Chapter 7: The Role of Cryptography in Information Security

Tom Olzak, MBA, CISSP
May 2012
InfoSec Institute

After its human resources, information is an organization's most important asset. As we have seen in previous chapters, security and risk management is data centric. All efforts to protect systems and networks attempt to achieve three outcomes: data availability, integrity, and confidentiality. And as we have also seen, no infrastructure security controls are 100% effective. In a layered security model, it is often necessary to implement one final prevention control wrapped around sensitive information: encryption.

Encryption is not a security panacea. It will not solve all your data-centric security issues. Rather, it is simply one control among many. In this chapter, we look at encryption's history, its challenges, and its role in security architecture.

Cryptography

Cryptography is a science that applies complex mathematics and logic to design strong encryption methods. Achieving strong encryption, the hiding of data's meaning, also requires intuitive leaps that allow creative application of known or new methods. So cryptography is also an art.

Early Cryptography

The driving force behind hiding the meaning of information was war. Sun Tzu wrote,

> Of all those in the army close to the commander none is more intimate than the secret agent; of all rewards none more liberal than those given to secret agents; of all matters none is more confidential than those relating to secret operations.

Secret agents, field commanders, and other human elements of war required information. Keeping the information they shared from the enemy helped ensure advantages of maneuver, timing, and surprise. The only sure way to keep information secret was to hide its meaning.

Early cryptographers used three methods to encrypt information: substitution, transposition, and codes.

Monoalphabetic Substitution Ciphers

One of the earliest encryption methods is the shift cipher. A cipher is a method, or algorithm, that converts plaintext to ciphertext. Caesar's shift cipher is known as a monoalphabetic substitution shift cipher. See Figure 7-1.

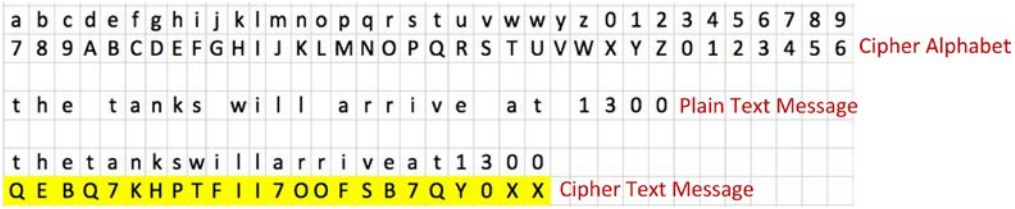

Figure 7- 1: Monoalphabetic Substitution Shift Cipher

The name of this cipher is intimidating, but it is simple to understand. Monoalphabetic means it uses one cipher alphabet. Each character in the cipher alphabet—traditionally depicted in uppercase—is *substituted* for one character in the plaintext message. Plaintext is traditionally written in lowercase. It is a shift cipher because we *shift* the start of the cipher alphabet some number of letters (four in our example) into the plaintext alphabet. This type of cipher is simple to use and simple to break.

In Figure 7-1, we begin by writing our plaintext message without spaces. Including spaces is allowed, but helps with cryptanalysis (cipherbreaking) as shown later. We then substitute each character in the plaintext with its corresponding character in the ciphertext. Our ciphertext is highlighted at the bottom.

Breaking monoalphabetic substitution ciphers

Looking at the ciphertext, one of the problems with monoalphabetic ciphers is apparent: patterns. Note the repetition of "O" and "X." Each letter in a language has specific behavior, or socialization, characteristics. One of them is whether it is used as a double consonant or vowel. According to Mayzner and Tresselt (1965), the following is a list of the common doubled letters in English.

"LL EE SS OO TT FF RR NN PP CC"

In addition to doubling, certain letter pairs commonly appear in English text:

"TH HE AN RE ER IN ON AT ND ST ES EN OF TE ED OR TI HI AS TO"

Finally, each letter appears in moderate to long text with relative frequency. According to Zim (1962), the following letters appear with diminishing frequency. For example, "e" is the most common letter in English text, followed by "t," etc.

"ETAON RISHD LFCMU GYPWB VKXJQ Z"

Use of letter frequencies to break monoalphabetic ciphers was first documented by Abu Yusuf Ya'qub ibnis-haq ibn as-Sabbath ibn 'om-ran ibn Ismail al-Kindi in the ninth century CE (Singh, 1999). al-Kindi did what cryptanalysts (people to try to

break the work of cryptographers) had been trying to do for centuries: develop an easy way to break monoalphabetic substitution ciphers. Once the secret spread, simple substitution ciphers were no longer safe. The steps are

1. If you know the language of the plaintext hidden by the ciphertext, obtain a page-length sample of any text written in that language.
2. Count the occurrence of all letters in the sample text and record the results in a table.
3. Count the occurrence of all cipher alphabet characters in the ciphertext.
4. Start with the most frequently occurring letter in the plaintext and substitute it for the most common character in the ciphertext. Do this for the second most common character, the third, etc.

Eventually, this *frequency analysis* begins to reveal patterns and possible words. Remember that the letters occur with relative frequency. So this is not perfect. Letter frequency, for example, differs between writers and subjects. Consequently, using a general letter frequency chart provides various results depending on writing style and content. However, by combining letter socialization characteristics with frequency analysis, we can work through inconsistency hurdles and arrive at the hidden plaintext.

Summarizing, monoalphabetic substitution ciphers are susceptible to frequency and pattern analysis. This is one of the key takeaways from this chapter; a bad cipher tries to hide plaintext by creating ciphertext containing recognizable patterns or regularly repeating character combinations.

Polyalphabetic Substitution Ciphers

Once al-Kindi broke monoalphabetic ciphers, cryptographers went to work trying to find a stronger cipher. Finally, in the 16th century, a French diplomat developed a cipher that would stand for many decades (Singh, 1999). Combining the work and ideas of Johannes Trithemius, Giovanni Porta, and Leon Battista Alberti, Blaise de Vigenère created the Vigenère cipher.

Vigenère's cipher is based on a Vigenère table, as shown in Figure 7-2. The table consists of 27 rows. The first row of lower case letters represents the plaintext characters. Each subsequent row represents a cipher alphabet. For each alphabet, the first character is shifted one position farther than the previous row. In the first column, each row is labeled with a letter of the alphabet. In some tables, the letters are replaced with numbers representing the corresponding letter's position in the standard alphabet. For example, "A" is replaced with "1," "C" with "3," etc.

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

Figure 7- 2: Vigenère Table

A key is required to begin the cipher process. For our example, the key is FRINGE. The message we wish to encrypt is "get each soldier a meal."

1. Write the message with all spaces removed.
2. Write the key above the message so that each letter of the key corresponds to one letter in the message, as shown below. Repeat the key as many times as necessary to cover the entire message

Key	F	R	I	N	G	E	F	R	I	N	G	E	F	R	I	N	G	E	F
Plain Text	g	e	t	e	a	c	h	s	o	l	d	i	e	r	a	m	e	a	l

3. Identify the rows in the table corresponding to the letters in the key, as shown in Figure 7-3. Since our key is FRINGE, we select the rows designated in the first column as F, R, I, N, G, and E. Each of these rows represents a cipher alphabet we use to encrypt our message.
4. Replace each letter in the message with its corresponding ciphertext character. For example, the first letter in our message is "g." It corresponds to the letter "F" in the key. To find its cipher character, we find the F row in the table and follow it to the column headed by "g" in the first row. The cipher letter at the intersection is "M." Following these steps to locate the appropriate cipher characters, our final encrypted message is:

MWCSHHNKXZKNKJJALFR

Key	F	R	I	N	G	E	F	R	I	N	G	E	F	R	I	N	G	E	F				
Plain Text	g	e	t	e	a	c	h	s	o	l	d	i	e	r	a	m	e	a	l				

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

Figure 7- 3: Selection of Table Rows Based on Key

Our encrypted message used six cipher alphabets based on our key. Anyone with the key and the layout of the table can decrypt the message. However, messages encrypted using the Vigenère cipher are not vulnerable to frequency analysis. Our message, for example, contains four e's as shown in red below. A different cipher character represents each instance of an "e." It is not possible to determine the relative frequency of any single letter. However, it is still vulnerable to attack.

MWCSHHNKXZKNKJJALFR

Breaking the Vigenère cipher

Although slow to gain acceptance, the Vigenère cipher was a very strong and seemingly unbreakable encryption method until the 19th century. Charles Babbage and Friedrich Wilhelm Kasiski demonstrated in the mid and late 1800s respectively that even polyalphabetic ciphers provide trails for cryptanalysts. Although frequency analysis did not work, encrypted messages contained patterns that matched plaintext language behaviors. Once again, a strong cipher fell because it could not distance itself from the characteristics of the plaintext language.

Transposition Ciphers

Other attempts to hide the meaning of messages included rearranging letters to obfuscate the plaintext: transposition. The rail fence transposition is a simple example of this technique. See Figure 7-4. The plaintext, "giveeachsoldierameal," is written with every other letter on a second line. To create the ciphertext, the letters on the first line are written first and then the letters on the second. The resulting cipher text is GVECSLIRMAIEAHODEAEL.

g		v		e		c		s		l		i		r		m		a	
	i		e		a		h		o		d		e		a		e		l

Figure 7- 4: Rail Fence Transposition

The ciphertext retains much of the characteristic spelling and letter socialization of the plaintext and its corresponding language. Using more rows helped, but complexity increased beyond that which was reasonable and appropriate.

