Roll Your Own Crypto Services (Using Open Source and Free Cryptography)

Edward Donahue

Roll Your Own Crypto Services (Using Open Source and Free Cryptography)

Edward C. Donahue

January 24, 2002

Abstract

This paper surveys the open source software available to secure the most common applications:

- email and file encryption,
- web access and server oriented services,
- IPsec and VPNs, and finally
- remote session encryption.

We first give an overview of the software available, and what platforms it runs on. Second, we address how to provide the resources assumed by the securing application, such as a source of unpredictable bits to use as keys, and to use to generate public key parameters such as an RSA modulus. We find that the open source cryptographic software community has produced applications that fill most every need, and, indeed, that there are GIAC student papers detailing how to solve a wide variety of problems creatively with this software. However, we also find that choosing and implementing cryptographic packages requires attention to the details, such as the protocol version to implement, the random number source, and recent cryptographic breakthroughs.

Introduction

Today there is an increasing need and demand for cryptographic services; demand for cryptography to provide authentication, privacy, and integrity. Cryptography is such a powerful tool in the information security arsenal because, in many situations, it is difficult to achieve these goals in any other way. On the other hand, security professionals and system administrators, especially those in small businesses or small consulting practices use a variety of open source and freeware tools, such as tcpdump and Ethereal, Snort, crack and l0phtcrack[1]. In software development, the open source movement is just as pronounced, with many efforts including software from several open source efforts. Developers are using databases, LDAP directory servers, and cryptographic packages. This ability of the open source movement to help us provide cryptographic services will be our focus. These cryptographic packages can fill achieve many of the goals for cryptographic services we mentioned above. In most cases, developers are using cryptographic packages which provide a service. Services include securing email and data on storage media, securing access to a web server, and implementing a virtual private network (VPN) to connect a number of local area networks or to provide road warriors secure access. By using these packages, these developers avoid development costs, and usually implement a popular, carefully designed and extensively evaluated standard. It is even more important to adhere to standards in cryptography than in other areas, because they not only make interoperability more achievable, but standards have been more thoroughly scrutinized, and are less likely to have serious security problems. Of course, all this scrutiny does not guarantee the

[1] Older versions.

standard is free of security problems and says absolutely nothing about the care with which the protocols were implemented in that particular software package.

While some question the safety of open source security software, a recent article in the BusinessWeek Online edition gave several examples of companies packaging this software as a business strategy, including Guardent, from Waltham (Mass.). [36] Other examples given in this article included EDS, IBM, Silicon Defense and Covalent. The most significant problem mentioned seemed to be that documentation, management and configuration of open source software was often inferior to what the commercial software offered. The lack of funds to pay for testing and accreditation by FIPS (Federal Information Processing Standards) and Common Criteria labs is another problem. Perhaps these commercial firms will be motivated to correct the situation.

In this paper we will review software applications which provide open source, integrated solutions to secure the most popular application classes. Our review will be cursory indeed, we shall ignore many important aspects, and concentrate on where to find the implementations for various operating systems, their maturity and where to get the 'randomness' that is essential for the packages to provide the advertised security. We shall identify a few security concerns to be aware of. We shall look at implementations of four protocols:

- first, Pretty Good Privacy (PGP) used for email and file encryption,
- second, the Secure Sockets Layer (SSL) or Transport Layer Security (TLS) most popularly used for secure access to a web server;
- third, IPSec used for VPNs; and
- finally, Secure Shell (SSH) used to replace unsecured remote access applications such as telnet, rlogin and ftp.

PGP: Email and File Encryption

For email and file encryption, by far the most popular solution is PGP, Pretty Good Privacy. While this originally referred to the application invented, designed and implemented by Phil Zimmerman, PGP has now been codified as an Internet standard RFC2440, OpenPGP, so that any vendor or organization can implement PGP as an open standard and inter-communicate with other versions of PGP. The commercial package is available from the McAfee and Sniffer business units of Network Associates at http://www.pgp.com. The current version is PGP Corporate Desktop 7.1. However, PGP freeware is available to U.S. citizens in the United States or Canadian citizens in Canada for personal, non-commercial use from the MIT distribution site for PGP at http://web.mit.edu/network/pgp.html, or, for others, from the International PGP Home Page at http://www.pgpi.org. The Gnu Privacy Guard, GPG, is a "complete and free replacement for PGP", to quote their web site at http://www.gnupg.org. PGPdisk, a program to encrypt entire disk partitions is not in the latest freeware PGP versions, but an earlier version PGP 6.02i is still on the International PGP Home Page and does include it. If you need to communicate between the geek world using GPG and the corporate world using the commercial version, you should be aware of an ongoing discussion on the cryptography mailing list (cryptography@wasabisystems.com). Only NAI version 7.0 and up claim to comply with the OpenPGP standard, so it is possible you will have

