第十一章 密钥交换协议(I): 基于口令的协议

从现在起我们用三章篇幅详细阐述密钥交换协议，更准确地说是带身份认证的密钥交换协议(*authenticated key-exchange*)。我们按照这类协议的应用形式分三种情形讨论，这一章着重阐述基于口令实现身份认证的密钥交换协议这一特殊情形；下一章阐述更一般、但也更复杂的带身份认证的密钥交换协议，但这两章仅限于 2-方会话情形；第 13 章阐述多方会话的情形，这是在新一代网络中正在开始广泛应用的密钥交换协议类型，也是最复杂的类型。

人人都使用过口令和基于口令的计算机协议，无论是在自动取款机还是在因特网上。然而除了少数专业工作者之外却很少有人知道，大多数基于口令的网络协议其实并不安全，正确设计基于口令的网络安全协议是一件颇不容易的事情；但另一方面，由于基于口令进行身份认证的实用价值，值得对基于口令的网络安全协议有一个深入的了解。基于口令的密钥交换协议的目标包括两方面：对协议双方的身份进行认证，然后在两者之间生成一个随机密钥用以对后续的会话进行加密(后续加密采用对称加密方案)。11.1 节以一个常见的设计错误为例详细解释字典攻击的概念和基于口令的网络协议的安全性概念；11.2 节讨论几类设计正确的协议实例。

11.1 从一个失败的例子谈起：字典攻击和口令安全性

口令的目的是为了在事先共享了秘密口令的双方之间进行身份认证，其中至少有一方是人类用户。口令看起来似乎是随机的，至少我们每个人相信我们生成的口令很难被他人正确地猜测出来，然而问题的关键恰在这里：口令真的那么难以猜测吗？换句话说，口令真那么随机吗？

很不幸，答案是“否”。口令不仅并非真正的随机数，而且也永远不可能做到随机，否则我们人类无法记忆从而口令也就失去了实用价值。不难设想，当你生成一长串随机数，例如 100 个字符的随机长口令，你为了记忆将不得不把它写下来并且记在什么地方，这样的口令还安全吗？因此为了记忆方便，我们(人类)不得不尽量生成短口令，例如 6-10 个字符的长度以便记在头脑里，即使确实需要生成长口令，例如 100 个字符的长度，也一定是有规律、便于记忆的长字符串。可以毫不夸张地说，对人类用户而言，长口令比短口令更缺乏随机性。一个有趣的例子参见习题 11-1。

用精确的数学概念表达以上观点就是：人类用户的口令并非在其样本空间上均匀分布。不仅不是均匀分布，而且多数情况是高度不均匀的分布。正因为如此，攻击者永远可能猜测到正确的口令！

举一个十分典型的例子。如果攻击者看到一个以秘密口令进行登录的协议消息，其中出现用户的名字或身份标识 zhouxc，攻击者完全可以这样来推测该用户的口令字，今后把所有这些候选口令字的集合称做字典：“zhouxc”显然是该用户的姓名，考虑到很多人都喜欢用自己的姓名加上自己的生日或某些自己喜欢的数字联结成口令，这些数字如“8”,“6”以及与自己生日有关的年月日，攻击者就可以据此把这些字符串合理组合以猜测候选口令。细心的读者也许会问这样做未必真能猜中口令，因为攻击者怎么知道该用户的生日？攻击者甚至不知道这位用户的完整的名字。但这并非什么实质性的困难，从当前倒退 100 年之内一共 36500 天之内必定包含这名用户的生日；遍历所有以“s”和“t”为声母的汉字[1]的拼音必定可以包含该用户的真实名字。如果读者认为这仍然不够完整，可以包含更多的年月日(例如人类纪元以来的一切年月日，毕竟用户可能用他/她的曾祖父或某位古代英雄的生日或事件的时间作为他/她的偏爱)，凡此等等，再加以各种组合，例如“zhouxingchi”、“xczhou”、“xingchiz”、“xingczhoust”、“zhouxc6612”、“zhouxc1266”、“1266zhouxc”、“120466xczhou”、“120466zhouxingchi”、“120466zxc”……、等等，等等，再进一步组合进一些特殊的字符串，例如当前流行的词汇、某些当红明星的名字、动物名字、魔幻数字、卖座大片的名字、常用符号如“-”、“+”等等，并且注意候选口令不必很长，如此一来，现在所生成的字典能够覆盖用户的真实口令的机率有多高？毫无疑问能够充分接近于 1！

