Information Security for Sensors by Overwhelming Random Sequences and Permutations

by

Shlomi Dolev, Niv Gilboa, Marina Kopeetsky, G. Persiano, P. G. Spirakis

Technical Report #10-06

August 2010

Information Security for Sensors by Overwhelming Random Sequences and Permutations *

(Preliminary Version)

Shlomi Dolev
Department of Computer Science
Ben-Gurion University
Beer-Sheva, 84105, Israel
dolev@cs.bgu.ac.il

Niv Gilboa
Department of Computer Science
Ben-Gurion University
Beer-Sheva, 84105, Israel
niv.gilboa@gmail.com

Marina Kopeetsky
Department of Software Engineering
Sami-Shamoon College of Engineering
Beer-Sheva, 84100, Israel
marinako@sce.ac.il

Giuseppe Persiano
Dipartimento di Informatica ed Applicazioni
Università di Salerno
Via Ponte Don Melillo Salerno
84084 Campania Italy
giuper@dia.unisa.it

Paul G. Spirakis
Department of Computer Engineering and Informatics
University of Patras and Research Academic Computer Technology Institute
N. Kazantzakis str., University Campus, 265 00 Rio, Patras, Greece
spirakis@cti.gr

ABSTRACT

We propose efficient schemes for information-theoretically secure key exchange in the Bounded Storage Model (BSM), where the adversary is assumed to have limited storage. Our schemes generate a secret One Time Pad (OTP) shared by the sender and the receiver, from a large number of public random bits produced by the sender or by an external source. Our schemes initially generate a small number of shared secret bits, using known techniques. We introduce a new method to expand a small number of shared bits to a much longer, shared key.

Our schemes are tailored to the requirements of sensor nodes and wireless networks. They are simple, efficient to implement and take advantage of the fact that practical wireless protocols transmit data in frames, unlike previous protocols, which assume access to specific bits in a stream of data.

Indeed, our main contribution is twofold. On the one hand, we construct schemes that are attractive in terms of simplicity, computational complexity, number of bits read from the shared random source and expansion factor of the initial key to the final shared key. On the other hand, we show how to transform any existing scheme for key exchange in BSM into a more efficient scheme in the number of bits it reads from the shared source, given that the source is transmitted in frames.

1. INTRODUCTION

*Partially supported by the ICT Programme of the European Union under contract number FP7-215270 (FRONTS), Microsoft, NSF, Deutsche Telekom, Rita Altura Trust Chair in Computer Sciences, Lynne and William Frankel Center for Computer Sciences, and the internal research program of Sami Shamoon College.

State of the Art. A major building block in security and cryptography is generation of a secret that two parties share. The secret may then be used as a symmetric encryption or authentication key.

We propose a scheme to generate a shared key in the Bounded Storage Model (BSM). The Bounded Storage Model was presented in Maurer's work [9]. This model investigates cryptographic tasks such as encryption and authentication in the presence of an adversary that has bounded storage capacity. While most of modern cryptography limits an adversary's resources, the usual approach is to place a bound on the adversary's time complexity. Given various unproven assumptions on the hardness of computational tasks, modern cryptography has many beautiful constructions of schemes that are secure against an adversary that has limited time complexity.

In the Bounded Storage Model, on the other hand, there is no need for computational assumptions. Given a source of random bits that broadcasts more traffic than the adversary can store, legitimate parties can perform cryptographic tasks in a way that is information-theoretically secure. This is true even if the storage of the legitimate parties is smaller than that of the adversary.

One of the main cryptographic tasks is for two parties to share a key, without leaking any of its bits to an adversary that monitors traffic. [9] showed that a key can be shared even when the two parties do not share any bits before the protocol begins. This work was improved by [3], and this second work was analyzed in [5] and shown to be essentially optimal in terms of the amount of data the two parties can share, given the ratio between the storage capacity of the adversary and the storage capacity of the two legitimate parties.