difficulty communicating between an earlier version of the commercial PGP and GPG. This discussion mentioned many claimed incompatibilities between the two worlds, ranging from text based signatures to GPG not shipping with IDEA. To find this discussion, search on "PGP & GPG compatibility". PGPfone is a software package which encrypts voice over IP links. Version 1 is available as freeware from the International PGP Home Page. A complementary solution is the Aegypten project from German authorities. The aim is to incorporate OpenGPG into open source desktop mail clients such as KMail and mutt. See http://www.gnupg.org/aegypten/.

Security: The PGP FAQ [39] discusses a variety of attacks against PGP including key stroke logging, reading memory, disk cache snooping and reworking code. There are key stroke logging viruses and trojans written to steal the passphrase to a PGP key. If you keep your PGP private key on the hard disk the bad guys are home free after stealing your passphrase. The FBI program Magic Lantern has been in the news lately as implementing this technique. [38] The PGP key stealing virus Caligula was available for download at codebreakers.org according to a cypherpunks message, though the site may have moved since I could not find it. [40]

The author has used the MIT version of PGP for several years. It integrated seamlessly with Microsoft Outlook. Though, it did not integrate with Netscape Messenger, the author has not found using the clipboard to encrypt and decrypt to be onerous, so he did not find integration with the mail client to be essential. On the other hand, the key management does take more attention than he would like. He has sometimes found it difficult to introduce even reasonably sophisticated computer users to any version of PGP that is not fully integrated with the email client. In addition, the key management often seems to go awry. They seem to lose keys, fail to back them up before reloading their operating system, etc. Straightening out these problems over email or the phone has given him an appreciation for those who man help desks. The author corresponds with a small circle of about twenty security professionals. New email addresses are frequent, and if you don't keep up, you find that you have three or four keys for a particular addressee, two of which may be no longer relevant. It is possible that better key management tools could improve this situation without abandoning the PGP 'Ring of Trust' model. In this circle, they have reduced the 'Ring of Trust' to a pseudo-Certificate Authority, since each key is signed by the security officer after he checks the fingerprint of each key via a telephone conversation the owner. This works well in a small circle, but clearly would not scale well.

Operating Systems -- GPG claims to compile and run on Linux, FreeBSD, OpenBSD, NetBSD and Windows 9x/NT/2K/ME.

Algorithms -- Most versions of PGP come equipped with large selection of algorithms. While you may eventually need to add an algorithm to PGP, the 'standard set' can usual fill your requirements. For example, Gnu Privacy Guard comes with El Gamal (signature and encryption), DSA and RSA for public key algorithms, AES; 3DES, Blowfish, Twofish and CAST5 for secret key algorithms and SHA-1, MD5, RIPE-MD and TIGER

for cryptographic hashing algorithms. The only high profile algorithms missing from the list are patented ones such as Idea. [11]

Source of Random -- The author looked only at the GPG source of random. In this case you need to tell the package what to use. They recommend /dev/random on Linux and BSD systems, as well as installing a package that adds /dev/random on Solaris systems. It is not clear what source you would use for random on a Windows OS, though they do mention the lack of a good entropy source on many operating systems and not on Windows.

SSL/TLS: Socket and Web Encryption

Most people are familiar with SSL as the method they use to secure credit card, bank and other sensitive data as it is sent to and from a web server. The current version of SSL is version 3.1, also known as TLS, Transport Layer Security. However, SSL, Secure Sockets Layer, as its name implies is not limited to securing web transactions. While Netscape's primary intention in the design of SSL may have been to secure HTTP sessions, they designed SSL so that it sits in the protocol stack between the TCP layer and the service/port layer and protects a socket. [9]. A socket is a source and destination IP address and port number quad. While most people identify SSL with encryption of transactions with web servers, there are several papers in the GSEC archive illustrating other applications for SSL. Christopher Ursich's GSEC paper [22] gives an example of using SSL to secure Pine IMAP email sessions. David Severski's paper [28] shows a similar example of using TLS to protect the email on the link to the local server. Of course, PGP is still useful in this situation, since it provides end-to-end security, while the SSL/TLS applications secure the link to or from the local server.