应用计算机生成以上字典显然并非难事，再考虑到绝大多数口令长度都很短，例如 10 个字符左右，不难估算出这样一个字典甚至可以完整存储于当今普通桌面电脑的内存。于是攻击者便具有了实施口令猜测的基本素材，即口令字典。

完整的口令猜测需要两个步骤：

第一步是象上面这样生成字典，注意这一步与拟攻击的具体协议无关，针对任何协议以上做法都是一样的；反过来说，任何协议也无法阻止攻击者完成这一步。然而仅仅这一步并不足以对用户的口令安全真正构成威胁。从攻击者的目的出发，如果字典中已经合理包含了一百万条候选口令，攻击者还需要完成第二步，就是验证究竟哪一条候选口令是用户的真实的口令，或相对其它候选口令而言更接近于真实的口令。这里需要强调的是，与第一步不同，

[1] 这里的讨论针对中国用户,但很容易看出针对任何民族的用户这里的思想都能适用, 而且事实证明(很不幸)确实如此。

在这一步攻击者需要借助于协议所提供的信息才能进行判定，而许多协议的失败之处就恰恰在于他们在这一点上帮助了攻击者。

看图 11-1 的协议实例。协议双方 A 和 B 事先共享秘密口令 *pw*，这一协议的目的是使 B 基于口令 *pw* 验证 A 的身份。协议应用了对称加密方案，其加密算法记做 E、解密算法记做 D，读者不妨假设 E 具有完美的保密性质(下面会看到这个例子重演了第 9 章中解释的许多设计错误，即理想的基本密码方案丝毫不足以保证一个结构错误的协议的安全性)。

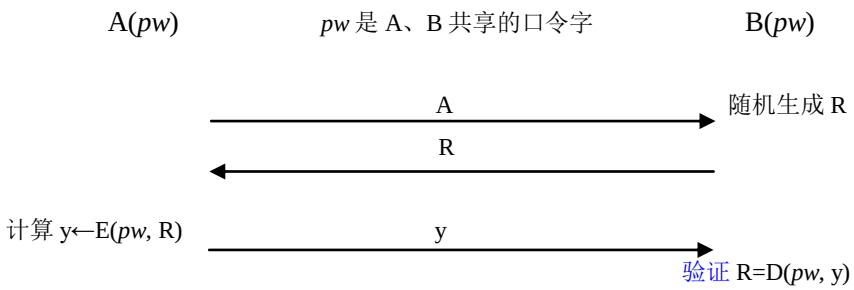


图 11-1 一个不能抵抗字典攻击的基于口令的身份认证协议

A(一个用户)首先向 B(例如一个服务器)发送自己的名字“A”；B 生成一个随机数 R 并将 R 发送回对方；A 用自己与 B 共享的秘密口令 *pw* 为对称加密方案的密钥对 R 加密后将密文发送给 B(精确地说，是 A 将口令 *pw* 散列为符合长度要求的对称密钥 *H*(*pw*)后，再以该散列值[2]为对称密钥加密 R，但为行文简洁今后一概用“用口令 *pw* 为密钥”这种表述)；B 根据第一步消息中的名字 A 取出对应的口令 *pw* 并以此解密第三步中接收的消息 y，若解密出的明文恰是第二步中自己随机生成的 R 则判定对方确实是 A，否则拒绝对方的身份。

既然 R 由 B 随机生成，除非对方确实持有口令 *pw*，否则不可能正确加密 R 从而使 B 以 *pw* 再行解密时还能得到原来的 R，因此这一协议的设计看上去合乎逻辑。这实际上代表了很广泛一类所谓 challenge-response 类型的身份认证协议的特点，然而就这里的例子来讲，却呈现出一个能够被口令猜测者所利用的漏洞。

一个攻击者观测 A 和 B 之间的以上协议会话，记录下所有消息：A、R 和 y，然后根据 A 的名字生成口令字典 Ω。接下来，攻击者逐一测试 Ω 中的候选口令 pw_i，具体方法是检验 $E(pw_i,R)=y$ 是否成立，如果不成立则 pw_i 当然不可能是真实的口令，而如果成立则 pw_i 很可能是真实的口令。在实践中，攻击者为了提高猜测概率会观测 A、B 之间足够多的协议会话，如果发现某个候选口令 pw_i 对所有这些会话的消息都满足以上等式，则观测的协议会话越

