SECURE DISTRIBUTED VIRTUAL CONFERENCING

W.A. Adamson, C.J. Antonelli, K.W. Coffman, P. McDaniel, J. Rees
Center for Information Technology Integration
The University of Michigan
Ann Arbor, MI

Abstract We describe a secure distributed virtual conferencing application (SDVC) that provides high quality streaming video and audio using IP multicast for efficient distribution, uses strong authentication via cryptographic means, and (optionally) provides fully encrypted communication without sacrificing the 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.

1. INTRODUCTION

The Secure Distributed Virtual Conferencing (SDVC) project, a first step towards the realization of fully virtual meetings, establishes a middleware infrastructure that reduces the barriers to building and deploying portable, interoperable, scalable, and secure applications. Such an infrastructure needs to be universally available, not just to selected applications.

The initial goal demonstrated here is to integrate technologies that provide high quality streaming video and audio with strong authentication[1] and encryption, without sacrificing quality of the medium or the user experience, over modern multicast-capable networks such as the vBNS [vBN].

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:

- **Security.** Strong authentication, data privacy for high quality video and audio multicast streams, and multicast distribution of data encryption keys are the main security goals of the project.

- **Single source.** Unlike its ancestors `vic` and `vat`, SDVC is not designed to provide video and audio streams between all participants. It provides secure encrypted group communication for a single high quality MPEG-1 video source and audio source to a large number of participants. The single source of MPEG-1 video is designated as the *session leader*, and is assumed to exist for the duration of an SDVC session. A small number of secure encrypted low quality H.261 video and audio back channels is also supported.

- **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.

- **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.

- **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 our middleware infrastructure, we prefer to share and use freely available code wherever possible.

2. PROJECT OVERVIEW

The Secure Distributed Virtual Conferencing (SDVC) application is an extension of the Secure Video Conferencing (SVC) work [HAC+98] demonstrated at a higher education consortium meeting held October

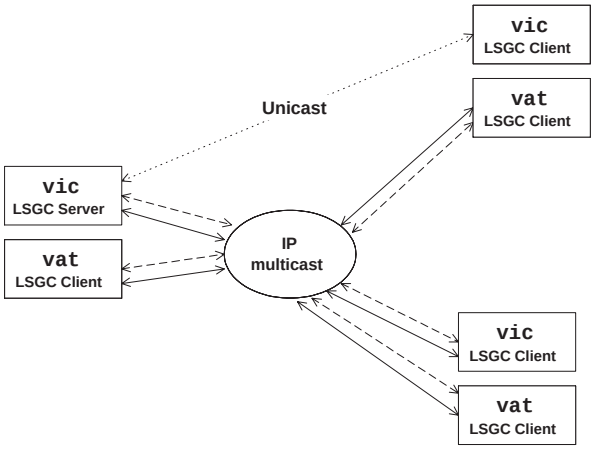

Figure 1 SDVC Architecture - Clients initiate the join process by authenticating with the session leader via unicast (dotted line). Thereafter, the group membership and session keys are negotiated among the group members over a bi-directional multicast channel (dashed line). The video and audio data is delivered over a uni-directional multicast channel (solid line).

1997. SVC, based on `vic` [Net96], 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 [glo] GSS API and SSLeay libraries for security context initialization. SDVC also adds all these capabilities to `vat` [Net96], 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) [MHP98]. 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 and audio multicast is shown by the solid arrows.

There is a natural tension between security and scalability. The initial version of SDVC described throughout has been designed to address security first and scalability second. The costs associated with the strong security guarantees provided by SDVC may limit its scalability; as this becomes a larger concern, we have the option to relax security where feasible to improve scalability.

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

3. SECURITY REQUIREMENTS

SDVC's security focus is on data privacy and strong user and entity authentication. This focus is driven by end-users who require the ability to deliver private video and audio content to a group of users. SDVC has the following standard security requirements.