The main open source version OpenSSL is based on the SSLeay library developed by Eric A. Young and Tim J. Hudson. The current version of OpenSSL 0.9.6c was released on 21 December, 2001 and is available for download at http://www.openssl.org. As is usual, this site is more oriented toward *nix users than Windows users, but there is a link to http://www.iconsinc.com/~agray/ossldev/dev/ which has MS Developers Studio workspace and project files to help compile and link the code using the MS Developers Studio environment. If you are using OpenSSL with an Apache web server, there is a package mod_ssl from code written by Ralf Engelschall which he developed based on an earlier package by Ben Laurie. This is available at http://www.modssl.org/. Another implementation of SSL is GNU Transport Layer Security Library (GNUTLS). Information is available at http://www.gnu.org/software/gnutls/ . While this implementation is not as mature as OpenSSL, it is available for testing.

Algorithms -- SSL handles algorithm choices as 'suites', that is a complete set of algorithms, modes and key lengths to provide all the required cryptographic services. The public key algorithms offered include RSA (both key transfer and signature), Diffie-Hellman, and DSS. The symmetric key algorithms are Triple DES, DES, RC4, RC2, and IDEA, while the cryptographic hash functions are MD5 and SHA. OpenSLL implements most of the algorithm suites for SSL 3.0 and TLS 1.0, eliminating only the 40 and 56 bit

'export' suites and the Fortezza suites primarily of interest to the U.S. Department of Defense. Currently the most common algorithm used for SSL is reputed to be RC4 [12]. This may be because RC4 is the most efficient encryption algorithm in the suite and so the default choice of servers since the main load of SSL falls on the servers which are attempting to service many sessions simultaneously. Unfortunately, a recent paper by Fluhrer, Mantin and Shamir shows that RC4 can compromise a few bytes of its key each time a message is sent with that key.[13] Since SSL does a key transfer or key agreement for each session, only these few bytes are compromised. If a long enough key is used (say 16 bytes, 128 bits) the algorithm may be fine in this usage and Ron Rivest, the designer of RC4 and the 'R' of RSA, has argued to this effect.[12] However the more conservative, read paranoid, among us may prefer to stay away from this algorithm. Certainly, we expect AES to be added to the TLS algorithm suite shortly. Conservative users may wish to add AES to the PGP implementation they are using or download a version which includes AES when these are available or add AES from on of the many sites where AES implementations are available such as Brian Gladman's site. [15] It is not clear how difficult this is to accomplish since AES has a block size[2] of sixteen bytes, while most current block ciphers have an eight byte block size.

Source of Random -- OpenSSL needs to have its pseudo-random number generator (PRNG) seeded with RAND_add() or RAND_seed() function. If your system has /dev/urandom or /dev/random, they recommend you use that to seed the PRNG. If not, they recommend you use the Entropy Gathering Daemon [16].Virtual Private Networks: IP Encryption

Virtual Private Networks: IP Encryption

IPsec is the Internet standard for IP level security or Virtual Private Networks (VPNs). There are an incredible number of commercial implementations, with a VPN capability often included in firewalls and routers. Many of these implementations were available before the IPsec standard was codified. The standard was partially an effort to help these disparate devices communicate, since IPsec is used to connect local area nets and computers over the wide area net or Internet. You can connect two sites over the Internet or connect single computers to the local site, either telecommuters, working at home or traveling workers. Two sites are usually connected via a gateway such as a VPN server. In smaller installations this is often included in another appliance such as a firewall. The gateway to gateway connections are tunneled, that is the original IP packet being sent from host to host is wrapped inside another IP packet from the sending gateway to the receiving one. [See 32, p.199 and p.214 for details] In this mode, authentication and/or encryption can be applied to nearly[3] the entire packet, In host to host connections, the original packet is sent with authentication and/or encryption applied to it.

