Department of Homeland Security: Control Systems Communications Encryption Primer

December 2009

Homeland Security

ABSTRACT

This primer addresses the use of encryption systems within control systems environments of the U.S. critical infrastructure. Control systems have operating parameters that differ significantly from traditional Information Technology systems. In control systems networks, security goals are normally prioritized in the following order: integrity (including safety), availability, and confidentiality. Proper application of encryption systems will assist control system security professionals to achieve these goals. However, applying encryption techniques to industrial control systems can introduce significant design challenges as they add complexities and operational limitations to the environment. If not implemented correctly, an encryption system will only provide an illusion of security and could introduce risks. The reader is informed of the basics of encryption systems (referred to herein as "cryptosystems") and then shown two specific implementations: Internet Protocol Security and Transport Layer Security/Secure Sockets Layer. Understanding these implementation examples will help the reader decide when encryption techniques could serve as an appropriate security control within the industrial control systems environment.

CONTENTS

FIGURES

ACRONYMS

AD	Active Directory
AH	Authentication Header
BitW	Bump-in-the-Wire
CA	Certificate Authority
CPU	Central Processing Unit
ESP	Encapsulated Security Payload
FEP	Front-end Processor
IDS	Intrusion Detection System
IKE	Internet Key Exchange
IP	Internet Protocol
IPSec	Internet Protocol Security
ISAKMP	Internet Security Association and Key Management Protocol
ISO	International Standards Organization
MAC	Message Authentication Code
NIST	National Institute of Standards and Technology
PIN	Personal Identification Number
PLC	Programmable Logic Controller
RAM	Random Access Memory
RTU	Remote Terminal Unit
SA	Security Association
SADB	Security Parameter Database
SCADA	Supervisory Control and Data Acquisition (systems)
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
VPN	Virtual Private Network
WAN	Wide Area Network

1. INTRODUCTION

By definition, an encryption system (herein referred to as a *cryptosystem*) is the combination of three elements: an *encryption engine*, *keying information*, and *operational procedures* for its secure use. In general, a cryptosystem refers to a suite of algorithms needed to implement a particular form of encryption and decryption technique. A cryptosystem consists of three algorithms: one for key generation, one for encryption, and one for decryption. Their application to industrial control systems may present design and operational challenges. This primer provides assistance to control systems security professionals to identify appropriate encryption techniques and determine whether to deploy a cryptosystem solution as a security feature in their specific control systems environment. This primer also presents examples of cryptosystem deployment solutions to assist users in identifying appropriate application for their specific system.

Cryptosystems have four intended goals:

- Confidentiality
- Integrity
- Authentication
- Nonrepudiation

Network traffic encryption has become a focus of information technology because of the inherent insecurity of the Internet. Information Technology's main focus is on server integrity and confidential client communications while often allowing that client's reputability. A control system will likely focus on the integrity of the client and nonrepudiation of communications, whereas confidentiality will be considered secondary. This primer discusses common encryption protocols and how each can be applied to control systems environments. The intent is to provide sufficient background information to enable security professionals to make informed decisions when planning a cryptosystem implementation for a control systems environment.

The following sections describe the four communication goals of an encryption system and provide background discussion on cryptosystem types and design considerations. Overall, this primer focuses on the two most common network cryptosystems and their appropriate application to control systems.

1.1 Confidentiality

Confidentiality is defined as keeping data unseen by others through preserving authorized restrictions on information access and disclosure, including means for protecting personal privacy and proprietary information. A loss of confidentiality is the unauthorized disclosure of information. Confidentiality, as it relates to security and encryption techniques can be obtained by encrypting messages such that only intended recipients are able to read them. For example, when a purchase transaction occurs between a consumer and secure online retailer, the consumer's financial and personal information should be encrypted to ensure that only that retailer has the ability to recover the credit card information. This ensures that a third party cannot intercept the credit card information in a form that is usable. In contrast, a control system may not need to ensure that every single data item is maintained as confidential; the vendor and asset owner will need to determine what does need to remain confidential.

Confidentiality is important when using encryption to wrap an insecure protocol because the underlying protocol may contain user names and passwords that can be used to compromise the system.

1.2 Integrity

Integrity is ensuring that the data presented are the true valid master source of the data and includes guarding against improper information modification or destruction to ensure information nonrepudiation and authenticity. A loss of integrity is the unauthorized modification, insertion, or destruction of information. For control systems, integrity describes the quality of the communications (transmission of correct data) and reflects the logical correctness and reliability of the operation of the system; the logical completeness of the hardware and software implementing the protection mechanisms; and the consistency of the data structures and occurrence of the stored data. Note that, in this document, integrity is interpreted to mean protection against unauthorized modification or destruction of information or messages sent between communicating parties on the control system.

One way of ensuring data integrity is by using simple checksums which prevent an attacker from forging or replaying messages. Checksums are usually implemented when the channel between communicating parties is not secure and ensure that the data has reached its destination with all bits intact, or, if bits have been modified, that the modification will not go unobserved.

Here are some examples:

- A command is sent to the application server. The application server needs to ensure the integrity of the message before acting on that command.
- Data from a Remote Terminal Unit (RTU) is sent to the Front-end Processor (FEP) or master terminal unit. The FEP needs to ensure the integrity of the RTU's message in order to prevent the system from acting on incorrect information.
- The batch program run by a Programmable Logic Controller (PLC) is downloaded from the PLC configuration server. The PLC must be able to ensure the batch program has been unaltered from the version designated by the FEP.

