SANS Institute
InfoSec Reading Room

Analysis of a Secure Time Stamp Device

This paper discusses the design of a Secure Time Stamp device used to securely timestamp digital data, such as computer documents, files, and raw binary data of arbitrary format. Thus, the device is used to prove two facts: Existence: That a file existed on a given date & time and Data Integrity: That the file was not altered since the time it was stamped.

Analysis of a Secure Time Stamp Device

GSEC Practical Assignment Version 1.2f
for GIAC Certification in
Security Essentials

Chris Russell
October 17 2001

Table of Contents

1 INTRODUCTION

This paper discusses the design of a *Secure Time Stamp* device used to securely timestamp digital data, such as computer documents, files, and raw binary data of arbitrary format. Thus, the device is used to prove two facts:

- **Existence:** That a file *existed* on a given date & time.
- **Data Integrity:** That the file was *not altered* since the time it was stamped.

These two facts are essential for a number of purposes, including but not limited to:

- Gathering and registering binary data to be used as forensic evidence, such as computer files, memory dumps, packet recorder data, security analysis logs, etc.
- Electronically "notarizing" the date and time of inventions and other time-critical documents, such as business plans, intellectual property, engineering documents, source code, contracts, etc.
- Generating secure audit logs for financial transactions, crypto key generation and management, system management, etc.

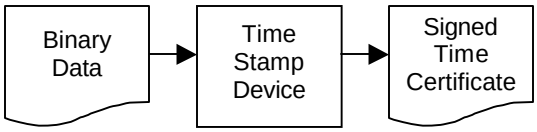
The design of this device is an excellent, real world example on the application of several security engineering principles and technologies, including:

- Nonrepudiation, authentication, and data integrity.
- Cryptographic algorithms.
- Formal analysis of security protocols, chain of trust, and belief logic proofs.
- Hardware tamper resistance, tamper detection, and secure audit trails.
- Secure real-time clocks.

This device was originally designed by Tom Rodriguez and myself in August 2001. Unfortunately, we subsequently discovered a patent filed by Robert Blandford [2] presenting a nearly identical design. Thus, our project was scrapped. Looking for the silver lining, we were obviously on the right track and our design was validated, albeit not unique.

