Cryptography: A Tutorial for Power Engineers

David Whitehead and Rhett Smith
Schweitzer Engineering Laboratories, Inc.

Published in
Sensible Cybersecurity for Power Systems: A Collection of Technical Papers Representing Modern Solutions, 2018

Originally presented at the
35th Annual Western Protective Relay Conference, October 2008

Cryptography: A Tutorial for Power Engineers

David Whitehead P.E., and Rhett Smith GSEC CISSP, *Schweitzer Engineering Laboratories, Inc.*

***Abstract*—Safeguarding power system communications channels from unauthorized use or access is becoming increasingly important. Internal and external utility organizations have suggested many cryptographic solutions as the proper ways to secure substation communications links. However, many proposed solutions are not practical for substation applications because of communications channel limitations, data size and/or data rate requirements, intelligent electronic devices (IED) capabilities, or similar infrastructure constraints.**

This paper is a cryptographic tutorial that addresses methods for protecting communications systems that will enable protection engineers to work more effectively with other departments within a utility.

This paper discusses common cryptographic techniques that are applicable in various substation communications systems. The paper then describes the benefits and limitations of these methods. Specific topics include:

1. **Overview of communications channels used in power systems.**
2. **Review of cryptographic protocols and how they work.**
3. **How cryptography can impact monitoring, control, and protection communications.**
4. **Differences between substation communications systems and corporate information technology (IT) systems.**
5. **Cryptographic impact to operations.**

I. Introduction

Cryptography—just the sound of the word can bring as much confusion as the science behind it brings to the information that it is trying to protect. Cryptography is the science of hiding information. This science has expanded far beyond the goal of keeping the data confidential: now it includes checking integrity and authenticating both the data and the sender. As with all engineering tasks, you must weigh the tradeoffs before selecting the appropriate cryptographic technology for your application. When selected correctly, cryptography can enable an organization to do many tasks more efficiently and effectively. In a way, cryptography can help you do twice the work in half the time.

II. Power System Communication

Electric power system communications are diverse, to say the least. Fig. 1 shows typical electric utility communications links.

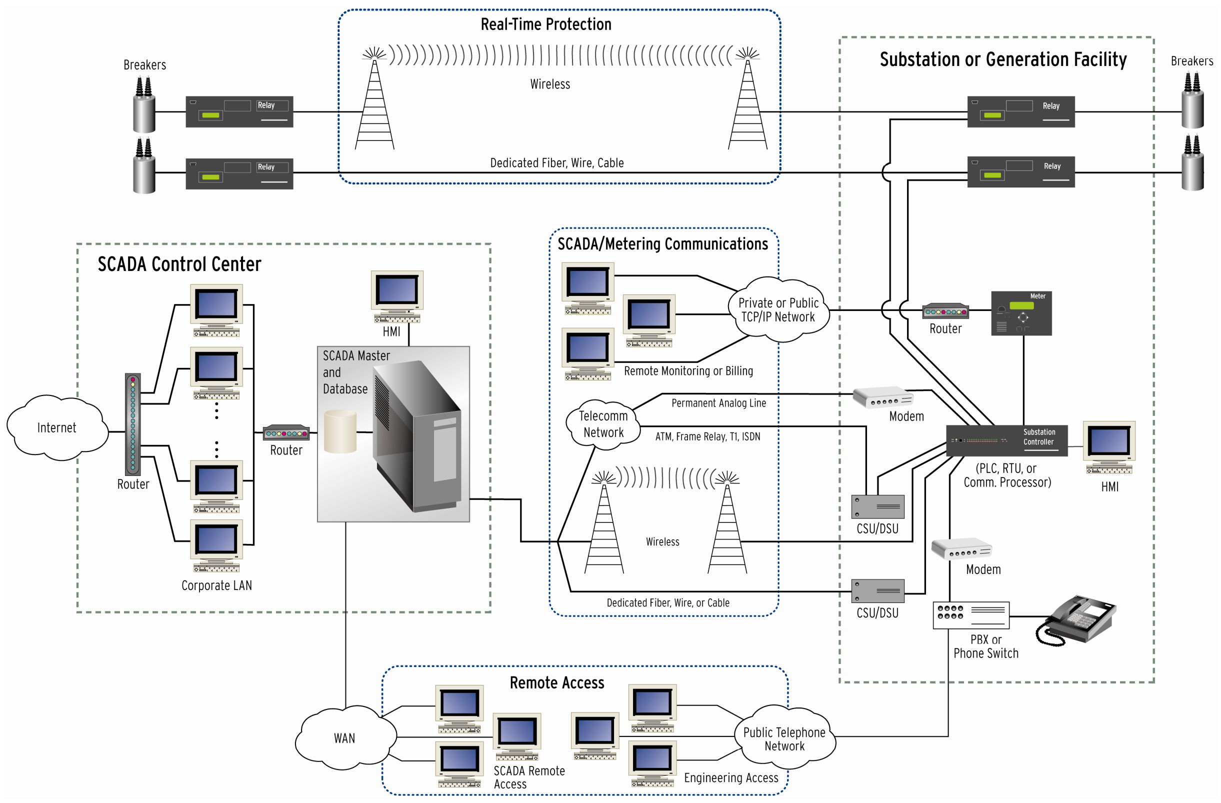
Real-Time Protection
Wireless
Dedicated Fiber, Wire, Cable
Breakers
Relay
Substation or Generation Facility
SCADA Control Center
SCADA/Metering Communications
Internet
Router
HMI
SCADA Master and Database
Corporate LAN
Private or Public TCP/IP Network
Remote Monitoring or Billing
Permanent Analog Line
Telecomm Network
ATM, Frame Relay, T1, ISDN
Modem
Meter
Substation Controller
(PLC, RTU, or Comm. Processor)
CSU/DSU
Dedicated Fiber, Wire, or Cable
PBX or Phone Switch
Remote Access
WAN
SCADA Remote Access
Engineering Access
Public Telephone Network