[2] H 是公开的散列算法，因此若 *pw* 已知则 *H*(*pw*)也已知，所以这一散列步骤不足以阻止口令猜测。

多，pw_i 是真实口令的概率就越高。但实际上，为了发现真实的口令只需要很少几次观测就足够了(请读者思考这是为什么？提示：R 是随机数)。

对以上攻击需要注意几点：第一，目前几乎所有常用的对称加密算法虽然依赖于随机初始向量(简称 IV，一个具有固定长度的真正的随机串)来随机化密文，但随机初始向量本身是不被保密的，因此在这个意义上讲，对称加密算法本质上是确定性算法。正因为如此，$E(pw_i,R)=y$(精确地表达就是 $E(pw_i,IV||R)=y$)对这里的攻击者除了 pw_i 以外没有任何未知的东西，所以这一检验是可行的。第二，以上攻击是完全被动式的，攻击者仅仅记录消息而没有以任何方式参与协议，这意味着攻击者永远不会暴露自己。这也正是这类攻击的危险所在。最后，这里所解释的攻击就是针对基于口令的网络协议的所谓字典攻击。实际上 90 年代初期美国学者就成功地开发出了工具软件，在当时的计算能力下实际花费几个小时便破译了 UNIX 服务器上的全部口令。

从上面的讨论所能得出的正面结论是：一个基于口令的网络安全协议必须阻止攻击者对其所猜测的候选口令的验证，也就是堵塞字典攻击的第二步。堵塞到什么程度才能保证安全？为了澄清这一问题，我们注意另一条极端的口令猜测途径：攻击者直接冒充 A 的身份与 B 会话，并且在会话中使用字典中的口令。这是一种主动身份欺诈，不难想象如果攻击者与 B 会话足够多次，例如一万次，最终攻击者能够有机会欺骗 B，但在成功欺骗的那一次会话之前，B 会看到“A”一直持续不断地使用错误的口令。因此，要堵塞这一主动攻击途径是非常容易的，办法正是目前常用的那种：如果“A”连续 N 次不能使用正确的口令，B 将永久拒绝与之会话，换句话说，A 的身份将对 B 失效。回到基于口令的网络协议的安全性概念，直观地表达就是：

如果对一个基于口令的网络协议的最有效的口令猜测途径是主动的在线口令猜测，则称该协议是口令安全的。

显然图 11-1 的协议不是口令安全的，对这个协议而言，更有效的口令猜测途径是字典攻击而毋须在线主动猜测。

11.2 安全协议的例子

这一节阐述几个正确设计的协议实例。这些基于口令的网络协议的目的都不是单纯的身份认证，而是属于带身份认证的密钥交换协议，其中口令的目的是对协议双方进行身份认证。带身份认证的密钥交换协议是应用非常广泛(也许可以说是最广泛)的一类网络安全协议，下

面两章章我们还要继续学习这类协议。

基于口令的带身份认证的密钥交换协议必须具备两个安全性质：第一个是上一节所刻画的口令安全性质，第二个是密钥保密性质，直观地说就是：除协议双方之外，任何第三方(包括被动和主动攻击者)都不可能有效推断出协议所协商出的会话密钥。

11.2.1 *Bellovin-Merrit* 协议

Bellovin-Merrit 协议发表于 1993 年，是第一个口令安全的密钥交换协议[3]。这个协议用到两个对称加密方案和一个公钥加密方案作为基本元素，两个对称加密方案的加密算法分别记做 E_0 和 E_1，公钥加密算法记做 E，协议结构如图 11-2。

