4-2006

An Asymptotic Secrecy Model and Intelligent Systems

Bo Yuan
Rochester Institute of Technology

An Asymptotic Secrecy Model and Intelligent Systems

Bo Yuan
Department of Networking, Security, and Systems Administration
Thomas Golisano College of Computing and Information Sciences
Rochester Institute of Technology
Rochester, New York 14623
E-mail: bo.yuan@rit.edu

Abstract

This paper proposes a new model for secure communication channels between two parties. The new model assumes that adversaries are storage space bounded, but not computationally bounded. At the initial phase of the secret communication, both parties exchange a large amount of random bits so that adversaries are not able to save them due to the storage space limitation. Each party only saves received data. At the second phase, each party regenerates the random bits, combines with received data, and generates an encryption key iteratively with a one-way hash function. The key is, then, used to encrypt the first transmission from one party to the other. After each transmission, the key is updated iteratively based on data received. The relationships between the model and intelligent systems are discussed.

1. Introduction

The basic problem in cryptography is secure communication over an insecure channel. Party *A* wants to send to Party *B* a secret message over a communication line which may be tapped by an adversary. The traditional solution can be illustrated in Figure 1. It requires two channels to realize secure communication between Party *A* and Party *B*. It is called the secret key approach. Through one channel with guaranteed secrecy, Party *A* and Party *B* exchange an agreed upon encryption method *E* and its associated secret key *k* and decryption method *D*. On the other insecure channel, Party *A* sends a cipher text $c=E(m,k)$ to Party *B*. When Party *B* receives the cipher text *c*, it performs $D(c,k)=D(E(m,k),k)=m$ to decrypt the cipher text. The adversary *C* should not be able to decrypt the cipher text without knowing the encryption method *E* and the secret key *k*.

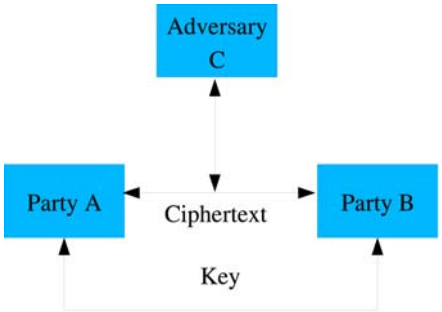

Figure 1: Traditional Solution

In 1949, Shannon proved that to reach absolute security, the length of the key, *k*, needs to be at least as long as the message *m* itself [1]. It is assumed that the adversary has unlimited computational resources in Shannon's theory. Another problem with the traditional solution is the key distribution. It requires a guaranteed secure channel to exchange a common key to both Party *A* and Party *B*. This is very difficult to realize in many real world applications, especially for today's online applications of e-commerce.

In 1976, Diffie and Hellman solved this key distribution problem in their seminal paper [2] and started so-called modern cryptography. Diffie and Hellman introduced a public key distribution system to eliminate the need of a secure key distribution channel. It is based on assumptions that the adversary is computationally bounded. That is, it is computationally infeasible for the adversary to decrypt cipher text. More specifically, the public key encryption is based on assumptions that some one way functions are "easy" to compute but "hard" to invert. Examples of such one way functions are factoring a very large integer, discrete logarithm, RSA functions, etc. For detailed discussion of these functions refer to [3]. Many applications have been developed for the public

key cryptography. In fact, the public key cryptography has enabled private data transactions on the Internet; and online shopping, online banking, and e-commence become reality.

However, the public key cryptography also has its shortcomings. First, the public key cryptography is also susceptible to the man in the middle attack [4]. Thus a public key infrastructure has to be established to authenticate public keys themselves. Secondly, the computationally infeasibility of some problems may be just temporary. As a matter of fact, in [5], Shor has shown that factoring large integers and discrete logarithm can be performed in polynomial time on a quantum computer. It leaves a possibility that a passive eavesdropper can record all secret communications between two parties, and later with more advanced computational algorithms and hardware, the adversary is able to decipher messages.

