CITI Technical Report 99−1

Secure Distributed Virtual Conferencing: Multicast or Bust

W.A. Adamson
C.J. Antonelli
K.W. Coffman
P. McDaniel
J. Rees

ABSTRACT

We describe a secure distributed virtual conferencing application (SDVC) that provides high quality streaming video and audio using IP multicast for efficient distribution, using strong authentication via cryptographic means and optionally providing fully encrypted communication without sacrificing quality of the medium or the user experience. We summarize our experiences with SDVC in a recent live demonstration and conclude with a discussion of future plans.

January 25, 1999

Center for Information Technology Integration
University of Michigan
519 West William Street
Ann Arbor, MI 48103-4943

Secure Distributed Virtual Conferencing: Multicast or Bust

W.A. Adamson
C.J. Antonelli
K.W. Coffman
P. McDaniel
J. Rees

INTRODUCTION

The Secure Distributed Virtual Conferencing (SDVC) project, developed at the Center for Information Technology Integration in partnership with the University Corporation for Advanced Internet Development (UCAID), is a vehicle to advance UCAID processes while highlighting Internet2 capabilities and features. A first step toward fully virtual meetings, SDVC broadens access and participation to Internet2 events and activities.

SDVC also addresses UCAID's objective of establishing a middleware infrastructure that reduces the barriers to building and deploying portable, interoperable, scalable, and secure applications. These infrastructures need to be universally available, not just to select applications.

The initial goal demonstrated here is to integrate technologies that provide high quality streaming video and audio with strong authentication and encryption, without sacrificing quality of the medium or the user experience.

Our model assumes a single source for video and audio and multiple receivers that form a multicast group. SDVC assumes that all members of the group can send and receive packets over the multicast address.

Project Goals

Goals of the SDVC project include:

- **Single source.** SDVC and its predecessor, `vic`, assume a single video source and multiple participants. While `vic` supports H.261 traffic between all participants it does not meet our goal of providing secure encrypted group communication for a single video source.
- **Multicast.** Single-server multiple-client distributed applications do not scale well as the number of clients increases. Conventional wisdom dictates that a virtual conferencing application based on multicast can alleviate server performance bottlenecks.
- **Middleware.** Security, naming, and reliable group protocols naturally occupy the middleware landscape. Implemented there, these protocols free applications and platform operating systems from dealing with these issues in machine-specific ways.
- **MPEG-1.** We chose MPEG-1 over MPEG-2 because of its relative simplicity, availability of high-performance freely available software decoders, and the availability of relatively inexpensive hardware encoders.
- **Software solutions.** We chose to perform encryption, decryption, and MPEG-1 decoding in software to make our solution ubiquitous and independent of specialized hardware.
- **Open source.** Given the widely distributed nature of Internet2, we prefer to share and use freely available code wherever possible.

PROJECT OVERVIEW

The Secure Distributed Video Conferencing (SDVC) application is an extension of the Secure Video Conferencing (SVC) work [1] demonstrated at the Internet2 Member Meeting held October 1997. SVC, based on `vic` [2], the popular MBONE videoconferencing tool, provides authenticated, encrypted, full frame video delivered over a unicast channel.

SDVC extends this work by adding protocols for secure multiparty group communications, MPEG-1 encoding and decoding capabilities, and Globus [3] GSS API and SSLeay libraries for security context initialization. SDVC also adds all these capabilities to `vat` [4], the MBONE audio tool.

All participants use the same data encryption key for the video and audio streams. When a new participant joins the group, or when a heartbeat detects that a participant has left the group, SDVC generates and distributes a new data encryption key to the participants. This assures that only active members of the group can see and hear the video and audio streams.

For group establishment and maintenance, SDVC uses Lightweight Secure Group Communications (LSGC) [5]. LSGC employs three protocol layers; reliable broadcast, process group management, and security services. The reliable broadcast layer ensures ordered and atomic reception of group messages. The process group management layer provides all processes with a consistent view of group membership. The security services layer provides facilities for ensuring secrecy, integrity, and freshness of the group communication.

