6-2007

Secure Communications with an Asymptotic Secrecy Model

Bo Yuan
Rochester Institute of Technology

Recommended Citation

B. Yuan, Secure Communications with An Asymptotic Secrecy Model, Knowledge-Bsed systems (2007), doi: 10.1016/j.knosys.2007.01.006

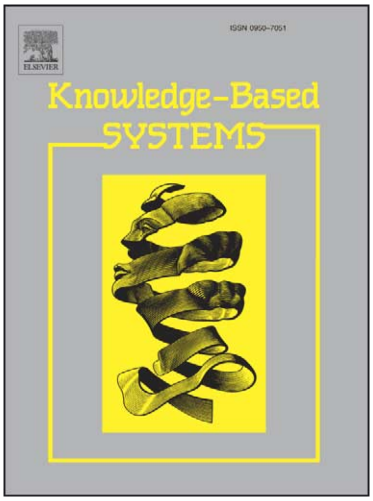
Secure Communications with An Asymptotic Secrecy Model

Bo Yuan

PII: S0950-7051(07)00015-9
DOI: 10.1016/j.knosys.2007.01.006
Reference: KNOSYS 1650

To appear in: *Knowledge-Bsed systems*

Received Date: 15 September 2006
Revised Date: 14 December 2006
Accepted Date: 8 January 2007



Secure Communications with An Asymptotic Secrecy Model

Bo Yuan

Department of Networking, Security, and Systems Administration
B. Thomas Golisano College of Computing and Information Sciences
Rochester Institute of Technology
Rochester, New York 14623

Abstract

As knowledge based systems become more sophisticated, communications between systems or among their subsystems often conducted over public channels such as the Internet, wireless medium, etc. To secure communications over public channels, the most often used method is Diffie and Hellman's public key infrastructure approach. This method requires a trusted third party to verify identifies, which does not play well with independent knowledge based systems, especially in the case of autonomous agents. In this paper, we proposes an asymptotic secrecy model to secure communications between and within knowledge based systems over public channels. 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 all of them due to the storage space limitation. Each party only saves received data. At the second phase, each party regenerates the random bits, combines them with received data, and generates an encryption key iteratively with a one-way hash function. The key is then used to encrypt the future transmissions from one party to the other. After each transmission, the key is also updated iteratively based on data received. Finally, the proposed model is applied to solve some problems in wireless sensor networks as examples to show how the model can be applied for knowledge based systems in general.

Key words: knowledge based systems, security, asymptotic secrecy model, wireless sensor networks
PACS:

Email address: `bo.yuan@rit.edu` (Bo Yuan).

1 Introduction

A basic problem in cryptography is securing communications over public channels. 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 communications 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 agree upon encryption method, E, its associated secret key, k, and decryption method, D. Over the other public 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.

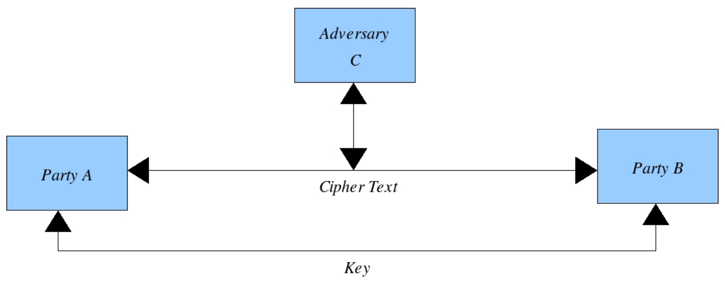


Fig. 1. The Traditional Solution

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. One 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 include one way functions which factoring a very large integer, the discrete logarithm, RSA functions, etc. For detailed discussion of these functions refer to [3]. Many applications have been developed for public key cryptography. In fact, public key cryptography has enabled private data transactions on the Internet; online shopping, online banking, and e-commence have become reality.

However, the public key cryptography also has its shortcomings. First, public key cryptography is 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 temporary. As a matter of fact, Shor has shown that factoring large integers and discrete logarithms can be performed in polynomial time on a quantum computer [5]. This 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 could decipher the messages.

To overcome the temporary nature of the computational infeasibility assumed in public key based cryptography, Aumann, Ding and Rabin introduced an 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 proposed by Aumann, Ding and Rabin in [6] still requires a shared secret key by both parties to select shared a bit stream from a public random bit stream.