Codebooks

In addition to transposition ciphers, codes were also common prior to use of contemporary cryptography. A code replaces a word or phrase with a character. Figure 7-5 is a sample code. Using codes like our example was a good way to obfuscate meaning if the messages are small and the codebooks were safe. However, using a codebook to allow safe communication of long or complex messages between multiple locations was difficult.

^	%	@	∞	⌫	≠
and	morning	night	tanks	bomb run	tomorrow
tanks and bomb run tomorrow night (Plaintext)					
∞^⌫≠@ (Coded message)					

Figure 7- 5: Code Table

The first challenge was creating the codes for appropriate words and phrases. Codebooks had to be large, and the effort to create them was significant: like writing an English/French dictionary. After distribution, there was the chance of codebook capture, loss, or theft. Once compromised, the codebook was no longer useful, and a new one had to be created. Finally, coding and decoding lengthy messages took time, time not available in many situations in which they were used.

Codes were also broken because of characteristics inherent in the plaintext language. For example, "and," "the," "I," "a," and other frequently occurring words or letters could eventually be identified. This provided the cryptanalysts with a finger hold from which to begin breaking a code.

Nomenclators

To minimize the effort involved in creating and toting codebooks, cryptographers in the 16th century often relied on nomenclators. A nomenclator combines a substitution cipher with a small code set, as in the famous one shown in Figure 7-6. Mary Queen of Scots and her cohorts used this nomenclator during a plot against Queen Elizabeth I (Singh, 1999). Thomas Phelippes (cipher secretary to Sir Francis Walsingham, principal secretary to Elizabeth I) used frequency analysis to break it. Phelippes' success cost Queen Mary her royal head.

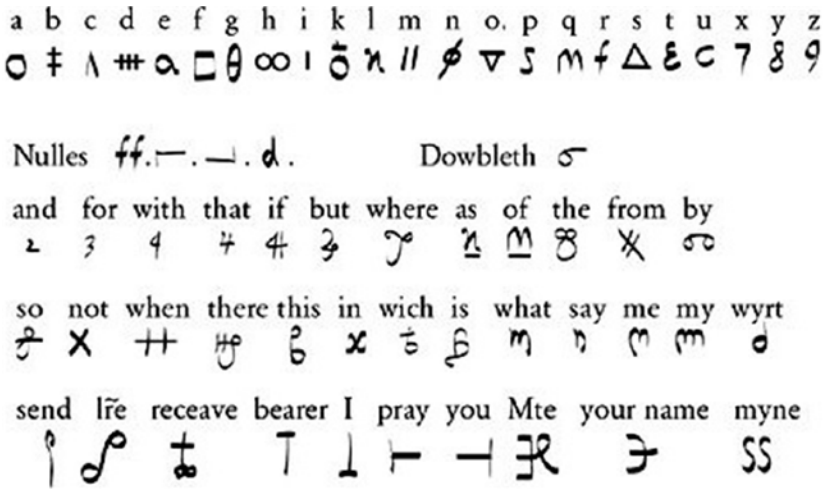

Figure 7- 6: Nomenclator of Mary Queen of Scots (Singh, 1999, loc. 828)

Contemporary Cryptography

Between the breaking of the Vigenère cipher and the 1970s, many nations and their militaries attempted to find the unbreakable cipher. Even Enigma fell to the technology-supported insights of Marian Rejewski and Alan Turing. (If you are interested in a good history of cryptography, including transposition ciphers and codes, see "The Code Book" by Simon Singh.)

Based on what we learn from the history of cryptography, a good cipher

> …makes it impossible to find the plaintext *m* from ciphertext *c* without knowing the key. Actually, a good encryption function should provide even more privacy than that. An attacker shouldn't be able to learn any information about *m*, except possibly its length at the time it was sent (Ferguson, Schneier, & Kohno, 2010, p. 24).

Achieving this ideal requires that any change to the plaintext, no matter how small, must produce a drastic change in the ciphertext, such that no relationship between the plaintext and the resulting ciphertext is evident. The change must start at the beginning of the encryption process and diffuse throughout all intermediate permutations until reaching the final ciphertext. Attempting to do this before the

late 20th century, and maintain some level of business productivity, was not reasonable. Powerful electronic computers were stuff of science fiction. Today, we live in a different world.

Block Cipher Modes

The standard cipher in use today is the Advanced Encryption Standard (AES). It is a *block cipher mode* that ostensibly meets our definition of an ideal cipher. However, it has already been broken… on paper. AES is a symmetric cipher, meaning that it uses a single key for encryption and decryption. Cryptanalysts have theoretically broken it, but we need better computers to test the discovered weaknesses. It will be some time before private industries have to worry about changing their encryption processes.

A block cipher mode "…features the use of a symmetric key block cipher algorithm…" (NIST, 2010). Figure 7-7 depicts a simple block cipher. The plaintext is broken into blocks. In today's ciphers, the block size is typically 128 bits. Using a key, each block passes through the block algorithm resulting in the final ciphertext. One of the problems with this approach is lack of diffusion. The same plaintext with the same key produces the same ciphertext. Further, a change in the plaintext results in a corresponding and identifiable change in the ciphertext.

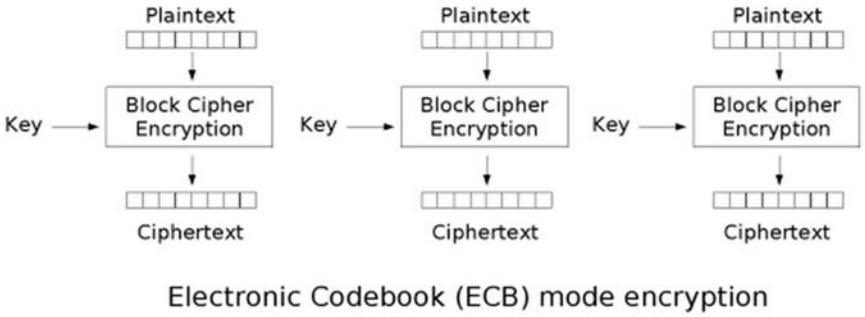

Figure 7- 7: Simple Block Cipher ("Electronic codebook," 2012)

Because of the weaknesses in simple block algorithms, cryptographers add steps to strong ciphers. Cipher block chaining (CBC), for example, adds diffusion by using ciphertext, an initialization vector, and a key. Figure 7-8 graphically depicts the encipher process (⊕ = XOR). The initialization vector (IV) is a randomly generated and continuously changing set of bits the same size as the plaintext block. The resulting ciphertext changes as the IV changes. Since the key/IV pair should never be duplicated, the same plaintext can theoretically pass through the cipher algorithm using the same key and never produce the same ciphertext.

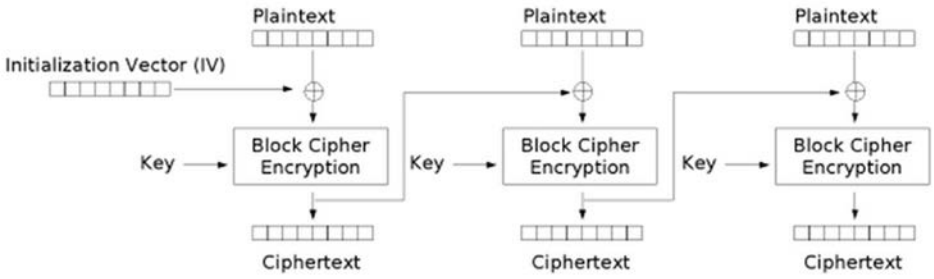

Figure 7- 8: Cipher-block Chaining Cipher Mode ("Cipher-block chaining," 2012)

When the CBC cipher begins, it XORs the plaintext block with the IV and submits it to the block algorithm. The algorithm produces a block of ciphertext. The ciphertext from the first block is XORed with the next block of plaintext and submitted to the block algorithm using the same key. If the final block of plaintext is smaller than the cipher block size, the plaintext block is padded with an appropriate number of bits. This is stronger, but it still fell prey to skilled cryptanalysts.

AES, another block cipher mode, uses a more sophisticated approach, including byte substitution, shifts, column mixing, and use of cipher-generated keys for internal processing (NIST, 2001). It is highly resistant to any attack other than key discovery attempts. However, cryptanalysts have theoretically broken AES (Ferguson, Schneier, & Kohno, 2010). This does not mean it is broken in practice; it is still the recommended encryption method for strong data protection.

For additional information on attacks against modern ciphers, see "Cryptography Engineering: Design Principles and Practical Applications" by Niels Ferguson, Bruce Schneier, and Tadayoshi Kohno.