The architecture is shown in Figure 1, where the dotted line indicates Globus unicast communication, two-way LSGC multicast is shown by the dashed arrows, and one-way video multicast is shown by the solid arrows.

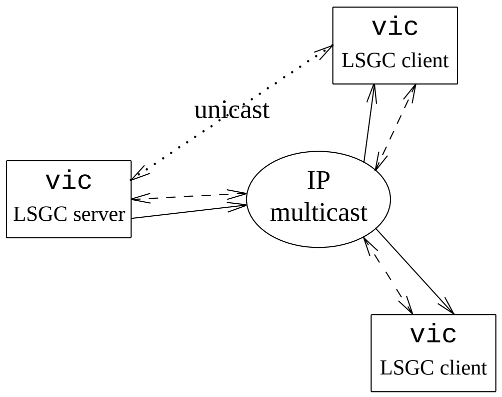

Figure 1: SDVC Architecture

There is a natural tension among security and scalability. Security comes at a cost, and that cost includes maintaining state among participants, which grows with higher levels of security and the number of participants in a group. Our approach starts with a high level of security. After evaluating scalability, we will relax security only where necessary to extend the limits of scalability, for example by retaining the same data encryption key throughout a session to avoid redistributing a new key each time a participant joins or leaves the group. This may be acceptable security depending on the circumstances. Our experience has shown that it is easier to relax existing security than to add security where there is too little.

CITI replaced the Leighton-Micali key distribution algorithm [6] in the original implementation of LSGC with Globus' GSS API SSLeay key exchange. We also enhanced LSGC's DES encryption with SVC's multiple choices for ciphers, retaining the ability to change ciphers on the fly.

SDVC thus integrates multicast video and audio with a scalable key exchange protocol and secure multiparty group communication to provide an authenticated, encrypted data stream.

SDVC COMPONENTS

In this section we describe our enhancements and additions to SVC.

MPEG encoding and decoding

As distributed, `vic` does not support MPEG. We added support for Sun hardware MPEG-1 encoding and multiple platform software MPEG-1 decoding.

MPEG is designed to be computationally asymmetric, with the encoding process requiring about 100 times the computing power of the decoding process. Accordingly, SDVC encodes the video stream in hardware using a SunVideo board [7] and decodes the MPEG video stream in software. Software decoding of the MPEG stream is CPU bound, which is acceptable in this world of rapid increases in CPU speed.

For encoding, SDVC extends the existing `vic` Sun XIL grabber interface to access the MPEG-1 stream generated by the hardware board. To facilitate frame reconstruction by the decoders in the face of dropped packets, SDVC configures the SunVideo board to produce an MPEG-1 stream consisting entirely of ‘‘I’’ frames.

SDVC integrates the publically available Berkeley MPEG decoder `mpeg_play` [8] into `vic` as the software decoding component for UNIX platforms.

`mpeg_play` is an X Window System application designed to play an MPEG-1 stream read from a UNIX file. A 400 KB buffer is filled from the file system and passed to the MPEG decoder in 80 KB chunks. While this read-ahead buffering gives the user a smooth view of an MPEG stream played from a local file, real time decoding of an MPEG stream from a network interface requires a different buffering scheme. SDVC reconstructs one frame of encoded MPEG-1 data into a buffer, then tries to pass the single frame to the MPEG decoder while reconstructing the next encoded MPEG frame into a second buffer. If the decoder is busy and cannot accept the first buffered frame, the frame is dropped and the buffer is used to reconstruct the next frame arriving on the network interface. This strategy allows `mpeg_play` to accommodate variances in network and CPU speeds while providing a real time view of the MPEG stream that is as smooth as possible.

SDVC uses the Win32 DirectShow interface for hardware and software decoding of the the MPEG-1 video stream on Win95, Win98, and NT. Microsoft's DirectShow architecture is integrated with their DirectX technologies to take advantage of any hardware acceleration available on the Windows platform automatically; otherwise, available software components for audio or video playback are used. DirectShow uses a filter graph architecture where individual filters are connected together into a ‘‘graph’’ to process data.