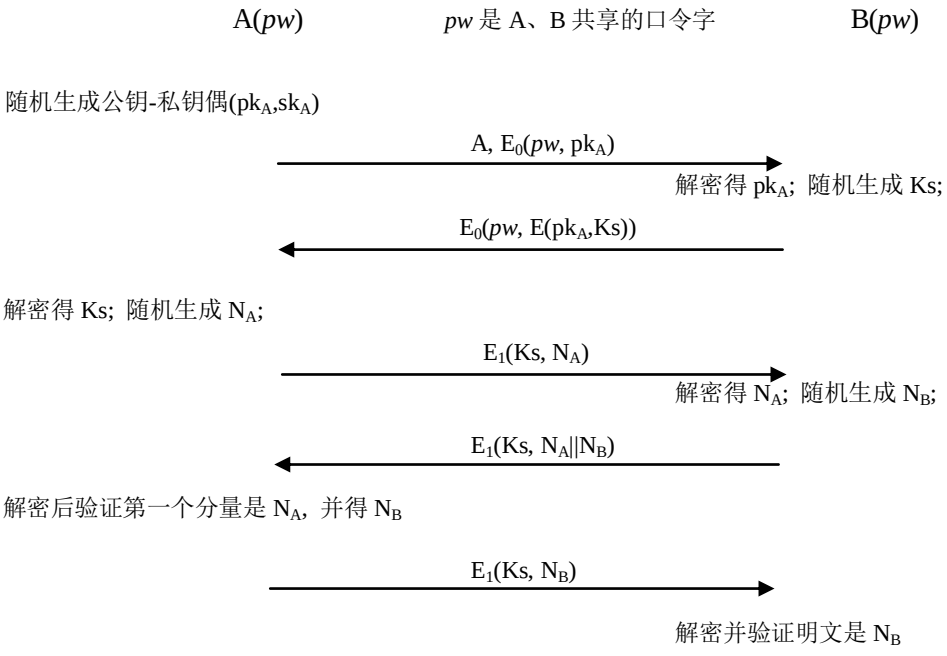


图 11-2 *Bellovin-Merrit* 协议

A、B 事先共享秘密口令 *pw*。A 每次开始与 B 发生协议会话时，随机生成一对新的、用于公钥加密方案 E 的公钥和私钥(pk_A,sk_A)，然后向 B 发送自己的身份标识和用 *pw* 加密的 pk_A 的密文；B 接收到该消息后，以 *pw* 解密密文得出 pk_A、随机生成会话密钥 K_s 并把经过两重加密的密文 E_0(*pw*, E(pk_A,Ks))发送回对方，这里的 K_s 将在该协议会话结束后用做紧接着

[3] 这一工作激发了关于口令安全性的一系列研究，正是他们建立了关于口令安全性的正确的直观概念，在他们的工作出现之后才建立起这一概念的精确形式。有趣的是该协议本身的口令安全性至今尚未得到证明，虽然普遍相信这一协议确实是口令安全的。Bellare 和 Rogaway 在 2000 年证明了一个形式与 Bellovin-Merrit 协议非常相似的协议的安全性，这就是著名的 Bellare-Rogaway 协议，见习题 11-5。

的一切数据通讯的对称加密密钥；A 接收到第 2 条消息后先用 *pw* 然后用 sk_A 解密得到 Ks、生成随机数 N_A、用 Ks 加密 N_A 并将密文 $E_1(Ks, N_A)$发送到 B；B 接到这一消息后以 Ks 解密得出 N_A、生成随机数 N_B、以 Ks 加密 $N_A||N_B$ 并将密文 $E_1(Ks, N_A||N_B)$发送回对方；A 在接收到第 4 条消息后以用 Ks 解密得出含两个分量的明文 $N=N_1||N_2$，验证其中的分量 N_1 是否等于 N_A(为简洁起见，今后这类判断均以"$=^?$"表示，例如图 11-3 中的表达式 $Vf(K_{a2}, 1||s||X^*||Y^*,t_1)=^?1$ 表示"计算 $Vf(K_{a2}, 1||s||X^*||Y^*,t_1)$并验证其值是否等于 1"，图 11-5 中的 $H(R)=^?D(pw, y_2)$表示"计算 H(R)和 $D(pw, y_2)$并验证两者的值相等")，如果成立则判定对方确实是 B，接下来用 Ks 加密另一个分量 N_2 并将密文 $E_1(Ks, N_2)$发送到 B；最后，B 接到第 5 条消息后用 Ks 解密得出 N_2 并验证 $N_2=^?N_B$，如果成立则判定对方确实是 A，协议会话结束。

