◎博士论坛◎

SSL 握手协议中客户端平衡密钥交换算法

齐 芳[1],贾维嘉[1,2],王国军[1]
QI Fang[1],JIA Wei-jia[1,2],WANG Guo-jun[1]

1.中南大学 信息科学与工程学院,长沙 410083
2.香港城市大学 计算机科学系,香港
1.School of Information Science and Engineering,Central South University,Changsha 410083,China
2.Department of Computer Science,City University of Hong Kong,Hong Kong,China
E-mail:csqifang@mail.csu.edu.cn

QI Fang,JIA Wei-jia,WANG Guo-jun.Client balanced secret exchange algorithm in SSL handshake protocol.Computer Engineering and Applications,2007,43(19):1-3.

Abstract: The primary goal of the Secure Socket Layer(SSL) protocol is to provide confidentiality and data integrity between two communicating entities.Since the step that is most computationally expensive and causes computational imbalance between clients and server in the SSL handshake protocol is the decryption computation of the server,we show that a client balanced secret exchange algorithm can be used to speedup SSL session initialization and undertake the previous computation tasks of server's decryption.It is also introduced that the estimation strategy of parameter which is the size of clients which will be decrypted at the same time.Finally,the proposed algorithm is evaluated to be efficient through simulation studies.
Key words: Secure Socket Layer(SSL);handshake protocol;secret exchange algorithm;decryption computation

摘 要:SSL 协议的基本设计目标是为两个通信实体之间提供数据的保密性和完整性。由于在 SSL 握手协议中最耗费计算资源和造成客户端与服务器端计算不平衡的步骤是服务器端解密运算,提出了客户端平衡的密钥交换算法,用来加速 SSL 会话的初始化和承担服务器端的解密的预运算。对算法中的同时对多个客户的请求进行解密的粒度的估计策略进行了阐述。模拟实验表明所提出的方案是有效的。
关键词:安全套接层协议;握手协议;密钥交换算法;解密运算
文章编号:1002-8331(2007)19-0001-03 **文献标识码**:A **中图分类号**:TP393

1 引言

伴随着在线交易如网上银行和电子商务应用的日益普及,因特网的服务模式已经由传统的信息浏览模式向电子交易转变,经常需要传输敏感信息。SSL(Secure Socket Layer,安全套接层协议)以及它的后续版本 TLS(Transport Layer Security,传输层安全协议)以多种公钥密码技术和对称加密技术为基础,实现客户身份鉴别、数据加密传输、数据完整性校验、数字签名等安全功能,是目前安全 Web 服务应用环境中最有前景的解决方案之一[1]。已有研究表明,SSL 协议对于客户端,仅仅是增加了握手的开销和随后加解密传输数据的开销,而对于服务器,在大量客户端连接的情况下,同时并发的握手就会占用大量的 CPU 资源,从而导致服务器处理能力的急剧下降[2]。国内外已经有大量的研究,从硬件和软件的角度考虑提高 SSL 协议在服务器端的性能[2-5]。

本文在总结了这些方法的基础上,研究了客户端平衡的密钥交换算法,用来加速 SSL 会话的初始化和承担服务器端的解密的预运算。第 2 章提出了客户端平衡的密钥交换算法(Client Balanced Secret Exchange Algorithm,简称 CBSEA);第 3 章阐述了该算法同时对多个客户的请求进行解密的粒度的估计策略;第 4 章是实验结果及分析;第 5 章是对研究工作进行总结和展望。

2 CBSEA 算法描述

本章定义 1 描述的是由 Rivest-Shamir-Adleman 等人提出

基金项目:国家重点基础研究发展规划(973)(the National Grand Fundamental Research 973 Program of China under Grant No.2003CB317003);国家自然科学基金(the National Natural Science Foundation of China under Grant No.60503007);国家教育部新世纪人才支持计划(the Program for New Century Excellent Talent from MOE of China)。
作者简介:齐芳(1978-),女,博士,讲师,主要研究领域为网络安全,移动计算;贾维嘉(1957-),男,博士,教授,博士生导师,主要研究领域为无线通讯,移动计算,并行和分布式计算;王国军(1970-),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为计算机网络,容错与可信计算,软件工程。

的 RSA 公钥算法，也是在 SSL 协议中使用的标准密钥交换算法。由定义 2 到定义 5 分别阐述了 CBSEA 算法的数据结构、加密过程、中国剩余定理对于解密阶段的质数划分以及解密过程。最后结合定义 5 的图示，给出了解密步骤的范例。该算法是加速了 SSL 会话的初始化和客户端的承担服务器端的解密的预运算，使用了同时对多个客户进行一次性解密的方法，并且利用中国剩余定理的思想进行分解，相对于逐一进行 RSA 算法的解密，提高了解密时间效率，在文章第 4 章实验结果部分进行了验证。CBSEA 算法中减轻客户端和服务器端计算不平衡性主要体现在定义 3 的步骤 2，由客户端承担服务器端的解密的预运算。