Fig. 1. Electric utility communications links

Some examples of electric utility communications links include:

- Radio frequency (RF)
- Dedicated fiber
- Ethernet
- Public telephone
- Telecommunications networks

These communications links are used to perform three primary functions:

- Real-time protection
- SCADA
- Engineering access

Real-time protection communication transfers information used to control the electric power system. Real-time protection data payloads range from a few bytes for pilot protection to a few hundred bytes for line current differential protection. Data exchanged between protection devices are very timely: devices must receive new data in milliseconds. Real-time protection is typically point-to-point, and no request is necessary to send protection information to the receiving end. Even IEC 61850 GOOSE is a subscription architecture where devices receive control data without request.

The time value of the data is short lived because the value of a current measurement is valid only for that particular instant.

The timing for SCADA communication is less strict: update rates range from hundreds of milliseconds to minutes. Usually, SCADA communication consists of a poll/response format. Typical electric utility SCADA protocols are DNP3, Modbus®, and IEC 61850.

Engineering access provides retrieval of event reports, access to fault locations, change settings, and other similar activities. Timing is generally not critical; if a relay responds to a request in 1.05 seconds instead of 1.00 seconds, the user would be hardly aware of the 50 ms delay. This is in contrast to real-time protection delay, where 50 ms could have a severe impact on power system stability. The time value of the data is longer lived because settings, event reports, and the like can have a long life.

Many utilities are moving communications systems to Ethernet/IP or SONET. However, there are and will continue to be many electric utility communications links that are serial based. For the purpose of this paper, serial networks have data rates from 300 bits per second (bps) to 115,200 bps. Most serial networks in electric utility systems operate between 1200 bps and 9600 bps.

Serial communications system protocols (such as Modbus, DNP3, and MIRRORED BITS®) and architectures were designed many years ago. These systems were developed to be reliable and dependable but not necessarily secure in a cryptographic sense. At the time of development for most serial protocols in common use in the United States, utilities assumed that these protocols were integrated into trusted networks—additional data security was never a concern.

As we rely more and more on automation, it is imperative that we secure all utility communications. However, adding cryptographic security to existing communications links is a challenge. The largest challenge is finding the additional bandwidth that the cryptographic functionality needs. Cryptography will consume some bandwidth and add latency to a communications link. The impact of bandwidth consumption or data latency will vary according to the communications link and the type of cryptography selected. For example, engineering access communications links are not time sensitive, so adding cryptographic mechanisms does not adversely impact the use of these data. However, when we add cryptography to real-time protection or SCADA data, bandwidth and data latency are major concerns.

Cryptographic algorithms and schemes generally require additional channel bandwidth in order to keep frames synchronized and to provide the basis for frame security. The additional bandwidth is for items such as initial values and counters. These additions can adversely impact bandwidth-limited communications systems. Engineers have designed their real-time protection and SCADA communications systems to take full advantage of the available communications channel bandwidth. For example, in a SCADA system, assume that there is a master that polls slave devices. A typical baud rate for an installed system is 9600 bps, using 8 data bits, no parity, and 1 stop bit (9600 bps, 8, N, 1). For a Modbus poll/request session, an analog value request is 8 bytes, and a slave response is 192 bytes. The total number of bytes per poll/response is 200 bytes. If we use the baud rate stated above, the total single poll/response time for the transaction is 208 ms. Allowing for some processing delay, the master could poll 9 slaves in about 2.1 seconds. This would leave approximately 228 ms of total idle time (required bandwidth: 0.208 x 9 = 1.872 seconds; idle time: 2.1 – 1,872 = 0.228 seconds or about 11 percent of the total channel bandwidth in an idle state). In order to provide data security and not disrupt the existing polling-cycle scheme, the cryptographic overhead must use less than 11 percent of the available channel bandwidth. The following sections discuss cryptography and how it can be applied to protect these communications links.

III. Cryptography

Let us start by identifying the goals of cryptography: confidentiality, integrity, and availability (CIA).

