A Protocol for Secret Key Infusion from Satellite Transmissions

Mohamed Ridza Wahiddin[1], Noor Shamila Shanaz Noor Sham[1],

Magdy Saeb[2], Mior Hadi Mior Hamdan[1]

1. Malaysian Institute of Microelectronic Systems (MIMOS), Kuala Lumpur, Malaysia,
2. Arab Academy for Science, Technology & Maritime Transport (AAST), Alexandria, Egypt.

mail@magdysaeb.net

Abstract: *In symmetric encryption systems, secure messages are exchanged with the encrypted message sent over a public channel and the key over a secure channel. Quantum key distribution is an alternative method to share a common key without having to transmit the key. However, it is still in the development stage and possibly proves to be expensive and impractical in the foreseen future. It is useful to have a cheaper and simpler protocol to locally generate the secret key at the sender and receiver locations. In this work, we propose the capturing of bits from an external source such as a radio or a satellite digital transmission to locally generate a mutual key. We introduce a synchronization technique and a framework to ensure that both parties will be receiving the same bit string. The protocol stages include a synchronizing bit-pattern, capturing a bit string and then processing it. The resulting pseudo random bit string is then hashed to provide the required key. The nature of the signal, the vast selection of the timeframe, the downloading sequence, the various source stations and other parameters ensure eavesdropping to be exceedingly unmanageable.*

Keywords

Key distribution protocol, secure key exchange, hash function.

1. Introduction

Secret key exchange for symmetric encryption is an essential element of data security between two entities. On the other hand, public key is a necessary building block of asymmetric encryption. However, public key cryptography consumes significant amount of power and computing resources. Moreover, public key cryptography has inherently certain limitations for sensitive data security applications. Quantum Key Distribution (QKD) is a good example of an innovation that does not use a public key [1, 2]. For sharing a secret between a sender and a receiver, QKD uses the Heisenberg's uncertainty principle. While quantum key distribution has been receiving a lot of attention by various researchers [3, 4], it is still expensive and very rare in practical applications. A more practical, a lot less expensive and a more flexible solution to the problem of sharing secret keys between two entities, say Alice and Bob, is to use existing sources of random bits, such as satellite transmissions. These transmissions act as the source of seeding bits to a pseudo random number generator (PRG) that is utilized to locally-generate the secret key between the two communicating entities. Our proposed method extracts the secret key by using a string of bits, downloaded from an agreed-upon satellite transmission, processing it and then utilizing it as a seed for a PRG. The received string of bits should be exactly the same for Alice and Bob if the locally-generated keys are to be identical. This can be achieved by using some synchronizing technique between the sender and receiver. The synchronizing technique is based on utilizing an agreed-upon synchronizing bit pattern such as $7E_h$ or equivalently 0111110_2. We call this procedure the Key Infusion Protocol (KIP). Compared to other methods using line and other types of communication noise to generate the key [5, 6, 7], our proposed technique is more reliable and does not depend on the notion of identical noise at both ends of the communication link and that the adversary cannot be very close to one of the communicating entities. In the following sections, we discuss the protocol, its security, implementation issues and finally, we provide a summary and our conclusions.

2. Methodology

The key infusion protocol (KIP) can be summarized as follows:

1. Start, at time 0, listening to an agreed-upon satellite channel,
2. Wait for the first occurrence of a specific bit pattern; say $7E_h$ or 01111110_2,
3. If this pattern occurs, store a specific number of bits in a particular sequence (say the next 4096 bits),
4. Apply Von Neumann corrector on the stored bits,
5. Apply XOR corrector on the resulting bit string,
6. Apply a selected hash function (based on an agreed-upon field in the resulting bit string) on the resulting random string along with the present time and date as input to get the key,
7. Store key,
8. Go back to step 1.

Optional: To ensure identical keys, exchange their hashes between the communicating entities using a different hash function from the one used in step 6.

The starting time can be chosen based on a global atomic clock [8]. In step 3, the sequence of the stored 4096 bits does not have to be directly downloaded but can be downloaded in an agreed-upon manner. The key size depends on the hash function used in step 6. For larger key sizes, one can resort to either using hash functions that provide the correct size or using concatenated multiple keys of smaller sizes. The number of stored bits, namely 4096, is just a suggested number. The sender and receiver can change this number as part of the shared secret. The shared secret is based on the following parameters:

- The start time, or what we refer to as time 0,
- The agreed-upon satellite transmission station,
- The bit pattern, $7E_h$, can be also changed between the two communicating entities,
- The number of stored bits can be also changed,
- The hash function utilized can be previously selected or can be randomly selected from a set of hash functions based on certain bits resulting from the XOR operation.

For less secure applications, all of these parameters can be considered public except the starting time (day, time) which has to be kept secret. The Von Neumann and XOR correctors are well- known techniques that were introduced in the1950's. These techniques are discussed in detail in [9]. The locally-generated keys can be checked that they are identical by exchanging their hashes between the two communicating entities. However, this step is not an essential protocol requirement.

3. The Adversary Model

In this protocol, the adversary Eve can listen to all communications between Alice and Bob. Eve also knows the key infusion protocol. However, the parameters utilized in the protocol are kept secret from the adversary. Eve has to acquire the time of the start of the protocol, the knowledge of the utilized bit-pattern, the number of bits downloaded and the hash function used to generate the key. If we assume that there is no evil Alice or Bob in this model, the adversary has negligible probability of guessing all of these parameters correctly. Still, in less-secure applications, some of these parameters can be made public such as the synchronizing bit-pattern and the number of bits downloaded. The essential parameter that is to be kept secret is the starting time and, in some cases, the hash table used to generate the key. Basically, Eve cannot disrupt the key infusion process between Alice and Bob since it is performed locally at their respective sites. Moreover, we presume that Eve cannot induce a person-in-the-middle attack, i.e., our protocol does not provide an authentication process for either Alice or Bob. Some authentication procedure has to be utilized to provide the required security for the protocol to oppose an active adversary. The thought discussed in this article, allows the generation of relatively large keys that are suitable even for one-time pad (OTP) based encryption [10, 11]. As mentioned before, this can be achieved by concatenating smaller keys to build up larger ones.

