A Known-Plaintext Attack on Two-Key Triple Encryption

Paul C. van Oorschot and Michael J. Wiener

BNR
P.O. Box 3511 Station C
Ottawa, Ontario, Canada, K1Y 4H7

1990 June 29

Abstract. A chosen-plaintext attack on two-key triple encryption noted by Merkle and Hellman is extended to a known-plaintext attack. The known-plaintext attack has lower memory requirements than the chosen-plaintext attack, but has a greater running time. The new attack is a significant improvement over a known-plaintext brute-force attack, but is still not seen as a serious threat to two-key triple encryption.

Key Words. triple encryption, cryptanalysis, DES.

1. Introduction

Due to questions raised (e.g., see [Diff77]) regarding the adequacy of security by the 56-bit key in the Data Encryption Standard (DES) [FIPS46], several varieties of multiple encryption have been considered. Given a few plaintext-ciphertext pairs, an exhaustive search defeats (single) DES in on the order of 2^{56} operations. Double DES encryption, using two independent 56-bit keys (see Figure 1), requires on the order of 2^{112} operations to attack by this naive approach. This may be reduced to on the order of 2^{56} operations and 2^{56} words of memory using a simple "meet-in-the-middle" attack [Diff77].

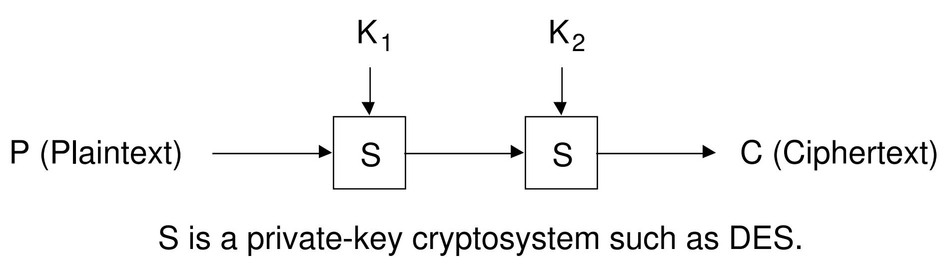

Figure 1: Double Encryption

Two-key triple DES (see Figure 2) can be defeated by the naive approach in on the order of 2^{112} operations. This may be reduced to on the order of 2^{56} operations and 2^{56} words of memory using a chosen-plaintext attack due to Merkle and Hellman which requires 2^{56} chosen-plaintext plaintext-ciphertext pairs [Merk81]. This latter attack, although impractical, is of interest in that it exhibits what Merkle and Hellman refer to as a "certificational" weakness in two-key triple encryption.

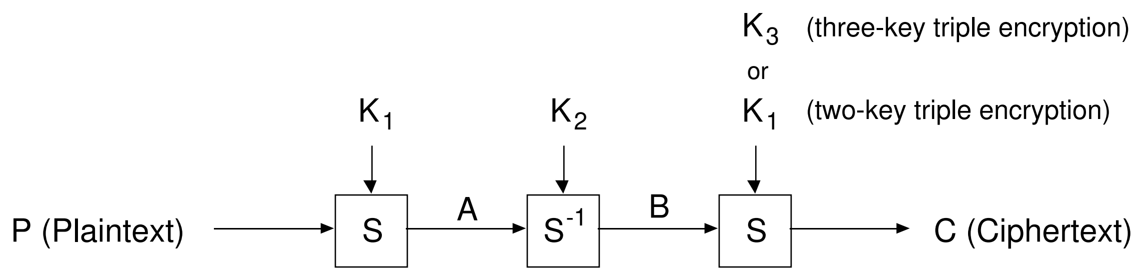

Figure 2: Triple Encryption

This paper presents a *known*-plaintext attack on two-key triple encryption. The Merkle-Hellman attack is first reviewed in §2. The new attack is presented in §3 and briefly analyzed in §4, showing it to require a running time on the order of $2^{120-\log_2 n}$ operations and n words of memory, where n is the number of available plaintext-ciphertext pairs. This is the best known-plaintext attack on two-key triple DES that the authors are aware of. In §5, we consider a hardware implementation of the new attack using $n = 2^{32}$.

