Applied Encryption: Ensuring Integrity of Tactical Data

Jennifer Skalski-Pay

Applied Encryption: Ensuring Integrity of Tactical Data

Jennifer Skalski-Pay
GSEC Version 1.4b
January 13, 2003

Abstract

"The Common Operational Picture (COP) has been defined as an electronic picture of the battlefield, depicting the disposition of own and enemy forces, combining a detailed "little picture" view fed by organic (tactical) sensors and a "big picture" view fed by national sensors."[1] The ability of our forces to view an identical electronic battlefield from workstation to workstation cannot be overstated. These operational views are shared as well as updated by all forces within the arena and, therefore, must be synchronized. This process is accomplished through the use of COP Synchronization Tools (CST) software. CST allows for the exchange of track data between participating CST nodes. It uses either Transmission Control Protocol/Internet Protocol (TCP/IP) or Universal Datagram Protocol/Internet Protocol (UDP/IP). This exchange of data is near real-time and accomplished via local area networks (LANs) using importer/exporter (IE) objects and wide area networks (WANs) using CST interfaces.

Currently, there is a security risk involved with the transference of data through the CST software. While the SECRET Internet Protocol Router Network (SIPRNet) provides a circuit encryption to all data traveling along its path, there is no encryption applied directly through CST. If the data leaving CST were to be intercepted by unauthorized personnel on the local network, this would expose the confidentiality of the data and could potentially destroy its integrity. This paper will provide the reader with a low-level understanding of the Global Command and Control System-Maritime (GCCS-M), CST, Track Database Manager (Tdbm) and SIPRNet. It will detail how data transmission is accomplished from server to server via CST highlighting the need for additional encryption of the CST data stream.

Encryption

Encryption is the process of changing plain text into cipher text through the application of a mathematical formula (encryption algorithm) and a secret value (encryption key). Plaintext refers to information in its readable form before it has been encrypted and ciphertext refers to the unintelligible sequence of characters output by the encryption process. An encryption algorithm is a set of steps applied to data to produce the transformation from plaintext into ciphertext. An

[1] Hiniker

encryption key is used by the encryption algorithm to encrypt and decrypt the data. "The encryption algorithm mathematically applies the key, which is usually a long string of numbers, to the information being encrypted or decrypted."[2]

There are four main goals of encryption: confidentiality, non-repudiation, authentication and integrity. Confidentiality refers to keeping information from being disclosed to unauthorized personnel. Non-repudiation is the ability to prove the source of the information, i.e the sender of the data cannot be denied. Authentication confirms the identity of the person or computer from where the data originated. Both non-repudiation and authentication can be accomplished through the use of digital signatures. Integrity is concerned with the accuracy of the information. The idea is to determine if the data was maliciously or inadvertently altered.

The history of encryption predates the birth of Christ. "Around 2000 B.C., the Egyptians used modified hieroglyphs carved into stone as funeral messages not necessarily to keep messages secret but to increase their mystery."[3] There is evidence of encryption in ancient Greece by Spartan generals and also in ancient Rome by Julius Caesar. In Sparta, the generals would wrap a cylindrical staff or "scytale"[4] with thin strips of parchment paper and print a message on it vertically. The only way to decode the message was to have a diametrically identical staff. Caesar used a rotational cipher (ROT-3) to encrypt messages to his generals. ROT-3 is an alphabetical character shift of three spaces. For example, the word 'elephant' in ROT-3 becomes 'hohskdqw'.

Today, encryption is extremely more complex. Many times, the encryption algorithms are publicly known with the strength of the encryption scheme directly proportional to the length of the encryption key. With the introduction of Elliptic Curve Crytography (ECC), key length is no longer a factor in determining encryption strength. Even with this improvement in technology there still exists a significant performance penalty associated with encryption. Most often, it is the time intensiveness of exchanging the encryption keys.

