An Improved Key-Agreement Protocol for Channels with Small Error Rate

Albert Guan[1], Chin-Laung Lei[2]

[1] Department of Applied Mathematics, National Sun Yat-sen University, Taiwan
[2] Department of Electrical Engineering, National Taiwan University, Taiwan
albertguan@math.nsysu.edu.tw, lei@cc.ee.ntu.edu.tw

Abstract

Symmetric cryptosystems are very important for secure communication. They are much more efficient then asymmetric cryptosystems. Secret keys are required for symmetric cryptosystems, such as AES. In 2017, Guan and Tzeng designed a secret key establishment protocol whose security depends on the unpredictability of noise in communication channel. They showed that these protocols are secure even if the eavesdropper has infinite computing power. Their protocol works most efficiently for channels with bit-flip rate around 0.01. For the case when bit-flip rate is much smaller than 0.01, their protocol needs a long random bit string to accumulate enough uncertainty to establish a secret key. In this article, an improved protocol for establishing secret key is proposed. The new method requires much less random bits for channels with small bit-flip rates, such as in the range [0.001, 0.010]. With the improvement of wireless communication technology, the bit-flip rate is reducing. Therefore, the new method is suitable for the current wireless networks technology.

Keywords: Key agreement, Lightweight protocol, Wireless network, Random noise, Binary symmetric channel

1 Introduction

A wireless sensor network typically consists of a set of sensors that monitor the environment, collect local data, and send it to the server. Sensor networks have many applications, such as military surveillance, industrial process monitoring, health care monitoring. For almost all applications, these sensors must constantly send and receive data in a secure way. Sensor nodes are usually resource limited devices, thus computationally light-weight cryptography protocols are required for these devices.

Secret keys are indispensable in secure communication. Diffie-Hellman key exchange protocol is a well-known protocol for establishing secret keys [1]. However, using Diffie-Hellman key exchange protocol to establish secret keys needs to consider two issues, especially for devices with limited computing resources. First, Diffie-Hellman key exchange protocol requires modulo exponentiation, which is a heavy computation. Second, Diffie-Hellman key exchange protocol is computationally secure, which implies that an attacker with enough computing power can break the protocol. It is known that quantum computer can solve discrete logarithm problem in polynomial time [2]. Therefore, Diffie-Hellman key exchange protocol is not secure if quantum computers are available.

Noise in the communication channel is usually not useful in message transmission. However, unpredictable noise can be useful in the design of cryptographic protocols which is secure against quantum computers. Guan and Tzeng used unpredictable noise in communication for establishing secret keys [3]. Their protocol can be described briefly as follows. Let A and B be the two nodes in the network that are trying to establish a common secret key for secure communication. Note that these nodes can be IoT devices which need to communicate securely. The first step of their protocol is to let the two nodes receive random bit strings from the same source at the same time. Due to noise in the communication channel, the messages received by A and B, as well as the message received by the eavesdropper, may not be the same. Guan and Tzeng have designed a simple and efficient method for nodes A and B to adjust their messages so that the adjusted message will be equal with very high probability. They have shown that, even if the eavesdropper can also use the message he/she eavesdropped in the network to adjust the message received by him/her, there must be some uncertainty for the eavesdropper. After enough uncertainty has been accumulated, a universal hash function can then be used by A and B to establish a secret key which is totally unknown to the eavesdropper. Since the security of their protocol depends only on the unpredictability of random noise, the protocol is secure even if the eavesdropper has infinite computing power.

*Corresponding Author: Albert Guan; E-mail: albertguan@math.nsysu.edu.tw
DOI: 10.3966/160792642019122007024

In Guan and Tzeng's protocol, the received string is first divided into blocks of 2 bits, and then the exclusive-or of each two bits are used to verify the equality of the messages received by A and B [3]. In this checking step, the eavesdropper can also learn the exclusive-or of each pair of the bits used in the protocol. They have shown that, even if the eavesdropper learned these information, he/she still has some uncertainty about the original bits. For example, if the exclusive-or of a pair of bits is 0, the value of the original bits can be 00 or 11. The eavesdropper cannot decide which case is correct. The eavesdropper will have 1 bit of uncertainty if he/she learns that the parity bit of his/her corresponding bits are different. Thus, designing a function to preserve the maximum uncertainty relative to the eavesdropper is a very crucial step.

There may be other ways to design a better function to maximize the uncertainty of the eavesdropper. In this paper, we focus on simple functions, such as, the exclusive of a sequence of l bits, $l > 1$. We show that the exclusive-or of a sequence of 3 bits can maximize the uncertainty of the eavesdropper. We show that the *average entropy* for each message bit will actually decreased if the exclusive-or of 4 or more bits are used. Therefore, the length of required random strings is minimized if the exclusive-or of adjacent 3 bits are used. We use this property to design a new protocol for establishing secret keys. In the case that the bit-flip rates are smaller than 0.01, the efficiency of the new protocol is much better than the protocol proposed in [3]. For example, for $p = 0.01$, in average, the original protocol needs 12227 bits to establish 128 bit keys, while the new protocol only needs 7539 bits. The saving is almost 40% for establishing a 128-bit secret key. For longer key length, the saving of random bits become more significant.