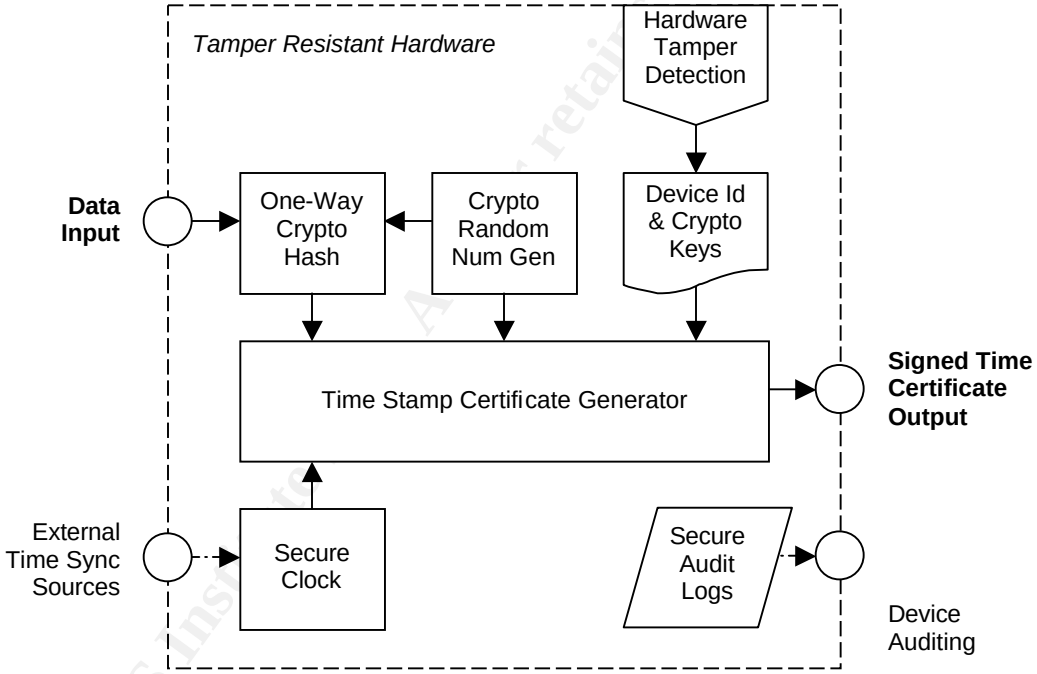
2 THE BASIC DESIGN

2.1 OVERVIEW



The timestamp device creates Signed Time Certificates for data input into the device. Certificates are stored into separate files (from the original data). Therefore, a file can be time stamped without altering its contents.

The basic system works as follows:



1. The data is input and buffered into the device.

2. A cryptographic one-way hash (such as SHA-1, MD5, SHA-256, etc.) is generated from the buffered data.

3. The current date/time is read from a secure real-time clock within the device.

4. A certificate is generated containing the hash, current date/time, and device serial number, and the certificate is signed using a private key stored within the device.

Thus, in order to read and verify a time stamp:

1. The signed time certificate is verified using the device's public key. If the signature does not match, then the certificate is invalid and thus it is rejected.

2. The hash stored in the certificate is verified by calculating a new hash from the data. If the hash values don't match, then either (1) the certificate does not belong to that particular data file, or (2) the data file has been altered, and thus it is rejected.

3. If both verifications (above) succeed, then the date/time is read from the certificate and returned as the authoritative result.

2.2 NON-REPUDIATION

Perhaps the most important feature of this device is non-repudiation. It is essential that a Signed Time Certificate be undeniable proof of the date & time that the data was notarized, and the specific Time Stamp device (serial number, etc.) that was used to perform the notarization.

Ensuring non-repudiation requires a number of features, including cryptography, a secure clock, secure (auditable) logs, tamper resistance, and device identification. Each of these features are essential components to auditing and forensically determining the validity of a Signed Time Certificate, such that a certificate could withstand scrutiny as key evidence in a court of law.

2.3 DIGITAL SIGNATURES

The Time Certificate is signed using a cryptographic digital signature algorithm, such as DSS, RSA, NTRU, etc.

Each Secure Time Stamp device has its own unique private key used to sign the certificates. Preferably, this key is generated within the tamper resistant hardware (such as FIPS 140-1 level 3 or 4 [3]) of the device and never exposed externally. Thus, no one can ever know the value of the private key without defeating the hardware tamper resistance features.

Alternatively, if the key is generated by software running on general purpose hardware (without hardware tamper resistance), then they key may be protected via a combination of portable crypto devices (smart cards, dongles, etc.), software tamper resistance, and/or code obfuscation with embedded split-keys (to ensure that the entire private key is never completely exposed in memory at any time). These techniques are used by a number of DRM (Digital Rights Management) technology providers, such as Microsoft, Real, Intel, Intertrust, and others.

2.4 POTENTIAL VULNERABILITIES

The basic system described in §2.1 above is an oversimplification of the full design. Without specific additions, it is vulnerable to a number of potential exploits, including:

1. A hash generated exclusively from the input data may be vulnerable to a Birthday Attack [5][13].

2. If the device is physically tampered, then the private key can be read from memory and used to forge Signed Time Certificates. Or, the device can be reprogrammed using a different, known private key.

3. Unless the public key (used to verify the signature) is securely authenticated, then a forged public key may be distributed and used to "verify" forged time certificates.

4. It may be possible to alter or "roll back" the internal real-time clock, thus producing fraudulent time certificates from a legitimate device.

The rest of this document describes the remaining details necessary to eliminate these vulnerabilities and create a secure, tamper-resistant device.

3 THE BIRTHDAY ATTACK

3.1 THE PROBLEM

A one-way cryptographic hash is a special type of hash function that exhibits the following properties:

- Given data X, it is easy (i.e., computationally efficient) to compute the hash H(X).