Subsequent works [1], [2], [4], [8] and [11] showed schemes to expand a small initial key to a much larger key that can be used as a One Time Pad (OTP). Both the initial key and the OTP are shared by the legitimate parties, but are unknown to the adversary. It is as-

sumed that the adversary has no information on the initial key with probability 1, while the probability that it has some information on the one-time pad is less than some parameter ϵ.

Our contribution. We propose a pair of two-stage scheme that first use the process for initial key generation of [3] to generate a short, shared key. Our schemes then employ a novel method for expanding a short initial key into a longer key. Our scheme has the basic property of key exchange schemes that passive attackers, who only monitor traffic, do not obtain information on the shared key, while active attackers may mount Man-in-the-Middle attacks. We may assume that such attacks are foiled by identification performed in the physical layer to distinguish between non corrupted and corrupted nodes. We note that authentication of a wireless node for which a shared secret should be established may be based on physical identification (e.g., [12]).

The basic step of our schemes is to use the initial key for both the sender and the receiver to select several blocks of bits from the shared random source. After all the random bits have been transmitted, the sender chooses a random permutation on all the stored bits and exchanges it with the receiver. After permuting the bits, both parties exclusive-or all the bits in a contiguous block of bits, thus obtaining a single bit of the OTP. Given enough such blocks, they construct the whole OTP.

We present two protocols the *Permutation Revealing Protocol* PRP and *Permutation Encrypted Protocol* PEP. The permutation in PRP is sent as clear text, deriving a single OTP from a shared random string of length n. In order to obtain another OTP, the two parties must exchange a new permutation. In PEP the permutation it is kept secret forever. Thus, PEP may be used with the same permutation to derive an exponential number of One Time Pads. However, to generate a new OTP the parties must share a new random string of length n bits.

We use the following notation: k denotes the security parameter which means that all schemes are information-theoretically secure with probability at least $1-\epsilon = 1-2^{-k}$. The length of the random string is denoted by n and the length of the OTP is denoted by m.

We view the random string as a matrix, where the number of columns is $m(k+\log m)$ and the number of rows is denoted by b and is equal to $n/m(k+\log m)$. We refer to the parameter b as the number of channels. A physical implementation of the random source may allow transmission in parallel over b channels in our protocol. If the implementation does not allow such parallel transmission, the b channels just define sections of size $m(k+\log m)$ bits within the n-bit random string.

We use the fact that wireless protocols transmit data in frames of several bits together for various reasons such as efficiency and error correction. The transmission of a shared random string requires just such a wireless protocol and we denote the frame length of this protocol by α bits.

The complexity of PRP under various measures is as follows. The computational complexity is $m(k+\log m)$. The number of bits read from the random source is $\lceil \frac{m}{\alpha} \rceil \alpha(\log m + k)$. The expansion factor, which is defined the ration between the initial secret (the product of the first stage of the protocol) and the OTP length m is $\frac{m}{\log b(\log m+k)}$. The storage required for the second stage of PRP is $O(m(k+\log m)$. The storage required for the first stage is identical to the storage of [3]. The main idea of that protocol is that the sender and the receiver can each choose a small set of locations from the random string and then store the bits in these locations. After the transmission of the random string ends, they exchange their chosen locations. Each shared location is associated with a shared bit. By the birthday paradox, the storage requirement is proportional to $\sqrt{n}$. We note that this storage can be reduced as much as needed (within logarithmic factors) if the random string is much larger than n. Or, in other words, if the two parties receive an n-bit, random string multiple times.

A second contribution is to transform a key exchange scheme that accesses distinct bits in the random strings into a scheme that accesses blocks of bits (where each block is identified with a frame of the wireless protocol). We can thus reduce the number of bits that a scheme reads. Applied to Vadhan's scheme [11], which is reads the least number of bits of all known schemes, we obtain scheme that reads $k+\log m$ bits (compared to $k+\log n$).

Comparison with Previous Work. In all of the works that expand an initial shared key to a longer shared OTP, ([1], [2], [5], [8] and [11]), the main measure of a scheme's efficiency is its expansion factor. That is, the ratio between the length of the one time pad and the length of the initial key. By setting the one time pad to always be of length m, the best scheme is the one with the shortest initial key.