1.3 Authentication

Authentication is the technique of verifying the identity of a user, process, or device, often as a prerequisite to allowing access to resources in a control system. Common cryptographic techniques for authenticating identity include certificates and knowledge proofs. Integrity is heavily intertwined with authentication when implemented with encryption, but the distinction lies in ensuring messages are sent by the correct or declared party whose identity has been verified.

Some examples include:

- Data sent from field devices should be authenticated to ensure an attacker is not forging data.
- When an operator should authenticate to a system (using a password, thumbprint, and/or retinal scan) before being granted access to the operator console.

1.4 Nonrepudiation

Nonrepudiation is ensuring that a traceable legal record is kept and has not been changed by a malicious entity. A loss of nonrepudiation would result in the questioning of the transactions that have occurred. Ensuring nonrepudiation within control system networks provides the capability to determine whether a given individual took a particular action such as creating information, sending a message, approving information and receiving a message. A simple example of nonrepudiation is signing a contract. The signers cannot claim they did not agree to a contract because there is evidence that they did agree. The difference is that a signature can be forged but good encryption cannot. The following is an example of how the concept of nonrepudiation can be useful for control systems:

- Utility A and Utility B agree to share some information. Utility A provides information that Utility B uses to make decisions. At a later point, that information proves to be false. In case Utility A denies sending the false information, Utility B wants to be able to prove that Utility A did send the information.

Nonrepudiation is outlined for message-handling systems in International Telecommunication Union Telecommunication recommendation F.400/X.400. Five types of nonrepudiation are outlined:

- Content Received
- Delivery
- Internet Protocol (IP) notification
- Origin
- Submission.

Simple cryptographic message signing provides for nonrepudiation of origin and content received. A cryptographically signed reply to the message would provide for nonrepudiation of delivery and IP notification. Nonrepudiation of submission applies to messaging systems that have an intermediate handler and can be met by cryptographic signing of a receipt message.

1.5 Cryptosystem Types

A large number of encryption algorithms are well established and new ones are being developed and tested each day. Symmetric and asymmetric algorithms form two of the basic classes. The choice between these classes changes the type of infrastructure that must be designed to support encryption on the system.

1.5.1 Symmetric versus Asymmetric Keys

All encryption is based on the use of large key values that should not discernable from the messages they are designed to protect. When using symmetric keys, all communicating parties use only one key to both encrypt and decrypt a message. When using asymmetric keys, parties use one key to encrypt and a different key to decrypt a message. Asymmetric keys should not be derivable from each other. Asymmetric encryption is synonymous with public-key encryption; the public key is the encryption key and the private key is the decryption key.

Asymmetric encryption provides a better basis for establishing nonrepudiation capabilities. The signature required for nonrepudiation can be the private key, which can be verified using the public key. If a symmetric key is used to sign a message, the symmetric key is also needed to verify the signature, which means that the entity that is verifying the signature has the ability to forge a signature. Despite this disadvantage, techniques do exist to use a symmetric key system to provide effective nonrepudiation.

Symmetric keys generally provide faster encryption than asymmetric keys. Hybrid encryption uses asymmetric keys to establish identity and share a new temporary symmetric key with each participant. The symmetric key is then used for the bulk of the messaging, which allows for a technique that incorporates most of the advantages of both cryptographic techniques.

1.5.2 Pre-shared versus Created Keys

For an encrypted conversation to occur, the participants involved must have the correct key information. Key infrastructure is the set of keys, servers, policies, and procedures used to ensure that each participant has the keys necessary to engage in the conversation. In a simple form, key infrastructure can refer to a person copying keys from one computer to another using a Universal Serial Bus key. This example demonstrates that the key is secure but not scalable, which refers to the ability to expand a system to handle larger amounts of work. Scalable cryptosystems are based on trusted systems that can be used to distribute additional keys as necessary.

The basis for any cryptosystem is a set of human-verified keys; these can be either symmetric or asymmetric keys. In the case of symmetric keys, all participants receive the same set of keys. In the case of asymmetric keys, each participant will have a unique set of keys consisting of their private key and all other participants' public keys. If all keys for every possible host are included in this initial set of keys, the cryptographic infrastructure is called pre-shared keying.

In a dynamic environment where new hosts are brought online and decommissioned hosts are revoked, pre-shared keying is impractical as a final solution because the overhead would be too large. Instead, the pre-shared keys are used to allow encrypted communication with a trusted key server. This key server maintains the necessary set of keys to manage encrypted communications between all participants. Another name for the key server is Certificate Authority (CA) when using asymmetric keys, or a Key Distribution Center in the case of symmetric keys.

Along with the actual distribution of keys, methods to create and revoke keys must be devised and tracked. Keys that meet the requirements of the chosen policy must be created, and revoking keys is important in rendering compromised keys useless. If the policy pertaining to the creation of keys changes, the cryptosystem needs to be able to revoke the current set of keys and distribute a new set of keys that meet the new policy. Federal Information Processing Standards Publication 140-2 discusses key management recommendations in detail.

1.5.3 Opportunistic versus Selective Encryption

Encryption can be used either opportunistically, meaning that everything is secured, or selectively, meaning that only certain parts of a communication are secured. Opportunistic encryption is generally the easiest way to retrofit an established protocol. The protocol can provide cryptographic security by simply being wrapped with Secure Sockets Layer (SSL) or Internet Protocol Security (IPSec). More granular control over the parts of the communication that need to be cryptographically secure can be achieved using Kerberos or another similar protocol. Selective encryption requires that the integrator understand which information is sensitive and explicitly secure those fields. Detailed descriptions of selective encryption schemes are outside the scope of this document.

