Journal of Engineering and Applied Sciences 13 (Special Issue 5): 4774-4777, 2018
ISSN: 1816-949X
© Medwell Journals, 2018

# Decrees the Possibility of Predicting Initialization Vector by Using Mathematical Methods

[1]Mohammad Jawad Kadhim Abood, [2]Faez Al-Maamori and [1]Wadhah R. Baiee
[1]Department of Software, University of Babylon, Babylon, Iraq
[2]Department of Information Networks, University of Babylon, Babylon, Iraq

**Abstract:** Initialization vector or starting variable is an arbitrary number employed only one time in any session. It can be used along with a secret key for data encryption in several modes in order to randomize the encryption to produce distinct cipher texts even if encrypting the same plaintext several times, so no need to the process of slower re-keying. The IV is nothing more than a fixed-size number that used for data encryption along with a key. Unlike the key, the secrecy of IV (initialization vector) is not usually required. Because of that, reusing the IV destroy the security. In OFB and CTR modes create a bit-stream depend on the password and IV only. In the other hand, the IV in CBC mode must be unpredictable at the time when it is encrypted. An initialization vector has different security requirements than a key, so, the IV usually does not need to be secret. Therefore, OFB and CTR, reusing an IV completely destroys security. This can be seen because both modes effectively create a bitstream dependent on the password and IV only. In CBC mode, the IV must in addition be unpredictable at encryption time because if the man in the middle know the IV he can check his guess in the plain text of some blocks that was use the same key to encrypted. The main attention in this article is how could make the probability of predicting the IV closed enough to zero value. On the other sence: How could make picking up the IV is impossible unless knowing the mathematical approach which used of building the IV.

**Key words:** Analytic number theory, initialization vector, IV attack, probability, picking, building

## INTRODUCTION

The Initialization Vector or (IV) is a property of the mode of operation, not of the block cipher itself. The block cipher is mapping blocks (block size usually depends on the cipher type, 64 bit for DES, 128 bit for AES) unto other blocks. Chaining mode means that how input data transformed into block values and combined the corresponding encrypted blocks.

ECB represent the simplest mode f chaining in which data can be divided into blocks (in the normal way). For a given key, the major challenge is the block-cipher will usually transform the same value of an input block into the same value of output block (deterministic algorithm) which means that messages encrypted by ECB where the input blocks are identical will result an identical block values.

In the CBC the challenge corrected by randomize the input-blocks with XORing each input-block with the encrypted block from the step before: encrypted blocks have a “random look”, so, the occurrence of multiple identical blocks impossible (Fig. 1).

CBC is secure when the plaintext block xored with the previous encrypted block values. To have a good block cipher, the IV must be chosen in a random ways. Suppose we use the same IV with the plaintext messages which start with a same value will output an encrypted messages start with a sequence of block values which is same and the man in the middle will note it.

All the facts above applies to DES and AES equally. Using blocks of 128 bit in AES make it vulnerable less because data in real world tends to show less redundancy at the 128 bit level than it does at the 64 bit level. This is still a weakness and you should use properly generated random IV for AES too.

**Example of IV attack:** In the Cipher Block Chaining (CBC) encryption mode of block ciphers in order to encrypt the data stream, divided it into plaintext blocks P1, P2 and each plaintext block is encrypted as:

$$C_i = f_K(P_i \oplus C_{i-1})$$

And decrypted as:

$$P_i = f_K^{-1}(C_i) \oplus C_{i-1}$$

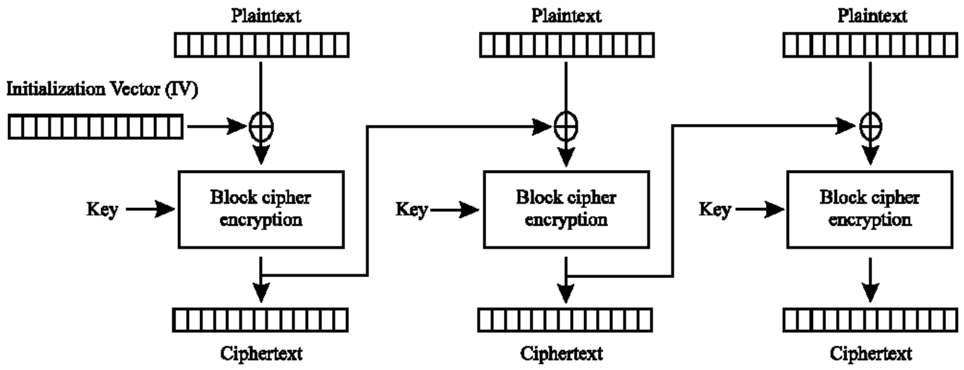

Fig. 1: Cipher Block Chaining (CBC) mode encryption

where, $f_k$ and $f_k^{-1}$ denote the block cipher encryption and decryption and "⊕" denotes the bit-wise exclusive-or. For the special case of the first block where there is no cipher text C0 to xor, an Initialization Vector (IV) is used instead:

$$\begin{aligned} C_1 &= f_K(P_1 \oplus IV) \\ P_1 &= f_K^{-1}(C_1) \oplus IV \end{aligned} \tag{1}$$