- *Authentication* - The session leader is guaranteed that all group members are authenticated. Likewise, all participants authenticate the session leader. In SDVC, a potential group member and the session leader mutually authenticate before access to the group is allowed. Authentication is achieved through the use of public key certificates.

- *Authorization* - A user is allowed access only to session content to which he/she is entitled. Currently, any user that has a signed certificate can join any SDVC session. A participant may view session content only for data transmitted while she is a member of the group; a participant may not view data transmitted before she joins the group or after she leaves.

- *Data Integrity* - The content of a secured communication cannot be altered. SDVC provides data integrity in conjunction with data privacy.

- *Data Privacy* - The contents of a secured communication are not disclosed to unauthorized parties. SDVC provides a panoply of ciphers that achieve this.

In SDVC, the policy under which the group operates is determined solely by the session leader. The three dimensions along which SDVC policy is defined are *security*, *authentication*, and *privacy*.

- The security policy defines the use of the security infrastructure. If this policy indicates that no security infrastructure be used, then no security related features are enabled, and the session behaves much as a standard `vic` or `vat` session.
- The authentication policy defines the mode of operation used in the authentication process. The authentication mechanism provides modes supporting authentication only, or authentication with integrity and privacy.
- The privacy policy identifies the cipher used to encrypt the session traffic. The selection of a cipher is influenced by both performance and security issues. We identify several of these issues below (see *Cryptographic Functionality*).

4. SDVC COMPONENTS

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

Security Architecture

SDVC uses the GSS API [Lin97] to allow different security mechanisms to be interchanged. SDVC replaces LSGC's Leighton-Micali [LM94] 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 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 securely pass it a participant key before the client can begin decoding the multicast video stream. SVC [HAC+98] uses a smart card based protocol for key distribution, while SDVC passes the key via SSL, using the Globus GSS API and SSLeay [You] implementations.

Globus runs the Certificate Authority. We distribute a Globus "gatekeeper" entity certificate and corresponding private key with the SDVC server and users obtain conventional Globus user certificates and private keys. The private key for gatekeeper certificates is not protected via encryption and therefore must be stored securely;[2] our security model requires good physical security for the server. User certificate requests are generated by the Globus software and digitally signed by the Globus Certificate Authority upon proof of identity. We distribute the requisite self-signed Globus CA certificate with the server and all clients.

To obtain a participant key and join the LSGC group, a client uses Globus to initiate a 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 and the unicast connection is released. 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.

Group Management

The Lightweight Secure Group Communication (LSGC) library provides the SDVC with two essential services: group membership and data encryption key distribution. The design of LSGC uses simple, well-understood technologies to implement a secure group. However, as future needs dictate, we can extend this design to include more robust, inexpensive group and security management services.

The single source assumed by our initial version of SDVC allowed us to make several simplifying assumptions in the design of LSGC. Because

the group has a logical leader (the video transmitter, called the *session leader*), we may assume the presence of a singular entity throughout the session. Furthermore, we may assume that the session leader is the sole entity implementing group admission policy.

Note that within SDVC, LSGC is used only for the distribution of group membership and session keys. The overhead associated with reliable delivery prohibits the use of reliable delivery for session data.

LSGC consists of three protocol layers; the *reliable broadcast layer*, the *process group management layer*, and the *security services layer*. The reliable broadcast layer provides the group with a reliable, totally ordered data stream. LSGC uses a sequencer based solution to ensure reliable delivery. In this design, the session leader (sequencer) determines and announces the order of all group messages. Members who detect lost messages acquire them from the sequencer directly.

The process group management layer provides all users with consistent *views*[3] of the group membership. The session leader distributes a new group view after each change in membership. As a performance enhancement, the group view is distributed simultaneously with new data encryption keys.

The security services layer handles access control policy and the distribution of data encryption keys. The data encryption key is specific to a group view. With each membership change, the group is rekeyed through the distribution and installation of a new data encryption key. 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 the new key.

