一个常见的需求是协议的某一方不先行暴露自己，仅在确认对方具有可信任的身份之后再向对方出示自己的身份。实际上，图 12-2 中的 *SIGMA* 协议已经具有这一特点：B 先向 A 出示自己的身份(第 2 条消息)，而 A 仅在认证了 B 的身份之后才向 B 表明自己的身份(第 3 条消息)。在实际应用中，这给了协议发起方 A 一个机会，一旦对方不属于自己所信任的实体，A 将立刻终止协议而不向 B 暴露自己的身份。这就是*第一类匿名性质，即协议的参与方之一总是在另一方的身份被完全认证之后才出示自己的身份*。这一性质也称为后验身份认证性质。图 12-2 中的 *SIGMA* 协议就使 A 具有对 B 的后验身份认证性质。

也可以设计出使 B 具有对 A 的后验身份认证性质的 SIGMA 协议，即 A 现行出示自己的身份，在完全认证 A 的身份之后，B 才出示自己的身份，从而给予 B 保护自己身份的机会。图 12-3 就是这样一个 SIGMA 协议，其中各个符号的涵义与图 12-2 完全一样。作为练习，请读者详细描述进程 A、B 的处理动作，以及 $\mathrm{MAC}(K_m, \mathrm{CA}(A||vk_A))$和 $\mathrm{MAC}(K_m, \mathrm{CA}(B||vk_B))$的作用。

注意，如果要保证抗身份欺诈性质，则同时保证协议双方都具有后验身份认证性质是不可能的。

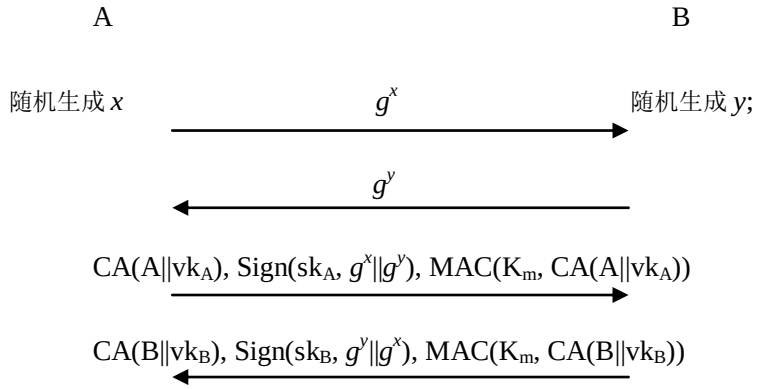


图 12-3 第一类匿名的 *SIGMA* 协议: B 后验身份认证

还有第二类匿名性质，即协议双方对任何非参与方完全匿名。这一要求在当代无线网络环境中是非常现实的：通讯双方不仅需要保密其传输的数据，而且经常期望不向任何第三方暴露自己的身份，以免暴露自己的行踪。*SIGMA* 协议略加修改就可以满足这一要求，办法是对含身份标识的消息进行对称加密，而加密密钥来自当前协议会话生成的随机数。这样，仅仅协议的合法参与方才能解密从而"看到"身份标识，这就是图 12-4 的 *SIGMA* 协议，其中符号$\{M\}_K$表示用 K 做密钥对括号中的消息 M 做对称加密，其他符号与图 12-2 完全相同。

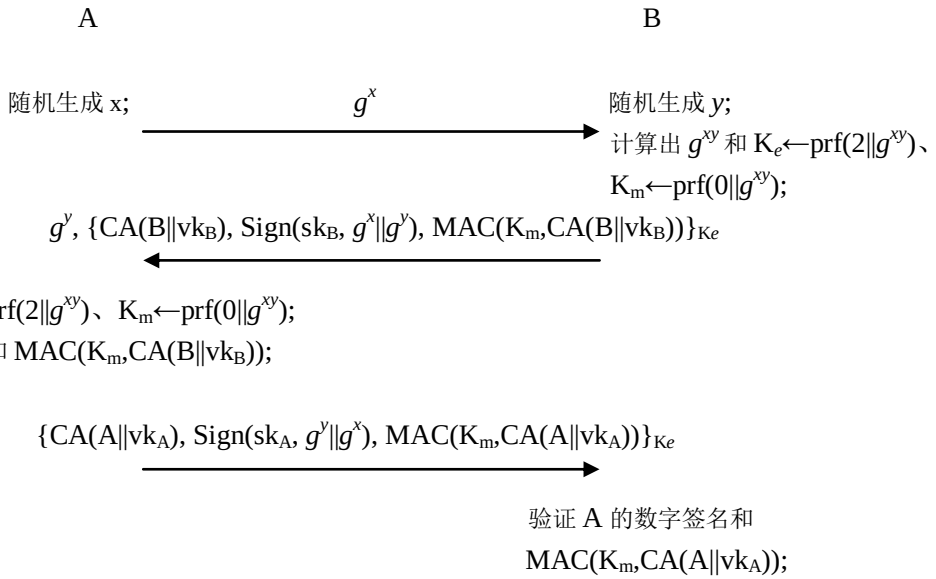


图 12-4 第二类匿名的 *SIGMA* 协议: A、B 对第三方匿名

所有以上协议最终生成的会话密钥 Ks 都是 prf(1||g^{xy})[5]。

[5] 实际协议导出会话密钥的方式比这里更为复杂,目的是为了满足所谓密钥的前向保密性质, 粗略地说就是

12.3.3 完整的协议实例：SIGMA-R

为了使我们的阐述最终接近现实，这一小节给出 *SIGMA* 协议的完整的实际版本之一(常称为 *SIGMA-R*)，协议过程见图 12-5。这一协议不仅具有 12.1 节描述的所有安全性质，而且还具有 B 对 A 的后验身份认证性质(第一类匿名)，并且 A、B 对任何第三方完全匿名(第二类匿名)。