Table 1 compares the expansion factor of previous schemes and our own. As a comment to Table 1, we notice that [4] requires fewer

Paper	***Length of initial secret***
Ding-Rabin [2]	$k \log n$
Dziembowski-Maurer [4]	$k \log n$
Lu [8]	$\frac{(k+\log n)^2}{\log n}$
Vadhan [11]	$k+\log n$
Our work	$\log b(\log m + k)$

Table 1: Comparing expansion factors

random bits than [2]; [8] requires $m \leq n^\gamma$ for some $\gamma \in (0,1)$, and [11] requires $k < n/2^{\log^* n}$.

Our work has a better expansion factor than [2] and [4] when

$$k \geq \frac{\log m \log b}{\log \frac{n}{b}}.$$

This is always true when $k \geq \log b$.

Our work is better than [8] when

$$k \geq \log b \log n.$$

Our scheme has a better expansion factor than [11] only for specific choices of parameters. If $k > n/2^{\log^* n}$, then the best scheme of [11] is not applicable. Additionally, if b is very small, e.g. $b=2$, then our scheme is better than [11] by a constant factor.

A somewhat theoretical measure to compare these schemes is an upper bound on m. An optimal upper bound is $m \leq n-k$, extracting almost all the random bits in the shared random string. [11] comes within a constant multiplicative factor of this bound. Our solution is slightly worse, since for a minimal b, $b=2$ we have $m \leq n/2(k+\log m)$. We note that typically $m << n$ and this bound is not reached.

Lu [8] and Vadhan [11] showed that all the above schemes fit into a unified "sample then extract" approach. The idea is to sample a small number (t) of bits from the n-bit random source so that, informally speaking, the small sample has almost the same random properties as the large, public string. Then, an extractor is applied to the t bits yielding m output bits for the one-time pad. An extractor is a function, that given a short random string (the shared, initial key) and a larger string, which may not be completely random (the t bits in the sample) outputs an m bit string which is statistically close to being uniformly random.

Various samplers and extractors can be plugged into the overall framework of [11]. The best expansion factor is reached by using a sampler that is based on a random walk on an expander graph and by using the extractor of [13]. Both of these, sampler and extractor, have relatively high computational complexity. Thus, the best scheme of [11] may not be as appropriate for constrained devices as our very simple scheme.

Another measure of the efficiency of such schemes is the number of bits that each party must read from the random source. Wireless traffic is sent in frames, in just about any wireless communication protocol. Denote a frame length by α. Our protocol is the only one that utilizes this property by sampling data in blocks of m bits, while all the previous protocols sample distinct bits. The following table compares the number of bits that each scheme reads from the random source.

Paper	***Number of bits read***
Ding-Rabin [2]	$mk\alpha$
Dziembowski-Maurer [5]	$mk\alpha$
Lu [8]	$mk\alpha$
Vadhan [11]	$(k+\log n)\alpha$
Our work	$\lceil \frac{m}{\alpha} \rceil \alpha(\log m + k)$

Table 2: Comparing number of bits read from random source

As α grows, our scheme becomes more efficient. Specifically, when $\lceil \frac{m}{\alpha} \rceil \leq \frac{\log n+k}{\log m+k}$, our scheme reads less bits than any other scheme.

Our scheme takes advantage of physical implementations in another way, which is not taken into account by previous schemes. Consider a shared random source that is actually transmitted over many physical channels in parallel. Previous schemes regard the whole source as a single string and potentially access any (small) set of bits. Such schemes may require a receiver to tune to more than one channel at once or to change channels faster than the physical equipment is capable of. In contrast, our solution is tailored for standard equipment: the receiver tunes to a channel, receives a block of contiguous bits and then switches to another channel.

Our schemes resemble the protocols of Aumann, Ding and Rabin [1] and [2]. These works introduce two protocols, *Protocol 1* and *Protocol 2*. Our PRP works in the same setting as *Protocol 1* and PEP works in the same setting as *Protocol 2*. Like these two protocols, our schemes do not perform any computationally expensive preprocessing. It is proven in [2] that the initially shared key can be used and reused for an exponential number (in the length of the initial shared key) of rounds, where in each round another portion of the shared random string is produced.