In this paper, an asymptotic secrecy model is introduced to address both issues of the secret key sharing and computational infeasibility. Its applications in knowledge based systems are also discussed. Finally, to demonstrate how the proposed model can be applied in general knowledge based systems, the model is applied to solve two problems in wireless sensor networks: the encryption key update problem and the shared key establishment problem.

2 The Asymptotic Secrecy Model

In the proposed secrecy model, the following are assumed.

(1) The channel between Party A and B is noiseless. That is, a passive adversary can hear all information 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 exactly same information as the legitimate 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 required. In the proposed asymptotic secrecy model, however, it is the second assumption that limits an adversary's capability of knowing all communications between A and B. Hence, the asymptotic secrecy model can be regarded as a special case of 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, even if the adversary is able to hear all messages exchanged on the noiseless channel.

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

$$Bits_{total} = \sum_{i=1}^{p} l_i. \tag{1}$$

Total bits Party A received is

$$Bits_A = \sum_{i=1, i|2}^{p} l_i, \tag{2}$$

and total bits received by Party B is

$$Bits_B = \sum_{i=1, i\nmid 2}^{p} l_i. \tag{3}$$

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

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

$$Bits_{total} > M, \tag{4}$$

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, combine them with saved bit streams, and use the combination 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. The timestamp is used to prevent the adversary from obtaining pre-calculated 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., it should not be possible to reconstruct the pseudo random numbers, seeds, or algorithm from some amount of random bits.

With this the shared knowledge, an encryption key can be generated in the following manner. Suppose $f : \{0,1\}^n \longrightarrow \{0,1\}^m$ 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.

$$k = f_p(R_p) \tag{5}$$

where

$$f_i(R_i) = f(R_i \parallel f_{i-1}(R_{i-1})) \tag{6}$$

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

2.3 Phase III: Updating keys

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

$$c_i = E(m_i, f_i(m_{i-1})) \tag{7}$$

for $i = 1, ..., t$, $m_{-1} = NULL$ and $m_0 = k$, where k is the key generated in Eq. 5. Then, decryption is performed by the following equation.

$$\begin{aligned} m_i &= D(c_i, f_i(m_{i-1})) \\ &= D(E(m_i, f_i(m_{i-1})), f_i(m_{i-1})) \end{aligned} \tag{8}$$

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 simply 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. Note that 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 is that both Party A and B do not know the storage space limit of the adversary, thus, Party A and B are not certain of the security of their channel. There are many ways to mitigate this risk. First, both parties should maximize the usage of the opposite party's storage space by transmitting random bit streams as much as possible. Secondly, Party A and B can exchange some apparently public meaningless information to discourage the adversary from saving the exchanged data. Third, party A and B should use a 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. Adversaries are assumed to be 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 Eq. 5, 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 underlying compression function, i.e., no iterative hash function. Thus, the initial key generated by Eq. 5 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 in R_p. This requires that the hash function f should be sensitive to at least one bit differences 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 many bits the adversary is unable to save. The more bits the adversary misses, the smaller the probability it has to guess the correct bits. It is obvious to see 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*

$$P(k) = 2^{(M-N)}. \tag{9}$$

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 $R_1, R_2, R_3, ..., R_p$, both parties transmit encrypted random bit streams generated by Eq. 7. In this case, we obtain the asymptotic secrecy model.

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*

$$P(k) = 2^{K(M-N)}. \tag{10}$$

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

Proposition 2 indicates that even if A and B have just a few extra bits than the adversary, they can still achieve asymptotic secrecy for the communication channel by applying the protocol iteratively. That is, $P(k) \to 0$, as $K \to \infty$.

From Eq. 7, we can see that encryption keys are updated whenever new data is received. Encryption keys are never reused, thus minimizing the risk of key discovery attacks. This leaves the adversary 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 for the adversary to crack keys.

4 Securing Communications of Knowledge Based Systems

As knowledge based systems become ubiquitous and more complex, communications between systems and among subsystems are required for many applications. Very often these communications need to be conducted over public channels such as the Internet, open air medium, etc. To secure communications over public channels, public key infrastructure (PKI) is often the only choice. However, PKI requires a trusted third party to certify public keys. It is an independent body from a knowledge based system; and it is very computationally complex; and it does not suit for those systems with limited computational and power resources.

