ELECTRONIC SECURITY OF REAL-TIME PROTECTION AND SCADA COMMUNICATIONS

ALLEN RISLEY, JEFF ROBERTS, AND PETER LADOW

SCHWEITZER ENGINEERING LABORATORIES, INC.
PULLMAN, WASHINGTON, USA

Presented before the

5th ANNUAL
WESTERN POWER DELIVERY AUTOMATION CONFERENCE
SPOKANE, WASHINGTON
APRIL 1–3, 2003

ELECTRONIC SECURITY OF REAL-TIME PROTECTION AND SCADA COMMUNICATIONS

Allen Risley, Jeff Roberts, and Peter LaDow
Schweitzer Engineering Laboratories, Inc.
Pullman, WA USA

INTRODUCTION

Recent events have placed a heightened emphasis on the electronic and physical security of our electric power infrastructure. The need to physically secure these critical assets is well understood. The risks posed by electronic attack, however, are not as obvious. Likewise, the procedures and technologies required to implement effective electronic security for our critical communications infrastructure are not well known.

In this paper, we summarize the properties of common communications functions that exist in the electric power industry. We outline the requirements of these systems and discuss some of the electronic vulnerabilities found in common implementations of these functions. Finally, we provide an overview of the theory and techniques of modern cryptography and show how we can employ these technologies to improve the security of our communications infrastructure.

COMMUNICATION APPLICATIONS WITHIN THE ELECTRIC POWER INDUSTRY

The communications infrastructure used within the electric power industry consists of a huge mixture of different technologies, protocols, and functionalities. In the following diagram (Figure 1), we have consolidated the common communications functions into three main categories: real-time protection, supervisory control and data acquisition (SCADA)/metering, and remote access. We discuss the distinct properties of each of these categories in the sections below.

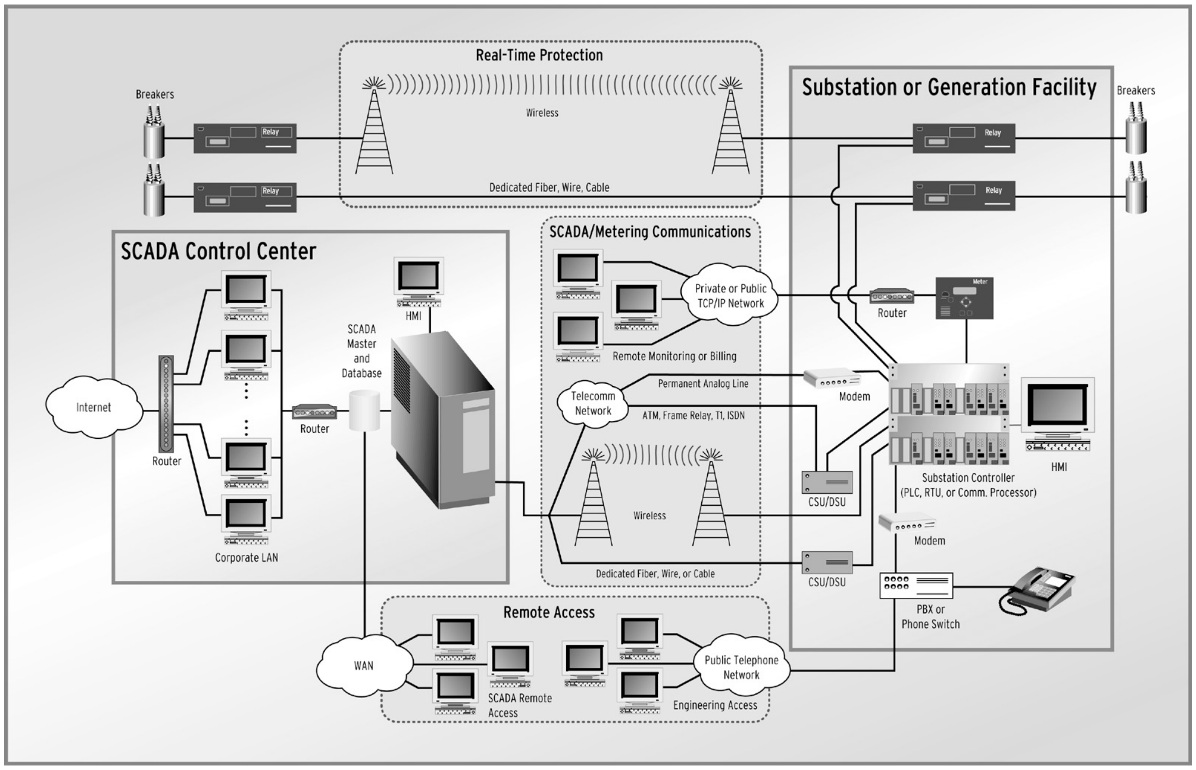

Real-Time Protection
Breakers
Relay
Wireless
Dedicated Fiber, Wire, Cable
Substation or Generation Facility
Breakers
SCADA Control Center
SCADA/Metering Communications
Private or Public TCP/IP Network
Router
Meter
HMI
SCADA Master and Database
Remote Monitoring or Billing
Internet
Router
Router
Telecomm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

Figure 1 Communications Functions Within the Electric Power Industry

Real-Time Protection

Power systems are operating close to their design limits. These operating conditions leave little room for error in the protection and control systems. We expect this trend to continue until significant funding is directed towards the power system infrastructure. System operators rely on high-speed fault clearing protection and control systems to preserve the transient stability of the power grid. These high-speed tripping schemes are called pilot protection because they utilize end-to-end communications to provide high-speed, simultaneous fault clearing.

Communication plays a vital role in these pilot schemes. Recently, SCADA systems have received a lot of security attention. Do the protection systems require the same attention? Would these systems benefit from the same type of electronic security solutions? We will outline the communications properties and requirements of real-time protection systems in this section and defer discussions of electronic security recommendations until later in the paper.

Pilot protection schemes and SCADA control schemes are similar in that either system can potentially initiate breaker tripping. The communications channels and equipment requirements for pilot protection schemes differ from those used for SCADA in the following ways:

- They are predominantly operated on private, closed, and deterministic networks.
- Signal transmission and reception must have known and minimal delays.
- With the exception of direct transfer trip schemes, most pilot protection schemes qualify received messages with locally measured quantities.

We will limit the discussion to two of the most popular pilot protection schemes: directional comparison and line current differential.

The most widely used pilot protection system is directional comparison [2]. Major reasons for this wide acceptance are the low channel requirements (i.e., lower data rate, small message sizes, etc.) and the inherent redundancy and backup of directional comparison systems. Although the channel bandwidth requirements are less than those of current differential schemes, the communication channel data integrity requirements are significant. We may classify directional comparison pilot protection systems as blocking or transfer trip. This classification corresponds to the way the local relay uses remote terminal information to generate the tripping signal.

A current differential (87L) system is another popular pilot protection scheme. Such schemes compare the phase of the currents from all terminals. This means that 87L schemes require a reliable, high-capacity communications channel. When communication fails, the differential protection portion of these schemes must be blocked from operating. Today, many 87L schemes use redundant communications to handle the loss of a single channel.

Blocking Directional Comparison Systems

Blocking systems do not require the remote signals to trip. The relay will generate a trip if an overreaching protective element picks up and the relay does not sense a block signal for a settable short delay. Typically, these systems use powerline carrier for communicating the equivalent of one bit of information. Blocking systems tend toward higher dependability than scheme security; failure to receive a blocking signal from a remote terminal can result in a misoperation for an external fault. In such a case, the local relay operates dependably but scheme security suffers.

If we consider electronic security, we must consider the possibility of receiving artificially generated blocking signals. In the presence of a block signal, the directional comparison blocking scheme trips with a long time delay (i.e., 20 to 30 cycles). Standing or excessive duration block trip signals are not normal. We can monitor and alarm for such conditions.

Transfer Trip Directional Comparison Systems

Transfer trip systems must receive a signal from the remote terminal(s) to issue a local tripping signal. Applicable communications channels include audio tone, microwave, fiber optic, and spread spectrum radio. These schemes exchange a minimum of one information bit. Unlike blocking schemes, precise signal or message timing is usually not critical. The more time it takes for a relay at one end to receive notification that the remote terminal relay also senses a forward fault, the longer the tripping time. A slightly delayed trip during an internal fault does not constitute a misoperation.

Transfer trip system requirements tend toward higher scheme security. It is accepted that a failure to receive a valid tripping signal can result in a failure to operate at high-speed for an internal fault. Transfer trip pilot systems are typically faster than blocking systems (because they do not require a short delay to wait for the receipt of a block signal for out-of-section faults, and because they operate on the premise that the received data check is sufficient to ensure that the data are valid).

Communications Requirements of Directional Comparison Protection Systems

Figure 2 shows a schematic diagram of a modern directional comparison system. This system uses directional or directional-distance relay elements to discriminate internal from external faults. For an internal fault in the protected line, all relays see the fault in the forward (tripping) direction; for an external fault, one relay senses the fault in the reverse (non-tripping) direction.

