RPM: Lightweight Communication Security from Additive Stream Ciphers

(NIST Lightweight Cryptography Workshop 2015)

Giovanni Di Crescenzo[*] Glenn O. Veach[†]

Abstract

We investigate recently introduced lightweight cryptographic techniques that efficiently target communication security to meet use cases such as interconnected devices working in Low Power Wide Area (LPWA) networks. First of all, we isolate these lightweight techniques as non-linear, additive, stream ciphers. Then, we describe theoretical analysis to support some evidence of plausible security: (a) high-period stream ciphers rule out certain classes of attacks; (b) idealized versions of the analyzed lightweight stream ciphers have high period. Finally, we also show empirical performance results, demonstrating effective algorithms that easily fit into the limited resources of constrained environments, while outperforming more traditional methods (e.g., block ciphers) by almost one order of magnitude.

1 Introduction

In today's interconnected world and the "Internet of Things", certain sufficiently constrained environments exist such that the traditional NIST-approved cryptographic algorithms do not perform to the operating requirements. Lightweight cryptographic primitives, although possibly not as secure as conventional cryptographic primitives (e.g., block ciphers, hash functions), may be efficient enough to enable applications over such constrained environment. For cryptography primitives in conventional desktop and/or server environments, several years of modern cryptography research were needed to find the right formal definitions so to rigorously state their security properties under carefully formulated models and assumptions. For lightweight cryptography, such a process has barely started. In fact, given the current state of the art in lightweight cryptography, giving evidence that any level (even though smaller) of security is achievable, while guaranteeing the desired higher performance, is already a non-trivial challenge. In this paper, we attempt to address such a challenge with respect to some recently proposed lightweight cryptographic solutions for certain relevant application use cases.

Our contribution. In this paper we analyze lightweight cryptographic solutions that are being used in RPM[1] [Rel14, Rel14b, Rel14c, Rel14d] to achieve communication security among interconnected

[*]Applied Communication Sciences, 150 Mt. Airy Rd., 1S-092, Basking Ridge, NJ, 07920. E-mail: gdicrescenzo@appcomsci.com

[†]Relevant Security Corp., 990 South Broadway, Suite 300, Denver, CO, 80209-4275. E-mail: gveach@rscorp.com

[1]RPM is covered by one or more of the following patents: U.S. Patent No. 8,649,520; U.S. Patent 8,144,875; U.S. Patent No. 8,144,874; U.S. Patent No. 7,899,185; U.S. Patent No. 6,445,797; U.S. Patent No. 6,058,189; U.S. Patent No. 6,002,769; U.S. Patent Publication No. 20060034456; U.S. Patent Publication No. 20030149876; Japan Patent No. 5047291; U.S. Patent 8,649,520; U.S. Patent Application No. 14/176,284 (pending)

devices working in Low Power Wide Area (LPWA) networks. These techniques were designed to take into consideration various constraining factors including lightweight computing of sensors, power requirements, data rates and bandwidth in conjunction with Quality of Service (QoS) requirements that are Medium QoS such as residential smart meters or HVAC, as well as High QoS such as heart monitors or power utility smart grid. In the rest of the paper we also detail application use cases for these and similar technologies.

Our technical contribution starts by showing quantitative aspects of the following known relationships between stream ciphers with long period and secure symmetric encryption schemes: (1) stream ciphers generating a pseudo-random sequence of values can be turned into secure symmetric encryption schemes; and (2) the period of stream ciphers directly bounds the time window of security for the associated symmetric encryption scheme.

We then consider the RPM constructions and isolate the underlying lightweight cryptographic primitives as two additive and non-linear stream ciphers. In RPM, these are used in combination with conventional cryptographic techniques and modes of operation to provide lightweight constructions that very efficiently target a number of interesting communication security properties, including: confidentiality via encryption, implicit mutual and constant authentication, continuous key management by key re-freshing.

On a more practical side, we report performance results, showing that symmetric encryption based on the two lightweight RPM stream ciphers is faster by almost one order of magnitude than AES-based symmetric encryption.