Global Command and Control System - Maritime

GCCS-M is the command, control, communications, computers and intelligence (C4I) capability in use by the U.S. Navy. It consists of Unix servers, Unix and Windows 2000 workstations, encryption devices, hubs, and routers. WAN connectivity is accomplished through the utilization of the SIPRNet, Non-Classified Internet Protocol Router Network (NIPRNet) and the Joint Worldwide Intelligence Communication System (JWICS). This architecture offers a network-

[2] Russell, p.170.
[3] Russell, p.166.
[4] Garfinkel, p.139.

centric environment in which U.S. and allied maritime forces can coordinate efforts.

> GCCS-M is the Navy's single command and control program-of-record that integrates and interfaces over 80 separate C4I systems providing naval commanders afloat and ashore a near-real-time COP. GCCS-M enhances the operational commander's warfighting capability and aids in the decision-making process by receiving, retrieving, and displaying information to allow warfighters to plan, coordinate, exercise, execute and evaluate naval and joint operations. It uses IP routing over the SIPRNet and is currently deployed on over 300 ships and submarines, 57 ashore sites and 30 tactical variants. Expeditionary Units are also using GCCS-M.[5]

A visual representation of GCCS-M illustrates the usefulness and necessity of such a tool and is the easiest way to gain an understanding of this product. This visualization is the operational picture and is commonly referred to as 'Chart'. It is an invaluable decision aid to have in time of peace or war. The Chart uses a mapping engine to detail the world with terrain shading, terrain contours, bottom contours of the oceans, and much more. With the appropriate data loaded, the resolution can be as specific as rivers, bridges and roads. These features are rendered as part of the background mapping. In the foreground, the various tracks are plotted, i.e. ships, submarines, sites (buildings), satellites, air traffic, sonobuoys, etc. These objects are tracked through time constantly updating the view of the COP.

To put this into perspective, the background map is like a canvas with details that can be selected and deselected. On top of this canvas, foreground objects are rendered and these can be mobile or stationary. Viewing the same area of the world over time, the background will remain constant while the foreground will represent real-world movement of objects. Figure 1 is a lab simulation snapshot taken off the coast of southern California. The blue curves in the background represent different depths of the Pacific Ocean. The three tracks in the foreground are all moving in the same direction and given time will not be visible within this window.

The Chart also provides tactical decision aids (TDAs). These add the ability to make various calculations based upon an object's attributes, e.g. position, course, speed, altitude, etc. Some of these TDAs are also used to plot additional foreground objects on the map which may be unique to a particular workstation or shared with other workstations within the LAN or to other subscribers on the WAN.

[5] Wester.

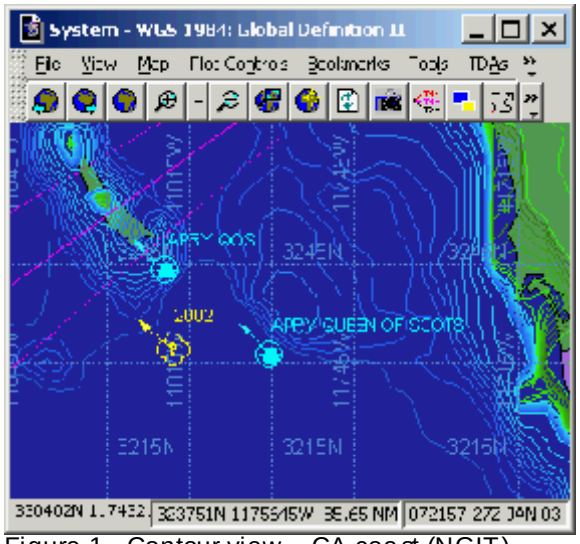

Figure 1. Contour view -- CA coast (NGIT)

