如何共享秘密

Adi Shamir

麻省理工学院

本文展示了如何将数据 D 分割为 n 份，使得 D 可以轻易地从任意 k 份中恢复，但即使完全知道 $k-1$ 份，也绝对无法获得关于 D 的任何信息。这项技术能够为密码系统构建稳健的密钥管理方案，即使不幸毁坏了一半份额，并且安全漏洞暴露了剩余份额中除一份之外的所有份额，系统仍能安全可靠地运行。

关键词与短语：密码学，密钥管理，插值
CR 分类：5.39, 5.6

1. 引言

在 [4] 中，Liu 考虑了以下问题：

十一位科学家正在进行一个秘密项目。他们希望将文件锁在柜子里，使得只有当六位或更多科学家在场时才能打开柜子。需要的最少锁的数量是多少？每位科学家必须携带的最少钥匙数量是多少？

不难证明，最小的解决方案需要 462 把锁和每位科学家 252 把钥匙。这些数字显然不切实际，并且随着科学家人数的增加，它们会呈指数级恶化。

在本文中，我们将问题推广为：秘密是某个数据 D（例如，保险箱密码），并且也允许非机械式的解决方案（即操作此数据）。我们的目标是将 D 分割成 n 份 $\mathbf{D}_1, \ldots, \mathbf{D}_n$，使得：

(1) 知道任意 k 份或更多 D_i 可以轻易计算出 D；
(2) 知道任意 $k-1$ 份或更少的 D_i 使得 D 完全无法确定（即其所有可能的值都是等可能的）。

这样的方案称为 (k, n) 阈值方案。

高效的阈值方案对密码密钥的管理非常有帮助。为了保护数据，我们可以加密它，但为了保护加密密钥，我们需要不同的方法（进一步的加密只会改变问题而不是解决问题）。最安全的密钥管理方案是将密钥保存在一个守卫森严的地方（计算机、人脑或保险箱）。这种方案非常不可靠，因为一次不幸事件（计算机故障、猝死或破坏）就可能导致信息无法访问。一个显而易见的解决方案是在不同位置存储密钥的多个副本，但这增加了安全漏洞（计算机入侵、背叛或人为错误）的风险。通过使用 $n = 2k-1$ 的 (k, n) 阈值方案，我们得到一个非常稳健的密钥管理方案：即使 n 份中的 $[n/2] = k-1$ 份被销毁，我们也能恢复原始密钥，但即使安全漏洞暴露了剩余 k 份中的 $[n/2] = k-1$ 份，对手也无法重构密钥。

在其他应用中，权衡不在于保密性和可靠性之间，而在于安全性和使用便利性之间。例如，考虑一家对所有支票进行数字签名的公司（见 RSA [5]）。如果每位高管都获得了公司秘密签名密钥的副本，系统使用方便但容易被滥用。如果签署每张支票都需要公司所有高管的合作，系统安全但不方便。标准解决方案要求每张支票至少有三个签名，并且可以用 $(3, n)$ 阈

值方案轻松实现。每位高管获得一张包含一份 D_i 的小磁卡，公司的签名生成设备接受其中任意三份来生成（然后销毁）实际签名密钥 D 的临时副本。该设备不包含任何秘密信息，因此无需防止检查。在这种方案中，不忠实的高管必须至少有两个同谋才能伪造公司签名。

阈值方案非常适用于一群相互猜疑、利益冲突的个体必须合作的场景。理想情况下，我们希望合作基于共同同意，但这种方式赋予每个成员的否决权可能会使团队的活动陷入瘫痪。通过恰当地选择 k 和 n 参数，我们可以赋予任何足够大的多数派采取行动的权力，同时赋予任何足够大的少数派阻止行动的权力。

