一种利用公钥体制改进 Kerberos 协议的方法*

刘克龙，卿斯汉，蒙 杨

(中国科学院 信息安全技术工程研究中心，北京 100080)；
(中国科学院 软件研究所，北京 100080)
E-mail：lkl@ercist.iscas.ac.cn
http://www.ercist.ac.cn

摘要：针对 Kerberos 认证协议由对称算法所带来的局限性，在保持原有协议框架的基础上，采用基于 ElGamal 公钥算法的 Yaksha 安全系统对 Kerberos 协议进行一定程度的改进.

关键词：ElGamal 算法；Yaksha 体制；Kerberos 认证协议；认证中心
中图法分类号：TP393 **文献标识码**：A

Kerberos 协议是当今最重要的实用认证协议，人们对其进行了大量的研究和改进. 我们认为，虽然其优点明显，但也存在以下一些局限性：① 它使用对称算法 DES(data encryption standard)作为协议的基础，而这就带来了密钥交换、密钥存储以及密钥管理的困难；② 利用字典攻击对 Kerberos 系统进行攻击是简单有效的；③ Kerberos 协议最初是设计用来提供认证和密钥交换的，因此它不能用来进行数字签名，因而也不能提供非否认机制.

究其原因主要在于最初在实现 Kerberos 时，没有使用公钥加密机制. 为了弥补这一缺陷，很多人在 Kerberos 协议中集成公钥算法对 Kerberos 进行改进，但却因对 Kerberos 协议改动太大，进而成为一个新的认证协议. 例如 J. Tardo 和 K. Alagappan 的 SPX 系统[1]. 由于 Kerberos 协议已经广泛应用到各行各业，要使人们重新接收一个新协议，困难不小，因此，我们要求在将公钥体制集成到 Kerberos 协议中的同时，对 Kerberos 协议的改动要很小，要符合 John T. Kohl 在文献[2]中提出的要求.

1 Kerberos 认证协议描述

Kerberos[3]是一种基于可信任第三方的 TCP/IP 网络认证协议，其认证模型基于 Needham-schroeder 可信任第三方协议[4]. 它采用对称密码体制——DES 算法. 版本 5 采用 CBC(cipher block chaining mode)认证模式(版本 4 利用一种较弱的非标准的认证模式 PCBC——plain and cipher block chaining mode)，本文只讨论版本 5，版本 4 的局限性描述见文献[5]. Kerberos 认证服务描述如图 1 所示.

(1) 如果用户需要某种远程服务，它首先向 Kerberos 密钥分发中心 KDC(Key Distribution Center)申请一份同本地票据授权中心 TGS(Ticket-Granting Server)通信的票据 TGT

* 收稿日期：1999-06-16；修改日期：2000-03-08
基金项目：国家自然科学基金资助项目(60083007)；国家重点基础研究发展规划 973 资助项目(G1999035810)
作者简介：刘克龙(1971－)，男，安徽桐城人，博士生，讲师，主要研究领域为信息安全理论与技术；卿斯汉(1939－)，男，湖南隆日人，教授，博士生导师，主要研究领域为信息安全理论与技术；蒙杨(1972－)，男，甘肃天水人，博士，讲师，主要研究领域为信息安全理论与技术.

(ticket-granting ticket):用户向KDC发送一条包含用户ID号以及相应IGS的ID号的消息,请求同TGS通信的票据TGT,即

$$c\_to\_k: c,tgs.$$

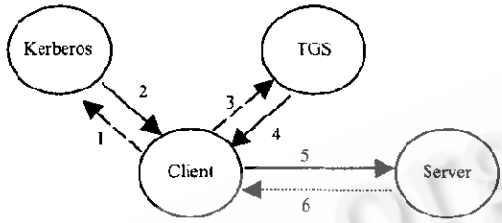

Fig. 1 Kerberos protocol authentication steps

图1 Kerberos协议认证步骤

(2) Kerberos密钥分发中心(KDC)确认此用户ID是否在数据库中,若在,则KDC产生一个客户端与TGS通信的会话密钥$K_{c,tgs}$(即签发票据许可票据TGT),以用户密钥对其加密,并产生用户相对于TGS的认证信息$T_{c,tgs}$(以TGS的密钥对其进行加密).所有这些信息一并返回给用户:

$$k\_to\_c: \{K_{c,tgs}\}K_c, \{T_{c,tgs}\}K_{tgs}.$$

(3) 用户得到消息后,随即向票据授权服务中心(TGS)发送与之会话的TGT以及服务申请($T_{c,tgs}$中含有远端Server ID号s、时间戳ts及$K_{c,tgs}$,可以被TGS用来验证$K_{c,tgs}$的正确性):

$$c\_to\_tgs: \{T_{c,tgs}\}K_{tgs}, \{ts,\ldots\}K_{c,tgs}.$$

(4) TGS验证票据并向用户签发服务票据:

$$tgs\_to\ c: \{K_{c,s}\}K_{c,tgs}, \{T_{c,s}\}K_s.$$

(5) 用户向远端Server提交服务票据和自己产生的认证码($A_{c,s}=\{c,t,\ldots\}K_{c,s}$),Server通过比较票据和认证码里的信息来验证可靠性,从而决定是否给予相应的服务:

$$c\ \ to\ \ s: \{c,ts,\ldots\}K_{c,s}\{T_{c,s}\}K_s.$$

(6) 如果需要交互认证,远端服务器将向客户端发送一条包含时间戳的消息,并以双方的会话密钥对之加密,客户端收到消息后经解密得到时间戳,若正确,则表明服务器知道自己的私钥,并将消息解密得到了相应的会话密钥,从而证明了自己的身份:

$$s\_to\_c: \{ts\ldots\}K_{c,s}.$$

2 Kerberos认证协议的局限性

Kerberos协议最初是为MIT(Massachusetts Institute of Technology)的Athena项目而设计的,将它推广到分布式系统中还存在一定的局限性[5],其中最为严重的有如下几类[6]:

• 由于Kerberos服务器保存所有用户或服务器的密钥,如果被攻破,其后果是灾难性的.

• Kerberos认证协议对口令攻击比较脆弱,从Kerberos协议的认证过程来看,AS(authenticate server,又称KDC)并不能验证用户的身份,而是基于发回用户的k_to_c消息是用K_c加密的,只有知道K_c的人才能够对之解密,而K_c和用户的passwd有关.由于很多用户不重视对强口令的选择[7],入侵者可以通过收集票据来对此解密,如果票据足够多,口令的恢复是可能的.

• Kerberos协议不能提供非否认机制(non-repuciation),也就是不提供数字签名.

3 利用基于 ElGamal 算法的 Yaksha 系统对 Kerberos 协议的修正

为了达到 John T. Kohl 在文献[2]中提出的要求,如文献[7]所述,在将公钥体制加入到 Kerberos协议的同时,应保证在原有协议基本不变的基础上只作微小改变:

- 不希望在协议中加入另外的步骤,应保证 Kerberos 协议中 6 个步骤的完整性.
- 不希望改变重要信息的结构,如票据的结构.
- 允许在消息中加入额外的结构,但是只限于在增强协议的安全性的目的上,并且不会对整个结构产生大的变化.

为此,我们利用基于 ElGamal 算法的 Yaksha 安全机制对 Kerberos 协议进行修正以达到上述目的.

3.1 基于 ElGamal 算法的 Yaksha 系统

Yaksha 体制[8]在 RSA 算法的基础上,将 RSA 算法中的私钥 d_a 拆分为两个密钥:d_{aa} 和 d_{ay},且 $d_{aa}\times d_{ay}=d_a \bmod n_a$,其中 d_{aa} 为用户 a 的私钥,d_{ay} 为用户 a 的 Yaksha 服务器私钥,且 d_{aa} 和 d_{ay} 不能相互推知.但是,由于该系统缺少公钥构架 PKI(public key infrastructure),使得公钥缺少可信任第三方的公证,并且 Yaksha 服务器是此系统中最大的安全隐患,一旦 Yaksha 服务器被攻破,整个系统将完全暴露在入侵者的眼下.为此,我们提出了基于 ElGamal 算法的 Yaksha 安全系统[9]:在原有 Yaksha 系统的基础上,引入可信任第三方——认证中心 CA,用来保存用户及服务器的公钥证书,并且选用 ElGamal 算法作为公钥体制,该系统可用作加密、数字签名、分布式认证、密钥交换等等.

