DES Encryption

Danny Crichton

June 3, 2010

1 Introduction

The history of modern cryptography is intimately linked with the rise of the computer in the 1950s and 1960s. As the capabilities of computers improved, their somewhat limited utility expanded to a diverse set of new applications. One particularly valuable application was archiving private and confidential data, where computers quickly became ubiquitous. As their popularity in this area increased, there was a clear realization among computer programmers, academics and policymakers that the nation was lacking the capability to secure confidential information through encryption. This concern led Congress to enact a law in 1965 known as the Brooks Act mandating the development of new standards and encryption protocols as well as requiring the Secretary of Commerce to promulgate new encryption standards for federal computers.[1] This law eventually resulted in the design of the Data Encryption Standard (DES).

In May 1973, the National Bureau of Standards (NBS) began the public process for developing the DES standard. As an official protocol of the U.S. government, DES had to be published for public comment in the Federal Register. These comments touched off years of debates surrounding the protocol, including questions over the length of the DES encryption key, the involvement of the intelligence community in the protocol's development, and the protocol's resistance to attacks. Despite several negative comments, the NBS published the protocol in January 1977 as an official encryption standard. It is the first cryptographic protocol endorsed by the government.[8]

This paper will begin by laying out the design of the DES protocol and will then consider possible attacks to the standard. Finally, the current status of the protocol will be presented.

2 DES Protocol

This section is based off of the current National Institute of Standards and Technology (the successor of the NBS) standard specification available online.[2]

Before discussing the implementation of DES, some basic design notes of the protocol need to be laid out. DES belongs to a class of cryptographic protocols known as block ciphers, in which the protocol encrypts a defined size block of data into an encrypted block of the same size. The DES protocol uses a block size of 64 bits. If the data to be encrypted is not exactly this size, it needs to be segmented and/or padded to create exactly 64 bit blocks. To encrypt each of these blocks, DES relies on 64 bit keys, with 56 bits used to encrypt data and eight of the bits devoted to parity checks (a form of error detection). DES is a symmetric key algorithm, which means that both the sender and the receiver must use the same key in order to decode the message. In addition to this key, the protocol requires the use of several bit selection tables which are defined as part of the standard and easily locatable online.

2.1 Initial Permutation

First, begin with a single block of 64 bits to be used as the input. Bits are numbered in DES from left-to-right (little endian in Computer Science). The first step of the DES protocol is to scramble this block through the "Initial Permutation" function, which moves each bit to a protocol-defined new location. For example, the first eight bits of this mapping are {58, 50, 42, 34, 26, 18, 10, 2}, meaning that the first bit (the leftmost bit) of the scrambled block was originally bit 58 of the input block. All 64 bits are similarly scrambled based on the complete bit-swapping table, which can be found in the protocol definition online.

2.2 Key Schedule Generation

This scrambled block will eventually be sent through 16 rounds of a "Fesitel function," each of which requires a different 48 bit key. Before continuing the discussion of encrypting our input though, we need to generate a set of 16 keys that can be used for this function. These keys are collectively called a key schedule, and they can be generated using the following algorithm.

First, take the 64 bit key shared between the sender and received and pass it through the "Permuted Choice 1" algorithm. Like the Initial Permutation, this matrix scrambles the locations of the bits in a defined manner. However, it also eliminates every eighth bit, since these are used to ensure parity. Thus, the original 64 bit key has now been reduced to a scrambled 56 bit key. We can divide this new key into two parts: the first 28 bits form C_0 and the last 28 bits form D_0.

The keys are generated by taking the bits in these two segments and shifting them to the left a defined number of times. A left shift in this instance is not like the traditional bit-wise shift operator in typical programming languages, but rather a wraparound shift. Thus, a shift to the left of one of the 28 bit blocks would yield a new block with bits numbered 2, 3 ... 28, 1. Another shift would yield 3, 4 ... 28, 1, 2, and so on. To get the n^{th} key, we iterate through the shift schedule (which is provided in the protocol) starting with these initial 28 bit blocks until we have shifted the needed number of times. For instance, C_1 is obtained from C_0 by shifting to the left once (D_0 follows in the same fashion). Another left shift on C_1 yields C_2, and two left shifts (remember, the schedule requires either a single left shift or two left shifts to get the next key) will yield C_3.