A well known problem in group key management is scalability. Mittra [Mit97] identifies the *1 affects n* problem, where a single group membership event (join, leave, eject), affects the entire group. A popular approach to addressing this limitation is to construct trees of subgroups, where each membership event affects only those entities in the local subgroup. However, this solution requires that messages between subgroups must be translated from the session key of one subgroup to another.

Recent developments in secure group management vastly reduce the costs of rekeying. The hierarchical tree approach [WHA98] defines a tree of keys where the root key is used to transmit session traffic. Each

member represents a leaf node in the hierarchy, and shares a secret with the server (session leader in LSGC). As members join or leave the group, the keys from the joining/departing user to the root (in the hierarchy) are modified. Thus, the total cost of rekeying scales logarithmically.

In LSGC, the session leader transmits a new session key after each membership event. We reduce the overall cost of session key distribution by distributing a single session key to the entire group. The size of this message increases linearly with group size, containing 32 bytes per member. Using a single distribution message simplifies the delivery sequencing and fault tolerance.

Future versions of SDVC will replace LSGC with the Antigone [MPH99] secure group communication middleware layer. Antigone provides a suite of *mechanisms* under which group management policy may be defined and implemented. These mechanisms are designed independently, and may be replaced as requirements dictate. Through the integration of newly defined mechanisms, we will address scalability, fault-tolerance, and policy issues. Furthermore, we hope to relax the requirement of a single sender and session leader.

Cryptographic Functionality

SDVC inherits cryptographic functionality and the ability to switch ciphers on the fly from its predecessor SVC, which added the ciphers RC4 [Sch96] and VRA [ARV95] to the XOR and DES ciphers shipped with `vic`. RC4 is a simple stream cipher reputed to be fast and secure. Described in Appendix A, VRA is a high performance stream cipher that exhibits good cryptographic properties and is based on provable mathematical properties.

Low latency is a requirement of many streaming video and audio applications, therefore we must minimize latency due to software encryption/decryption. As reported in [HAC+98], the relative throughput of the supported ciphers were measured on 166 Mhz Pentium systems running Windows 95 and using hardware MJPEG encoding and decoding. We found that DES was four times as slow as RC4 which was in turn twice as slow as VRA. As future work, these results will be expanded to include new ports of SDVC running on high-speed machines, and several different ciphers such as triple-DES, and the Advanced Encryption Standard (AES) candidate algorithms [oSN].

MPEG Encoding and Decoding

As distributed, `vic` does not support MPEG. We added support for Sun hardware MPEG-1 encoding, multiple platform software MPEG-1 decoding, and WIN32 hardware decoding.

Platform, OS	fps	Stream Rate	CPU %
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 RS/6000, AIX 4.1	19	1.9 Mbps	74%
400Mhz Pentium, OpenBsd 2.3	24	3.0 Mbps	80%
200Mhz Pentium, NT4.0 (hardware)	30	3.5Mbs	65%
200Mhz Pentium, NT4.0 (software)	10	1.8Mbs	100%
400Mhz Pentium II, NT4.0 (software)	30	3.5Mbs	40%

Table 1 This table shows measured performance for platforms receiving a 3.5 Mbps, 30 fps, VRA encrypted 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.

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 [Mic] 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 Berkeley MPEG decoder `mpeg_play` [Cen99] 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 Microsoft's DirectShow^{TM} interface for hardware and software decoding of the MPEG-1 video stream on Win95, Win98, and NT. The DirectShow architecture is integrated with the DirectX^{TM} 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 fed 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 causes us to suspect that this was due to the non-standard 320x240 size of the input stream, but we have not yet been able to verify our suspicions.

The SunVideo board produces an MPEG-1 encoded video stream at a resolution of 320x240 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.

5. EXPERIENCES AND FURTHER WORK

The LSGC protocol uses bi-directional multicast to communicate between the clients and session leader. The inability to multicast from a client to the LSGC session leader prevents that client from joining the SDVC group.