3.2 对 Kerberos 协议的修正

为了达到既对 Kerberos 协议只作微小修改,又可集成公钥体制等要求,文献[7]提出了在 Kerberos 协议中集成 Yaksha 体制的方法,但正如上文所述,由于 Yaksha 系统缺少可信任的第三方公证,其安全性仍不完美,为此,我们在本系统中加入一认证中心 CA,并作如下修正:

- 加入认证中心 CA.
- 将证书作为额外的字符串加入到某些消息中.
- 以公钥体制代替单钥体制(对称算法),由于协议和算法是两回事,所以算法的改变不会造成协议架构的变化.

修正 Kerberos 协议的详细过程如下:

(1) 系统构造及用户注册

可信任认证中心 CA 保存所有用户和服务器的 ElGamal 公钥(y)、Yaksha 公钥(y_a)以及相应的 Yaksha 私钥(x_{ay})和所有用户的通用公钥(g,p).首先,YS 随机选择一个大素数 p,并且 $p-1$ 有大素数因子,选择一个本原根 $g\in Z_p^*$,公开(g,p),供本系统中所有用户使用,用户注册过程的基本步骤如下:

① 用户 a 选择一个随机数 x_{aa},使得 $x_{aa}\in z_{p-1}^*$.计算 $y_{aa}=g^{x}aa(\bmod p)$,x_{aa} 即为用户 a 的私钥.

② 将 a 的用户号 ID_a 及用 YS 的公钥 y_s 加密的 y_{aa} 传送至 YS,即

$$a \rightarrow \mathrm{YS}:\mathrm{ID}_a, E_{y_s}[\mathrm{ID}_a, T_a, y_{aa}],$$

其中 T_a 是用户 a 加盖的时戳,防止重放攻击.同时,为了防止已知明文攻击,T_a 应是时间的函数.$E_{y_s}[M]$ 表示用 y_s 对 M 进行加密.

③ 根据 Diffic-Hellman 密钥交换协议,YS 选择一个随机数 x_{ay},使得 $x_{ay}\in z_{p-1}^{*}$. 计算 $y_a=y_{ac}^{x_{ay}} \bmod p$,使得 $y_a\neq 1$,否则重新选择 x_{ay}. 其中,x_{ay}是用户 a 的 YS 私钥,由 YS 保存. y_a 是用户 a 的公钥;由 YS 用 CA 的公钥加密传送至 CA,由 CA 产生 a 的公钥证书:

$$\mathrm{Cert}_a=E_{S_{ca}}[\mathrm{ID}_a,(g,p),\mathrm{lifetime},y_a].$$

其中 lifetime 为 x_{aa},x_{ay}和 y_a 的有效期.

④ YS 用 a 的公钥 y_a 及 YS 的私钥 x_{ay}加密 T_a 传送至用户 a.

用户 a 收到消息后,解密得到 T_a,验证其正确性,若正确,则用户注册成功,与此同时,用户 a 就有了用户私钥 x_{aa},对应的用户 Yaksha 私钥 x_{ay}、用户公钥 y_a 以及这些密钥的有效期 lifetime.

(2) 对 Kerberos 协议的 6 步消息的修正

在对 Kerberos 协议进行修正之前,我们沿用 Kerberos 协议对符号的定义,除此之外,我们再定义如下符号:

$\{M\}\hat{}\ k$ 表示用密钥(k,g,p)对消息 M 按 ElGamal 体制进行加密;

$\{M\}k$ 表示用密钥 k 对消息 M 按单钥体制进行加密;

x_{cc}表示 c 的私钥,x_{cy}表示 c 的 Yaksha 私钥,y_c 表示 c 的 Yaksha 公钥.