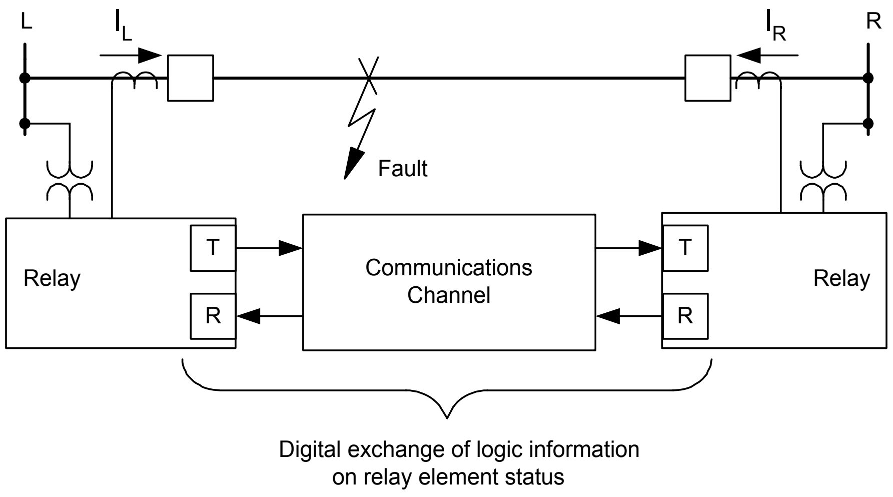

L
I_L
I_R
R
Fault
Relay
T
R
Communications Channel
T
R
Relay
Digital exchange of logic information on relay element status

Figure 2 Schematic Diagram of a Modern Directional Comparison System

Relays at all terminals require locally measured current and voltage information to determine fault direction. The protection system uses the communications channel to exchange only logic information about relay element and contact status. For the patented system shown in Figure 2, the relay interface to the communications channel is digital. With a slight increase in channel bandwidth, the protection system communicates the status of eight digital outputs and eight inputs. The advantage of this system is that it can communicate up to eight times the information (important for modern trip and control schemes) while simultaneously monitoring the health and availability of the communications channel.

An advantage of these tripping schemes is that channel time delay is not critical. A minor delay in receiving the remote signal in a permissive overreaching transfer trip scheme may delay tripping, but a reasonably short delay does not affect whether the trip or restrain decision is correct. In blocking schemes, you must know and design for your worst-case delay.

For these systems to be useful, the delays must be kept within reason. Reference [1] describes the number of bits required for the system in Figure 2 to meet the data and command integrity requirements of the IEC 834-1 standard. Table 1 lists the minimum number of noise bursts required to produce an undesirable output.

Table 1 IEC 834-1 Protection Scheme Security Requirements

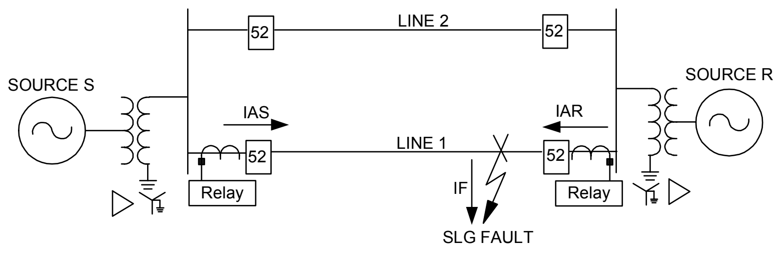
Scheme Type	Data Integrity/Security (bursts/undetected error)
Blocking	10^4
Permissive Tripping	10^7
Direct Transfer Tripping	10^8

As described in [1], transmitting eight data bits with the required integrity requires a total of 36 bits—24 bits for redundancy. In protection schemes, if the receiver detects one corrupt bit, it rejects the entire message. Following this condition, many receivers or relays require at least two consecutive good messages before accepting the messages as usable. These requirements increase scheme security but decrease dependability. For example, if the communications channel is noisy, a properly designed error detection scheme identifies each corrupt data packet and the relays do not accept the corrupted data. If the frequency of corrupt data packets is such that very few, if any, messages are being received as valid, the overall scheme is secure but we do not have a viable protection scheme. Before applying any such system, you should ensure that the communication scheme design meets the probability of receiving an unwanted command (PUC) requirements listed in Table 1. Increasing message length to improve data integrity or electronic security will increase security but may penalize operating speed.

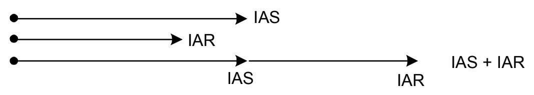
Line Current Differential Pilot Protection Systems

Line current differential schemes are a class of unit protection schemes. Each protective relay instrument combines the currents it measures with those currents measured and communicated by remote relays on the same power line.

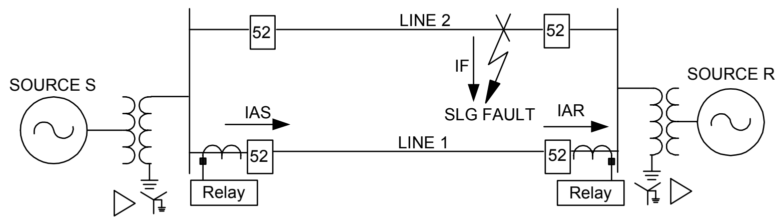
Figure 3 illustrates the basic principle of current differential protection for internal and external faults. Note that this basic principle is a simple application of Kirchhoff's Current Law. If the fault is internal, the vector sum of the currents at all line ends does not sum to zero. For external faults, the vector sum equals zero (ignoring the effects of line capacitance and current transformer saturation).



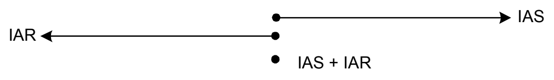
a. Single Line Diagram Showing Internal Single-Line Ground Fault



b. Faulted Phase Phasors for Internal Single-Line Ground Fault



c. Single Line Diagram Showing External Single-Line Ground Fault



d. Faulted Phase Phasors for External Single-Line Ground Fault

Figure 3 Basic Line Differential Principle

All of these schemes require current measurements from all line ends to make a trip decision. Data measured at each line end must be communicated to the remote terminal relays. Line current differential (87L) schemes rely on proper channel performance. Complete channel loss, poor data integrity, and excess delays can all defeat the 87L scheme security.

Line Current Differential Pilot Protection Channel Requirements

When a protective relay compares local samples or phasors with samples or phasors generated in a remote relay for the purposes of differential protection, the local relay must know the relative time between the local samples and the remote samples. Many 87L relays calculate this relative channel delay by using time information contained in the received data packets. Digital line differential protection systems typically use independent (non-synchronized) sampling clocks and align the computed phasors or resampled data. The universal technique for aligning data is known as the ping-pong technique. This technique involves measuring the time delay between the sampling pulses at two different locations. Once the relay establishes the delay, it time aligns the data by applying the appropriate compensation rotation to the received phasors.

The ping-pong technique assumes equal delay or symmetry in the communications channel. In some channels, the transmit path and the receive path have a different propagation delay. This asymmetric communication delay can exist, for example, on digital communications networks employing SONET self-healing ring technology. The level of asymmetry depends on the architecture of the communication system. The delay differences often range from 1 to 2 ms.

Communications channel asymmetries generate an error in the phase shift angle, θ, between local and remote phasors. Excessive phase error can result in unintended breaker operation or misoperations.

Unless the communications channel is direct fiber (i.e., the communications channel delays are very symmetric), we must select or design the 87L scheme to tolerate the maximum expected error.

As with directional comparison applications, you must ensure that the communications channel equipment has a minimum amount of buffering. In commercial multiplexers used to exchange non-time-critical files, the equipment buffers data until it receives an optimal amount. Such buffering could add an unacceptable delay.

The amount and type of data used for 87L schemes are much different from the contact equivalent data transmitted and used by the directional comparison schemes we described earlier. The communications also differ in that the 87L scheme requires time alignment of the data. A common data set used in modern 87L schemes includes three phase currents, timing data for measuring channel delay, miscellaneous control bits, and error detection bits.

SCADA/Metering

SCADA and metering applications are the backbone of the daily operation of the electric power grid. There is little difference between the two applications; both are characterized by the automated gathering of power system information. The main purpose of such systems is to collect data values from many remote points and to concentrate them at a control center where an operation function can act on them. The operation "function" is most often a visual mockup of the system intended to give human operators a view of the current grid status.

Most SCADA systems provide the additional capability of controlling various command points at these remote locations. System operators can then issue commands that operate breakers, enable or disable reclosing, add VAR support, and execute various logic functions (i.e., change the protection settings group).

Metering applications are often limited to gathering very accurate power usage measurements for the purposes of billing and power management. The metering devices do not generally offer the control capabilities required in a SCADA system. The amount of security required by a metering system may not be the same as that required by a true SCADA system (provided the two functions have separate and isolated communications infrastructures). The amount of security applied to a given communications infrastructure should reflect the importance of the function and the potential damage that can result from an attack on, or compromise of the system. One can argue that effective security measures are more important in a traditional SCADA system because of the added control capability and the importance of the SCADA function.

The flow of data in a SCADA system is almost always dictated by a communications protocol designed specifically for such application. The most common protocols are open (published) standards such as DNP 3.0, MODBUS, and IEC 60870[1]. Proprietary protocols are also supported

[1] Standard protocols are extensively documented. You can acquire details on all of these algorithms from a technical literature distributor.

in many remote terminal units (RTUs) and communications processors. Such protocols, however, are similar in function to the open standards. The vast majority of existing SCADA systems in the electric power industry are configured in a master/slave hierarchy. In such a configuration, a master device or a front-end processor in the control center polls the remote devices on a periodic schedule. The polls are requests for information in the form of network variable values or register contents held within the slave devices. Some protocols support "report-by-exception" as well. In such schemes, the remote devices transmit only the data values that have changed by some programmable amount. The variable updates are simply transmitted from a remote IED to the central SCADA "server" at the control center. In both cases, the SCADA servers (or master devices) typically are computers or mainframes, and the remote devices (or slave devices) typically are RTUs, communications processors, protective relays, or other intelligent electronic devices (IEDs).