- Given the hash H(X), it is practically impossible (i.e., "NP-Complete" [1]) to find another data Y that generates the same hash value (i.e., H(X) = H(Y)), even if the value of X is known.

- Using cryptographic techniques, a minor change in the binary value of X produces major, chaotic changes to the value of H(X). In essence, H(X) behaves like a one-way pseudo-random number generator.

If a hash is n bits long, then a brute force attack (to find a value Y such that H(Y) = H(X)) is typically $O(2^n)$ in complexity [5][1]. For example, the MD5 algorithm generates a 128-bit hash. Therefore, a brute force attack should require 2^{128} attempts, which is well beyond the range of current technology.

However, the Birthday Attack reduces the complexity to only $O(2^{n/2})$ [5]. That means only 2^{64} attempts are needed to crack for MD5, which is possible using specialty, *massively* parallel computers [13]. The attack works by generating a sequence of different data values (X_1, X_2, …) and finding *any two* that generate the same hash (i.e., $H(X_n) = H(X_m)$). It is much simpler to find two values (X_n and X_m) that generate the same hash ($H(X_n) = H(X_m)$) than to find a single value (Y) that generates a given hash (H(Y) = H(X)).

Thus, it is possible to find two documents that generate the same hash value. While the first document may contain legitimate data, the second could be crafted to contain malicious or contrary data. If these documents were legal contracts, just imagine how much havoc they could create! It is essential to properly authenticate (or identify) the correct document that corresponds to a given time certificate.

3.2 THE SOLUTION

It is not possible to perform 2^{64} attempts using a single Secure Time Stamp device. Therefore, the crack must be performed offline using massive parallel processing. Once two documents (X_n and X_m) are found that generate the same hash value($H(X_n) = H(X_m)$), then only one of them needs to be input into the Secure Time Stamp device. The resulting certificate would then work for either of the two documents.

To eliminate this attack, the device appends a nonce to the data prior to hashing. A nonce (short for "number used once") is a single-use number. A new nonce value is generated every time a new document is time stamped.

Therefore, even through the hash values of the two documents are identical:

$$H(X_n) = H(X_m)$$

once the nonce is added, the hash values no longer match:

$$H(N + X_n) \bullet H(N + X_m)$$

and thus the signed time certificate properly identifies the valid document and rejects the forged one.

If the nonce is predicable (such as an simple count sequence: 1, 2, 3, …), then the attack can be modified to locate two matching documents for a given nonce value. Therefore, the nonce values must be unpredictable.

Ideally, nonces should be created using a truly random number generator. Such a generator would rely on the chaotic physical properties (physics) of the hardware, such as thermal vibrations, in order to generate a nondeterministic sequence of truly random, unpredictable numbers. Intel builds a truly random number generator into their Pentium III (and subsequent) processors using the RNG [6] instruction. Other truly random number generators in history included:

- SGI's *Lavarand*, a bunch of lava lamps at SGI that were periodically photographed and processed to produce a chaotic sequence of numbers. (Used to be available at http://lavarand.sgi.com.)
- Rand Corporation's *RAND Tables*, which used an electrical random pulse generator to create an "electronic roulette wheel" [1 page 422].

Alternatively, a pseudo-random number generator may be used. However, if the generator function is reversible (not one-way), then an inverse function can be used to calculate the seed value and hence crack subsequent random numbers. Instead, a cryptographically random number generator is required. One technique is to encrypt (or cryptographically) hash the pseudo-random numbers prior to being used, thus ensuring

the generator function is one-way (i.e., "NP-Complete" to crack). Appendix 3 of the DSS standard [4] defines two algorithms for generating cryptographically random numbers, one using SHA-1 and the other using DES. Other, more efficient algorithms exist, such as ISAAC [7]. This is an area of continuing research.

4 TIME STAMP ALGORITHM

This section describes the specific algorithms in more detail for generating and verifying time certifications. On the left side are the algebraic steps written using GNY Logic nomenclature [10], and on the right side is a textual description for each step.