定义 1 RSA(Rivest-Shamir-Adleman)算法[6]。RSA 算法中公开密钥和私有密钥是一对大素数 p、q(100 到 200 个十进制数或更大)的函数。从一个公开密钥中恢复出明文的难度等价于分解两个大素数之积 $N(N=pq)$。随机选取加密密钥 e，使得与 N 的欧几米德函数 $\phi(N)=(p-1)(q-1)$互质。最后计算解密密钥 d，满足 $d=e^{-1} \bmod((p-1)(q-1))$。$d$ 与 N 也互为质数。(e, N)是公开密钥，d是私有密钥。加密的过程为将明文消息 m 分解成小于 N 的长度的密文 m_i，加密得到相同长度的密文 c_i组成密文 c，其中 $c_i=m_i^e \bmod N$。解密消息时，取每一个加密后的长度 c_i计算 $m_i=c_i^e \bmod N$。RSA 算法的公钥的指数在 X.509 证书中建议采用 65 537 位(65 537 的二进制表示中只有两个 1，所以它只需要 17 次乘法来实现指数运算)。

定义 2 CBSEA 算法数据结构。定义一棵为二进制的完全二叉树的解密树 T_d。以 $e_1,\cdots,e_b$作为二进制二叉树的叶结点的权值($e_1,\cdots,e_b$选取原则在定义 3 中给出)。完全二叉树加密过程中的每个中间结点上的权值为其左孩子结点和右孩子结点上的权值的乘积。定义 d_i为以 e_i标记为叶结点的树枝的长度。T_d的构建按照使得权值 $W=\sum_{i=1}^{b} d_i \log e_i$取得最小值的原则，过程与构建哈夫曼编码树的过程类似[3]。

按照定义 2 对 CBSEA 算法数据结构的定义，在定义 3 中的客户端加密，利用了数据结构的叶结点来表示服务器端的解密输入。

定义 3 CBSEA 客户端加密。设算法中同时对多个客户的请求进行解密的粒度为 b，且设定 b 为偶数。

步骤 1 选取一系列的不相同的两两互为质数的公钥指数 $e_1,\cdots,e_b$，即 $\gcd(e_i,e_j)=1, i\neq j$ 并且共享一个公钥的模 N。并要求 $0<e_1,\cdots,e_b<\phi(N)$，同时满足与 N 的欧几米德函数 $\phi(N)=(p-1)(q-1)$是互质的，即 $\gcd(\phi(N),e)=1$。n 表示模 N 的二进制比特长度。给定 b 个明文消息 $m_1,\cdots,m_b$，$(0<m_1,\cdots,m_b<N)$，分别用对应的秘密密钥 e_i加密，得到 b 个消息密文 $v_1,\cdots,v_b$(i.e. $v_i=m_i^{e_i} \bmod N$)。如图 1 叶子结点的输入 $v_1,\cdots,v_4$所示(例如 $b=4$)。

步骤 2 进一步计算 b 个消息密文 $v_1',v_2',\cdots,v_b'$。如果 i 为奇数，则取 $v_i'=v_i^{e_{i+1}} \bmod N$；如果 i 为偶数，则取 $v_i'=v_i^{e_{i-1}} \bmod N$。如图 2 中的二叉树的叶子结点输入 v_1^5、v_2^3、v_3^{11}和 v_4^7所示(例如 $b=4$)。

定义 3 中步骤 2 的实现是在设客户端已经得到作为服务器端公钥加密指数的质数表的基础上，将 $e_1,\cdots,e_b$按照顺序两两配对。这一步骤在公钥系统中是作为服务器端解密的预运算。但在该算法中，质数表对客户端已知，实现这一步骤是可行的。将部分本来由服务器端的运算，转到各个客户承担，减轻了服务器端和客户端的计算不平衡。

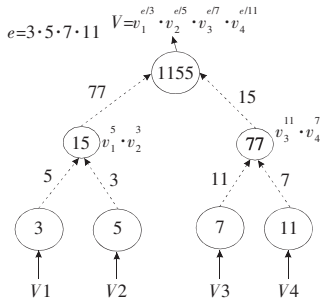

图 1　客户端加密步骤 1

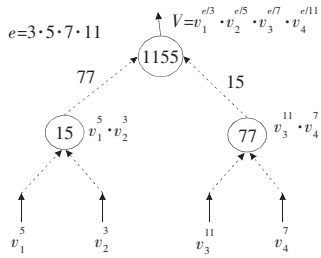

图 2　客户端加密步骤 2 和服务器端解密步骤 1

定义 4 中国剩余定理[3]的质数分类。如果已知 X 是关于两两互为质数的一些质数 $e_1,\cdots,e_b$同类余集，就可以重构该数。即 $X=0(\bmod e_1)$，$X=0(\bmod e_2)$，$\cdots$，$X=0(\bmod e_k)$，$X=1(\bmod e_{k+1})$，$X=1(\bmod e_{k+2})$，$\cdots$，$X=1(\bmod e_b)$，由中国剩余定理可得到方程组 X 有唯一解[5]。设定同类余集为 0 的指数集 $e_1,\cdots,e_k$为解密树 T_d左子树的叶结点上的质数集，则同类余集为 1 的 $e_k,\cdots,e_b$为解密树 T_d右子树的叶结点上的质数集。