2 Related Works

A well-known key agreement protocol is the Diffie-Hellman key exchange protocol [1]. Due to solving discrete logarithm problem is *computationally hard*, the eavesdropper cannot feasibly determine the key computed by the two parties, even if the eavesdropper has learned the messages sent through a public network. Therefore, the security of Diffie-Hellman key exchange protocol is based on solving discrete logarithm problem in a large finite group is hard. However, it can be broken if the attacker has enough computing power, such as by using quantum computers. In 2012, Ding et al. presented a Diffie-Hellman-like protocol, but the security is based on *learning with error problem* [4]. The learning with error problem has been shown to be computationally hard, and currently no quantum algorithms can solve this problem efficiently. There are other key agreement protocols such as [5-7]. In this paper, we will focus on the key agreement protocols whose security is not based on computationally hard problems.

Recently, key agreement protocols which do not depends on computationally hard problems were proposed by many authors. Aumann et al. proposed a message encryption method based on bounded storage model [8]. Tsai et al. proposed a key establishment protocol for wireless sensor network in the bounded storage model [9]. In this model, it is assume that none of the nodes can store all random bits from a common random source. To establish a secret key with high probability, both nodes need to store at least $2(\alpha k)^{1/2}$ bits to ensure that they have at least k bits in common, where k is the security parameter and α is the length of the public random string. Due to limited storage, the eavesdropper cannot store all bits needed to compute the secret key. Unfortunately, the value of α needs to be very large. Thus, the secret key bit rate of their scheme is very small.

Another way to establish a secret key is to store a set of keys in advance. For the pre-deployment of keys, Eschenauer et al. proposed a method to assign a random subset of the keys to each sensor node. They proved that two neighboring sensor nodes can establish a shared key from their own key pools with very high probability if large enough number of keys have been stored in each node in advance [10]. Liu and Ning improved the basic random key pre-distribution scheme of Eschenauer and Gilgor by using multiple random key pools for each sensor node [11]. Ren et al. discussed how to pre-distribute keys in large scale [12]. Miller and Vaidya proposed a key pre-distribution scheme which assumes that the communication channels between sensor nodes use the orthogonal frequency-division multiplexing technology [13]. All of the above protocols are information theoretically secure, but they all need large amount of storage.

Noisy channel has many applications in cryptography. Wyner showed that two parties A and B can exchange a secret key, and eavesdropper E can only obtain a small fraction of the information as long as the binary symmetric channel connected for E is worse than the channel connected for A and B [14]. Crépeau proposed oblivious transfer and bit commitment protocol based on binary symmetric channel [15]. Maurer et al. focus on secret key rate at which Alice and Bob can generate over an insecure, but authenticated channel. The two scenarios they used are binary symmetric channel and erasure channel [16]. Later, Maurer and Wolf provide stronger definitions of secrecy capacity and secret key rate [17].

In this paper, we present a protocol for key agreement by using unpredictable random noise in communication channel. The protocol requires much less random bits as compared to the bounded storage model. Our protocol works even if the quality of the channel used by the eavesdropper is the same as the one used by A and B.

3 Preliminary

The security of our key establishment protocol depends on the unpredictable random noise in communication network. In the application level of a computer communication network, it is generally assumed that a receiver can receive all messages in the communication without errors. This may not be true at the lower level of the network in a practical environment. Noise and other factors can make message communication less reliable. Applying error correction codes or even re-sending some parts of a message is required to make sure that the message received is intact.

Although unpredictable random noise may not be useful in message transmission, it can be used in the key establishment protocol. In 2017, Guan and Tzeng proposed a key establishment protocol by using unpredictable random noise [3]. A brief description of their key establishment protocol can be described as follows. Assume that, in a communication channel, node *A* and node *B* try to establish a secret key. The key establishment protocol consists of the following 3 steps.

(1) Receive random bit strings from a common random source,

(2) Adjust received strings to make them equal, and

(3) Apply hash function to the adjusted string to obtain a common secret key.

Random noise plays an important role in the establishment of a secure key in our protocol. In the first step of our protocol, both *A* and *B* try to receive random bit string from a common source, such as a beacon node. Note that, at the same time, the eavesdropper *E* is also eavesdropping on the communication channel. The main goal of the protocol is to make the eavesdropper has almost no information about the final key *K*. This is possible due to the unpredictability of noise in the communication channel.