Key Management

The processes underlying all widely accepted ciphers are and should be known, allowing extensive testing by all interested parties: not just the originating cryptographer. We tend to test our expectations of how our software development creations should work instead of looking for ways they deviate from expected behavior. Our peers do not usually approach our work in that way. Consequently, allowing a large number of people to try to break an encryption algorithm is always a good idea. Secret, proprietary ciphers are suspect. A good encryption solution follows Auguste Kerckhoffs' principle:

> The security of the encryption scheme must depend only on the secrecy of the key… and not on the secrecy of the algorithm (Ferguson, Schneier, & Kohno, 2010, p. 24)

If a vendor, or one of your peers, informs you he or she has come up with a proprietary, secret cipher that is unbreakable, that person is either the foremost

cryptographer of all time or deluded. In either case, only the relentless pounding on the cipher by cryptanalysts can determine its actual strength.

Now that we have established the key as the secret component of any well-tested cipher, how do we keep our keys safe from loss or theft? If we lose a key, the data it protects is effectively lost to us. If a key is stolen, the encrypted data is at higher risk of discovery. And how do we share information with other organizations or individuals if they do not have our key?

AES is a symmetric cipher; it uses the same key for both encryption and decryption. So, if I want to send AES-encrypted information to a business partner, how do I safely send the key to the receiver?

Principles of Key Management

Managing keys requires three considerations:

1. Where will you store them?
2. How will you ensure they are protected but available when needed?
3. What key strength is adequate for the data protected?

Key Storage

Many organizations store key files on the same system, and often the same drive, as the encrypted database or files. While this might seem like a good idea if your key is encrypted, it is bad security. What happens if the system fails and the key is not recoverable? Having usable backups helps, but backup restores do not always work as planned…

Regardless of where you keep your key, encrypt it. Of course, now you have to decide where to store the encryption key for the encrypted encryption key. None of this confusion is necessary if you store all keys in a secure, central location. Further, do not rely solely on backups. Consider storing keys in escrow, allowing access by a limited number of employees ("key escrow," n.d.). Escrow storage can be a safe deposit box, a trusted third party, etc. Under no circumstances allow any one employee to privately encrypt your keys.

Key Protection

Encrypted keys protecting encrypted production data cannot be locked away and only brought out by trusted employees as needed. Rather, keep the keys available but safe. Key access security is, at its most basic level, a function of the strength of your authentication methods. Regardless of how well protected your keys are when not used, authenticated users (including applications) must gain access. Ensure identity verification is strong and aggressively enforce separation of duties, least privilege, and need-to-know.

Key Strength

Most, if not all, attacks against your encryption will try to acquire one or more of your keys. Use of weak keys or untested/questionable ciphers might achieve compliance, but it provides your organization, its customers, and its investors with a false sense of security. As Ferguson, Schneier, and Kohno (2010) wrote,

> In situations like this (which are all too common) any voodoo that the customer [or management] believes in would provide the same feeling of security and work just as well (p. 12).

So what is considered a strong key for a cipher like AES? AES can use 128-, 192-, or 256-bit keys. 128-bit keys are strong enough for most business data, if you make them as random as possible. Key strength is measured by key size and an attacker's ability to step through possible combinations until the right key is found. However you choose your keys, ensure you get as close as possible to a key selection process in which all bit combinations are equally likely to appear in the key space (all possible keys).

Key Sharing and Digital Signatures

It is obvious from the sections on keys and algorithms that secrecy of the key is critical to the success of any encryption solution. However, it is often necessary to share encrypted information with outside organizations or individuals. For them to decrypt the ciphertext, they need our key.

Transferring a symmetric cipher key is problematic. We have to make sure all recipients have the key and properly secure it. Further, if the key is compromised in some way, it must be quickly retired from use by anyone who has it. Finally, distribution of the key must be secure. Luckily, some very smart cryptographers came up with the answer.

Asymmetric Cryptography

In 1978, Ron Rivest, Adi Shamir, and Leonard Adelman (RSA) publicly described a method of using two keys to protect and share data; one key is public and the other private. The organization or person to whom the public key belongs distributes it freely. However, the private key is kept safe and is never shared. This enables a process known as asymmetric encryption and decryption.

As shown in Figure 7-9, the sender uses the recipient's public key to convert plaintext to ciphertext. The ciphertext is sent and the recipient uses her private key to recover the plaintext. Only the person with the private key corresponding to the public key can decrypt the message, document, etc. This works because the two keys, although separate, are mathematically entwined.

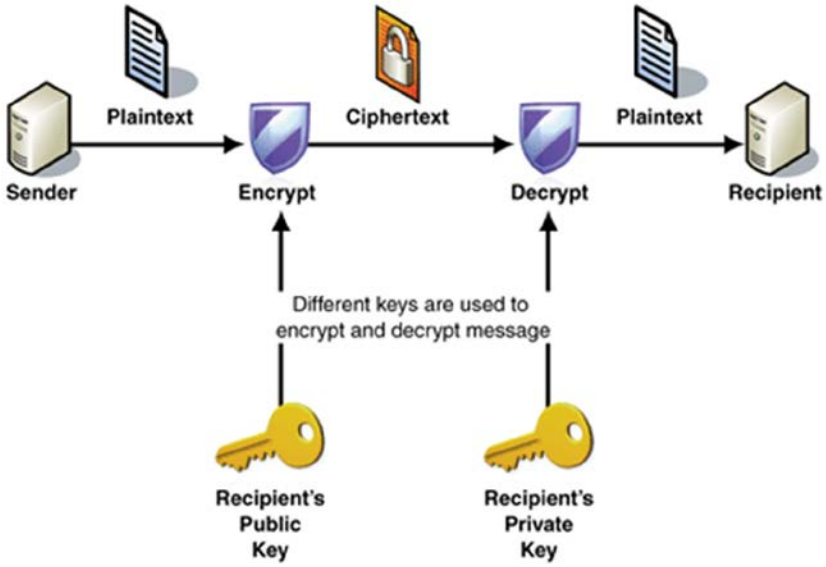

Figure 7- 9: Asymmetric Cryptography (Microsoft, 2005)

At a very high level, the RSA model uses prime numbers to create a public/private key set:

1. Creation begins by selecting two extremely large prime numbers. They should be chosen at random and of similar length.
2. The two prime numbers are multiplied together.
3. The product becomes the public key.
4. The two factors become the private key.

There is more to asymmetric key creation, but this is close enough for our purposes.

When someone uses the public key, or the product of the two primes, to encrypt a message, the recipient of the ciphertext must know the two prime numbers that created it. If the primes were small, a brute force attack can find them. However, use of extremely large primes and today's computing power makes finding the private key through brute force unlikely. Consequently, we can use asymmetric keys to share symmetric keys, encrypt email, and various other processes where key sharing is necessary.

The Diffie-Hellman key exchange method is similar to the RSA model and it was made public first. However, it allows two parties who know nothing about each other to establish a shared key. This is the basis of SSL and TLS security. An encrypted session key exchange occurs over an open connection. Once both parties to the session have the session key (also know as a shared secret), they establish a virtual and secure tunnel using symmetric encryption.

So why not throw out symmetric encryption and use only asymmetric ciphers? First, symmetric ciphers are typically much stronger. Further, asymmetric encryption is far slower. So we have settled for symmetric ciphers for data center

and other mass storage encryption and asymmetric ciphers for just about everything else. And it works… for now.

Digital Signatures

Although not really encryption as we apply the term in this chapter, the use of asymmetric keys has another use: digital signatures. If Bob, for example, wants to enable verification that he actually sent a message, he can sign it.

Refer to Figure 7-10. The signature process uses Bob's private key, since he is the only person who has it. The private key is used as the message text is processed through a hash function. A hash is a fixed length value that represents the message content. If the content changes, the hash value changes. Further, an attacker cannot use the hash value to arrive at the plain text.

