文章编号:1001 - 9081(2006)05 - 1141 - 03

SRTP 协议包索引维护算法的分析与改进

吴 斌[1],潘雪增[1],Douglas L. Whiting[2]
(1. 浙江大学 计算机科学与技术学院,浙江 杭州 310027; 2. Hifn 公司,美国 加州 洛斯盖多斯 95032)
(bin_wu@saianmicro.com)

摘 要:安全实时传输协议(Secure Real-time Transport Protocol,SRTP)给 RTP 流提供了机密性、完整性以及抗重播攻击等安全特性。实现 SRTP 的难点之一是 SRTP 包索引的计算以及 ROC 与 s_l 的更新。分析并改进了 RFC3711 中提供的包索引维护算法,以提高在硬件流水和多处理器体系结构下 SRTP 的性能,并以 Hifn 公司的 HIPP 芯片实现为例比较了算法改进前后的区别。

关键词:安全实时传输协议;实时传输协议;包索引;溢出计数器;高智能包处理
中图分类号: TP393.04 **文献标识码**:A

Analysis and improvement on packet index maintenance algorithm of SRTP

WU Bin[1], PAN Xue-zeng[1], Douglas L. Whiting[2]
(1. *College of Computer Science, Zhejiang University, Hangzhou Zhejiang* 310027, *China;*
2. *Hifn Company, Los Gatos California* 95032, *America*)

Abstract: SRTP provided confidentiality, message authentication, and replay protection to the RTP traffic. One of the hard point to implement SRTP was to calculate the packet index and update the ROC and s_l. The packet index maintenance algorithm suggested in the RFC3711 was analyzed, based on which introduced an optimized algorithm to improve the SRTP performance under hardware pipeline and/or multiprocessor architecture implementation, and a comparison on the two algorithms was done based on the HIPP security chip's SRTP implementation.

Key words: SRTP; RTP; packet index; ROC(Rollover Counter); HIPP

0 引言

SRTP 作为 RTP 的一个扩展协议,定义了对 RTP 与 RTCP 流进行加密、认证以及抗重播攻击等的实现框架[1]。

RTP[2]传输流工作在 UDP 之上以满足实时要求。RTP 包结构中定义了 16 位的序列号(Sequence Number,SEQ)用于接收方检测包丢失以及重新排序,在此基础上 SRTP 定义了 32 位的溢出计数器(ROC)用于记录 RTP 包中序列号溢出 16 位(65 535)的次数,其目的是控制密钥的安全性。所以 SRTP 包索引(Index)由 ROC 与 SEQ 组成,总共有 48 位:Index = ROC < <16 + SEQ,其中 32 位的 ROC 不在 SRTP 包中传输,SRTP 包的格式[1]见表 1。

表 1 SRTP 包的格式

V=2	P	X	CC	M	PT	Sequence number
Timestamp						
Synchronization source identifier (SSRC)						
Contributing source identifier(s) (CSRC)						
RTP Header Extension (optional)						
Encrypted Payload…						
…			padding			pad count
SRTP MKI (optional)						
Authentication tag (recommended)						

由于 ROC 不在 SRTP 包中传输,而包索引 INDEX 在从主密钥(Master Key)生成会话密钥(Session Key)以及 RTP 包加解密时都要用来计算初始向量 IV,参与 SRTP 会话的通信各方必须保证该 IV 一致(否则处理出错),因此必须共同维护并同步 ROC 的值。RFC3711 3.3.1 分析了一个基于 SEQ 以及 s_l 来估计 ROC 的算法(文中称之为算法 1)。虽然算法 1 实现了预期的功能,但在硬件流水和多处理器体系结构下会大大影响 SRTP 的性能。本文在对算法 1 进行深入分析的基础上提出了改进的算法 2,并以 HIPP 安全芯片实现为例比较了两个算法的区别。

1 算法 1 分析

1.1 ROC 与 s_l 的初始化

通信各方在 SRTP 会话建立时初始化 ROC 为 0。发送者在 SEQ 溢出 16 位时对 ROC 加 1;接收者用接收到的第一个包的 SEQ 初始化 s_l,对于以后收到的包根据其 SEQ 以及维护的 ROC、s_l,三者的关系估计当前包所属的 ROC,记之为 ROC1,ROC1 将用于计算 HMAC 以及加解密用的 IV,根据处理结果决定是否更新当前所维护的 ROC 与 s_l。