The most visible open source implementation is FreeS/WAN, which besides implementing the required parts of the IPsec RFCs, also implements the IKE key

[2] That is, AES naturally encrypts 16 bytes at a time, though cipher modes such as Cipher FeedBack or Output FeedBack can change this.
[3] The Checksum and Time to Live fields cannot be authenticated or encrypted because they change during transmission.

exchange. It is designed for Linux platforms. Information and downloads can be found at http://www.freeswan.org. The currently recommended releases, as of Jan 1, 2002, are 1.91 and 1.92. Release 1.95 is currently scheduled to debut at the end of January 2002. FreeS/WAN is included in certain Linux distributions (SuSE in Germany, Corel in Canada, among others), and in several firewall and router distributions (Gibraltar (http://www.vianova.at/products/gibraltar/), the Linux Router Project and Astaro. The KAME project (http://www.kame.net/) is an alternative source of free IPsec implementations for several BSD Unix operating systems. This is primarily a project of a consortium of Japanese companies (Fujitsu, Hitachi, NEC, and Toshiba among others). Its aim is IP v6 as well as IPsec. Their web site claims that IPsec is working well for both IPv4 and IPv6 and that it interoperates well. The IKE daemon is claimed to also be ready and to interoperate well, though certificate support 'may need some stabilization'.

CIPE, designed by Olaf Titz, is an alternative to IPsec and FreeS/WAN for encryption at the IP layer. [37] CIPE is intended to be embedded in encrypting routers and so is as 'lightweight' as possible, encapsulating the IP packet in a UDP datagram. There are no public key services available and the key management is rudimentary, primarily consisting of exchanging session keys using a key management key. There is no provision for replacing the key management key, or indeed for negotiating one in the first place. Its algorithm choices are limited to IDEA and Blowfish and it only handles algorithms with 64 bit block lengths, so there would need to be a protocol extension to handle AES with its 128 bit block length. It needs initialization vectors, but they appeared to be the user's responsibility. There is a Windows port to NT based systems but not for W9x systems. The Windows port currently only supports Blowfish.

Operating Systems -- FreeS/WAN is only available for Linux to the author's knowledge, while KAME operates on BSD UNIX systems.

Source of Random -- Linux and BSD have /dev/random available as a source of randomness. KAME uses ipsec_ranbits to get a number of bytes of random bits.

Session Encryption

SSH is designed to secure remote sessions in place of telnet and rlogin. There are many papers on SSH in GSEC archives. Damian Zwamborn's paper gives a succinct explanation of the protocol, its functionality and the attacks it counters.[24] However, most of the others show a wide variety of uses of SSH, for example:

- Shawn Lewis concentrates on installation of the software, both on a Cisco router and on a Unix server;
- Nicholas Lee Capace shows how to 'super-encrypt' a PPTP tunnel with SSH;

The commercial version of this protocol suite is done by SSH Communications Security (www.ssh.com). Their current version is SSH Secure Shell 3.1 which supports version 2 of the protocol. This version is currently embodied in a suite of Internet Drafts at http://www.ietf.org/ID.html. Hopefully, the most important of these will soon be in RFC form. The original version of the protocol, SSH1 has now been denigrated because of

several security problems. These include an RC4 vulnerability and a CRC used as a hash with RC4, an additive cipher.[18] These vulnerabilities preceded the most recent problem with RC4 that has been given so much press in the wireless LAN situation. These version 1 vulnerabilities are probably the reason that SSH is reported as one of the favorite services to attack on www.incidents.org. On the day I looked January 22, 2002, SSH was the fourth most popular target. The CRC/RC4 combination in version 1 fails to prevent the alteration of the underlying plain text it was designed for which it was designed to provide integrity protection. To execute the attack, the adversary must modify the plaintext in the blind. The attacker has to guess the original text and add the difference between the original and the desired text, while also adding in the CRC of the difference into the checksum. However, much plaintext is formatted and stereotypic and this can be a serious vulnerability. The RC4 vulnerability allowed an attacker to recover characters in the password with an multi-stage attack, recovering a character at a time. In response to this, SSH Communication Security disabled RC4 from their distributions in 1997.

