Encryption Key Distribution Applying Key Embedding and Environmental Initiation

Magdy Saeb,

Arab Academy of Science, Technology & Maritime Transport,

Alexandria, Egypt

mail@magdysaeb.net

Abstract: In this approach, we utilize the transmitted encrypted file as a modulated channel that carry the decryption key of the succeeding file. We initialize the procedure employing an environmental random string. The procedure deployed is very similar to a linked list data structure where the data part is the encrypted file and the embedded key, for decrypting the consecutive cipher, is the address part.

Keywords: Key, Distribution, Encryption, Cipher, Pseudorandom, Linked List, Data Structure.

1. Introduction

In a previous work [2], we have demonstrated that we can securely distribute the encryption key by embedding it in the cipher text itself. We presume that steganalysis statistical techniques, at the present, can be hardly successful in this case since we embed a pseudorandom string, utilized to generate the key, in a pseudorandom file (the ciphertext). In addition, the adversary does not know the actual ciphertext file size in advance. This is because of the embedding procedure, which we can modify to hide the cipher text size as well. In section 3 of this discussion, we show, that finding the key and rearrangement of its segments requires more trials than a brute force attack. We describe the method in the following section.

2. The Procedure

In the following few lines, we summarize the procedure required to establish the proposed technique.

Sender:

a. Invoke a Bit String (S_e) located in a shared secret environmental file [1], send m = $h_0(S_e)$ to receiver over an insecure channel,
b. Encrypt a text file using K = $h_1(S_e)$,
c. Embed another PR bit string S_A in the resulting cipher text, which is a pseudo random file, using $L_c = h_2(S_e)$,
d. Send the encrypted file as a modulated channel carrying the consecutive key,
e. Repeat 4 to transmit another encrypted file,

Receiver:

a. For starting the procedure, observe a certain field in the secret environmental file to get S_e,
b. If $h_0(S_e) = m$ then $K = h_1(S_e)$, $L_c(S_A) = h_2(S_e)$,
c. Retrieve the other string S_A using $L_c(S_A)$ to decrypt another file with $K_A = h_1(S_A)$,
d. Decrypt the transmitted file using K,
e. Wait for the consecutive file to be decrypted with K_A,

As shown in Figure 1, the method is similar to a linked list of data (cipher) and address (key). It carries a Payload (file to decrypt) and a Key to decrypt the consecutive file. The initializing trigger is S_e. We will refer to this procedure as Embedded key Distribution or EKD.

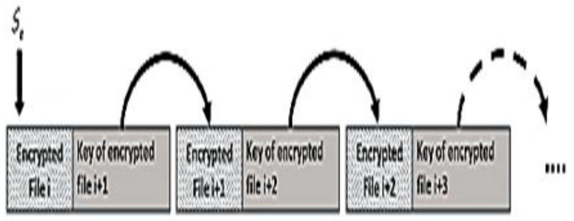
S_e
Encrypted File i
Key of encrypted file i+1
Encrypted File i+1
Key of encrypted file i+2
Encrypted File i+2
Key of encrypted file i+3
....

Fig. 1: The proposed Embedded Key Distribution (EKD)

3. The probability of revealing the key

The probability of exposing the key is the product of the probabilities of finding correctly all the key segments times the probability of arranging them correctly. That is:

Pr {Revealing the key} = Pr {Finding the key segment locations correctly}. Pr {Arranging the segments correctly}

In the accompanying example, we show that the probability of revealing the encryption key is smaller than that of a brute force attack.

Given that the key length is L, the key segment length is l and the encrypted message length is M_L, where L = 128 bits, l = 16 bits, M_L = 256 bits. Then the number of key segments N is given by N = L/l = 128/16 = eight segments. Accordingly, Pr {Revealing the key} = (8/ (256+128)). (1/8!) This is equal to 5.03586 e-7. However, if l = one bit, then N = 128, Pr {Finding the key} = (128/ (256+128). (1/128!) , This is equal to 8.42468 e-217. A brute force attack requires 2^{128} that is equal to 3.40282 e38 trials with a probability of success equal to 2^{-128} or 2.93874 e-39 assuming all of the 128 bits as security bits. Neglecting the message and key length factor, we can find that the breakeven point located just after 35 segments as shown in Figure 2 a and b.