The proposed asymptotic secrecy model can be applied to knowledge based systems to enhance their securities in communications among their modules and/or subsystems. As knowledge based systems become more and more complex, their subsystems, modules or components become more and more focused to certain specific tasks. In many applications these subsystems may be separated physically and have to communicate via unsecured channels. For instance, a wireless 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 that 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 a knowledge based system or between two independent systems, over unsecured channels, encryption is the only choice to guarantee the confidentiality and integrity of the data. The proposed asymptotic security model is particularly suitable for communications between independent knowledge based 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 knowledge based systems. 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,

$$k_i = k_{i-1} \otimes h_i(m_{i-1}) \tag{11}$$
$$h_i(m_{i-1}) = h(m_{i-1} \| h_{i-1}(m_{i-2})) \tag{12}$$

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 m_{-1} and h_0 are defined as null string and function, respectively. When Part A transmits a message to Part B, the first message from A to B is the cipher text of a random text, m_0, encrypted with the shared master key, k_0; 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.

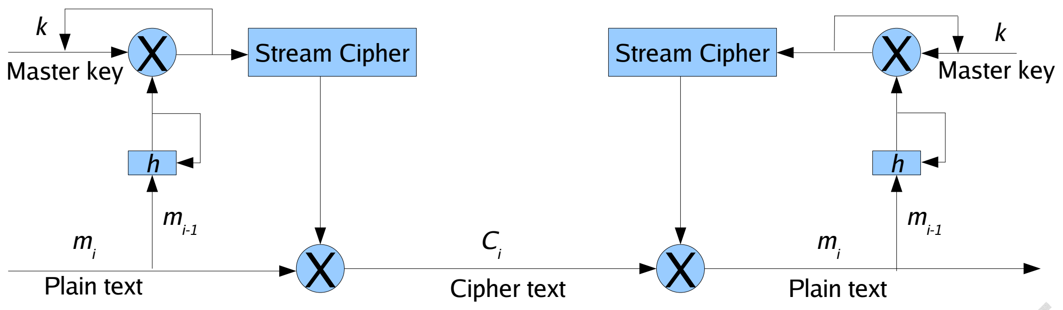


Fig. 2. An Encryption Scheme with Self-updating Keys

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.

Note that in Figure 2, a stream cipher is used. The key auto-updating scheme is not specific to any cipher type. However, a stream cipher is preferred in sensor networks as reported by Luo et al in [11].

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 received 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.

This encryption scheme has also some weaknesses:

- Some encryption algorithms need an initialization time before encrypting message. The more key updates, the more initialization time is spent, thus prolonging the overall transmitting time.
- It is common in real world applications that some messages may never reach the intended receiver. If the receiver misses some messages, the encryption and decryption keys are not synchronized.

To address the weakness of the key auto-updating scheme, first we can configure how often a key is updated according to the encryption algorithm employed. If the encryption algorithm needs a longer time to initialize a new key, less frequent updating should be specified. However, the hash value, which is used to generate the key, still needs frequent updates. More formally, we can use the following equations.

$$k_i = \begin{cases} k_{i-1} \otimes h_i(m_{i-1}), & \text{for } i \equiv 0 \mod T, \\ k_{i-1}, & \text{otherwise.} \end{cases} \tag{13}$$

where T is an integer that specifies how often to update the key. When $T = 1$, the encryption key is updated after every message; when $T = 2$, it is updated every other message, etc. This can be set according to the encryption algorithm so that best overall performance is reached.

To address the problem of message loss, the so-called efficient ACK method can be implemented. After transmitting T messages, the sender waits for an acknowledge message from the receiver. The acknowledge message can be very compact. It can be as little as T bits with bit 1 indicating a received message and bit 0 indicating a missed message. When the sender receives the ACK, it uses only those messages acknowledged to update the encryption key. Note that this ACK message is also encrypted with the current encryption key.

5 Applications in Wireless Sensor Networks

A wireless sensor network can be considered as a complex system. Wireless nodes are deployed to an area where people are difficult or don't want to reach. Nodes collect data then relay them back to a central location for processing. Nodes also accept commands and instructions from a central control system for their tasks. Thus, sensor nodes are actually a part of a big system that includes the central control system and data processing unit, etc.