1.5.4 Proprietary versus Conventional Algorithms

When implementing encryption on the control system, the most important thing to avoid is writing your own proprietary cryptographic algorithms. Cryptographic algorithms that have withstood repeated and in depth examinations can provide better assurance. Many conventionally used algorithms are available to implement encryption, even on control systems. Because of the complexity of cryptology, a secure implementation requires many people contributing and analyzing the techniques and approach. Robust cryptographic algorithms and protocols are designed and tested to ensure security, even with full knowledge of the function of the algorithm. The achievement of full-system security requires an in-depth knowledge of how encryption techniques will impact the control system performance; because a system's exposure to unnecessary risk occurs when small details are overlooked.

1.6 Design Considerations

The unfavorable impacts of encryption on the design of control systems environments raise a number of concerns. The four most common concerns are latency, bandwidth, availability, and interaction with an Intrusion Detection System (IDS). With proper planning and system design, these impacts can be avoided or reduced. Readers should remember that latency, bandwidth, and throughput must all be actively managed to ensure that the network delivers quality service.

1.6.1 Latency

Encryption introduces additional computational load at both ends of a communication. The sender must compute the cipher or signature, while the receiver must decrypt the cipher or verify the signature.

The additional time for these computations on modern computers is generally not sufficient to cause noticeable latency, but a number of variables are important to consider. The first is the algorithms used for the encryption. Different algorithms use different amounts of a Central Processing Unit's (CPU's) time, and some algorithms are specifically designed to be fast. If the algorithm needed for the system cannot be performed fast enough to prevent latency, additional cryptographic hardware can be added to the system to offload the encryption process and reduce latency. Encryption and decryption both exhibit low jitter, creating an effect on latency that should be bounded (close to constant and measurable) and most applications can compensate for this latency in input and output. If latency issues do appear, they can be overcome using other techniques.

Additionally, encryption can increase the size of packets, and cause a higher packet fragmentation at routing and switching nodes which would also increase latency. Routers and switches should be equipped with an amount of Random Access Memory (RAM) to accommodate the additional overhead that encryption imposes.

As a rule of thumb, links across Wide Area Network (WAN) connections as well as the Internet need to be managed to maintain ping times of 200 milliseconds or less to ensure acceptable latency management.

1.6.2 Bandwidth

To give a rough estimate of performance impact, an Intel Pentium D (2.8 GHz) is capable of performing 1.6 Gbit/sec of SHA1 (Secure Hash Algorithm 1, a common hash algorithm used for authentication), but only capable of 624 Mbit/sec of AES-128-CBC (Advanced Encryption Standard, Cipher Block Chaining, a common encryption algorithm). The numbers shown reflect raw performance. Network bandwidth is assumed sufficient to handle the additional overhead and that the servers are either disk input/output bound or idle, pending input. This load reflects Supervisory Control and Data Acquisition systems (SCADA), but does not necessarily reflect process control systems.

Some systems, particularly those using serial communications (copper or radio links) are bandwidth constrained. Encryption does have overhead, but when developing a protocol, various tradeoffs can be weighed to meet the bandwidth constraint: cipher choice (stream vs. block), number of bits used in the message authentication code, etc.

1.6.3 Availability

Availability is defined as providing the data when needed or "ensuring timely and reliable access to and use of information. Control systems are usually built with redundancy to ensure availability to operator personnel. Encryption on control systems should be built-in with the same design goals. If the system is designed with two layers of redundancy, the cryptosystem should also have two redundant key servers. As with any availability design, each additional component introduces another potential failure point. Careful design and implementation will allow encryption to be used with minimal impact on availability. Application of encryption can even improve availability by preventing unintended communications from impacting a system.

1.6.4 Algorithm

Many cryptosystems support multiple algorithms that can be used to protect information. The choice of the algorithm becomes very important for meeting the requirements of a particular protocol. An isochronous protocol will require a constant time algorithm to ensure the time that the cryptographic algorithm needs can be factored into the timing needs of the protocol. Different protocols will need different algorithms to meet their specifications, and the number of pairings are too numerous to list in this document.

1.6.5 Interaction with Intrusion Detection Systems (IDS)

Encryption can impact IDS performance by inhibiting the IDS from fully inspecting communications. This undesirable situation occurs when the IDS attempts to examine the content of encrypted messages. The IDS may still be effective using one of the following solutions:

- Allow the IDS to access the keys necessary to decrypt messages and configure the IDS to perform decryption of the content. This solution will still create latency in the IDS's ability to detect attacks.
- Do not use the confidentiality component of the cryptosystem.
- Use an IDS that does not require content inspection, such as a flow analyzer that looks for discrepancy in standard flow patterns.
- Use host-based IDSs that perform operations after the encryption has been completed.

Allowing the IDS access to the keys makes it a highly valuable target. If the IDS can be compromised, the entire cryptographic infrastructure is compromised. Disabling the use of the confidentiality component may not be appropriate in all environments. This decision is strictly application specific.

Flow analysis IDS does not have the ability to catch packet-level variations of data. Instead, flow analysis can only examine the source, destination, time, and quantity of data sent between endpoints. Although flow analysis is a valid method of intrusion detection, this technique is also far more heuristic than packet-based intrusion detection.

Finally, host-based IDS can process the data on the receiving host, post encryption, but it will impose a load on the host (be it server, RTU, etc.) The additional load may be unacceptably high in some applications.

1.7 Choosing a Cryptosystem

The two most common network cryptosystems are IPSec and SSL. The primary difference between IPSec and SSL is the logical level at which they operate. IPSec is a host-level cryptographic solution, which means that all network traffic between two hosts will be cryptographically secured with no cognizance of the traffic's contents—the encryption is transparent to all host applications. SSL is an application-level solution, which means that traffic is cryptographically secured on a single application between the hosts. Logic level differences between IPSec and SSL lead to the secondary difference between the two systems: applications using IPSec require only host configurations in order to be implemented. Each application must be modified to use SSL. The overhead of configuration is similar for single applications, but when many applications require distinct configuration for SSL, the distinction becomes clearer.