Older devices often limit the capacity and speed of existing SCADA communications systems in the electric power industry. Figure 4 shows a typical SCADA architecture with some older equipment providing the network connections between the SCADA master and the remote IEDs. This scheme employs front-end processors to poll the RTUs with an agreed-upon SCADA protocol. Frequency shift keyed (FSK) modems internal to the front-end processors and matching modems internal to the RTUs communicate over 4-wire, twisted pair connections. The medium over which these analog signals travel may be implemented as a leased, permanent circuit, or as a stretch of wire owned by the utility. The limited processing power in these older RTUs and front-end processors, as well as the relatively low transmission rates of the internal modems effectively limit the rate at which the data is transmitted between the SCADA center and the remote IEDs. Rates of 1200 bps or less are common. A utility would have to replace the RTUs and the front-end processors to retrofit such a system. Extensive equipment replacement and system redesign can be extremely expensive. If neither equipment nor the system is broken, why fix it? For this reason, a large percentage of existing SCADA systems is limited to transmission rates that would be considered unacceptable for most networking applications.

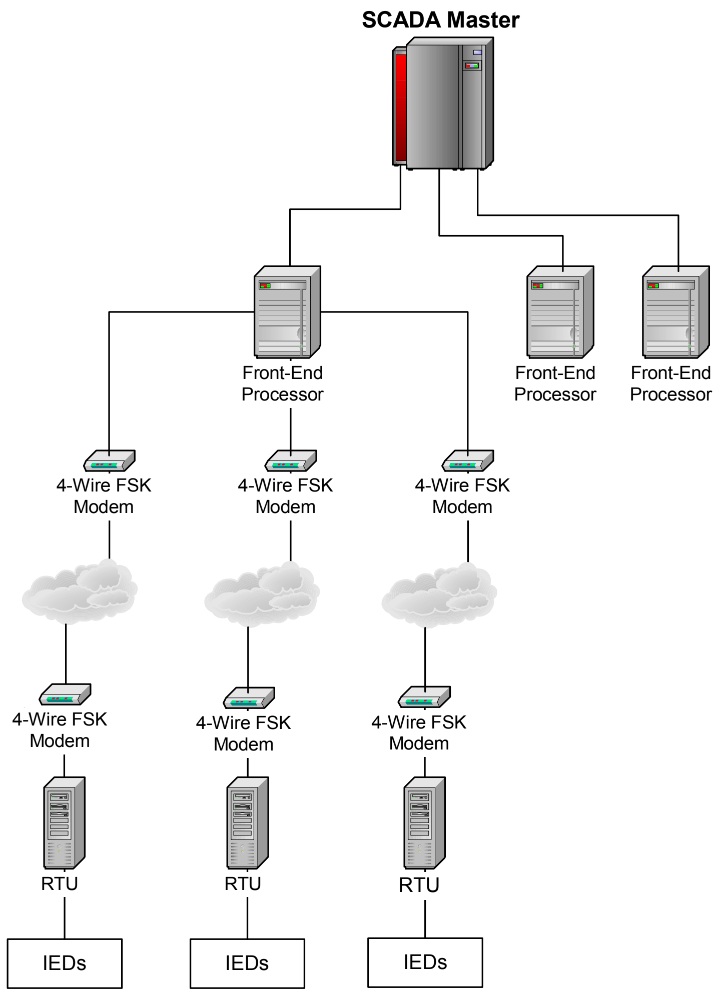


Figure 4 Traditional SCADA Communication System

The properties of the SCADA information transfer and existing communications channels will prove important in our discussions on how to secure these systems. The following observations apply to SCADA communication schemes:

- The information request packets transmitted by the SCADA master are typically shorter than the replies sent by the remote IEDs. In this sense, the communications load on a given network link can be quite asymmetric.
- There are seldom any provisions for communications security. Because the information transfer is an automated process, there is no concept of a user "login" inherent in the protocols. Furthermore, most protocols do not include digital authentication or encryption technologies.
- Polled, master/slave SCADA implementations can be quite sensitive to communication delays or latencies. The system information update rate depends upon the number of polled SCADA slaves and the time required to request and receive data from each slave device (the two-way transfer time). Clearly, any latency added to this two-way transfer time reduces system polling frequency.
- It is very common to have relatively slow communication paths between the SCADA center and the remote devices.

Remote Access

Remote access can be defined as any "user-driven" communication with a remote system device. The user/device interactions are often implemented through proprietary software applications. It is common to implement remote access to protective relays for the purpose of verifying device settings, checking device status, or downloading event reports. Engineering access to common protection devices is often an "all or nothing" prospect. In other words, remote operators often have the same capabilities they would have if they were accessing the device locally.

The following observations apply to remote access communications:

- Most security features found in IEDs are limited to user login or password entry schemes. The login dialogues are transmitted as plain text (ASCII code) for serial login or TCP/IP telnet sessions, and can potentially be intercepted and read from the communications media.
- Digital authentication and encryption are seldom used in existing relays, RTUs, or other IEDs.
- On-demand, remote access is often implemented with dialup modems using relatively inexpensive public telephone lines. Unprotected modems are much more common than password-protected modems, dialback modems, or cryptographic modems.
- Communication delays, within reason, are tolerable in remote access schemes.
- Message lengths vary greatly in remote access communications. A remote user is likely to send both short messages(user commands) and long messages (large settings file transfers). Similarly, the communications from the device to the user can also be both short (a returned command prompt) and long (a very large event report).

CRYPTOGRAPHY PRIMER

Modern communication schemes transmit data between physically separated devices. The media over which these data travel are often insecure. "Untrusted" individuals can access the media and intercept messages exchanged between legitimate system users, or inject data of their own in an attempt to gain information or access to devices to which they are not entitled. Cryptography moderates the risks posed by these insecurities.

Modern cryptography consists of functions and algorithms operating on data to provide the necessary security. The details of the functions and algorithms are published, so the security of the data relies on the secrecy of a key value rather than on the secrecy of the algorithms themselves. This allows the algorithm to be publicly scrutinized to discover any weaknesses in the underlying functions. The strength of a cryptographic algorithm is determined by the difficulty of deriving the secret key and reversing or bypassing the protection provided by the cryptographic functions. A secret key must take on a very large number of values to be resistant to guessing (brute-force) attacks.

At the most basic level, cryptographic functions provide data confidentiality; encrypted data must be decrypted for the information to be read. Some algorithms ensure data integrity by protecting against forgery or tampering. You can also apply cryptographic functions to provide authentication or proof of a message source.

ENCRYPTION

Data Confidentiality: Two parties should be able to send private messages over an insecure medium without exposing the contents of the messages to attackers.

Encryption is the process of transforming a digital message from its original form into a form that cannot be interpreted by an "unauthorized" individual. The output of the encryption process is a function of the message and an encryption key. This encryption process must be completely reversible by an "authorized" individual with access to the secret decryption key. Authority to read a message is only granted by sharing knowledge of the secret decryption key.

There are two main classes of encryption functions. Symmetric key encryption relies on the same secret key value to perform both the encryption and decryption transformations. The encryption function uses the key, K, to transform the digital message, M, into unreadable ciphertext, C. The decryption function uses the same key, K, to reverse the transformation and recover the original message. Public-key encryption functions, on the other hand, use different keys for encryption and decryption.

The one-time pad is the only encryption method that can provide near perfect security. The process involves generating a truly random encryption stream that is the same length as the plaintext. The encryption stream is then combined with the plaintext to produce the ciphertext. This method is not practical because it relies on the secure exchange of volumes of random encryption material between the parties involved in the conversation. Modern encryption functions must trade off security for convenience. For example, to transmit an unlimited amount of data between two individuals in a fairly secure manner (more secure than sending the plaintext message, but less secure than applying a one-time pad) they can securely exchange a relatively short encryption key and apply a modern cryptographic encryption technique.

Common Encryption Modes

Most encryption algorithms operate on blocks of data. These block encryption functions can be employed in many different ways. These "modes" are simply different ways to connect the inputs and outputs of the encryption function to get different system performance or data properties. There are several important factors, such as efficiency, fault tolerance, and relative security that must be considered when choosing an encryption mode.

Electronic Codebook Mode (ECB)

Electronic codebook mode is the simplest mode of encryption. For every block of plaintext, there is one corresponding block of ciphertext. With these pairs, it is possible to create a table of plaintext-ciphertext pairs, or a codebook.

Figure 5 shows the encryption process using the ECB mode. Each message, M_i is encrypted with the secret key, K, to produce ciphertext C_i.

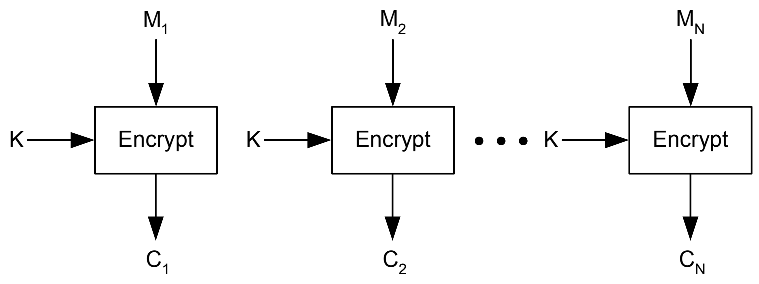


Figure 5 ECB Encrypt

This method is very fault tolerant. An error in transmission of the ciphertext (either by random sources, or by an intentional attack), will only affect the current data block.

A serious concern with the security of ECB is the possibility of a replay attack. For example, if attackers monitor the transmission channel and observe that a breaker trips every time a particular encrypted packet is sent to a device, they can capture the packet and retransmit (replay) it to the device to cause the same action.

Cipher Block Chaining (CBC)

A solution to the electronic codebook replay problem is to put a feedback mechanism into the block cipher. The results of the previously encrypted block are fed forward and combined with the input of the encryption of the current block. The encryption operation for CBC mode is shown in Figure 6. The first plaintext message, M_1, is XORed with an initial value and encrypted. The resulting ciphertext, C_1, is transmitted. Additionally, this ciphertext is XORed with the next plaintext message, M_2. This result is then encrypted to produce ciphertext C_2. This process continues for the entire session.