1. Confidentiality: Concealing information from unintended viewers. Encryption scrambles information by using known algorithms and secret keys to make information incomprehensible to unintended users. It is best to use encryption algorithms approved by the National Institute of Standards and Technology (NIST) to ensure mathematical robustness. There are two types of key structures in encryption algorithms: symmetric key and asymmetric key. In symmetric key cryptography, the two parties who want to exchange information securely use the same algorithm and the same secret key to encrypt and decrypt. Symmetric cryptography is computationally efficient, but it requires that each sender/receiver pair has a unique, secret key. The operational difficulty in this is how the sender and receiver initially agree on a secret key. If the communications channel is untrusted, passing a symmetric encryption key needs an out-of-band transport solution. Asymmetric key cryptography uses two keys: public and private. In this system, the public key is known to everyone. The private key is a secret, and the private key owner never shares it with anyone. If two parties need to exchange information securely, the sender uses the receiver's public key to encrypt the message, and the receiver uses the private key to decrypt the message. Asymmetric cryptography is much more computationally burdensome, but it does not have the initial key distribution problem that symmetric keys have. The receiver can send a public key to everyone without worry of it being eavesdropped. Modern cryptographic systems use a hybrid approach that uses asymmetric key cryptography to send symmetric keys. This solves the initial symmetric key exchange problem and allows the use of symmetric encryption for the information transfer, leveraging the speed of symmetric key cryptography.
2. Integrity: Ensuring information has not been tampered with or altered. Integrity checking is usually accomplished by running information through a cryptographic algorithm, called a hash, and appending the hash to the end of the data message. When the receiver gets the message from the sender, the receiver reruns the data portion of the message through the same hash algorithm to create a calculated hash. If both the calculated and received hashes match, the receiver can be assured the data were not altered. The hash algorithm can be used with or without a secret key.
3. Availability: Ability to use information when it is needed or desired. When you need a resource, you want it to be there, a requirement similar to that for power system protection communications channels. The most common cyberthreat is a Denial-of-Service (DoS) attack, where a third party disrupts or otherwise makes the communications media unavailable.

Confidentiality conceals and protects information from anyone who is not the intended receiver. This keeps the data confidential or indecipherable, scrambled in such a way that no one except the intended receiver can understand the information. Past methods used physical concealment of a message. One ancient Roman example involved shaving the head of a messenger, tattooing the message on the head, and then letting the hair grow back. The intended receiver knew to shave the hair of the messenger to read the message. A faster method of physical encoding was the scytale, a message written on a long strip of cloth that, once wrapped around a specific pole of the correct diameter, would reveal the message. More recent encryption techniques used alphabetic replacement methods to conceal the message. In a simple example, the sender replaces every A with a Z, and the intended receiver knows to replace all Zs with As. Alphabetic replacement took on many forms, each more complicated than the next, to protect against cracking or unintended receivers breaking the decryption method and understanding the transmitted information. Today, cryptography is based on mathematical permutation and substitution.

It is important to investigate many factors before selecting the most appropriate cryptographic solution for your system. The most important first step is to select NIST-approved algorithms, but knowing that a solution claims 256-bit Advanced Encryption Standard (AES) encryption does not ensure that it will protect your information correctly.

Implementation architectures are just as important. There are many different encryption implementation methods. Identifying the most appropriate method for your network is critical.

A. Confidentiality

Confidentiality, or encryption, hides the content of a data message from unauthorized viewers. For the purposes of this discussion, we will assume that the data exchanged in a real-time protection system, SCADA system, or engineering access are digital. Encryption of digital data will be kept simple in this example: a data message is XOR with a secret key only known by the sender and receiver of the message. For example, if we want to encrypt the ASCII character for 1 with a secret key represented by an ASCII character for 7, the result would be:

ASCII Char	Secret Key	Result
00110001	00110111	00000110

The most important part of the above encryption process is using key material that can keep the data secure. Two examples to illustrate the importance of selecting the right encryption architecture to generate this key material are electronic codebook (ECB) and counter mode (CTR).

ECB is very fast, and much of the encryption process can be precomputed. The drawback is that for every given block of information there is one encrypted output. This is bad if the information you are sending is often repeated, as in a control system command response environment.

CTR mode seeds the key algorithm with a count that changes, which results in a different encryption key. Even if information is sent repeatedly, the encrypted output is different as long as the count does not repeat.

Table I shows examples of implementation methods and a few of their advantages and disadvantages.

A stream or block cipher denotes how much data are processed at any one time, which results in a requirement of how much data need to be received before calculations can start. The second column in Table I shows whether any calculations can be done in advance of receiving data. The more calculations that can be done in advance, the lower the impact cryptographic functions have on your real-time processing. Advance calculations speed up the encryption or decryption process. Lastly, does an error in one of the sent messages cascade and corrupt following messages? If a mode has “No” in the Cascading Errors column, there will be corruption in the current message, but these errors will not carry over to the next message.