Let $x = x_1, x_2, \ldots, x_m$ be the message received by *A*, $y = y_1, y_2, \ldots, y_m$ be the message received by *B*, and $z = z_1, z_2, \ldots, z_m$ be the message received by the eavesdropper. Although these messages were all obtained from the same beacon node, due to noises, the messages *x*, *y*, and *z* are unlikely to be the same.

To analyze the protocol, we assume that the communication channel is a *binary symmetric channel* with bit-flip probability *p*. In practical communication channels, the value of *p* is usually small. In this work, we assume that $0.001 \le p \le 0.015$. Thus the differences between any pair of these strings is relatively small.

In the second step of the protocol, nodes *A* and *B* try to make the strings received by them equal, that is, to make *x* = *y*. After this is done, a secret key *K* can be obtained by applying a hash function *h* randomly chosen from a universal hash family. This is done in the third step. That is, a secret key can be computed by $K = h(x) = h(y)$.

As stated before, due to noises in the communication channel, the message *x* received by *A* and the message *y* received by *B* may not be the same. The second step is to make *x* = *y*. Some information must be sent through the communication channel so that *A* and *B* can make corrections on their strings.

Assume that *A* sends some information based on his message *x*, say $f(x)$, to *B*. For example, in Guan and Tzeng's protocol, $f(x)$ is defined as:

$$\alpha = \alpha_1, \alpha_2, \ldots, \alpha_{m/2},$$

where $\alpha_i = x_{2i-1} \oplus x_{2i}$, the exclusive-or of pair of bits of *x*.

Based on $\alpha = f(x)$ and the message *y* he/she received, *B* sends $g(\alpha, y)$ to *A*. For example, in Guan and Tzeng's protocol, $g(\alpha, y)$ is defined as the subset of the indices *i* such that $x_{2i-1} \oplus x_{2i} \ne y_{2i-1} \oplus y_{2i}$.

According to the information $I = g(\alpha, y)$, for each $i \in I$, *A* resets $x_{2i-1}x_{2i} = 00$ and *B* also resets $y_{2i-1}y_{2i} = 00$. After this step, it can be shown that the string *x* and *y* will be equal with very high probability.

The problem is that the eavesdropper can also learn the message $f(x)$ and $g(f(x), y)$ used by *A* and *B* to make *x* and *y* equal. They showed that the eavesdropper must have some uncertainty about *x* (or *y*), even if he/she knows and $g(\alpha, y)$.

To make the protocol works more efficiently when the bit-flipped rate is small, a better function to maximize the rate to accumulate uncertainty for eavesdropper about the common string shared by *A* and *B* (namely *x* or *y*) is required and it is described in the next section.

4 Uncertainty Preserving Function

In this section, we present a new uncertainty preserving functions $f(x)$ and $g(f(x), y)$ for the key establishment protocol for communication channel with small bit-flip rate, such as $p \le 0.015$. Note that the purpose of these functions is to make the string *x* received by *A* and the string *y* received by *B* equal. More importantly, we would like to maximize the uncertainty of the attacker about the strings *x* and *y*.

In the original protocol proposed by Guan and Tzeng [3], these functions use exclusive-or of a pair of bits to check if *x* and *y* are equal or not. Let the random bit string received by *B* be $y = y_1, y_2, \ldots, y_m$. After receiving $f(x) = \alpha$ from *A*, *B* checks if α_i equals $y_{2i-1} \oplus y_{2i}$ or not, and sends $g(f(x), y) = \{\alpha_i \ne y_{2i-1} \oplus y_{2i}\}$ to *A*.

Since the bit-flip rate is small, if $\alpha_i = y_{2i-1} \oplus y_{2i}$ then $x_{2i-1} \oplus x_{2i} = y_{2i-1} \oplus y_{2i}$ with high probability. If

$x_{2i-1} \oplus x_{2i} \neq y_{2i-1} \oplus y_{2i}$, then A sets $x_{2i-1}x_{2i} = 00$, and B sets $y_{2i-1}y_{2i} = 00$. After these adjustments of their strings, it can be shown that x equals to y with very high probability.

Assume that the random bit string received by the eavesdropper is $z = z_1, z_2, \ldots, z_m$ Suppose that $\alpha_i \neq z_{2i-1} \oplus z_{2i}$, the eavesdropper does not know the values of $x_{2i-1}x_{2i}$ (or $y_{2i-1}y_{2i}$). For example, if $\alpha_i = 0$, it is possible that $x_{2i-1}x_{2i} = 00$ or $x_{2i-1}x_{2i} = 11$. The eavesdropper does not know which one is correct. Therefore, in this case, the eavesdropper has 1 bit of uncertainty about the two bits $x_{2i-1}x_{2i}$.

