DES Encryption Attack Report

wcrr51

January 2021

1 Introduction

This report looks to explore an attack plan to decrypt the following 16-byte ciphertext:

`0x903408ec4d951acfaeb47ca88390c475` (1)

The following information is provided:

- The corresponding plaintext is a *what3words* [**?**] location
- The plaintext was encrypted using DES in ECB mode with a 64-bit key

Firstly, the possible *what3words* plaintexts are

$$\{x.y.z : x, y, z \in Z\} \tag{2}$$

such that

$$\text{len}(x) + \text{len}(y) + \text{len}(z) + 2 \leq 16 \tag{3}$$

where alphabet Z is the set of words in the English dictionary used by *what3words*, and $\text{len}(x)$ is the length (in characters) of word x.

Secondly, an executable encryption oracle is provided. It is relatively slow since every call to it requires a virtual python interpreter to be started. To get around this, many calls to the oracle can be made at once by exploiting the fact that it uses the ECB mode of encryption (where each 64-bit block is independently encrypted with the key).

2 Attack to Identify the Key

While the key is advertised to be 64-bit, DES only uses 56 of these - the remaining 8 are either discarded or used as parity bits. This means the keyspace can be reduced by a factor of 256 by ignoring the last bit of each byte. Thus, a full brute-force attack would take up-to 2^{56} attempts.

As demonstrated in (2), the plaintext must contain two full stop characters (`0x2e`). Without access to an oracle, this, potentially combined with a dictionary, could be used to help check for a valid plaintext when the correct key is used. Given access to the encryption oracle however, this can theoretically be reduced by many factors of 2 ($2^{40} - 2^{50}$) using techniques such as linear cryptanalysis [**?**], and differential cryptanalysis [**?**]. On a single machine, even with GPU acceleration, this may take several days to months.

For this reason, this attack will not be carried out.

3 Attack to Identify only the Plaintext

As the desired outcome is the what3words location, the wiser strategy is to run a plaintext dictionary attack using the encryption oracle to find the three words. Each plaintext guess is sent to the oracle and has its ciphertext checked for equality against (1). This means that for a dictionary of length n, the time complexity of an exhaustive search is $O(n^3)$.

[**?**] states that, in the English version, the minimum and maximum word lengths, $l_{\min}$ and $l_{\max}$ are

$$l_{\min} = \min\{\text{len}(x) : x \in Z\} = 4$$
$$l_{\max} = \max\{\text{len}(x) : x \in Z\} = 18$$

Thus, combining this with (3), the lower bound $p_{\min}$ and upper bound $p_{\max}$ for the length of the

plaintext is

$$p_{\min} = 3l_{\min} + 2 = 14$$
$$p_{\max} = 16$$

These reasonably tight bounds mean that only up to two bytes of padding may be used. In this case it would be sensible to try appending 0x00 and 0x0000 to any applicable ciphertexts of length 15 and 14 respectively. If this does not work, brute force could be attempted on the last one or two bytes (where non-zero padding is used).

For simplicity however, the assumption will be made that no padding is used, hence $p = p_{\max} = 16$. This means there are only a few combinations of word lengths $\{\text{len}(x), len(y), len(z)\}$, these are

$$\{6,4,4\},\{4,6,4\},\{4,4,6\},\{5,5,4\},\{4,5,5\},\{5,4,5\}$$

According to the *what3words* Wikipedia page [**?**], a 40,000-long English word-list is used to cover all sea and land squares. Supposing all words from this could be used in the ciphertext, this, if known, would require up to $40,000^3 = 64 \times 10^{12} \approx 2^{46}$ tests for a guaranteed ciphertext, which is roughly in-line with a linear cryptanalysis attack. Given the presented constraints however, this can be significantly reduced while maintaining an incredibly high chance of attaining success. Hence, any dictionary can safely discard words of length less than 4 and greater than 6, as well as words considered to be rude or offensive. Additionally, the three words are likely to be relatively popular given their short length, and the need for them to be memorable.

The word lists at [**?**] contain around 10,000 of the most popular words. Full searches of these would require up to $10,000^3 = 1 \times 10^{12} \approx 2^{40}$ tests with the oracle. For this example, `google-10000-english-no-swears.txt` will be used, which, by default, contains 9894 words. After removal of words of length less than 4 and greater than 6 this is reduced to 3958, yielding a maximum of $3958^3 \approx 62 \times 10^9 \approx 2^{36}$ oracle tests.

As this attack would require testing two blocks at a time, and given the windows command line input size limit is 32,767 characters, around 1024 block-pairs (depending on the length of the executable filename) can be sent to the oracle at a time. This means the attack can theoretically be carried out with high probability of success in around $61 \times 10^6 \approx 2^{26}$ direct calls to the oracle before accounting for the fact that only some combinations are valid.

4 Attack Results

After about 4.5 hours of carrying out the attack in Python on an i7-6700k with multithreading, using [**?**] as the dictionary, the three words {`'tile', 'bills', 'print'`} are found as the 16-byte plaintext `tile.bills.print` which encrypts to match ciphertext (1). These three words refer to the coordinates `40.026102, -75.030026` in Philadelphia, Pennsylvania, USA.

5 Aside

Hypothetically, if one wanted to access the key in the knowledge that the oracle executable was built using PyInstaller, they could extract the executable's appended data, and from this extract the compiled python source file. Without the need to decompile it, opening it in a text (or hex) editor would reveal that the 64-bit key used is `0x98a1bef23455dc03`.

If it was brute-forced (with all parity bits set to zero), the 64-bit key that is found would be `0x98a0bef23454dc02` (derived from the 56-bit key `0x9942ff934ab701`).