TABLE I
COUNTER MODE IMPLEMENTATION EXAMPLES

Modes	Stream/Block Cipher	Advanced Work	Cascading Errors
Electronic Code Book	Block	Yes	No
Output Feedback	Stream	Yes	Yes
Cipher Feedback	Stream	Partial	No
Cipher Block Chaining	Block	Partial	No
Counter	Stream	Yes	No

Another concept that must be investigated can be summed up in the term key space. Key space is all the possible keys the system can have. If an alphabetic encryption system has a code word to encipher and decipher messages, the key space is all possible letter combinations that code word may be. A four-letter code word number of possibilities is easier to break than a ten-letter code word number of possibilities. For data, the key space is 2^n possibilities where n is the number of bits long the key is. For example, if you have a 128-bit key, there are 2^{128} = 340,282,366,920,938,463,463,374,607,431,768,211,456 possibilities. It would take a computer with a 2.5 GHz processor 4.316 x 10^{21} years to guess the correct key if the processor could have a new guess on every clock cycle and guessed the correct key on the last guess.

Computer processing power doubles every two years. Applying this fact to control system equipment that may be installed for ten years shows that the key space must be strong enough to protect against computers running 80 GHz processors. Looking again at the 128-bit key example, we see that it would take an 80 GHz computer 1.349 x 10^{20} years to break the key under the same assumptions. This shows us that selecting a key space of 128 bits or greater is appropriate.

Let us dig a little deeper into the structures of stream ciphers and block ciphers. Stream ciphers encrypt or decrypt information one digit at a time, while block ciphers combine multiple digits to perform encryption functions on all information at once. In real-time systems, it may not be acceptable to wait for larger blocks of data on a serial line to collect before encryption functions can start, whereas engineering access may be tolerant of the additional latency. This makes stream ciphers attractive for control system applications. However, stream ciphers may be susceptible to man-in-the-middle attacks. These types of attacks occur when attackers know the original message that is being sent (such as a DNP3 trip command) and also know what they want to change the message to (such as a DNP3 close command). As illustrated in Fig. 2, these attacks are simple to perform if the stream cipher does not include some sort of man-in-the-middle attack prevention.

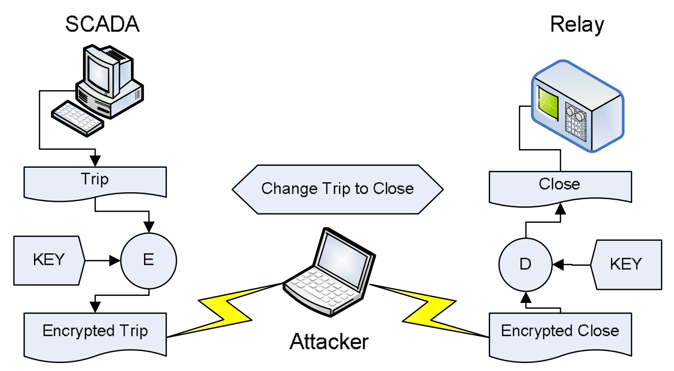

Fig. 2. Man-in-the-middle attack

Let us take a closer look at how someone might accomplish this attack. We use a small data string to show the attack more clearly, but we can scale the principles of the process to any size of information. Suppose we want to send the ASCII letter A, which in binary is 01000001, and the attacker wants to change it to the ASCII letter Z, which in binary is 01111010. The original sender encrypts 01000001 with a secret key 10001011. For example purposes, we will XOR the data to encrypt and decrypt, so the cipher text would be:

A:	01000001
Key:	10001011
Cipher Text:	11001010

Under this attack, we assume that the attackers know the original message being sent, A, the message they want to send, Z, and that the attackers need only to capture the transmitted cipher text, 11001010. Here are the steps to successfully achieve this attack:

1. Attackers intercept encrypted data 11001010.
2. Attackers determine which bits to flip.

A is different from Z in: 1st, 2nd, 4th, 5th, 6th bits:

Bits:	87654321
A:	01000001
Z:	01111010

If 11001010 is the encrypted message for 01000001 and we flip all the bits that are different between A and Z, the result is a new encrypted message:

Bits:	87654321
Old encrypted message:	11001010
New encrypted message:	11110001

If we send this new encrypted message and allow the receiver to decrypt it with the key, they will receive a Z and not an A.

Modified message:	11110001
Key:	10001011
Plain text:	01111010 “Z”

Please note that the attackers did not need to know the secret key or even crack the key; they just used the encryption and decryption process as it was set up and allowed it to decrypt the message appropriately.

Technology to mitigate this type of attack exists for streaming ciphers. One solution is bit scrambling. Bit scrambling reorders the data within each digit that is being encrypted; the receiver places the data back into the appropriate order after the data are decrypted. Because attackers do not know the bit-scrambling order, they do not know what bits to flip to manipulate the data. Even attackers who know the original message and what message they want to change it to still do not know which bit to flip in each digit.

If we look at the same example as before and add bit scrambling, we can see how it stops man-in-the-middle attacks. For this example, we will swap each pair of bits, Bit 8 with Bit 7, Bit 6 with Bit 5, and so on through each byte.