Security challenges for wireless sensor networks fall into the following areas. First, sensor nodes in typical wireless sensor networks are usually resource bounded. They have small computational power, low memory capacity and low transmission bandwidth. These limitations prevent certain popular security models, such as PKI and encryption algorithms, from being employed in sensor networks. Secondly, sensor networks are often deployed in hostile environments. Sensor nodes are subject to capture and tampering. Wireless transmissions among sensor nodes can be easily overheard by adversaries. Hence, resilience against node capture and eavesdropping are required for wireless sensor networks.

Several security schemes have been proposed for wireless sensor networks. Perrig et al proposed SPINS framework in [12]. In SPNS, It is assumed that each node shares a secret key with a base station in the SPINS framework. Two sensors cannot establish a secret key directly. They need to use the base station as a trusted third party to establish a shared secret key. One advantage of this scheme is that each node only needs to store one shared key with its base station. The disadvantage, however, is that additional key exchange messages need to be transmitted between base stations and nodes. This scheme does not work in those sensor networks that do not have base stations.

Eschenauer and Gligor [13] originated the random key predistribution scheme based on random graph theory. Before they are deployed, sensor nodes are preloaded with some number of random keys from a large key pool. After deployment, a common random key that two neighboring nodes process, if it exists, is used as a shared secret key to encrypt communications between the two nodes. A shared random key is not guaranteed, but it is proved that as the number of total nodes in the network n increases, each node needs to preload $(n-1)(ln(n)+c)/n$ random keys from the key pool in order to ensure the network connectivity with a probability $P_c = e^{-e^{-c}}$, where c is a constant. The advantage of this scheme is that it does not require a key exchange protocol. There are no additional messages exchanged. A disadvantage of this scheme is that each node needs to store $(n-1)(ln(n)+c)/n$ number of random keys. Another disadvantage is that there is no mechanism to update the keys.

In LEAP, proposed by Zhu et al [14], each node is preloaded with a common master key. Once deployed, each node immediately identifies its neighbors and generates shared keys for all of its one-hop neighbors; then, the master key is erased from memory. One advantage of LEAP is its resilience to node capture. If one node is captured, only its neighboring nodes are potentially vulnerable to attack. One disadvantage of the LEAP, however, is that it does not have a key update mechanism. It is usually a not good practice to use a fixed encryption key for a long period of time. First, a single key encryption will provide a large amount of cipher text for adversaries attempting to crack. Secondly, if the encryption key is compromised, all previous transmitted data with the same key is also compromised. In other words, the LEAP encryption scheme does not provide so-called "perfect forward secrecy." Using the asymptotic secrecy model introduced previously, we can solve the following two problems in sensor networks.

(1) updating secret keys to achieve "perfect forward secrecy"
(2) establishing secure communication channels for non-neighboring nodes

To achieve "perfect forward secrecy" in sensor networks, it is assumed that the same initial master key distribution method as in LEAP is used. That is, before deployment all nodes are loaded with a shared secret master key. Once deployed, each node identifies its neighboring nodes and exchange some random text using the asymptotic secrecy model with the master key as the initial encryption key. As soon as this is done, each node stores a shared secret with each of its neighboring nodes, and the master key is erased from memory.

In fact, a time can be set so that if a node cannot find a neighboring node with certain time, the master key is erased automatically. Thus, after a pre-set time, there is no node possessing the master key. This property will mitigate the node capture problem of the sensor networks. When two neighbors need to communicate to each other, they use the stored shared secret and the key automatic update scheme depicted in Figure 2 to encrypt and decrypt their messages. With constantly updated keys, “perfect forward secrecy” is achieved.

When non neighboring sensor nodes need to communicate to each other, a shared secret needs to be established. In [15], Chan, Perrig and Song introduced a multi-path key reinforcement scheme to mitigate the compromised node problem. First, two nodes, Party A and Party B, identify a common key, k, from the predistributed key pool. This method assumes that all disjoint paths between the two nodes are known. Part A generates n random text messages, $v_1, v_2, \cdots, v_n$ and sends them via n different paths to Party B. Then a new key can be generated by $k' = k \otimes v_1 \otimes v_2 \otimes \cdots \otimes v_n$. The secrecy of the new key is protected by all n random text messages, since the adversary has difficulty eavesdropping on all n paths. As pointed by the authors, the more paths used, the more security the new key provides for the channel between Parties A and B.