Paper organization. The structure of the paper is as follows. We present the setting and introduce notation in Section 2. The Permutation Revealing Protocol PRP is presented in Section 3. The Permutation Encrypted Protocol PEP and its improved version are described in Section 4. The improving of the key exchange algorithms is discussed in Section 5. Conclusions appear in Section 6.

2. SETTING AND NOTATION

Consider a *wireless network* WN which consists of several nodes. A sender, S wishes to send information securely to a receiver, R. S intends to encrypt its message in blocks of m bits. Each block is encrypted by a one-time pad of length m bits. S and R perform a key exchange scheme to share an m-bit one-time pad prior to sending an encrypted block.

We assume a bounded storage model in which all wireless nodes have the same storage capacity sp, while an adversary has capacity s_{Ad} such that possibly $s_{Ad} > sp$. S shares m bits with R by generating and sending T random bits, $sp < s_{Ad} < T$.

The T bits are sent over b channels, which may have different physical implementations such as different frequencies or different time slots on the same frequency. We denote the channels by $c_1, c_2, \ \ldots, c_b$.

The sender node S and the receiver node R simultaneously run two independent processes in order to generate the shared OTP.

Process 1 runs in the background continuously; its purpose is to generate (a small number of) shared random bits by using the scheme of [5] as follows. S transmits to R a random string α of length T bits. In order to generate one secret shared bit, S and R randomly record $O(\sqrt{T})$ bits and their indexes, using $O(\sqrt{T}\log)T$ bits of memory. Then, S and R send each other the indexes of the stored bits, without revealing the actual values of these bits. Due to the Birthday paradox [5], with high probability there is at least one shared index for S and R. Assuming that T is significantly larger than s_{Ad}, there is high probability that the adversary does not know this shared bit. Standard techniques may use repetitions to make the probability that the bit is unknown to the adversary as close to 1 as necessary.

The shared secret bits produced by *Process 1* are expensive it terms of the time (and number of non-shared random bits produced by the sender) needed to produce a secret shared bit. *Process 2* expands this computationally expensive random string and derives a much longer OTP.

We define the expansion factor χ as a relation between the length of the obtained OTP and the length of the initial shared key s used for generating the OTP.

The adversary we consider is passive, in the sense that it only monitors the data between S and R, without actively taking part in the communication. Informally, we say that an adversary *breaks* a key exchange scheme if it succeeds in correctly recovering a single bit of the shared key. However, an adversary can always guess a single bit in a key with probability $1/2$ by flipping a coin. Thus, breaking a scheme means that an adversary can recover a bit with significantly greater probability than $1/2$.

We parameterize the advantage that an adversary has over coin flipping by a security parameter k, $k \in \mathbb{N}$. We can now formalize the notion of a secure key exchange protocol in our setting.

DEFINITION 1. *A key exchange scheme for two parties in the Bounded Storage Model is a two-party protocol that accepts as input three parameters: s_{Ad}, a bound on the storage size of the adversary, m, the length of the shared key and k, a security parameter. The scheme's output is a shared key of length m bits. The scheme is information-theoretically secure if a computationally unbounded adversary can not obtain any bit of the shared key with probability greater than*

$$\frac{1}{2} + \frac{1}{2^k}.$$

3. PERMUTATION REVEALING PROTOCOL

This section describes the Permutation Revealing Protocol (PRP).

The input of PRP is a set of channels, $c_1, \ldots, c_b$, a security parameter k and the required length m of a shared OTP, which is the output of PRP. As previously stated, PRP has two processes. The first process (lines 6-10) begins without any shared random bits and generates a small shared secret for R and S. This shared secret, of length $\log b(\log m + k)$ is regarded as $\log m + k$ indexes. Each index determines one of the b channels.

