基于 Gstreamer 的安全视频流传输系统的实现

宫 健,吴 蒙
(南京邮电大学,江苏 南京 210003)

摘 要:文中设计并实现了一种基于 PandaBoard ES 硬件平台和 Gstreamer 开发框架的安全视频流传输系统。在系统中使用了 V4L2 视频采集框架、H.264 压缩编码技术、RTP 流媒体传输协议实现了端到端的视频流实时传输,并且采用 AES 加密算法针对实时视频流进行加密,保证了视频数据的安全性。文中全面展示了实时视频系统的搭建过程,并且通过 Gstreamer 管道测试验证了该系统方案的可行性。测试结果表明,系统能够实时、准确、稳定、安全地将现场采集视频流信息显示到客户端,满足了远程视频监控系统的低时延、高安全的需求。

关键词:实时视频传输;Gstreamer;H264 压缩编码;RTP;AES 加密
中图分类号:TP302 **文献标识码**:A **文章编号**:1673-629X(2016)04-0177-05
doi:10.3969/j.issn.1673-629X.2016.04.039

Design of Secure Video Stream Transmission System Based on Gstreamer

GONG Jian, WU Meng
(Nanjing University of Posts and Telecommunications, Nanjing 210003, China)

Abstract: In this paper, a security video stream transmission system based on PandaBoard ES and Gstreamer platform is designed. In the system, V4L2 framework is used for collecting video source and H.264 compression coding technology is used for encoding the video stream. In terms of transmission, RTP is used to ensure good real-time performance. Last but not least, the AES encryption algorithm is also adopted to guarantee the security of the video data. The implementation of real-time video transmission system is comprehensively shown in this paper and through the analysis of the data acquired in the experiment. It is found that clients can receive the live video stream collected by the server timely, accurately and safely, which means that the secure video stream transmission system described in this paper is real-time, accurate, stable and secure.

Key words: real-time video transmission; Gstreamer; H.264 compression coding; RTP; AES encryption

0 引 言

近年来,随着互联网的广泛普及和多媒体技术的飞速发展,人们对基于网络的实时视频服务需求不断增长。典型的网络实时视频应用包括交互式视频服务、视频电话、远程教学以及远程医疗等。而嵌入式媒体平台由于体积小巧、功耗低、成本低廉的优点,成为了远程视频传输系统的良好载体[1]。

H.264 作为继 MPEG4 之后的新一代数字视频编解码技术,对实时视频流传输有很好的支持,在能够还原出高质量图像的基础上,不但具备可观的压缩比,较低的工作时延,还提供了纠错检错机制[2]。

实时传输协议(Real-time Transport Protocol, RTP)是一种网络传输协议,负责对流媒体数据进行封装,提供了时间信息和同步信息,实现了流媒体数据端对端的实时传送。但 RTP 并不保证服务质量,服务质量交由与其配套的实时传输控制协议(Real-time Transport Control Protocol, RTCP)来控制[3]。RTP 和 RTCP 均采用 UDP 协议发送数据包。

AES 是美国国家标准技术研究所(NIST)取代 DES 的新一代加密标准[4],是一种对称分组密码体制,使用 Rijindael 作为核心算法,具有强安全性、高性能、高效率、易用灵活等优点。

文中使用了 TI 公司的嵌入式平台 PandaBoard ES 作为硬件平台,结合了 H.264 编码技术和 RTP 协议,实现了视频流端到端的实时传输。并以此为基础,对传输的视频流进行了 AES 加密,保证了视频流数据的

收稿日期:2015-06-23 **修回日期**:2015-09-25 **网络出版时间**:2016-02-18
基金项目:国家"973"重点基础研究发展计划项目(2011CB302900);江苏省高校自然科学研究重点项目(10KJA510035);南京市科技发展计划重大项目(201103003)
作者简介:宫 健(1990-),男,硕士研究生,研究方向为无线通信与信号处理技术;吴 蒙,教授,研究方向为无线通信与信号处理技术。
网络出版地址:http://www.cnki.net/kcms/detail/61.1450.TP.20160218.1636.066.html

安全性。

1 系统硬件设计

文中所述系统搭建在 PandaBoard ES 系列开发板上。PandaBoard ES 采用 OMAP4460[5] 作为核心处理器。OMAP4460 使用了双核 Cortex-A9 对称架构,其主频可达 1.2 GHz,并且内置了图形加速器 PowerVR SGX540,是一款功能强悍的高性能处理器,可提供对称多处理(SMP)性能以及丰富的多媒体与 3D 图形支持。PandaBoard ES 支持包括 HDMI v1.3、DVI-D 在内的多种媒体接口,搭载了 IVA3 多媒体加速器,支持全高清、1080p 30 fps 多标准高清视频的录制与播放[6]。