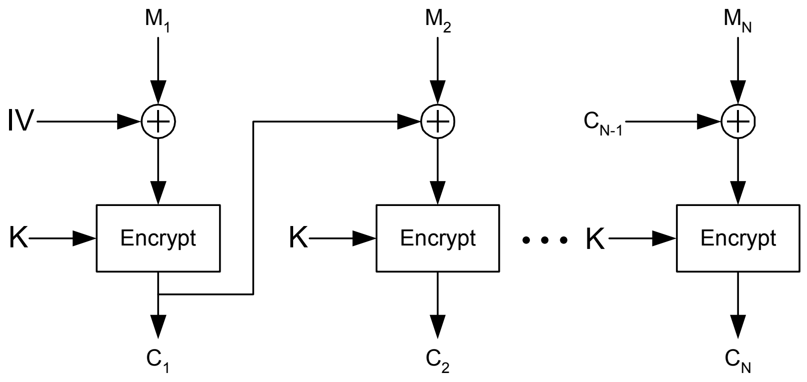


Figure 6 CBC Encrypt

Note the use of an initial value that is XORed with the first block of plaintext, M_1. Without a unique initial value, each session with an identical series of messages would encrypt to the same stream.

This mode suffers the same efficiency problems as EBC. As with EBC, an entire block must be buffered before it can be encrypted. This may introduce intolerable latency and overhead into the communications path.

The feedback characteristic of CBC causes a degree of fault intolerance. Transmission errors in the ciphertext propagate beyond the current block. Errors in the current decrypted block are fed forward to the next block. This causes the following block to be erroneously decrypted as well. Blocks after the second erroneously decrypted block are not affected, so CBC is a self-recovering mode.

Cipher Feedback (CFB)

A stream cipher generates a stream of bits that appear totally random. This stream is then XORed with a stream of plaintext data to produce a stream of ciphertext. The stream of pseudorandom bits, P, is produced using some known encryption algorithm. On the decryption side, an identical stream of pseudorandom bits is generated, and the ciphertext, data, C, is XORed again to recover the plaintext, M.

The strength of this cipher lies in the generation of the random bitstream. If the stream of bits is purely random, you have perfect security (known as the one-time pad). In reality, the security of stream ciphers lies between XORing with a repeating pseudorandom sequence and the one-time pad.

Cipher feedback mode is the first mode discussed so far that can be used to operate a block-based cipher as a stream cipher. Unlike ECB or CBC, which require an entire block before encryption or decryption can occur, CFB data can be operated on in any size (one bit, one byte, etc.) up to the block size of the base algorithm.

The CFB operation is shown in Figure 7. An initial value is first input into the encryption algorithm. The j most significant bits of encrypted output are XORed with j bits of plaintext, M_1, to product C_1. The remaining encryption output bits are discarded. The initial value is then shifted to the left by j bits, and the j-bit ciphertext, C_1, is inserted. The process repeats for each j-bit piece of the plaintext message.

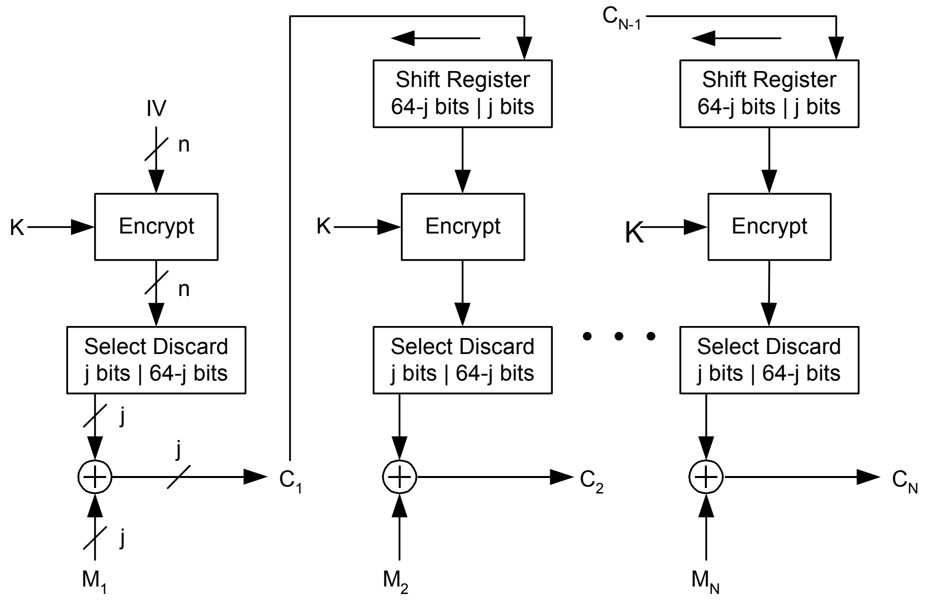


Figure 7 CFB Encrypt

The decryption process is identical to the encryption process in the sense that they both produce the exact same stream of pseudorandom bits.

Very little buffering (just *j* bits) is needed before encryption can occur, and the ciphertext is available soon after receipt of the plaintext. Furthermore, message padding is not required to fill an integral number of encryption blocks, so there is potentially no channel overhead associated with sending encrypted pad bits.

The fault tolerance of this system is better than that of CBC and only slightly worse than that of ECB. If an incoming *j*-bit block of ciphertext is corrupted, it not only affects the current plaintext output, but the next *n* bits (*n/j*, *j*-bit blocks) of plaintext as well.

Output Feedback (OFB)

Output feedback is another method of operating a block cipher as a stream cipher. It is almost identical to CFB, except that *j*-bits of the pseudorandom sequence are fed back into the encryption input. CFB, in comparison, feeds the ciphertext back (the pseudorandom sequence XORed with the message).

The security of this mode, however, suffers. Several results [2] [6] have shown that this mode should only be used when the feedback size, *j*, is the same as the algorithm block size, *n*. For less feedback, the pseudorandom sequence will repeat too soon and compromise the security of the data.

Symmetric-Key Encryption

The best known types of symmetric encryption are the Data Encryption Standard (DES), Triple DES (3DES), the Advanced Encryption Standard (AES), and Rivest Cipher (RC4). The former three are block ciphers (they can, however, be configured in the OFB or CFB stream cipher modes), while RC4 is a true stream cipher.

DES

DES is a block cipher function that takes in 64 bits of plaintext and outputs 64 bits of ciphertext. The key is 56 bits, although it typically is seen as 64 bits (the least significant bit of each byte is assumed to be parity). DES was adopted by the U.S. Government as a standard for the protection of data in 1976 and was later adopted by other standardization committees (i.e., ANSI and ISO). It has withstood many years of cryptanalysis and has been a worldwide standard for over 20 years.

DES uses two basic operations to transform the plaintext: confusion and diffusion. Confusion distorts the relationship between the plaintext and the ciphertext. The easiest method of doing this is substitution, i.e., replacing blocks of data with different data. The diffusion operation spreads redundancies in the plaintext across the ciphertext.

The real heart of the DES algorithm is the key-dependent transformation function (shown in Figure 8). It consists of three main components: the expansion-permutation box (E-box), substitution box (S-box), and permutation box (P-box). These three components form a single round of the DES function. There are 16 such rounds.

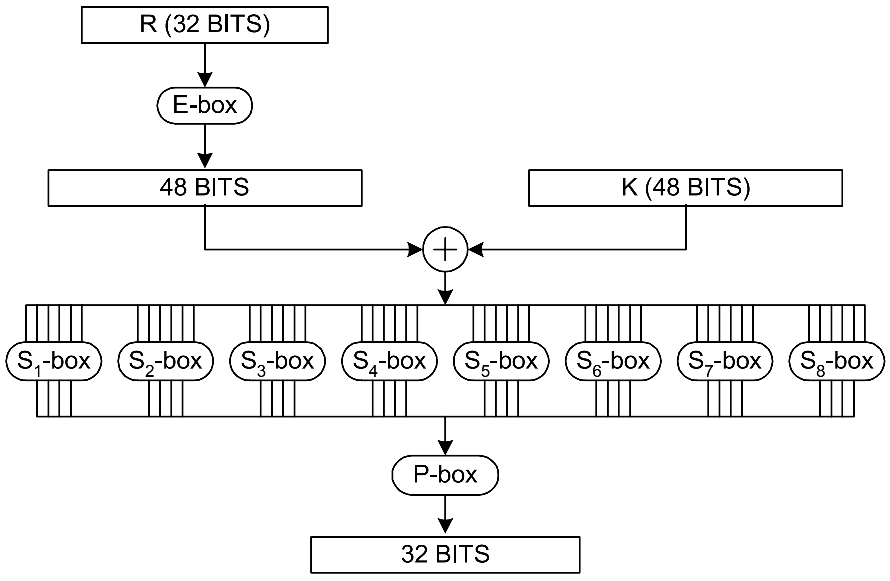


Figure 8 Diagram of a Single Round of DES

The expansion-permutation not only shuffles the bits, but also duplicates some. This transforms each 32-bit input into a scrambled, 48-bit result. The P-box is a simple bit shuffling or permutation. It does nothing more than move bits around. However, the positioning of the bits in the shuffling done by the P-box was carefully designed and far from arbitrary.

The complex part of the design is in the S-boxes. After public analysis of the DES algorithm, IBM published the design criteria for these boxes. A significant amount of effort went into the development of the S-boxes in the DES standard. Now, it is fairly simple to compute S-boxes to meet these criteria, but in the 1970s when DES was developed, it was much more difficult.

For decryption, the operation runs in reverse. The ciphertext initially enters the inverse permutation. The data then flows through the 16 rounds with the key schedule applied in reverse.