As with the Merkle-Hellman attack, the new attack poses no serious threat to two-key triple encryption in practice. However, it is of interest in that it may be used to both reduce the memory requirements and relax the chosen-plaintext condition in the Merkle-Hellman attack, and may lead to further advances. It is also highly amenable to parallel implementation. As with the Merkle-Hellman attack, the ideas discussed in this paper are not restricted to DES, but apply to any similar cipher.

2. The Merkle-Hellman Attack on Two-Key Triple Encryption

Let $C = S_K(P)$ denote that the plaintext P, enciphered using key K, results in ciphertext C. Then as in [Merk81], denote two-key triple encryption by the function $Enc()$:

$$C = Enc(P) = S_{K_1}(S^{-1}_{K_2}(S_{K_1}(P))). \quad (1)$$

Let A and B be the intermediate values in $Enc(P)$:

$$A = S_{K_1}(P) \quad \text{and} \quad B = S^{-1}_{K_2}(A). \quad (2)$$

The Merkle-Hellman attack finds the desired two keys $K_1 = \kappa_1$, $K_2 = \kappa_2$ by finding the plaintext-ciphertext pair such that intermediate value A is 0. The first step is to create a list of all of the plaintexts that could give $A = 0$:

$$P_i = S_i^{-1}(0) \qquad \text{for } i = 0, 1, \ldots, 2^{56} - 1. \tag{3}$$

Each P_i is a chosen plaintext and the corresponding ciphertexts are obtained from the holder of keys κ_1 and κ_2:

$$C_i = Enc(P_i) \qquad \text{for } i = 0, 1, \ldots, 2^{56} - 1. \tag{4}$$

The next step is to calculate the intermediate value B_i for each C_i using $K_3 = K_1 = i$.

$$B_i = S_i^{-1}(C_i) \qquad \text{for } i = 0, 1, \ldots, 2^{56} - 1. \tag{5}$$

A table of triples of the following form is constructed:

$$(P_i \text{ or } B_i, i, \mathit{flag}),$$

where *flag* indicates either a P_i-type or B_i-type triple. Note that the 2^{56} values P_i from equation (3) are also potentially intermediate values B, by equation (2). All P_i and B_i values from equations (3) and (5) are placed in this table, and the table is sorted on the first entry in each triple, and then searched in order to find consecutive P and B values such that $B_i = P_j$. If $B_i = P_j$, then i, j is a candidate for the desired pair of keys κ_1, κ_2. This fact is illustrated in the two-key triple encryption depicted in Figure 3.

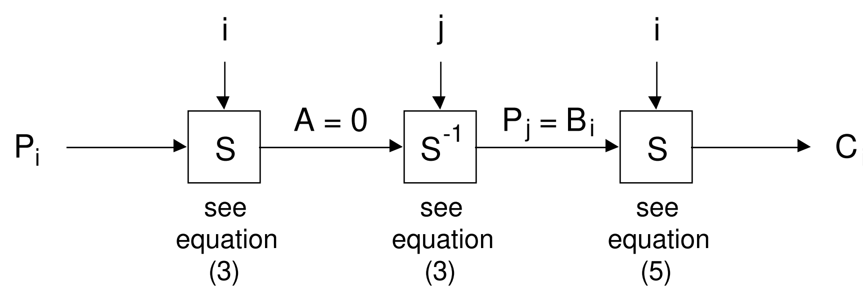

Figure 3: Two-Key Triple Encryption with a Candidate Pair of Keys

Because $C_i = Enc(P_i)$ for both the candidate pair of keys i, j and the desired keys κ_1, κ_2, it is reasonable to expect that the two pairs of keys might be equal. Each candidate pair of keys found from the sorted table is tested on a few other plaintext-ciphertext pairs to filter out "false alarms". The reason the attack succeeds is that a match $P_j = B_i$ is found in the table with $i = \kappa_1$; this is that i for which $S_{\kappa_1}(P_i) = 0$. Testing all candidate pairs guarantees that κ_1 and κ_2 will be found [Merk81].