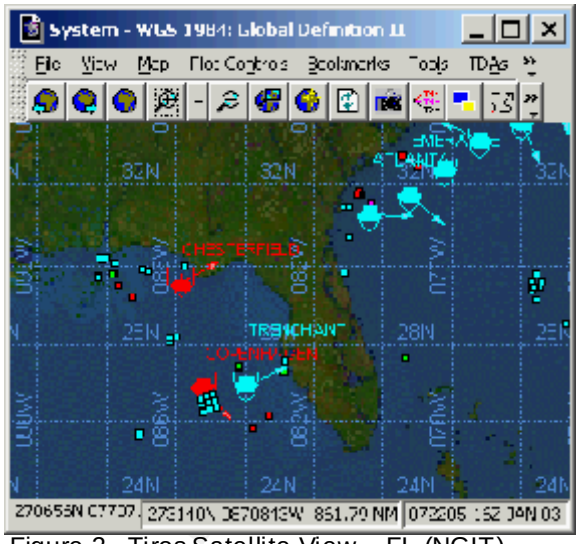

Figure 2. Tiros Satellite View – FL (NGIT)

Track Database Manager

Tdbm is a database manager for tactical track data as well as a provider for multi-source correlation of this data. If you ask ten people familiar with Tdbm what the acronym breakdown is you'll most likely get a 50/50 split on the responses. 'Track Database Manager' and 'Tactical Database Manager' are commonly used to refer to the same piece of software. Both are self-descriptive terms and either one is appropriate.

The Tdbm architecture is best described as a single master (server) with many slave (client) relationships. Operating within a LAN, the Tdbm master manages the synchronization of all database activities. The designated server is running a master Tdbm process while each client is running a slave Tdbm process. A copy of the track database is kept in memory on each client machine and data queries are performed locally through application programmer interfaces (APIs). Thus, increasing performance by removing the overhead associated with data transfer between server and client. The master server updates its database when any track updates and modifications occur on a client workstation. Consequently, all other client workstations within the LAN will reflect these same modifications as the databases are synced.

A track is defined as any object whose existence and/or movement has been reported or tracked. Tracks consist of ships, submarines, aircraft, land units and other objects (mobile or immobile) within the theater of operation. Tracks have associated attributes and positional histories. Figure 2 is a lab simulated Tiros satellite view of numerous tracks in the Gulf of Mexico and along the Atlantic coast of United States. The fully visible symbols on the map represent submarines of varying affiliation. The colored 'dots' on the map are other tracks that have been rendered this way to unclutter the current map view. They are

valid tracks with sufficient attribute data to be plotted as fully detailed symbols. Tracks of no interest can also be filtered out meaning they will not be plotted on the map.

Specific attributes of each track are represented by the color, shape and directional indicator of each symbol. Track colors indicate the threat level or affiliation of an object as follows: yellow (unknown or pending), red (hostile or suspect), blue (friendly or assumed friendly), green (neutral), and magenta (ambiguous). An ambiguity implies an inability to identify a track due to a lack of attribute data in the report. As additional reports are received and correlated on the ambiguity, the identity of the track will be established and the symbol correctly updated to reflect its affiliation. Another attribute of a track that is of great importance is its position. The symbol's position on the map is the last reported position of the object being tracked. Depending on the time of the report, this visualization could be misleading. If a client workstation is visualizing Tdbm data while in dead reckoning (DR) mode, then the symbols are moving on the map according to the last reported position, speed and direction relative to the current time.

These type of calculations and visualizations are all a functionality of 'Chart'. They are described here in some detail because none of these would be possible without the information provided through Tdbm.

SECRET Internet Protocol Router Network

Within the Department of Defense there is a need to distribute SECRET data. This is accomplished via the SIPRNet. The SIPRNet is restricted to individuals with at least a U.S. government clearance of SECRET and is vital to our warfighting capabilities by providing a protected network through which intelligence data can be exchanged.

> The SIPRNet is a closed loop system, meaning that it is completely separated from all other computer systems. Each access circuit and backbone trunk is encrypted using military grade hardware encryption to ensure integrity of information. This means the data signal is encrypted after leaving the computer by a separate machine loaded with a cryptographic key or black box. These keys are high level algorithms that change sometimes as often as daily, and in some situations hourly.[6]

[6] No-Mad.