GNY Logic is a method for analyzing belief systems for security protocols and is an extension and improvement over the more traditional BAN Logic [11] approach. (Granted, I'm using the GNY nomenclature less formally in order to illustrate the algorithm rather than to prove correctness of belief inferences.) Methods for proving trust and belief systems in security protocols are an active area of research.

For simplicity, the signed time certificate $(N, H(N, X), T, S, \{H(N, H(N, X), T, S)\}_{-K})$ is also referred to as (C). These two expressions are synonymous.

4.1 CREATING A SIGNED TIME CERTIFICATE

In order to timestamp binary data (X), the Secure Time Stamp device (D) performs the following steps, in order:

1.	$D \mid\equiv (+K, -K, S)$	The device (D) internally generates a public key (+K) and private key (-K) pair. Furthermore, the device has a unique serial number (S). These values are generated once and stored into persistent storage (such as NVRAM) within the device.
2.	$D < X$	The binary data (X) to be time stamped is input into the device (D).
3.	$D \mid\equiv \#(N)$	A nonce (N) is created using a cryptographic or truly random number.
4.	$D \mid\equiv H(N, X)$	The nonce is appended to the data and a digital fingerprint of the nonce+data is generated using a one-way cryptographic hash, such as MD5, SHA-1, SHA-256, or similar algorithm.
5.	$D \mid\equiv (N, H(N, X), T, S)$	A certificate is created, containing the value of the nounce, the digital fingerprint of the data, the current date & time, and serial number of the time stamp device.

6. $D \mid\equiv \{H(N, H(N, X), T, S)\}_{-K}$

   The signature is generated by encrypting (asymmetric encryption) a hash of the certificate (or similar signing algorithm).

7. $D \mid\sim C$

   The resulting Signed Time Certificate $(N, H(N, X), T, S, \{H(N, H(N, X), T, S)\}_{-K})$ is output from the device.

4.2 VERIFYING A SIGNED TIME CERTIFICATE

In order for user (U) to read and verify the timestamp (C) associated with binary data (X), the following steps are performed in order:

1. $U \ni (X, C)$

   In order to read and verify a timestamp, the user (U) must have the data (X) and corresponding signed time certificate (C), which is equal to $(N, H(N, X), T, S, \{H(N, H(N, X), T, S)\}_{-K})$.

2. $D \mid\sim (+K, S)$

   The device transmits its public key (+K) and unique serial number (S).

3. $U \mid\equiv f(S)$

   If the serial number transmitted by the device matches the certificate, then proceed. Otherwise, the certificate is rejected. This is not a cryptographically strong form of authentication and thus is not sufficient for believing the certificate, only for rejecting it.

4. $$\frac{U \mid\equiv D \mid\sim +K}{U \mid\equiv \overset{+K}{\rightarrow} D}$$

   If it can be proven that the public key was transmitted by the device (and not spoofed), then +K is guaranteed to be the correct key to validate the signature.

   Note: This is a potential weakness in the protocol. Since the transmission (+K, S) is not authenticated, it can be spoofed. Therefore, this communication much be performed over a physical point-to-point connection or similar technique to prevent spoofing.

5. $$\frac{U < +K,\ U \ni \{H(N, H(N, X), T, S)\}_{-K}}{U \ni H(N, H(N, X), T, S)}$$

The signature is decrypted, returning the hash of the certificate. (However, the certificate is not trusted yet.)

6. $$\frac{U \mid\equiv f(H(N, H(N, X), T, S))}{U \mid\equiv D \mid\sim C}$$

The hash value is recomputed from the cleartext of the certificate (N, H(N, X), T, S) and compared against the decrypted hash from the signature. If the hashes match, that proves the certificate (C) was generated by the device (D). Otherwise, it is rejected.

7. $$\frac{U \mid\equiv f(H(N, X))}{U \mid\equiv C \mid\Rightarrow X}$$

The hash of the nounce+data H(N, X) is recomputed and compared against the hash stored in the certificate. If the hashes match, that proves the data belongs to the certificate. Otherwise, it is rejected.