Let the function $f(x) = \alpha_1, \alpha_2, \ldots, \alpha_{m/2}$ be defined as $\alpha_i = x_{2i-1} \oplus x_{2i}$, $i = 1, 2, \ldots, m/2$. For the eavesdropper, this function generates 1 bit of uncertainty for each pair of bits when these two bits are different from the A's (or B's). In order to compute a key K of length λ, the protocol must accumulate at least λ bits of uncertainty. Thus, it is desirable to design functions which can generate more uncertainty for the eavesdropper to improve the efficiency of the new key agreement protocol, for the case that the bit-flip rate is small.

There may be other ways to design a better function *f* to be used in the protocol. We focus on the exclusive-or of *l* consecutive bits of the message. For *l* = 2, for every 2 bits, the average uncertainty is 1 for the attacker. Assume that the value of *l* is small. For any *l* > 1, if $x_{l(i-1)+1} \oplus x_{l(i-1)+2} \oplus \cdots \oplus x_{li} \neq y_{l(i-1)+1} \oplus y_{l(i-1)+2} \oplus \cdots \oplus y_{li}$, it is reasonable to assume that $z_{l(i-1)+1} \oplus z_{l(i-1)+2} \oplus \cdots \oplus z_{li}$ and $x_{l(i-1)+1} \oplus x_{l(i-1)+2} \oplus \cdots \oplus x_{li}$ differ in only 1 bit. Without any other information, the eavesdropper does not know which bits should be flipped to make the two strings equal. Therefore, the uncertainty for the eavesdropper is log *l* bits. Thus, in average, each of the *l* bits contribute (log *l*)/*l* bits of uncertainty.

Figure 1 shows the function $h(x) = (\log_2 x)/x$ when the domain of the function is the reals **R**. The function *f* has a maximum value at $x = e \approx 2.71828$. Since the function is concave and $h(2) = 0.5 < h(3) \approx 0.52832$, we choose l = 3 in our new key agreement protocol. Thus, let $f(x) = \alpha_1, \alpha_2, \ldots, \alpha_{m/3}$, where $\alpha_i = x_{2i-1} \oplus x_{2i}$, $i = 1, 2, \ldots, m/3$.

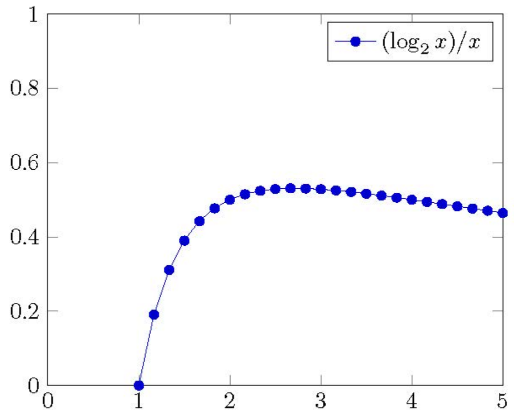

Figure 1. The function $h(x) = (\log_2 x)/x$ has maximum at $x = e \approx 2.71828$.

Note that, in the above analysis, we assume that only 1 bit is flipped in every *l* bits. This may not be true, since other bits may also be flipped with very small probability. However, when the value of *p* is small, the case that only 1 bit is flipped in each block of *l* bit is a good estimation of the uncertainty for the eavesdropper. All the other cases contribute a very small amount to the uncertainty for the attacker about the string *x* (or *y*). Detailed analysis including all the cases will be given in Section 6.

5 Description of the New Protocol

Our protocol is shown in Figure 2 and Figure 3. Figure 2 describes the basic steps of our protocol for collecting random bit strings from a beacon node. These steps are repeatedly executed until enough uncertainty of the eavesdropper about the random strings has been accumulated. Then a secret key can be computed by using the random bit strings. The detailed steps of the main protocol are described in Figure 3.

1. Let m be a multiple of 3. Two nodes A and B receive m random bits broadcast by the beacon node at the same time.
2. Let the m bits that A received be $x_1, x_2, \ldots, x_m$.. and the m bits that B received be $y_1, y_2, \ldots, y_m$.
3. A computes $\alpha = \alpha_1, \alpha_2, \ldots, \alpha_{m/3}$, where $\alpha_i = x_{3i-2} \oplus x_{3i-1} \oplus x_{3i}$, and send α to B.
4. B also computes $\beta = \beta_1, \beta_2, \ldots, \beta_{m/3}$, where $\beta_i = y_{3i-2} \oplus y_{3i-1} \oplus y_{3i}$.
5. B computes the set $I = \{i \mid \alpha_i \neq \beta_i\}$ and sends I to A.
6. B sets $y_{3i-2} = y_{3i-1} = y_{3i} = 0$ for every $k \in I$.
7. After receiving I, for each $k \in I$, A sets $x_{3i-2} = x_{3i-1} = x_{3i} = 0$.