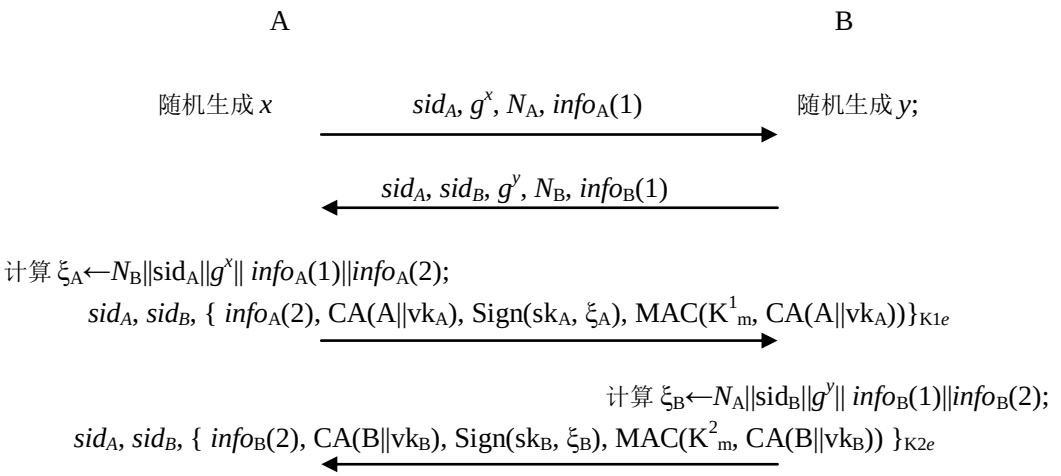


图 12-5 *SIGMA-R* 协议

请读者自行详细描述协议进程 A、B 的动作。较之前面的协议，协议 *SIGMA-R* 增加了一些实际应用中需要用到的元素。sid_A、sid_B 分别是由 A、B 随机生成的会话号，每条消息都带上该编号，这些编号还出现在数字签名中，目的是使当 A、B 之间并发存在多个同类协议会话时，总能由 sid_A 和 sid_B 共同唯一确定每个消息属于哪个会话实例，以避免并发交错攻击(例如第 9 章描述的对 Woo-Lam 协议的攻击)。N_A、N_B 分别是由 A、B 随机生成的随机数，目的是使得每个协议会话都具有唯一的 $N_A||N_B$，把这种唯一的随机数结合进协议消息以抵抗重放攻击。生成的密钥 Ks 也与这两个随机数有关，目的是使密钥更加不可预测。$Info_A(1)$、$info_A(2)$、$info_B(1)$和 $info_B(2)$是一些信息块，根据具体应用情形赋予特定的内容。密钥交换阶段用来实现对第三方匿名和验证消息认证码的对称密钥一共有 4 个：K^1_m、K^2_m、K^1_e、K^2_e，每个方向一个而不是象原来那样两个方向共享一个，这样做也是为了提高攻击的难度。所有

密钥的泄露不影响以前数据的保密性，对此章后列举的 Krawczyk 的研究论文、IKE 的因特网标准 RFC#2409 和 TLS/SSL 的因特网标准 RFC#4346 都有详细描述，这里不再详细阐述了。