Process 2 is performed in two phases. During the first phase

(lines 13-20), S sends to R a large number of random bits. S and R use the small key they share to determine which of the bits that S sends in the first phase must be received and stored. The product of the first phase is a large number of shared bits for S and R. In the second phase (lines 22-36), S and R combine subsets of their shared bits to derive an m-bit shared key. The main point of PRP is that the adversary does not have enough space to store all the random bits of phase 1. The combination of bits in phase 2 make it very likely that for every bit in the OTP, the adversary misses at least one of the bits that generate it.

Diving into the details we note that S transmits random data over b different channels in *Process 2*, phase 1. Let $\lambda = \log m + k$. The data in each channel is organized in λ blocks of m bits each. Thus for every $j = 1, \ldots, \lambda$ there are b different blocks of m bits (one on each channel). The shared key s defines the correct channel for block j. This channel, number s_j, is the only channel that R intercepts in lines 15-16.

In phase 1, S sends $bm\lambda$ bits. Both S and R store only $m\lambda$ bits, denoted in the algorithm by $R_{1,s_1}, \ldots, R_{\lambda,s_\lambda}$.

In the second phase of PRP, S sends to R in clear text (possibly monitored by the adversary), a random permutation π. This permutation defines a reordering of the bits of the concatenated shared string $R_{1,s_1} \| \ldots \| R_{\lambda,s_\lambda}$ (protocol for S, lines 23-25). In order to determine π, S has to generate and send $m\lambda \log(m\lambda)$ random bits. Here $m\lambda$ bits is the number of bits in the concatenated shared string, and $\log(m\lambda)$ is the number of bits needed to encode an index in this concatenated string.

Upon reception of π, R permutes the $m\lambda$ random bits received during the first phase, and generates a matrix P of λ rows and m columns (protocol for R, lines 23-29). The $i-th$ bit of OTPis computed as an exclusive-or of all bits of the $i-th$ column (lines 31-36).

The following theorem proves that PRP is an information- theoretically secure key exchange scheme in the Bounded Storage Model. We prove the result for a limited adversary. However, we conjecture that the protocol is secure for any adversary with bounded storage.

THEOREM 1. *Assume, that an adversary Ad is computationally unbounded, but its storage is limited to s_{ad} bits, $s_{ad} < L/2$, where $L = min(T, bm\lambda)$. Assume further that Ad is limited to storing bits of the random storing and does not store a function of these bits. Then,* PRP *outputs an* OTP*of m bits that S and R share, and the probability that the adversary determines even a single bit of the* OTP *correctly is less than* $1/2 + 2^{-k}$.

PROOF. Since the adversary can store no more than $L/2$ bits, it can store at most half of the $bm\lambda$ bits that S transmits during the execution of PRP.

We set an index i, $1 \leq i \leq m$ and bound the probability that the adversary can reconstruct the i-th bit of the one-time pad. PRP computes that bit as $OTP_i = \oplus_{j=1}^{\lambda} p_{i,j}$. If the adversary does not store at least one of the bits $p_{i,j}$, $j = 1, \ldots, \lambda$ then it has no information at all on OTP_i, since the bit that the adversary does not store is completely random. In this case, the adversary has probability $1/2$ to correctly guess OTP_i.

If the adversary *does* store all the bits $p_{i,j}$, $j = 1, \ldots, \lambda$ then when π is revealed the adversary may be able to correctly compute OTP_i. Therefore, the probability that the adversary correctly computes OTP_i is at most the probability that the adversary stores all of $p_{i,j}$, $j = 1, \ldots, \lambda$.

S and R choose an λ-tuple at random to create OTP_i from all the λ-tuples that PRP allows. They use the shared key of *Process 1* to choose their λ channels, so there are b^λ possible choices. Given the λ channels they have λm shared bits, out of which they choose uniformly at random λ bits. On the other hand, the adversary stores at most $bm\lambda/2$ bits and must choose a λ-tuple out of these bits.