The searched function f (N) is

$$f(N) = (N/N!) - 2^{-128} = (1/(N-1)!) - 2^{-128}$$

Therefore, the number of embedded key bits should be equal or less than the floor of 3.7, which is three bits with the number of segments, is equal to 43 segments. However smaller segment lengths such as two and one bit provide additional security.

In the above analysis, we have neglected the probability of finding the key segments. Now if we assume a file size of one M bits, then this probability is equal to 128/ (128 + $10^{6)}$ which is approximately equal to 10^{-4}. The probability of revealing the key, in this case, will be equal to 8.42468 e-221.

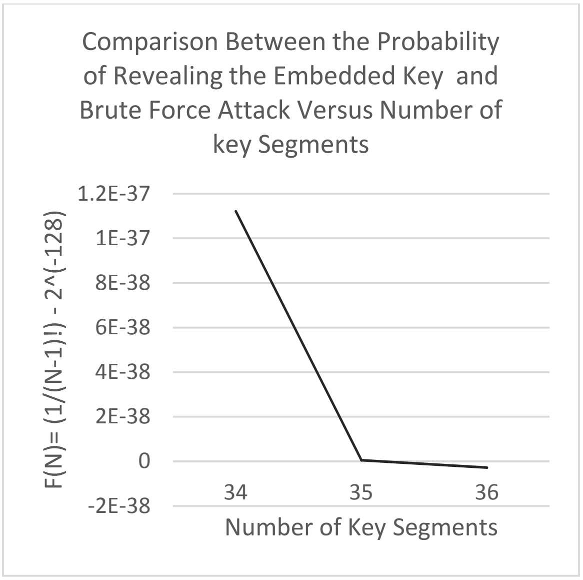

Fig. 2a: A *Comparison between the probability of revealing the embedded key and brute force attack versus number of key segments*

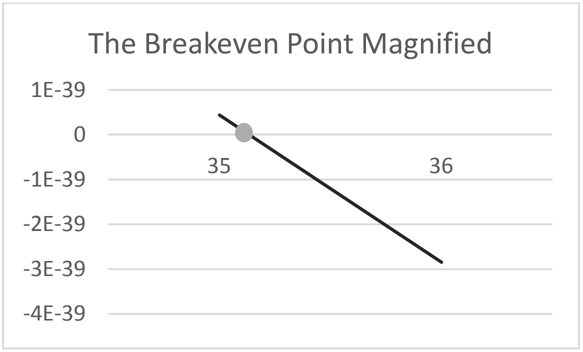

Fig. 2b: The breakeven point magnified

Conclusion

We have demonstrated that the encrypted file can act as a secure channel to carry the decryption key of the consecutive file. Embedding a pseudorandom string in a pseudorandom file leaves no structural clues that may help an opponent using present day steganalysis statistical techniques to retrieve the embedded key. The probability of success of such an attack, we have shown, is far less than a brute force attack. The governing idea behind this approach is to use the cipher itself as a secure channel.

References:

1. Bruce Schneier, James Riordan, “Environmental Key Generation towards Clueless Agents,” Mobile Agents and Security, G. Vigna, ed., Springer-Verlag, pp. 15-24, 1998.

2. Magdy Saeb, "Encryption Key Distribution Applying Steganographic Techniques," The International Journal of Computer Science & Communication Security (IJCSCS), Volume 4, Aug 2014. http://www.ijcscs.org/journal

Magdy Saeb received the BSEE. School of Engineering, Cairo University, in 1974, the MSEE, and Ph.D. Degrees in Electrical & Computer Engineering, University of California, Irvine, in 1981 and 1985, respectively. He was with Kaiser Aerospace and Electronics, Irvine California, and The Atomic Energy Establishment, Anshas, Egypt. He is a professor in the Department of Computer Engineering, Arab Academy of Science, Technology & Maritime Transport, Alexandria, Egypt; He was on-leave working as a principal researcher in the Malaysian Institute of Microelectronic Systems (MIMOS). His current research interests include Cryptography, FPGA Implementations of Cryptography and Steganography Data Security Techniques, Encryption Processors, Mobile Agent Security.
www.magdysaeb.net.