A:	01000001
A (scrambled):	10000010
Key:	10001011
Cipher:	00001001

The attacker would still assume the same bits to flip to turn an A to a Z and would alter the cipher text to:

Old encrypted message:	00001001
New encrypted message:	00110010

The receiving side would now receive the new encrypted message and use the key to decrypt it, resulting in:

Encrypted message received:	00110010
Key:	10001011
Plain text:	10111001

This is not a Z and most likely will not be any matching command, and the message will be dropped.

B. Integrity

Selecting the correct cryptographic solution for your system depends on the goals of your organization. Message integrity is very important for utility communications. Integrity is the process of ensuring that the message received was not altered in transit and that it was sent by an authorized user. Hashing, keyed hashing, and digital signatures accomplish this process. Hashing is computing the information in a one-way process, where the outcome is a fixed-length answer that cannot be reversed. If any part of the original message is altered, the hash changes. The process uses the following steps:

1. The sender runs the data through a cryptographic hash algorithm and generates a message digest.
2. The message digest is appended to the end of the data message and sent to the receiver.
3. The receiver runs the data portion of the message through the same hash algorithm. If the received message digest and the calculated message digest match, then the receiver can be assured that the data were not altered.

No matter how big or small the amount of information that is hashed, the answer, or digest, will always be the same length. This protects the original information from any manipulation because an unintended receiver of the hash digest cannot assume anything about the original information. The message digest strength is the probability of the data message being altered and the hash not changing. This strength is a function of the algorithm and length of the message digest. Other unique qualities of hashing, or a hash digest, are that the process is one way, and no one, given the hash digest, can reconstruct the original information. Any change in the original information, no matter how small, makes a large impact on the hash digest.

To accomplish integrity checking and sender authentication, a combination of hashing and asymmetric encryption is used. The sender of information hashes the message and encrypts the hash digest with a private key. Lastly, the sender appends the encrypted digest to the original message. Once the intended receiver gets the message, the receiver strips off the encrypted hash digest, rehashes the information, decrypts the received hash digest with the sender's public key, and compares it to the digest they computed. If the two hash digests match, the receiver has authenticated the sender and verified that the information received was not altered in transit. This is illustrated in Fig. 3, where H represents the mathematical act of hashing, E is encrypting, D is decrypting, and CT is cipher text.

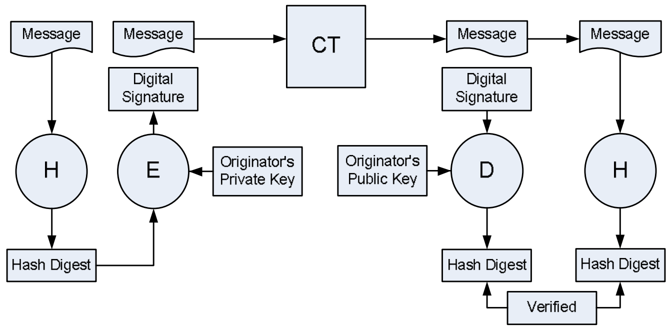

Fig. 3. Information hashing process

We can perform the hash function with or without a secret key. If we use a nonkeyed hash, then we can verify only data integrity. If we use a keyed hash, we can then verify both data integrity and authenticity, assuming the key is kept a secret.

From a power engineering perspective, authenticating each message comes at the price of additional bandwidth requirements. For example, commonly used hash functions generate 160-bit hash. If we are protecting a DNP message that is 64 bits, then the DNP and hashed message we send is 224 bits, and we have added 71 percent overhead. In many existing protection or SCADA communications channels, there is not enough available channel bandwidth to accommodate this much overhead. Fortunately, hashes can be truncated, so that we append only 120 bits instead of 160 bits. This truncation reduces the bandwidth requirements at the cost of slightly less security for detecting a message alteration.

System availability in a utility communications structure is at the top of the priority list. We can accomplish this through redundant communications paths. These paths can use the same or different technologies. For example, we may have Ethernet for the main channel and a backup channel built on dial-up.

C. System Impact

Let us evaluate two serial cryptographic protocols and their impact on protection communications links.

1) Protocol Design 1 Objectives

1. Minimal latency
2. Minimal cryptographic overhead
3. Defense against modification, splicing, replay, man-in-the-middle, forging, and reordering
4. Maintain intercharacter timing
5. NIST-approved Federal Information Processing Standard (FIPS) publications AES (FIPS 197 Advanced Encryption Standard) encryption

This protocol provides data confidentiality and session authentication but not individual message/frame authentication. A cryptographic frame consists of a header and the encrypted data. The header is 7 bytes and consists of start-of-frame characters, a counter, and other data to keep the encryption/decryption process synchronized. The data length of the message, protected by the header, may be user defined. The user can configure the protocol frame to match the encryption frame structure.

Matching frame structure minimizes delays that could arise between data frame size and encryption frame size. Typically, the data length is set to the maximum message length of the protocol. For example, data length would be 100 bytes for Modbus. When the encryption device receives the first byte of data to be protected, the frame header is sent followed by the encrypted byte. Thereafter, each plaintext byte is encrypted and sent on a byte-per-byte basis until the frame data length is exceeded. When the frame length is exceeded, a new frame header with an incremented counter is created, and the process repeats. Fig. 4 shows the session data and link header and the encrypted data layers for a single frame.