密钥，包括会话密钥都从主密钥 g^{xy} 导出，具体算法不在此详述了。

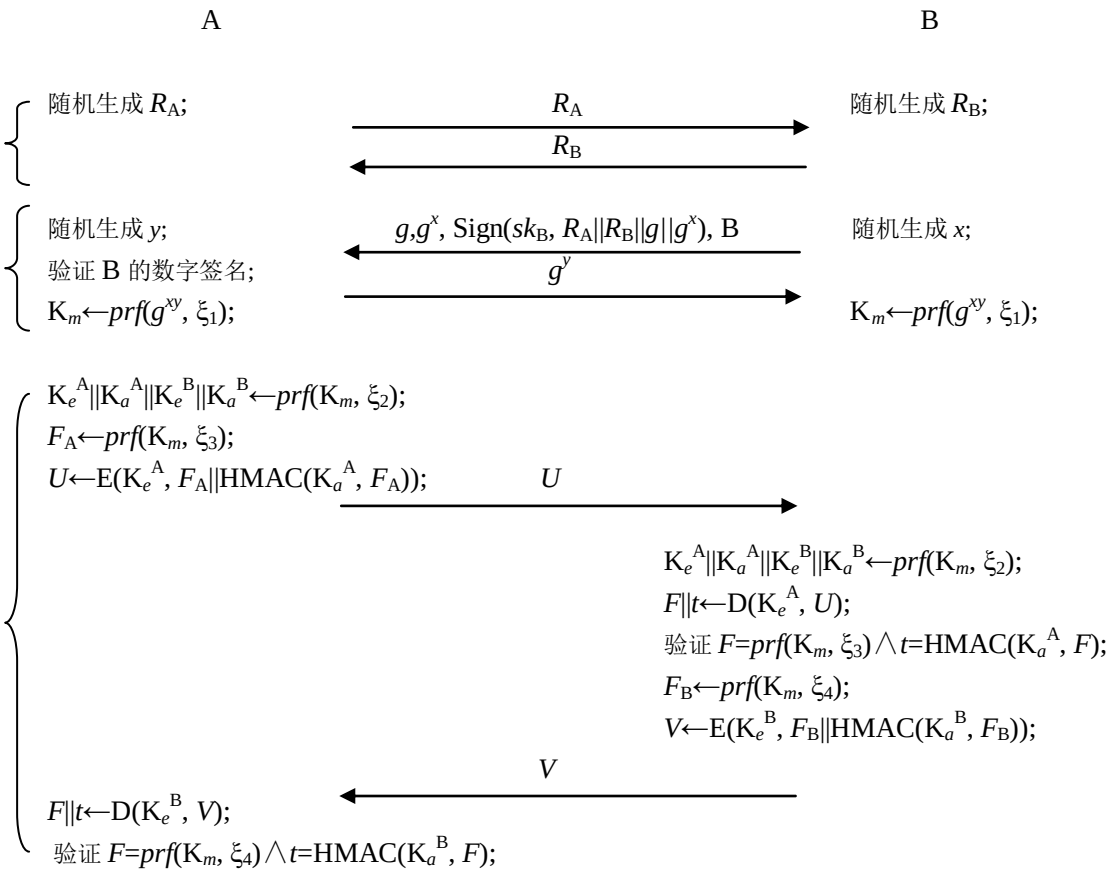
12.4 *SSL/TLS* 协议

SSL/TLS 是两个非常相似的协议，广泛应用于浏览器/*Web* 服务器之间的安全通讯。早期的安全套接字协议 SSL 源于 Netscape 的设计，后来成为因特网标准，即所谓传输层安全协议 TLS。SSL/TLS 包含一族子协议，其中握手子协议用来在双方之间实现带身份认证的密钥交换，也是最复杂的一个子协议；记录协议应用握手协议生成的会话密钥对经过 TCP 连接进行传输的数据进行保密和完整性保护(防篡改)；告警子协议用来报告协议会话中的异常事件，切换子协议用来使一方通告对方开始启用新的保密通讯参数。后面这两个子协议非常简单，而记录协议则完全在会话密钥生成之后执行数据保密，只有握手协议才是真正意义上的带身份认证的密钥交换协议。

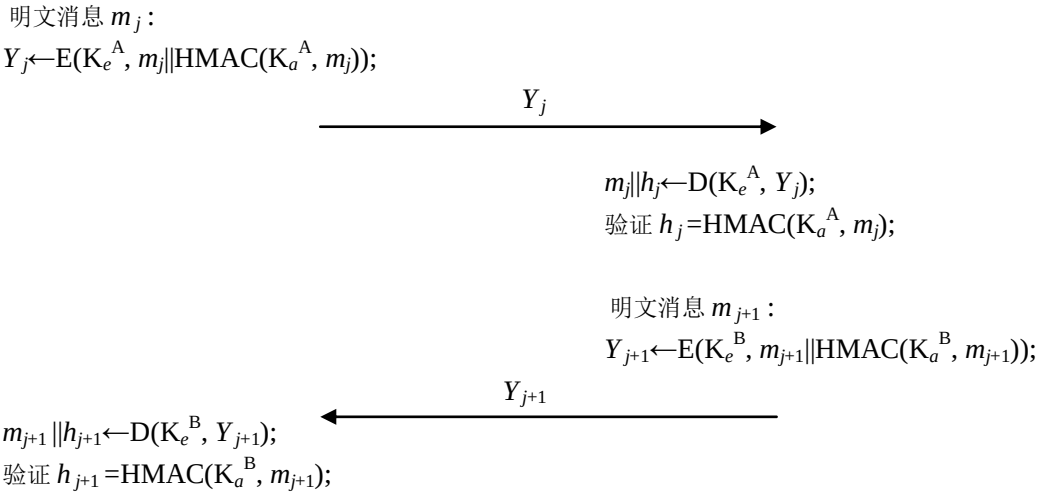
SSL/TLS 协议过程如图 12-6，其中(a)是握手协议，(b)是记录协议。图中的符号 E 和 D 分别表示某个选定的对称加密方案的加密算法与解密算法，*prf* 表示某个拟随机函数，HMAC 表示某种消息认证码生成算法，Sign 是选定的数字签名方案的签字算法。

握手协议的会话过程可以被划分成几个明确的阶段，在第一个阶段双方交换两个随机数 R_A 和 R_B，第二个阶段的目的是生成预共享密钥 K_P 并以此为基础导出主密钥 K_m，第三个阶段的目的是彼此认证对方的身份。图中 ξ_1～ξ_4 是一些特定的字符串，一般来讲包含到当前为止已经交换过的协议消息，即当前的协议上下文，具体细节不在这里深入讨论了。握手协议最终生成 4 个密钥 K_e^A、K_a^A、K_e^B 和 K_a^B，其中 K_e^A 和 K_e^B 分别用来从 A 向 B 和从 B 向 A 加密传输数据，而 K_a^A 和 K_a^B 则分别用于保证从 A 向 B 和从 B 向 A 方向的数据完整性，即防止数据被篡改。

需要指出的是，图 12-6 中 *SSL/TLS* 协议的第二阶段有许多可选方案，图中描述的只是其中一种，称为 *Diffie-Hellman* 预共享密钥交换方案，相应的预共享密钥 $K_P=g^{xy}$。除此而外还有其它许多可选方案，例如预共享密钥分发方案，这时 B 向 A 发送自己的身份标识“B”或其加密公钥的数字证书 $CA(B||pk_B)$，而后 A 生成预共享密钥 K_P 并以密文 $E^a(pk_B, K_P)$的形式发送到 B，这里 E^a 是某个公钥加密方案的算法，而后 A、B 双方都以公式 $K_m \leftarrow prf(K_P, \xi_1)$ 计算出主密钥 K_m。注意这种方案中的密钥基本上由协议的一方(A)完全确定，而非由协议双方共同生成。



a) SSL/TLS 握手协议



b) SSL/TLS 记录协议

图 12-6 SSL/TLS 协议

在实际应用中，*SSL/TLS* 协议需要底层的通讯机制具有面向连接的特点，因此在因特网上几乎总是用来保护通过 TCP 连接传输的数据，特别是 HTTP 协议的会话信息，这一点与用于因特网 VPN 的 *IKE* 协议不同，后者用来协商用以任何通讯机制的会话密钥，例如直接

通过 IP 协议传输的数据，而非限于 TCP 协议。有趣的是，由于 *SSL/TLS* 协议与 HTTP 协议结合如此紧密，因此出现过 HTTPS 协议，即所谓安全的 WWW 协议，它在本质上其实就是 HTTP+SSL/TLS，为此专门负责因特网命名分配的 IANA 组织还特别约定了一个公共的 TCP 端口号 443。