文中所使用的 e-CAM51_44x 摄像头是专为搭载 OMAP4 系列处理器 PandaBoard 开发板设计的相机子板。其搭载了具有 500 万像素的自动聚焦图像传感器 OV5640,可以使用 V4L2 直接驱动,同时 PandaBoard 提供的 100 M 以太网口也使得远端传输视频成为了可能。

2 系统软件设计

2.1 视频的采集与编码

(1)V4L2 实现视频采集。

文中系统使用 V4L2 框架来实现视频源的采集。V4L2(Video For Linux Two)是 Linux 系统提供给应用程序访问音、视频驱动的统一接口,在嵌入式多媒体终端中有着广泛的应用[7]。

系统采集的视频帧格式为 YUV,大小为 1 080 * 720,因此在设置视频采集参数时需指定帧格式为 V4L2_PIX_FMT_YUYV,帧高 1 080,帧宽 720。此外,视频采集流的帧率 V4L2_CAP_TIMEPERFRAME 设置成 30 fps。

(2)使用 H.264 进行视频编码。

完成视频采集后,需要将采集的码流进行压缩编码以适应网络传输。文中选用 H.264 编码,在获得较高视频压缩比的同时,也能够保证实时视频流的传输质量。

H.264 协议中定义了三种帧:I 帧、P 帧和 B 帧[2]。I 帧为帧内编码帧,其编码不依赖于已经编码的图像数据。P 帧为前向预测帧,记录的是该帧与前一个 I 帧(或 P 帧)的差别。B 帧为双向预测帧,记录的是该帧和前后帧的差别。I 帧与 B 帧编码时都需要根据已编码的帧(即参考帧)进行运动估计。

在进行 H.264 压缩编码时,存在两种核心算法:一种是帧内压缩编码(也称为空间压缩),用于生成 I 帧。编码器在实现时会首先选择相应的帧内预测模式进行帧内预测,随后对实际值和预测值之间的差值进行变换、量化和熵编码,同时将编码后的码流经过反量化和反变换之后重建预测残差图像;然后再与预测值相加以得出重构帧;最后将得出的结果经过去块滤波器平滑后送入帧缓存。另一种是帧间压缩编码,用于生成 B 帧和 P 帧。编码器在实现时,会将输入的图像块首先在参考帧中进行运动估计,以得到运动矢量。运动估计后的残差图像经整数变换、量化和熵编码后与运动矢量一起送入信道传输。同时将另一路码流以相同的方式重建后经去块滤波再送入帧缓存作为下一帧编码的参考图像。当压缩后的码流进入解码器时,解码器会先判断。该帧若是帧内编码所得,则直接进行反量化、反变换加以重建;若是帧间编码所得,需要根据帧缓存中的参考图像进行运动补偿再叠加补偿前的残缺图像,从而还原出真实图像帧。

由于系统的实时性要求很高,文中使用了支持并行处理的 X264 编码器对视频流进行 H.264 编码,并且为了使系统获得更高的性能,订制了 X264 编码框架所使用的参数。将 X264 的 profile 参数设定为 baseline(基本档次)以适应视频实时通信应用。设置比特率参数 bitrate 为 300,设置 tune 为 zerolatency。编码器会自动工作在低时延状态,禁用一些会增加时延的功能如 sliced-threads(开启基于分片的线程)、sync-lookahead(启用线程预测的帧缓存)等。

2.2 实时视频流网络传输模块设计

文中系统使用 RTP 协议来完成端到端的视频传输,并且配合使用 RTCP 协议来完成网络流量控制和拥塞控制,保证网络传输的质量[8]。

根据上文所述,当服务器端视频流完成编码后,H.264 编码器会将完成的 VCL(视频编码层)数据封装到 NAL(网络抽象层)单元中以适应网络传输。RTP 协议会对这些 NAL 单元进行再次打包封装,写入时间戳和序列号等信息用以在接收端恢复准确的时间顺序。之后放入到 RTP 发送缓冲区交由 UDP 处理,封装成 UDP 数据包发送至网络。客户端接收到数据包时先进行 UDP 解包,得到 RTP 数据包,根据 RTP 包头的同步信息进行重排序,再对 RTP 包的负载进行解析。与此同时,使用 RTCP 协议分析当前网络状态。在服务器端,RTCP 会周期性的发送 SR(发送方报告)包,接收 RR(接收方报告包),客户端则周期性接收 SR 包,反馈 RR 包。服务器端根据发送的 SR 包和接收的 RR 包分析出丢包率,从而动态调节 RTP 的发包速率(可以通过调节编码的速率来实现),以适应当前网络状况。整体工作流程如图 1 所示。