4. Implementation

The simplicity of the method would lend itself considerably to different types of software and hardware implementations. A desk-top computer equipped with a satellite receiver card and screen capture card with the required software can be used. A mobile phone with GPS receiver capability can also be used. Moreover, the Internet as a source of random bits can be also used where a given site page can be processed as a string of random bits to generate the key as described before. The simplicity of the Von Neumann corrector algorithm can be hardware-implemented as shown in Appendix A2 using xor logic gate and a tri-state buffer. The randomness tests suggested by NIST test suite are related to the hash functions utilized to generate the key bits rather than to the proposed key infusion protocol. In other words, randomness tests are not required here to validate the correctness of the proposed procedure since a hash function should provide this randomness based on an arbitrary input. Otherwise, it should not have been used in the first place.

Summary & Conclusions

We have introduced a protocol for key infusion from satellite or other digital radio transmissions. We have also shown that the adversary has a minuscule chance of guessing the key if all the parameters used in the protocol are kept secret. However, for less secure applications, some of these parameters can be made public to facilitate the application of the protocol. The only shared secret can be reduced to be the listening starting time. There is no restriction on the key size that is locally-generated. Therefore, generating large keys, that are even equal to the size of the file to be encrypted, is quite feasible. This may lead to the use of pseudo one–time pad based encryption in security-critical applications. The simplicity of the proposed technique, with the increased sources of pseudo random bits will probably establish its suitability for present day data security requirements.

References

[1] CH. Bennett, F. Bessette, G. Brassard, L. Salvail, and J. Smolin, "Experimental quantum cryptography," *J. Cryptol,"* 5(1):3–28, 1992.

[2] G. Brassard and L. Salvail, "Secret key reconciliation by public discussion," Lecture *Notes in Computer Science*, 765:410–423, 1994.

[3] Jesni Shamsul Shaari, Mohamed Ridza Wahiddin, "Qunits in Two Way Quantum Key Distribution," First International Conference on Quantum, Nano, and Micro Technologies (ICQNM'07), pp. 11, 2007.

[4] L. Greenemeier, "Election Fix? Switzerland Tests Quantum Cryptography," Scientific *American*, October 2007.

[5] Suman Jana, Sriram Nandha Premnath, Mike Clark, Sneha K. Kasera, Neal Patwari† Srikanth V. Krishnamurthy,**"**On the Effectiveness of Secret Key Extraction from Wireless Signal Strength in Real Environments," *MobiCom'09,* , Beijing, China, September 20–25, 2009.

[6] B. Azimi-Sadjadi, A. Kiayias, A. Mercado, and B. Yener, "Robust key generation from signal envelopes in wireless Networks," *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security*, pages 401–410, Nov. 2007.

[7] V. Brik, S. Banerjee, M. Gruteser, and S. Oh, "PARADIS: Wireless device identification with radiometric signatures," *ACM MOBICOM Conference*, Sept. 2008.

[8] http://www.worldtimeserver.com/atomic-clock/

[9] Robert Davies, Hardware Random Number Generators, http://statsresearch.co.nz, http://www.robertnz.net, 14, October, 2000.

[10] Michael O. Rabin, "Hyper Encryption and Everlasting Secrets; A Survey," Lecture Notes in Computer Science, Volume 2653/2003, p. 634, Springer Berlin / Heidelberg, 2003.

[11] Michael O. Rabin, Hyper Encryption by Virtual Satellite, Science Center Research lecture Series, Harvard, http://athome.harvard.edu/programs/hvs/, May 29, 2004.

Appendices:

A 1: An Example using this procedure:
Assuming that the downloaded string of bits, say 32 bits for simplicity, is given by:

1001 0110 1000 1000 0111 1010 1011 0001,
Applying Von Neumann Corrector, we get:
10011101110,
Applying the XOR corrector, we get:
11010

Applying a hash function on this string, say MD5, we get:
a1a2c3fed88e9b3ba5bc3625c074a04e

Applying SHA1, we get:
7b599c349de601d9b5a39e378f6dcc250c389191

Applying SHA512, we get:
84e8fcd444d6158314fa3033061aa244f64264cb0da7
bae251dc33d9c1c96747f4f5e89519a75b86a70051da
8de72e9d4176708965a47fe49d3e719cf07fa5db

A 2: A proposed digital circuitry for hardware Von Neumann corrector implementation

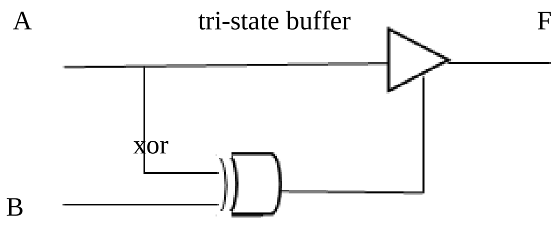

Figure A2. Von Neumann Corrector Implementation

The truth table of the circuit, shown above in Fig. A 2, is given by:

Table A1: Circuit truth table

A	B	F
0	0	hi-Z *
0	1	0
1	0	1
1	1	hi-Z

* Hi-impedance state