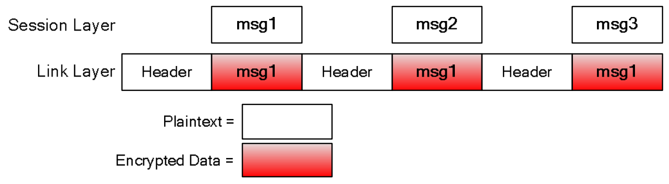

Fig. 4. Protocol Design 1 session and link layer

This simple protocol is very efficient because it does not burden the existing SCADA channel with a lot of cryptographic overhead. The frame header is the only additional channel burden. The frame header is relatively small at seven bytes, and the user can tailor the frequency at which the header is sent, based on the data frame size. After the frame header is sent, each byte received by the cryptographic protocol is encrypted and sent. The encryption process, therefore, only incurs a one-byte delay. Referring to our original SCADA in Section II, the cryptographic overhead only consumes 7 percent of the remaining 11 percent.

2) Protocol Design 2 Objectives

1. Message integrity protection
2. Defense against injection, modification, splicing, replay, man-in-the-middle, and reordering
3. Authentication
4. Confidentiality
5. NIST-approved Federal Information Processing Standard (FIPS) publications AES (FIPS 197 Advanced Encryption Standard) encryption

This protocol considers message integrity and authentication the most important design goal, more so than confidentiality. The rationale for this approach is that in a protection or SCADA system most messages between a SCADA master and slave are not secret. For instance, an open breaker in a SCADA system is known and expected, but it is extremely important to ensure that the message a remote device receives instructing it to open the breaker is from an authorized source. Fig. 5 shows the session and link layer.

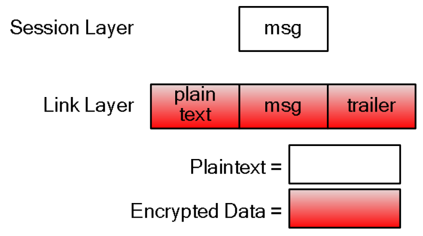

Fig. 5. Protocol Design 2 session transport and link layer

This protocol must receive an entire frame before further processing can occur. If the frame passes all cryptographic tests, the system forwards the frame to the host. If the cryptographic tests fail, the message is discarded. The advantage to this mode is that each message is authenticated before the system passes the frame on to the host. The disadvantage is that the host must receive the entire frame before the host can authenticate the frame. This wait for the entire frame results in latency. In some applications this may not be acceptable.

Refer back to our Modbus example in Section II. This results in 200 bytes plus 2 • 22 overhead bytes = 244 bytes. The total time for the request/response of the system in Section II is 2.28 seconds; recall that our polling rate was 2.1 seconds. With this protocol addition, we have extended the required poll cycle by 0.18 seconds. In other words, we must either increase the polling rate or decrease the number of polled devices to meet the same 2.1-second requirement.

Note that even if we remove the encryption and decryption functions of this protocol, the additional overhead remains the same. Removing encryption allows you to troubleshoot the communications channel, but it does not lower the additional bandwidth requirements and cryptographic overhead.

We developed the test system in Fig. 6 to validate the actual impact of these two encryption schemes on SCADA traffic.

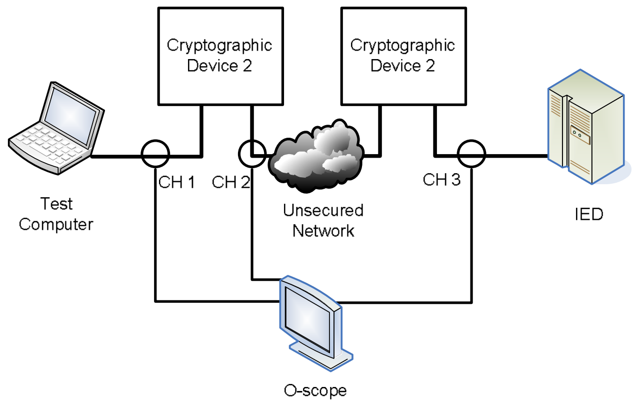

Fig. 6. Test configuration

SCADA traffic was generated on a test computer acting as a SCADA Master. A single DNP3 slave at the remote end was configured to respond to the SCADA master traffic. The two serial encryption devices, in turn, were placed between the SCADA master and the SCADA slave. These devices were configured for a data rate of 9600 bits per second, 8 data bits, no parity, and one stop bit. Protocol 1 was configured to use an initial value (IV) size of 3 bytes, AES-128 encryption, and to delay delimited framing (in other words, idle times between transmissions indicated frame boundaries). Protocol 2 was configured for ad-hoc framing (in other words, idle times between transmissions or a maximum frame size indicated frame boundaries), a maximum frame length of 64 bytes, AES-128 encryption, and HMAC-SHA1 with a 128-bit key. Recall that this protocol holds back an entire frame until it is fully received and passes all cryptographic validation tests.