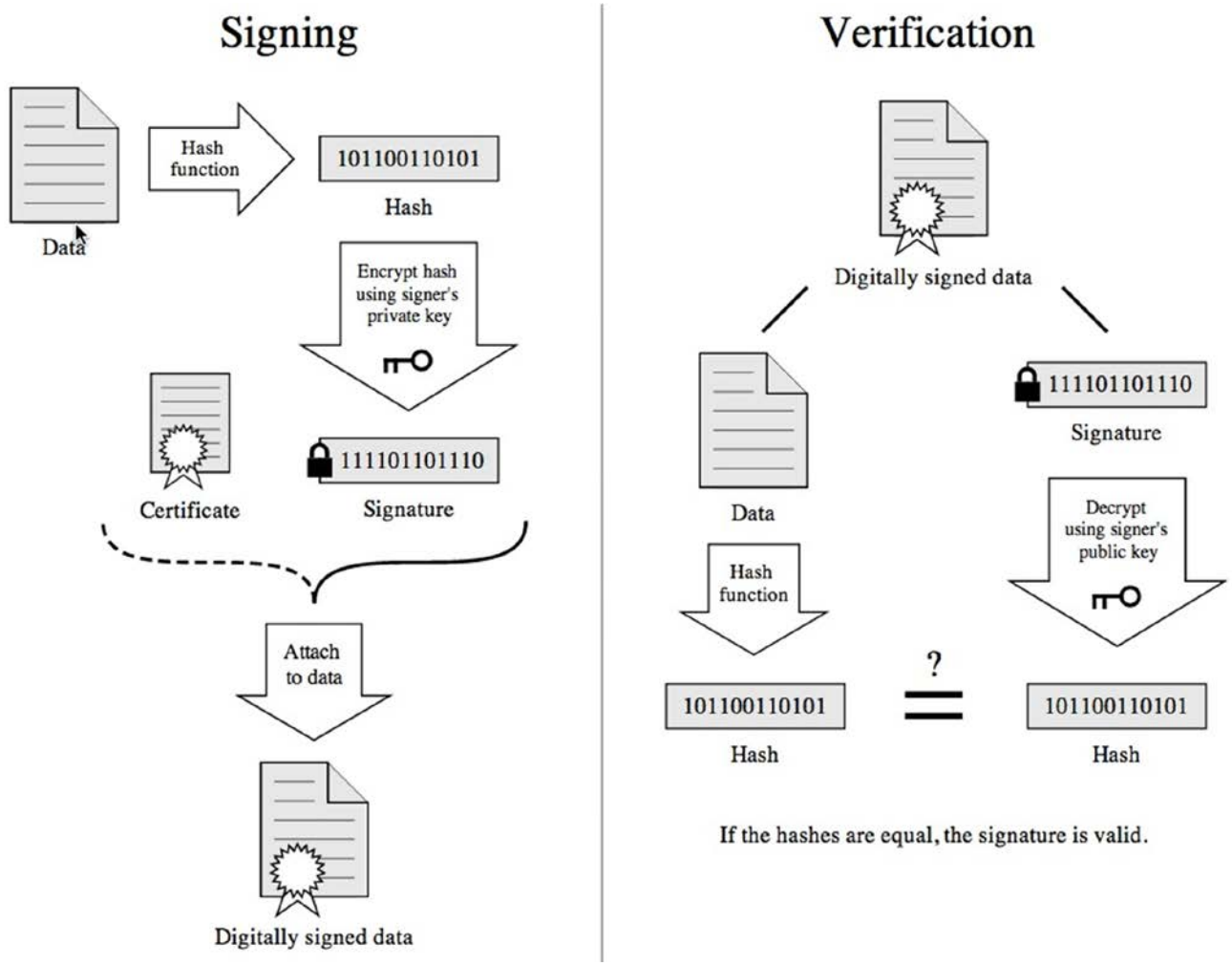

Figure 7- 10: Digital Signing ("Digital signature," 2012)

When Alice receives Bob's message, she can verify the message came from Bob and is unchanged: if she has Bob's public key. With Bob's public key, she rehashes the

message text. If the two hash values are the same, the signature is valid, and the data reached Alice unchanged.

If hash values do not match, either the message text changed or the key used to create the signature hash value is not Bob's. In some cases, the public key might not be Bob's. If attacker, Eve, is able to convince Alice that a forged certificate she sends to her is Bob's key, Eve can send signed messages using a forged "Bob" key that Alice will verify. It is important for a recipient to be sure the public key used in this process is valid.

Public Key Infrastructure (PKI)

Verifying the authenticity of keys is critical to asymmetric cryptography. We have to be sure that the person who says he is Bob is actually Bob or that the bank Web server we access is actually managed by our bank. There are two ways this can happen: through hierarchical trust or a web of trust.

Hierarchical trust

Private industry usually relies on the hierarchical chain-of-trust model that minimally uses three components:

1. Certificate authority (CA)
2. Registration authority (RA)
3. Central directory/distribution management mechanism

The CA issues certificates binding a public key to a specific distinguished name provided by the certificate applicant (subject). Before issuing a certificate, however, it validates the subject's identity. One verification method is domain validation. The CA sends an email containing a token or link to the administrator responsible for the subject's domain. The recipient address might take the form of postmaster@*domainname* or root@*domainname*. The recipient (hopefully the subject or the subject's authorized representative) then follows verification instructions.

Another method, and usually one with a much higher cost for the requestor, is extended validation (EV). Instead of simple administrator email exchange, a CA issuing an EV steps through a rigorous identity verification process. The resulting certificates are structurally the same as other certificates; they simply carry the weight of a higher probability that the certificate holder is who they say they are, by

- Establishing the legal identity and physical existence/presence of the website owner
- Confirming that the requestor is the domain name owner or has exclusive control over it
- Using appropriate documents, confirming the identity and authority of the requestor or its representatives

A simple certificate issuance process is depicted in Figure 7-11. It is the same whether you host your own CA server or use a third party. The subject (end-entity) submits an application for a signed certificate. If verification passes, the CA issues a certificate and the public/private key pair. Figure 7-12 depicts the contents of my personal VeriSign certificate. It contains identification of the CA, information about my identity, the type of certificate and how it can be used, and the CA's signature (SHA1 and MD5 formats).

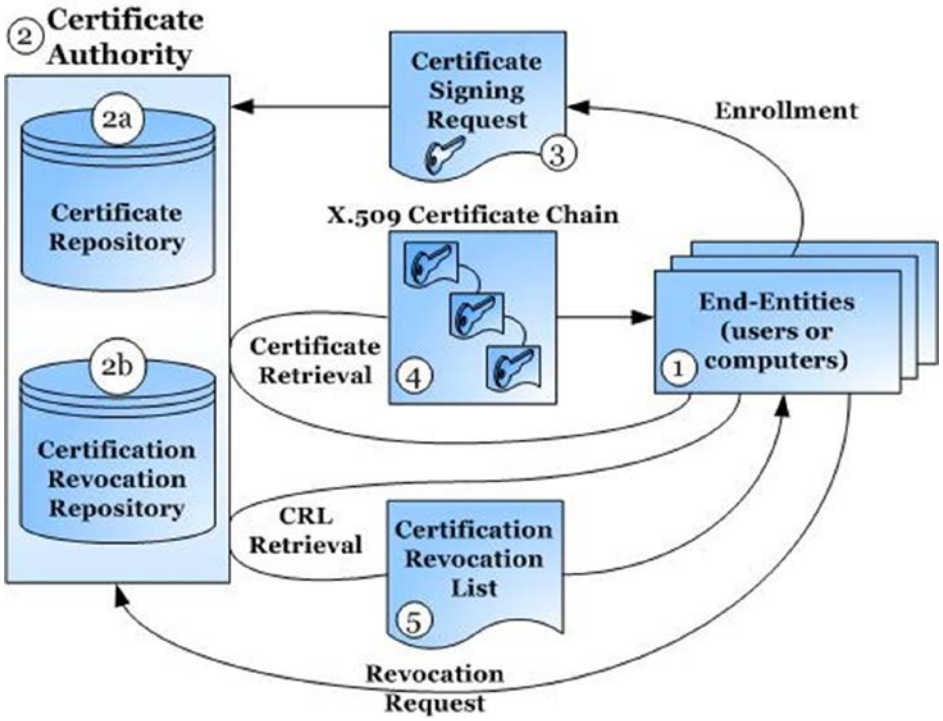

Figure 7- 11: PKI (Ortiz, 2005)

The certificate with the public key can be stored in a publicly accessible directory. If a directory is not used, some other method is necessary to distribute public keys. For example, I can email or snail-mail my certificate to everyone who needs it. For enterprise PKI solutions, an internal directory holds all public keys for all participating employees.