Filters are categorized into source filters, transform filters, and rendering filters. For SDVC, we created a network source filter that is provided the MPEG-1 video data from `vic` as it is received from the network. The source filter sends the data downstream to the transform and rendering filters for display. Care is taken to reduce copying of data by using buffers supplied by downstream filters whenever available. If a DirectShow compatible hardware MPEG-1 decoder is available, the DirectShow graph editor should automatically include it as the rendering filter. However, we found that the software MPEG-1 decoder filter, and not the Netstream hardware decoding filter, was selected when rendering the output of our source filter. To take advantage of the Netstream hardware MPEG-1 decoder, SDVC therefore specifically includes and connects the hardware decoding filter when building the filter graph. Inquiries to Sigma Designs support resulted in a theory that this was due to the non-standard size (320×240) of the input stream (but we have not yet been able to verify this theory).

The SunVideo board produces an MPEG-1 encoded video stream at a resolution of 320×240 pixels. Video bandwidth depends on the quality setting of the capture device as well as the visual complexity of the video stream. At 30 frames per second, the MPEG-1 stream generated by the SunVideo board varies from approximately 1.5 Mbps at a low quality setting to approximately 7 Mbps for the highest quality setting.

Software MPEG-1 decoding is CPU bound. On our isolated test network consisting of three 10 Mbps switched nets connected to a Cisco 4000 router, we observe mixed results, shown in Table 1. The source is sending 30 frames per second at 3.5 Mbps. Reductions in frame rate and Mbps at the receiver are due to the software MPEG-1 decoder dropping frames as the CPU is unable to keep up with the input rate while decoding the input video stream.

Cryptographic Functionality

SDVC inherits cryptographic functionality and the ability to switch ciphers on the fly from its predecessor SVC, which adds the ciphers RC4 [9] and VRA [1], to the XOR and DES ciphers shipped with `vic`. RC4 is a simple stream cipher reputed to be fast and secure. VRA is Bellcore's ‘‘provably secure’’ high-performance stream cipher.

In SVC, VRA convincingly outperforms RC4. VRA has another advantage over RC4: it can decrypt out-of-order packets, which RC4 can not. If a packet is lost, the receiver rolls its RC4 state forward and continues to decrypt data following the lost packet. But there is no way to roll back the RC4 state, so packets received out of order cannot be decrypted. On the other hand, the inner loop of VRA, which selects rows in a wide binary table to XOR with the stream data, requires only the packet sequence number to pick the rows. For these reasons, we dropped RC4 support in SDVC.

Security architecture

As in SVC, SDVC uses the GSS API to allow different security mechanisms to be interchanged. SDVC replaces LSGC's Leighton-Micali authentication mechanism with the Globus GSS API but preserves the existing participant key structure. The participant key is a shared secret between a participant and the LSGC group server. The participant key is used to encrypt the common group data key. In this way, the data key is delivered only to participants that have authenticated to the LSGC group server.

Unicast peers using the GSS API create and hold a GSS security context for the life of a secure connection. In a unicast environment, the

Platform, OS	fps	Effective Stream	CPU Usage
200Mhz Sun Ultra-5, Solaris 2.6	30	3.5 Mbps	84%
50Mhz Sun Sparc-20, Solaris 2.5	10	1.0 Mbps	85%
122Mhz IBM 42T, AIX 4.1	19	1.9 Mbps	74%
400Mhz Pentium, OpenBsd	24	3.0 Mbps	80%

Table 1 This table shows measured performance for platforms receiving a 3.5 Mbps, 30 fps MPEG-1 input stream. Only one of the receivers is capable of displaying the full stream; the other receivers are CPU-bound and can not decode the input stream at the offered rate.

security context is used both for authentication and for data encryption; obtaining the security context implies that authentication has occurred, and the security context stores the shared secret used for data encryption. Secure group communication requires additional security contexts because group peers need not only to authenticate individually to join the group using the existing GSS security context, but also need to share a common data encryption key. Extending the GSS API to include these group security issues is beyond the scope of the SDVC project, primarily because it would be necessary to maintain multiple GSS contexts on the server and multiplex between them in communicating with participants. This would raise many GSS API implementation specific issues.