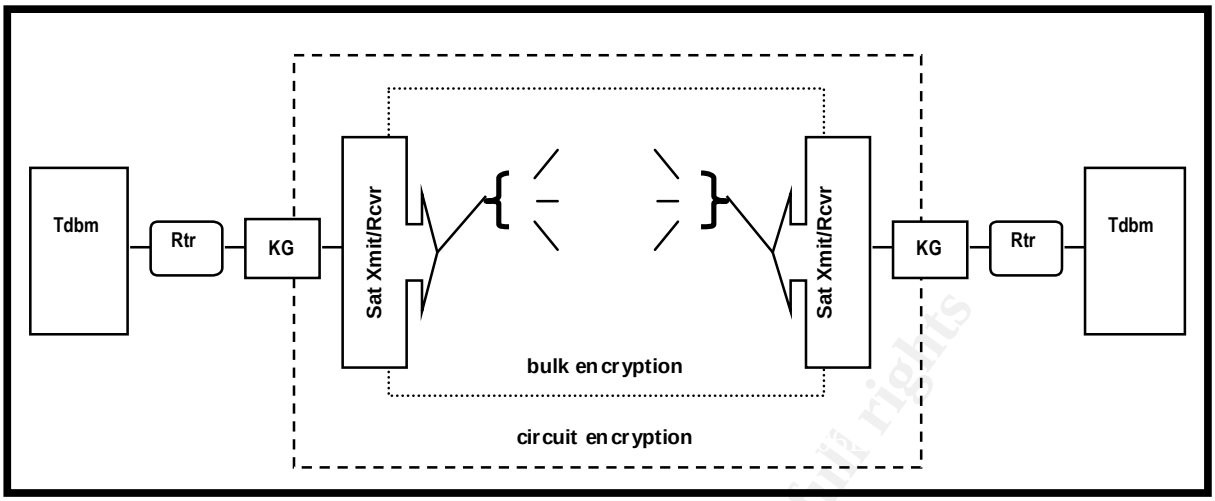

Figure 3. Data exchange between Tdbm servers across a WAN.

Figure 3 depicts a possible scenario for data transmission between two Tdbm masters. The router (Rtr) is the point at which the DoD subscriber connects to the SIPRNet and the key generator (KG) is the black box responsible for the circuit encryption of data over the SIPRNet. Because there is a satellite to satellite data transmission, there will be an additional bulk encryption of the data as it goes over the air waves. KG-81, KG-84A, KG-84C, KG-94, KG-94A, KG-194, KG-194A, KG-95-1, KG-95-2, and KIV-19 are all encryption devices used by the military and examples of possible key generators.

"The initial implementation of SIPRNet went on-line in March 1994 and by May 1995 consisted of a collection of 31 backbone routers interconnected by high-speed serial links to serve the long-haul data transport needs of secret-level DoD subscribers. Since then, it has matured to be the core of our warfighting command and control capability."[7] The SIPRNet can be compared to the regular Internet, but on a much, much smaller scale. It contains search engines, web servers, news groups and email. Due to the fact that this network is not intended for the average consumer, the available data is sensitive and defense related. There are web sites for CIA, FBI, DOE and other government entities. What is not there are sites such as entertainment, shopping, finance, etc. It can be likened to a health food store where the merchandise is extremely specialized.

Common Operational Picture (COP) Synchronization Tool

CST is the component of GCCS-M responsible for synchronizing the operational picture across all battlefields. It does this by providing an automated method of transferring data. CST uses interfaces to rapidly transmit binary data to all participating nodes updating the COP across the WAN within a few seconds.

[7] Pike.

These interfaces are designed without the knowledge of the type of data being sent or received. They are attached to incoming and outgoing queues that are periodically polled to check for new data. This architecture is an improvement over the periodic broadcasts of network channels where messages are sent at the discretion of the network or satellite communications resulting in track databases across the WAN being updated in an untimely fashion or not at all.