Figure 2. Basic steps of the protocol

1. Repeatedly run the basic steps of the protocol *r* times to obtain a message of length *mr*.
2. *A* computes $T = h_c(x_1, x_2, \ldots, x_{mr})$, and it to *B*.
3. *B* computes $T' = h_c(y_1, y_2, \ldots, y_{mr})$ and send it to *A*.
4. If $T = T'$, then $x_1, x_2, \ldots, x_{mr} = y_1, y_2, \ldots, y_{mr}$ with very high probability. Both *A* and *B* set the secret key $K = (x_1, x_2, \ldots, x_{mr}) = h_u(y_1, y_2, \ldots, y_{mr})$
5. Otherwise, $T \neq T'$, the protocol aborts.

Figure 3. Description of the protocol

The main idea of our protocol is to establish a common string of length *mr* for both nodes *A* and *B*. This is done by repeatedly executing the basic steps of the protocol *r* times. The value of *r*, as well as other parameters, will be discussed in the Section 7.

Note that, in the basic steps of our protocol, assume that $\alpha_1, \alpha_2, \ldots, \alpha_{m/3} = \beta_1, \beta_2, \ldots, \beta_{m/3}$, we conclude that the two sequences $x_1, x_2, \ldots, x_m$ and $y_1, y_2, \ldots, y_m$ are equal with very high probability. Although there is a very small probability that the two strings are different, for security reason, the protocol does not check if they are equal or not in the basic steps.

The checking will be done after *A* and *B* have collected strings of length *mr*, in step (2) and step (3) of the main protocol by using a cryptographic hash function h_c. The hash function $h_c : 2^{mr} \to 2^{\delta}$. is used to check that the adjusted messages of *A* and *B* is equal or not. Usually, the value of $\delta = 80$. The larger the value of δ the higher probability that $x = y$. However, the amount of information the eavesdropper learns will also be increased. If they are equal, then *A* and *B* can apply the hash function h_u to extract secret key.

At the final step of the protocol, a universal hash function h_u is required. Let *s* be the length of the key to be established. Any universal hash function from $\boldsymbol{Z}_2^{mr}$ to $\boldsymbol{Z}_2^{s}$ can be used, provided that hu is chosen randomly from universal hash function family. Note that h_u can be announced in public. The secrecy of the final key does not depend on the secrecy of h_u. It depends on the secrecy of the string *x* and *y* agreed by *A* and *B*.

6 Analysis of the New Protocol

In this section, we first analyze the successful rate of our key establishment protocol. We then analyze the uncertainty of the eavesdropper about the strings agreed by *A* and *B*.

Define *rounds* to be the number of times the full protocol is executed, and *runs* the number of times the basic steps shown in Figure 2 are executed in a round. We first estimate the average number of rounds *R* needed to be executed for the two nodes to obtain a common secret key.

We first show that the successful rate of the proposed protocol is high. Assume that the basic steps of the protocol are executed *r* times in each round. Note that the proposed protocol may fail if $x \neq y$ after adjustments. Suppose that nodes *A* and *B* repeatedly execute the protocol until they successfully establishe a common string of length *mr*. The following theorem can be used to estimate the number of rounds *R* for the nodes to establish a secret key *K*.

We first show that the successful rate of the proposed protocol is high. Assume that the basic steps of the protocol are executed *r* times in each round. Note that the proposed protocol may fail if $x \neq y$ after adjustments. Suppose that nodes *A* and *B* repeatedly execute the protocol until they successfully establishe a common string of length *mr*. The following theorem can be used to estimate the number of rounds *R* for the nodes to establish a secret key *K*.

Theorem 1. *Let G be the random variable for the number of rounds executed until nodes A and B successfully establish a common string of length mr in the protocol. The expected value R = E(G) of the random variable G is*

$$1 - 3p'^2(1-p')^{mr/3} \text{ where } p' = 2p(1-p).$$

Proof. Since the bit-flip probability is *p*, the probability that the two bits x_i and y_i, *i* = 1, 2, ..., *m* of *A* and *B* are different is $p' = 2p(1-p)$. The only case that $\alpha_i \neq \beta_i$ but $x_{3i-2}x_{3i-1}x_{3i}$, and $y_{3i-2}y_{3i-1}y_{3i}$ are different is that 2 of the 3 bits are flipped. The probability that these two of the 3 bits are flipped is $3p'^2(1-p)$.. Since there are *m* bits at each run, the probability that the *mr* bits that *A* received are the same with *B* is $1 - 3p'^2(1-p')^{\frac{mr}{3}}$. Let $p'' = 1 - 3p'^2(1-p')^{\frac{mr}{3}}$, $\Pr[G=t] = p''(1-p'')^{t-1}$. Hence, $E(G) = \sum_{t=1}^{\infty} tp''(1-p'')^{t-1}$ $= p''\sum_{t=1}^{\infty} t(1-p'')^{t-1} = 1/p''$.