The security of DES has been heavily scrutinized over the years, and while an aged algorithm, it has proven quite effective. Even the most successful attacks require 2^{43} known plaintexts. This has been accomplished in 50 days using 12 HP9000/735 workstations. The algorithm itself is very strong, but the small, 56-bit key is susceptible to brute-force attacks. Frequent key exchanges can easily defeat this attack.

3DES

Triple DES (3DES) is a chaining of DES that uses three DES encryption/decryption units to provide a greater degree of security. Applying DES three times, with three different keys, increases the effective key length. One might think that this makes the 3DES algorithm three times more resistant to brute force key guessing attacks than the original DES algorithm. In reality, there are attack techniques that reduce the effective key length of this scheme to about twice that of single DES [2].

AES

The Advanced Encryption Standard (AES) was developed as a replacement for DES and 3DES. It supports key lengths of 128, 192, and 256 bits and a variable block length. AES is based on the Rijndael encryption algorithm, developed by Joan Daemen and Vincent Rijmen.

The Rijndael algorithm is a symmetric block cipher that supports block sizes of 128, 192, and 256 bits. It is highly efficient and well suited to software implementations.

During the evaluation of candidates for the AES standard, Rijndael was analyzed by some of the world’s best cryptanalysts. It has proven to be very effective against known attacks, very efficient, and simple to implement.

RC4

RC4 is a stream cipher developed by Ron Rivest for RSA Data Security Inc. It was a proprietary algorithm until 1994 when an anonymous person posted the source code on the Cypherpunks mailing list. Current users verified the compatibility, and RC4 became a publicly analyzed algorithm.

RC4 was initially designed to operate in software. It operates in OFB mode so the pseudorandom sequence can be pre-generated. It is also very fast (about 10 times faster than DES).

RC4 does have some security vulnerabilities. There is a known bias in the output of the first few bytes. With this statistical aberration, it is possible to glean information about the key from the first few bytes of the pseudorandom sequence. The severe security flaws in the IEEE 802.11 wireless LAN standard [4] were a direct consequence of this weakness in RC4 and some poor security protocol design principles on the part of the 802.11 standards committee.

Public-Key Encryption

Public-key encryption algorithms are based on the concept of a "one-way" function. Such a function has the following properties:

- Given x, it is easy to compute f(x).
- Even with all of the knowledge required to compute f(x) from x (i.e., the "forward" direction), it is extremely difficult to compute x from f(x).

This is in stark contrast to the properties of the symmetric encryption algorithms previously outlined. Namely, if an individual has the knowledge required to perform the forward computation (encryption), it is trivial to perform the reverse computation (decryption). This is a direct consequence of the fact that the keys necessary to perform the forward and reverse transformations are one and the same.

For public-key encryption, on the other hand, the knowledge necessary to perform the forward transformation does not make the reverse transformation computationally feasible. Clearly, if the one-way function is to be useful in an encryption scheme, the legitimate receiver must be able to calculate the reverse transformation in a timely manner. Fortunately, there are several classes of functions that contain mathematical "trapdoors". A trapdoor, in this context, is a piece of secret knowledge that makes the reverse transformation mathematically tractable.

In the context of public-key encryption, the knowledge required to perform the encryption is known as the public key, and the trapdoor required to perform the decryption is known as the private key. The public and private keys are mathematically related to each other through the one-way function that defines the encryption scheme. All one-way functions used to define public-key encryption schemes have the desirable property that knowledge of both the public key value and the encrypted message does not make recovery of the private key or recovery of the original message (the plaintext) computationally feasible. The advantage of this lies in the ease with which one can acquire a key and initiate communications with another individual. The security of such a scheme is not jeopardized by freely disseminating your public key and, in essence, allowing many people to securely communicate with you. This is in stark contrast to the key dissemination methods required by symmetric key encryption algorithms. To ensure that communication between any two parties is secure (i.e., only the two parties involved in the conversation can decrypt the traffic), each pair of individuals must have their own distinct encryption/decryption key. Clearly, the number of keys required to allow secure communication among many individuals can become extremely large (it grows approximately as the square of the number of individuals involved [7]).

The concept of public and private keys, mathematically related to one another through a one-way function, is perhaps best illustrated by detailing a representative public-key encryption scheme. One of the first and most widely used public-key encryption schemes is the RSA algorithm, invented by Ron Rivest, Adi Shamir, and Len Adleman of MIT in 1977 [6]. The encryption and decryption procedure is as follows:

1. Choose two random, very large prime numbers[2], p and q.
2. Calculate the product, $n = pq$.

[2] A prime number is an integer that is divisible only by itself and one (1, 3, 5, 7, …).

3. Choose a random encryption key, e, such that e and the product, (p-1)(q-1), are relatively prime[3].

4. Calculate the decryption key, d, such that ed = 1 mod(p-1)(q-1). Where mod is the modulus operator[4].

5. Define the public key as the pair, n and e, and the private key as the pair, n and d. The numbers, p and q can now be thrown away, but must not be revealed.

6. Define the message as a number, m, less than n (m can be the decimal representation of the group of bits that make up the message). If m is larger than n, break up the message into smaller blocks such that all blocks have a value less than n.

7. Calculate the encrypted message, c, as $c = m^e \bmod n$.

8. Decrypt the message as $m = c^d \bmod n$.

The ability to encrypt the message and, upon receipt by the legitimate recipient, decrypt it and recover the original message is granted by the theoretical properties of modular arithmetic. The strength, or equivalently, the security of the above process is determined by the difficulty of recovering the private key, *d*, given knowledge of the public key pair, *e* and *n*. It is widely accepted that, in order to do so, an attacker would have to successfully factor *n*. So far, the world's mathematicians have not produced an efficient algorithm for factoring large numbers. The level of effort required to factor a 512-bit (154-decimal digit) integer is almost out of reach of the most advanced distributed computers available today. It is believed that making *n* a 1024-bit (308-decimal digit) integer will provide sufficient security for many years to come.

There are several other variants of the public-key encryption concept. All are based on the theory of one-way functions with trapdoors.

Comparison of Public-Key and Symmetric Key Encryption

In the previous section, we outlined some of the benefits of public-key encryption algorithms. In particular, it is very convenient to be able to establish a secure channel of communication without the need for a secure key delivery method. One may wonder why we do not rely entirely on these public-key encryption methods. The short answer is that public-key encryption methods are much more complex than comparable symmetric key methods. RSA, for example is about 1000 times slower than DES [2]. This is partly a result of the fact that secure key lengths for public-key algorithms are about 100 times longer than comparable-strength symmetric keys. It is also a result of the fact that the mathematical operations required to implement the popular flavors of public-key encryption are much more complicated than those required for popular symmetric-key algorithms.

We mentioned the difficulties associated with managing key pairs for multiple symmetric-key "conversations". Fortunately, these problems have a solution in the form of hybrid cryptosystems. Most modern encryption schemes (at least those that require communication between many individuals) use a combination of public-key encryption and symmetric-key

[3] Two numbers, *a* and *b*, are relatively prime if their greatest common denominator is one.
[4] A number, *n*, can be represented as a product of two integers, *k* and *d*, plus a remainder, *m* (i.e., $n = k*d + m$). The remainder, *m*, resulting from division of *n* by *d*, is represented in shorthand by the modulus function, $m = n \bmod d$. For example, 5 mod 3 = 2.

encryption to solve these key management and dissemination difficulties. In such systems, the public-key algorithms are used to securely transmit keys for a symmetric-key encryption algorithm. For example, RSA may be used to transmit a new, randomly generated 3DES key on a periodic basis. The much faster 3DES encryption would then be used to encrypt the data between the two individuals. These hybrid schemes allow for automated, periodic exchanges of symmetric keys (dynamic key exchange) with protocols like the Diffie-Hellman key exchange [2].

ONE-WAY HASH FUNCTIONS AND MESSAGE AUTHENTICATION CODES

Message Integrity: An intruder should not be able to alter a legitimate message or to substitute a false message in its place.

There are many cryptographic applications that require protection against message alteration. Imagine if an attacker could change the monetary value in a digital bank deposit message from \$1,000 to \$100,000. A more relevant example would involve an attacker altering a file containing the configuration or settings of a very critical protective relay. How would you know if a few of the bits in the file were changed (i.e., the overcurrent element pickup threshold)? Fortunately, there are cryptographic algorithms that can be used to create a unique signature for a given message. The two main classes of algorithms that you can employ to create a message "fingerprint" are one-way hashes and message authentication codes (MAC). Both classes perform the same basic operation, but there are subtle differences:

- The only input to a one-way hash function is the message itself. Hash algorithms do not utilize a secret key. Because of this, they will always produce the same output for a given, fixed input.
- A true MAC function has two inputs: the message itself and a secret key. This gives the MAC algorithms a certain degree of built-in authentication capability.

Both the hash and MAC functions take a variable-length message as input and generate a fixed-length hash value. The hash is then a condensed fingerprint or signature of the message input. If someone changes the contents of the message, the hash value appended to the message would not match the value that would result if a new hash value were calculated over the new, altered message. If the hash function were keyed (i.e., a MAC), then an attacker would be unable to recalculate a new, valid hash value over the altered message and hide the fact that the message has been altered. The same data protection can be obtained by encrypting the output of a non-keyed hash with an encryption algorithm.

The output of a hash (or MAC[5]) function is very different than that of an encryption function. The former is a *fingerprint* of the original message, not a *representation* of the original message. In other words, the hash value does not preserve all of the information in the original message and, because of this, it is not possible to reconstruct the original message from the hash. To produce a secure signature of a message, a hash function must have the following properties:

- Given a hash function, $H(m)$, and its output, h, it is extremely difficult to derive a message, m, such that $H(m) = h$.

[5] Because hash functions and MAC functions both generate a hash value, we can refer to both as hash functions. We must keep in mind, however, that some hash functions are keyed (MAC functions) and some are not.