2.3 基于 Gstreamer 的视频传输系统实现

文中使用了 Gstreamer[9-10] 流媒体开发框架设计实

现了实时视频传输系统。Gstreamer 是一个基于插件和管道结构的媒体开发框架。Gstreamer 流媒体框架通过基本元素构件 Element 相连接组成管道。Element 元件根据功能又可以分为 Src(数据源)元件、Filter(过滤器)元件、Sink(接收器)元件。而元件内的数据流传输又以 Pad(衬垫)的形式实现,可分为 Src Pad(输入衬垫)和 Sink Pad(输出衬垫)。

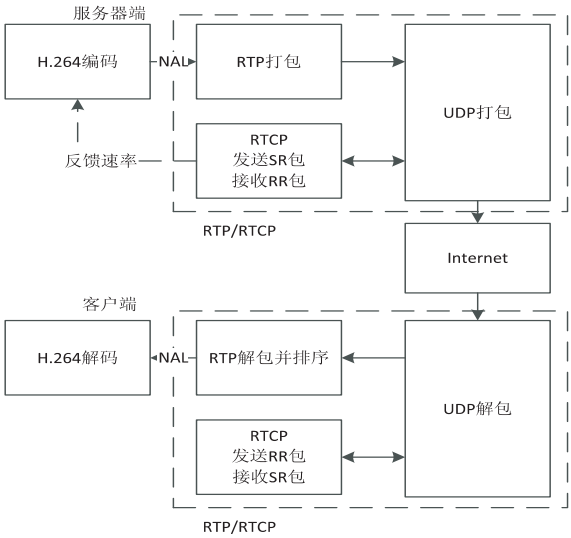

图 1　视频流网络传输模块流程图

当 Element 元件变得越来越多,数据管道越来越复杂时,Gstreamer 允许将一组相互连接的元件组合成一个大的逻辑元件,称之为 Bin(箱柜),从而以对 Bin 进行操作取代对单个 Element 元件的操作。

文中系统正是在 Gstreamer 插件和管道的基础上,由视频流管道构成。

(1)服务器端管道。

服务器端的视频数据流开始于 V4L2 从摄像头采集到的视频数据。使用数据队列插件 queue 对采集到的视频数据流进行缓冲,经缓冲的数据再通过 videorate 插件调整帧率以得到稳定的、适合播放的视频输入源。之后调用 X264enc 编码器插件对视频输入流进行 H264 编码。完成编码后的码流开始进入 RTP 处理流程。首先经过 rtph264pay 插件进行 RTP 打包,然后通过 rtpbin 箱柜。rtpbin 箱柜是一个将 rtpsession、rtpssrcdemux、rtpjitterbuffer、rtpptdemux 组件整合到一个组件的箱柜,用以控制 rtp 会话,并将数据包以 RTP 协议封装(或解封装),然后交由 udpsink 插件以 UDP 包的形式发送至客户端端口。同时,规定另外两个端口用于发送和接收 RTCP 包,rtpbin 会根据反馈的 RTCP 包,进行流量控制和拥塞控制,如图 2 所示。