To overcome the temporary nature of the computational infeasibility assumed in the public key based cryptography, Aumann, Ding and Rabin introduced a bounded storage model in [6]. In the bounded storage model, it is assumed that an adversary is computationally unbounded, but is bounded by the amount of storage available to store the output of computation. The authors also proved information-theoretic security in this model. However, the storage bounded model does require a shared secret key by both parties, which may be the reason why the storage bounded model is not as successful as the public key model in real world applications.

In this paper, we introduce a model, called the asymptotic secrecy model, to address both issues of the secret key sharing and computational infeasibility. We also discuss the relationship between the model and intelligent systems and how both two can benefit from each other.

2. The Asymptotic Secrecy Model

In this model, the following assumptions are made.

1. The channel between Party *A* and *B* is noiseless and a passive adversary can capture all data exchanged between *A* and *B*.
2. The adversary is storage space bounded. That is, the storage space of the adversary is less than the total storage spaces available to Party *A* and *B*.

Many information based security theories assume a noise channel between two parties, which implies that an adversary is not able to obtain the exact the same information as parties involved. With this assumption, researchers are able to prove that the information-theoretic security is able to be achieved with privacy amplification. See [7], [8] and [9]. In many real world applications, especially today's data networks, noiseless channels are usually required. It may be the reason why it is hard to find real world applications for information-theoretic based security theories. In our model, however, it is the second assumption that limits an adversary's capability of knowing all communications between *A* and *B*. Hence, our model can be regarded as a special case of the information theoretic cryptography. The protocol of the asymptotic secrecy model can be described in three phases.

2.1 Phase I: Initialization

Party *A* and Party *B* exchange unencrypted random texts. Both parties save only the received random texts. They exchange enough random texts so as to consume all of their storage spaces. Since the adversary's storage space is less than the total storage spaces of both parties, the adversary is not able to keep all random text exchanged between *A* and *B*.

Suppose R_1, R_2, R_3,…, R_p are p random bit streams exchanged between Party *A* and Party *B*. $l_i = |R_i|$ denotes the length of the bit stream i for i=1,...,p. Suppose again R_i is transmitted from *A* to *B*, when i is odd; from *B* to *A* when i is even. Thus the total number of bits transmitted between *A* and *B* is

$$Bits_{Total} = \sum_{i=1}^{p} l_i$$

Total bits Party *A* received is

$$Bits_A = \sum_{i \text{ is even}} l_i$$

and total bits received by Party *B* is

$$Bits_A = \sum_{i \text{ is odd}} l_i$$

It is easy to see $Bits_{Total} = Bits_A + Bits_B$. Note that when l_i is 0, it means no bits exchanged between Party *A* and Party *B*.

The assumption that the adversary is storage space bounded, then, can be formulated as

$$Bits_{Total} > M$$

where *M* is the size in bits of the maximum storage space that the adversary possesses.

2.2 Phase II: Generating a shared secret

Both parties regenerate random bit streams, combining with save the bit streams, and use it as the shared secret for both *A* and *B* to generate a key with a current timestamp. Since the adversary is not able to save all the random text exchanged in Phase I, it does not have the shared secret. The timestamp is public. The current timestamp can be synchronized with a message exchange. It is used to avoid the adversary to pre-calculate keys in order to save storage spaces. A reasonably good pseudo random number generator should be used with given seeds so that *A* and *B* can regenerate identical random bits. It should not be reversible, i.e., from some random bits to reconstruct the pseudo random number generate algorithm or seeds used.

With this the shared knowledge, we generate an encryption key in the following manner.

Suppose $f : \{0,1\}^n \to \{0,1\}^m$ with $m<n$ is a one-way hash function which is known to both parties and the adversary. Then a secret key *k* can be generated by the following equations.

$$k = f_p(R_p)$$

where

$$f_i(R_i) = f(R_i \| f_{i-1}(R_{i-1}))$$

for i=1,...,p; R_0 should be the current timestamp; and f_0 is the identity function. The symbol || stands for the concatenation of the two bit streams.