At the time of our testing, vBNS maintained a native IP multicast service using a PIM Dense-Mode (PIM-DM) [DEF+98] configuration among all vBNS Cisco routers [Tho99]. Multicast connections to the vBNS consisted of tunnels to PIM routers and PIM connections to routers over point-to-point virtual connections. MBGP [TC99], a draft multicast router protocol, was under test deployment on the vBNS.

PIM-DM relies on existing unicast routing tables to find routes back to the source. Thus, the deployment of multicast routing requires careful attention to unicast configuration, taking care that the unicast routes point to routers configured for multicast traffic. Identifying and configuring all the potential intermediate routers is technically and administratively problematic. Moreover, caching may take precedence over statically assigned routes, leading to unexpected multicast routing.

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

We first tested SDVC in our 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 our vBNS point of presence [Mer] while our POP 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, staff at our POP 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 those encountered during the testing done at our POP. 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. Consortium sessions were broadcast over the vBNS and received by several institutions.

6. 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 VRA security components. We are actively addressing the latter issue by working 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.

Notes

1. Throughout, we use the term authentication to refer to *entity authentication*. The Globus architecture [glo] is used to authenticate all SDVC session participants.

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

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

References

[ARV95] W. Aiello, S. Rajagopalan, and R. Venkatesan. "Design of Practical and Provably Good Random Number Generators".

In *Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1–9, January 1995.

[Cen99] Berkeley Multimedia Research Center. Berkeley MPEG Tools, May 1999. `http://bmrc.berkeley.edu/frame/research/mpeg/`.

[DEF+98] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, A. Helmy, D. Meyer, and L. Wei. "Protocol Independent Multicast Version 2 Dense Mode Specification (Draft)". *Internet Engineering Task Force*, November 1998.

[GG81] O. Gabber and Z. Galil. "Explicit Constructions of Linear-Sized Superconcentrators". *Journal of Computer and System Sciences*, 22(3):407–420, June 1981.

[GL89] O. Goldreich and L. A. Levin. "A Hard-Core Predicate for All One-Way Functions". In *21st Ann. ACM Symposium on Theory of Computing*, pages 25–32, May 1989.

[glo] The Globus Project. `http://www.globus.org`.

[HAC+98] P. Honeyman, W. A. Adamson, K. W. Coffman, J. Janakiraman, R. Jerdonek, and J. Rees. "Secure Videoconferencing". In *Proceedings of the Seventh USENIX Security Symposium*, pages 123–130. USENIX Association, January 1998.

[Lin97] J. Linn. Generic Security Service Application Program Interface, Version 2. *RFC 2078, Internet Engineering Task Force*, January 1997.

[LM94] T. Leighton and S. Micali. Secret-key Agreement without Public-Key Cryptography. In *Advances in Cryptology - CRYPTO '93*, pages 456–479, 1994.

[Mer] Merit, Inc. Merit Network Home. Ann Arbor, MI, `http://www.merit.net/`.

[MHP98] P. D. McDaniel, P. Honeyman, and A. Prakash. "Lightweight Secure Group Communication". Technical Report 98-2, CITI, University of Michigan, April 1998.

[Mic] Sun Microsystems. Desktop Products - Multimedia. Part number X1085A.,`http://www.sun.com/`.

[Mit97] Suvo Mittra. "Iolus: A Framework for Scalable Secure Multicasting". *Computer Communication Review*, 27(4):277–288, October 1997.

[MPH99] P. D. McDaniel, A. Prakash, and P. Honeyman. "Antigone: A Flexible Framework for Secure Group Communication". In *Proceedings of the 8th USENIX Security Symposium*, August 1999.

[Net96] Network Research Group, Lawrence Berkeley National Laboratory. LBNL's Network Research Group, July 1996. `http://www-nrg.ee.lbl.gov/`.

[oSN] National Institute of Standards and Technology (NIST). "Advanced Encryption Standard (AES) Development Effort". `http://csrc.nist.gov/encryption/aes/`.