Configuration IPSec on a network is historically difficult to properly implement, especially if the network is comprised of mixed operating systems. Unlike IPSec, SSL is much easier to configure, but configuration must be built into the application and cannot be easily added postproduction.

Finally, when an SSL system is configured to comply with Public Key Infrastructure, X.509[a], SSL supports the Privilege Management Infrastructure, which is used to manage user privileges based on identity. Each user is then given a unique key, and can therefore use cryptographically secured applications with individual permissions. Unique user keys can be key features in the control systems environment where many users share a single host.

---

[a]. R. Housley, W. Ford, W. Polk, D. Solo, "Internet X.509 Public Key Infrastructure Certificate and CRL Profile," RFC 2459, http://www.faqs.org/rfcs/rfc2459.html, January 1999.

2. INTERNET PROTOCOL SECURITY

Internet Protocol Security (IPSec) is a suite of protocols designed to secure node-to-node communications. Four main security properties are provided by IPSec: confidentiality (Section 1.1), integrity (Section 1.2), authenticity (Section 1.3), and replay protection. Replay protection means that data sent once can only be used once and not replayed at some point in the future.

Primarily, IPSec consists of two protocols: Authentication Header (AH) and Encapsulated Security Payload (ESP)—IPs 51 and 50, respectively. AH provides integrity, authenticity, and optional replay protection; ESP adds confidentiality and optional replay protection.

Figure 1 shows a normal Transmission Control Protocol (TCP)/IP datagram encapsulated in an Ethernet frame. This datagram is not secured in any way; an attacker between the sender and receiver can drop, replace, modify, or replay this datagram.

Figure 1. Normal IP datagram.

In operation at the IP protocol layer, IPSec has no awareness of the application or even of the data that it is securing. IPSec acts to secure packets sent between two nodes regardless of application protocol (TCP, User Datagram Protocol [UDP], Internet Control Message Protocol, etc.). In Figure 2, AH has been added to the datagram. The AH contains a cryptographic signature of the entire TCP header and the data as well as part of the IP header. When receiving this datagram, the receiver will verify that the signature matches before processing the encapsulated data. Note that no data has actually been encrypted. Therefore, although a third party may examine the data, manipulation of data is very difficult because of the addition of the cryptographic signature.

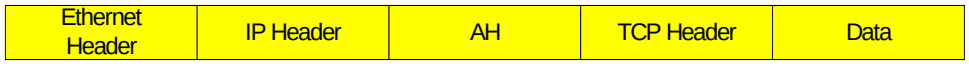

Figure 2. IP datagram with AH applied.

Figure 3 shows the result of adding ESP to an IP datagram. In this case, the TCP header and the data are both encrypted. A third party able to intercept the packet would have a difficult time interpreting the encrypted packet. All that can be determined is that the source and destination nodes are communicating, but the substance of that communication cannot be determined without knowing the cryptographic keys held by the source and destination nodes. If the packet is manipulated, generally, the checksum or other data in the upper layers (TCP or the data itself) will prevent proper decryption and indirectly add authenticity to ESP as a secondary effect.

Figure 3. IP datagram with ESP applied.

Keyed Message Authentication Codes (MAC) are compatible with ESP providing authentication as a primary effect. Figure 4 shows the layout of this type of packet. The IP header, ESP header, TCP header, and data are all authenticated, but only the TCP header and data are encrypted.

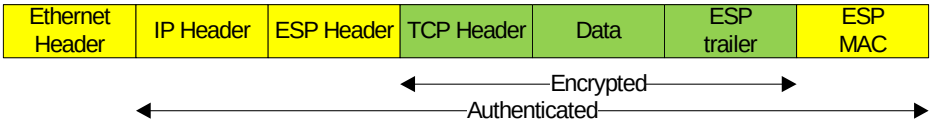

Figure 4. IP datagram with ESP (authentication and encryption).

Both AH and ESP can be employed as shown in Figure 5 to maximize the benefit of IPSec. The payload data is encrypted (confidentiality) and signed (integrity and authenticity). The signature is computed over the IP header, ESP header, and the rest of the encrypted data. Upon receipt, the sender verifies the signature then strips off the AH. The ESP is then decrypted, and packet processing continues as normal.

Figure 5. IP datagram with AH and ESP applied.

2.1 Key Management

ESP and AH are the work horses of IPSec because they perform the bulk data transfer between two nodes. This section examines the method by which two nodes identify themselves to each other and then exchange cryptographic keys. ESP and AH are used once these security associations (SAs) are created. When a packet is to be sent between two nodes, the security parameter database (SADB) is consulted. If no policy is found, the packet is sent without using IPSec. Otherwise, the SADB applies either or both AH and ESP to the packet given the cryptographic keys from the database.

To build this database, nodes must identify themselves to each other and then agree on parameters (algorithms, keys, key length, etc.). This process can either be static or dynamic. For the static case, the SADB is simply created manually by specifying all of the keys and algorithms. According to the specification, statically keyed IPSec does not employ replay protection. Reciprocal configurations must be built for each node in a statically configured network. Static configurations are difficult to maintain and do not scale well because when a new node is added to the network, reciprocal configurations must be installed on all nodes with which the new one will communicate.