1.2 新 ROC 的估计(Estimation)

ROC1 从 ROC-1,ROC,ROC + 1 中选择其一,使得当前接收的 SRTP 包的索引(由 ROC1 与 SEQ 组成)"尽量靠近"已经接收并成功处理的 SRTP 包的最大索引(由 ROC 与 s_l 组成)。这个假设是符合实际的,因为 UDP 虽然不能保证有序传输,但相邻时间接收的包索引相差不会太大,只要新接收包的索引在已接收并成功处理的包的最大索引的[- 32 768, + 32 767]范围内,根据算法所估计的值都是正确的。

收稿日期:2005 - 11 - 23;**修订日期**:2006 - 01 - 11

作者简介:吴斌(1982 -),男,浙江义乌人,硕士研究生,主要研究方向:计算机系统结构、网络安全; 潘雪增(1942 -)男,教授,博士生导师,主要研究方向:计算机系统结构、安全操作系统、网络安全、分布式数据库; Douglas L. Whiting,加州大学计算机博士.

图 1 形象表示了算法 1 的思想。横向从右到左表示序列号[0,$2^{16}-1$],纵向自下而上表示 ROC 值[0,$2^{32}-1$]。假设当前 ROC = 2,而 s_l = n。根据算法 1,当 s_l < 32 768 时(图 1(a)),如果当前接收到的包的序列号 SEQ 在图中 C 的范围[n + 32 768,65 535],那么 ROC1 = ROC - 1,否则 ROC1 = ROC。当 s_l >= 32 768 时(图 1(b)),如果当前接收到的包的序列号 SEQ 在图中 B 的范围[0, n - 32 768],那么 ROC1 = ROC + 1,否则 ROC1 = ROC。

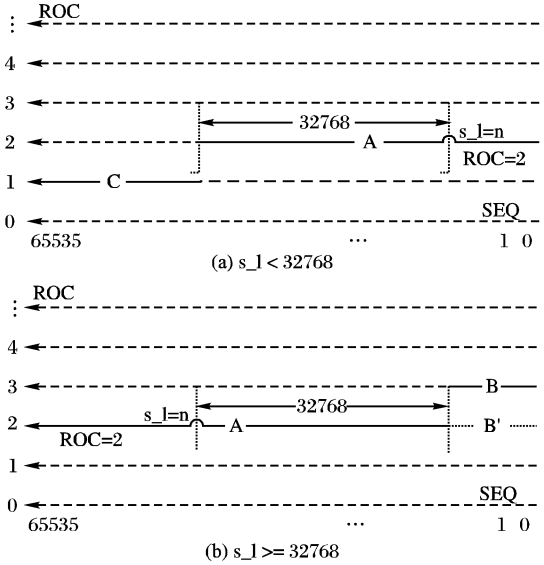

图 1 ROC1 与 ROC、s_l、SEQ 的关系

1.3 ROC 与 s_l 的更新(Update)

ROC1 计算出来以后要用于 HMAC 的计算,而包索引(INDEX = ROC1 < < 16 + SEQ)要用于计算初始向量 IV。在包处理完之后,如果认证成功,则要更新 ROC 与 s_l。更新算法的思想是包索引"向上"更新,或者说包索引的值往大的方向单调更新,以始终保证 ROC 与 s_l 组成的包索引是当前已经成功处理包的最大索引,从而保证 ROC 估计算法的正确性。具体描述如下:在估计 ROC1 时,如果:

1) ROC1 = ROC - 1:ROC 与 s_l 都不需更新;

2) ROC1 = ROC,SEQ > s_l:ROC 不变,s_l 更新为 SEQ;

3) ROC1 = ROC,SEQ <= s_l:两者都不更新;

4) ROC1 = ROC + 1:两者都更新,ROC 更新为 ROC1,s_l 更新为 SEQ。

一般来说,在 SRTP 通信过程中,2)在顺序接收情况下发生,概率最高;3)是乱序的情况下发生;1)跟 4)发生在 s_l 接近 2^{16} 或者 0 附近,概率很低。