3. Known-Plaintext Extension of the Merkle-Hellman Attack

Because the Merkle-Hellman algorithm computes a table based on the fixed value $A = 0$, and it is not known *a priori* which plaintext P results in the intermediate value $A = 0$, it is necessary to test all 2^{56} possibilities (i.e., $S_i^{-1}(0)$ for all possible keys i). Also, the attacker must request that each of these plaintexts be enciphered for him by his adversary. This makes the Merkle-Hellman attack far from practical. The idea for extending the algorithm is to remove the reliance on a single, fixed value of A; rather, we choose values for A at random, and for each choice, carry out a tabulation. We continue until a "lucky" choice of A is made which results in the success of the algorithm. As the attacker, we no longer require access to the adversary's *Enc*() function. Instead, we assume that we are given n plaintext-ciphertext pairs.

The new algorithm proceeds as follows. Tabulate the (P, C) pairs, sorted or hashed on the plaintext values (see Table 1 in Figure 4). Table 1 is independent of A and requires $O(n)$ words of storage. Now randomly select and fix (for this stage of computation) a value a for A, and create a second table (see Table 2 in Figure 4) as follows. For each of the 2^{56} possible keys $K_1 = i$, calculate what the plaintext value would be if i were used for K_1:

$$P_i = S_i^{-1}(a).$$

Next, look up P_i in Table 1. If P_i is found in the first column of Table 1, take the corresponding ciphertext value C and compute the intermediate value

$$B = S_i^{-1}(C).$$

Place this value of B along with the key i into Table 2. Table 2 is sorted or hashed on the B values.

Each entry in Table 2 consists of an intermediate B value and corresponding key i which is a candidate for κ_1; as described above, each (B, i) pair is associated with a (P, C) pair from Table 1 which satisfies $S_i(P) = a$. The remaining task is to search for the desired value of K_2. For each of the 2^{56} candidate keys $K_2 = j$, calculate what the intermediate B value would be if j were used for K_2:

$$B_j = S_j^{-1}(a).$$

Next, look up B_j in Table 2. For each appearance of B_j (if any), the corresponding key i along with key j is a candidate for the desired pair of keys κ_1, κ_2. (To handle the rare case that a given B-value appears more than once in Table 2, a few bits could be added in Table 2 entries to indicate the multiplicity of each B-value.) Each candidate pair of keys (i, j) is tested on a few other plaintext-ciphertext pairs. If all of these additional (P, C) pairs have P mapped to C by the key pair (i, j), then $(i, j) = (\kappa_1, \kappa_2)$ and the task is complete.

This algorithm will find κ_1 and κ_2 the first time *any one* of the available (P, C) pairs has a first intermediate value ($S_{\kappa_1}(P)$) that is equal to a chosen a. If the algorithm does not succeed for a given a, the process is repeated for another value of A until ultimately the desired keys κ_1, κ_2 are found.

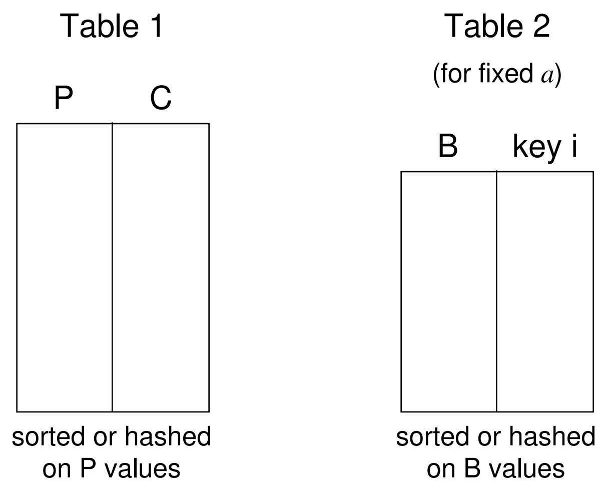

Figure 4: Tables used in the Known-Plaintext Attack

4. Time and Space Analysis

In this section, we briefly summarize the running time and memory requirements of the known-plaintext attack.

The time required for building and hashing Table 1 is the time required to hash n items. This time is dominated by other computations required in the attack, for $n < 2^{56}$. The space required for Table 1 is $O(n)$.

For each value of A that is tried, the time required to build Table 2 is on the order of 2^{56}, assuming that Table 1 is hashed on the plaintext values so that lookups take constant time. Because only 2^{56} out of 2^{64} possible texts are searched for in Table 1, the expected number of entries in Table 2 is $n/2^8$. This space is reusable across different values of A. The time required to work with Table 2 to find candidate pairs of keys is on the order of 2^{56}.