On a more theoretical side, we analyze the security of encryption schemes based on the two lightweight RPM stream ciphers. Specifically, we show that idealized versions of them have a long period. As perhaps of independent interest, the analysis is based on estimating the number of collisions in lightweight functions, and establishing linear independence and/or ranks of matrices of a specific structure. Our analysis methodology is based on carefully deriving idealized version of each specific lightweight construction, and obtaining a proof that the idealized version has large period to potentially infer some confidence of security against some class of attacks (i.e., attacks based on low stream cipher periods) for the original, non-idealized, version. On one hand, this idealization is desirable in that it makes the analysis more tractable, it provides analysis related to constructions for which no analysis exists, and it sheds light on potential construction properties. On the other hand, we caution the reader that formal statements proved for an idealized version may not translate to the non-idealized version. This fact is not new: as an example, as part of an active cryptography research area, researchers often consider idealized versions of cryptographic constructions and prove them to be secure under the existence of a random oracle. Indeed, the random oracle methodology, first suggested in [BeRo93], is being often used, in many variants, to give some informative evidence of security in schemes for which a proof is otherwise unavailable or hard to obtain. However, caution is often advocated by researchers in extrapolating these results, as several scheme examples have been constructed where an idealized variant is secure but the non-idealized is insecure (starting with [CaGoHa04]).

Organization of the paper. In Section 2 we detail basic definitions and relationships between symmetric encryption schemes and stream ciphers. In Section 3 and 4 we present the two stream ciphers underlying the RPM lightweight communication security solutions. In Section 5 we analyze the period of idealized versions of the described stream ciphers. We conclude with performance results in Section 6 and application use cases in Section 7.

2 Definitions, Background and Preliminaries

In this section we give some basic definitions, including pseudo-random generators, random functions, symmetric encryption schemes and stream ciphers. We then recall some known facts relating properties like pseudo-randomness and period of stream ciphers to the security of encryption schemes.

Basic definitions and primitives. We say that two distributions D_0 and D_1 are (t, ϵ)-*indistinguishable* if for any algorithm A running in time t, it holds that

$$\left| \text{Prob}\left[\ x \leftarrow D^0 : A(x) = 1\ \right] - \text{Prob}\left[\ x \leftarrow D^1 : A(x) = 1\ \right] \right| \leq \epsilon$$

(i.e., no algorithm running in time t can distinguish with probability $\geq \epsilon$ if a random sample came from one distribution or the other).

A function $\{G_{n,m} : n \in \mathcal{N}, m \geq n+1\}$, with $G_{n,m} : \{0,1\}^n \to \{0,1\}^m$, is a (t, ϵ)-*pseudo-random generator (family)* if the distribution $\{x \leftarrow \{0,1\}^n : G_{n,m}(x)\}$ is (t, ϵ)-indistinguishable from the uniform distribution over $\{0,1\}^m$. A function $\{R_n : n \in \mathcal{N}\}$,, with $R_n : \{0,1\}^n \to \{0,1\}^n$ is a *random function* if the function R_n is chosen with distribution uniform across all possible functions with n-bit inputs and outputs. A *random permutation* is a random function such that R_n is also a permutation from the input domain to the output domain.

Symmetric Encryption. We will define (conventional) symmetric encryption schemes, as well as a stateful version of them. Except for specifically treated cases, definitions are applicable to both.

Let K be a key space, M be a message space, and C be a ciphertext space. A *(stateless) encryption scheme* $ES = (E, D)$ and a *stateful encryption scheme* $sES = (sE, sD)$ are each specified by a pair of algorithms, with the following syntax and properties.

Syntax. The *encryption algorithm* E is a probabilistic algorithm that, on input a key $k \in K$ and a *plaintext* $m \in M$, returns a *ciphertext* $c \in C$. The *decryption algorithm* D is a probabilistic algorithm that, on input a key $k \in K$ and a ciphertext $c \in C$, returns either a plaintext $m' \in M$ or a special symbol $\perp$ indicating that the ciphertext is invalid.

The *stateful encryption algorithm* sE is a probabilistic algorithm that, on input a key $k \in K$, a current state es_i, for an integer $i \geq 0$, and a *plaintext* $m \in M$, returns a *ciphertext* $c \in C$ and an updated state es_{i+1}. The *decryption algorithm* D is a probabilistic algorithm that, on input a key $k \in K$, a current state ds_i, for an integer $i \geq 0$, and a ciphertext $c \in C$, returns an updated state ds_{i+1}, and either a plaintext $m' \in M$ or a special symbol $\perp$ indicating that the ciphertext is invalid.