Participant key distribution

We use Globus to distribute participant keys on demand. When a client participant joins the multicast group, the server must pass it securely a participant key before the client can begin decoding the multicast video stream. SVC uses a smart card based Shoup-Rubin protocol [10] for key distribution, but SDVC passes the key using SSL, using the Globus GSS API and SSLeay [11] implementations.

We distribute a Globus ‘‘gatekeeper’’ certificate and corresponding private key with the SDVC server and obtain conventional Globus user certificates and private keys for all clients. The private key for gatekeeper certificates is not protected via encryption and therefore must be stored securely;† our security model requires good physical security for the server. Certificate requests are generated by the Globus software and digitally signed by the Globus Certificate Authority. We distribute the requisite self-signed Globus CA certificate with the server and all clients.

To obtain a participant key in order to join the LSGC group, a client uses Globus to initiate a secure GSS connection by calling `gss_init_sec_context`. This requires the user to enter the pass phrase protecting the private key stored on the client. The server maintains a thread that listens for such requests and completes the Globus context-establishment loop with the client by calling `gss_accept_sec_context`. As part of the context-establishment protocol, the client determines that the common name stored in the server’s certificate is as expected, *i.e.,* the server authenticates itself to the client. The server then generates and stores a client participant key, and uses `gss_wrap` to encrypt and send it to the client, which uses `gss_unwrap` for retrieval. Server and client now share a key that can be used for subsequent secure communications and the GSS security context is mutually destroyed. Concurrent client key requests are processed serially at the server.

We built the participant key exchange code using the domestic version of the Globus security library and a pair of modified test programs supplied by Doug Engert [12].

Multiparty key distribution

The distribution and management of the group membership and data encryption keys is performed by the Lightweight Secure Group Communication library. The LSGC security layer provides interfaces for the negotiation of the group communication (data encryption) keys.

The data encryption key is specific to each group view.‡ As each membership change occurs, a new data encryption key is distributed and

† Storing the gatekeeper key encrypted is possible, but this requires entering a password at the server each time a new client connects. A potential smartcard solution is attractive.

‡ A group view is a snapshot of group membership during a period in which no membership changes occur.

installed. Re-keying ensures that only members in the current view have access to the conferencing content.

The server begins re-keying by generating a new data encryption key. This key is encrypted with the participant key of each existing member of the group, and the resulting encrypted blocks are concatenated to form the session key distribution message. The message is forwarded to the group using reliable broadcast. On receipt, each participant uses its cached participant key to extract the new session key and immediately begins using it.

EXPERIENCES AND FURTHER WORK

IP Multicast or Bust

There are numerous multicast routing protocols in use. Simple techniques include flooding and spanning tree methods. More common are source-based tree solutions, where a spanning tree is built and rooted at each potential multicast source. Both the Distance Vector Multicast Routing Protocol (DVMRP) [13] and the Protocol-Independent Multicast (PIM) [14] routing protocols use source based trees. The majority of the routers in the MBONE use some version of PIM or DVMRP.

The vBNS maintains a native IP multicast service via a PIM dense-mode configuration among all vBNS Cisco routers [15]. DVMRP unicast routing is used, allowing the vBNS to support delivery. Tunnels to PIM routers or mrouted hosts and PIM connections to routers over point-to-point VCs make up the multicast connections to the vBNS. MBGP is currently under test deployment on the vBNS. MBGP allows BGP unicast routes to be tagged as multicast; this allegedly addresses the scaling problem inherent in DVMRP’s single routing domain.