12.5 VPN 与 IPSec 协议

到目前为止我们已经学习了多种网络安全技术和几类最典型的网络安全协议，这一节以 VPN 为例概要阐述它们的实际应用。

12.5.1 VPN

所谓 VPN 是指虚拟私有网(*Virtual Private Network*)或虚拟专用网，它在不可信任的公共网络设施—例如因特网—上通过安全协议等技术途径实现完全可信任的逻辑通讯环境，其效果就好象在完全私有的网络环境上一样。这些安全技术包括数据保密、完整性保护、身份认证、访问控制和会话密钥在线协商等。从实际应用的需求来讲，很多企业和组织因为地域分布广阔，最典型的如大规模跨国企业，只能借助于因特网这类公共通讯系统建立企业网，连接不同地域的企业局域网，包括延伸到移动用户， VPN 对这类用户是一个现实的解决安全通讯问题的技术方案，如图 12-7。因为公共网络上的安全保证完全由边界网络设备上的协议软件实现，因此相对于建立物理专有网络而言，实施 VPN 能够其用户节省大量费用，而且访问灵活，通过配置不同的安全策略就能使之适应不同的应用需求。

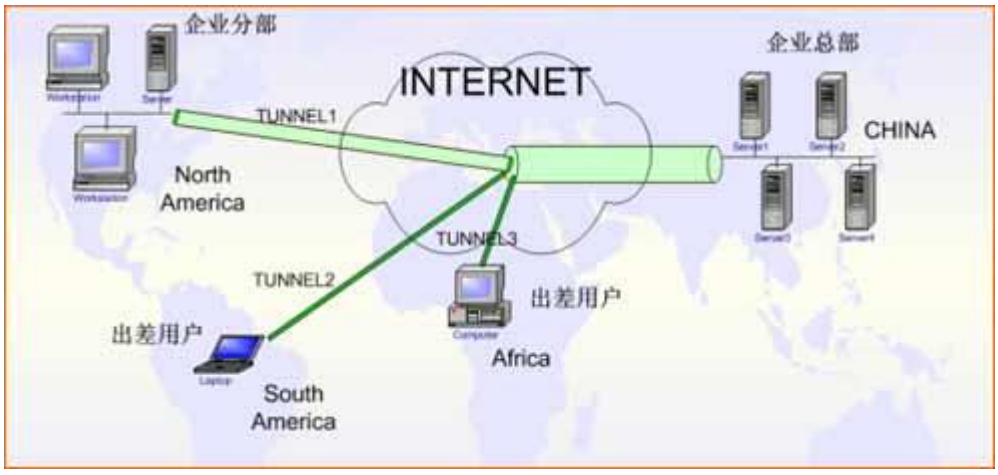


图 12-7 从用户观点看到的 VPN

从使用者的观点看，VPN 分为内联网 VPN、外联网 VPN 和远程接入 VPN。远程接入 VPN 在远程用户或移动用户和企业内部网之间建立虚拟专用连接[6]。典型的工作方式是用户拨号到网络访问服务器(NAS)，发出 PPP 连接请求，NAS 收到呼叫后，在用户和 NAS 之间建立 PPP 链路，然后 NAS 对用户进行身份验证、与总部内部网连接以访问企业网络内部资源。

内联网 VPN(i-VPN)通常指连接企业远程分支机构 LAN 和总部 LAN 之间的 VPN，这种连接虽然实际上穿越了因特网这一公共网络环境，但完全依靠软件实现，毋须建立数字专线(DDN)，避免了由此所带来的高额费用。

外联网 VPN(*e*-VPN)在不同组织的网络之间建立可信任的连接，使第三方不能获取这类网络上传输的明文消息。最典型的例子如企业与其供应商、商业合作伙伴的 LAN 之间的虚拟私有连接。

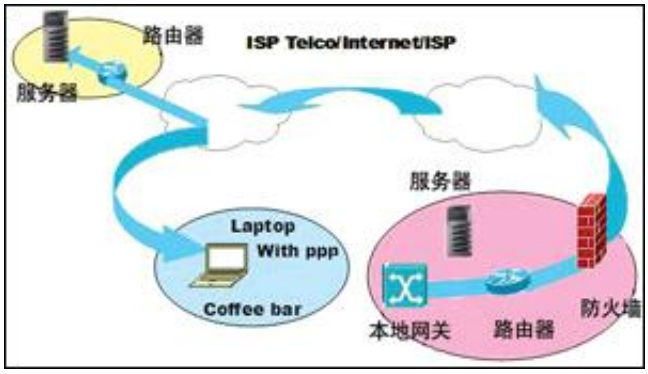


图 12-8 远程接入(拨号)VPN

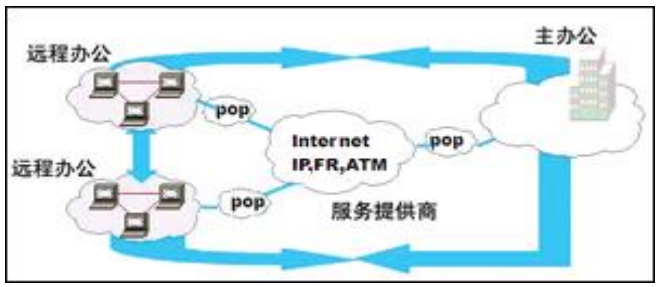


图 12-9 内联网 VPN

[6] 这里的词汇“连接”都是指逻辑通讯链路，是一个很宽泛的概念，并非特指 TCP 连接，两者不要混淆。

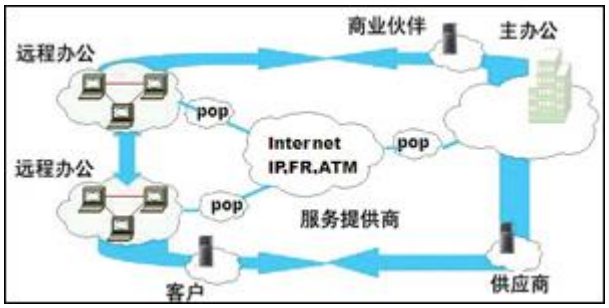


图 12-10 外联网 VPN

VPN 也可以按照部署模式分类，这里所谓部署模式从本质上讲是指 VPN 逻辑连接的起点和终点。这时 VPN 可以分为端到端(end-to-end)模式和供应商-企业(Provider-Enterprise)模式。不同的模式适用于不同的应用环境。