The probability of selecting a value of A that leads to success is $n/2^{64}$. The expected number of draws required to draw one red ball out of a bin containing n red balls and $N - n$ green balls is $(N + 1)/(n + 1)$ if the balls are not replaced. Therefore, assuming that one does not try the same value of a more than once, the expected number of values of a that must be tried is

$$(2^{64} + 1)/(n + 1) \approx 2^{64}/n \quad \text{for } n \text{ large.}$$

Thus, the expected running time for the attack is on the order of $(2^{56})(2^{64}/n) = 2^{120-\log_2 n}$, and the space required is $O(n)$.

5. Parallel Hardware Implementation

In this section we present one possible parallel hardware implementation of the known-plaintext attack on two-key triple DES, assuming that $n = 2^{32}$ plaintext-ciphertext pairs are available. Given a number of assumptions concerning the cost of components and the performance that can be achieved by present-day technology, the illustrated implementation of the attack is shown to be four orders of magnitude faster (for an attacker with fixed resources) than a brute-force known-plaintext attack. This is the best known-plaintext attack the authors are aware of, but this attack is still not feasible. We conclude that two-key triple DES is currently not vulnerable to attack in practice.

The following hardware implementation is suitable for an attacker with a large amount of resources. We will assume that the attacker has 1 billion (10^9) dollars and $n = 2^{32}$ plaintext-ciphertext pairs available to him. Note that the execution time is not particularly sensitive to n (provided that n is not too small) because as n increases, the number of operations required for the attack ($2^{120-\log_2 n}$) decreases, but memory requirements increase, and the number of machines that can be built with a fixed amount of money decreases.

Each machine for attacking two-key triple DES (see Figure 5) consists of a central component containing Table 1, and 512 peripheral components each containing its own version of Table 2 (for distinct sets of values for A).

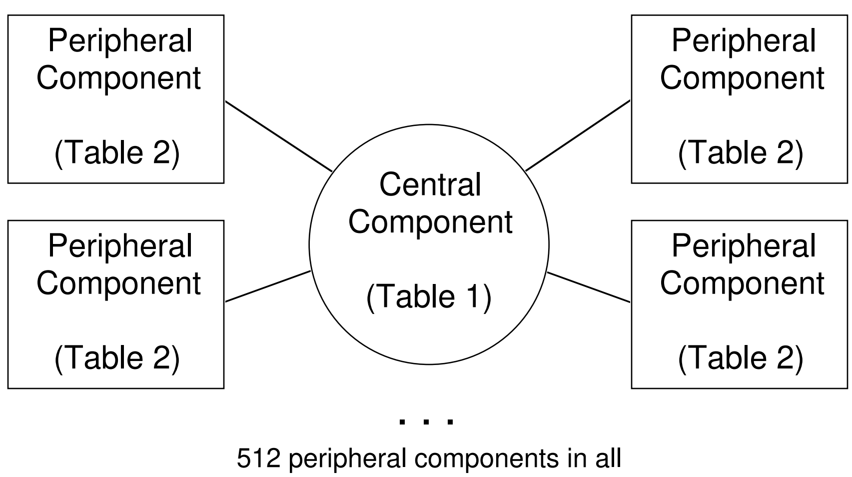

Figure 5: A Single Machine for Attacking Two-Key Triple DES

The function of the central component is to service requests from the peripheral components for the ciphertexts (if any) which correspond to a specified plaintext. In order to service these requests quickly, Table 1 is hashed on the plaintext values. To reduce overhead during table lookup of hashed values caused by hashing collisions, the density of the hashing table is restricted to 50%. In this case, the total memory required for Table 1 is

$$2(2^{32} \text{ words})(64 + 64 = 128 \text{ bits per word}) = 2^{40} \text{ bits}.$$

Assuming that bulk memory can be obtained for \$10/Mbit, the cost of this memory is approximately \$10 million.

If each memory chip is 1M x 1-bit, then Table 1 is organized as approximately 8000 rows, with 128 chips in each row. These rows are independent and can be accessed in parallel. This makes it possible for the central component to service the requests from the peripheral components in parallel. Each request will be directed to one of the 8000 rows. There should be few collisions among 512 requests out of 8000 rows. We will assume that the cost of the complex routing and arbitration circuitry required to make this work will double the cost of the memory making the total cost of a central component \$20 million.