为什么上一节描述的字典攻击对 *Belllovin-Merrit* 协议不再有效？回顾字典攻击的第二步，在那里攻击者成功的关键在于能够找到这样一种关系式 $F(pw, X_1,X_2,\ldots)=1$，其中 $X_1,X_2\ldots$ 或者是可以从公开的参数推断出来的数据、或者是可以从协议消息直接观测到的数据、或从这两者进一步有效计算出来的数据，而 *pw* 是这组关系中唯一未知的对象。攻击者正是利用这类关系逐一测试字典中的候选口令得以发现真实的口令。然而，对 *Belllovin-Merrit* 协议并未给攻击者提供这类关系，这就是该协议能够抵抗抗字典攻击的关键。当然，以上描述仅仅是一种直观解释而不应该看做严格的论证。

11.2.2 *Abdalla-Pointcheval* 协议

Belllovin-Merrit 协议需要使用对称和公钥加密方案，特别是公钥加密计算在软件实现时代价不菲。这一节描述的 *Abdalla-Pointcheval* 协议发表于 2005 年，是一个非常高效的基于口令的带身份认证的密钥交换协议。这一协议与 *Belllovin-Merrit* 协议还有一个重要差别：在 *Belllovin-Merrit* 协议中会话密钥 Ks 完全由协议的一方(B)生成，而 *Abdalla-Pointcheval* 协议中会话密钥 Ks 由协议双方共同生成，单独任何一方都没有能力完全决定 Ks。所谓"密钥交换"指得就是这一特点(相比之下 *Belllovin-Merrit* 协议严格说来应该称做"密钥发布"协议)。

Abdalla-Pointcheval 协议需要这样几个公开的参数和基本密码方案：一个判定性 *Diffie-Hellman* 问题难解(回顾 8.3.2 节)的素数阶循环群 G、G 的生成子 g、G 上的两个任意给定的元素 M_1 和 M_2、两个抗冲突的散列算法 H 和 *f* [4]和一个抗伪造的消息认证方案 Σ=(KG, Tag, Vf)(回顾 8.2.1 节)。协议会话的过程如图 11-3。

协议中的表达式 M_1^{pw} 意味着把口令 *pw* 当做一个整数进行指数运算，这应该理解为先对

[4] 精确地说, H 是所谓随机-oracle 而 *f* 是所谓拟随机函数, 但这里我们不深入到这些微妙的理论概念。

pw 进行某种(公开方式的)编码，然后将编码结果用于指数运算。只要编码是一一对应的，编码方式本身不影响协议安全性。协议消息中的 s 是会话标识号，用以区分并发的协议会话。

请读者象上一小节那样自行详细描述该协议的每一步骤，并且验证 $K_1=K_2=g^{xy}$ 和 $sk_1=sk_2=\mathrm{H}(U_1||U_2||X^*||Y^*||pw||g^{xy})$。$sk_1$、$sk_2$ 就是这一协议所生成的会话密钥。