端到端模式是最为彻底的 VPN 网络。在这种模式中，企业对 VPN 上的安全策略具有完全的控制权，整个网络上的数据通讯都在加密和完整性保护的逻辑隧道中完成，非常可靠。但这种部署模式对 VPN 的拥有者的技术能力要求高、投资大。

供应商-企业模式形象地说是一种外包 VPN，通常由缺乏足够技术能力的用户(例如小企业)委托其因特网接入服务商(ISP)建立。与端到端模式不同，在这里客户本身不需要购买专门的隧道边界设备和软件，完全由 ISP 来建立可信任的接入通道并实施接入安全策略。

12.5.2 *IPSec* 协议族

上一小节主要从使用者的观点阐述 VPN 的概念及类型，但无论哪种类型的 VPN，其技术性的本质都在于建立一种可信任的端到端的网络层，只要两台主机 H_1、H_2 接入 VPN 则在 H_1、H_2 之间传输的一切消息—IP 分组承载的数据甚至 IP 分组本身—都可以按照指定的方式被保密和认证。这当然蕴涵着实施 VPN 的所有主机之间在基于 IP 协议通讯之前都(自动地)进行身份认证和会话密钥协商，而且任何这类协商都按照事先配置的网络安全策略进行。宏观地看，VPN 可以视为公共因特网上的一群可信任的安全区域，每个区域实施一组特定的针对 IP 分组的数据保密、认证及访问控制策略。IPSec 协议族就是针对因特网实现这一安全目标的具体技术。

图 12-11 表示出 IPSec 协议族中各个子协议及其相互关系，其中最主要的子协议是认证首(AH)协议和封装安全荷载(ESP)协议，以及因特网密钥交换协议(IKE)。

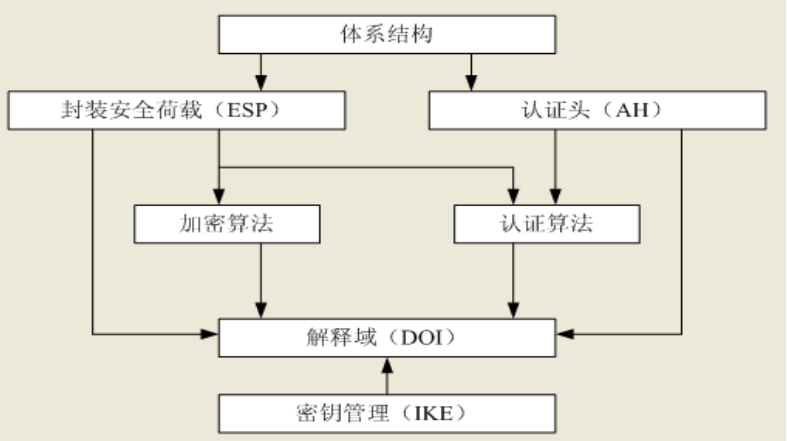


图 12-11 IPSec 协议族的体系结构

IPSec 提供的基本安全服务包括数据保密和数据完整性保护。IPSec 使 IP 分组在离开始发计算机之前就被加密，直到到达目标计算机之后才被解密，这就是端到端加密。数据完整性保护机制使计算机在收到 IP 分组时可以验证其中的数据是否被篡改过。

IPSec 可以在主机/主机之间、安全网关/路由器之间、主机/安全网关之间或主机/路由器之间实施。实施的模式有两种：传输模式和隧道模式。传输模式用于保护 IP 分组所承载的上层协议数据，这种模式在原始分组的 IP 首部与上层协议的消息单元(例如 TCP 段、UDP 消息)之间插入一个 IPSec 首，但整个 IP 分组仍以传统方式在因特网上被路由/转发。隧道模式用来保护整个 IP 分组的内容，包括原始 IP 分组的首部和数据载荷部分，具体做法就如同把已经装有信件的信封套装在另一层信封里那样：VPN 隧道的起始端(一台主机或边界网关)在原始 IP 分组的外部添加 IPSec 首，加密和认证原始 IP 分组，然后再附加一个外部 IP 首来封装整个被处理过的分组。被附加的外部 IP 首可以有独立于原始 IP 分组的始发 IP 地址和目标 IP 地址。这样一来，整个 IP 分组在因特网上就被完全按照外部 IP 首的目标地址(而非原始 IP 分组的目标地址)被转发和路由，形象地说就好象原始 IP 分组沿着一条由外部 IP 首地址导向的隧道传输一样，直到该隧道的出口—另一台主机或边界网络—原始 IP 分组才被重新恢复出来，而在沿隧道传输期间。这种模式的最大好处是使原始 IP 分组在因特网上完全不可见：不仅数据载荷、而且原始 IP 分组的首部信息都不可见，从而不仅保密数据而且抵抗流量分析[7]。两种模式的 IPv4 分组的形式如图 12-12 所示。

[7] 流量分析(Traffic Analysis)通过观测诸如特定 IP 地址或端口之间的信息流量的空间或时间分布来推断某些信息, 是一种常用的攻击手段。

图 12-12 不同传输模式下的 IP 分组(以承载 TCP 段的 IP 分组为例)

在接下来进一步描述 IPSec 之前，需要解释 IPsec 的两个重要概念，这就是安全关联(Security Association，简称 SA，中文名称也译为安全联盟)和安全策略(Security Policy，简称 SP)。