Properties: Decryption Correctness. The decryption correctness property of an encryption scheme states that the decryption algorithm (almost) always returns the same plaintext used to compute the ciphertext using the encryption algorithm. Formally, we say that scheme (KG,E,D) satisfies *decryption correctness* if for any $m \in M$, it holds that the following probability is negligible:

$$\text{Prob}\left[\ k \leftarrow K; c \leftarrow E(k,m); m' \leftarrow D(k,c) : m' \neq m\ \right].$$

The formalization of this property for stateful encryption schemes is similar.

Properties: Security. Several formalizations for the security notion of symmetric encryption schemes have been considered in the literature, under different adversary models. Here, we consider the (most typically used) adversary model where the adversary is allowed to: (1) eavesdrop the communication between sender and receiver; (2) perform a chosen-message attack. A number of security definitions in this adversary model have been proposed in the literature (see, for instance, [Bdjr97] for comparison among many of these notions). Here, we use a notion that was called "real-or-random" security.

Informally speaking, real-or-random security says that an adversary would be unable to tell apart encryptions of known data from encryptions of random data. Formally, let A be an adversary with access to an oracle O; that is, A can repeatedly ask a query to O and obtain an associated reply. Consider O as either (1) the encryption algorithm $E(k,\cdot)$ that, on input a query q by A, returns $E(k,q)$, where k is random element from K and unknown to A, or (2) the encryption algorithm $rE(k,\cdot)$, that, on input a query q by A, returns $E(k,r)$, for a random element r from M, and a random element k from K, both r and k being unknown to A. We say that scheme (KG,E,D) satisfies (t,q,ϵ)- *"real-or-random" security* if for any $m \in M$, and any adversary running in time at most t, and making at most q oracle queries, it holds that:

$$\Pr\left[out \leftarrow A^{E(k,\cdot)} \,:\, out = 1\right] - \Pr\left[out \leftarrow A^{rE(k,\cdot)} \,:\, out = 1\right] \;\; \leq \; \epsilon.$$

Remarks. The main goal in the design of symmetric encryption schemes consists of designing a scheme that satisfies t, q, ϵ, for t, q polynomial and ϵ negligible in the security parameter or, more concretely, for 'large enough' t, q and 'small enough' ϵ. We also note that by the transitive property, the inability to distinguish an encryption of any message from an encryption of a random message, implies the inability to distinguish encryptions of any two messages. Finally, we note that the formalization of this property for stateful encryption schemes is similar.

Examples. We recall the "one-time pad" encryption scheme, as its properties will be implicitly used in the rest of the paper. It is defined as follows. The key space K, the message space M and the ciphertext space C are defined to be $\{0,1\}^\ell$, for the same positive integer ℓ. On input $k \in K$ and $m \in M$, the encryption algorithm returns $c = m \oplus k$. On input $k \in K$ and $c \in C$, the decryption algorithm returns $m' = c \oplus k$. This scheme satisfies decryption correctness and (t,q,ϵ)-"real-or-random" security, for unbounded t, q and $\epsilon = 0$, but this level of security does require $|K| \geq |M|$.

For block ciphers like AES, when combined with block cipher modes of operation, like CBC or Counter Mode, one can obtain $|K|$ constant with respect to $|M|$ and prove that they satisfy (t,q,ϵ)-"real-or-random" security, for 'large enough' t, q and 'small enough' ϵ, assuming the underlying block cipher is 'sufficiently close' to a random permutation with the same input and output lengths.

Stream ciphers. Informally, a stream cipher can be considered as a method to generate a pseudo-random sequence of keys, along with some state information.

Let K be a key space, S be a state space, A_k be a key alphabet, A_m be a plaintext alphabet, A_c be a ciphertext alphabet, and $M = A_m^*$ (resp., $C = A_c^*$) be a message space (resp., ciphertext space) formed by an indefinite cartesian product of the plaintext alphabet (resp., ciphertext alphabet). A *(synchronous) stream cipher* $SC = (F, G, H, H^{-1})$ is specified by a tuple of algorithms, with the following syntax and properties.