Once we get the C_n or D_n we desire, there is one final thing we need to do in order to turn it into a key. To get the n^{th} key, we need to combine C_n or D_n together into a 56 bit block and then send this combined block through Permutation Choice 2, another scramble matrix. PC-2 takes this 56 bit block and reduces it to 48 bits while also scrambling the bits. Once we have calculate all sixteen keys, we have created our key schedule and can begin encrypting our block.

2.3 Feistel Function

Now that we have our key schedule, we need to return to the scrambled block output from the Initial Permutation and begin encrypting it. First, break the block into two parts, a 32-bit left half and a 32-bit right half, to be called L and R. To get the encrypted block, we need to calculate the following equation:

$$L' = R \tag{1}$$
$$R' = L \oplus f(R, K_n) \tag{2}$$

In words, the new block's left half is just the former block's right half, and the new block's right half is the former block's left half XORed with the Feistel function (see online for information on the XOR function).

The Feistel function takes a 32 bit block input and a 48 bit key and outputs a new 32 bit block. To start, the function expands the input block to 48 bits through another defined matrix, this one called E. We now have two 48 bit blocks: the output of the E matrix (which we will call T), and the key given to the function (K). Calculate M as follows:

$$M = T \oplus K \tag{3}$$

Now we will subdivide M into eight, 6 bit chunks. Each of these chunks is given to a unique selection function named S_1 through S_8, each of which outputs a 4 bit chunk. These S functions are tables with four rows and sixteen columns, and

each cell holds a number between 0 and 15 —essentially four bits of binary. To calculate the row, we use the first and last bits of the input together to get a two bit integer between 0 and 3. To calculate the column, take the middle four bits of the input to get a number between 0 and 15. Find the correct cell to get the encrypted four bits, which then becomes the output. This process is conducted similarly for each of the eight bit chunks from M to get 32 bits of output. This is the result of the Fesitel function.

The process above is considered one round. DES requires that the output from the Initial Permutation go through sixteen rounds of the Feistel function. Thus, the left and right chunks of our encrypted block will move back and forth sixteen times, with each chunk going through the Fesitel function a total of eight times.

2.4 Final Steps

After sixteen rounds of the Feistel function have been applied, we have what is called the preoutput of the DES protocol. To get the final 64 bit encrypted block, we need to send the preoutput through the Inverse Initial Permutation, defined like the earlier Initial Permutation in the DES protocol. Once the bits have been scrambled this last time, we have a DES-encoded block, and the protocol is complete.

2.5 Decryption

The encrypted block output from the DES protocol is then sent to the receiver who wishes to decrypt it. In order to do so, the receiver will need the key K used in the encryption process. There is nothing tricky about decryption. Basically, the process described above for encrypting message is identical to the process for decrypting the message, just in reverse. Importantly, each key in the key schedule must be applied in reverse order (therefore, K_{16} will be used first), and the left and right chunks of the blocks need to be shifted with new equations:

$$R = L' \tag{4}$$

$$L = R' \oplus f(L', K_n) \tag{5}$$

Conduct all sixteen rounds in reverse, and then apply the Initial Permutation to return to the original cleartext.

3 Possible Attacks

As with all cryptographic protocols, there are a range of possible attacks against DES with varying abilities to break the protocol. From the moment it was first

published, DES has been criticized for its small key size, as well as its cryptographic effectiveness. We will now consider several different attacks and evaluate each.

3.1 Brute Force Method

Since the key used by DES is relatively short compared to modern cryptographic systems, there is a potential risk of brute force attacks —essentially trying all 2^{56} keys with an intercepted encrypted message and continuing until an understandable cleartext is found. However, this method is still out of the processing range of most computers. Even so, there have been theoretical computer designs that could be specifically built for the purpose of brute force attacking the DES protocol. By 1997, groups were decrypting DES messages with brute force techniques and in 1998, the Electronic Frontier Foundation — a group dedicated to privacy rights on the internet — demonstrated that DES could be broken in a matter of days with custom utilities.[4, 5] It should be noted that the protocol itself is not at fault here, but rather the relatively short key size compared to newer standards that range from 128 bit to 512 bit encryption. An increase of a single bit squares the number of possible keys, and thus the increase in key size increases the difficulty of breaking the protocol exponentially.