OpenSSH is an open source version of SSH available at http://www.openssh.com. The current version is 3.0.2 and supports SSH1 and SSH2. OpenSSH is developed in OpenBSD and then made portable so it will run on as many operating systems as possible. The OpenSSH suite has three protocols: ssh, scp and sftp. The ssh protocol replaces rlogin and telnet, while scp replaces rcp, and sftp replaces ftp. It consists of a server and a client.

There are many alternatives, especially for the client portion. The primary motivation for these alternatives seems to have been making a version that runs on alternate platforms. They include:

- OSSH (ftp://ftp.pdc.kth.se/pub/krypto/ossh/) for the original SSH1 protocol. (This version comes from SSH1 and was the original basis for OpenSSH.) See above for the security problems with SSH1 before using.
- LSH/psst (http://www.net.lut.ac.uk/psst/) for SSH2 and FreSSH (http://www.fressh.org) .
- For Windows, there are no server implementations that the author knows about, but there are several clients, including
  - PuTTY (http://www.chiark.greenend.org.uk/~sgtatham/putty/) by Simon Tatham, and
  - Cygwin (http://www.cygwin.com), a Unix environment for Windows, allows a Windows machine to run the portable version of Open SSH.
- MacSSH (http://www.lysator.liu.se/~jonasw/freeware/niftyssh/) is an SSH2-only client implementation, while
- NiftyTelnet 1.1 SSH (http://www.macssh.com/) is an SSH1 only client.
- For PalmOS, there is Top Gun ssh (http://www.ai/~iang/TGssh/) using pilotSSLeay.
- For Java, there are several SSH1-only implementations including an old version of MindTerm. Mindterm is a commercial product available at [33].

Operating Systems -- OpenSSH runs on a wide variety of *nix systems: AIX, HP-UX, Linux, Irix, SCO Solaris, MacOS X, and more. There are clients for Windows, Mac, Java and even Palm.

Source of Random -- Most of these implementations which are Unix connected use /dev/random. The Windows version uses a user supplied random seed file, so this appears to be a pseudo-random number generator with no update at all from a random pool.

Components Needed

At least three types of components are needed to produce a secure cryptographic system: algorithms, protocols and finally unpredictable secrets. Most of this paper has discussed protocols or more correctly services containing a suite of protocols. The second component is a set of algorithms including symmetric encryption engines such as Blowfish, DES, AES, public key algorithms such as RSA, Diffie Hellman, El Gamal,, cryptographic hashing algorithms such as SHA-1 and MD5 and others. We shall mostly ignore algorithms, because in the course of our research to write this paper, we became convinced that most of these packages contain a large enough suite of algorithms to solve most needs. Should they not, one of the referenced sources may contain the source code for the algorithm you need. [15, 19, 20] It is then usually possible to merge this algorithm into the package being used. While we have touched on this requirement as we discussed each service, we shall now pay more detailed attention to it.

Source of Entropy

Having both a secure algorithm and a secure protocol will not provide a secure solution if an adversary can predict the keys used. A source of entropy or unpredictable data is necessary. Deciding whether this source is sufficiently unpredictable is often difficult. For an in depth discussion of these problems, see RFC1750 [14]. In some cases, the application suite provides such a package, more commonly, they assume that you have such a source available, e.g /dev/random. Most available solutions are pseudo-random number generators though 'true' random number generators that derive their unpredictable data from physical phenomena known or presumed to be unpredictable are intrinsically preferable. Pseudo-random number generators are used where 'true random' is not available or where additional insurance is desired in case the physical true random source deteriorates in some way. These attempt to produce a data stream that cannot be distinguished from random statistically. We shall discuss the cryptographically useful ones further below.

Hardware Solutions -- The hardware solution relevant to most developers today is the Intel Random Number Generator. [1,2] This is a service of the 82802 Firmware Hub Device which stores and manages system and video BIOS. This chip is compatible with the Intel^{R} 8xx chip sets which include Celeron, Pentium III and Pentium 4. This author is not sure how common the 82802 is on PCs sold today and in the past couple of years. It was announced in 1999 and so is a relatively new product. There has also been some discussion on cryptography discussion groups on how to access this device. The Linux version of /dev/random does access the Intel generator, if it is available, in kernel release 2.4.3 and beyond. [41]

Where the Intel generator is not available, the hardware alternative is a commercial random number generator, or a cryptographic co-processor which includes a random number generator. Protego makes two serial port random number generators, the Protego SG100 and SG200.[4] The SG100 is priced at $140.00 or less. While reasonable, this is hardly free. Other hardware random number generators are referenced on the Mandala Center web site: http://mandala.co.uk/links/random.