- Given a message, *m*, it is extremely difficult to find another message, *m*', that produces the same hash output.

The first condition states that the output of a hash function should not give away any clues about the form, or classes of messages, that would likely produce the same hash value. The second condition, known as collision-resistance, states that there should not be any bias in the mapping of inputs to outputs that would aid an attacker in finding messages that produce identical hash values. Both conditions simply make it functionally impossible (given all realistic resources) to alter a message in such a way as to produce the same hash value. Such an alteration would not be detected by many cryptographic protocols that utilize hash functions. Clearly, the length of the hash value generated by a given function must be quite long to ensure reasonable security. It is generally accepted that a "good" hash function should produce at least 128-bit hash values. For truly critical applications, the hash values should be at least 160 bits long [2]. This observation will prove very important in our discussions on the feasibility of applying authentication algorithms in the electric power industry.

Two of the most common one-way hash functions (non-keyed) are MD5 and the Secure Hash Algorithm (SHA). Both algorithms operate on the stream of data to diffuse the message information throughout the eventual output hash value (128 bits for MD5 and 160 bits for SHA). The design of a good hash algorithm is such that a small change in the message will produce a very significant change in the final hash value.

AUTHENTICATION SCHEMES

Authentication: It should always be possible for a receiver to unambiguously determine the origin of a message.

The concept of authentication is as important as the concepts of message integrity and confidentiality discussed in the previous sections. Modern communication networks are often quite susceptible to malicious traffic. It is highly desirable to be able to guarantee that you can detect and ignore all traffic that does not originate from a trusted individual. In this section, we will draw upon the concepts and algorithms described in the previous sections to describe effective methods of proving the origin of a message.

Cryptographic authentication schemes are based on the assumption that a "trusted" individual has possession of a piece of secret information that only he and other trusted individuals share. In theory, an effective authentication scheme will make it obvious when a potential attacker lacks knowledge of this secret. The symmetric-key encryption algorithms discussed above are capable of providing a very rudimentary level of authentication.

Two parties, Alice and Bob, can communicate securely with symmetric-key encryption only if they both possess the key value. Assume that all traffic between Bob and Alice is encrypted prior to transmission and decrypted upon receipt. Now assume that a third party, Eve (potentially an attacker), without knowledge of the secret key, transmits a message to either Bob or Alice. After decryption, the message from Eve will appear to be random garbage. This will likely be true for all messages that were not encrypted with Bob and Alice's shared key. Figure 9 illustrates this concept. It is then safe to assume that any message received by Bob (Alice) that is legible or consistent with the expected message format, must be from Alice (Bob). This scheme, though simple, is not foolproof. If an attacker were to send random messages to either Alice or Bob, there is a chance that one of those messages will decrypt to something meaningful. This is especially true if the messages expected by Alice or Bob are typically short.

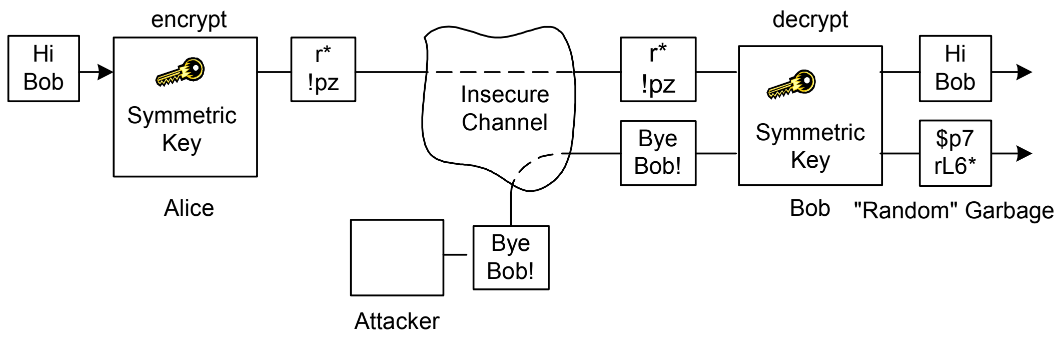


Figure 9 Symmetric Key Encryption Provides Rudimentary Authentication

When used in the “traditional” manner, public-key encryption does not provide any authentication functionality. Everybody with access to your public key can send you encrypted messages. Furthermore, in the absence of additional information inside the message itself, there is no way to distinguish the origin of the message from among everyone with knowledge of the public key.

We can, however, operate the algorithm “in reverse” by encrypting messages with the private key, and decrypting them with the public key. There is no distinction between the two values other than the fact that one is considered public knowledge and the other is a tightly guarded secret. It is extremely important to note that if you operate the public-key algorithm in reverse, you lose all data confidentiality! If you encrypt the data with a private key, anybody with access to the public key (potentially many untrusted individuals) can decrypt the message. It is, however, a very powerful method for implementing strong authentication schemes.

Imagine that Alice wants to know if it is actually Bob on the other end of the channel. She can simply ask Bob to identify himself. Bob can then encrypt a unique message with his private key and send it to Alice. Alice can then decrypt the message with Bob’s public key and, if the decryption process does not garble the message, she knows that, with high probability, Bob must have sent it.

Because of the complexity of public-key encryption algorithms, it is undesirable to encrypt large amounts of data with them. Consequently, it is not feasible to add authentication to all messages that Bob transmits by encrypting each one with his private key. Furthermore, if Bob demanded data confidentiality for all of his messages, he would have to apply yet another encryption algorithm to the already encrypted message. This is because, as stated before, everyone with Bob’s public key can decrypt the message unless another encryption scheme is used to reencrypt the message with a key that is independent from the one used for the authentication coding.

It is extremely convenient to enable anybody to verify your identity without compromising the security of the authentication process. This is the benefit of using public-key algorithms to formulate a strong authentication scheme: anybody with access to your public key can verify your identity. We can use the public-key algorithms in conjunction with other cryptographic algorithms to provide extremely convenient and strong authentication schemes.

One of the most common methods of authentication involves combining public-key encryption with a non-keyed hash function. First, you calculate a hash value, *h*(*m*), from the message that requires authentication. You then encrypt the hash value with your private key and append the result, called a **digital signature**, to the original message.

$$m \,||\, E_{k_{private}}[h(m)]$$

Again, we can add digital confidentiality to this scheme by encrypting the message (most likely with a symmetric-key encryption algorithm) either after the digital signature calculation:

$$E_{k_2}[m \,||\, E_{k_{private}}[h(m)]]$$

or before the digital signature calculation:

$$E_{k_2}[m] \,||\, E_{k_{private}}[h(E_{k_2}[m])]$$

There are several common variations on this theme, but almost all modern message authentication schemes involve appending additional information to the end of the original message. All of the above techniques make it extremely difficult to alter the message in transit without that change being detected. They also provide strong authentication or proof of message origin.

Existing Security Solutions (COTS Cryptography Devices)

There are many commercial off-the-shelf (COTS) products available that provide strong cryptographic functions, including encryption and authentication. Several commercial products provide the ability to protect traffic with a "bump-in-the-wire" solution. Such devices can simply be placed on either end of an existing communications link. They then secure the data on the link while (ideally) appearing transparent to the existing system.

SafeNet, Inc. provides simple bump-in-the-wire solutions for serial, frame relay, and ATM communications. Their LSA Encryptor provides a solution that implements 3DES encryption on EIA-232 serial communications with support for dialup modems. It is byte oriented, meaning that for every byte it receives, a byte is immediately transmitted. Because 3DES is a block algorithm, the SafeNet device most likely uses the CFB encryption mode. This raises the issue of error propagation. A byte-based use of CFB for 3DES causes errors of up to 9 bytes for each error in the received ciphertext.

Another option involves the use of known encryption and authentication standards for TCP/IP, such as IPSec or SSL. These are well-defined and mature options that have numerous implementations. They have been analyzed extensively and have been shown to provide strong security. Linksys, Inc. provides an inexpensive Virtual Private Network (VPN) solution that encapsulates data and transmits it across an insecure TCP/IP network. There are many serial-to-Ethernet transceivers on the market that will convert an existing serial (EIA-232) data source into TCP/IP over Ethernet. The converted packets can then be secured with a VPN solution. The reverse operation at the other end of the data link will return the secure (encrypted and authenticated by the IPSec protocol) TCP/IP packets back to the original serial signal.

For analog communications over the public switched telephone network (PSTN), Western DataCom, Inc. offers analog modem solutions that include 3DES encryption. A secure, encrypted channel is created between two CryptoCom devices programmed with the same symmetric 3DES key. These devices operate like any other modem, but with the added features of encryption and rudimentary authentication (authentication through symmetric key encryption).

To determine the impact of these devices on a serial communications system, we measured the latency added by inserting the devices into the communications path. The architecture of the measurement setup is shown in Figure 10. In the setup we have two systems, represented as PCs in the figure, communicating over a serial (EIA-232) connection. Any network devices required to implement and/or secure the connection are placed between the two data sources. We then tap data entering the first network device on one end of the connection, as well as data exiting the last network device on the other end of the connection. The time difference between the entry and exit from the devices is the overall system latency. For comparison, we recorded the latency measurements for network connections with and without cryptographic security.

