Introduction to Cryptography and PIN Security

Effective: 1 January 2002

Contents

Overview 1

Background 1

Scope.. 1

Audience 2

What Cryptography Is 3

Secret Codes 5

Substitution Ciphers 5

Transposition Ciphers 6

How DES Works 9

Conclusion 13

Overview

Background

For many years, Visa has taken part in an active worldwide effort designed to raise the general level of compliance with Visa's published PIN security requirements. All of these requirements are based on relevant international and banking industry specifications and are designed to guard the secrecy of a PIN associated with a Visa-branded product, such as Visa, Visa Electron, Interlink, and Plus.

While some of Visa's PIN security requirements cover operational processes, such as equipment inspection and physical security, most concern various aspects of DES (Data Encryption Standard) cryptography, the algorithm used to protect cardholder PINs while they are being processed by the payment system. An understanding of cryptography and its strengths and weaknesses is vital to a manager who has to make business decisions.

This document provides background information on DES cryptography for non-mathematicians so they can work with technical personnel to understand and deal with an increasingly aggressive and capable group of adversaries.

Scope

The scope of Visa's PIN security requirements is limited to those organizations acquiring or processing interchange PINs. This document contains general information about DES and key management. Information relating to specific hardware, application software, or processes is beyond the scope of this introductory document.

Audience

Managers responsible for ATM or other PIN-based debit operations, internal auditors, physical security specialists, and technical staff new to processing that includes cryptography make up the primary target audience for this document.

What Cryptography Is

Cryptography is a process designed to conceal confidential information by changing its appearance and/or structure. The first recorded examples of cryptography come from ancient civilizations, including Babylonia and Egypt. The Greeks and Romans used substitution and transposition ciphers; increasingly complex encoding processes have evolved for civilian and military purposes.

The banking industry has long used cryptography and secret codes to protect confidential information, such as account numbers and balances, wire transfer routing details, and cardholder PIN numbers. Since 1978, the only acceptable method for protecting cardholder PINs has been the Data Encryption Algorithm (DEA) as defined in ANSI X3.92, which is also known as Data Encryption Standard (DES).

Secret Codes

Before today's cryptographic methods were developed, some type of secret code usually protected confidential information. Secret codes fall into two major families: substitution and transposition.

Substitution Ciphers

A substitution code replaces one character of the cleartext information with some other character, which could be a letter, number, or special character.

Example—Substitution Cipher

Cleartext

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 0 1 2 3 4 5 6 7 8 9 $.

Code Equivalence Table

R F $ 5 M T D . S X C 4 Y 7 B Z J 9 A H U E 3 Q 0 V G L 6 N K 1 P O W 2 8 I

Cleartext Message - MEET ME AT 8

Ciphertext Message - YMMH YM RH W

In Edgar Allen Poe's story, "The Gold Bug," the coded instructions which led to the buried treasure were fairly easy to decipher. By using letter frequency counts and common word structures as guideposts, the text became clear, step-by-step. It's a little tougher with numbers and special characters, however. The pirate in Poe's story made the job a lot easier than it might have been otherwise by writing out the entire message in words.

Transposition Ciphers

A transposition cipher is a simpler form of substitution cipher where one character is substituted for another in a predictable pattern. For example, a transposition cipher rule might be to replace every cleartext character with the character 24 positions higher in the alphabet, "wrapping" the last three characters around to the beginning.

Example—Transposition Cipher

Cleartext

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 0 1 2 3 4 5 6 7 8 9

Code Equivalence Table

C D E F G H I J K L M N O P Q R S T U V W X Y Z 0 1 2 3 4 5 6 7 8 9 A B

Cleartext Message - MEET ME AT 8

Ciphertext Message - OGGV OG CV A

This encoding method is susceptible to the same sort of letter frequency and word structure analysis as the substitution cipher example.

Block ciphers make the job of a code breaker tougher. In a block cipher, the ciphertext output is presented in character groups of a preset size, with padding characters used to fill out any partial groups. The primary military code used by the U.S. in World War II was a block cipher, with all output expressed as groups of five characters. Using the substitution and transposition ciphertext examples printed above, with a cipher block length of 4 characters and pad characters of ABC, the following would be produced:

Example—Block Cipher

Substitution Ciphertext - YMMH YM RH W

Block Cipher Version -YMMH YMRH WRF$

Transposition Ciphertext - OGGV OG CD A

Block Cipher Version - OGGV OGCD ACDE

The weaknesses of both substitution and transposition codes became evident during World War II, as codes were broken with increasing ease. In an attempt to create an unbreakable code, Germany devised a combination cipher, which used both substitution and transposition on a letter-by-letter basis. Each letter went through a substitution process, and then the result was transposed to a different position in the alphabet. Finally, the result was formed into blocks of five characters.