The alternative to static configurations is dynamic keying. In this environment, the nodes will use a protocol such as Internet Security Association and Key Management Protocol (ISAKMP[b]). ISAKMP uses UDP port number 500 for communication between nodes. ISAKMP is mainly a framework upon which key management and security parameters can be implemented. The most common protocol within ISAKMP is Internet Key Exchange (IKE[c]) and more recently IKEv2.[d] IKE can authenticate nodes with digital signatures, public key cryptography, or a pre-shared key.

The simplest authentication method in IKE is the use of pre-shared keys, which are generated and then distributed to each node. The key used between two nodes must be the same on each node and the keys must be symmetric. Certificates are used to authenticate nodes for public key encryption.

---

[b]. D. Maughan, M. Schertler, M. Schneider, J. Turner, "Internet Security Association and Key Management Protocol," RFC 2048, http://www.faqs.org/rfcs/rfc2408.html, November 1998.

[c]. D. Harkins and D. Carrel, "The Internet Key Exchange (IKE)," RFC2409, http://www.faqs.org/rfcs/rfc2409.html, November 1998.

[d]. C. Kaufman (ed.), "Internet Key Exchange (IKEv2) Protocol," RFC4306, http://www.faqs.org/rfcs/rfc4306.html, December 2005.

Once the identities of the two nodes are verified, an ISAKMP SA is used to exchange keys. IKE uses this ISAKMP SA and an algorithm like Diffie-Hellman[e] to securely exchange keys for the IPSec SAs (for use with AH and/or ESP, described earlier). The reason for the multistage key exchange is to provide Perfect Forward Secrecy. The idea behind Perfect Forward Secrecy is that the compromise of a key used for authentication (pre-shared key, certificate, etc.) should not allow an attacker to reconstruct the session key used for AH or ESP. This protection is achieved by removing the ISAKMP SA as soon as the session keys are established. Even if an attacker was able to compromise the authentication key, the attacker would not be able to work to the session key because the intervening step (ISAKMP SA) is no longer available.

2.2 IPSec Discussion

IPSec can be used to secure all communications between nodes at the IP layer. The protocol has been heavily researched and vetted with only minor changes but has historically been difficult to configure, and interoperability was not assured with competing products. Great effort has been put into interoperability testing, and vendors have endeavored to make it easier to configure and maintain. Firewall to firewall Virtual Private Network (VPN) interoperability has been hampered by different vendors' interpretation of the IPSEC standard; specifically in the granular timing settings in IKE phase 2 timing settings. Slight granular settings differences between some vendors can cause extended troubleshooting when trying to build and maintain VPN tunnels.

Of particular note with respect to ease of use is the implementation of IPSec included with Microsoft Windows XP, Windows Server 2003, and newer operating software. This implementation takes vantage of the identity proofing afforded by Active Directory (AD). This ability to prove the identity of computers on the network combined with ISAKMP provides the basis for key negotiation on Windows. Further, the use of IPSec can be pushed to new workstations as a part of Group Policy Management Objects, which allow AD nodes to be automatically configured as soon as they join the network to use IPSec with their network peers.

Traditional network IDSs will not be able to decipher ESP packets or the majority of ISAKMP traffic. The data is encrypted such that no third party (including IDS systems) can eavesdrop on the communications. IPSec with ESP makes network based intrusion detection difficult, but host-based intrusion detection products can be configured to examine the incoming or outgoing unencrypted messages and make decisions before they are processed further.

AH does not present quite the same challenge on the surface, but network IDSs can be fooled by invalid traffic. Given two packets, one with a valid signature and the other with an invalid signature forged by an attacker, a network IDS cannot determine which of the two is valid, since it will not have the recipient's key. This inability to make a determination can be used to circumvent a network IDS. IDSs can, however, be used to verify policy; for example, if two nodes are supposed to be using IPSec and are seen communicating in the clear, a network IDS can alert to this anomaly.

IPSec allows the person deploying it to make choices about the level of security and performance required; for instance, if confidentiality of the data is not important, AH can be used without ESP. This set of deployment options will provide strong integrity to the data at a fraction of the computational overhead (the hash functions used in AH are roughly an order of magnitude less computationally expensive than the encryption functions in ESP).

Not discussed thus far is "tunnel mode" versus "transport mode" IPSec. This primer considers node-to-node communications without the presence of a Virtual Private Network (VPN) gateway. "Tunnel mode" IPSec is used to tunnel network traffic from one network to another, and usually involves a VPN

---

[e]. E. Rescorla, "Diffie-Hellman Key Agreement Method," RFC 2631, http://www.faqs.org/rfcs/rfc2631.html, June 1999.

gateway as one of the endpoints of IPSec. Transport mode has less overhead because it does not add an additional header to each packet to describe the tunnel. Figure 6 shows the difference between the two.

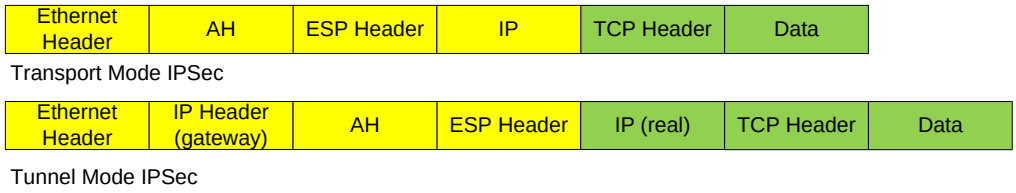

Figure 6. Transport versus Tunnel Mode IPSec.

As shown in Figure 6, an additional header is added. This header is used to route the packet between IPSec endpoints, which are not necessarily the nodes communicating. Once received by a VPN gateway, the outer IP header is discarded and the rest of the packet is authenticated and decrypted. The inner IP address is then used to route the data on to its destination. This overhead is not necessary for node-to-node communications.