Three points within the communications systems were tapped and routed to an oscilloscope to measure the relative timing between the transmission of frames. The transmit lines of the SCADA master to the first encryption device, from the first encryption device to the second, and the second encryption device to the SCADA slave were captured.

The DNP "Read Binary Inputs" command was used to elicit a response from the slave device (see Fig 7). DNP3 requests are composed of a header of 10 bytes, followed by a data field containing the command. The first two bytes are a synchronism field composed of 0x05 and 0x64. The next byte is the length of the data portion of the frame, including the addresses and link layer field, but excluding any CRCs. The next byte is a link layer field indicating any link layer operations, such as frame check validation and link layer verification. The next four bytes are the destination and source addresses. The last two bytes of the header are a 16-bit CRC. The remainder of the frame is the actual request. The request is raw data, and a 16-bit CRC is inserted every 16-bytes.

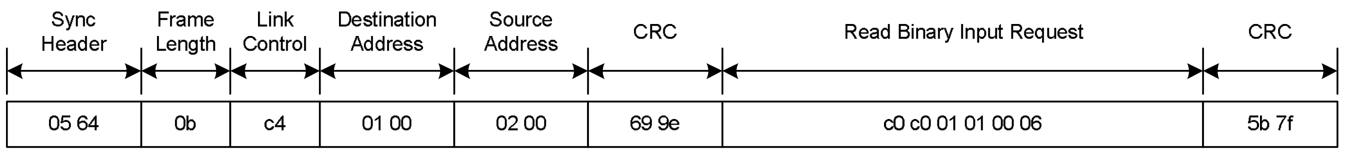

Fig. 7. Read binary inputs DNP3 frame

D. *Protocol 1 Data Latency*

Fig. 8 shows the latency of transmission of a “Read Binary Inputs” request introduced by Protocol 1 on a DNP3 frame at 9600 bps, 8 data bits, no start bit, and 1 stop bit data rates.

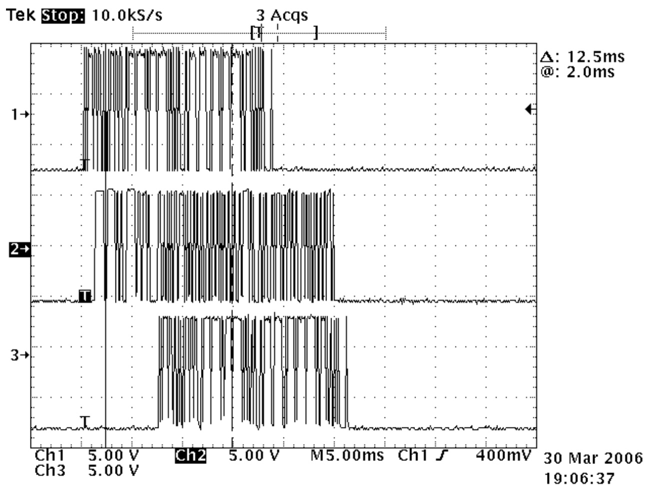

Fig. 8. Protocol 1 mode DNP3 latency

The overall latency introduced by the protocol is roughly 7 ms. This is to be expected because we added 7 bytes of header information.

E. *Protocol 2 Data Latency*

Protocol 2 has significantly different results. Fig. 9 shows a latency of approximately 59 ms.

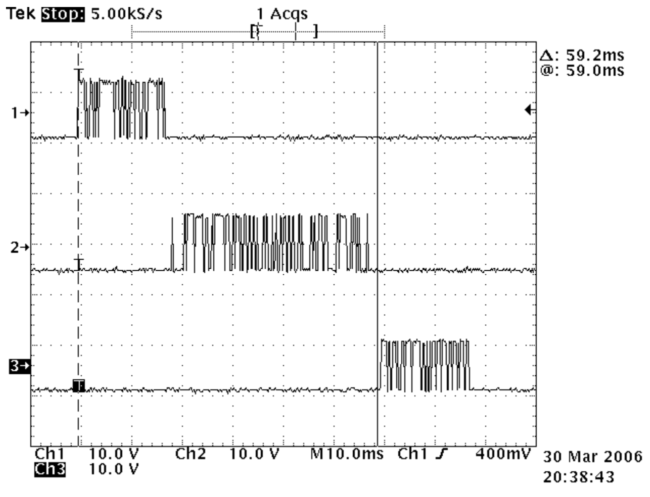

Fig. 9. Protocol 2 DNP3 latency

It is important to note that output from the encryption and decryption devices does not even begin until after the entire frame is received. The initial delay is caused by the holdback mode of operation, which encrypts blocks of 16 bytes. No output is generated until at least 16 bytes of data are received. Protocol 2 performs cryptographic tests on the received frame (i.e., authentication) before transmitting the decrypted results. This authentication creates output latency from the decryption device.