3.2 Differential Cryptanalysis

Another type of attack uses carefully-chosen plaintexts to acquire information about the key used in a DES-encoded message. This type of analysis is called Differential Cryptanalysis because it involves inputting slightly different plaintexts through the DES protocol and then evaluating the ciphertexts that emerge to backtrack to the original key. The goal is to use knowledge of the S bit transform matrices to "follow" a bit from the cleartext to the ciphertext and use the output to create a probability model for possible keys. However, careful design of the S transforms in the DES protocol ensured that this tactic is practically impossible, and nearly as hard as a brute force attack. The National Security Agency was aware of this type of attack before the protocol was originally published, and worked to design the S bit transforms to be immune.[3]

3.3 Intercepting Keys

Both brute force attacks and differential cryptanalysis have been mostly ineffective until recently in breaking the DES protocol. However, there are other means of breaking the protocol then just attempting to determine the key. DES is a symmetric key protocol, which means that both the sender and the receiver of an encrypted communication must have the same key. This key has to be sent in the

clear, or through another encryption protocol. Otherwise, if the key is passed in the clear, it may be easy to intercept. Several years after DES was approved by the NBS, Diffie and Hellman designed the public key protocol, which is a more secure method to protect keys. In modern day communications, one of the evolved forms of DES is often used in conjunction with public key cryptography to create an encrypted channel.

4 Current Status of DES

DES was the first published federal cryptographic standard. Since then, the field of cryptography has expanded dramatically, and dozens of new protocols have been developed. Starting in the 1990s, there was growing concern in the cryptography community about the weakness of DES to possible attacks. These concerns, and the breaking of the protocol by the EFF, led NIST to begin a search for a replacement for the DES protocol. The first protocol was Triple-DES, and the second was the Advanced Encryption System (AES).

The biggest flaw in the DES protocol is widely considered to be its small key size, which at 56 bits provides little protection against today's high-powered computers. A stopgap measure was announced by NIST called Triple-DES that would reuse the design of DES but would rely on three keys instead of one, significantly increasing the effectiveness of DES. The basic idea is that a set of three 56 bit keys will be given to the Triple-DES function, which will process the cleartext in succession through the original DES protocol by first encrypting the cleartext with key one, decrypting it with key two and finally encrypting it with key three (thus there are three layers of encryption since the decryption with key two does not cancel the encryption with key one). Decryption is simply the process in reverse. While the key size has increased, Triple-DES remains at risk because the underlying algorithm itself is beginning to show its age. NIST continued to search for a replacement, finding it in AES.[7]

AES is the direct successor of DES and was approved by the government in November 2001. NIST solicited proposals for possible encryption algorithms, and a total of sixteen were submitted. The final protocol uses the Rijndael cipher to encrypt its data, which is based on an S-box —not related to the S bit transforms in the Feistel function of DES. The possible key lengths for AES are 128 bits, 192 bits and 256 bits, which are significantly longer than DES and prevents some of the brute force attacks possible with the older protocol. Despite similar key lengths to Triple-DES, AES holds several advantages over the older protocol, including a more sophisticated encryption algorithm and a faster implementation. There have been several published attacks against AES, although none of them currently break the protocol. Today, AES is the most common form of encrypted communication

in the world, and remains the standard for federal government security.[6]

References

[1] Public law 89-306 (Brooks Act). October 1965.

[2] Data encryption standard. December 1993.

[3] Mark C. Chu-Carroll. Differential cryptanalysis. October 2008.

[4] Matt Curtin and Justin Dolske. A brute force search of DES keyspace. *;login:*, May 1998.

[5] EFF. "EFF DES cracker" machine brings honesty to crypto debate. July 1998.

[6] Jim Hurst. Explanation of AES, February 2007.

[7] RSA Laboratories. What is triple-DES? 2010.

[8] Miles E. Smid and Dennis K. Branstad. The Data Encryption Standard: Past and future. *Proceedings of the IEEE*, 76(5):550–559, May 1988.