2.3 Recommended Configuration

Current best practice with IPSec is to use ISAKMP for key negotiation between peer nodes. For identity proofing of the peers, either Kerberos/AD or digital certificates should be employed for control systems nodes. For remote access, some kind of two-factor authentication should be used (username/PIN combined with a SecurID token, for example). Using ISAKMP over static keying provides Perfect Forward Secrecy as well as replay protection. Manual SA configuration should not be used in production systems because of lack of replay protection and the difficulty in managing key infrastructures.

IPSEC can be deployed with or without encryption. Without encryption, either AH or ESP is used to ensure the integrity of data in transit. If the data is sensitive, then encryption should also be used. The sensitivity of the data should govern the decision to use encryption. Examples of sensitive data could be usernames and passwords and personal employee information including, but not limited to, social security numbers, employee numbers, home addresses, phone numbers, etc.

National Institute of Standards and Technology (NIST) Special Publication 800-77[f] contains a wealth of information regarding specifics of recommended configurations. Their recommendations are directly applicable to a control system implementation.

[f]. S. Frankel, et. al., Guide to IPSec VPNs, National Institute of Standards and Technology, December 2005, http://csrc.nist.gov/publications/nistpubs/800-77/sp800-77.pdf

3. TRANSPORT LAYER SECURITY AND SSL

The Transport Layer Security (TLS) protocol is the Internet Engineering Task Force's standard for the SSL protocol.[g] TLS was designed to ensure that implementations of SSL met the standards necessary for securing business transactions over the Internet. Terms for TLS and SSL are used interchangeably, and most modern implementations of SSL are implemented in terms of TLS. For clarity, this document will use the term SSL, but the use of TLS-compliant implementations of SSL is suggested.

The SSL protocol provides application-level confidentiality, integrity, and authentication by means of encryption. Though SSL provides these features, their use is not mutually exclusive, meaning that SSL can be used to provide authentication alone, authentication and integrity, or the full suite. By design, SSL is optionally protected from replay attacks by the use of MAC. SSL is implemented as a client/server system in which the server can be authenticated to the client, or the server and the client can be mutually authenticated. SSL consists of three phases:

- Algorithm negotiation
- Key exchange
- Communication.

In phase one, algorithm negotiation, the client provides information about the various cryptographic ciphers, key exchange algorithms, and hash functions it supports. The server chooses the strongest mutually supported cipher, algorithm, and hash function and provides the client with that information. The server also provides the client with its digital certificate. This certificate is meant to authenticate the server to the client and contains the server's name, the name of the trusted CA who generated the certificate, and the server's public encryption key. The client may then contact the CA to verify that the server is legitimate. In the case of mutual authentication, the client will also provide the server with its certificate, and an additional key exchange will take place. Once authenticated, the server and client begin phase two.

During phase two, key exchange, the client and server generate the session keys necessary to encrypt the data for the remainder of the communication. The client sends the server a random number encrypted with the server's public key. Since the server alone can decrypt the number, only the client and the server can ever know what the random number is. Mutual knowledge of the random number is important because generating the encryption keys in accordance with the previously agreed-upon algorithms requires its use. With the completion of this phase, phase three is initiated and encrypted data transfer can begin on an established secure communication channel.

Figure 7 and Figure 8 illustrate a clear text TCP packet and a TCP packet encrypted with SSL, respectively.

Figure 7. Clear text TCP packet.

Figure 8. TCP packet encrypted with SSL. The SSL header is authenticated and the data is authenticated and encrypted.

[g]. T. Dierks and C. Allen, "The TLS Protocol," RFC 2246, http://www.ietf.org/rfc/rfc2246.txt, January 1999.

3.1 SSL Discussion

When implemented correctly, SSL can provide cryptographically secure application-level communications between two nodes. Having application-level encryption in lieu of host-level encryption allows the majority of the host's network traffic to be analyzed by an IDS. However, the integrator needs to be aware of introducing unintended weaknesses that could help an attacker compromise the communication or even lead to a full system compromise. This section will discuss common implementation errors and the vulnerabilities they can introduce into a system.

Many of the vulnerabilities associated with the implementation of SSLs surround the use of certificates and CAs. A properly deployed CA is essential to the reliability and scalability of SSL's public key infrastructure. Each node in the network should have a copy of the trusted CA's root certificate, which should be the only self-signed certificate in the network. The purpose of the CA is to verify that all certificates being used in the network are both genuine and legitimate, making the importance of "mutual authentication" enforcement vital. Mutual authentication requires that any two endpoints of a communication provide the other end point with a valid certificate which is signed either by the organization's CA or by a recognized third-party CA, such as VeriSign or Thawte. Organizations should not use self-signed certificates because communication endpoints cannot sufficiently validate the signature. An attacker can exploit the self-signed certificate to perform an SSL man-in-the-middle attack. In this case, all the attacker needs to execute an attack is to craft a self-signed certificate containing information that appears to be legitimate and offer that up to an endpoint the attacker is trying to communicate with.

Again, applications configured to accept self-signed certificates have no way to verify the certificate's authenticity. An attacker could use a tool that has the ability to perform man-in-the-middle attacks by automating the presentation of fraudulently crafted certificates to the victim endpoints or to users. Some applications, like web applications, require a user to accept a self-signed certificate as legitimate before use. Users of poorly configured SSL deployments will see an alert from their web browsers telling them that the browser was unable to validate the offered certificate. Users who regularly see this behavior can become desensitized and may unknowingly accept a fraudulent certificate, thus compromising their session.