A special machine, known as Enigma, was designed to perform this complex process. The Germans were certain that this code was impregnable, but it was broken by a team of British cryptographers located at Bletchley Park, outside London.

Around the same time, the "unbreakable" Japanese Naval code, JN-25, was broken by a team of American cryptographers led by Commander Joe Rochfort. The confidential information obtained by the Allies after the code was broken was crucial to victory in World War II.

The development of computers after the war, combined with manual and semi-automated techniques that had been successfully used by the Allies' code-breaking teams, meant that new methods were required to protect confidential information.

Many of the algorithms which were developed after the war used reversible mathematical transforms to perform encryption and decryption functions. One of the best of these algorithms was the Lucifer process, which was developed by IBM's Watson Research Laboratory in the early 1970s and was offered, royalty-free, to the American National Standards Institute (ANSI). After some modifications and a thorough analysis by various security agencies, this algorithm, now known as DES (Data Encryption Standard), was established as the only acceptable method for protecting cardholder PINs as they passed through the U.S. electronic payment system network. It still is.

How DES Works

DES works by combining confidential information (the cleartext) with an encryption key to form the ciphertext. The ciphertext is neither a substitution cipher nor a transposition cipher, although both substitution and transposition operations take place during the execution of the algorithm. The value of any character in the ciphertext is not predictable, since each cleartext character can assume any of the valid ciphertext character values. For example, the cleartext value '222' might result in the ciphertext '4A6'.

DES is a block cipher. Sixteen hexadecimal characters (8 bytes) of information is input to the algorithm and the same amount of data is delivered as output. If cleartext is input, ciphertext is output; if ciphertext is input, cleartext is output.

Like everything else stored in digital computers, information is fed to DES in the form of binary digits (1s and 0s), also known as a base two numbering system. We are more familiar with a base ten numbering system, and if we could only figure out how to build transistors that could assume one of ten values, instead of only one of two values, we could use base ten.

Hexadecimal (number base 16) is a shorthand way of writing binary numbers. Two squared (multiplied by itself) equals four. Two to the fourth power equals sixteen. We use “Hex” because it is easier to write, infinitely easier to read, and it dramatically reduces data entry or transcription errors. All of the information (cleartext, key, and ciphertext) is actually stored on the computer as binary numbers, not as hexadecimal characters.

For example, a DES encryption key contains 8 bytes, which is 16 hexadecimal characters, or 64 binary digits long.

A typical DES key might have the Hex value of 3A56C228FA510C42.

In binary this is expressed as:
0011101001010110110000100010100011111010010100010000110001000010.

It’s easy to understand why hexadecimal is used.

According to the ANSI (American National Standards Institute) rules, a DES key can be generated by any random or "pseudo"-random process which is neither predictable nor reproducible. Examples of random processes that can be performed by a human being are tossing a coin or drawing a prize-winning ticket out of a

fishbowl. Flipping a coin 64 times, recording a 1 (one) for every "heads" and a 0 (zero) for every "tails" simulates the way a computer would generate a DES key.

Another ANSI rule states that no person must know any portion of the active key, and mathematics provides us with a method to manage DES keys securely by using a type of reversible transform known as the Exclusive Or (XOR) operation. A reversible transform is an operation that converts a starting value to something else. When the same operation is applied to the "something else," it returns the original value. The operation is defined as "The result C is True if A is True or B is True, but not both."

The XOR operator returns one of two values: 1 (True) or 0 (False). When XOR is applied to binary data (the kind that computers use), four results are possible.