Thus, the probability that the adversary obtains the correct bits is at most

$$\begin{aligned} \frac{\binom{\frac{bm\lambda}{2}}{\lambda}}{b^\lambda \binom{m\lambda}{\lambda}} &\leq \frac{(bm\lambda/2)!(m\lambda-\lambda)!}{(bm\lambda/2-\lambda)!b^\lambda(m\lambda)!} \\ &= \frac{(bm\lambda/2)\cdot(bm\lambda/2-1)\cdots(bm\lambda/2-\lambda+1)}{b^\lambda(m\lambda)\cdot(m\lambda-1)\cdots(m\lambda-\lambda+1)} \\ &\leq 2^{-\lambda} \end{aligned}$$

Therefore, the probability that the adversary succeeds in obtaining any of the m bits is at most

$$\Sigma_{i=1}^{m} 2^{-\lambda} = m2^{-\log m - k} = 2^{-k}.$$

□

4. PERMUTATION ENCRYPTED PROTOCOL

In PEP, the number of bits shared in *Process 1*, is larger than in the PRP case. The shared key is reusable for an exponential (in the security parameter k) number of encryptions.

PEP is similar to PRP, but instead of a permutation revealing phase, the shared bits of *Process 1* define the permutation π that is used in *Process 2*. The same permutation is used over and over in N rounds to generate successive blocks of m bits for the OTP.

In this section we use $\lambda(N)$ to denote $\log(mN) + k$. Thus, the notation λ used in the PRP section can be written as $\lambda(1)$.

The length of the shared key after *Process 1* of PEP is equal to

$$\lambda(N)\log b + m\lambda(N)\log(m\lambda(N)).$$

The first summand, $\lambda(N)\log b$, defines $\lambda(N)$ blocks of $\log b$ bits. Each such block determines a channel on which to receive a block of bits $R_{j,i}$ (similarly to PRP).

The second summand, $m\lambda(N)\log(m\lambda(N))$, determines a permutation on all the $m\lambda(N)$ bits that R and S share in order to obtain m bits for an OTP.

S and R perform a similar procedure to *Process 2* in PRP to derive m bits. In phase 1, S sends $m\lambda(N)$ random bits to R over each of the b channels. For each block of m bits, only a single channel is correct, while all other channels carry random dummy bits. Similarly to PRP, the correct channel is defined by bits shared in *Process 1*.

In phase 2, S and R use their shared permutation to reorder the $m\lambda(N)\log(m\lambda(N))$ shared bits in a matrix of size $\lambda(N) \times m$. An exclusive-or on all the bits of a matrix column yields an OTP bit. Performing this process on each of the m columns of the matrix produces an OTP of length m bits.

S and R repeat this process N times. Each time S sends new random bits and both S and R use the same permutation.

THEOREM 2. *Assume, that an adversary Ad is computationally unbounded, but its storage is limited to s_{ad} bits, $s_{ad} < L/2$, where $L = min(T, bm(\log m + \log N + k))$. Then,* PEP *outputs an* OTP *of mN bits that S and R share, and the probability that the adversary obtains even a single bit of the* OTP *is less than* 2^{-k}.

PROOF. Since the adversary can store no more than $L/2$ bits, it can store at most half of the bits that S transmits during the execution of PEP.

We set an index i, $1 \leq i \leq m$ and bound the probability that the adversary can reconstruct the i-th bit of the one-time pad.

PRP. Protocol for Sender S

Input:
$C = \{c_1, \ldots, c_b\}$ is a set of b
channels, m is the output length
and k is a security parameter.

Process 1:
Generate a shared key with R
$s = \{s_1, \ldots, s_{\log m+k}\}$,
    where s_i is a block of $\log b$ bits
    for every $i = 1, \ldots, \log m + k$.

Process 2:
Phase 1:
for $j = 1$ to $\log m + k$ do
    for $i = 1$ to b do
        Generate a random string
        $R_{j,i} = \{r^1_{j,i}, \ldots, r^m_{j,i}\}$
        Transmit $R_{j,i}$ over channel c_i.
    end for
end for

Phase 2: Permutation Sharing
Choose a random permutation π
    over a set of $m(\log m + k)$ elements.
Send π to R
Let λ denote $\log m + k$.
Let P be a bit matrix of size $\lambda \times (n)$.
Generate P by applying π to
$R^1_{1,s_1}, \ldots, R^m_{1,s_1}, \ldots, R^1_{\lambda,s_\lambda}, \ldots, R^m_{\lambda,s_\lambda}$.