Recent research shows that even valid certificates utilizing the MD5 hashing algorithm can be forged by an attacker due to collision vulnerability in the MD5 algorithm. This vulnerability makes it possible for attackers to craft certificates that appear to be legitimate but have identical MD5 hashes compared with the legitimate certificate. Readers are encouraged to verify that a more secure hashing algorithm, such as SHA256, is used in any certificates they deploy.[h]

Another vulnerability surrounding the use of certificates occurs when applications use encrypted certificates. Some applications that do not need user interaction will use an SSL certificate that has been encrypted with a password. Since no user is present to enter the password, many of these applications have the password hard-coded in them. This scenario makes certificate update very difficult as the application has to be rebuilt with a new password. Additionally, if this solution is used, readers are strongly encouraged to use different passwords for each certificate and should escrow these passwords in a secure, preferably encrypted, manner.

Alternatively, some applications utilize pre-shared keys instead of certificates. When these keys are password protected, and the password is hard-coded into the binary (as often is the case with firmware) an attacker can extract the key and the password from the binary and use that knowledge to compromise any future communications. Both scenarios give reason to utilize a CA with certificate revocation capabilities.

[h] http://www.win.tue.nl/hashclash/rogue-ca/

Multiple nodes often share a single certificate. This scenario allows an attacker to easily send data posing as a legitimate node. Since multiple nodes share a single certificate, there is no reliable mechanism for authentication. All of these scenarios make updating SSL library versions difficult, which introduces the possibility of root-level compromises due to outdated libraries. A large number of exploits can be found for various versions of SSL libraries; many have led to remote code execution.

3.2 Recommended Configuration

If a user determines that SSL is the appropriate cryptographic protocol for use in a given situation, the following best practices should be observed:

- The implementation of SSL should be thoroughly researched to make sure there are no known vulnerabilities for that particular library.
- The version of SSL used should be implemented in terms of TLS v1.0 or greater.
- Only strong ciphers, such as AES and strong hashes, such as SHA-256, should be allowed. Additionally, keys should have a length of no less than 64 bits.
- Replay protection should be used by enabling MAC or HMAC.
- Diffie-Hellman should be the preferred key-exchanging protocol and implemented wherever possible.
- A trusted CA should be deployed and its root certificate stored on all nodes. The CA private key should be the most rigorously protected key on the network.
- All CAs not directly involved within the network should be removed from all nodes' certificate stores.
- X.509 is the preferred standard for public key infrastructure and should be used when possible, including its certificate revocation capabilities.
- Self-signed certificates should never be used. All certificates should be signed by a recognized CA.
- Readers are encouraged to use SHA256 in lieu of MD5 hashing in certificate generation. If readers use commercially provided certificates, they should contact their vendors to ensure that SHA256 is used in lieu of MD5.

In addition to these recommendations, NIST Special Publication 800-113[i] contains a wealth of information regarding specifics of recommended configurations. The recommendations in this document are directly applicable to a control system implementation.

[i]. S. Frankel, et. al., "Guide to SSL VPNs," National Institute of Standards and Technology, July 2008, http://csrc.nist.gov/publications/nistpubs/800-113/SP800-113.pdf

4. APPLICATION TO CONTROL SYSTEMS

The previous sections provide an overview of cryptosystems in general and describe two of the most common cryptographic protocols deployed on local area networks. This section describes the application of these protocols and general cryptographic principles to control systems.

Not all data requires encryption, but almost all data requires authentication. The difference between the two, from a practical perspective, is the cost of encrypting data is considerably higher than adding cryptographic authentication. Encryption cost is more noticeable on low-end devices (RTUs, PLCs, etc.) than on high-end servers, which tend to be I/O bound.

The precise data requiring the additional CPU cost will depend on the requirements of the asset owner (the privacy of data may be mandated by regulatory concerns or local policy). An example of data that should be encrypted is login credentials: usernames, passwords, PINs, etc. This data should never be sent "in the clear," authenticated or not. Conversely, data from an anemometer on the roof of a building is unlikely to require privacy.

The security of a control system often has impacts beyond the standard business models of data theft and loss of revenue. If a control system is compromised, either by incorrect information or loss of control to another entity, the results can be catastrophic and include the loss of life or significant threat to the security of a nation.

Proper use of encryption can significantly lower the chance of such occurrences by preventing data corruption and virtually isolating the system from unknown entities. Encryption also has implications for existing network protocols as examined in the following specific applications.

4.1 Bump-in-the-Wire (BitW)

Bump-in-the-Wire (BitW) devices are placed symmetrically on either side of a network between two devices. Figure 9 shows the communication path before and after insertion of BitW devices.

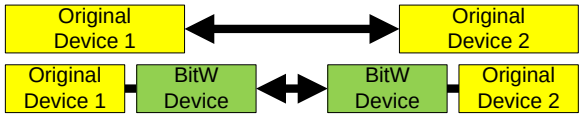

Figure 9. Insertion of BitW devices.

Two additional devices are added to this network. The original devices communicate with their respective BitW device. The two BitW devices communicate with each other and provide encryption services without modification of the original devices. This technology can be applied to local area network communications or serial communications. BitW devices are available for a wide range of interfaces (RS-232, Ethernet, etc.) and provide a method for retrofitting existing hardware devices with encryption without the need for replacing or upgrading the hardware. Readers are encouraged to be sure to configure these devices to accommodate the necessary bandwidth and throughput requirements.