The multi-path approach by Chan, Perrig and Song is used as a reinforcement to the secrecy of a link between Parties A and B. It is assumed that Party A and Party B already share a key from the predistribution key pool. In the asymptotic secrecy model, however, there is no shared key between Parties A and B. The initial shared master key should have been erased from memory a short time after the deployment. Also note that it may not be easy to find multiple disjoint paths between two nodes.

Suppose a path between non-neighboring nodes A and B is known. The protocol of the asymptotic secrecy can be used to establish a shared secret between A and B. At the initial stage of the protocol, both A and B transmit random text to each other and exhaust storage spaces of both nodes. Since all nodes should have more or less the same storage space in the same sensor network, the total amount of random text is about twice the storage space of any compromised nodes on the path. Thus, compromised nodes are not able to store all random text. Moreover, hop to hop transmissions on the path are all encrypted with shared secrets of neighboring nodes. Nodes that are not on the path should not be able to understand the traffic on the path. When A and B send encrypted data, even nodes on the path between A and B do not understand the traffic passing between them. Therefore, the asymptotic model can be used to establish a share secret between non neighboring nodes with just one path.

6 Conclusions

The proposed asymptotic secrecy model is based on the assumption that an adversary is storage bounded. It is a special case of the privacy amplification in information-theoretic security. The protocol of the model is applied to solve two problems in wireless sensor networks. One is the “perfect forward secrecy” problem; the other is the problem of the shared secret establishment between non neighboring nodes. It is demonstrated that how the asymptotic secrecy model can be applied to knowledge based or intelligent systems in general.

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

References

[1] C. E. Shannon, Communication theory of secrecy systems, Bell System Technical Journal 28 (1949) 657–715.

[2] W. Diffie, M. E. Hellman, New directions in cryptography, IEEE Trans. Inform. Theory IT-22 (1976) 644–654.

[3] S. Goldwasser, M. Bellare, Lecture Notes on Cryptography, http://www.cs.ucsd.edu/users/mihir/papers/gb.html, 1996.

[4] S. M. Bellovin, M. Merritt, An attack on the interlock protocol when used for authentication, IEEE Transactions on Information Theory 41 (1994) 273–276.

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

[6] Y. Aumann, Y. Z. Ding, M. O. Rabin, Everlasting security in the bounded storage model, IEEE Transactions on Information Theory 48 (2002) 1668–1680.

[7] U. Maurer, S. Wolf, Secret-key agreement over unauthenticated public channels part I: Definitions and a completeness results, IEEE Transactions on Information Theory 49 (2003) 822–831.

[8] U. Maurer, S. Wolf, Secret-key agreement over unauthenticated public channels part II: The simulatability condition, IEEE Transactions on Information Theory 49 (2003) 832–838.

[9] U. Maurer, S. Wolf, Secret-key agreement over unauthenticated public channels part III: Privacy amplification, IEEE Transactions on Information Theory 49 (2003) 839–851.

[10] B. Schneider, Applied Cryptography, Wiley, New York, 1996.

[11] X. Luo, K. Zheng, Y. Pan, Z. Wu, Encryption algorithms comparisons for wireless networked sensors, in: Proceedings of IEEE International Conference on Systems, Man and Cybernetics, 2004, pp. 1142–1146.

[12] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, D. E. Culler, SPINS: Security protocols for sensor networks, Wireless Networks 8 (2002) 521–534.

[13] L. Eschenauer, V. D. Gligor, A key-management scheme for distributed sensor networks, in: Proceedings of the 9th ACM Conference on Computer and Communications Security, Washington, DC, 2002, pp. 41–47.

[14] S. Zhu, S. Setia, S. Jajodia, LEAP: Efficient security mechanisms for large-scale distributed sensor networks, in: CCS'03, 2003, pp. 62–72.

[15] H. Chan, A. Perrig, D. Song, Random key predistribution schemes for sensor networks, in: IEEE Symposium on Security and Privacy, 2003, pp. 197–213.