Table 1, shows the expected number of times the proposed protocol needs to be executed. The values in Table 1 can be obtained from Theorem 1 for

$$\frac{1}{R} = p'' = 1 - 3p'^2(1-p')^{\frac{mr}{3}}$$

Table 1. The average number of rounds required for $m = 321$

	$r = 10$	$r = 20$	$r = 30$
$p = 0.001$	1.01	1.03	1.04
$p = 0.005$	1.37	1.88	2.57
$p = 0.010$	3.44	11.80	40.58
$p = 0.015$	15.24	232.11	3536.34

where R is the expected number of rounds until the protocol succeeds. This number depends on the common string length m for each run, the number of runs r and the bit-flip probability p. In this example, we assume that $m = 321$.

The values shown in Table 1 show that our protocol is practical when $p < 0.015$, except the case that the number of runs $r = 30$ and bit-flip probability $p = 0.015$, in which the expected number of rounds until the protocol succeeds is 3536, which is a bit large.

The next theorem states that when the eavesdropper has his/her own parity bits γ_k and node A's parity bits α_k, $1 \le k \le n/3$, the amount of uncertainty about the corresponding 3 bits $x_{3i-2}x_{3i-1}x_{3i}$. Finally, we divide this value by 3 to get the secret key rate.

Theorem 2. *Let* $S = \{0, 1\}^3$, $X \in S$, *and* X *is a random variable for 3 bits corresponding to the parity bit* α_k. *Then the eavesdropper has uncertainty*

$$-(3f_1(p)\log f_1(p)) + f_2(p)\log(f_2(p)))$$
$$-(3f_1(q)\log f_1(q))$$

for X, where $f_1(p) = (1-p)^2/p^2 + 3(1-p)^2$, $q = 1-p$, $f_2(p) = 1-3f(p)$, $\alpha = p(p^2+3(1-p)^2)$, $\beta = (1-p)(1-p^2+3p^2)$.

Proof. Let Y be the random variable such that $Y = 0$ if eavesdropper's parity bit k is not equal to A's parity bit α_k, $Y = 1$ if eavesdropper's parity bit is equal to α_k, $Y = \perp$, if A and B have different parity bit.

$\mathrm{H}(X|Y) = \Pr[Y = 1]\mathrm{H}(X|Y = 1) + \Pr[Y = 0]\mathrm{H}(X|Y = 0) + \Pr[Y = \perp]\mathrm{H}(X|Y = \perp)$.

Since eavesdropper knows the 3 bits when two nodes have different parity bit, that is, 000. This implies that

$\mathrm{H}(X|Y = \perp) = 0$, and
$\mathrm{H}(X|Y) = \Pr[Y = 1]\mathrm{H}(X|Y = 1) + \Pr[Y = 0]\mathrm{H}(X|Y = 0)$.

$\Pr[Y = 1] = (1-p)((1-p)^2 + 3p^2)$, and
$\Pr[Y = 0] = p(p^2 + 3(1-p)^2)$.

Without loss of generality, we assume that the 3 bits the eavesdropper received is $X' = 000$. Then

$\Pr[X = 000 \mid Y = 1] = (1-p)^3/((1-p)^3+3p^2(1-p)) = (1-p)^2/((1-p)^2+3p^2)$, and
$\Pr[X = 110 \mid Y = 1] = p^2(1-p)/(1-p)^3 + 3p^2(1-p)) = p^2/((1-p)^2 + 3p^2)$.

Similarly, we can derive

$\Pr[X = 101 \mid Y = 1] = p^2/((1-p)^2 + 3p^2)$, and
$\Pr[X = 011 \mid Y = 1] = p^2/((1-p)^2 + 3p^2)$.

For the case $Y = 0$,

$\Pr[X = 111 \mid Y = 0] = p^3/(p^3 + 3p(1-p)^2)$
$= p^2/(p^2 + 3(1-p)^2)$, and
$\Pr[X = 100 \mid Y = 0] = p(1-p)^2/(p^3 + 3p(1-p)^2)$
$= (1-p)^2/(p^2 + 3(1-p)^2)$.

Similarly, we can derive

$\Pr[X = 010 \mid Y = 0] = (1-p)^2/(p^2 + 3(1-p)^2)$, and
$\Pr[X = 001 \mid Y = 0] = (1-p)^2/(p^2 + 3(1-p)^2)$.

$\mathrm{H}(X \mid Y = 1) = 3f_1(q)\log(f_1(q)) + f_2(q)\log(f_2(q))$, and
$\mathrm{H}(X \mid Y = 0) = 3f_1(p)\log(f_1(p)) + f_2(p)\log(f_2(p))$.

This implies that

$$\mathrm{H}(X \mid Y) =$$
$$-\alpha(3f_1(p)\log f_1(p)) + f_2(p)\log(f_2(p)))$$
$$-\beta(3f_1(q)\log f_1(q)) + f_2(q)\log(f_2(p))).$$

7 Selecting Parameters of the Protocol

In this section we show how to select suitable parameters in our protocol.