4.2 Retrofitting Protocols

Older control systems protocols were designed with data reliability in mind. These older protocols may include cyclic redundancy checks and other data integrity checks, but little thought was put into authenticating or encrypting the data as an additional means of securing it. Modbus and similar protocols fall into this category. Protocol types such as these strive to ensure that data integrity is maintained, but there are no provisions for ensuring the data is authentic. This situation provides only reliability in message communication and will not ensure that the data is accurate, just that it has not been altered in transit. An additional layer of authentication and integrity can be provided by retrofitting existing protocols to use SSL or IPSec.

4.2.1 Retrofitting with SSL/TLS

Programs like Stunnel[j] provide tunneling of arbitrary TCP connections using SSL. When using SSL, however, caution should be taken that the authentication is mutual. Normal SSL connections require the server to prove its identity to the client, but not the other way around—the server has no proof of the identity of the client in most cases without an external mechanism such as username/password credentials, etc.

To properly deploy an SSL tunnel, a trusted CA must be deployed and used to generate the web of trust. SSL depends heavily on the reliability of the Domain Name System, therefore, deployment of Domain Name System security measures is recommended.[k]

When an SSL tunnel is used, the normal connection made between two sockets is diverted through the tunnel, similar to the BitW devices described in the previous section. The major difference is that instead of a piece of hardware, software is employed to provide the secure path for communications. An SSL tunnel can be used to encapsulate an existing protocol, adding an additional layer of transport security.

4.2.2 Retrofitting with IPSec

In some respects, retrofitting security to existing protocols is easier with IPSec if an implementation is available for the operating system(s) in question. Normally, IPSec is mostly implemented within the operating system kernel and applications need not be aware of whether it is employed or not. While SSL/TLS requires some amount of configuration for each application to setup the tunnel, IPSec need only be configured at the operating system level for all applications to take advantage of it.

The difficulty with deploying IPSec is that a mechanism must be deployed for identity proof of network nodes. This mechanism deployment issue is resolved to a large extent in Microsoft Active Directory deployments, but can be problematic in non-AD environments. The underlying Kerberos authentication of AD provides identity proof and the rest of IPSec can be configured via group policy objects. For non-AD environments the best available configuration is often a CA similar to SSL/TLS.

4.2.3 Dangers of Retrofitting

When a cryptosystem is added to an application, underlying vulnerabilities should be analyzed and addressed. IPSec is often deployed with a policy that requires capable devices to use IPSec, and not all devices are capable (printers and other legacy equipment, for example). If an attacker gains a foothold on a machine known to be incapable of IPSec, unencrypted access is often granted.

A would-be attacker's access ability can be mitigated by applying strict access policies (such as a host-based firewall), but these mitigations are simply layers of defense. Covering all scenario vulnerabilities is beyond the scope of this document, so protocol vulnerabilities outside the encryption layer should still be examined and addressed. Retrofitting encryption technologies on top of legacy protocols can cause unintended consequences such as issues with provisioning and increased latency. Sufficient testing is required to ensure that new encryption controls do not violate operating parameters.

---

j. http://www.stunnel.org.

k. R. Arends, R. Austein, M. Larson, D. Massey and S. Rose, "DNS Security Introduction and Requirements," RFC 4033, http://www.faqs.org/rfcs/rfc4033.html, March 2005.

5. SUMMARY AND RECOMMENDATIONS

As previously stated, application of an encryption technique has four intended goals: confidentiality, integrity, authentication, and nonrepudiation. The following paragraphs present a high-level summary of the recommendations and observations necessary to achieve these goals.

The ideal security for key infrastructure utilizes a hybrid of both asymmetric and symmetric keys. Asymmetric keys are used to establish identity and then a new, temporary symmetric key is used for the bulk of the messaging. Asymmetric algorithms are computationally intensive and their use should be limited to operations that need not be done often (identity verification, digital signatures, etc.). Symmetric keys do not natively provide identity and verification, but they do provide fast encryption services.

Avoid proprietary cryptographic algorithms and servers. Cryptosystems are complex and require a significant investment in resources to produce a secure implementation. Many pre-vetted algorithms already exist in the public sector. Since cryptosystems are designed to be secure, even with intimate knowledge of how the algorithms and servers work, using a reputable, publicly available architecture rather than a proprietary system reduces the chance of inadvertently introducing vulnerability.

Encryption latency, bandwidth, and availability issues can be resolved through traffic analysis and network design. Faster algorithms or additional cryptographic hardware can be added to resolve latency. Cryptosystem components should also be constructed with the same level of redundancy as the main system they service.

IDS degradation issues can be resolved by proper configurations. These configurations include specific parameters on the IDS as well as selecting an IDS that is not affected by the deployed encryption process.

The two most popular cryptosystems (IPSec and SSL) both have advantages and disadvantages. IPSec is a host-level cryptographic solution that is difficult to configure but only requires host configuration. SSL is an application-level solution that is relatively easy to configure but requires individual application configuration. IPSec is recommended for networks with single (or few) operating systems but many applications. SSL is recommended for networks with mixed operating systems. SSL is also recommended when many users share a single host. The recommended configuration specifics for both cryptosystems are discussed in Sections 2.3 and 3.2 of this primer.

Encryption can also be deployed using existing network architectures and protocols. Some suggested ways of retrofitting existing architectures include placing encryption devices on either end of a communications pathway (BitW), retrofitting older protocols with SSL tunnels, or adding IPSec to the host operating system. However, care must be taken to use very strict access policies in these circumstances.

In summary, the application of appropriate encryption techniques can improve the security of industrial control systems. Encryption is not a panacea for control systems security, nor is there an "always correct" answer for control systems traffic encryption. However, more robust and secure data communications can be achieved by analyzing the current or proposed control systems architecture and including encryption as a key component in the overall design.