在构造完系统之后,我们利用与文献[7]类似的方法对 Kerberos 协议进行相应的修正,以便能与该文有所比较,其具体修正过程如下:

第 1 步:在 Kerberos 中,初始 c_to_k 消息为

$$\mathrm{Kerberos}:c\_\mathrm{to}\_k:c,tgs.$$

相应的 ElGamal Yaksha 消息为

$$\mathrm{ElGamal\ Yaksha}:c\_\mathrm{to}\ \ k:c,tgs\{\{c,y_{c,temp},\mathrm{lifetime}\}\hat{}\ x_{cc},n\}\hat{}\ x_{ca}.$$

其中 $y_{c,temp}$是用户 c 临时产生的 ElGamal 体制(请注意,ElGamal 体制与 ElGamal Yaksha 体制的区别)的公-私钥对$(y_{c,temp},x_{c,temp})$的公钥部分,私钥部分由用户自己保留. Lifetime 为此公-私钥对的有效期. 所有这些加上用户 ID 以用户的私钥 x_{cc}对其加密,其后加上一段随机串 n,然后再以 x_{cc}对其进行加密,这样做的目的是为了防止攻击者通过对$\{c,y_{c,temp},\mathrm{lifetime}\}x_{cc}$进行字典攻击得到$\{c,y_{c,temp},\mathrm{lifetime}\}$,即$\{c,y_{c,temp},\mathrm{lifetime}\}\hat{}\ \mathrm{guess}=\{c,y_{c,temp},\mathrm{lifetime}\}\hat{}\ x_{cc}$.

Yaksha(YS)服务器收到消息后,利用 x_{cy}以及相应的 ElGamal 公钥(g,p,y_c)对其进行解密,获得$\{c,y_{c,temp},\mathrm{lifetime}\}$,同时也证明了客户端的身份(因为只有客户端才有自己的私钥).

第 2 步:服务器在验证通过客户端的身份后,向客户端发送:

$$\mathrm{Kerberos}:k\ \ \mathrm{to}\_c:\{K_{c,tgs}\}K_c,\{T_{c,tgs}\}K_{tgs},$$

$$\mathrm{ElGamal\ Yaksha}:k\_\mathrm{to}\_c:\{K_{c,tgs}\}\hat{}\ y_{c,temp},\{T_{c,tgs}\}\hat{}\ x_{tgsy}$$

$$\{\{c,y_{c,temp},\mathrm{lifetime}\}\hat{}\ x_{cc}\}\hat{}\ x_{cy}.$$

由此可见,在 ElGamal Yaksha 体制中,其 k_to_c 的前两部分与 Kerberos 协议的 k_to_c 相似,只不过第 1 部分 Kerberos 是用客户端与服务器共有的私钥 k_c 加密的,ElGamal Yaksha 是由用户临时产生的 ElGamal 体制的公钥加密的,这是为了保证在 Kerberos 协议中集成公钥体制. 后者的第 3 部分是另外加上的,是为了验证服务器(相当于 Kerberos 协议中的 KDC)的身份,即通过服务器用客户的 Yaksha 私钥对客户发来的消息$\{c,y_{c,temp},\mathrm{lifetime}\}\hat{}\ x_{cc}$进行签名以验证服务器的身份.

第 3 步:通过第 2 步以后,客户端就有了同 TGS 会话的密钥 $k_{c,tgs}$,接着进行第 3 步:

Kerberos: $c\_to\_tgs: \{T_{c,tgs}\}K_{tgs}, \{ts...\}K_{c,tgs}$,

ElGamal Yaksha: $c\_to\ tgs: \{\{T_{c,tgs}\} \hat{} x_{tgsy}\} \hat{} x_{c,temp}\{ts...\}K_{c,tgs}$

$\{\{c, y_{c,temp}, \text{lifetime}\} \hat{} x_{ca}\} \hat{} x_{cy}$.