CST provides four different interfaces to exchange data between nodes. These interface types include CSTTCP, CSTMCAST, CSTMDPv2 and CSTMDP1WY. CSTTCP utilizes TCP/IP requiring a distinct connection to be established between two directly connected nodes. CSTMCAST, CSTMDPv2 and CSTMDP1WY are multicast interfaces. These generally use UDP/IP and are more efficient than the CSTTCP interface because they allow for a single transmission from one node to be sent to all nodes within the CST network. CSTTCP requires data to be passed point-to-point between each node. The difference between these multicast interfaces is in the reliability factor. CSTMDPv2 and CSTMDP1WY have specifically implemented an error-correcting protocol to ensure correct receipt of the transferred data. CSTMDP1WY is different in that data only flows in one direction. Thus, only the master node sends data and all other nodes not configured as the "master" receive data.

The focus of this paper will be the use of the CSTTCP interface as it applies to the transmission of track data. This interface allows for a maximum of five child nodes per parent node. The master node or "Top COP" is responsible for configuring the data of each of its child nodes. Any child of the master node possessing secondary configuration privilege will then configure the data of its own children. This data flow continues down the tree structure until all nodes have been synchronized. A child node will also update its parent. This update will continue up the tree to the master node where it will be sent to all other nodes in the tree via the parent/child connections. Figure 4 shows the relationship of the master nodes to all other nodes for the CSTTCP interface.

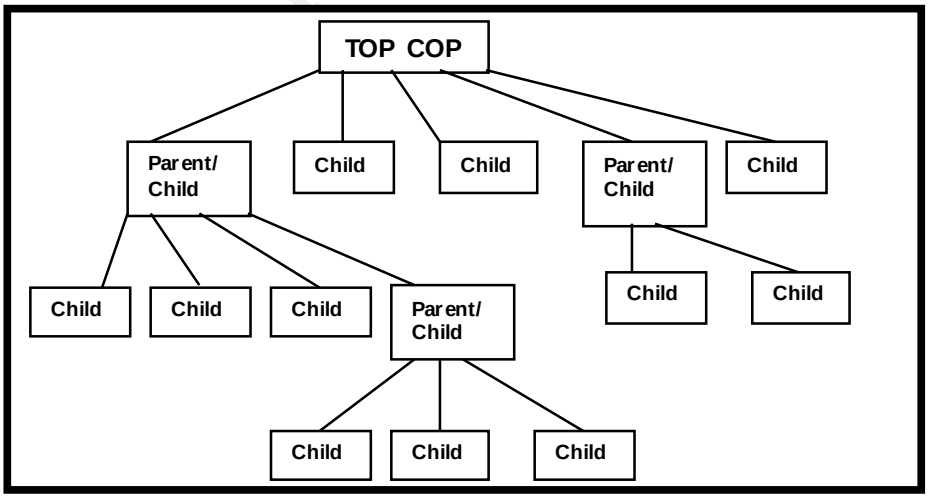

Figure 4. Tree-like structure indicative of the CSTTCP interface.

When tracks are to be sent, the track data is compressed, put into a COP message format and queued to the CST track encoder. The three queues associated with sending track data are normal, high priority and low priority. In the normal queue, there is one slot per Tdbm track record and only the most recent track update is kept in the queue. The high priority queue is for high priority data, e.g. missile tracks. Data in this queue is transmitted before data in the normal queue. The low priority queue data is typically used for historical track data and is only sent if there is no data in the other two track queues. When tracks are received, they arrive on the incoming socket and are queued to the CST track decoder. All incoming track data is compared to the related track data currently in Tdbm. If the information is duplicated, then the data is discarded minimizing the amount of processing for Tdbm.

CST is selective in the track data it forwards to other participating nodes. It will not forward tracks classified as terminal (specific to a single workstation), local (specific to workstations within a LAN) or ambiguous. Platform, Unit, Elint, Acoustic, Link, Missile, Space and General are all Tdbm track types currently supported in CST.

Securing the CST Data Stream

Track data being transmitted between Tdbm servers through the CSTTCP interface is in the form of a binary data stream. This data will travel for a short distance on a LAN before it is routed to the SIPRNet. In the blink of an eye, there is opportunity. This instant in time determines the need for an additional layer of protection. The slightest modification to a track's attributes by an ill intending person can produce grave results. By enciphering this data stream, unauthorized track changes become significantly more difficult.