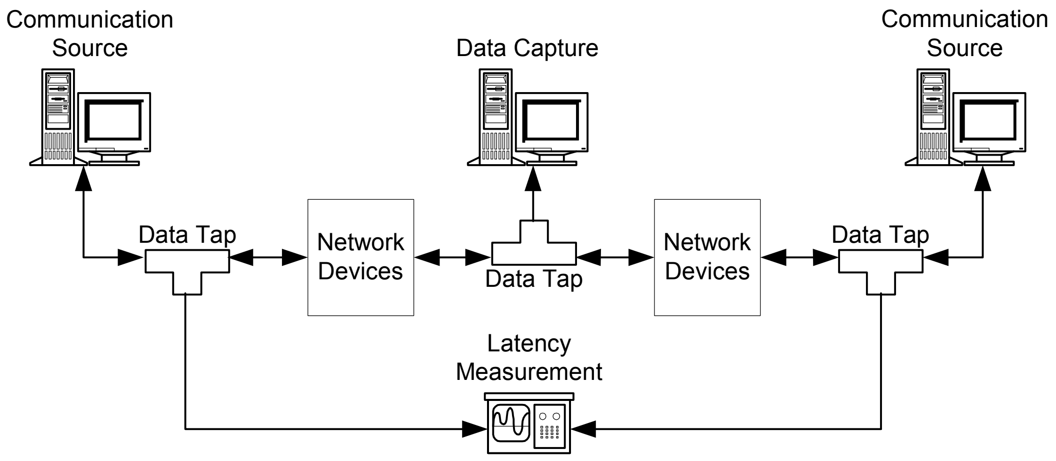


Figure 10 Latency Measurement Setup

We measured two cryptographic systems to determine the additional processing time required to perform basic encryption operations. The two systems were the CryptoCom 2000 encrypting modem from Western DataCom, and a Linksys Virtual Private Network (VPN) system with two SEL-2890 serial-to-Ethernet converters. Overall, it can be seen that the cryptographic solutions presented impose very little overhead.

The first system is based upon the CryptoCom 2000. As a baseline metric, we compared the performance to two non-encrypting US Robotics 56k modems. For both schemes, the modems communicated through a phone line simulator. The recorded system latency is the sum of the delays added by the two modems, and that of the analog phone line simulator. The graph in Figure 11 shows the measured system latency versus the transmission rate of the modems. The left half of the graph (2400 baud to 19200 baud) shows a dramatic increase in system latency as the rate decreases. This is largely a result of the added time required to pull the byte off of the line. For example, it takes 4.16 ms to transmit a single byte on a 2400-baud line (8 data bits plus a start bit and a stop bit). This delay would be added in both modems (the data byte must be received and buffered before it can be processed), for a total minimum delay of over 8 ms. This minimum delay would be reduced to just over 1 ms for a 19200-baud line. The fact that there is more than the expected 7 ms of added delay on the 2400-baud data point versus the 19200-baud data point is most likely a reflection of the internal hardware delays that also vary with baud. An example would be a forced delay of five character time intervals to wait for additional data prior to further processing (i.e., to gather as much data as possible before further processing in order to increase hardware efficiency). The graph clearly shows that the latency added by the *non-encrypting* modems is more than that added by the CryptoCom *encrypting* modem. The additional latency is most likely the result of older, slower hardware in the US Robotics modems. The newer CryptoCom is simply able to perform the standard modem operations (compression, modulation, etc.) and the encryption operations faster than the time it takes the older US Robotics device to perform just the standard modem operations (without encryption).

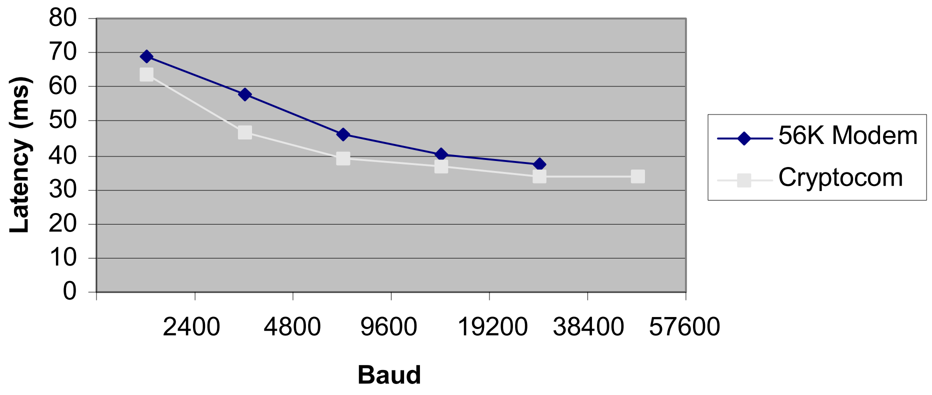


Figure 11 Modem Latency Measurements

For the second system, we used consumer-class VPN devices from Linksys to secure the link. These are Ethernet based, so we used the SEL-2890 to convert the data between the serial and Ethernet formats. Figure 12 shows the total latency for the system with and without the VPN devices. The system without encryption (no VPN) consisted of just two SEL-2890 Ethernet transceivers connected by a high-speed (10/100 Mbps) twisted pair cable. The VPN devices were simply added between the serial-to-Ethernet transceivers to form the system with strong electronic security. The graphs show that the addition of the VPN devices introduced very little additional latency (1–2 ms). The VPN device that we tested was configured to implement strong authentication with a keyed hash (MAC), and 3DES encryption of the data packet. The calculations required to carry out these complex cryptographic calculations took less than 2 ms per packet. It is important to note, however, that the delays associated with the transmission of the authentication hash bits (over 128 bits per packet including the overhead of the authentication header) are not accurately reflected on the graph. The Ethernet interface is very high speed, so the delays associated with the transmission of 128 additional bits per packet is negligible. This will not typically be the case for a real-world, wide area network connection (i.e., for SCADA communications or remote access). On such systems, it is possible to turn the authentication protocol off on the VPN device and rely on authentication through symmetric key encryption.

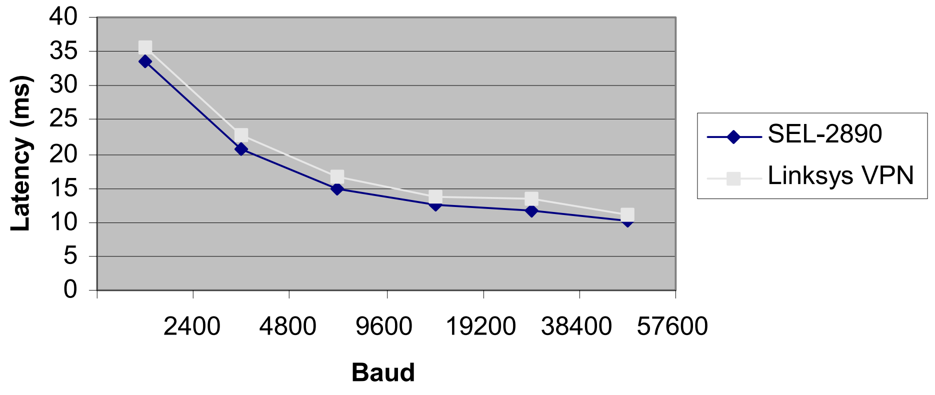


Figure 12 VPN Latency Measurements

Application of Cryptographic Technologies within the Electric Power Industry

Real-Time Protection Communications

There are two main questions that must be answered when identifying potential security solutions for a given communication system. First, we must analyze the *need* for security on the system. This is the process of identifying which, if any, cryptographic functions we would recommend for the system. Second, we must analyze *feasibility* of the identified cryptographic functions given the properties of the communications channel and the requirements of the protocol or function that is implemented over the channel. We analyze the need for, and feasibility of, cryptographic technologies in common real-time protection schemes below.

We introduced directional comparison and current differential protection schemes earlier in this paper. Both types of these schemes are characterized by the need for a reliable communications channel between the line-end devices. It is not necessary to extend or network the connections to any other devices. In practice, the majority of these communications channels are deployed on wholly owned (i.e., not leased from a telecomm provider) media such as fiber or the power line itself. Because of this, most real-time protection communications have very limited exposure to potential electronic attack.

Assuming that attackers are able to access the communications media (either electronically or physically), they could potentially execute the following general attacks:

- **Denial of Service (DOS):** Cause a break in the normal transmission of real-time protection messages.
- **Traffic Manipulation (TM):** Intercept legitimate traffic and/or inject malicious traffic on the line.

The effect of a DOS or TM attack depends upon the type of protection scheme. Table 2 shows the action and results for the various schemes we discussed in this paper.

Table 2 Electronic Attack Result Summary

Scheme	DOS		TM	
	Action	Result	Action	Result
Blocking	Block any Block Trip (BT) Signal	Out-of-section (OOS) fault Overtrip[1]	Cause a Standing BT Signal	Time-Delayed Trip[1] for In-Section Fault
Permissive	Block Permissive Trip (PT) Signal	Time-Delayed Trip[1]	Cause a Standing PT Signal	Overtrip[1] for OOS fault
DTT	Block DTT Signal	No Trip	Send DTT Signal	Trip
87L	Disrupt Communications	No Trip	Alter or Delay Transmitted Data	Trip[2]

[1] Only if local fault-detecting elements pick up, i.e., no trip unless an actual fault occurs at attack time.

[2] Assumes time delay greater than the maximum design limit or that altered data would cause a trip.

The blocking and permissive trip protection schemes provide high immunity to any potential attack damage (it is simply not possible to cause a severe misoperation through manipulation of the communications channel). For the direct transfer trip (DTT) scheme, we can eliminate the possibility of tripping the local breaker with local supervision. Examples of local supervision are overcurrent, undervoltage, power, and rate-of-change elements. Finally, for 87L protection schemes, you can eliminate the loss of line protection resulting from channel failure (either accidental or deliberate) with effective backup communications and protection schemes.

Earlier, we noted that 87L schemes are extremely dependent upon communications: a DOS attack on a line current differential scheme does disable the primary, 87L protection on the line. However, many schemes include true hot-standby 87L communications and directional comparison protective schemes in the same device. Thus, in the event of an attack, the complete scheme would disable one of the 87L schemes and alarm, yet line protection would remain intact. It is possible, however, to initiate a false trip for DTT (without supervision) and 87L protection schemes with a TM attack. This may not be a cause for concern because of the limited exposure of most real-time protection communications.