Subject Name	
Organization	VeriSign, Inc.
Organizational Unit	VeriSign Trust Network
Organizational Unit	www.verisign.com/repository/RPA Incorp. by Ref.,LIAB.LTD(c)98
Organizational Unit	Persona Not Validated
Organizational Unit	Digital ID Class 1 - Netscape Full Service
Common Name	Thomas Olzak
Email Address	tolzak@me.com
Issuer Name	
Country	US
Organization	VeriSign, Inc.
Organizational Unit	VeriSign Trust Network
Organizational Unit	Terms of use at https://www.verisign.com/rpa (c)09
Organizational Unit	Persona Not Validated
Common Name	VeriSign Class 1 Individual Subscriber CA - G3
Serial Number	27 7F 15 44 8B D4 59 2B 89 E0 A2 6B C5 13 84 BF
Version	3
Signature Algorithm	SHA-1 with RSA Encryption (1.2.840.113549.1.1.5)
Parameters	none
Not Valid Before	Thursday, May 3, 2012 8:00:00 PM Eastern Daylight Time
Not Valid After	Saturday, May 4, 2013 7:59:59 PM Eastern Daylight Time
Public Key Info	
Algorithm	RSA Encryption (1.2.840.113549.1.1.1)
Parameters	none
Public Key	256 bytes : E5 A2 E9 13 02 02 0D F8 3C 3E 16 01 76 AE 7C 2D 5A 29 FC 5D 5B BE 48 EE 96 33 12 5D 64 AE 73 E5 7B BF 66 55 B4 C6 86 57 57 BB 83 87 66 55 18 33 8B 03 3E 2D BC D7 28 AF 96 25 51 7F F0 15 16 F6 40 8F FF 92 B5 4F B5 15 16 19 CA BA F7 4F AA CD 47 91 0D 5A 9D 17 F7 48 EA 43 BB 65 B4 01 22 14 63 2C 64 A3 02 C7 71 E7 CE 44 2F EE 9E 87 F1 1E D8 15 22 8F 3B CE 56 7A ED 69 27 34 C8 0F 22 A1 34 D7 D5 F5 BE D6 93 F5 3C EA E9 3D 3D E6 16 9E 67 EC 46 7A 82 62 73 31 41 49 B2 DE AB 92 DF ED D9 F2 15 6C 98 8F 9F 09 D1 37 CF BC 68 D4 AB 7D 80 F3 74 4E 00 BE 54 F9 B7 75 A1 F0 CF 39 DD 92 C1 84 F0 4A D0 09 75 C6 67 18 6C F3 35 0D 9B B4 07 D5 FE FD 81 88 05 AF F7 1A 8A FB 38 F6 F1 5B C4 75 8D 9F 9E 96 0C 59 AD D9 FD 8F FE C3 ED 65 84 17 59 AE 0F A1 C8 66 1A F1 15 33 F4 99 D5 B3
Exponent	65537
Key Size	2048 bits
Key Usage	Encrypt, Verify, Wrap, Derive
Signature	256 bytes : D2 9C 36 D8 72 BD 6D 4D ...
Extension	Key Usage (2.5.29.15)
Critical	NO
Usage	Digital Signature, Key Encipherment
Extension	Basic Constraints (2.5.29.19)
Critical	NO
Certificate Authority	NO
Extension	Extended Key Usage (2.5.29.37)
Critical	NO
Purpose #1	Email Protection (1.3.6.1.5.5.7.3.4)
Purpose #2	Client Authentication (1.3.6.1.5.5.7.3.2)
Extension	Certificate Policies (2.5.29.32)
Critical	NO
Policy ID #1	(2.16.840.1.113733.1.7.23.1)
Qualifier ID #1	Certification Practice Statement (1.3.6.1.5.5.7.2.1)
CPS URI	https://www.verisign.com/rpa
Extension	CRL Distribution Points (2.5.29.31)
Critical	NO
URI	http://indc1digitalid-g3-crl.verisign.com/IndC1DigitalID-G3.crl
Fingerprints	
SHA1	0A 07 25 50 30 AF 9B AE A7 06 EA F3 4E A1 30 24 F4 5B 70 91
MD5	39 FF 2F 7D 23 E9 06 B3 2B E4 64 3F 8F D2 F6 13

Figure 7- 12: Personal Certificate

The hierarchical model relies on a chain of trust. Figure 7-13 is a simple example. When an application/system first receives a subject's public certificate, it must verify its authenticity. Because the certificate includes the issuer's information, the verification process checks to see if it already has the issuer's public certificate. If not, it must retrieve it. In this example, the CA is a root CA and its public key is included in its root certificate. A root CA is at the top of the certificate signing hierarchy. VeriSign, Comodo, and Entrust are examples of root CAs.

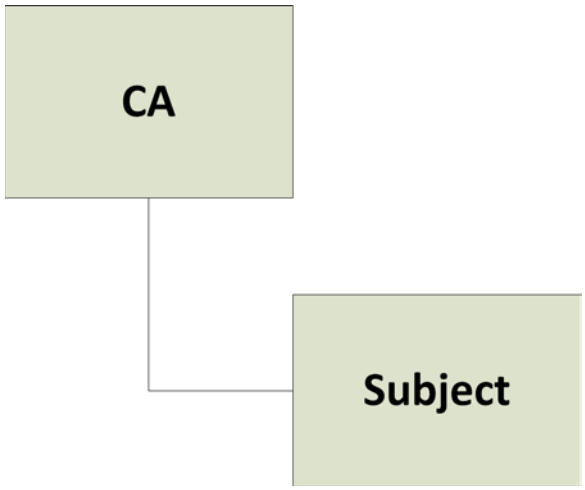

Figure 7- 13: Simple Chain of Trust

Using the root certificate, the application verifies the issuer signature (fingerprint) and ensures the subject certificate is not expired or revoked (see below). If verification is successful, the system/application accepts the subject certificate as valid.

Root CAs can delegate signing authority to other entities. These entities are known as intermediate CAs. Intermediate CAs are trusted only if the signature on their public key certificate is from a root CA or can be traced directly back to a root. See Figure 7-14. In this example, the root CA issued CA^1 a certificate. CA^1 used the certificate's private key to sign certificates it issues, including the certificate issued to CA^2. Likewise, CA^2 used its private key to sign the certificate it issued to the subject. This can create a lengthy chain of trust.

When I receive the subject's certificate and public key for the first time, all I can tell is that it was issued by CA^2. However, I do not implicitly trust CA^2. Consequently, I use CA^2's public key to verify its signature and use the issuing organization information in its certificate to step up the chain. When I step up, I encounter another intermediate CA whose certificate and public key I need to verify. In our example, a root CA issued the CA^1 certificate. Once I use the root certificate to verify the authenticity of the CA^1 certificate, I establish a chain of trust from the root to the subject's certificate. Because I trust the root, I trust the subject.

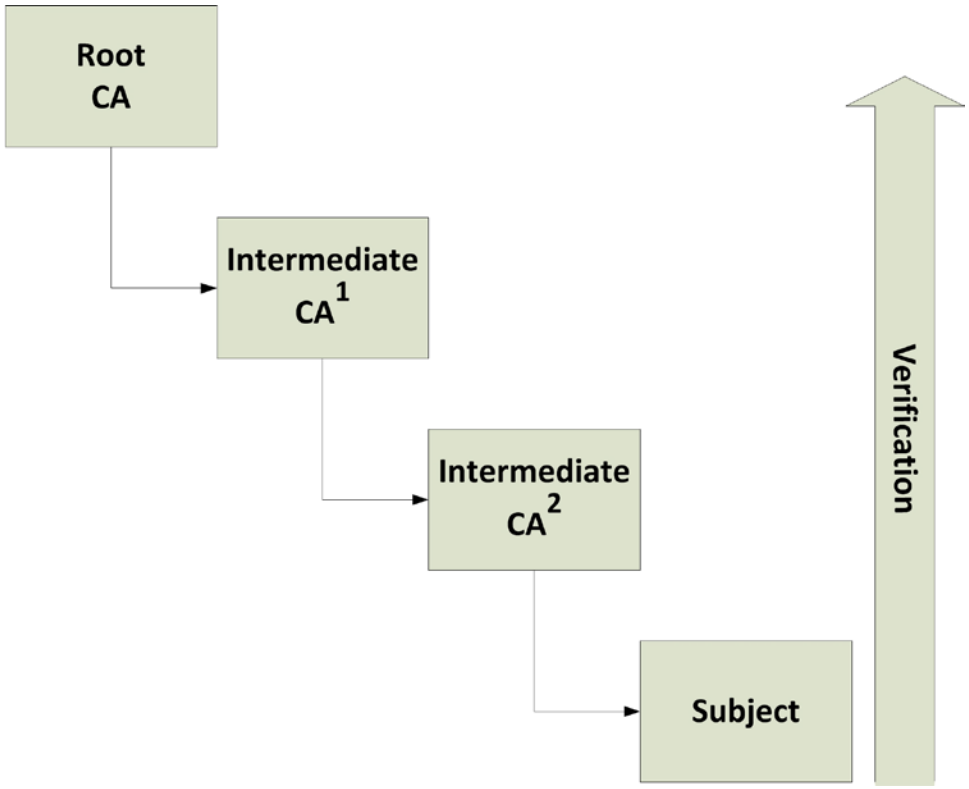

Figure 7- 14: Chain of Trust

This might seem like a lot of unnecessary complexity, and it often is. However, using intermediate CAs allows organizations to issue their own certificates that customers and business associates can trust. Figure 7-15 is an example of how this might work. A publicly known and recognized root CA (e.g., VeriSign) delegates certificate issuing authority to Erudio Products to facilitate Erudio's in-house PKI implementation. Using the intermediate certificate, Erudio issues certificates to individuals, systems, and applications. Anyone receiving a subject certificate from Erudio can verify its authenticity by stepping up the chain of trust to the root. If they trust the root, they will trust the Erudio subject.

Revocation

Certificates are sometimes revoked for cause. When this happens, it is important to notify everyone that the revoked certificates are no longer trusted. This is done using a certificate revocation list (CRL). A CRL contains a list of serial numbers for revoked certificates. Each CA issues its own CRL, usually on an established schedule.