When selecting an encryption scheme, a few considerations are performance, software cost and ease of implementation. For these reasons, the late December 2002 release of OpenSSL v0.9.7 is the target API set. It is open source meaning it is free to use and applies no restrictions to its implementation. "OpenSSL is a cryptographic library of the industry's best-regarded algorithms, including encryption algorithms, message digest algorithms and message authentication codes (MACs)."[8] It is ideal for CST's network communication needs as it implements the Secure Socket Layer (SSL) protocol and the Transport Layer Security (TLS) protocol.

Our goals are obvious: data authenticity, integrity, confidentiality and, if possible, non-repudiation. Up front, OpenSSL doesn't provide any form of accountability. A symmetric cipher and a one-way hash function (or MAC) will fulfill our

[8] Viega, p.1.

confidentiality and integrity requirements, respectively. Our two remaining objectives can be satisfied by applying a digital signature.

The most widely used symmetric ciphers come in two types: block and stream. A block cipher encrypts a predefined block size of data at one time. It breaks data into these block sizes padding the final block if necessary. Block ciphers encrypt each block of the current data stream in the same manner. Thus, potentially revealing patterns and creating vulnerabilities. Stream ciphers are faster than any available block cipher. They take a randomly generated stream of bits and using the bitwise XOR function with the plaintext data stream create an encrypted stream. There is typically no padding associated with stream ciphers. If any was required, it would be on the order of bits, not bytes. Given this, a stream cipher will be a suitable solution.

The chosen stream cipher is RC4. This cipher was originally a proprietary algorithm of RSA Security until it was anonymously reverse engineered and posted on the internet in 1994. It uses "variable-length keys that can be up to 256 bytes long."[9] OpenSSL v0.9.7 provides two algorithms for RC4 encryption. One uses an insecure key length of 40 bits, the other a 128-bit key length. The choice is obvious. Please note there are known weaknesses in the key scheduling algorithm for RC4. These "can be prevented by discarding the first 256 output bytes of the pseudo-random generator before beginning encryption."[10]

Our source for routines is housed within OpenSSL's EVP interface. This API set will provide all of the functions necessary to encrypt data, decrypt data and generate keys. The first step in the encryption process is initialization of the cipher context. This "is a data structure that keeps track of all relevant state for the purposes of encrypting or decrypting data over a period of time."[11] This context will be used to associate the key stream to the enciphered text. Separate contexts will be created for the encryption and decryption of data.

There are four parameters passed to the 'encryption' and 'decryption' cipher context objects: generic cipher context object, cipher type, encryption key and initialization vector (IV). The IV used to initialize both context objects per data stream must be identical. The key and vector are randomly generated using functions in the OpenSSL's RAND package. A vital step in this process is selecting the data to seed the random number generator as a poorly derived seed can lead to predictable output at all stages.

There are two remaining steps to the encryption process: update and finalize. The actual data encryption occurs here. Among the parameters to these functions are the buffer containing the plain text, the cipher context created during the initialization step and the buffer for the resulting encrypted data. As

[9] Viega, p.178.
[10] Rivest.
[11] Viega, p.179.

applied to stream ciphering, there will be only one update and one final call. When block ciphering, the roles of these functions is more apparent. Each round of update encrypts a block of data. Update rounds will continue until the length of the remaining data is equal to or less than the defined block size. At that time, finalize will be called to pad the last portion of plaintext up to the defined block size and the final round of encryption is completed.

Message authentication will be employed to transport the IV from sender to receiver. There is only one MAC implementation offered by OpenSSL, that is HMAC. The HMAC context functions are similar to those of the cipher context. These include initialize, update and finalize. Additionally, a call to the HMAC_cleanup function is required to free associated resources. One of the parameters to the HMAC context is a message digest or hashing algorithm. For this, the 160-bit SHA1 algorithm will be used. MAC initialization also calls for an encryption key. This key must be different from the one previously used to encrypt the plaintext data stream. To "validate the integrity of the MAC's data, simply recompute the hash value and compare it against the transmitted hash value. "[12] The output of these computations should be identical.