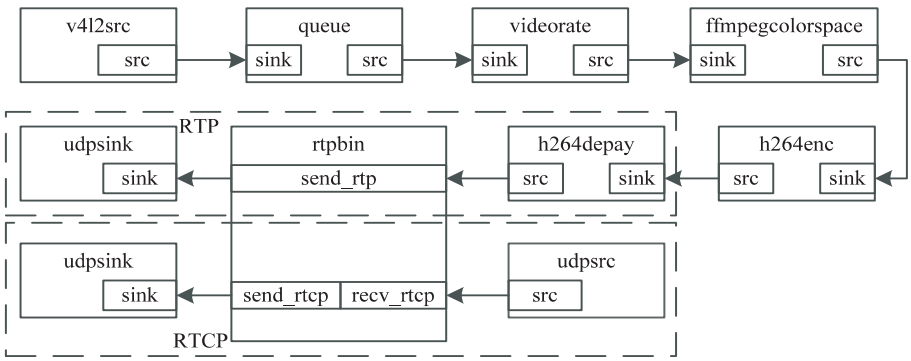

图 2　服务器端管道构成

(2)客户端管道

客户端用于监听数据的端口收到了服务器端的数据流,经由 udpsrc 插件接收到了 UDP 数据包,交由 rtpbin 处理。同时,规定另外两个端口用于发送和接收 RTCP 包配合 rtpbin 进行流量控制和拥塞控制。rtpbin 中的 rtpjitterbuffer 组件能够缓冲网络抖动并且将数据包重新排序,将顺序正确的 RTP 包交由 rtph264depay 插件以将 RTP 包解包为 NAL 单元负载流。

使用高性能解码器 ffmpeg 对 NAL 单元负载流进行 H.264 解码,所得码流将色彩空间转换成 RGB 以适应 ximagesink 播放插件的媒体能力。最终,可以通过 ximagesink 播放器显示出实时传输的视频流,如图 3 所示。

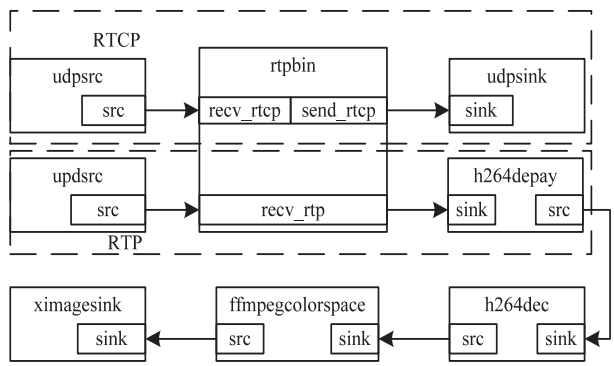

图 3　客户端管道构成

2.4　传输加密的设计

(1)AES 加密算法的原理。

文中系统使用分组长度和密钥长度为 128 bit 的 AES 加密算法。信息被分成 16 组,按顺序排列成 4×4 的状态矩阵,该状态矩阵就是 AES 算法中的数据最小单元。密钥也同样被表示为 4×4 的矩阵。AES 算法对数据的加密实则是将输入明文与密钥由轮函数经

N_r +1 轮迭代来实现的。N_r 由分组长度和密钥长度共同决定,当分组长度和密钥长度均为 128 时, N_r 取 10。在进行轮变换时,初始轮仅对明文和密钥进行异或(或称为轮密钥加)。在之后的 N_r -1 轮变换中,需要一次进行字节置换,行变换,列变换以及轮密钥加。在最后一轮变换中,省去了列变换,其余变换依次执行。每一轮所使用的轮密钥由密钥扩展函数所得[11]。整个加密过程如图 4 所示。解密过程则是整个加密过程的逆过程。

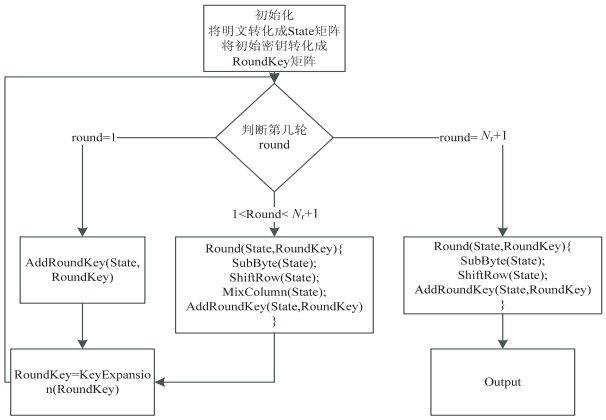

图 4 AES 算法流程图

(2)使用 AES 加密算法实现视频流加密。

为了使系统在获得较强安全性的同时兼顾流媒体系统低时延的特性,决定对视频流的加密处理置于视频编码完成后,进行 RTP 打包前。此时编码后的视频流输出为 NAL 单元,而 NAL 单元头部信息相对固定,且不包含视频信息,并不存在加密的价值,因此可以仅对 NAL 负载单元进行完全加密[12]。

由于系统由 Gstreamer 流媒体框架搭建,因此需要以 Gstreamer 插件的形式来实现视频流加解密。创建名为 AesCip、AesDecip 的 Gstreamer 插件[13]。AesCip 插件的逻辑如图 5 所示。