2. 一个简单的 $(\mathbf{k}, \mathbf{n})$ 阈值方案

我们的方案基于多项式插值：给定二维平面上的 k 个点 $(x_1, y_1), \ldots, (x_k, y_k)$，其中 x_i 各不相同，存在唯一一个 $k-1$ 次多项式 $q(x)$ 使得对所有 i 满足 $q(x_i) = y_i$。不失一般性，我们可以假设数据 D 是一个数字（或可以转换成一个数字）。为了将其分割成 D_i 份，我们随机选择一个 $k-1$ 次多项式 $q(x) = a_0 + a_1 x + \ldots + a_{k-1}x^{k-1}$，其中 $a_0 = D$，然后计算：

$$D_1 = q(1), \ldots, D_i = q(i), \ldots, D_n = q(n).$$

给定这些 D_i 值中任意 k 个（连同它们的标识索引），我们可以通过插值找到 $q(x)$ 的系数，然后计算 $D = q(0)$。另一方面，只知道 $k-1$ 个这样的值，不足以计算出 D。

为了使这一论断更精确，我们使用模运算而不是实数运算。以素数 p 为模的整数集构成一个域，在该域中插值是可行的。给定一个整数值数据 D，我们选择一个大于 D 和 n 的素数 p。$q(x)$ 中的系数 $a_1, \ldots, a_{k-1}$ 是从 $[0, p)$ 上的整数中均匀随机选择的，值 $D_1, \ldots, D_n$ 则模 p 计算。

现在假设对手知道了这 n 份中的 $k-1$ 份。对于 $[0, p)$ 中的每个候选值 D'，他都能构造出唯一一个 $k-1$ 次多项式 $q'(x)$，使得 $q'(0) = D'$ 且 $q'(i) = D_i$ 对于给定的 $k-1$ 个自变量成立。根据构造，这 p 个可能的多项式是等可能的，因此对手绝对无法推断出 D 的真实值。

[1] 和 [3] 中讨论了多项式求值和插值的高效 $O(n \log^2 n)$ 算法，但即使是简单的二次算法对于实际的密钥管理方案来说也足够快。如果数字 D 很长，建议将其分成较短的比特块（单独处理）以避免高精度算术运算。块不能任意短，因为 p 的最小可用值是 $n+1$（必须在 $[0, p)$ 中至少有 $n+1$ 个不同的自变量来求 $q(x)$ 的值）。然而，这并不是一个严重的限制，因为对于多达 64,000 份 D_i 的应用，16 位模数（可由廉价的 16 位算术单元处理）就足够了。

与机械锁和钥匙的解决方案相比，这种 (k, n) 阈值方案的一些有用特性包括：

(1) 每份的大小不超过原始数据的大小。
(2) 当 k 固定时，可以动态添加或删除 D_i 份（例如，当高管加入或离开公司时），而不影响其他 D_i 份。（只有当离开的高管使其完全无法访问时，才会删除一份，即使对他自己也是如此。）
(3) 很容易更改 D_i 份而不改变原始数据 D——我们只需要一个新的具有相同常数项的多项式 $q(x)$。频繁进行此类更改可以极大地增强安全性，因为安全漏洞暴露的份额无法被累积，除非它们都是同一版本 $q(x)$ 多项式的值。
(4) 通过使用多项式值的元组作为 D_i 份，我们可以得到一个分层方案，其中确定 D 所需的份额数量取决于它们的重要性。例如，如果我们给公司总裁三个 $q(x)$ 值，每位副总裁两个 $q(x)$ 值，每位高管一个 $q(x)$ 值，那么 $(3, n)$ 阈值方案使得支票可以由任意三位高管签署，或者由任意两位高管（其中一位是副总裁）签署，或者由总裁单独签署。

G.R. Blakley [2] 最近开发了一种不同的（效率稍低的）阈值方案。

收到日期 1979年4月；修订日期 1979年9月

参考文献

1. Aho, A., Hopcroft, J., and Ullman, J. The Design and Analysis of Computer Algorithms. Addison-Wesley, Reading, Mass., 1974.
2. Blakley, G.R. Safeguarding cryptographic keys. Proc. AFIPS 1979 NCC, Vol. 48, Arlington, Va., June 1979, pp. 313-317.
3. Knuth, D. The Art of Computer Programming, Vol. 2: Seminumerical Algorithms. Addison-Wesley, Reading, Mass., 1969.
4. Liu, C.L. Introduction to Combinatorial Mathematics. McGraw-Hill, New York, 1968.
5. Rivest, R., Shamir, A., and Adleman, L. A method for obtaining digital signatures and public-key cryptosystems. Comm. ACM 21, 2 (Feb. 1978), 120-126.