IV. Personnel Impact

After looking at the system impact of cryptography, our next consideration is the operational impact. In other words, if we use cryptography, what are the ongoing tasks needed to keep it running correctly? This is answered by investigating scalability and maintainability. Scalability refers to the readiness of a system to grow or shrink in an efficient manner. Maintainability focuses on what needs to be done to make sure the cryptographic technology continues to be used in compliance with your policies and procedures after initial rollout: how much financial and procedural overhead will this technology incur on operations? Evaluating the business impact of this additional procedural overhead is very important.

The first step in investigating maintainability is understanding your operational needs. In a corporate networking environment, confidentiality usually holds the highest priority. Engineering access communication in control system confidentiality is important but may be trumped by availability, whereas in SCADA, communication authentication and integrity hold higher priority.

You must understand the policies and procedural requirements of your organization to select the appropriate cryptography. Once you identify requirements and select technology, the next step is to test, set initial configurations, and plan deployment. An understanding of cryptography is necessary for analyzing and testing proposed solutions.

You can leverage accredited third-party validation processes such as Federal Information Processing Standards (FIPS) validation to provide a level of assurance. In testing, focus on analyzing whether the cryptographic technology meets your objectives, and make sure the type of cryptography matches the type of communication it is applied to. For example, if you are applying the cryptography to an engineering access communications channel and only authenticate and integrity check the information, your cryptographic solution falls short.

You should also apply confidentiality to protect passwords and settings being communicated across this channel. If you are using encryption on SCADA data, do not use ECB because it is vulnerable to replay or man-in-the-middle attacks. If attackers know when the trip command flies by, they can just record it and send it later to cause a trip whenever they want.

The second half of maintainability covers all the additional requirements of operations to update and support the technology after deployment. These requirements include account and key change controls, event and log retrievals, updates and patch management, and periodic validation testing. This cost depends on the type of features the product includes. For example, using a product that has central authentication or includes software to do account updates and creation from a central location to all installed units will be faster than having to physically visit each installed unit. In-band messaging can help, but the trade off is the use of bandwidth. Depending on your network, this increased bandwidth may be acceptable.

Scalability costs arise when your control system demands change and the system grows or reduces in size. Suddenly you must establish new trust relationships. Cryptography is about shared secrets and the trust that a secret has not been compromised. The cost of scalability is how much it will cost your organization to establish the new system trust as well as the cost of additional deployment.

V. Control System vs. Corporate IT

Corporate IT systems now use a variety of solutions. There is a danger, though, in selecting solutions for a control system just because they have succeeded in the corporate environment. IT goals and objectives are predominantly ordered as confidentiality, integrity, and availability. In a control system, it is the reverse: availability, integrity, and then confidentiality. The procedural requirements on how work gets done are very different as well. Control system equipment demands very long life cycles, and installation of this equipment is often in remote installations where little to no remote telecommunication is available. Many installations have high levels of customization, resulting in elevated levels of testing before patching or changes are implemented. Even with all of these differences, cryptographic solutions can and should be applied to control systems to provide higher availability, confidence in data integrity, and confidentiality of sensitive information.

VI. Recommendations

Securing real-time protection, SCADA, and engineering links is practical and prudent. As we have shown, if we choose cryptography correctly, even bandwidth-limited serial channels can be cryptographically secured.

Ultimately, you want cryptography to enable effective and efficient operations and communication. This is accomplished by allowing information to be placed in easily accessible locations for authorized users who have confidence that the information is accurate and private. Cryptography lowers the potential for mistakes or for work to be done on compromised information. It also allows an organization to understand clearly who is accessing information and how that information is being used. Cryptography can be applied to utility communication networks without impacting system design and operations. CIA has an appropriate place in utility communications.

- Confidentiality protects sensitive data like passwords.
- Authentication ensures control commands are from an authorized source.
- Availability can be addressed with backup or redundant communications channels.

Stream ciphers are better applied to real-time protection data, while block ciphers are better applied to engineering access data. Cryptography enables you to use more interproduct communication, centralize data, and access equipment from anywhere, speeding up work force efficiency.

VII. Biographies

David Whitehead, P.E. is the vice president of Research and Development at Schweitzer Engineering Laboratories, Inc. Prior to joining SEL he worked for General Dynamics, Electric Boat Division, as a combat systems engineer. He received his B.S.E.E. from Washington State University in 1989, his M.S.E.E. from Rensselaer Polytechnic Institute in 1994, and is pursuing his Ph.D. at the University of Idaho. He is a registered professional engineer in Washington and Maryland and a Senior Member of the IEEE. Mr. Whitehead holds seven patents with several others pending. He has worked at SEL since 1994 as a hardware engineer, research engineer, and chief engineer/assistant director and has been responsible for the design of advanced hardware, embedded firmware, and PC software.

Rhett Smith is the Development Manager for Security Solutions in Research & Development at Schweitzer Engineering Laboratories, Inc. In 2000, he received his B.S. degree in Electronics Engineering Technology, graduating with honors. Before joining SEL, he was an application engineer with AKM Semiconductor. Mr. Smith has his GSEC, GIAC Security Essentials Certification and is a Certified Information Systems Security Professional.