在这一步骤中作了两点改动:其一是加入了认证信息$\{c, y_{c,temp}, \text{lifetime}\} \hat{} x_{ca}\} \hat{} x_{cy}$,这是为了让ElGamal Yaksha TGS服务器能够通过向认证中心CA申请,以得到客户端的公钥证书$\{\text{cert}_c\} \hat{} x_{ca}$(其中$x_{ca}$为CA的ElGamal私钥,其ElGamal公钥是公开的).利用客户端的公钥解密$\{c, y_{c,temp}, \text{lifetime}\} \hat{} x_{ca}\} \hat{} x_{cy}$得到客户的临时ElGamal公钥$(y_{c,temp}, x_{c,temp})$,从而对消息$\{\{T_{c,tgs}\} \hat{} x_{tgsy}\} \hat{} x_{c,temp}$进行解密得到$\{T_{c,tgs}\} \hat{} x_{tgsy}$.然后利用自己的Yaksha私钥$x_{tgug}$对消息进行进一步解密得到票据$T_{c,tgs}$,而票据中含有与客户会话的密钥$K_{c,tgs}$.其二,对票据的加密是用TGS的Yaksha私钥和客户的临时ElGamal私钥进行加密的,这可以防止一旦KDC被攻破,无法给"假"客户端有效的票据(TGT)的现象的发生,同时也验证了客户端的身份.

第4步:ElGamal Yaksha TGS服务器收到消息解密并得到票据后,向客户端发出:

Kerberos: $tgs\_to\_c: \{K_{c,s}\}K_{c,tgs}, \{T_{c,s}\}K_s$,

ElGamal Yakasha: $tgs\_to\_c: \{K_{c,s}\}K_{c,tgs}, \{T_{c,s}\} \hat{} x_{sy}$

$\{\{T_{c,tgs}\} \hat{} x_{tgsy}\} \hat{} x_{tgs,tgs}$.

与上一步相同,客户端向认证中心CA申请得到ElGamal Yaksha TGS的公钥,利用此公钥来验证ElGamal Yaksha TGS服务器是否对$T_{c,tgs}$进行了有效的签名(因为$T_{c,tgs}$中含有会话密钥$K_{c,tgs}$,而客户端是知道$K_{c,tgs}$的).一个被攻破的ElGamal Yaksha TGS服务器由于不知道$X_{tgs,tgs}$,无法产生一个有效的票据$T_{c,tgs}$.在这一点上,可以说是对ElGamal Yaksha TGS服务器的身份认证.验证通过后,客户端利用$K_{c,tgs}$得到同远端服务器会话的密钥$K_{c,s}$.

第5步:客户端得到$K_{c,s}$后,向远端服务器发出:

Kerberos: $c\_to\_s: \{c, ts, ...\}K_{c,s}, \{T_{c,s}\}K_s$,

ElGamal Yakasha: $c\_to\_s: \{c, ts, ...\}K_{c,s}\{\{T_{c,s}\} \hat{} x_{sy}\} \hat{} x_{c,temp}$

$\{\{c, y_{c,temp}, \text{lifetime}\}x_{ca}\} \hat{} x_{cy}$.

远端服务器收到消息后,从可信任认证中心CA处取得客户端的公钥对$\{c, y_{c,temp}, \text{lifetime}\} \hat{} x_{ca}\} \hat{} x_{cy}$进行解密,得到客户端的临时ElGamal公钥$(y_{c,temp}, x_{c,temp})$,并进一步利用$x_{c,temp}$和自己的Yaksha私钥$x_{tgstgs}$对消息$\{\{T_{c,tgs}\} \hat{} x_{tgsy}\} \hat{} x_{c,temp}$解密得到票据$T_{c,tgs}$,而如前所述,此票据中既含有同客户会话的密钥$K_{c,tgs}$,又含有客户端的ID号,这样就验证了客户端的身份.