Key exchange is the next endeavor and is the most resource intensive part of the encryption process. This is especially true in this application being that key reuse is not recommended for successive stream encipherings. Given this scenario, keys must be exchanged for every encrypted stream sent over the wire. The sender and receiver will each randomly generate a key. Each key will be encrypted using the intended recipient's public key. Public and private key pairings are commutative. Encryption with a specific public key can be decrypted with the corresponding private key. After exchange of these keys, both parties will be in possession of two keys. These keys can then be combined in a predetermined way to obtain a secret key. To satisfy our need for authentication and accountability, these keys can also be digitally signed with the respective sender's private key. The signature can be verified using the respective sender's public key.

Decryption is handled in much the same way. The decryption cipher context is initialized using the original key, original vector and a newly created generic context. Following this, the update and final functions are called to convert the encrypted stream into plaintext.

Data Flow Summarization

The track data for the CST outgoing socket will be encrypted using a randomly generated key (keyA) and initialization vector (IV). The vector will be placed in a one-way hash providing for an additional form of authentication. A second randomly generated key (keyB) is required for the hash encryption algorithm.

[12] Viega, p.201.

KeyA and keyB will be computed from separate key values previously exchanged between sender and receiver. Next, three pieces of information will be transferred from sender to receiver: encrypted data, IV and MAC. Optionally, keyB can be used to validate the vector data in the MAC. Finally, the encrypted data stream, keyA and the IV will be passed to the decryption routines. The resulting unencrypted stream has been thoroughly protected and it's business as usual for CST.

Acronym List

API – Application Programmer Interface
C4I – Command, Control, Communications, Computers and Intelligence
CIA – Central Intelligence Agency
COP – Common Operational Picture
CST – COP Synchronization Tools
DoD – Department of Defense
DOE – Department of Energy
DR – Dead Reckoning
ECC – Elliptic Curve Cryptography
FBI – Federal Bureau of Investigation
GCCS-M – Global Command and Control System Maritime
IE – Importer/Exporter
IP – Internet Protocol
IV – Initialization Vector
JWICS – Joint Worldwide Intelligence Communication System
KG – Key Generator
LAN – Local Area Network
MAC – Message Authentication Code
NIPRNet – Non-Classified Internet Protocol Router Network
SIPRNet – SECRET Internet Protocol Router Network
SITREP – Situational Report
SSL – Secure Socket Layer
TCP – Transport Control Protocol
TDA – Tactical Decision Aid
Tdbm – Track Database Manager
TLS – Transport Layer Security
TMS – Tactical Management System
UDP – User Datagram Protocol
WAN – Wide Area Network

References

Garfinkel, Simson and Gene Spafford. Practical UNIX & Internet Security. Sebastopol: O'Reilly & Associates, Inc, 1996.

Hiniker, Paul Johnson. "The Common Operational Picture as Evolving Schema for Command Centers as Complex Adaptive Systems". URL: http://www.dodccrp.org/Proceedings/DOCS/wcd00000/wcd0006f.htm (17 December 2002).

No-Mad. "What we can learn from the SIPRNET". Collusion. Volume 7. January 2000. URL: http://www.collusion.org/Article.cfm?ID=140 (11 December 2002).

Pike, John. "Secret Internet Protocol Router Network (SIPRNET)". 03 March 2000. URL: http://www.fas.org/irp/program/disseminate/siprnet.htm (21 November 2002).

Rivest, Ron. "RSA Security Response to Weaknesses in Key Scheduling Algorithm of RC4". 2002. URL: http://www.rsasecurity.com/rsalabs/technotes/wep.html (12 January 2003).

Russell, Deborah and T.T. Gangemi Sr. Computer Security Basics. Sebastopol: O'Reilly & Associates, Inc, 1991.

Viega, John, Matt Messier and Pravir Chandra. Network Security with OpenSSL. Sebastopol: O'Reilly & Associates, Inc, 2002.

Wester, Tom. "Global Command & Control System – Maritime (GCCS-M)". 2001. URL: http://www.globalsecurity.org/intell/library/reports/2001/compendium/gccs-m.htm (21 November 2002).

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**