Generating OTP
for $i = 1$ to m do
    for $j = 1$ to $\log m + k$ do
        $OTP_i = \oplus_{j=1}^{\lambda} p_{i,j}$
    end for.
end for.

PRP. Protocol for Receiver R

Input:
$C = \{c_1, \ldots, c_b\}$ is a set of b
channels, m is the output length
and k is a security parameter.

Process 1:
Generate a shared key with S
$s = \{s_1, \ldots, s_{\log m+k}\}$,
    where s_i is a block of $\log b$ bits
    for every $i = 1, \ldots, \log m + k$.

Process 2:
Phase 1:
for $j = 1$ to $\log m + k$ do
    Store R_{j,s_j}, the m bits received on
    channel s_j.
end for

Phase 2: Permutation Sharing
Receive from S a random permutation π
Let λ denote $\log m + k$.
Let P be a bit matrix of size $\lambda \times (n)$.
Generate P by applying π to
$R^1_{1,s_1}, \ldots, R^m_{1,s_1}, \ldots, R^1_{\lambda,s_\lambda}, \ldots, R^m_{\lambda,s_\lambda}$.

Generating OTP
for $i = 1$ to m do
    for $j = 1$ to $\log m + k$ do
        $OTP_i = \oplus_{j=1}^{\lambda} p_{i,j}$
    end for.
end for.

Figure 1: Permutation Revealing Protocol PRP.

PEP computes that bit as $OTP_i = \oplus_{j=1}^{\lambda(N)} p_{i,j}$. The probability that the adversary obtains the correct bits is at most

$$\begin{aligned}\frac{\binom{\frac{bm\lambda(N)}{2}}{\lambda(N)}}{b^{\lambda(N)}\binom{m\lambda(N)}{\lambda(N)}} &\leq \frac{(bm\lambda(N)/2)!(m\lambda(N)-\lambda(N))!}{(bm\lambda(N)/2-\lambda(N))!b^{\lambda(N)}(m\lambda(N))!} \\ &= \frac{(bm\lambda(N)/2)\cdots(bm\lambda(N)/2-\lambda(N)+1)}{b^{\lambda(N)}(m\lambda(N))\cdots(m\lambda(N)-\lambda(N)+1)} \\ &\leq 2^{-\lambda(N)}\end{aligned}$$

Therefore, the probability that the adversary succeeds in obtaining any of the mN bits is at most

$$\Sigma_{i=1}^{mN} 2^{-\lambda(N)} = mN2^{-\log(mN)-k} = 2^{-k}.$$

□

Improved PEP. The expansion factor of PEP can be often improved by the following procedure, which we refer to as *improved* PEP.

1. Begin with a shared, initial key of length $\log b(\log \xi + k + 1)$, where ξ is the length of the initial key for PEP (with security parameter $k+1$).
2. Use PRP to expand this key to a shared output string of length ξ.
3. Perform PEP with a shared key of length ξ.

The expansion factor of improved PEP is better than that of PEP when $\xi \geq \log b(\log \xi + k)$.

5. IMPROVING KEY EXCHANGE ALGORITHMS

All previous BSM key exchange algorithms required the participants to sample bits at random locations. If the random source is transmitted in frames of length α, then each such protocol must read α times as many bits as the protocol would require if bits were accessible individually. In this section we show how to modify a given key exchange protocol to reduce the number of bits that it must read from the random source.

Denote by P the given key exchange protocol, and denote by $f(n)$ the number of bits that a key exchange protocol reads from a random source of length n when distinct bits are available individually. In our setting, such a protocol must read αf bits.