The limited risks outlined above may warrant additional electronic security if the communications channels used to implement a DTT or 87L protection scheme are not “sufficiently” secure. Such a decision can only be made by weighing the potential costs of an inadvertent breaker trip versus the risk of electronic attack.

It may be possible to implement cryptographic solutions on real-time protection communications links. Such a solution would have to be very high-speed (low latency) and highly deterministic (predictable, fixed latency). Furthermore, the scheme must have a negligible impact on the system reliability. It is not important to provide encryption because the data does not contain sensitive information. The system may, however, benefit from authentication to protect against malicious data tampering and/or injection.

The stringent latency requirements can be satisfied with an efficient hardware implementation of the cryptographic functions and a reasonably high-speed communications link. As we pointed out earlier, the system latency is highly dependent on the speed of the link, the amount of data overhead added by the cryptographic functions, and the amount of data that the algorithm must buffer prior to performing the cryptographic calculations. It is probably not possible to add hash-based authentication to existing real-time protection links because of the amount of overhead that such schemes would add to the data stream. We may, however, be able to use a symmetric-key encryption scheme to provide some authentication capability without adding communications overhead. A likely candidate is a symmetric-key encryption algorithm operated in one of the streaming cipher modes (OFB or CFB). These feedback schemes are also necessary to protect against possible replay attacks: the repetitive, small blocks of data transmitted by real-time protection schemes may be susceptible to such attacks. The error propagation caused by the use of these schemes may be a problem on noisier channels, but the effects can be mitigated with the proper addition of error correction and/or detection bits (with a small increase in data overhead).

It is important to point out that a real-time protection scheme may not support dynamic key exchange protocols because of the periodic addition of large amounts of system latency during key exchanges. The protection scheme may degrade in performance or even fail during such exchanges.

SCADA Communications

The control capability that SCADA systems provide is essential for the safe and efficient operation of our electric power grids. The system-wide monitoring and control functions provided by such systems may make them an attractive target for electronic attack. Furthermore, the complexity of the network architectures required to connect physically separated sites creates the potential for numerous electronic access points. As stated before, the level of electronic security required on a given communications infrastructure must be determined by analyzing the potential damage that can be caused by unauthorized access to the system, as well as the vulnerability of the system to such unauthorized access. At the very least, a complex SCADA system has many potential points of electronic access. The relative security of each of these access points varies widely depending on the technology used to implement the network connection. Wholly owned communications media can be considered relatively secure compared to leased lines or wireless network links. A large SCADA infrastructure may contain a mixture of these technologies and thus, a mixture of potential levels of vulnerability. We have already discussed the implications of a successful security breach of a SCADA system. All SCADA protocols, by definition, provide the means to operate connected devices. For electric power protection devices, this capability includes the operation of breakers and potential destabilization of the power grid. It is therefore important to analyze our SCADA systems and apply additional electronic security measures wherever potential vulnerabilities exist.

There are three main avenues of electronic access:

1. Access through dedicated SCADA equipment
2. Access from internal non-SCADA networks
3. Access from external non-SCADA networks

The first category involves unauthorized access to the equipment that implements the SCADA function (HMI workstations, communications processors, RTUs). The security of these devices can be controlled by limiting physical access to the devices and by implementing strong electronic access policies (passwords, biometric authentication, access logging, etc.) [7].

The second category involves any connections that may exist between the SCADA network and the other functional networks within the company. An "internal" network is one that is under the full or partial control of the same company that controls the SCADA system. Examples include connections between the business LAN and the SCADA network, or even connections provided to third-party companies for the purpose of contracted system maintenance. Securing such connections, or ensuring that they do not exist in the first place, is traditionally the responsibility of the corporate Information Technology (IT) staff. If such connections must exist, it is essential that the system be treated as a whole. Separation of jurisdiction between the connected networks can lead to security blind spots that can jeopardize the critical SCADA infrastructure. It is important to remember that many of the networks within a corporation are much more exposed to electronic attack than the SCADA network itself. Direct, unprotected connections to such networks should be avoided.

The third category, and focus of this paper, involves an attack on the SCADA infrastructure from an external network. Our primary concern involves the injection of malicious packets onto the SCADA network. If access to the SCADA equipment itself (the first attack avenue) is sufficiently secured, then this can only occur if an intruder gains access to the SCADA communications media itself. Likely scenarios include electronic compromise of the leased telecommunications network (hacking) or physical compromise of the transmission media (wiretapping or manipulation of wireless traffic).

SCADA communications, like real-time protection communications, do not necessarily require encryption. It is, however, hard to argue against the potential benefits of authentication. The ability to detect and reject SCADA traffic that does not originate from a "trusted" source would drastically reduce the risks posed by a traffic manipulation or injection attack. For slow communication links, we can employ the same techniques proposed for securing real-time protection communications. More robust protocols, such as dynamic key exchange, may be added if sufficient communications bandwidth exists and if the additional system latency is tolerable. Again, it is important to point out that an effective encryption/authentication scheme for SCADA communications will likely include some kind of feedback mode to reduce the effectiveness of a replay attack (SCADA protocols are highly repetitive).

Remote Access Communications

In the electric power industry, many of the devices that have remote access capability are performing essential protective functions. Furthermore, it is usually possible (with the right passwords) to gain full access privileges over these remote links. Such privileges grant a user the ability to change protection settings and operate connected devices (i.e., breakers). Effective security is very important on remote access communications links because of the potential damage that can be caused by misuse of these privileges. Now we must ask ourselves how vulnerable these links are to unauthorized electronic intrusion. Clearly, this depends on how the specific remote access link is implemented. However, as we noted before, many of the existing remote access communications links in use in the electric power industry today are *not* secure against potential unauthorized electronic intrusion.

Fortunately, the performance requirements of remote access communications are much less stringent than those of SCADA and real-time protection communications. We should be able to apply encryption to hide the transmitted passwords and cryptographic authentication to block unauthorized access attempts with very little perceptible performance degradation. Furthermore, COTS devices already exist that can provide all or most of these functions.

Conclusions

Modern cryptography has provided us with many functions and protocols that we can use to secure the communications infrastructure within the electric power industry. When securing these systems, we must choose a cryptographic solution that meets the specific requirements of each communications link. We have shown that these challenges are surmountable. Cryptographic devices already exist that can be used to increase the level of electronic security in some SCADA, remote-access, and real-time protection communication systems. Finally, we can design optimized solutions to effectively secure systems that cannot tolerate the small amounts of latency that these existing devices add.

References

[1] Schweitzer III, E.O., Behrendt, K., and Lee, T. "Digital Communications for Power System Protection: Security, Availability, and Speed," *Proceedings of the 23rd Annual Western Protective Relay Conference*, Spokane, WA, October, 1998. <http://www.selinc.com/techpprs/6083.pdf> (4 Mar. 2003)

[2] Roberts, J., Tziouvaras, D., Benmouyal, G., and Altuve, H. "The Effect of Multiprinciple Line Protection on Dependability and Security," *Proceedings of the 54th Annual Conference for Protective Relay Engineers*, Texas A&M Univ., College Station, TX, April 2001. <http://www.selinc.com/techpprs/6109-Paper-WPRC.pdf> (4 Mar. 2003)

[3] Schneier, B. *Applied Cryptography Second Edition: Protocols, Algorithms, and Source Code in C*, John Wiley & Sons, Inc., 1996.

[4] Risley, A., and Roberts, J. "Electronic Security Risks Associated with the use of Wireless, Point-to-Point Communications in the Electric Power Industry," *DistribuTECH 2003*, February 4-6, 2003, Las Vegas, Nevada.

[5] Nichols, R.K. *ICSA Guide to Cryptography*, McGraw-Hill, 1999.

[6] Trappe, W., and Washington, L.C. *Introduction to Cryptography with Coding Theory*, Prentice Hall, Inc., Upper Saddle River, NJ, 2002.

[7] Oman, P., Risley, A., Roberts, J., and Schweitzer III, E.O. "Attack and Defend Tools for Remotely Accessible Control and Protection Equipment in Electric Power Systems," *55th Annual Conference for Protective Relay Engineers*, Texas A&M University, April 9–11, 2002, College Station, TX. <http://www.selinc.com/techpprs/6132.pdf> (4 Mar. 2003)

Biographies

Peter LaDow is an Associate Software Engineer at Schweitzer Engineering Laboratories, Inc. in Pullman, WA. Prior to joining SEL he worked for Advanced Hardware Architectures as a Senior Design Engineer. He has patents pending for communications systems and information theory. He received his BSEE from Washington State University in 1997. He is a general member of the IEEE.

Allen D. Risley is a Security Analyst at Schweitzer Engineering Laboratories, Inc. in Pullman, WA. Prior to joining SEL, he worked at Advanced Hardware Architectures as a Senior Research Engineer specializing in information theory and forward error correction. He received his Master of Science degree in Electrical Engineering from Washington State University in 1998. He has presented papers at the 1998 Conference on Information Sciences and Systems, as well as the 2001 ISCTA conference. His work has been published in the *Proceedings of the International Symposium on Information Theory* and the *IEEE Transactions on Communications*.

Jeff Roberts is a Research Fellow at Schweitzer Engineering Laboratories, Inc. in Pullman, WA. Prior to joining SEL, he worked for Pacific Gas and Electric as a Relay Protection Engineer. He received his BSEE from Washington State University in 1985. Mr. Roberts holds 19 patents and has several other patent applications pending; he has written many papers in the areas of distance element design, sensitivity of distance and directional elements, directional element design, and analysis of event report data. He has delivered papers at the Western Protective Relay Conference, Texas A&M University, Georgia Tech, Monterrey Symposium on Electric Systems Protection, and the South African Conference on Power System Protection. He is a Senior Member of the IEEE and was recognized by the Spokane chapter of the IEEE as Engineer of the Year for 2001.