LSGC protocol negotiations use multicast between clients and server. Although LSGC provides reliable communications via retransmission, inability to multicast from a client prevents that client from joining the SDVC group. This means that, unlike most multicast applications in existence today, our application requires multicast in both directions.

Multicast routing protocols deployed on the vBNS follow unicast routes to locate hosts. This means that the deployment of bi-directional multicast routing requires careful attention to IP unicast configuration on the routers deploying multicast, taking care that the IP unicast routes point to routers configured to handle the multicast traffic. This is difficult in production networks due to the existence of default IP routes and IP route caching that may take precedence over statically assigned routes and mroutes.

The bi-directional multicast requirements of SDVC combined with the complex unicast dependencies of the multicast environment on the vBNS proves to be the biggest challenge in deploying a prototype SDVC.

We first tested SDVC in the CITI lab after turning on IGMP in our Cisco 4000 router. We tested a seven-way communication with one SDVC 30 fps MPEG-1 video source, an SDVC audio source, and six machines that received the encrypted video and audio streams. As a proof of concept, each receiver also sent H.261 encrypted video and audio over the multicast channel. All encryption used the group data key. This test behaved properly.

We then tested SDVC at the Merit vBNS point of presence [16] while Merit was configuring its multicast connection to the vBNS. We were able to send and receive multicast packets to and from the vBNS, but setting up a reliable bi-directional multicast connection was difficult, requiring coordination with intermediate vBNS router administrators to install static or default mroutes into the production routers to make sure the reverse path forwarding tree was correct. Bi-directional multicast then worked intermittently. Suspecting the interface between the entry routers to the vBNS and the MBGP functional vBNS router software as the culprit, Merit loaded Cisco routers with an IOS with MBGP functionality to provide the vBNS connection. We tested SDVC over the vBNS and were able to connect successfully with three to four institutions, but an equal number of institutions were unable to receive the SDVC transmission because the existing multicast configuration could not meet SDVC’s bi-directional multicast needs.

SDVC was demonstrated at the Internet2 Member meeting September 27–29, 1998 in San Francisco. We had results similar to the testing done at Merit. The reverse path forwarding tree was confirmed to be correct, yet bi-directional multicast still did not work. Disabling IP route caching enabled SDVC to function. The Internet2 Plenary sessions and Breakout sessions were broadcast over the vBNS and received by several institutions.

LSGC Issues

Rekeying Synchronization

In the current implementation, a client may receive packets encrypted with a new session key before receiving the session key distribution message. To address this limitation, a *safe delivery* extension to the reliable group broadcast protocol is required. Safe delivery has strong delivery semantics that ensure that all processes in the current group view have received a message before any process delivers it to the application. In this way, LSGC will be able to better synchronize the client transition to new session keys.

Server Restart Recovery

Currently, when a server is shut down and restarted, clients must be manually restarted before gaining access to the new group. No user-level indication of the server restart is provided. Notification and recovery from server restarts is necessary.

Security Protocol Improvements

The current implementation of the LSGC library does not provide protection against message reordering. An adversary may intercept, alter, and broadcast protocol messages to effect an arbitrary ordering of the messages in the group. One dire implication of this problem is that a client may be coerced into using an old session key. To address this problem, a protocol extension will be added to include message authentication codes covering the process group management and reliable broadcast fields of messages.

Further Work

GSS API

We will include two more GSS API interface choices to SDVC: an exportable GSS API implementation that performs no encryption and passes the participant key in the clear, and the Kerberos5 GSS API. These will be compile time options.

QoS

As part of the University of Michigan and Merit Internet2 Qbone Testbed proposal, SDVC will be enhanced with an RSVP [17] component, and will be instrumented to measure end-to-end performance of QoS attributes such as bandwidth, latency, packet loss and jitter.

CONCLUSIONS

SDVC integrates multicast video and audio with a scalable key exchange protocol and secure multiparty group communication to provide an authenticated, encrypted data stream to members of the multicast group.