First of all, for piratical applications, such as establishing keys to be used in AES, the length of the key s should be 128, 192, or 256. Thus, the amount of uncertainty for the eavesdropper about the common string x and y shared by A and B should be no less than $s + \delta$, where δ is the amount of information revealed for checking if $x = y$ or not in the main protocol.

The purpose of repeated execution of the basic step of the protocol is to accumulate sufficient uncertainty bits against eavesdropper. The value of r is set so that the mr common bit strings of A and B will have at least s uncertain bits to eavesdropper. The possible values of r is given in Table 2 for some practical values of p = 0.005, 0.01, 0.015.

Table 2. The number of uncertainty bits for eavesdropper, $m = 321$

	$r = 10$	$r = 20$	$r = 30$
$p = 0.005$	51	77	103
$p = 0.010$	109	164	218
$p = 0.015$	154	231	308

The values in Table 2 can be derived from Theorem 2. First we compute the uncertainty for eavesdropper for a single run, then multiply it by the number of runs r to get the final result.

For example, for $r = 30$ and bit-flip probability p = 0.01, after executing the protocol, the common string will have 164 bits of uncertainty for eavesdropper. In general, for our protocol to run successfully for $s = 160$, it is always possible to select the parameters for m =

321, and $0.001 \le p \le 0.015$. This shows that our protocol is practical in current technology, For the case that the error rate is very small, i. e., $p < 0.001$, the two nodes can randomly flip more bits to make the same effect as $p > 0.001$.

Let s be the security parameter. In the next theorem, with proper choosing of parameters, we show that the key established by our protocol is s-bit secure.

Theorem 3. *Assume that h_c is the cryptographic hash function and h_u is the universal hash function chosen randomly from universal hash function family. Then, at the end of the protocol, the eavesdropper has at least s bits of uncertainty about the value of the key K.*

Proof. Let W be a random variable uniformly distributed from $\{0,1\}^n$. Note that W can be viewed as the string broadcast by the beacon node. Let $BSp(W)$ be another random variable that represents the string received by the receiver when W is sent through a binary symmetric channel with bit flip probability p. It is clear that $BSp(W)$ can be viewed as the string received by A or B.

Let G be a random variable for universal hash function $h_u : \{0,1\}^n \to \{0,1\}^s$, $s < n$. It can be shown that [18].

Theorem 4 (Bennett et al.) *Let $H_b(p)$ be the binary entropy function and $H(.|.)$ be the conditional Shannon entropy. For any $\delta > 0$ and for all sufficiently large n, for $t = n(H_b(p) - \delta) - s, H(G(W) \mid BSp(W), G) \ge s - 2^{-t}/\ln 2$.*

Because $H(G(W) \mid BSp(W), G) = s$ means that given $BSp(W)$, G, the dishonest player has no information about the hash value $G(W)$. The last term is exponentially close to 0. This implies that almost no information about the hash value $G(W)$ is leaked. We cannot use the theorem directly to prove the security for nodes A and B, because we have sent the parity bits in the protocol. However, we can apply this theorem with some modifications.

Recall that $q = 1 - p$, $\alpha = p(p^2 + 3(1-p)^2)$, $\beta = (1-p)((1-p^2) + 3p^2)$, and

$$h = (\frac{1}{3})(-\alpha(3f(p)\log(f(p)) + g(p)\log(g(p))) \\ - \beta(3f(q)\log(f(q))) + g(q)\log(g(q))))$$

Then for any $\in > 0$ and for all sufficiently large $n = mr$, for $t = n(h - \frac{\delta}{n} - \varepsilon) - s$, $H(G(W) \mid \alpha, \beta, BSq(W), G)$ $\ge s - 2^{-t}/\ln 2 \approx s$.

Note that our hash function is from mr bits to s bits.

8 Conclusions

We have proposed a new key establishment protocol for two nodes in a communication network to establish a secret key which is information-theoretically secure. In this new protocol, we use the exclusive-or of a block of 3 bits to verify the consistency of the strings received by A and B. We also showed that using a blocks of size 3 is an optimal choice. Not only the successful rate of the protocol would be decreased, the uncertainty of the eavesdropper about the common bit string is also deceased. The proposed protocol is a light-weight protocol. It only needs to do exclusive-or and hash operations. Thus it is suitable to devices with limited computing resources.

Acknowledgments

The research of the first author is supported in part by the MOST project 107-2218-E-110-017-MY3. The research of the second author is supported in part by the MOST project 106-2811-E-002-085.

References

[1] W. Diffie, M. Hellman, New Directions in Cryptography, *IEEE Transactions on Information Theory*, Vol. 22, No. 6, pp. 644-654, November, 1976.

[2] P. W. Shor, Polynomial-time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer, *SIAM Journal on Computing*, Vol. 26, No. 5, pp. 1484-1509, October, 1997.