1.4 深入分析

在恶劣的网络环境下,ROC1 的计算有可能不符合实际情况。如图 1 中 s_l > = 32 768 的情况,如果 SEQ 在[0,n - 32 768]的范围,根据算法 1 我们就认为它属于 B(ROC 加 1)而不是 B′(ROC 不变),而实际上属于 B′也是有可能的。那如果判断错误会发生什么?认证(如果被配置)会失败,因为认证的范围除了 RTP 包,还包括了 ROC1 的值。IETF 推荐 SRTP 使用认证,认证不仅保证 SRTP 包的完整性,还保证了通信各方 ROC 一致性。

虽然更新 ROC 与 s_l 本身很简单,但更新必须以认证成功为前提,否则很容易受攻击[1]。算法 1 要求对接收到的每个包都判断是否要更新,而概率最高的情况 2)是要更新的(顺序接收情况下每个包都要更新 s_l),由于更新需要等待延时相对比较大的认证操作的完成,这给硬件流水实现带来一定的性能损失(如果更新概率很高,就会频繁的打断流水操作);而在多处理器体系结构下需要解决参数在各处理器缓存(Cache)之间的同步[3]问题,频繁的参数更新同样也给多处理器环境下的实现增加了负载。

2 改进的算法 2

在功能上,算法 1 很好地解决了通信各方包索引的同步问题。但在效率上存在改进的空间。改进的思想是尽量减少更新 ROC 和 s_l 的概率,以减少对认证结果的等待。维护 s_l 的最高位,称之为 MSB(Most Significant Bit),而不是所有 16 位可以达到上述目的。ROC 与 MSB 的初始化同算法 1 一样,不过 MSB 替代了 s_l,相当于 s_l 的最高位。改进的算法称为算法 2,其流程如图 2 所示。

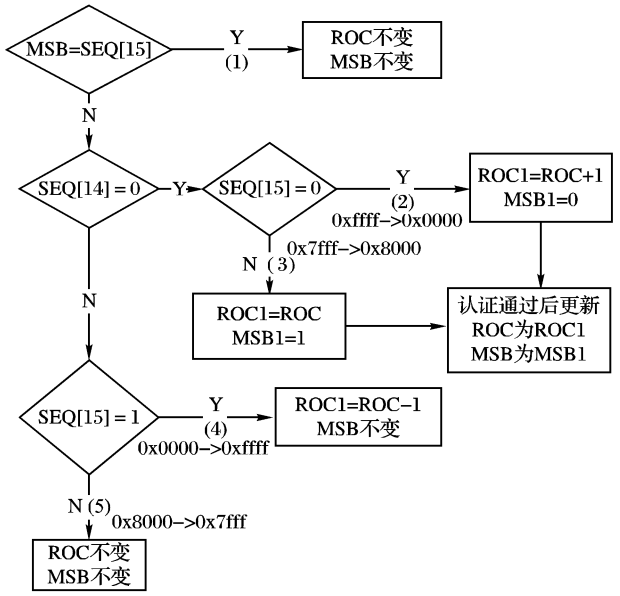

图 2 改进的序列号维护算法

图 2 既包括了 ROC1 的估计算法,也包括了 ROC 与 MSB 的更新算法。在估计 ROC1 时,图中用(1) ~ (5)标出了 5 种处理可能,其中(2) ~ (5)标出了满足相应条件的典型情况的 SEQ 值的变化。SEQ[15]表示 SEQ 的第 15 位(最高位)。图 2 可以对应到图 1 的相应部分:(1)、(3)、(5) 三种情况的 SEQ 组成图 1 中的 A 的范围(ROC 不变);(2)等于 B 的范围(ROC 加 1);(4)等于 C 的范围(ROC 减 1)。

认证通过后的 ROC 与 MSB 的更新算法保持了包索引"向上"更新的原则,也可以对应到算法 1 的 4 种更新情况,图 2 中只列出了需要更新的 2 种情况。改进前后的算法在功能上是一致的,可以说算法 2 细化了对 SEQ 的判断,虽然判断稍微复杂,但判断的结果更有价值。

在顺序接收的情况下,MSB 需要更新的概率为 $1/2^{15}$,而算法 1 中 s_l 的更新概率为 1;ROC 需要更新的概率为 $1/2^{16}$。在硬件流水以及多处理器体系环境下,即使处理一次更新所需的代价也比较大,但是由于更新的概率很低,影响基本可以忽略。