定义 5 CBSEA 服务器端解密。

步骤 1 解密乘积阶段：使用该树作为引导，从叶结点到根结点进行对整个解密树的构建。构建的过程是一个迭代的过程。计算每一个内部结点时，计算以左孩子结点的值 L 为底数，右边树枝的幂的乘积迭代结果 E_R为幂，同时以右孩子结点的值 R 为底数，左边树枝的幂的乘积迭代结果 E_L为幂。得到结点的值 $L^{E_L}\cdot R^{E_R}$。构建解密树的结果在根结点得到 $v=\prod_{i=1}^{b} v_i^{e/e_i} \bmod N$，其中 $e=\prod_{i=1}^{b} e_i$，如图 2。

步骤 2 解密分解阶段：以 T_d的根结点得到的结果 V(eg. $V=v_1^{e/3}\cdot v_2^{e/5}\cdot v_3^{e/7}\cdot v_4^{e/11}$)进行进一步的推导，$m=v^{1/e} \bmod N=\prod_{i=1}^{b} v_i^{1/e_i} \bmod N$。接下来的目标是以 T_d作为引导，与解密树的构建过程相反，从根结点到叶结点对整个解密树进行分解。将根结点 m 分解成为两个左右子树结果 m_L和 m_R。这个分解已经隐含在构建二进制解密树的过程中。定义 X，按照定义 4 的中国剩余定理的质数分类，同时满足 $X=0 \bmod E_L$和 $X=1 \bmod E_R$。接下来定义 $X_L=X/E_L$和 $X_R=(X-1)/E_R$。可以迭代的过程，得到 $m_R=m^X/(v_L^{X_L}\cdot v_R^{X_R})$，$m_L=m/m_R$，重复这个迭代过程，最终分解成 b 个因子，

$m_i=v_i^{1/e_i}$。如图 3。

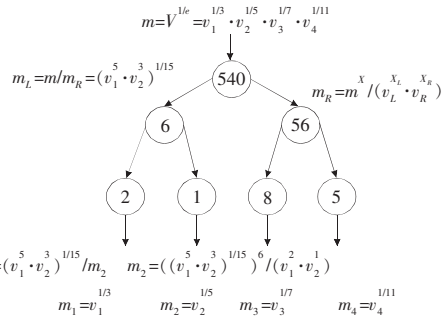

图 3　服务器端解密步骤 2

例　解密步骤 2 的数学描述:为了使计算时间更小,公钥指数 $e_1,\cdots,e_b$ 选取不同的最小的质数[5]。以 b=4 为例,如图 3 左子树所示:公钥指数 e_1 和 e_2,如 e_1=3 和 e_2=5。可以在大约对一个密文进行解密的时间代价内,解密两个密文[4]。其中如定义 5 中步骤 1e_1=3 和 e_2=5 分别对应于 E_L 和 E_R。设 v_1 和 v_2 是用公钥密钥对$(N,3)$和$(N,5)$加密后得到的密文。按照定义 5 的步骤 1 构建解密树的过程计算 $m=(v_1^5\cdot v_2^3)^{1/15} \bmod N$ 的值。按照定义 5 的步骤 2,进行分解的过程,为了得到解密的结果 m_1 and m_2,分别计算 $v_1^{1/3}$ 和 $v_2^{1/5} \bmod N$ 的步骤如下:

步骤 1　$\begin{cases}X=0 \bmod E_L\\X=1 \bmod E_R\end{cases}=\begin{cases}X=0 \bmod 3\\X=1 \bmod 5\end{cases}\Rightarrow X=6$

步骤 2　$X_L=\dfrac{X}{E_L}=\dfrac{6}{3}=2\quad X_R=\dfrac{X-1}{E_R}=\dfrac{6-1}{5}=1$

步骤 3　$m_1=v_1^{1/3} \bmod N=\dfrac{m^{10}}{v_1^3\cdot v_2^2}\quad m_2=v_2^{1/5} \bmod N=\dfrac{m^6}{v_1^2\cdot v_2}$