SA 是两个通讯实体经协商建立起来的一种协定，协定内容主要针对 IP 分组的安全保护机制，例如数据的加密/解密算法、会话密钥、随机数、身份认证算法和对应的参数等。一个计算机上可以有任意多个 SA，每个 SA 对应一个计算机到另外一个特定计算机的数据传输。因为一个传输方向只关联一个 SA，因此两个计算机之间的安全通讯需要保持两个 SA，每个方向对应一个。形式地说，SA 由三个元素描述，第一个元素是所谓安全参数索引(SPI)，这是一个 32 位 2-进制数，用于唯一标识同一个机器上的多个 SA 之一，它通常出现在子协议 AH 或 ESP 的 IPSec 首部之中；第二个元素为目标 IP 地址，它是 SA 的终点地址，该终点可以是终端用户系统或防火墙、路由器这样的网络系统，后两者一般是作为隧道的一端；第三个元素是安全协议标识符，表示哪一个子协议使用该 SA，或等价地说，该 SA 应用于哪一个子协议。目前的 IPSec 子协议指 AH 或 ESP 之一，分别用以数据完整性保护和数据加密，详见下面的阐述。注意每个 SA 只能应用于 AH 或 ESP 之一，不能同时应用于两者。如果安全策略需要同时应用 AH 和 ESP 两者，则需要(按指定的顺序)使用两个不同的 SA。在实现 IPSec 时，所有的 SA 都记录在 SA 数据库(SADB)内。

SP 是 IPSec 的另一个重要组成部分，它定义两个运行 IPSec 的机器之间的安全通信特征，并决定在该通信中为所有 IP 分组提供的安全服务。一个运行 IPSec 的机器的所有 SP 都记录在安全策略数据库(SPDB)中，根据 SP 选择符(selector，包括源 IP 地址、目标 IP 地址、上层协议、传输层端口号等属性)进行检索。

SP 由 VPN 管理者具体配置，它们定量反映实施 VPN 的系统需求和目标。SA 由密钥交换协议 *IKE* 动态生成，但生成什么样的 SA 由 SP 决定。两台机器之间的 SA 一旦生成，它

们之间所有 IP 分组的传输都接受相应的 SA 的控制。

现在可以解释 IPSec 的两个重要的子协议：用以实现 IP 分组的数据完整性保护的 AH 协议和用以 IP 分组数据保密的 ESP 协议。这两个协议都有其特定的首部，统称为 IPSec 首，在具体情形则称为 AH 首或 ESP 首。

AH(Authentication Header)协议把含密钥的 HMAC 散列值附加到 IP 分组上，以此实现数据完整性、数据源认证和抗重放攻击。AH 的认证范围包括了 IP 首部中所有那些在传输过程中不发生变化的字段，例如 IP 地址、分组长度等。注意 AH 仅仅实现完整性保护而非加密，因此原始 IP 分组中的所有数据都公开可见。AH 首位于 IP 首和被保护的数据载荷之间(图 12-13(a))。

a) 受 AH 保护的 IP 分组

b) AH 首

图 12-13 AH 协议

AH 首部结构如图 12-13(b)，包括下一个首部的类型号、荷载长度、保留位、安全参数索引、序列号和认证数据诸字段：下一个头(8 比特)标识紧跟 AH 首的下一个首部类型，也就是紧随 AH 首的下一个协议；载荷长度(8 比特)是以 32-位字为单位的 AH 首的长度再减去 2 的值；保留位(16 比特)全为 0；安全参数索引(32 比特)用于标识一个 SA，它就是控制该 AH 协议的那个 SA；序列号(8 比特)是一个单调增加的计数器值；认证数据可变长但值为 4 的整数倍，包含该 IP 分组的 HMAC 散列值。

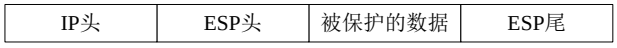
AH 使用的 HMAC 消息认证算法基于 MD5 和 SHA-1，它们都属于迭代型的杂凑算法，具体细节在 RFC#2402-2404 三个因特网标准中有详细描述。

ESP(Encapsulating Security Payload)协议也表现为插入 IP 分组的一个 IPSec 协议首。ESP 对 IP 分组的数据载荷进行加密，然后对密文进行 HMAC 散列，从而不仅实现数据保密而且实现了数据完整性、数据源认证和抗重放攻击。ESP 的加密与认证算法由 SA 决定，而抗重放攻击通过 IP 分组的接收方检验 ESP 首中的一个单向递增的序列号来实现。

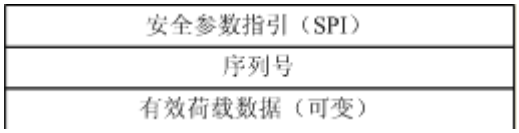
ESP 应用时在 IP 首和被保护的数据之间插入一个 ESP 首，同时在被保护的数据后附加加一个 ESP 尾(图 12-14)。ESP 首包含安全参数索引字段，用来标识作用于该 ESP 的 SA；

序列号，用于抵抗重放攻击；有效荷载数据加密的初始化向量，用于启动 ESP 加密计算。注意 ESP 首的所有字段都是不加密的，因为在解密过程中首先需要获得它们的值。ESP 尾包括填充项(某些加密算法都要求被加密的数据长度是某个数的整数倍，若数据长度不满足这个要求，就需要在后面附加特定的填充项)、填充项长度、下一首部类型、数据密文的认证数据(参见 8.4.1 小节后半部分加密方案的例子)。