Software solutions -- Because hardware solutions are often not available, many end up having to use software solutions. Software solutions are by definition pseudo-random number generators, though they often continually add a contribution from a 'random pool' where an allied process attempts to gather unpredictable data. For performing a science experiment, often a pseudo-random number generator such as is found on most computers may be satisfactory. Most of these are linear congruential generators or LCGs.[34 p. 276] However, for cryptographic purposes, the requirements are different. The entire stream from an LCG is predictable based only on the knowledge of a few outputs. [6, 7, 8] The cryptographic community has developed other pseudo-random number generators, which are not so easily predictable. Peter Gutmann's paper [3] details some of the recommended designs for these, ranging from the ANSI 9.17 Generator and Fortezza Generator to Peter Gutmann's own. He also details how Skip, ssh and /dev/random generate their random. Almost all of these generators assume there is a 'random pool' being gathered for them as time goes on so that they do not depend only on the secrecy of the values used to key the cryptographic algorithms that are part of the generator. Since the /dev/random solution appears to be the most widely usable solution, it is the most important to the most people. /dev/random estimates the amount of entropy in the pool and /dev/urandom returns the number of bytes of random the caller requests. The randomness pool is supplied by keyboard, mouse timing, interrupt and block device timing supplied by the kernel in Unix or by DOS interrupts. This data is mixed into the pool with a non-cryptographic mixing function for speed, but mixed with MD5 or SHA-1 upon output. This solution is available on Linux and BSD systems and can be added on Solaris. I am not sure about other Unix systems. Where this is not available, the Entropy Gathering Daemon, a user space substitute for /dev/random written in Perl, is a possible substitute.[16] Its entropy gathering strategy is copied from Peter Gutmann's Unix random pool entropy polling mechanism and so is tailored for Unix. In the Windows arena, Peter Gutmann's Cryptlib has a Windows random gatherer, which is available in source code, but charges for commercial usage.[35] He also has random gatherers for many other operating systems and interfaces to several hardware solutions as well. In Java, Java Secure Random is a solution, though there seem to be a variety of implementations of it from looking at a Google search on Java Secure Random, and there are very few details on how individual implementations have implemented this class. I have not yet come across a specification for the implementation of the class or even a specification of any individual implementation. A final possibility is Yarrow from Bruce Schneier's Counterpane Systems, however, the implementation available at Counterpane is an earlier design and not the design described in the paper.[31] For most of us, the /dev/random solution will currently be the most useful. Hopefully, soon the Intel hardware solution or alternatives will be more easily usable.

Conclusion

The bottom line is that there are open source software suites to provide robust encryption services in many, if not most application situations. On the other hand, providing the service on a particular platform may be a problem. Most solutions run on Linux, and BSD Unix is also well served. Other flavors of Unix are quite well served. There are many clients for Windows, but Windows servers are more of a problem.

Finally, several words of caution should be mentioned. Cryptography is much different than other applications. Just because it runs and provides the service does not mean it is providing the needed security. In a sense, it is similar to a lock. You are never absolutely sure it worked, but you certainly don't want to find out the hard way that it didn't. Given this, the implementation should be tested, not just for functionality, but also to attempt to see that it is doing what it is supposed to. It is easy to believe traffic is encrypted when it is not. Checks such as were run in [22] to verify that the Stunnel did indeed seem to be encrypted are essential. Correct cipher text is hard to prove, but plain text is often easy to spot.