If $\alpha \geq k + \log m$, we can improve the protocol as follows. Choose $k+\log m$ random blocks of $f(n)$ bits and read them. After the transmission of the random source ends, exchange a random permutation for an $f(n)\times(k+\log m)$ matrix. Obtain $f(n)$ random bits by running exclusive-or on all bits of a column of the permuted matrix. Complete the scheme by executing the extractor of P on the $f(n)$ random bits. The improved scheme reads $\alpha(k+\log m)$ bits instead of $\alpha f(n)$ bits.

In practice, $\alpha \geq k + \log m$ is the typical scenario. For example, if $k = 64$ and $m \leq 2^{56}$ bytes, we need α to be greater than 16 bytes, which is certainly the case for most wireless protocols.

A second scheme improves the original protocol when

$$f(n) \leq f(\frac{n}{\alpha}) + \frac{(\alpha-1)f(\frac{n}{\alpha})}{k+\log m}.$$

T he participants run P for a random source of n/α bits. Each location that P samples determines a block of α bits instead of a single bit. Therefore, the two parties read $\alpha f(n/\alpha)$, compared to $\alpha f(n)$ in the original scheme. The sender and receiver extract enough bits by taking the first bit of each block for $f(n/\alpha)$ bits. They obtain more bits by the same method we have already used of permuting the remaining bits, a matrix of $k+\log m$ over $\frac{(\alpha-1)f(\frac{n}{\alpha})}{k+\log m}$ bits and getting $\frac{(\alpha-1)f(\frac{n}{\alpha})}{k+\log m}$ random bits by exclusive-or on all the bits of each column.

6. CONCLUSIONS

We present a new technique based on defining sections of a random sequences, rather than bits, and the (later) use of random permutation of the bits among the concatenation of the chosen sections. The technique fits the multi-frequency wireless communication among sensors and mobile ad-hoc devices where the choice of a subset of the frequencies (rather than a single one as analyzed above) implies exponentially growing security parameter.

For completeness we mentioned known techniques that authenticate other non adversarial devices by physical layer fingerprints, and the ability to establish a short secret (in the bounded storage model) from scratch, using the birthday paradox.

We believe the combined techniques are simple to implement and can be efficiently used in practice.

7. REFERENCES

[1] 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.

[2] Y. Z. Ding, M. O. Rabin, "Hyperencryption and Everlasting Security", *Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pp. 1-26, 2002.

[3] C. Cachin, U. Maurer, "Unconditional Security Against Memory-Bounded Adversaries", CRYPTO'97, pp. 292-306, 1997.

[4] S. Dziembowski, U. Maurer, "Tight security proofs for the bounded-storage model",34th Annual ACM Symposium on Theory of Computing (STOC'02), pp. 341-350, 2002.

[5] S. Dziembowski, U. Maurer "On Generating the Initial Key in the Bounded-Storage Model", *Advances in Cryptology-EUROCRYPT 2004*, Vol. 3027, pp. 126-137, 2004.

[6] D. Harnik, M. Naor, "On Everlasting Security in the Hybrid Bounded Storage Model", *ICALP*, 2006.

[7] Chi-Jen Lu, "Encryption against Storage-Bounded Adversaries from On-Line Strong Extractors", *Journal of Cryptology*, Vol. 17 No. 1, pp. 27-42, 2004.

[8] U. Maurer, "Conditionally-Perfect Secrecy and a Provably-Secure Randomized Cypher", *Journal on Cryptology*, Vol. 5, No. 1, pp. 53-66, 1992.

[9] D. R. Stinson, *"Cryptography. Theory and Practice"*, Chapman and Hall/CRC, Third edition, 2006.

[10] S. P. Vadhan, "Constructing Locally Computable Extractors and Cryptosystems in the Bounded-Storage Model", *Journal of Cryptology*, Vol. 17 No. 1, pp. 43-77, 2004.

[11] L. Xiao, L. Greenstein, N. Mandayam, and W. Trappe,"Fingerprints in the ether: Using the physical layer for wireless authentication," *Proc. IEEE International Conference on Communications*, Glasgow, Scotland, June 2007.

[12] D. Zuckerman, "Randomness-Optimal Oblivious Sampling", *Random Struct. Algorithms Journal*, 11(4), pp. 345-367, 1997.