We will assume that the average time required to service a request from a peripheral component is 250 ns. This may seem slow considering the current speed of memories, but this figure takes into account delays caused by the routing and arbitration circuitry, delays due to collisions among the 512 requests, and delays due to hashing collisions which lead to extra probes into Table 1.

The expected number of words required for Table 2 is $n/2^8 = 2^{24}$. Again, restricting the density of Table 2 to 50%, the total memory required for Table 2 is

$$2(2^{24} \text{ words})(64 + 56 + 4 = 124 \text{ bits per word}) \approx 4000 \text{ Mbits}.$$

(Four extra bits have been allocated to handle the problem of possible duplicate B-values as indicated in §3.) Assuming that bulk memory can be obtained for \$10/Mbit, the cost of this memory is \$40 000. For all 512 peripheral components in a machine, the total memory costs are approximately \$20 million. Peripheral components have some circuitry other than memory, such as DES chips, but there is just enough of this circuitry that the 250 ns request rate is not slowed down. The cost of this circuitry is negligible compared to the cost of memory. Then the total cost of one machine is \$40 million. Therefore, the attacker who has \$1 billion can afford to build 25 machines.

The expected number of values of A that must be tried is $2^{64}/n = 2^{32}$. For each value a of A, 2^{56} accesses of Table 1 are required to build Table 2. Also 2^{56} accesses of Table 2 are required to find all candidate pairs of keys. Assuming that accesses of Table 2 also require 250 ns, the expected time required to find the desired pair of keys is

$$(2^{32})(2^{56} + 2^{56})(250 \text{ ns}) \,/\, (25 \times 512 \text{ peripheral components}) \approx 4 \times 10^8 \text{ years}.$$

Next, we consider a brute-force known-plaintext attack. Analysis indicates that a DES chip could be built in volume for about $10/chip [BNR]. A similar chip with added comparison circuitry and modified input/output could be built for about the same cost and used for attacking DES. The cost of building a machine for attacking two-key triple DES would include overhead in addition to the cost of the DES chips; assume this overhead cost to be roughly equal to the total cost of the DES chips. Then for $1 billion, the attacker could afford to build a machine with 50 million DES chips. Using current technology, each DES chip could perform a DES operation in about 500 ns. One would expect to have to search through about half of the 2^{112} pairs of keys, and testing each pair of keys requires 3 DES operations. Therefore, the expected time required for a brute-force search is

$$(3)(0.5)(2^{112})(500 \text{ ns}) / (50 \times 10^6 \text{ DES chips}) \approx 2.5 \times 10^{12} \text{ years.}$$

Therefore, the known-plaintext attack is approximately four orders of magnitude faster than a brute-force search, based on the assumptions made in the preceding arguments. However, this is of little practical consequence unless new ideas improve the running time of the former by several more orders of magnitude.

6. Conclusion

The new attack presented in this paper demonstrates a known-plaintext variation of the chosen-plaintext Merkle-Hellman attack, with a decreased memory requirement. The penalty that is paid for these improvements is increased running time.

The new attack gives approximately four orders of magnitude improvement over a brute-force known-plaintext attack, provided that a sufficient number of plaintext-ciphertext pairs are available. Despite the improvement, for practical purposes, two-key triple encryption remains currently invulnerable to known-plaintext attacks.

The authors encourage others to pursue other known-plaintext attacks on two-key triple encryption, which further reduce the running time.

References

[Merk81] Merkle, R. and M. Hellman, "On the Security of Multiple Encryption", *Communications of the ACM*, vol. 24, no. 7, pp. 465-467, July 1981. See also *Communications of the ACM*, vol. 24, no. 11, p. 776, November 1981.

[Diff77] Diffie, W. and M. Hellman, "Exhaustive Cryptanalysis of the NBS Data Encryption Standard", *Computer*, vol. 10, no. 6, pp. 74-84, June 1977.

[FIPS46] "Data Encryption Standard", National Bureau of Standards (U.S.), Federal Information Processing Standards Publication (FIPS PUB) 46, National Technical Information Service, Springfield VA, 1977.

[BNR] Internal study, BNR, Ottawa, 1989.