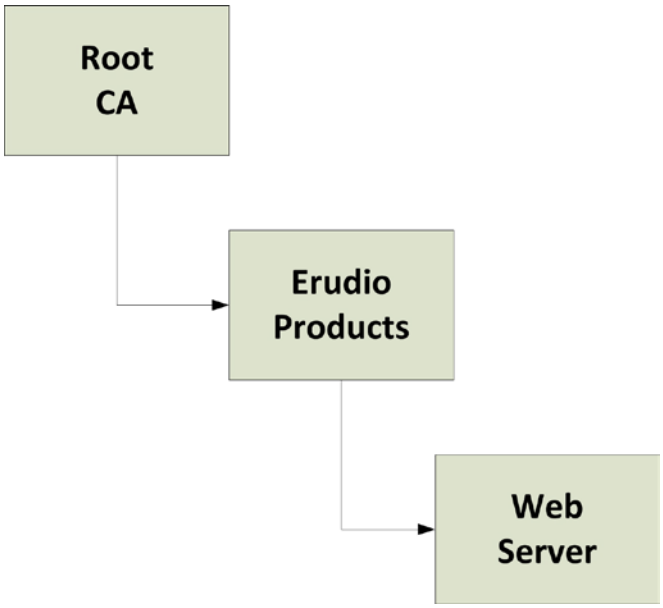

Figure 7- 15: Business as Intermediate CA

Web of Trust (WoT)

Although most medium and large organizations use the hierarchy of trust model, it is important to spend a little time looking at the alternative. The WoT also binds a subject with a public key. Instead of using a chain of trust with a final root, it relies on other subjects to verify a new subject's identity. Phil Zimmerman (1994), creator of Pretty Good Privacy (PGP), puts it this way:

> As time goes on, you will accumulate keys from other people that you may want to designate as trusted introducers. Everyone else will each choose their own trusted introducers. And everyone will gradually accumulate and distribute with their key a collection of certifying signatures from other people, with the expectation that anyone receiving it will trust at least one or two of the signatures. This will cause the emergence of a decentralized fault-tolerant web of confidence for all public keys.

WoT implementations are available for free download and use by anyone (http://www.pgpi.org/download/). But the simplicity of the approach for one or two people can disappear as the size of the subject organization grows. Further, complexity is introduced as WoT-based organizations try to interact with hierarchy of trust participants. Consider all business challenges carefully before implementing a WoT solution in your organization.

Enterprise Key Management

Managing key sharing is easy; managing keys for enterprise encryption implementations is often difficult and ineffective. Good keys are difficult or impossible to remember, must be kept secure, and must be available when production processes require access to encrypted data. Further, all this provides an additional point of attack that, if successful, completely invalidates encryption as a

security control (Ferguson, Schneier, & Kohno, 2010). Distributed management of keys is not the answer, as it introduces even more opportunities for key compromise.

Centralized Key Management Example

Central key management governed by organization-specific encryption policies is likely the best key management option. In addition, only two or three people should have administrative access to key management controls. Vormetric Data Security (Vormetric, 2010), as shown in Figure 7-16, is an example of a product providing these capabilities. In addition to ensuring key security, this type of solution also allows auditing of key creation, use, and retirement. Further, Figure 7-17 depicts the key administrator's ability to grant custodial access to sets of keys based on job role, enforcing separation of duties.

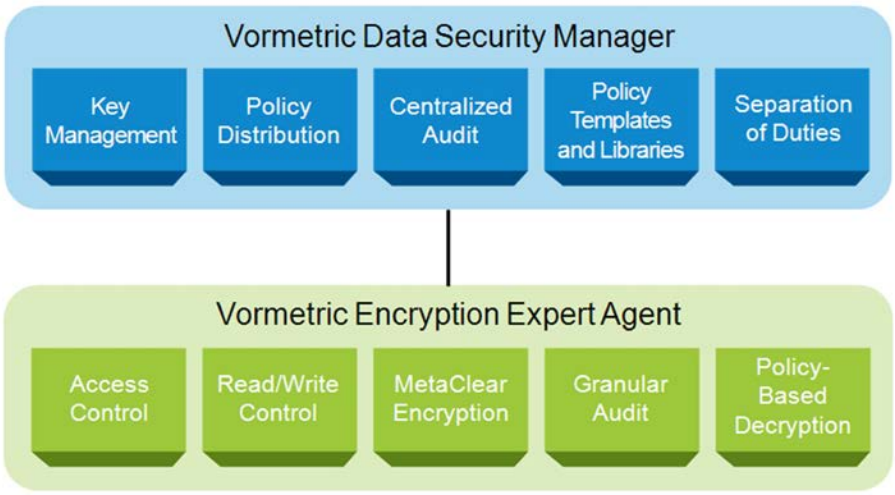

Figure 7- 16: Centralized Key management Services

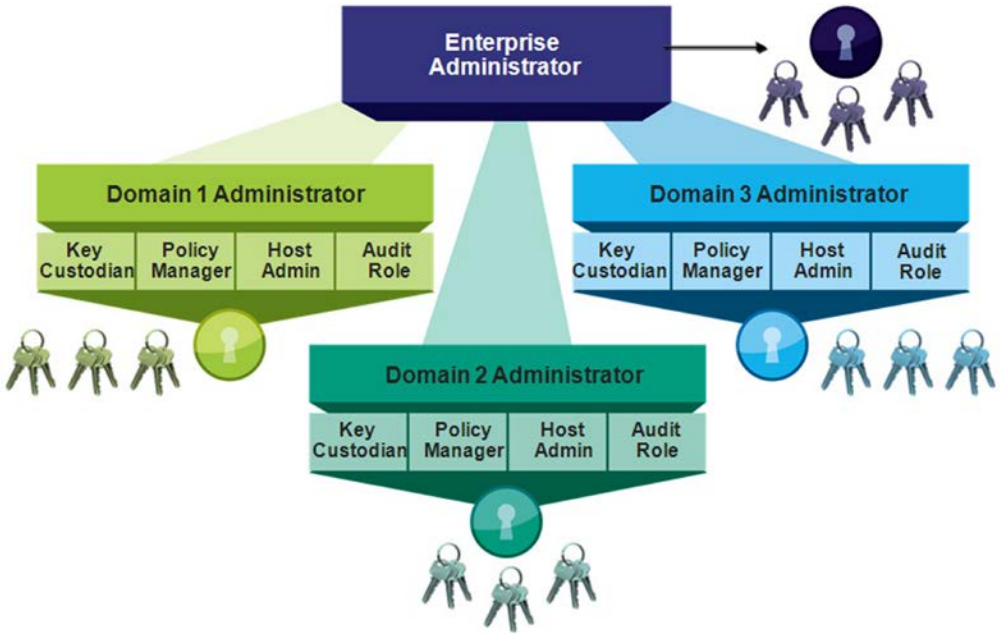

Figure 7- 17: Separation of Key Administration

Centralized encryption helps ensure keys are always available and that data is not encrypted when it is not necessary, appropriate, or wanted. Keys are encrypted and easy to backup or export for escrow storage.

Cryptography's Role in the Enterprise

Encrypting every piece of data in your organization does not guarantee it is protected from unauthorized access. The only thing guaranteed with this approach is unnecessary costs and potentially unhappy production managers. Before looking at when and what to encrypt, it is important to understand where encryption fits in overall security controls architecture.

Just Another Layer

Encryption is just another security control; it adds an additional prevention layer, nothing more. The biggest mistake many organizations make is relying on it as a panacea for all security risk.

For example, data is decrypted on servers and end-user devices when processed. What good is encryption in transit when the system attack surfaces are ignored? An attacker who compromises an online payment server could care less whether or not you use encryption. Everything he needs is in plaintext.

In other cases, a key might be compromised. Assuming encryption provides 100% protection might cause a security team to ignore the importance of detection (e.g., SIEM) or response policies and controls. Again, inspect what you expect. Never assume anything, including encryption, is achieving expected outcomes.

Before or while deploying encryption, implement the following controls or processes (Olzak, 2006):

- Reduce attack surfaces for applications, systems, and the network
- Implement strong application access controls
- Implement log management and monitoring solutions to detect anomalous activity across the network, on systems, and in/around databases
- Separate data access from data management; business users of sensitive data should access it through applications with strong, granular user access controls
- Ensure reasonable and appropriate physical security for network, storage, and system components
- Implement strong authentication and authorization controls between applications and the databases they access
- Follow best practices for securing databases and the servers that house or manage them

When to Encrypt

A few years ago, Rich Mogull (2005) wrote a paper for Gartner that defines three laws for deciding if to encrypt data. They still apply today; I added number four:

1. **Encrypt data that moves**
   Data moving from one trust zone to another, whether within your organization or between you and an external network, is at high risk of interception. Encrypt it.

   Data moving from trusted portions of the network to end-user devices over wireless LANs almost always at high risk. Encrypt it.

2. **Encrypt for separation of duties when access controls are not granular enough**
   For flat file storage, encrypting a spreadsheet file in a department share provides an additional layer of separation. Only employees with the right authorization have access.

   Application access controls protecting databases often do not provide granular control to strictly enforce need-to-know or least privilege. Using database solutions with field- and row-level encryption capabilities can help strengthen weak application-level access controls.

3. **Encrypt when someone tells you to**
   And then there are local, state, and federal regulations. Couple regulatory constraints with auditor insistence, and you often find yourself encrypting because you have to. This type of encryption is often based on generalizations instead of existing security context. For example, just

because you encrypt protected health information does not mean it is secure enough… but it satisfies HIPAA requirements.