[3] A. Guan, W.-G. Tzeng, A Secret Key Establishment Protocol for Wireless Networks Using Noisy Channels, *Journal of Computer Security*, Vol. 25, No. 2, pp. 139-151, May, 2017.

[4] J. Ding, X. Xie, X. Lin, *A Simple Provably Secure Key Exchange Scheme Based on the Learning with Errors Problem*, Cryptology ePrint Archive, Report 2012/688, December, 2012.

[5] C.-Y. Chen, H.-C. Chao, A Survey of Key Distribution in Wireless Sensor Networks, *Security and Communication Networks*, Vol. 7, No. 12, pp. 2495-2508, December, 2014.

[6] H. F. Rashvand, H. C. Chao, *Dynamic Ad-Hoc Networks*, The Institution of Engineering and Technology, 2013.

[7] F.-H. Tseng, T.-T. Liang, L.-D. Chou, H.-C. Chao, On-line Evaluation System for Examining Website Content Consistency between IPv4 and IPv6, *TENCON 2014 - 2014 IEEE Region 10 Conference*, Bangkok, Thailand, 2014, pp.1-5.

[8] Y. Aumann, Y. Z. Ding, M. O. Rabin, Everlasting Security in the Bounded Storage Model, *IEEE Transactions on Information Theory*, Vol. 48, No. 6, pp. 1668-1680, June, 2002.

[9] S.-C. Tsai, W.-G. Tzeng, K.-Y. Zhou, Key establishment Schemes Against Storage-bounded Adversaries in Wireless Sensor Networks, *IEEE Transactions on Wireless Communications*, Vol. 8, No. 3, pp. 1218-1222, March, 2009.

[10] L. Eschenauer, V. D. Gligor, A Key-management Scheme for Distributed Sensor Networks, *Proceedings of the 9th ACM Conference on Computer and Communications Security*, CCS '02, Washington, DC, USA, 2002, pp. 41-47.

[11] D. Liu, P. Ning, Establishing Pairwise Keys in Distributed Sensor Networks, *Proceedings of the 10th ACM Conference on Computer and Communications Security*, CCS'03, Washington, DC, USA, 2003, pp. 52-61.

[12] K. Ren, K. Zeng, W. Lou, A New Approach for Random Key Pre-distribution in Large-scale Wireless Sensor Networks: Research Articles, *Wireless Communications Mobile Computing*, Vol. 6, No. 3, pp. 307-318, May, 2006.

[13] M. J. Miller, N. H. Vaidya, Leveraging Channel Diversity for Key Establishment in Wireless Sensor Networks, *Proceedings INFOCOM 2006. 25th IEEE International Conference on Computer Communications*, Barcelona, Spain, 2006, pp. 1-12.

[14] A. D. Wyner, The wire-tap channel, *Bell System Technical Journal*, Vol. 54, No. 8, pp. 1355-1387, October, 1975.

[15] C. Crépeau, Efficient Cryptographic Protocols Based on Noisy Channels, *Proceedings of the 16th Annual International Conference on the Theory and Application of Cryptographic Techniques*, EUROCRYPT'97, Konstanz, Germany, 1997, pp. 306-317.

[16] U. M. Maurer, S. Wolf, Unconditionally Secure Key Agreement and the Intrinsic Conditional Information, *IEEE Transactions on Information Theory*, Vol. 45, No. 2, pp. 499-514, March, 1999.

[17] U. Maurer, S. Wolf, Information-theoretic Key Agreement: From Weak to Strong Secrecy for Free, *Advances in Cryptology- EUROCRYPT 2000*, Bruges, Belgium, 2000, pp. 351-368.

[18] C. H. Bennett, G. Brassard, C. Crepeau, U. M. Maurer, Generalized Privacy Amplification, *IEEE Transactions on Information Theory*, Vol. 41, No. 6, pp. 1915-1923, November, 1995.

Biographies

Albert Guan received a bachelor's degree in Applied Mathematics from National Sun Yat-Sen University, in 2008, and a Ph.D. degree in Computer Science from National Chiao Tung University in 2017. From 2017 to 2018, he was a postdoctoral research fellow in the Department of Electronic Engineering at National Taiwan University. In 2018, he joined the Department of Applied Mathematics at National Sun Yat-Sen University as an assistant professor. His research interests include discrete mathematics, cryptography and its applications.

Chin-Laung Lei received the B.S. degree in electrical engineering from National Taiwan University, Taipei, in 1980 and the Ph.D. degree in computer science from the University of Texas at Austin in 1986. From 1986 to 1988, he was an assistant professor in the Computer and Information Science Department, Ohio State University, Columbus. In 1988, he joined the faculty of the Department of Electrical Engineering, National Taiwan University, where he became a professor in 1996. He is a cowinner of the first IEEE LICS test-of-time award, and has published more than 250 technical articles in scientific journals and conference proceedings. His current research interests include network security, cloud computing, and big data analytics.