3 两算法在 HIPP 上的实现比较

HIPP[4]是美国 Hifn 公司开发的高性能网络安全处理芯片,已经支持 PPTP、IPSec、SSL 等多种流行的安全协议。芯片采用流水线体系结构,分单处理结构(如 HIPP 7855)与多处理结构(HIPP Ⅱ8155,由 4 个处理芯片组成)。

图 3 是 HIPP 处理芯片配置成 Encode 模式时内部流水结构简图,Decode 模式加密与认证两个引擎在流水线上的位置互换。图中省略了 SRTP 实现不需要的压缩(Compression)、对齐(Padding)以及校验和(Checksum)模块。DPU(Dynamic Protocol Unit)是一个嵌入式微处理器,有自己简单的指令集,

它以执行二进制代码的方式控制每个功能模块,可以在数据流的适当位置打开/关闭各个功能引擎,以对数据进行不同的处理。通过编程可以使 DPU 处理不同的安全协议,它是 HIPP 芯片可以支持多安全协议的关键模块。

在 Encode 时,输入为 RTP 包,输出为 SRTP 包,Decode 时相反。会话所采用的加密、认证算法由配置模块设置,而对每个包的处理由 DPU 通过执行事先编译好的可执行代码来控制,作者已经用 DPU 的指令集实现了 SRTP(包索引维护算法采用改进的算法 2),并编译成二进制可执行代码,DPU 执行该代码就可以完成对 SRTP 处理的控制。

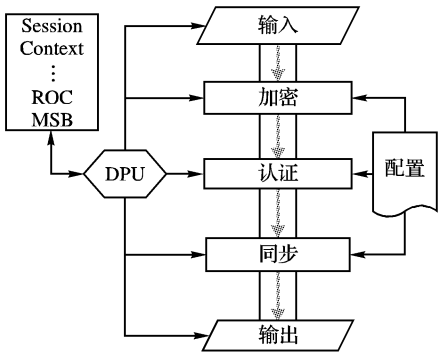

图 3　HIPP 的流水体系结构

HIPP 体系结构下,芯片在做 Decode 处理时认证引擎计算并比较 HMAC 的结果,该结果由认证引擎以一个标志位通知发送该包到芯片来处理的客户程序,而 DPU 无法知道该结果(这样设计的目的是优化整个流水线的效率,DPU 无需等待认证结束而可以尽快启动下一个操作)。SRTP 实现中 ROC 等的更新维护由 DPU 控制,更新前 DPU 必须知道认证的结果,DPU 完成这个功能需要控制 3 遍流水操作,认证引擎只负责计算 HMAC,其比较由 DPU 自己来完成:

(1) 正常处理,如果发现 ROC 与 MSB 需要更新,触发第二遍流水;

(2) 控制认证引擎计算 HMAC 并将结果添加到数据流的末尾,触发第三遍流水;

(3) DPU 比较第二遍计算的 HMAC 值与 SRTP 包中的 HMAC 值是否匹配,如果匹配则进行相应更新,否则返回错误值。最后结束该包的操作。

而如果不需要更新 ROC 等,DPU 也就不需要知道与等待认证的结果(HMAC 的比较可以由认证引擎完成),所有操作可以一遍流水完成。在算法 1 的情况下,需要更新 ROC 与s_l 的概率很高,所以需要多遍流水处理的概率也很高。而算法 2 把多遍处理的概率降低到了 $1/2^{15}$,基本上忽略不计,可以成倍的提高效率。

4　结语

为满足实时的要求,RTP 包一般比较小。SRTP 在为 RTP 提供安全特性的同时所增加的额外数据的大小将会严重影响 RTP 通信效率及实时性。32 位的 ROC 记录了 RTP 包序列号溢出 16 位的次数,但它不在 SRTP 包中传输,而由通信各方共同维护与同步。

ROC 值的同步本身并不复杂,RFC3711 提供了一个算法,很好地完成了同步的功能,但由于该算法要求更新 ROC 与 s_l 的概率太高,并且更新必须在认证通过后进行,导致在硬件流水或者多处理器体系结构下算法实现的代价很高。本文提出一个功能一致但更新概率很低的算法,很好地解决了效率问题,并以美国 Hifn 公司的 HIPP 芯片为例讨论了改进前后算法在实现效率上的区别。

参考文献:

[1] The Secure Real-time Transport Protocol[S]. RFC3711.