The sender is usually choose the IV randomly and send it to the destination. If an IV is sent in without encrypted it which mean it does not protected by any authentication mechanism, the man in the middle can modifying the IV and according to that, he can modify the first block of the decrypted plaintext according to Eq. 1. Such attack are referred as the IV attacks (Al-Maamori and Hilberdink, 2015). Also in the ESP RFC (McCubbin *et al.*, 2000), it is mentioned that the IV "may be carried explicitly in the payload field" and "usually is not encrypted per se".

## MATERIALS AND METHODS

This part of the article addresses the necessary definition and theorem without prof as mentioned by Al-Maamori *et al.* (2016).

**Definition 2.1:** Let P be the set of the actual primes. That is P = {2, 5, 7, ...,} the counting function of prime for any >1 defined to be x real number:

$$C(x) = \sum_{p^k \le x} \log p$$

where, k∈N and p∈P:

$$C(x)\sum_{k=1}^{\infty} \sum_{p \le x^{\frac{1}{k}}} \log p = \sum_{k=1}^{\infty} F\left(x^{\frac{1}{k}}\right)$$

In the other hand. Where:

$$F\left(x^{\frac{1}{r}}\right) = \sum_{p<x} \log p$$

Mobius inversion formula, tell us that:

$$F(x) = \sum_{n=1}^{\infty} \mu(n) C\left(x^{\frac{1}{n}}\right)$$

where, μ (n) is a $^{\text{mābius}}$ function (Al-Maamori and Hilberdink, 2015). Suppose C (x) defined to be the integer part of X minus y, this tell us that C (x) in increasing and tending to ∞ as x→hence:

$$F(x) = \sum_{r=1}^{\infty} \mu(n)\left(\left[x^{\frac{1}{r}}\right] -1\right)$$

One can show that F (x) is an increasing step function and Danish for r>logx/log2. Now, we concentrate on building the prime P from a large real number x. Therefore, we need to go through some relevant theorems from the literature. This theorem are without proof, so we start with (Kent and Atkinson, 1998; Apostol, 1974, 1976).

**Theorem 2.1:** F(x) = F ([x]) for any positive large x. That is for any new n≤x<n+1 ⇒f (x) = f (n).

**Theorem 2.2:** For any n, m∈N, define:

$$g(n, m) = \Delta\left(\left[(m-1)^{\frac{1}{n}}\right], \left[(m)^{\frac{1}{n}}\right]\right)$$

Then:

$$g(n, m) = \begin{cases} 0 \text{if } m \ne r^n \text{ for any } r \text{ in } N \\ 1 \text{if } m = r^n \text{ for som } r \text{ in } N \end{cases}$$

This tell us that if g (n, $r^{\delta}$) is 1 if n divides δ and zero otherwise.

**Theorem 2.3:** The jump of f (x) for any m∈N we have $\Delta f = \Delta$ (f (m-1), f (m)) is either 1 or 0. Furthermore, $\Delta f$ = f (m-1)-f(m) = 1 if f $m \neq n^{r}$ for any n, r>1 and m>2 and also $\Delta f = 0$ if $m^{r}$ for some n&r∈N.

**Remark:** The proof of the above theorems can be seen by Al-Maamori *et al*. (2016), Al-Maamori (2017), Al-Maamori and Hilberdink, 2015). Finally, we end the procedure of building the prime number from a large x by addressing the so havy proposition.

**Proposition 2.4:** For any large real number x let f^(x = (f(x))↑(i (f(x)))) where i(f(x)) is either y or zero. Furthermore, i (f(x)) is zero if f (x) is a multiple of some natural number a way from 1 and f (x) is 1 otherwise.

**Algorithm (generate initialization vector):** By the following algorithm we will explain how we will get P from the output of Mobius function and what we should do to generate IV from P.