2.3 Phase III: Updating keys

Suppose $m_1, m_2, \ldots, m_t$ are *t* messages needed to be exchanged between Party *A* and *B*. Then, the cipher text should be generated by

$$c_i = E(m_i, f_i(m_{i-1}))$$

for i=1,...,t and m_0=k, where *k* is the key generated in Phase II. Then, decryption is performed by the following equation.

$$m_i = D(c_i, f_i(m_{i-1})) = D(E(m_i, f_i(m_{i-1})), f_i(m_{i-1}))$$

for i=1,...,t.

3. Analysis of the Protocol

The basic assumption of the protocol is that the adversary is storage space bounded. Its limit is less than the total storage spaces of Party *A* and Party *B*. When the adversary collects all data exchanged between *A* and *B* from the very beginning, the adversary does not have enough storage space to save all captured data, hence it does not have shared information between *A* and *B*. For *A* and *B*, they just need to store all data received from the other party and they are able to regenerate their own data transmitted to the other party. If the adversary does not collect data from the very beginning, it does not know the shared data either. This protocol does not exclude the old fashion security practice, i.e., exchanging a shared security through a guaranteed channel, which, in fact, provides more confidence in the privacy of the channel between Party *A* and Party *B*.

One concern, one may raise, is that both Party *A* and *B* do not know the storage space limit that the adversary has, thus, Party *A* and *B* are not sure how secure their channel really is. There are many way to mitigate this risk. For example, first, both parties should maximize the usage of the opposite party's storage space by transmitting random bit streams as much as it can. Secondly, Party *A* and *B* can exchange some apparently public meaningless information to discourage the adversary to save the exchanged data. Third, party *A* and *B* should use the private

channel only when it is necessary. Thus, data exchanged on the open channel can be used as common knowledge to generate secret keys, and the adversary may not save data on the open channel.

On the other hand, the same concern can be raised for computationally bounded assumptions as well. We should assume adversaries are resource limited individuals and not big organizations or governments.

The function *f* in the protocol is a one-way hash function. It can be any hash function with reasonable strength. From Phase II of the protocol, we can see that the initial secret key is derived by applying the hash function *f* iteratively in same way that an iterative hash function is applied. Note that the function *f* may be an iterative hash function itself. According to [10], an iterative hash function is at least as secure as its underline compression function, i.e., no iterative hash function. Thus, the initial key generated in Phase II should be reasonably strong.

The worse case scenario is when the size of the storage space of the adversary is just one bit less than the total storage space of Party *A* and *B*. In this case, the adversary has all common knowledge between Party *A* and Party *B* except one bit shy in R_p. This requires that the hash function *f* should be sensitive to at least one bit difference in inputs. That is, two inputs with only one bit difference should yield very different hash values. On the other hand, the adversary can guess the missing bit, since it is just one bit. In general, the security of the protocol relies on how much bits that the adversary cannot save. The more bits the adversary misses the smaller probability it has to guess the key correctly. Thus, it is obvious to verify the following proposition.

Proposition 1 *Suppose the maximum total storage space of the two parties A and B is N bits and the maximum storage space of the adversary is M<N bits. Then the probability that the adversary obtains the initial key k is*

$$P(k) = 2^{M-N}.$$

On the other hand, the protocol can be applied iteratively. That is, at the second round of the protocol, instead of transmitting plain random bit streams, both parties transmit encrypted random bit streams.

Proposition 2 (Asymptotic Security) *Suppose the maximum total storage space of the two parties A and B is N bits and the maximum storage space of the adversary is M<N bits. Then the probability that the adversary obtains the initial key k is*

$$P(k) = 2^{K(M-N)},$$

where K is the number of rounds the protocol is applied.