8. $$\frac{U \mid\equiv D,\ U \mid\equiv D \mid\sim C,\ U \mid\equiv C \mid\Rightarrow X}{U \mid\equiv C,\ U \mid\equiv T}$$

Because the user trusts the device, proved the device generated the certificate, and proved the certificate corresponds to the data, therefore the user trusts the certificate and extracts the data/time value from it.

Note: This last step illustrates another potential vulnerability: The user must trust the physical device. If the device has been tampered with, then all bets are off! Therefore, hardware tamper resistance and secure auditing are necessary in order to justify the user's leap of faith in his hardware.

5 TAMPER RESISTANCE

5.1 CHAIN OF TRUST

Every security protocol has a chain of trust. For the Secure Time Stamp device, the root of trust is the physical device itself.

1. First, the device transmits its public key.
2. Then, the public key authenticates the signed time certificate.
3. Finally, the time certificate authenticates (and verifies data integrity) of the data.

Each signed time certificate identifies the specific Time Stamp device used. Thus, for added security in determining the authenticity of a certificate, the actual device can be identified, physically inspected to determine if tampering has occurred, and audited using its secure logs.

If the device itself has been tampered, then the public key and time certificates may have been forged. Therefore, it is essential to protect against hardware tampering *and* be able to detect if any tampering has occurred.

5.2 HARDWARE TAMPER RESISTANCE

Hardware Tamper Resistance ensures that the hardware cannot be physically compromised, thus preventing a user from accessing private crypto keys, altering the real-time clock, tampering with the software, forging certificates, etc.

A case alarm may be installed onto the physical case. If an attempt is made to open the case, then an alert is recorded into the secure log, the private key is permanently erased, and the device is deactivated.

Hardware tamper resistance may necessitate a power source to be built into the device. One implementation is to store private keys and critical data (but not the logs!) into volatile memory and power the memory from an internal power source. If the case is opened, then power is cut off and the contents in memory are instantly[1] lost forever.

PCBs (printed circuit boards) and wire etchings may be coated with an epoxy or similar substance. Attempting to physically remove the epoxy would result in ripping the chips off the board, thus damaging the PCB and destroying the private keys and critical data. This approach is used by nCipher (http://www.ncipher.com) in the design of their FIPS 140-1 level 3 tamper resistant hardware systems, such as nShield (http://www.ncipher.com/nshield).

The PCBs may also be wrapped in a tamper resistant bag (similar to an anti-static bag, except it contains a mesh of tiny conductors along its surface). If the bag is punctured or torn, then power is disrupted and the private keys and critical data are lost.

The hardware should provide a level of tamper resistance equivalent to FIPS 140-1 level 3 or 4 [3].

[1] The book *Security Engineering* [13] discusses various high-tech attacks possible against hardware (pages 280-284), such as freezing RAM below -20ºC to preserve its data, using recovering memory remanence data from powered down RAM chips, etc. Unfortunately, these attacks are out of scope and the Secure Time Stamp device is most likely susceptible to them. You have to draw the line somewhere! ☺

5.3 PHYSICAL TAMPER EVIDENCE

Physical Tamper Evidence makes it possible to determine through physical inspection if a device has been tampered or cloned.

Evidence may include:

- A self-destructing case that physically breaks when opened.
- A serial number inconspicuously etched onto the PCB.
- Security tags that self-destruct when they are removed. Ideally, these tags should be difficult to counterfeit, such as holographic tags, and are installed both inside and outside the case of the device.
- Etc.

5.4 SOFTWARE TAMPER RESISTANCE

A software-only implementation cannot take advantage of hardware tamper resistance and therefore may employ software tamper resistance techniques (code obfuscation, self modifying code, encrypted code segments, anti-debugging, hardware monitoring, virtual memory page table change detection, etc) and split keys (to ensure that private keys are never completely exposed in memory at any time) to help strengthen security and non-repudiation. Portable crypto devices (such as smart cards, dongles, etc.) may be used as well.