[Sch96] B. Schneier. *Applied Cryptography.* John Wiley & Sons, Inc., second edition, 1996.

[Sti95] D.R. Stinson. *Cryptography: Theory and Practice.* CRC Press, Inc., 1995.

[TC99] D. Thaler and B. Cain. "BGP Attributes for Multicast Tree Construction (Draft)". *Internet Engineering Task Force*, February 1999.

[Tho99] K. Thompson. vBNS Multicast Overview, February 1999. `http://www.vbns.net/multicast/overview.html`.

[vBN] vBNS. `www.vbns.net/`.

[WHA98] D. M. Wallner, E. J. Harder, and R. C. Agee. "Key Management for Multicast: Issues and Architectures (Draft)". *Internet Engineering Task Force*, September 1998.

[You] Eric A. Young. `SSLeay`. `ftp://ftp.psy.uq.oz.au/pub/Crypto/SSL/`.

Appendix: A - VRA

Developed at Bellcore, VRA uses a DES-based Goldreich-Levin [GL89] pseudo-random number generator (PRNG) as an initial source of random bits. Goldreich-Levin is expensive, requiring one or more calls to DES for each output bit, so VRA "stretches" these bits into a much longer sequence in two ways. The resulting sequence of pseudo-random bits is then XORed into the data stream.

The first stretching technique uses a long, wide table filled with random bits. A subset of the rows of the table is selected at random and combined with XOR. By selecting in advance the total number of rows n and the number of rows selected at random k, the difficulty of recovering the rows from their XOR sum can be made proportional to a desired (n choose k).

The key to effective stretching is to precompute a wide table, so that a lot of bits are produced from a few calls to Goldreich-Levin. In our application, we use a table with 256 rows, 2,048 bits per row. Initializing

this table is expensive, but once built a table can be used without limit in multiple sessions.

This stretching technique exhibits good short-term randomness, with a key strength of approximately log(nk), or 48 bits for our choices of n and k, but, like any PRNG, admits a birthday attack [Sti95] that effectively halves the key strength.

To compensate for these long-term correlations, VRA uses a second stretching technique, based on random walks through expander graphs. Intuitively, this is a family of sparse graphs with "dense" interconnectivity. (A sparse graph is one in which the ratio of edges to vertices is upper bounded by a constant.) By dense interconnectivity we mean that for any division of the vertices into equal-sized subsets, the ratio of the number of edges between them to the number of vertices is lower bounded by a constant.

The essential property of expander graphs is that a short random walk in an expander graph arrives at a truly random node. Specifically, if we start at any of the graph's n vertices and take log(n) random steps, then the final vertex is very nearly equally likely among all the vertices. A huge value for n foils birthday attacks.

The expander graph we use has $2^{1,024}$ nodes, each node having six neighbors. Such a graph is enormous but VRA uses Gabber-Galil expander graphs [GG81], which can be computed on-the-fly as random steps are made. This ability obviates construction of the entire graph, which is utterly infeasible, and allows the procedure to maintain minimal state, just the neighborhood it is currently traversing.

To avoid making too many Goldreich-Levin calls, each node on the path of the pseudo-random walk is used as output, producing log(n) pseudo-random bits at each step. This concession to performance certainly exhibits some short-term correlations, but any outputs more than log(n) steps apart are essentially independent.

The table and graph techniques produce two streams of pseudo-random bits, one with good short-term characteristics, the other with good long-term ones. These bit streams are XORed together, each masking the other's weaknesses. The resulting stream is the ultimate output of the VRA PRNG.

VRA is a keyed PRNG. The key is the set of bits used to initialize Goldreich-Levin, and can be of any size. VRA has essential cryptographic properties, is based on concrete mathematical arguments, and passes numerous tests of randomness, including Knuth's multidimensional tests and Marsaglia's Diehard battery of tests (see [ARV95]). Furthermore, and of utmost importance for our videoconferencing application, VRA is fast.