Fortunately, the level of knowledge of cryptography in the security community continues to increase and most professionals are conservative in their use of cryptography. They use protocols, software suites, and algorithms that have been extensively examined and tested such as the GNU suites with their well known algorithms. On the other hand, the author has met many neophytes who believe they can develop robust algorithms and protocols. Often their belief persists in the face of their attempts being repeatedly broken. I would urge extreme caution in the deployment of cryptography. Stick to the well worn paths and be alert to recent breakthroughs. Usually there will be some indication of a recent breakthrough on the sites devoted to the protocol suite. One final caution, this paper has not discussed where open source software cryptography is appropriate. The people employing the software will have to decide this based on the threats they foresee. Software cryptography is inherently vulnerable to physical attacks, or hacking attacks. Certain threats such as modification of the software or capture of the keys from the storage will need to be countered by additional measures or a different solution such as a cryptographic smart card will need to be employed. However open source software cryptography can make a significant contribution to security in most environments, just be careful out there.

References

1. THE INTEL® RANDOM NUMBER GENERATOR, CRYPTOGRAPHY RESEARCH, INC. WHITE PAPER PREPARED FOR INTEL CORPORATION by Benjamin Jun and Paul Kocher, April 22, 1999 available at http://developer.intel.com/design/rng/CRIwp.htm
2. Intel Random Number Generator (RNG): Next Steps, available at http://developer.intel.com/design/rng/rngnxt.htm
3. Chapter 6, Random Number Generation *by* Peter Gutmann available at http://www.cs.auckland.ac.nz/~pgut001/
4. Protego Information AB web page, http://www.protego.se/

5. Intel Chip Sets, http://www.intel.com/design/chipsets/rng/faq.htm
6. INFERRING A SEQUENCE GENERATED BY A LINEAR CONGRUENCE by Joan B Plumstead, Advances in Cryptology, Electronic Proceedings of the Eurocrypt and Crypto Conferences 1981-1997.
7. Cryptanalysis of Pseudo-Random Number Sequences Generated by a Linear Congruential Recurrence of Given Form by Anne Bauval found in Advances in Cryptology, Electronic Proceedings of the Eurocrypt and Crypto Conferences 1981-1997.
8. Lattice Reduction: a Toolbox for the Cryptanalyst by Antoine Joux and Jacques Stern, published in the Journal of Cryptology: the journal of the International Association for Cryptologic Research, May 18, 1994
9. E-Commerce Security: Weak Links, Best Defenses, Anup K. Ghosh, John Wiley & Sons, Inc. , 1998, p.104
10. SANS Security Essentials, Section 1.3.2: Networks, Routers and Firewalls: IP Concepts I, Slide 22
11. http://www.gnupg.org/backend.html
12. "RSA Security Response to Weaknesses in Key Scheduling Algorithm of RC4" by Ron Rivest, found at http://www.rsasecurity.com/rsalabs/technotes/wep.html
13. "Weaknesses in the key scheduling algorithm of RC4", by S. Fluhrer, I. Mantin and A. Shamir, Eighth Annual Workshop on Selected Areas in Cryptography (August 2001).
14. Randomness Recommendations for Security, Request for Comments 1750, D. Easllake, S. Crocker and J. Shiller, December 1994, available at http://nis.nsf.net/internet/documents/rfc/rfc1750.txt.
15. www.gladman.uk.net
16. EGD: The Entropy Gathering Daemon available at http://egd.sourceforge.net/
17. Sun Java 2 Platform Std. Ed. v1.3.1, Package java.security at http://java.sun.com/j2se/1.3/docs/api/java/security/package-summary.html
18. "SSH1 Vulnerabilities" on http://www.ssh.com/products/ssh/cert/vulnerability.cfm
19. Libgcrypt -- http://www.gnu.org/directory/libgcrypt.html is a general purpose crypt library based on the code from GnuPG (see below). It has a wide selection of algorithms.
20. Crypto++4.2 - a Free C++ Class Library of Cryptographic at www.eskimo.com/~weidai/cryptlib.html
21. AES: The Making of a New Encryption Standard by Mitch Richards, Sept. 5, 2001, available at http://www.giac.org/GSEC.php
22. Establishing and Verifying the Stunnel SSL Encryption of Pine IMAP Sessions by Christopher Ursich, Sept 17, 2001 available at http://www.giac.org/GSEC.php
23. A Discussion of SSH Secure Shell by Shawn Lewis, August 4, 4001 (sic) available at
24. An Introduction to SSH Secure Shell by Damian Zwamborn, May 15, 2001 available at at http://www.giac.org/GSEC.php
25. IPsec's Role in Network Security: Past, Present, Future by Christopher Smith, Sept. 17, 2001 available at http://www.giac.org/GSEC.php
26. Sudo and SSH: A Scheme for Controlling Administrator Privileges and System Account Access, Liam Forbes, July 9, 2001, available at http://www.giac.org/GSEC.php