If implemented without software tamper resistance, then the application may require users to personally sign and certify time certificates when they are created, so they are in essence personally vouching for the authenticity and correctness of the certificate.

Alternatively, a server implementation may encrypt the signing keys with a symmetric key "passphrase-based" algorithm. Thus, if they encrypted key is even stolen, it is useless without the passphrase. When the application is launched (either manually, when the machine is rebooted, or otherwise), a trusted administrator must enter the passphrase in order to decrypt the signing key. The signing key is only stored into volatile memory (such that it disappears when power is removed) and never stored unencrypted into permanent storage, such as flash memory, a hard drive, etc. This technique has been used in the past for certificate authority servers, where security is paramount.

5.5 INTEGRITY DETECTION

All critical data and files stored within the device may be digitally signed, including software, firmware, and microcode. When the Time Stamp device is powered up, the integrity of these files are verified by checking the digital signatures.

This power up self-check feature should be implemented into read-only hardware, such as ROM or password protected read-only flash memory (e.g., DiskOnChip [14]).

6 AUTHENTICATING THE PUBLIC KEY

The device's public key is used to authenticate signed time certificates. However, there is no specific authentication mechanism implemented for the public key itself. Therefore, it is possible for someone to forge a public key and claim that it belongs to the device.

As indicated in §5.1 above, the physical device is the root of all trust, and the device will always advertise its public key when requested. Therefore, as long as the public key is always obtained directly from the device and not from some other source, its authenticity is assured.

Special care must be taken to ensure another device cannot intercept the request and spoof a forged public key advertisement. This is most easily accomplished by using a direct link-level connection (point-to-point) to the device, thus physically isolating it from any other networks and devices.

More versatile forms of link security and authentication are possible; however, these features are outside the scope of the initial design for this device.

7 SECURE CLOCK

The real-time clock must be accurate. Otherwise, the Time Stamp device generates time stamps using the wrong date & time and is worthless.

Despite how accurate the hardware clock may be, it is insufficient to rely solely on its accuracy without performing periodic recalibration. Furthermore, there should be mechanism for certifying and auditing the accuracy of the clock.

7.1 TIME RECALIBRATION

The clock may be synchronized against known secure time sources, such as GPS, NIST WWV/WWVH/WWVB broadcasts [9], NIST Internet Time Service [8], or secure networked time servers.

A GPS or time broadcast radio receiver (such as NIST WWV, WWVH, & WWVB in the United States) may be built into the Time Stamp device. If a GPS is built in, then longitude and latitude information is recorded and securely logged as well.

The device can use other time protocols in order to synchronize against known time servers. The IETF (Internet Engineering Task Force) is currently developing a Secure NTP protocol (stime) [12]. Unfortunately, it is currently only in draft form and not yet an official standard.

7.2 TIME CERTIFICATION

The Time Stamp device may be certified by an approved certification lab against a known time source. Once the device is recalibrated and certified, then the lab will

download a signed recalibration certificate (signed using the private key of the certification labs) into the device to be stored and securely logged.

7.3 TIME AUDIT LOGS

Every time the value of the real-time clock is changed, the event is recorded to a secure log, including: the old value of the clock, the new value, source of the new time value (GPS, NIST, a specific NTP server, etc.), and so forth.

The device periodically records the current time and current GSN (Global Sequence Value, see §8 below) to a secure log (and, of course, increments the GSN in the process), thus enabling detailed auditing and forensic analysis on the accuracy of the real-time clock if necessary.

8 SECURE LOGS

The device records a number of secure logs, including:

- Time logs, indicating every time the built-in real-time clock is updated, how much the time was changed, etc.
- System logs, indicating when the device is powered on or rebooted, when the firmware or software is updated, etc.
- Hardware logs, indicating if/when the device has been tampered, opened, etc.
- Periodic internal self-check logs.
- And so forth.

Each log is securely signed by the Time Stamp device, protecting it against tampering. Individual entries in a log may be individually signed. Nonces are added as necessary to protect the security of the digital signature algorithm.