3　粒度 b 的估计策略

本文对算法 CBSEA 中同时对多个客户的请求进行解密的粒度 b 进行估计。令 b 近似等于 λ/μ,其中 λ 表示客户的平均到达速率,μ 表示服务器的平均服务速率。为了满足客户对于稳定性的要求,b 个客户的平均到达率应小于服务器对于 b 个客户同时进行处理的平均服务速率。即对 b 的选取满足 b 个客户的平均服务时间 τ 应小于连续 b 个客户的平均到达时间,即 b 与泊松分布的到达时间间隔的乘积。定理 1 详细地描述了排队模型稳定性约束,并给出了证明。

定理 1　排队模型稳定性约束。为了满足客户对于稳定性的要求,即系统中的平均到达速率应小于平均服务速率,系统才能达到稳定状态,在系统中的客户长度不会无限长。而对 b 个客户的平均服务时间 τ 应小于算法 CBSEA 中的同时对多个客户的请求进行解密的粒度为 b 与泊松分布的到达时间间隔的乘积。其中 λ 表示客户的平均到达速率,$\tau<b/\lambda$。

证明　令 $X_i(i=1,2,\cdots)$为两个连续请求的到达时间分布,设 Y 为 b 个连续到达时间的分布。如果在 M/D/1 模型中当时间 t 趋向于无穷,系统趋于稳定状态时,满足服务器对 b 个客户同时进行解密的平均服务时间间隔 τ 应该满足小于平均到达时间间隔,即 $T_b<E(Y)$。这里 $E(Y)$是分布 Y 的期望值。因为是独立同分布的变量,所以 b 个连续的请求的平均到达时间为:

$$E(Y)=E(\sum_{i=1}^{b}X_i)=bE(X_i)=\frac{b}{\lambda}$$

定理得证。

由定理 1 可知,即选取 b 近似等于 λ/μ,同时满足 $\tau<b/\lambda$。而对于这一估计值和约束条件得到的结果的正确性和可行性,在本文第 4 章的实验结果中得到了验证。

4　实验结果

在实验中,服务器端配置为 Dell Intel Pentium IV 处理器,3.20 GHz 主频和 1 GB RAM。算法中公钥秘钥长度设定为 1 024 bit。SSL 协议使用 48 Byte 长度的字符串作为 Pre-MasterSecret。假设明文长度为 384 bit。

表 1 验证了对于算法 CBSEA 中 b 的策略的近似解。对于相对小的到达率,b 几乎和通过模型计算出来的解相同。由于客户的平均到达率较小,测试结果中当 λ<15 时,只有非常小的几率需要进行改进密钥交换算法,所以 b 的解也相对小。当 b 的取值小于 2 时,用定义 1 的 RSA 算法作为密钥交换算法。在比较高的到达率时,分析模型的结果和试验结果也非常接近。对于约束条件,即当到达率满足 λ<30 时,b 随着到达率 λ 的增加而增加。实验结果表明,当到达率大约 λ>30 时,约束条件的范围,随着到达率 λ 的增加反而减小。

表 1　CBSEA 粒度 b 验证

λ	CBSEA 粒度 b					
	理论值			测试值		
	约束前	约束条件≤	约束结果	约束前	约束条件≤	约束结果
10	1	8	1	1	8	1
15	2	10	2	2	10	2
20	2	12	2	2	12	2
30	4	12	4	4	12	4
40	4	10	4	4	12	4
50	6	10	6	6	10	6
60	6	8	6	6	8	6
70	8	6	6	8	8	8
80	8	6	6	8	6	6
90	10	4	4	10	4	4
10	10	4	4	10	4	4

本文实验中定义的响应时间是指从客户发起 SSL 请求到服务器收到 SSL 请求随后响应返回到达客户所经历的时间。图 4 纵坐标轴表示的是平均响应时间,横坐标轴是 CBSEA 粒度 b。如图 4 所示,当 λ=30,平均响应时间 T 也随着到达率的增加而基本符合线性增长的规律。平均响应时间 T 的分析和实验结果,表明算法 CBSEA 使用表 1 估计粒度值 b,取得了较好性能。

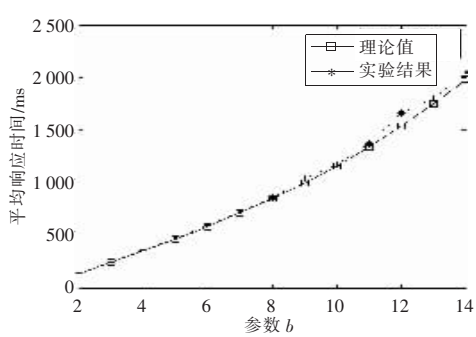

图 4　平均响应时间的验证

在理论分析值和实验结果中,当客户到达率大于平均客户处理服务的时间时,即 $\lambda>1/T_{rsa}=1/0.032=31.25$,其中 T_{rsa} 为实验

(下转 6 页)