从输入流得到一个 NAL 单元后,忽略 NAL 单元头(即 NAL 单元的第一个字节),对第二个字节开始的 NAL 负载数据进行数据分组。以 128 bit 为一个分组,若最后一个分组不足 128 bit,则使用一些无意义的附加数据(例如均取 1)将其填满[14]。最后调用 AES 加密函数处理每个数据分组,得到一个加密后的 NAL 单元。由于加密时将负载数据填充到了 128 bit 的倍数,因此解密时需要识别出最后一个分组的附加数据并且将其删除,以免对解码视频数据造成影响。调用 AES 解密函数处理每个分组,完成 NAL 负载解密[15]。在使用加解密插件时,需要在插件前后添加 queue 插件作为数据缓冲,以协调编码、加密、RTP 打包的处理速率。在文中的实验环境中,测得视频数据传输速率小于 1 M/s 而使用 1.2 GHz 的 OMAP4460 处理器进行 AES 分组加解密时,加解密速度均在 2 M/s 以上,因此不会造成传输的阻塞,也不会产生很大的时延。

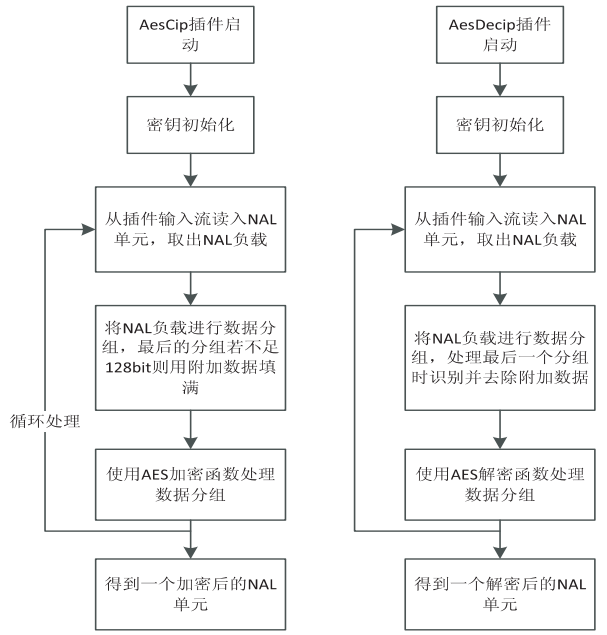

图 5 加解密插件程序流程图

3 系统测试分析

文中使用 Gstreamer 的管道调试工具 gst-launch 进行系统测试。测试时,以 PandaBoard ES 开发板作为服务器,以 PC 端虚拟机为客户端,服务器与客户端处于同一局域网下,IP 分别为 10.10.101.80 和 10.10.101.169,服务端和客户端均使用端口 5000 来传递视频数据,使用端口 5001 和 5002 来收发 RTCP 包进行网络流量控制和拥塞控制。

根据上文所述系统编写服务器端和客户端的 shell 脚本来调用调试工具。使用"!"管道符号连接起来,形成系统管道以供测试。测试时,先在客户端运行 client.sh,再在服务器端运行 server.sh。服务器端和客户端的 shell 脚本如下:

```
server.sh:
DEST=10.10.101.169
VOFFSET=0
VELEM="v4l2src device=/dev/video0"
#VELEM="videotestsrc is-live=1"
VCAPS="video/x-raw-yuv,width=1280,height=720,framerate=30/1"
VSOURCE=" $VELEM ! queue ! videorate ! ffmpegcolorspace ! $VCAPS"
VENC="x264enc tune=zerolatency byte-stream=true bitrate=300 ! rtph264pay"
VCIP=" queue ! aescip key-size=128 key=000102030405060708090a0b0c0d0e0f ! queue"
VRTPSINK="udpsink port=5000 host=$DEST ts-offset=$VOFFSET name=vrtpsink"
```

VRTCPSINK = " udpsink port = 5001 host = $ DEST sync = false async = false name = vrtcpsink"

VRTCPSRC = " udpsrc port = 5002 name = vrtcpsrc"

gst-launch -v gstrtpbin name = rtpbin \

$ VSOURCE ! $ VENC ! $ VCIP ! rtpbin. send_rtp_sink_0\

rtpbin. send_rtp_src_0 ! $ VRTPSINK \

rtpbin. send_rtcp_src_0 ! $ VRTCPSINK \

$ VRTCPSRC ! rtpbin. recv_rtcp_sink_0

client. sh:

DEST = 10. 10. 101. 80

LATENCY = 200

VCAPS = " application/x-rtp, media = (string) video, clock-rate = (int) 90000, encoding-name = (string) H264"

VDEC = " rtph264depay ! ffdec_h264"

VDECIP = " queue ! aesdecip key - size = 128 key = 000102030405060708090a0b0c0d0e0f ! queue"

VSINK = " ffmpegcolorspace ! autovideosink"

gst-launch -v gstrtpbin name = rtpbin latency = $ LATENCY \

udpsrc caps = $ VCAPS port = 5000 ! rtpbin. recv_rtp_sink_0 \

rtpbin. ! $ VDEC ! $ VDECIP ! $ VSINK \

udpsrc port = 5001 ! rtpbin. recv_rtcp_sink_0 \

rtpbin. send_rtcp_src_0 ! udpsink port = 5002 host = $ DEST sync = false async = false

测试结果如图 6 所示。

图 6　客户端接收到的视频效果图

测试结果表明,系统能够稳定、实时、安全地进行视频流传输,并且将传输时延控制在一定范围内,符合系统的设计初衷。完成测试后,将上述管道封装到 client. c,server. c 程序中,以供产品模式使用。

4　结束语

文中提出了一种基于 Gstreamer 流媒体框架,利用 V4L2 视频驱动、H. 264 压缩编码技术、RTP 流媒体传输协议实现的实时视频流传输系统。并且使用 AES 加密算法对传输视频流进行加密,在要求低时延的情况下完成了视频流的安全传输。该系统可应用于视频监控、视频会议、在线教育等,具有广阔的应用前景。

参考文献:

[1] 杨　泽,裴海龙. 基于 ARM 与 DSP 的实时视频传输系统[J]. 计算机工程与设计,2013,34(12):4184-4188.

[2] 毕厚杰. 新一代视频压缩编码标准—H. 264/AVC[M]. 北京:人民邮电出版社,2005.

[3] IETFRTP:a transport protocol for real-time applications[S/OL]. 2003. http://www. ietf. org/rfc/rfc3550. txt.

[4] NIST. Advanced Encryption Standard(AES)[M]. [s. l.]: Federal Information Processing Standards Publication,2001.

[5] TI. OMAP4460 multimedia device data manual[M/OL]. [s. l.]: TI. 2012. http://www. ti. com/lit/ds/symlink/omap4460. pdf.

[6] PandaBoard. OMAP4460 Pandaboard ES system reference ma-nual[M/OL]. [s. l.]:PandaBoard. 2011-09-29. http://pandaboard. org/sites/default/files/board_reference/ES/Panda_Board_Spec_DOC-21054_REV0_1. pdf.

[7] Schimek M H, Dirks B, Verkuil H. Video for Linux two API specification[S/OL]. 2006-02-03. http://www. linuxtv. org/downloads/legacy/video4linux/API/V4L2_API/spec-single/v4l2. html.

[8] 王彦丽,陈　明,陈　华,等. 基于 RTP/RTCP 的数字视频监控系统的设计与实现[J]. 计算机工程与科学,2009,31(3):58-60.

[9] The Gstreamer Team. Gstreamer application development manual[M/OL]. [s. l.]: The Gstreamer Team. 2010. http://gstreamer. freedesktop. org/data/doc/gstreamer/head/manual/manual. pdf.

[10] The Gstreamer Team. Gstreamer Plug-ins[M/OL]. [s. l.]: The Gstreamer Team. 2010. http://gstreamer. freedesktop. org/documentation/plugins. html.

[11] Daemen J, Rijmen V. 高级加密标准(AES)算法—Rijindael 的设计[M]. 谷大武,徐胜波,译. 北京:清华大学出版社,2003.

[12] 姜　浩. 基于 H. 264 的实时视频传输流密码加密研究[D]. 南京:南京林业大学,2012.

[13] The Gstreamer Team. Gstreamer plugin writer's guide[M/OL]. [s. l.]:The Gstreamer Team. 2010. http://gstreamer. freedesktop. org/data/doc/gstreamer/head/pwg/pwg. pdf.

[14] 陈道敏,周金泉. 加密技术在流媒体安全传输中的应用[J]. 网络安全技术与应用,2004(11):53-55.

[15] 蒋建国,李　援,梁立伟. H. 264 视频加密算法的研究及改进[J]. 电子学报,2007,35(9):1724-1727.