4. **Encrypt when it is a reasonable and appropriate method of reducing risk in a given situation**
   This law is actually a summary of the previous three. After performing a risk assessment, if you believe risk is too high because existing controls do not adequately protect sensitive information, encrypt. This applies to risk from attacks or non-compliance.

How to Encrypt

Implementing secure and operationally efficient encryption solutions is not easy, and maintaining them adds to total cost of ownership (TCO). Further, data is often spread between internal and cloud-based storage. Any solution you select must support all current and future data storage and transport characteristics.

One approach is to purchase a system, install it in your data center, and assign in-house staff to manage it. While this might seem like a good idea, the opportunity costs are high. As with most commodity security controls, encryption solutions can be managed by anyone; they do not require the special knowledge of the business possessed by you or other members of the internal security and LAN teams. Your skills are better applied to projects, assessments, and other business critical activities. Consequently, consider outsourcing encryption and key management.

Encryption-as-a-Service (EaaS) vendors provide all the services and protection we discussed, including key management and encryption according to business policy. In addition to encrypting the data center, they can also serve as a third-party that ensures all data housed by your other cloud service providers is managed by encryption policies as if it were in your own data center. Figure 7-18 is an example of an EaaS solution.

The EaaS provider does not house your data, only your keys. Your in-house administrator, via a Web interface, performs configuration of encryption policies and subject access. Software as a service (SaaS) or storage as a service providers have no access to data while at rest. Finally, the "cloud" can also mean your own data center.

Whether in house or outsourced, make sure your centralized encryption solution meets the following requirements:

- Central storage, management, protection, and management of keys
- Enforcement of your data encryption policies across all relevant data, wherever it is in your network or in the cloud
- Granular access to policy and key management functions based on separation of duties and least privilege
- Publicly known, tested, and unbroken ciphers used for all encryption

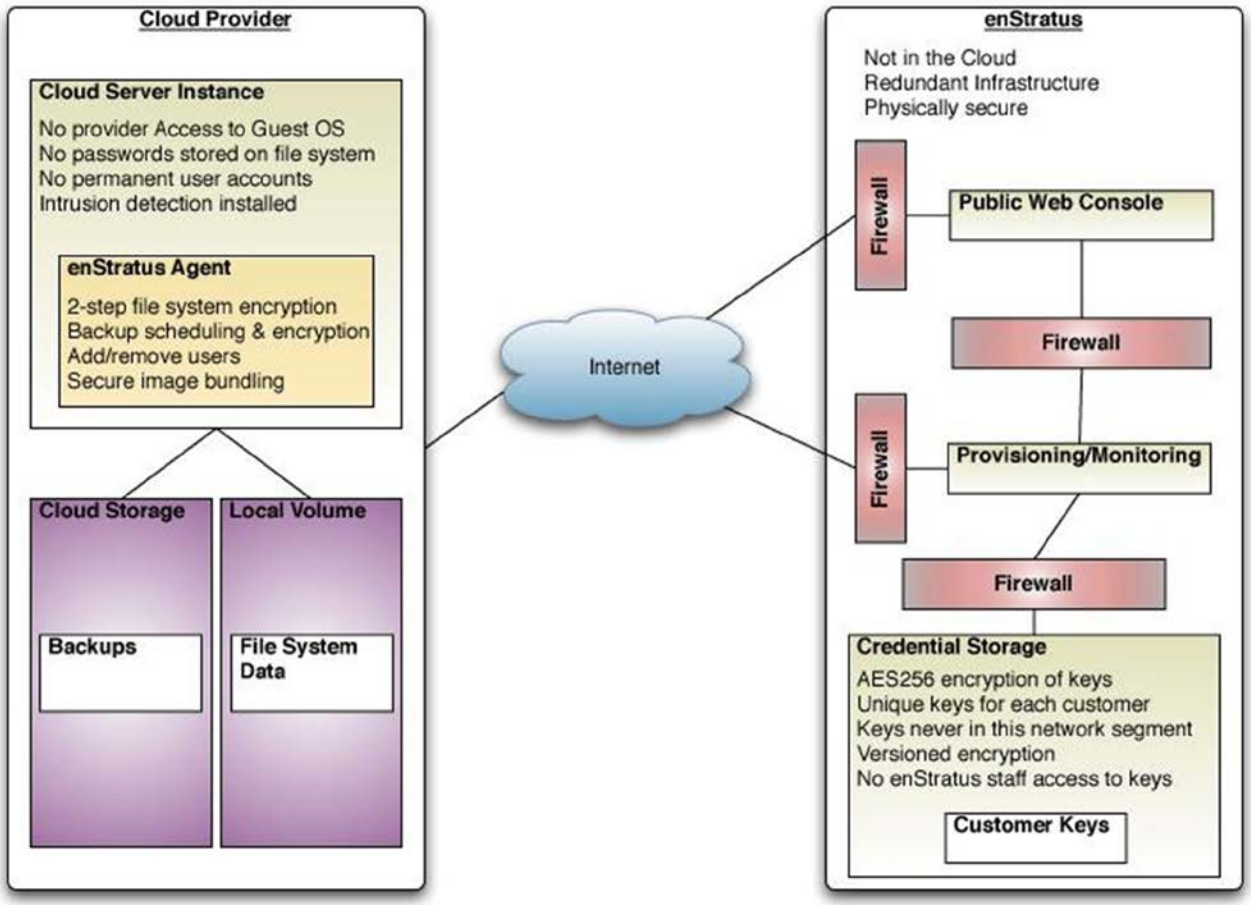

Figure 7- 18: EaaS Configuration (enStratus, 2012)

Tokenization

Although not considered encryption, the payment card industry's acceptance of tokenization as a secure method of managing payment card data makes tokenization an important concept to understand. Primary account numbers (PANs) are not encrypted; they are replaced by a series of alphanumeric characters of the same length.

Also called aliasing, tokenization substitutes an arbitrary value for a PAN. If the PAN is all digits, the token is all digits. In other words, the token takes on the same length and type characteristics of the PAN (RSA, 2009). This allows use of tokens in existing business applications where data length and type matter. After a token is assigned, employees, point-of-sale systems, and other applications use it instead of the actual PAN. This limits the number of points of possible compromise.

Figure 7-19 shows how a financial institution might use tokens. Customer PANs are converted to tokens by a token management system. Token/PAN pairs are stored in a secure database. When various departments access customer information, the token appears instead of the actual PAN.

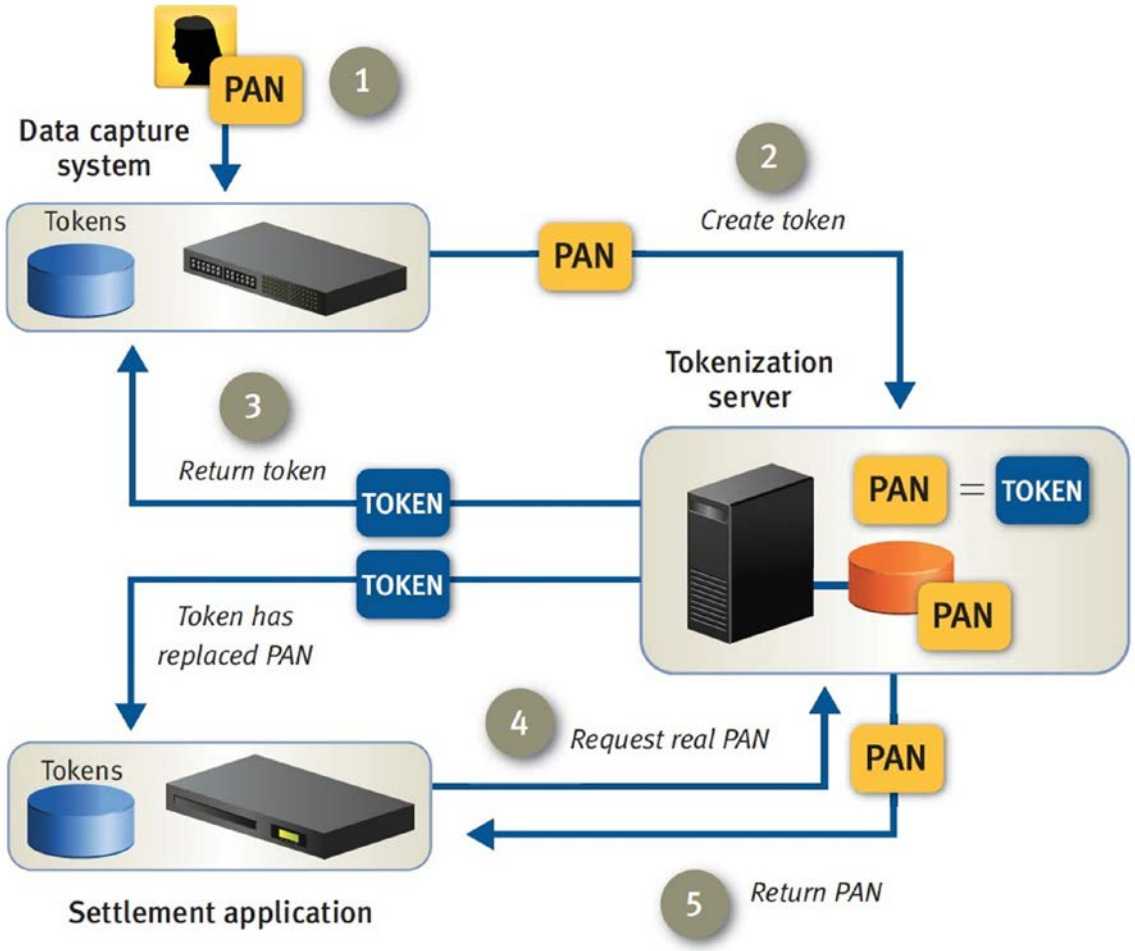

Figure 7- 19: Tokenization Process (RSA, 2009, p. 6)

The process as it appears in Figure 7-19…

1. An employee enters customer data into the data capture system. The data includes the customer's real PAN.
2. The data capture system sends the PAN to the tokenization server where a token is assigned and the PAN/token relationship established.
3. The data capture system receives back a token. All future transactions by employees dealing directly with customers use the token instead of the PAN.
4. If an application, such as the settlement application, needs the actual PAN, it sends a request.
5. If the settlement application is authorized to receive the PAN, the tokenization system honors the request.