IPSec 的因特网协议标准 RFC#2405-2406 详细描述了 ESP。

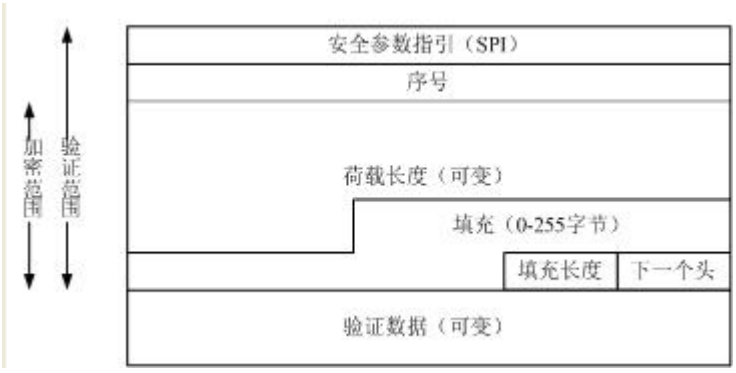


a) 受 ESP 保护的 IP 分组



b) ESP 首

c) ESP 尾



d) 完整的 ESP 格式

图 12-14 ESP 协议

以上解释的子协议 AH 和 ESP 都是在 SA 生成之后才起作用。对 IPSec，履行通讯双方身份认证和动态生成共享密钥的协议是 IKE。前面在讨论 *SIGMA* 协议时已经指出过，IKE 协议本质上就是 *SIGMA* 协议的变体，详细描述参考因特网标准 RFC#2408、RFC#2409 和 RFC#2412，其中 RFC#2409 是对 IKE 本身的详细规范，这里不再向其细节深入了，仅给出图 12-15 刻画其运行模型。图 12-16 是在 *IKE* 协商完成之后，也就是输入/输出方向的 SA 被明确建立之后，IP 分组的发送和接收的处理流程，从图中可以清楚地看出安全机制是怎样被包括进去的。

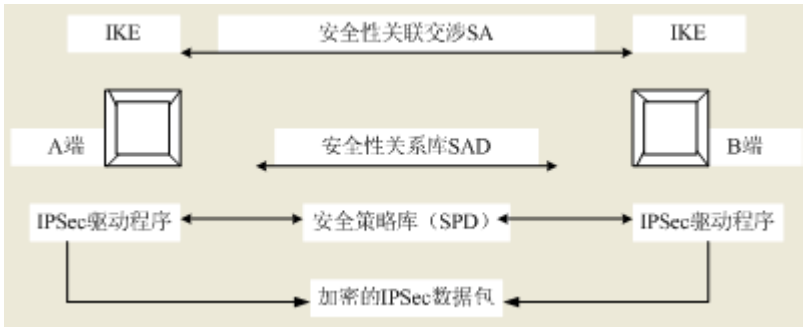
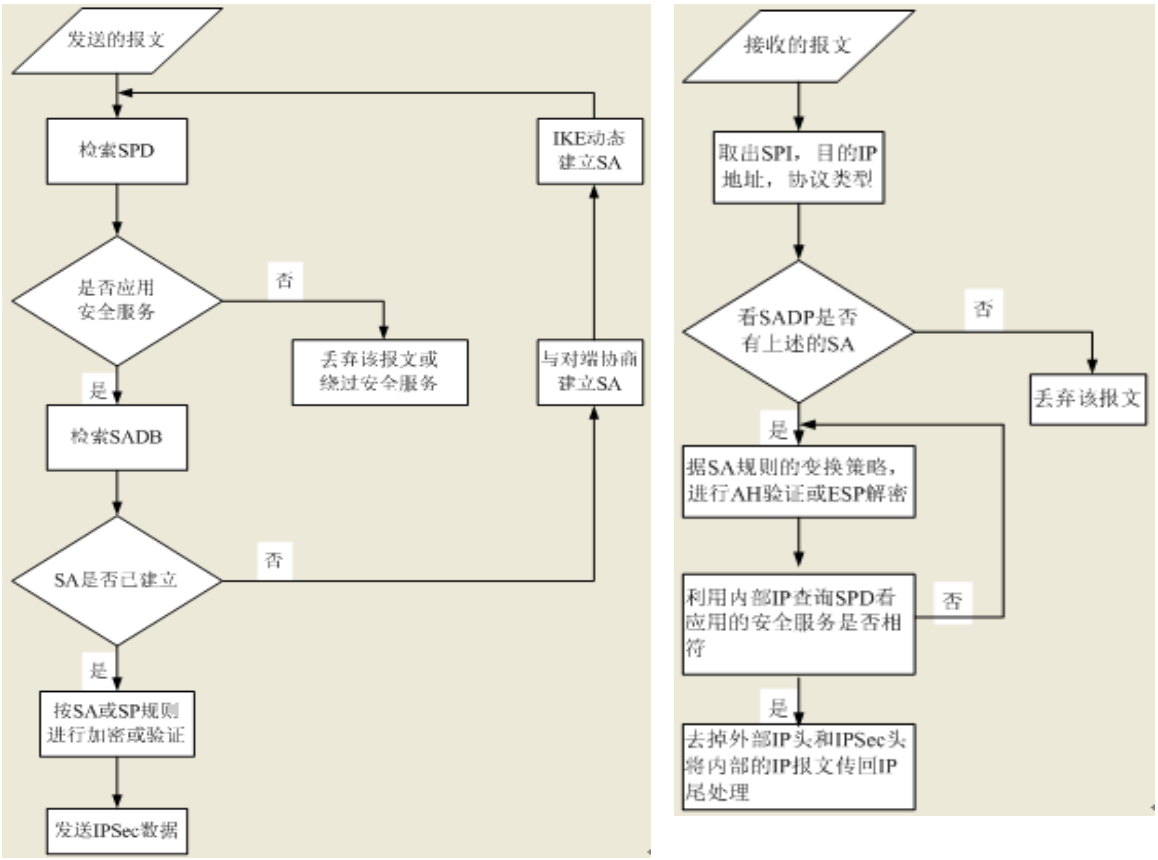


图 12-15 IPSec 运行模型



a) 发送 IP 分组 b) 接收 IP 分组

图 12-16 IPSec 对 IP 分组的输入/输出流程

12.6 小结与进一步学习的指南

带身份认证的密钥交换协议是使用最广泛的基础性安全协议，其完整的安全性质包括抗身份欺诈、一致性和密钥保密性，三者必须同时满足，缺一不可。此外，针对某些特殊应

用还可能要求协议具备附加的安全性质，例如匿名性质。本章阐述了几个最典型、也是应用最广泛的协议实例，特别分析了其中某些消息分量的重要作用。作为综合应用，最后较详细地解释了在因特网上实现虚拟私有网的技术途径，即 IPSec 协议族的工作机理。

关于 *SIGMA* 协议，其作者 Krawczyk 发表于 2003 年的总结性论文有非常透彻而又深入浅出的精彩阐述，值得一读：

H.Krawczyk *SIGMA: The "SIGn-and-MAc" Approach to Authenticated Diffie-Hellman and its Use in the IKE Protocols*.