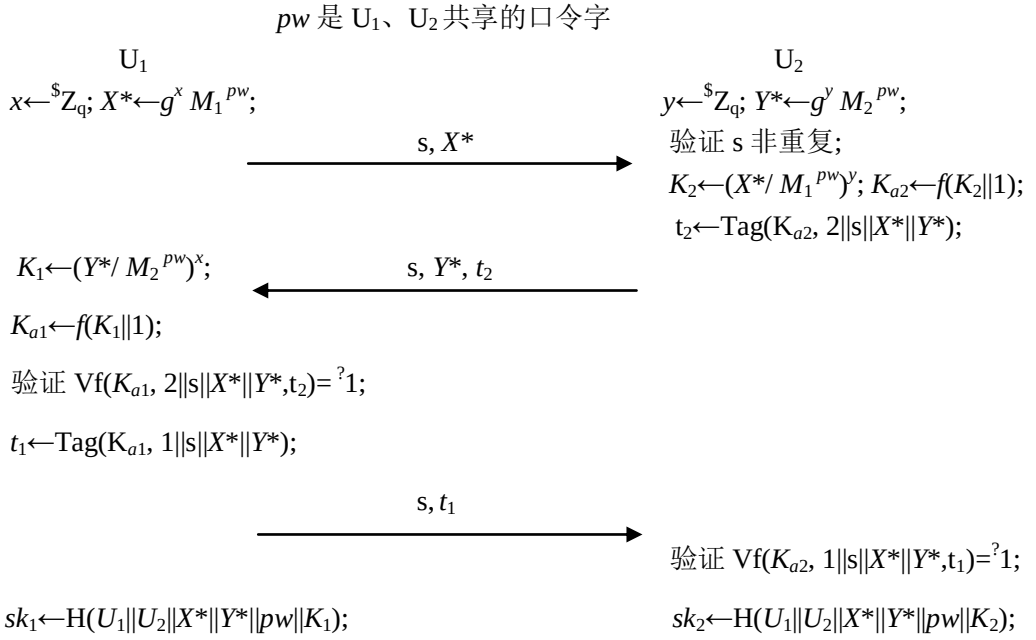


图 11-3 *Abdalla-Pointcheval* 协议

已经证明，在相应的难解性假设之下 *Abdalla-Pointcheval* 协议口令安全而且密钥保密。

11.3 小结与进一步学习的指南

口令是用以对人类用户实施身份认证的传统手段，但口令并非真正的随机数而且大多数情况下都有充分的规律可循，从而能被有效地猜测出来，因此基于口令的身份认证机制必须避免给攻击者提供任何有效的用以验证其口令猜测是否正确的依据。如果在针对一个基于口令的协议实施口令猜测的所有途径之中，相对最有效的途径就是在线猜测，这样的协议就是口令安全的。

虽然字典攻击技术可以参考很多关于系统安全的教科书，例如一些讨论 UNIX 安全的教科书，但关于口令安全的网络协议的详细讨论则很少，除了参考第 8 章列举的 W. Mao 的教科书《现代密码学理论与实践》第 11 章的部分内容之外，更深入的内容只能研读研究论文。

在研究领域，理论上有严格证明、同时又具有足够良好的计算效率的口令安全的协议至今并不是很多。本章描述的例子和习题 11-5 都属于所谓随机-oracle 范型的协议，取自以下论文：

S.Bellovin, M.Merrit *Augmented Encrypted Key Encryption*: *Password-based Protocol Secure against Dictionary Attacks*, Proc. IEEE Symp. On Security and Privacy, 72-84, 1992. 这篇论文有关于口令安全性质的精彩讨论。

M.Belleare, P.Rogaway *Authenticated Key Exchange Secure Against Dictionary Attacks*, Proc. Eurocrypt, Lecture Notes in Computer Science Vol.1807, 139-155, 2000. 这篇论文建立了口令安全性质的精确的数学模型并证明了 Belleare-Rogaway 协议的安全性。

M.Abdalla, D.Pointcheval *Simple Password-based Encrypted Key Exchange Protocols*, Proc. Topics in Cryptology, CT-RSA, Lecture Notes in Computer Science Vol. 3376, 191-208, 2005.

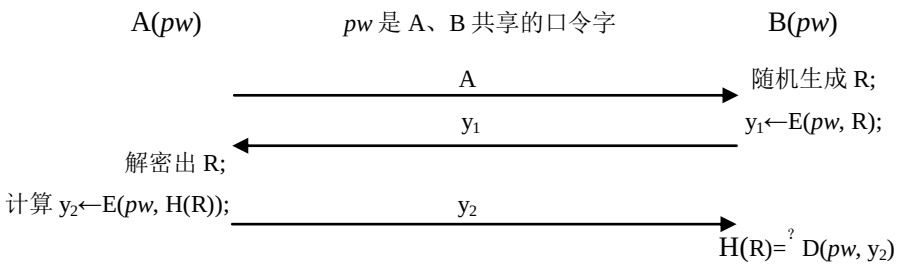
最近还建立了非随机-oracle 范型的口令安全协议，例如 Canetti-Halevi-Katz-Lindell 协议，这类协议更安全，能适合于任何应用环境，但也更为复杂。

习 题

11-1 和纯粹的随机数相反，口令越长其随机性反而越小。不妨一试：请分别写下 6 个字符、12 个字符、20 个字符、50 个字符、100 个字符和 200 个字符的可以保证在很长时间内准确记忆(!)的口令，然后从两方面比较这些口令：这些口令中重复出现的字符数量，这些口令内在的结构(例如从某个位置开始倒转字符的排列)。是长口令的规律明显还是短口令的规律明显？

11-2 根据第 11.1 节讨论字典攻击的思路写一个程序，在接受一组字串后尽可能大地生成一个具有指定长度的口令的字典。实际测量你的程序生成的字典大小和运行时间。