[2] RTP: A Transport Protocol for Real-Time Applications [S]. RFC3550.

[3] PATTERSON DA, HENNESSY JL. Computer Architecture: A Quantitative Approach[M]. 3rd edition Chapter 6. 北京: 机械工业出版社, 2002.

[4] http://www.hifn.com/products/Security.html[EB/OL].

(上接第 1140 页)

值 H 时,启动速度控制项,实现 API-V 算法,快速响应网络的动态变化。理论分析和模拟仿真实验表明,API-V 算法能够自适应动态变化的网络环境,性能优于 PI 算法及其已有的改进算法。

参考文献:

[1] HOLLOT CV, MISRA V, TOWSLEY D, *et al.* A control theoretic analysis of RED[A]. AMMAR M, ed. Proceedings of the IEEE INFOCOM[C]. Anchorage: IEEE Communications Society, 2001. 1510 - 1519.

[2] LE L, AIKAT J, JEFFAY K, *et al.* The effects of active queue management on Web performance[A]. Proceeding of the ACM SIGCOMM 2003[C]. Karlsruhe, 2003. 265 - 276.

[3] FLOYD S, GUMMADI R, SHENKER S. Adaptive RED: An algorithm for increasing the robustness of RED's active queue management[EB/OL]. http://www.icir.org/~floyd, 2001.

[4] KUNNIYUR S, SRIKANT R. A time scale decomposition approach to adaptive ECN marking[A]. AMMAR M, ed. Proceedings of the IEEE INFOCOM[C]. Anchorage: IEEE Communications Society, 2001. 1330 - 1339.

[5] ATHURALIYA S, LOW S, LI VH, *et al.* REM: Active queue management[J]. IEEE Network, 2001, 15(3): 48 - 53.

[6] MISRA V, GONG WB, TOWSLEY D. Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED[A]. Proceedings of the ACM SIGCOMM 2000[C]. Stockholm, 2000. 151 - 160.

[7] CHEN Q, YANG OWW. A ST-PI-PP controller for AQM router[J]. IEEE International Conference on Communications, 2004, 27(1): 2277 - 2281.

[8] WANG CG, LI B, SOHRABY K. API: adaptive proportional-integral algorithm for active queue management under dynamic environments[A]. Proceeding IEEE HPSR 2004[C]. 2004. 51 - 55.

[9] CHANG XL, MUPPALA JK, JEN-TE YU. A robust nonlinear PI controller for improving AQM performance[A]. IEEE International Conference on Communications[C], 2004. 20 - 24.

[10] ZENG ZM, ZHANG TK, FEBNG CY, *et al.* An AQM School for Fast Response[J]. Journal of Beijing University of posts and Telecommunications, 2005, 28(4): 5 - 9.

[11] PARK EC, LIM H, PARK KJ, *et al.* Analysis of the virtual rate control algorithm in TCP networks[J]. Globecom '02. IEEE, 2002, 3: 2619 - 2623.

[12] PADHYE J, FIROIU V, TOWSLEY D, *et al.* Kurose J. Modeling TCP throughput: A simple model and its empirical validation[A]. ORAN D, ed. Proceedings of the ACM SIGCOMM[C]. Vancouver: ACM Press, 1998. 303 - 314.

[13] LU XC, ZHANG MJ, ZHU PD. API: A Adaptive PI Active Queue Management Algorithm[J]. Journal of software, 2005, 16(5): 903 - 910.

[14] HOLLOT CV, MISRA V, TOWSLEY D, *et al.* Analysis and design of controllers for AQM routers supporting Tcp flows[J]. IEEE Transactions on Automatic Control, 2002, 47(6): 945 - 959.

[15] CAI XL, WANG XF, WANG ZQ, *et al.* Analysis and Improvement of PI control for Active queue Management[J]. Journal of Nanjing University of Science and Technology, 2005. 29(3): 368 - 371.

[16] VINT project U C Berkeley/LBNL, ns2[J/OL]. http://www.isi.edu/nsman/ns/ns - allione, 2004 - 04 - 05.