下面这篇论文对理解 *SIGMA* 协议的设计思想也有很好的启发，这篇论文以著名的 STS 协议[8]为例详细讨论了包括证书替换攻击在内的许多攻击途径，此外 Krawczyk 论文的某些方面需要借助于这篇论文来深入理解：

S.Blake-Wilson, A.Menezes *Unknown Key-share Attacks on the Station-to-station*(*STS*) *Protocol*, Proc. PKC'99, LNCS Vol.1560, 154-170, 1999.

SIGMA 协议的完整的安全性证明在下面这两篇论文中建立，它们已成为网络安全协议方面的经典工作：

R.Canetti, H.Krawczy *Analysis of Key-exchange Protocols and Their use for Building Secure Channels*, Advances in Cryptology-Eurocrypt 2001, LNCS Vol. 2045, 453-474, 2001.

R.Canetti, H.Krawczy *Security analysis of IKE's Sigature-based key Exchange Protoco*l, Advances in Cryptology, LNCS Vol. 2442, Full version at eprint.iacr.org/2002/120.

对因特网密钥交换协议 *IKE* 的完整阐述包括三个标准文件：描述协议通用消息框架 ISAKMP 的 RFC#2408、描述 IKE 协议本身的 RFC#2409 和其中重要密码算法的 RFC#2412，所有这些材料都可以从 www.ietf.org 上得到。

SSL/TLS 协议在几乎所有网络安全教科书中都有描述，但描述的形式多少都有些不同，这是因为该协议存在一些选项，原本目的是使协议具有灵活性，但也不可避免地造成理解和软件实现上的复杂性，而且为此可能导致一些理解和实现上的错误，因此请读者特别小心。关于 SSL 的软件实现目前有著名的开放源代码 OpenSSL，与该协议的 RFC 标准文档结合起来可以作为学习如何实现网络安全协议极好的材料。值得一提的是，OpenSSL 中的密码算法库已经发展成为当前很多安全协议所调用的标准库之一。这个协议的最新因特网标准文档资料可以参考 RFC#4346：

T.Dierks, E.Rescorla *The Transport Layer Security Protocol, Version 1.1*, Proposed IETF

[8] 该协议也属于 *Diffie-Hellman* 类型而且有很多变体，其早期的设计以众多的安全缺陷而"闻名遐迩"。

Standard, 2006.

有趣的是，应用如此广泛的网络安全协议却一直没有得到过完备而严格的分析与证明，直到作者写作本教程时才出现了以下两篇研究论文精确证明该协议的安全性质：

S.Gajek, M.Manulis, O.Pereira *Universally Composable Security Analysis of TLS – Secure Sessions with Handshake and Record Layer Protocols*, eArchive#2008/251, eprint.iacr.org/2008/251

P. Morrissey, N.P.Smart, B. Warinschi *A Modular Security Analysis of the TLS Handshake Protocol*, eArchive#2008/236, eprint.iacr.org/2008/236.

许多读者或许会感觉以上两篇论文数学味道太重，难以准确理解，但这是安全分析所必须的。对偏重于应用的读者，下面这篇直观的分析论文值得一读，论文作者之一是著名的 Schnerier 博士（参见第 8 章注 9），作于 SSL3.0 发表之初：

B.Schnerier, D.Wagner *Analysis of the SSL 3.0 Protocol*, Proc. USENIX Workshop on Electronic Commerce, 1996.

关于因特网上如何实施 VPN 有不少优秀著作，但总的来讲，彻底理解 VPN 仍然需要仔细阅读 *IPSec* 协议族的因特网标准 RFC#2401-2412，特别是其中的 RFC#2401 非常透彻地解释了 *IPSec* 实现因特网安全的思想和途径。*IPSec* 协议族本来是在 90 年代中期开始发展的下一代因特网(以 *IP*v6 为核心协议)的安全机制，但它对 *IP*v4 同样适用，而且事实上今天的 VPN 大多数都是 *IPSec* 与 *IP*v4 的结合物。

习 题

12-1 假如图 12-1 中的协议缺少两个密文分量 $E(g^{xy}, 0||A||B)$和 $E(g^{xy}, 1||B||A)$，请对这样一个协议给出完整的重放攻击过程，使攻击者成功地对 A 冒充 B，或者对 B 冒充 A。

12-2 图 12-1 中协议的两个密文分量 $E(g^{xy}, 0||A||B)$和 $E(g^{xy}, 1||B||A)$能用同一个明文的密文代替吗(例如都用 $E(g^{xy}, 0||A||B)$)？

12-3 图 12-2 中第二条消息的分量 $Sign(sk_B,g^x||g^y)$如果被替换为 $Sign(sk_B,g^y)$可以吗？为什么？第三条消息的分量 $Sign(sk_A,g^x||g^y)$如果被替换为 $Sign(sk_A,g^x)$可以吗？为什么？

12-4 对图 12-3 中的协议，分析如果没有消息认证码 $MAC(K_m, CA(A||vk_A))$和 $MAC(K_m, CA(B||vk_B))$将会存在什么后果。

提示：参考 12.3.1 小节的分析。

12-5 精确描述图 12-4 中的协议进程 A 和 B 的动作。

12-6 仿照图 12-4 中的协议，将图 12-3 中第一类匿名的 *SIGMA* 协议改造成具有第二类匿名性质的协议。

12-7 精确描述图 12-5 中的协议进程 A 和 B 的动作。

12-8 精确描述图 12-6 中的协议进程 A 和 B 的动作。

12-9 从网络协议的 OSI 分层模型的观点看，在数据链路层解决保密通讯和数据完整性问题比在网络层解决问题(例如象 *IPSec* 那样)更彻底，但实际上这种方案并不实际，特别对广域网而言矛盾更加突出，请问为什么？