11-3 回顾 11.1 节的讨论，分析对图 11-1 中协议的以下“修改”方案，其中 H 是一个已知的函数，这些“修改”方案的共同点是都不再直接暴露 R。



(1)设 H(R)由 R 完全确定，例如约定 H(R)=R 的 MD5 或 SHA 散列值、R 的各位反转、R 的平方、R||A 等等。

(2)约定 H 有一定随机性，例如 $H(R)=R_1\|R$，其中 R_1 是 A 独立生成的另一个随机数。

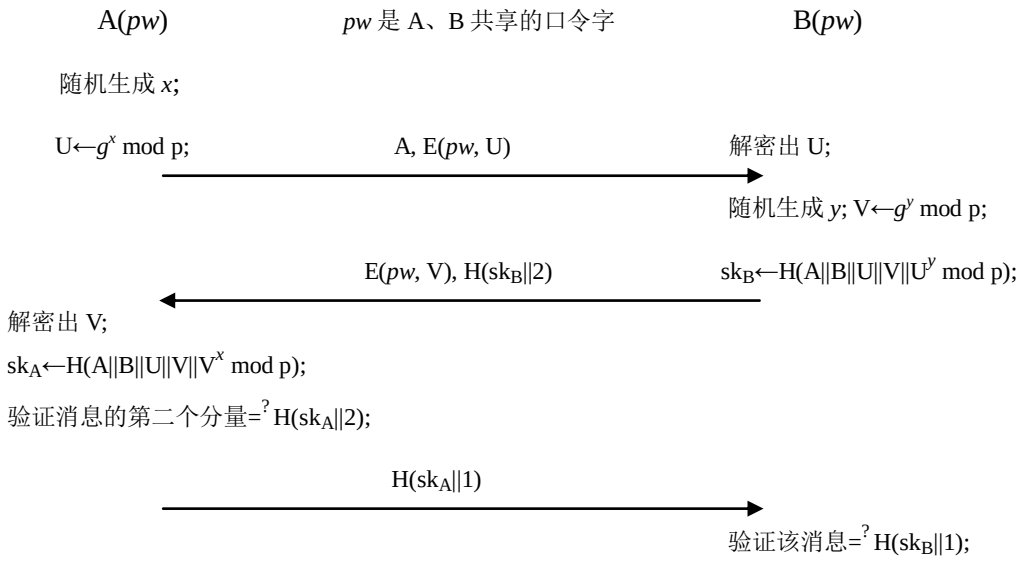
(3)约定 A 和 B 所用的加密算法各不相同。

以上所有这类方案仍然不能够抵抗字典攻击，为什么？

提示：设 D 表示 E 的解密算法，对(1)，消息 y_1 和 y_2 之间存在关系 $E(pw,H(D(pw,y_1)))=y_2$。

11-4 11.2.1 节中的 Bellovin-Merrit 协议(图 11-2)中的公钥-私钥对(pk_A,sk_A)如果不是每次随机生成，而是生成一次而后被重复使用，该协议还安全吗？为什么？如果其中的随机数 N_A、N_B 也被在多个协议会话之中重复使用，该协议还安全吗？为什么？

11-5 分析下面基于口令的密钥交换协议(Bellare-Rogaway 协议,2000)，其中 g、p、H 是公开约定的参数，p 是素数、1<g<p、H 是单向散列函数(例如 MD5 或 SHA/SHA1)；E 是对称加密方案的加密算法。A、B 通过该协议认证对方的身份并计算出共同的会话密钥 $K_S=H(A\|B\|U\|V\|g^{xy} \bmod p)$。



(1) 象第 11.2.1 节那样详细描述 A 和 B 的动作；

(2) 协议终止时，A 和 B 分别计算出 $H(sk_A||0)$和 $H(sk_B||0)$并以此为其生成的会话密钥，验证 $H(sk_A||0)=H(sk_B||0)$；

(3) 定性地解释为什么字典攻击对以上协议无效。

(4) 如果 H 不是单向函数，而是一个容易计算逆映像的函数，以上协议不能抵抗字典攻击，

为什么？

11-6 分析下面基于口令的密钥交换协议，它与上一题中的 Bellare-Rogaway 协议很相似，但把原来放在密文外面被单独散列的 A||B||U||V||g^{xy} mod p 放到密文内部。

(1) 写出 K_m 的算术表达式；B 如何计算出同一个 K_m？

(2) 这个协议并非口令安全，为什么？