As $K \to \infty, P(k) \to 0,$ since $M - N < 0$. Thus, Proposition 2 indicates that even if *A* and *B* have just a few extra bits than the adversary, they still can achieve the asymptotic security for the communication channel by applying the protocol iteratively.

From Phase III of the protocol, we can see that encryption keys are updated whenever new data is received. Thus, encryption keys are never reused to minimize the risk of key discovery attacks. This leaves the adversaries the only choice of brute-force attacks. Since keys are hash values, dictionary attacks do not apply. With a reasonable length of hash values such as 256 bits coupled with a strong encryption algorithm *E*, it will be very difficult to crack keys for the adversary. It is also a symmetric encryption; *E* can be selected as a strong algorithm without compromise the performance.

4. Intelligent Systems and the Asymptotic Security Model

There are two facets in relationships between intelligent systems and the asymptotic security model. On one hand, intelligent systems can be applied to enhance overall security of the model. On the other hand, the model can be applied to intelligent systems to enhance their securities in communications among their modules and/or subsystems.

As intelligent systems become more and more complex, their subsystems, modules or components become more and more focused to certain specific tasks. More often these subsystems may be separated physically and have to communicate via unsecured channels. For instance, an intelligent sensor network is comprised of many sensor nodes that are deployed to a wide spread area to collect various information. These sensor nodes can not only operate independently to perform specific tasks they are programmed to do, but also collaborate among them so as to function as a team. Some of them just relay data from sensor nodes to a central system for further process. When communicating between subsystems within an intelligent system or between two independent intelligent systems, over unsecured channels, encryption is the only choice to guarantee the confidentiality and integrity of the data. Our asymptotic security model is particularly suitable for communications between independent

intelligent systems, when the two systems talks for the first time and they do not have prior shared knowledge to derive a secret encryption key.

Figure 2 depicts an automatic key updating scheme that can work with the asymptotic security model for communications between independent intelligent systems.

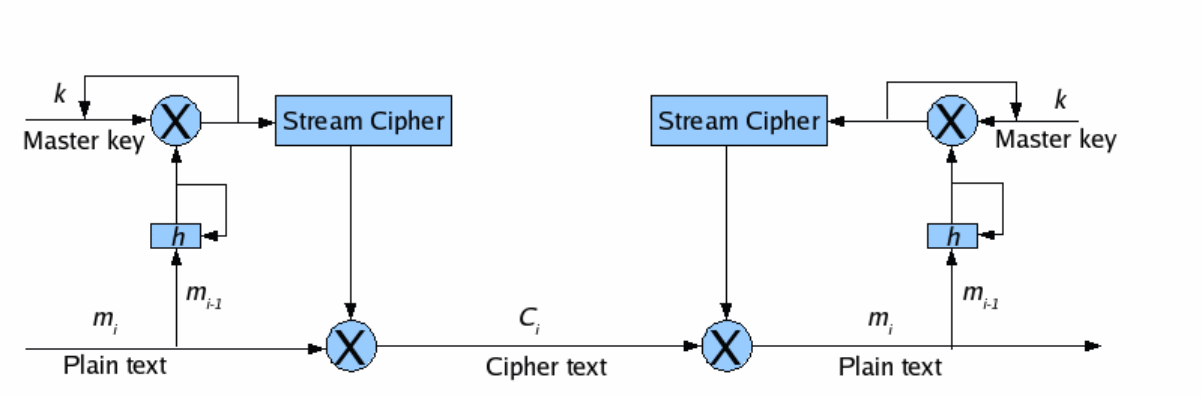

Figure 2 An Automatic Key Updating Scheme.