Example—XOR Results

A		B		C	
0	+	0	=	0	(A False, B False, C False)
1	+	0	=	1	(A True, B False, C True)
0	+	1	=	1	(A False, B True, C True)
1	+	1	=	0	(A True, B True, C False)

For even more security, two random strings of 64 binary digits can be XOR'ed together by using a secure device, such as a Tamper-Resistant Security Module (TRSM) for the XOR process. For example:

String 1 =
0100000111010101001110011111001001101101101000010011110111100000 0

String 2 =
0010110101001001001010010010111100101001010010010001001010101001

The XOR'ed result =
0110110010011100000100001100101111110010000011110110100101101001

Written in hexadecimal shorthand:

String 1 = 41D539E4DB4273C0

String 2 = 2D49292F294912A9

The result of XORing these two strings = 6C9C10CBF20F6969

Each of the individual component holders knows half of the information necessary to form the final key, but neither person knows any part of the actual key value unless they get information about the

other component. In this way, people can generate and/or manage hard copy key components while maintaining the secrecy of the actual key itself. These principles are known as "dual control and split knowledge."

The following steps illustrate how to use a reversible transform, a key and the XOR process to encrypt a checking account number during electronic transfer:

1. The checking account number is 2445487233. Two decimal digits can be packed into each byte of data. Since the account number is 10 digits long, it can be contained in 5 bytes of storage, but DES always takes 8 bytes in and delivers 8 bytes out.
2. Rather than leave the last 3 bytes as zeros, the normal practice is to "pad" them with a non-zero character to help disguise the key. In this case, the last three bytes are padded with the value Hex 'F'.
3. For more security, the account number is XOR'ed with a key created using the principles of dual control and split knowledge.
4. First, the account number and the padding are converted to binary. The result is: 00100100010001010100100001110010001100111111111111111111111111111111.
5. XOR is used to combine the key to with the data.

   Key = 01101100100111000001000011001011111100100000111101101001011101001

   Data = 00100100010001010100100001110010001100111111111111111111111111111111

   Code = 01001000110110010101100010111001110000011111000010010110100101110

6. The resulting code string is converted to hexadecimal. The account number field of the message now reads 48D958B9C1F09696, which in no way resembles the original account number, 2445487233.

7. There is no substitution or transposition operation that will decode the encrypted value. The only way restore the account number is to XOR the code with the key again, as follows:

   Code =
   0100100011011001010110001011100111000001111100001001011010010110

   Key =
   0110110010011100000100001100101111110010000011110110100101101001

   Data =
   0010010001000101010010000111001000110011111111111111111111111111

8. When the data string that results from the XOR operation is converted back to hexadecimal, the account number plus the padding is restored, just as it was entered, as 2445487233FFFFFF.

The actual DES algorithm adds more steps to the process described above. It starts with a straightforward XOR of the data with the secret key. Then it executes 16 intermediate operations or "rounds" to create the ciphertext. Each round consists of passing the result of the preceding round through an "S" or Substitution-Box. These are not physical boxes, but rather rules that combine, move, and XOR data according to very specific rules. An example of such a rule might be to "XOR Bits 1 and 3 together and place the result in Bit 4." The effect of these operations is to scramble and mask the cleartext to such an extent that only knowledge of the secret key would enable the cleartext data to be retrieved. In the world of cryptographers, these techniques are known as "confusion and diffusion." One important effect of these operations is to thoroughly hide the identity of data that is almost all zeros.

The output from the 16th round of the DES algorithm is the actual ciphertext that is placed into electronic messages for transferring confidential information, such as cardholder account numbers or PINs.

In this way, data and a secret key are combined to create ciphertext that cannot be broken by classical code breaking methods. It is not a substitution cipher, nor is it a transposition cipher, nor even the difficult substitution/transposition process used during the 1930s and 1940s. Rather, it is a reversible mathematical transform that makes use of a secret key and of a series of rules that are individually simple, but incredibly complex as a whole, to scramble confidential data so that it can be transported safely.

Conclusion

The DES algorithm is an effective method for protecting confidential data under certain conditions. However, since the algorithm is published and readily available, the security of the encrypted information depends on preventing anyone from knowing even a part of the secret key. Knowledge of each bit reduces by half the number of combinations needed to "break" the key.

Cryptographic key management processes must protect both the institution and staff that have been asked to serve as key custodians by strictly observing the principles of dual control and split knowledge. By insisting upon randomly generated keys that are managed securely, the level of protection offered by DES when it was adopted as an industry standard can be ensured. Failure to follow these principles can lead to a complete breakdown of security for sensitive data with a corresponding financial exposure.