**Algorithm 1; Algorithm (generate initialization vector):**

```
Step1: Initialization ()
Step2: Calculate_ Mobius ()
Step3: Generate_Prime ()
Step4: Generate_IV ()
End algorithm
Procedure: Initialization ()
    1.1: n is Integer
                Min $n \leftarrow 1$
                Max $n \leftarrow \infty$
    1.2: X is any Random Float Number
Procedure: Calculate_ Mobius ()
    2.1: if n=1
                $\mu(n) \leftarrow 1$
    2.2: if n is prime
                $\mu(n) \leftarrow -1$
    2.3: if n = pa, a>1
                $\mu(n) \leftarrow 0$
Procedure: Generate_Prime ()
    3.1: $f(n) \leftarrow$(power (X, 1/n)-1)
    3.2: Repeat
    3.3: $ø(X) \leftarrow$sum [($\mu(n)$ f(n))+($\mu(n+1)$
    f(n+1))+...+($\mu(n+n)$)]
    3.4: Until f (n) = 0
    3.5: $ø(X) \leftarrow$nearest prime number to ø(X)
Procedure: Generate_IV ()
    4.1: initialization vector$\leftarrow$ md5(ø(X))
```

## RESULTS AND DISCUSSION

Regarding to the mentioned algorithm, in the third step we got the nearest prime number for a real number after implementing the Mobius function on it.

The real number we used was consist of 8 numbers (as our study case) and it is possible to make it more but for us we find that 8 is enough to implement our research. The result of this step illustrated in the following Table 1.

Table 1: Explain the generated prime number for a random real numbers

| ID | Real number | Mobius function result | The nearest prime number |
|---|---|---|---|
| 1 | 86754212 | 86744897.8058856 | 86744921 |
| 2 | 96742531 | 96732695.2218915 | 96732697 |
| 3 | 68547891 | 68539611.6346258 | 68539613 |

Table 2: Explain the results we got after implement the MD5 on the prime number

| ID | Prime number | IV after implementing MD5 |
|---|---|---|
| 1 | 86744921 | 986699b7a079d2eb3de497fa386f3ec5 |
| 2 | 96732697 | 7a81cc22dbf2ad5544f33d818beed298 |
| 3 | 68539613 | 64b868595cd599ec43660041c633e25c |

Now, we move our attention to satisfy the main aim which is "generate an unpredictable IV". We use the prime numbers in Table 1 to generate the IV as we mentioned in step 4 in our algorithm. We can use any type of hashing functions to encrypt the IV in our case we used MD5 to decries the possibilities of finding the initialization vector and generating the new IV as it is illustrating in the following Table 2.

## CONCLUSION

In the real world the communication between two devices represented in several types of networks. In each type the data will be transfer from device to another, no matter what form it take but it all need to be encrypted (in secure form) to be shore that the no body will be able to read, fabricate or stall it in the way between the source and destination. The sender device use the plaintext to generate the cipher text with the help of a key but this is not enough because the key can be guessed. Therefore, another factor is needed to increase the security. The new factor is initialization vector and it is nothing more than a fixed-size number that used for data encryption along with a key.

The problem is that the IV can be guises in some cases and this process known as the TLS CBC IV attack (when we use CBC mode). In our case, we try to increase the security by decreasing the possibility of predicting the IV. To do that we suggest a mathematical equation to generate anew number as we mentioned in step 2 of our algorithm and from the result number we calculate the P which is the nearest previous prime number from an 8-digit number. The 8-digit number can be more or less depending on the requirement. As a last step, we encrypt the prime number P by md5 to generate the IV.

## REFERENCES

Al-Maamori, F. and T. Hilberdink, 2015. An example in Beurling's theory of generalised primes. Acta Arithmetica, 168: 383-395.

Al-Maamori, F., A. Hadi and A. Al-Salih, 2016. Building the primes P and Q (of the public key algorithm) by using function of reals. Res. J. Appl. Sci., 11: 1050-1053.

Al-Maamori, F.A., 2017. Examples of beurling prime systems. Math. Slovaca, 67: 321-344.

Apostol, T.M., 1974. Mathematical Analysis. 2nd Edn., Addison-Wesley, Boston, Massachusetts, ISBN:9780201002881, Pages: 492.

Apostol, T.M., 1976. Introduction to Analytic Number Theory. Springer, Berlin, Germany, ISBN:9789624300352, Pages: 338.

Kent, S. and R. Atkinson, 1998. IP Encapsulating Security Payload (ESP). Internet RFC., 1: 1-22.

McCubbin, C.B., A.A. Selcuk and D. Sidhu, 2000. Initialization vector attacks on the Ipsec protocol suite. Proeedings of the IEEE 9th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE'2000), June 14-16, 2000, IEEE, Gaithersburg, Maryland, USA., pp: 171-175.