第6步:同样,如果需要交互认证,服务器将向客户端证明自己的身份:

Kerberos: $s\_to\_c: \{ts...\}K_{c,s}$,

ElGamal Yaksha: $s\_to\_c: \{\{ts...\} \hat{} x_{sy}\} \hat{} x_{ss}$.

这样,客户端同远程服务器即可基于Kerberos协议安全地进行通信.

4 结束语

为了在Kerberos协议中集成公钥体制,解决因对称算法带来的局限性,许多人提出了不同的解决方案,但无论是J. Tarso和K. Alagappan的SPX系统,还是Bellowin和Merrit的加密密钥交换系统[10],都与Kerberos协议有着较大的差别,将它们集成到Kerberos中都需要重新构造Kerberos系统.为了使修正后的协议既要对原协议改动不大,又不能带来安全性的降低,因此任何对

Kerberos 协议的改动,都应秉承它原有的机制和风格.我们提出的基于 ElGamal 的 Yaksha 系统是对原有 Yaksha 的补充和发展,本文在此基础上,进一步利用其对 Kerberos 协议进行了一定程序的修正,解决了本文第 2 节中指出的 Kerberos 协议的几个重要的局限性问题.

References:

[1] Tardo, J., Alagappan, K. SPX: global authentication using public-key certificates. In: Proceedings of the 1991 IEEE Computer Society Symposium on Security and Privacy. IEEE Computer Society Press. 1991. 232~244.

[2] Kohl, J., Neuman, B., Ts'o, T. The evolution of the Kerberos authentication service ver.5. In: Brazier, F., Johansen, D., eds. Distributed Open Systems. IEEE Computer Society Press, 1994. 78~94.

[3] Neuman, B., Ts'o, T. Kerberos: an authentication service for computer networks. IEEE Communications, 1994,32(9).

[4] Needham, R.M., Schroeder, M.D. Using encryption for authentication in large networks of computers. Communications of the ACM, 1978,21(12):993~999.

[5] Bellovin, S.M., Merritt, M. Limitation of the Kerberos authentication systems. ACM SIGCOMM Computer Communication Review, 1990,20(5):119~132.

[6] Anderson, R.J., Lomas, T.M.A. Fortifying key negotiation schemes with poorly chosen passwords. Electronics Letters, 1994,30(13):1040--1041.

[7] Ganesan, R. Yaksha: augmenting Kerberos with the public key cryptography. In: Proceedings of the Internet Society Symposium on Network and Distributed System Security. IEEE Computer Society Press, 1995. 132~143.

[8] Ganesan, R. The Yaksha security system. Communications of the ACM, 1996,39(3):55~60.

[9] Meng, Yang, Liu Ke-long, Qing Si-han. A new type of YAKSHA security system. Journal of Software, 2000,11(5):616~619 (in Chinese).

[10] Bolovin, S.M., Merrit, M. Encrypted key exchange: password-based protocol secure against dictionary attacks. In: Proceedings of the 1992 IEEE Computer Society Conference on Research in Security and Privacy. 1992.

附中文参考文献:

[9] 蒙杨,刘克龙,卿斯汉.一种新型综合安全系统研究.软件学报,2000,11(5):616~619.

An Improved Way on Kerberos Protocol Based on Public-Key Algorithms*

LIU Ke-long, QING Si-han, MENG Yang

(*Engineering and Research Center for Information Security, The Chinese Academy of Sciences, Beijing* 100080, *China*);
(*Institute of Software, The Chinese Academy of Sciences, Beijing* 100080, *China*)

E-mail: lkl@ercist.iscas.ac.cn

http://www.ercist.ac.cn

Abstract: To overcome the Kerberos' limitations caused by using symmetric algorithm, an improved way using Yaksha security system of ElGamal Public-Key algorithm is presented based on the original protocol framework.

Key words: ElGamal algorithm; Yaksha mechnism; Kerberos authentication protocol; certificate authority

* Received June 16, 1999; accepted March 8, 2000

Supported by the National Natural Science Foundation of China under Grant No. 60083007; the National Grand Foundamental Research 973 Program of China under Grant No. G1999035810