The device generates a monotonically increasing sequence of integers (e.g., 1, 2, 3, …) called the Global Sequence Number (GSN). Every time a new entry is recorded into a secure log, the current GSN value is incremented and recorded with the entry. Thus, it is always possible to determine the precise order of events recorded across multiple log files. This is essential for auditing the device and forensic analysis.

GSN numbers are used instead of date/time values to audit the precise order of events because the real-time clock can be modified and therefore is not guaranteed to be monotonically increasing.

9 CONCLUSION

This paper used the design of a Secure Time Stamp device to illustrate several important security engineering principles and techniques, listed in §1 above.

It also illustrated the importance (and complexity!) of logically analyzing security protocols, such as initial beliefs (the root of trust) and belief inferences (the chain of trust), and how these logic proofs can be used to identify assumptions, security weaknesses, and limitations that may have been otherwise overlooked.

10 BIBLIOGRAPHY

1. *Applied Cryptography, Second Edition*, Bruce Schneier, John Wiley & Sons, © 1996, pages 237-242.

2. *Devices to Supply (1) Authenticated Time and (2) Time Stamp and Authenticate Digital Documents, U.S. Patent #5,189,700,* http://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO1&Sect2=HITOFF&d=PALL&p=1&u=/netahtml/srchnum.htm&r=1&f=G&l=50&s1='5189700'.WKU.&OS=PN/5189700&RS=PN/5189700, Robert Blandford.

3. *FIPS 140-1, Security Requirements for Cryptographic Modules*, http://www.itl.nist.gov/fipspubs/fip140-1.htm, NIST.

4. *FIPS 186-2, Digital Signature Standard*, http://csrc.ncsl.nist.gov/publications/fips/fips186-2/fips186-2.pdf, NIST, appendix 3.

5. *Handbook of Applied Cryptography*, Alfred Menezes et. al, CRC Press, © 1997, pg 369-371.

6. *The Intel Random Number Generator* (whitepaper), http://developer.intel.com/design/security/rng/techbrief.htm, Intel Platform Security Division.

7. *ISAAC, A fast cryptographically random number generator*, http://www.burtleburtle.net/bob/rand/isaacafa.html, Bob Jenkins.

8. *NIST Internet Time Service*, http://www.boulder.nist.gov/timefreq/service/its.htm, NIST.

9. *NIST Radio Station*, http://www.bldrdoc.gov/timefreq/stations/wwv.html, NIST.

10. *Reasoning About Belief in Cryptographic Algorithms*, http://java.sun.com/people/gong/papers/gny-oakland.ps.gz or http://citeseer.nj.nec.com/gong90reasoning.html, Gong, Needham, & Yahalom.

11. *The Scope of Logic Authentication*, http://citeseer.nj.nec.com/burrows90scope.html, Burrows, Abadi, & Needham.

12. *Secure Network Time Protocol (stime)*, http://www.ietf.org/html.charters/stime-charter.html, Tim Polk & Patrick Cain.

13. *Security Engineering, The Guide to Building Dependable Distributed Systems*, Ross Anderson, Wiley Press, © 2001, pages 83-84.

14. *Write Protecting the DiskOnChip by Software,* http://www.m-sys.com/files/Appnotes/doc/App_Note_011_Rev_2.0.pdf, M-Systems Inc.