Multicast delivery increases scalability at the server at the cost of increased complexity at the multicast routers. This is not a new result, yet our experiences with the bi-directional multicast employed by SDVC force us to conclude that multicast routing requires changing many routing configurations, a manual process that we know does not work well at present.

Our goal of using and extending freely available code has largely been met, save for the domestic Globus and Bellcore VRA security components. We are actively addressing the latter issue by working with Bellcore on VRA licensing and are looking toward the AES effort to deliver a stream cipher free of licensing restrictions and fast enough for our needs.

Our goal of software solutions on the client naturally led to CPU limitations there. These limitations can be removed with a liberal application of cash *e.g.,* hardware decoders or faster client CPUs, or by waiting for Moore’s law to deliver up the computers we need.

ACKNOWLEDGEMENTS

We thank Doug Engert and the Globus project staff for supplying us with the exportable Globus code and assisting us in its use.

We thank Bert Rossi and Merit staff for their assistance in multicast routing configuration from the University of Michigan to the vBNS.

We thank Bob Riddle and the UCAID staff along with Robert Poston and the IBM Internet2 demo team for their organizational help at the 1998 September Internet2 Member meeting.

We thank Jorge Lepre of the IBM T.J. Watson Research Center for his insight and assistance with the DirectShow programming for the Windows platform.

We thank Jerry Sobieski and the Internet2 Network team for their assistance in multicast routing

configuration from the San Francisco Hilton to the vBNS for the 1998 September Internet2 Member meeting.

This work was partially supported by the University Corporation for Advanced Internet Development.

REFERENCES

1. P. Honeyman, W.A. Adamson, K.W. Coffman, J.E. Janakiraman, R. Jerdonek, and J. Rees, ‘‘Secure Videoconferencing,’’ in *Proc. 7th USENIX Security Symp.*, San Antonio (January 1998).
2. Network Research Group, in `www-nrg.ee.lbl.gov/vic/`, Lawrence Berkeley National Laboratory.
3. Globus Project, in `www.globus.org/`.
4. Network Research Group, in `www-nrg.ee.lbl.gov/vat/`, Lawrence Berkeley National Laboratory.
5. P. McDaniel, P. Honeyman, and A. Prakash, ‘‘Lightweight Secure Group Communication,’’ CITI Technical Report 98–2, Center for Information Technology Integration, University of Michigan, Ann Arbor (April 1998).
6. T. Leighton and S. Micali , ‘‘Secret-key Agreement without Public-Key Cryptography,’’ in *Advances in Cryptology: Proc. of Crypto 93* (1994).
7. Sun Microsystems, in `www.sun.com/desktop/products/Multimedia/SunVideo.html`. Part number X1085A.
8. Berkeley Multimedia Research Center, in `www-plateau.cs.berkeley.edu/mpeg/index.html`.
9. B. Schneier, *Applied Cryptography, Second Ed.*, 1996.
10. V. Shoup and A.D. Rubin, ‘‘Session Key Distribution Using Smart Cards,’’ in *Proc. of Eurocrypt,96* (1996).
11. Eric A. Young, ‘‘`SSLeay`,’’ in `ftp://ftp.psy.uq.oz.au/pub/Crypto/SSL/`.
12. Doug Engert, Argonne National Laboratory. Private communication.
13. S.E. Deering, ‘‘Multicast Routing in a Datagram Internetwork,’’ in *Ph.D. Thesis, Stanford University* (1991).
14. S. Deering , D. Estrin , D. Farinacci , M. Handley , A. Helmy , V. Jacobson , C. Liu , P. Sharma , D. Thaler , and L. Wei , ‘‘Protocol Independent Multicst-Sparse Mode (PIM-SM): Protocol Specification,’’ in *IETF RFC 2362* (1998).
15. vBNS, in `www.vbns.net/multicast/overview.html`.
16. Merit, Inc., in `www.merit.net/`, Ann Arbor.
17. R. Braden and L. Zhang, ‘‘Resource ReSerVation Protocol(RSVP) -- Version 1 Message Processing Rules,’’ in *IETF RFC 2209* (1997).