27. Tunneling PPTP Through SSH2 Connections, Nicholas Lee Capace, April 28, 2001, available at http://www.giac.org/GSEC.php
28. Protecting Email in a Hostile World with TLS and Postfix, David Severski, August 22, 2001, available at http://www.giac.org/GSEC.php
29. An Integer Overflow Attack Against SSH Version 1 Attack Detectors, David J. Bianco, March 1, 2001, available at http://www.giac.org/GSEC.php
30. "X Tunnels through SSH", Layne Bro, November 30, 2000, available at http://www.giac.org/GSEC.php
31. "Yarrow-160: Notes on the Design and Analysis of the Yarrow Cryptographic Pseudorandom Number Generator", John Kelsey, Bruce Schneier, and Niels Ferguson, Sixth Annual Workshop on Selected Areas in Cryptography, Springer Verlag, August 1999, available at http://www/counterpane.com/yarrow-notes.html
32. IPSec: Securing VPNs, Carlton R. Davis, Copyright 2001, Osborne/McGraw-Hill.
33. http://www.appgate.org/products/mindterm/personal/mindterm_old_downloads.html
34. Numerical Recipes in C: The Art of Scientific Computing, Second Edition, William H. Press, Saul A Teukolsky, William T.Vettering and Brian P. Flannery, Cambridge University Press, 1992.
35. http://www.cs.auckland.ac.nz/~pgut001/cryptlib/
36. "Is Open-Source Security Software Safe?" , Alex Salkever, BusinessWeek online, December 11, 2001, http://www.businessweek.com/bwdaily/dnflash/dec2001/nf20011211_3015.htm?$se
37. http://sites.inka.de/sites/bigred/devel/CIPE-Protocol.txt
38. http://www.usethesource.com/articles/01/11/20/121216.shtml
39. http://www.stack.nl/~galactus/remailers/attack-5.html#4
40. http://www.shmoo.com/mail/cypherpunks/feb99/msg00059.html
41. http://www.linuxhq.com/kernel/v2.4/patch/patch2.4.3/linux_drivers_i810_rng.c.html

Last Updated: September 25th, 2020

Upcoming SANS Training

Click here to view a list of all SANS Courses

SANS October Singapore 2020	**Singapore, SG**	**Oct 12, 2020 - Oct 24, 2020**	**Live Event**
SANS Community CTF	**,**	**Oct 15, 2020 - Oct 16, 2020**	**Self Paced**
SANS SEC504 Rennes 2020 (In French)	**Rennes, FR**	**Oct 19, 2020 - Oct 24, 2020**	**Live Event**
SANS SEC560 Lille 2020 (In French)	**Lille, FR**	**Oct 26, 2020 - Oct 31, 2020**	**Live Event**
SANS Tel Aviv November 2020	**Tel Aviv, IL**	**Nov 01, 2020 - Nov 06, 2020**	**Live Event**
SANS Sydney 2020	**Sydney, AU**	**Nov 02, 2020 - Nov 14, 2020**	**Live Event**
SANS Secure Thailand	**Bangkok, TH**	**Nov 09, 2020 - Nov 14, 2020**	**Live Event**
APAC ICS Summit & Training 2020	**Singapore, SG**	**Nov 13, 2020 - Nov 21, 2020**	**Live Event**
SANS FOR508 Rome 2020 (in Italian)	**Rome, IT**	**Nov 16, 2020 - Nov 21, 2020**	**Live Event**
SANS Community CTF	**,**	**Nov 19, 2020 - Nov 20, 2020**	**Self Paced**
SANS Local: Oslo November 2020	**Oslo, NO**	**Nov 23, 2020 - Nov 28, 2020**	**Live Event**
SANS Wellington 2020	**Wellington, NZ**	**Nov 30, 2020 - Dec 12, 2020**	**Live Event**
SANS OnDemand	**OnlineUS**	**Anytime**	**Self Paced**
SANS SelfStudy	**Books & MP3s OnlyUS**	**Anytime**	**Self Paced**