Last Updated: September 24th, 2017

Upcoming SANS Training

Click Here for a full list of all Upcoming SANS Events by Location

Rocky Mountain Fall 2017	**Denver, COUS**	**Sep 25, 2017 - Sep 30, 2017**	**Live Event**
SANS Baltimore Fall 2017	**Baltimore, MDUS**	**Sep 25, 2017 - Sep 30, 2017**	**Live Event**
Data Breach Summit & Training	**Chicago, ILUS**	**Sep 25, 2017 - Oct 02, 2017**	**Live Event**
SANS Copenhagen 2017	**Copenhagen, DK**	**Sep 25, 2017 - Sep 30, 2017**	**Live Event**
SANS London September 2017	**London, GB**	**Sep 25, 2017 - Sep 30, 2017**	**Live Event**
SANS Oslo Autumn 2017	**Oslo, NO**	**Oct 02, 2017 - Oct 07, 2017**	**Live Event**
SANS DFIR Prague Summit & Training 2017	**Prague, CZ**	**Oct 02, 2017 - Oct 08, 2017**	**Live Event**
SANS Phoenix-Mesa 2017	**Mesa, AZUS**	**Oct 09, 2017 - Oct 14, 2017**	**Live Event**
SANS October Singapore 2017	**Singapore, SG**	**Oct 09, 2017 - Oct 28, 2017**	**Live Event**
Secure DevOps Summit & Training	**Denver, COUS**	**Oct 10, 2017 - Oct 17, 2017**	**Live Event**
SANS Tysons Corner Fall 2017	**McLean, VAUS**	**Oct 14, 2017 - Oct 21, 2017**	**Live Event**
SANS Brussels Autumn 2017	**Brussels, BE**	**Oct 16, 2017 - Oct 21, 2017**	**Live Event**
SANS Tokyo Autumn 2017	**Tokyo, JP**	**Oct 16, 2017 - Oct 28, 2017**	**Live Event**
SANS Berlin 2017	**Berlin, DE**	**Oct 23, 2017 - Oct 28, 2017**	**Live Event**
SANS Seattle 2017	**Seattle, WAUS**	**Oct 30, 2017 - Nov 04, 2017**	**Live Event**
SANS San Diego 2017	**San Diego, CAUS**	**Oct 30, 2017 - Nov 04, 2017**	**Live Event**
SANS Gulf Region 2017	**Dubai, AE**	**Nov 04, 2017 - Nov 16, 2017**	**Live Event**
SANS Miami 2017	**Miami, FLUS**	**Nov 06, 2017 - Nov 11, 2017**	**Live Event**
SANS Milan November 2017	**Milan, IT**	**Nov 06, 2017 - Nov 11, 2017**	**Live Event**
SANS Amsterdam 2017	**Amsterdam, NL**	**Nov 06, 2017 - Nov 11, 2017**	**Live Event**
SANS Paris November 2017	**Paris, FR**	**Nov 13, 2017 - Nov 18, 2017**	**Live Event**
Pen Test Hackfest Summit & Training 2017	**Bethesda, MDUS**	**Nov 13, 2017 - Nov 20, 2017**	**Live Event**
SANS Sydney 2017	**Sydney, AU**	**Nov 13, 2017 - Nov 25, 2017**	**Live Event**
SANS London November 2017	**London, GB**	**Nov 27, 2017 - Dec 02, 2017**	**Live Event**
SANS San Francisco Winter 2017	**San Francisco, CAUS**	**Nov 27, 2017 - Dec 02, 2017**	**Live Event**
SIEM & Tactical Analytics Summit & Training	**Scottsdale, AZUS**	**Nov 28, 2017 - Dec 05, 2017**	**Live Event**
SANS Khobar 2017	**Khobar, SA**	**Dec 02, 2017 - Dec 07, 2017**	**Live Event**
SANS Munich December 2017	**Munich, DE**	**Dec 04, 2017 - Dec 09, 2017**	**Live Event**
European Security Awareness Summit & Training 2017	**London, GB**	**Dec 04, 2017 - Dec 07, 2017**	**Live Event**
SANS Austin Winter 2017	**Austin, TXUS**	**Dec 04, 2017 - Dec 09, 2017**	**Live Event**
SANS Frankfurt 2017	**Frankfurt, DE**	**Dec 11, 2017 - Dec 16, 2017**	**Live Event**
SANS Bangalore 2017	**Bangalore, IN**	**Dec 11, 2017 - Dec 16, 2017**	**Live Event**
SANS SEC504 at Cyber Security Week 2017	**OnlineNL**	**Sep 25, 2017 - Sep 30, 2017**	**Live Event**
SANS OnDemand	**Books & MP3s OnlyUS**	**Anytime**	**Self Paced**