Our example reflects a process occurring in financial institutions. However, it also applies to retail stores. If a store's payment processor uses tokens, the retail point-of-sale system can retain payment card information (with the PAN replaced by a token) and retain compliance with the payment card industry data security standard (PCI DSS).

Figure 7-20 provides a closer look at tokenization architecture. Tokens and associated PANs are encrypted. Instead of PANs existing in business transaction files, only the token appears. If an application requires the actual PAN, employee authentication is not enough. The application must be authorized to retrieve it. Further, all access to PANs is logged and anomalies identified. Instead of tracking PAN use at various locations across an organization, monitoring and control of sensitive customer information is centrally controlled and managed.

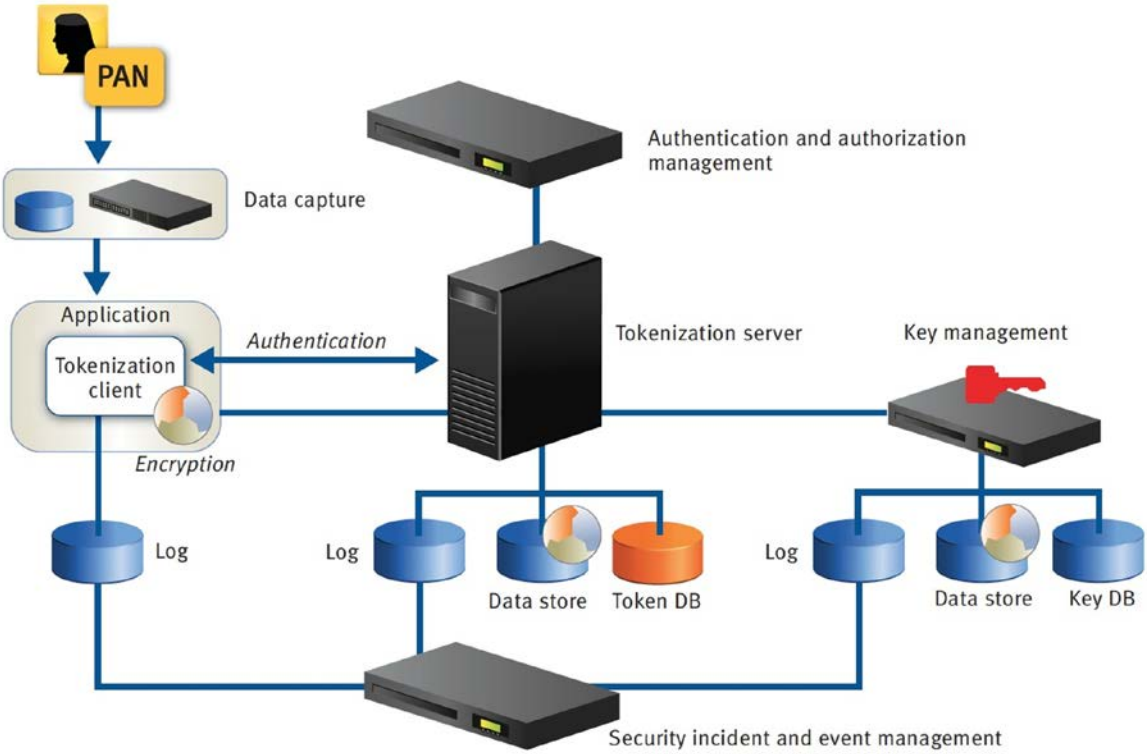

Figure 7- 20: Tokenization Architecture (RSA, 2009, p. 6)

Finally, tokenization provides an easy way to move production data to test environments. If supported, a tokenization server can filter sensitive field data as it moves from production to test. All sensitive fields not already tokenized are filled with tokens for testing changes or new applications, eliminating another potential point of attack.

Conclusion

The history of cryptography is filled with the back-and-forth between cryptographers creating "unbreakable" ciphers and cryptanalysts breaking the unbreakable. However, valuable lessons from the centuries-old battle are used to strengthen today's ciphers. Particularly, any cipher creating ciphertext containing frequency and character/word socialization characteristics of the plaintext language is not secure. The more the ciphertext changes after a change to the plaintext the stronger the cipher.

Key management is an important and often overlooked aspect of enterprise encryption. Ensuring keys are always available, secure, and locked away from everyone except a handful of key administrators is a good start. Further, central key management usually comes with the ability to apply common encryption policies across all data on all managed devices.

The battle cry, "ENCRYPT EVERYTHING!" is not reasonable and appropriate application of encryption as a security control. Decide when and where to encrypt based on risk assessments and management's appetite for risk.

Finally, tokenization is sometimes a better solution than encryption for protecting individual data items. Social security numbers, credit card numbers, and patient insurance information are good examples of possible token targets. If you have to keep these data elements, resist distributing them across desktop and laptop screens when a token will suffice.

References

Cipher-block chaining. (2012). In *wikipedia.org*. Retrieved from http://en.wikipedia.org/wiki/Block_cipher_modes_of_operation#Cipher-block_chaining_.28CBC.29

Digital signature. (2012). In *wikipedia.org*. Retrieved from http://en.wikipedia.org/wiki/Digital_signature

Electronic codebook. (2012). In *wikipedia.org*. Retrieved from http://en.wikipedia.org/wiki/Block_cipher_modes_of_operation#Electronic_codebook_.28ECB.29

enStratus. (2012). *enStratus Security Architecture.* Retrieved May 21, 2012, from enStratus Networks, Inc.: http://enstratus.com/media/document/1/security_architecture.pdf

Ferguson, N., Schneier, B., & Kohno, T. (2010). *Cryptography Engineering: Design Principles and Practical Applications* (Kindle ed.). Indianapolis, IN: Wiley Publishing.

Key escrow. (n.d.). In *Webopedia.* Retrieved from http://www.webopedia.com/TERM/K/key_escrow.html

Microsoft. (2005, December). *Data Confidentiality*. Retrieved May 16, 2012, from MSDN: http://msdn.microsoft.com/en-us/library/ff650720.aspx

Mogull, R. (2005, August). *Management Update: Use the Three Laws of Encryption to Properly Protect Data.* Retrieved February 4, 2006, from Gartner: http://www.gartner.com

NIST. (2001, November 26). *Advanced Encryption Standard.* Retrieved May 15, 2012, from NIST Computer Security Resource Center: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

NIST. (2010, February 16). *Block Cipher Modes*. (N. I. Technology, Producer) Retrieved May 15, 2012, from Computer Security Resource Center: http://csrc.nist.gov/groups/ST/toolkit/BCM/index.html

Olzak, T. (2006, February). *Data Storage Security.* Retrieved May 19, 2012, from Adventures in Security: http://adventuresinsecurity.com/Papers/Data_Storage_Security.pdf

Ortiz, C. E. (2005, September). *The Security and Trust Services API (SATSA) for J2ME: The Security APIs*. Retrieved 18 2012, May, from Oracle Sun Developer Network: http://developers.sun.com/mobility/apis/articles/satsa2/

RSA. (2009). *Securing Sensitive Data with Tokenization: An Emerging Technology.* Retrieved May 19, 2012, from RSA Security Inc.: http://rsa.com

Singh, S. (1999). *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography* (Kindle ed.). New York: Anchor Books.

Vormetric. (2010). *Vormetric Data Security Architecture.* Retrieved May 19, 2012, from Vormetric, Inc.: http://enterprise-encryption.vormetric.com/wp-vormetric-data-security-architecture.html

Zim, H. S. (1962). *Codes and Secret Writing.* Scholastic Book Services.

Zimmerman, P. (1994, October 11). *PGP User's Guide, Volume 1: Essential Topics*. Retrieved May 19, 2012, from Michigan State University: http://www.pa.msu.edu/reference/pgpdoc1.html#section-4