The basic idea of the automatic key updating is to XOR a hash value of previously transmitted secret messages with the current key to generate the next encryption key. More formally,

$$k_i = k_{i-1} \otimes h_i(m_{i-1})$$
$$h_i(m_{i-1}) = h(m_{i-1} \| h_i(m_{i-2}))$$

where m_{i-1} is the secret message transmitted under key k_{i-1}, for $i \in \{1,2,3,...\}$; and h is a one-way hash function; k_0 is a shared master key; m_0 is random data generated during the key agreement phase; $\|$ denotes string concatenation; and we define m_{-1} and h_0 are null string and function, respectively.

When Part *A* transmits a message very first time to Part *B*, the first message from *A* to *B* is the cipher text of an random text m_0 encrypted with the shared master key k_0, which can be established via the asymptotic security model; the second message is the cipher text of m_1 encrypted with a new key that is the XOR of the previous key with the hash value of m_0; the third message is the cipher text of m_2 encrypted with another new key that is XOR of the previous key with the hash value of the concatenated string of m_1 and the hash value of m_0; and so on.

At the receiver side, the decryption process is symmetric. Key updating is identical. Both sides share the same one-way hash function and the secret key.

An encryption algorithm paired with the proposed key auto-updating scheme has the following characteristics.

- Each message is encrypted with a different key. Keys are constantly updated with respect to messages.
- Encryption and decryption keys are self-synchronized assuming the receiver receives all cipher text.
- It has the "perfect forward secrecy" property. That is, if one key is compromised, messages encrypted under other keys are still secure.

The encryption scheme, however, has also some weaknesses. Some encryption algorithm needs an initialization time before encrypting any message. The more key updates, the more initialization time is spent. Thus prolong the overall transmitting time. If the communication channels are noisy, and the receiver misses some messages, the encryption and decryption keys are not synchronized. These weaknesses can be dealt with some additional configurations and flow control.

On the other hand, intelligent systems can be employed to enhance the overall security of the model. For instance, instead of just taking a hash of previously transmitted messages, one can use an intelligent system to extract knowledge from previous messages, than take the hash of the extracted knowledge. Further more, an intelligent system can help both the receiver and the transmitter synchronize encryption keys.

5. Conclusions

This paper proposed an asymptotic secrecy model based on the assumption that an adversary is storage bounded. A protocol of the model is introduced and its properties are discusses. An automatic key update scheme is also

introduced to work together with the asymptotic model. They can be applied to facilitate communications between intelligent systems. And intelligent systems can also be employed to work with the asymptotic model.

6. References

[1] C. E. Shannon, "Communication theory of secrecy systems," *Bell System Technical Journal*, vol. 28, pp. 657–715, 1949.

[2] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Trans. Inform. Theory*, vol. IT-22, pp. 644–654, Nov. 1976.

[3] S. Goldwasser and M. Bellare, *Lecture Notes on Cryptography*.

http://www.cs.ucsd.edu/users/mihir/papers/gb.html, 1996.

[4] S. M. Bellovin and M. Merritt, "An attack on the interlock protocol when used for authentication," *IEEE Transactions on Information Theory*, vol. 41, pp. 273–276, January 1994.

[5] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM J. Computing*, vol. 26, pp. 1484–1509, 1997.

[6] Y. Aumann, Y. Z. Ding, and M. O. Rabin, "Everlasting security in the bounded storage model," *IEEE Transactions on Information Theory*, vol. 48, pp. 1668–1680, June 2002.

[7] U. Maurer and S. Wolf, "Secret-key agreement over unauthenticated public channels part i: Definitions and a completeness results," *IEEE Transactions on Information Theory*, vol. 49, pp. 822–831, April 2003.

[8] ------, "Secret-key agreement over unauthenticated public channels part ii: The simulatability condition," *IEEE Transactions on Information Theory*, vol. 49, pp. 832–838, April 2003.

[9] ------, "Secret-key agreement over unauthenticated public channels part iii: Privacy amplification," *IEEE Transactions on Information Theory*, vol. 49, pp. 839–851, April 2003.

[10] B. Schneider, *Applied Cryptography*.1em plus 0.5em minus 0.4emNew York: Wiley, 1996.