Syntax. Let s_0 be the initial state. The *next state algorithm* F is a deterministic algorithm that, on input a (master) key $k \in K$, and a *current state* $s_i \in S$, returns a *next state* $s_{i+1} \in S$, for any index integer $i \geq 0$. The *next key algorithm* G is a deterministic algorithm that, on input a (master) key $k \in K$, and a current state $s_i \in S$, returns a *next key* $k_i \in A_k$, for any index integer $i \geq 0$. The *character encryption algorithm* H is a deterministic algorithm that, on input a current key $k_i \in A_k$ and a plaintext character $m_i \in A_m$, returns a ciphertext character $c_i \in A_c$. The *character decryption algorithm* H^{-1} is a deterministic algorithm that, on input a current key $k_i \in A_k$ and a ciphertext character $c_i \in A_c$, returns a plaintext character $m_i \in A_m$.

Other Properties of stream ciphers. Let $m = n \cdot t$, for some integer $t > 0$. Define function $prG_{n,q}$ as the function that, for any index integer $i > 0$, returns $(k_i, \ldots, k_{i+t})$, where $s_{i+1} = F(k, s_i)$ and

$k_i = G(k, s_i)$, for $i = 1, \ldots, q$. We say that the stream cipher $SC = (F, G, H, H^{-1})$ returns a (t, ϵ)-*pseudo-random sequence of q keys* if the function $prG_{n,q}$ is a (t, ϵ)-pseudo-random generator.

We say that the stream cipher $SC = (F, G, H, H^{-1})$ is *additive* if $H(k_i, m_i) = k_i \oplus m_i$ and $H^{-1}(k_i, c_i) = k_i \oplus c_i$, for all integers $i \geq 0$.

A sequence $s = (s_0, s_1, \ldots)$ is called *periodic* if there are integers $i_0, t > 0$ such that $s_{i+t} = s_i$ for and all $i \geq i_0$. The least positive integer t satisfying this property is called the *period of sequence s*. The period of the sequence of next keys returned by a stream cipher SC is called the *period of stream cipher SC*.

Relationships between stream ciphers and symmetric encryption. We formulate two known results on using stream ciphers for stateful symmetric encryption. (The qualitative results are known, but we have never seen such results with detailed quantitative analysis.) The first proposition says that an additive stream cipher returning a pseudo-random sequence of keys can directly be used to produce a secure encryption scheme.

Proposition 1 Let SC be an additive stream cipher. If SC returns a (t, ϵ)-pseudo-random sequence of q keys, then SC-ES is a stateful symmetric encryption scheme that satisfies decryption correctness and (t', q', ϵ')-real-or-random security, for $t' = t - O(q)$ and $\epsilon' = \epsilon$.

The next proposition says that the period of an additive stream cipher limits the number of messages that might be securely encrypted using a scheme based on that stream cipher.

Proposition 2 Let SC be an additive stream cipher with period p. Also, let SC-ES be a stateful symmetric encryption scheme that satisfies decryption correctness. For any $\epsilon > 0$, SC-ES does not satisfy (t, q, ϵ)-real-or-random security, for $t = O(p)$ and $q \geq p$.

Remarks. Proposition 1 suggests a way to build efficient encryption schemes based on stream ciphers that produce pseudo-random sequences of keys. The best known approaches to design stream ciphers that are conjectured to have this property are based on block ciphers or cryptographic hash functions, which we try to outperform in lightweight cryptography constructions. Thus, Proposition 1 can also be interpreted as setting a goal when designing lightweight stream ciphers: trying to approximate the pseudo-randomness property using more efficient constructs than block ciphers. This is one rationale behind the design of the constructions in Section 3 and 4.

Proposition 2 suggests a way to limit a class of natural attacks when building efficient encryption schemes based on lightweight stream ciphers: design a stream cipher with a high period. This is another rationale behind the design of the constructions in Section 3 and 4.

3 A First RPM Stream Cipher

In this section we describe the first RPM cipher. We first present its two primitive functions, and then define the stream cipher based on these two functions.

3.1 A Primitive Function: PDAF

The first primitive function is called Position Digit Algebra Function (briefly, PDAF). It takes as input two n-element lists x and y of numbers from a set of size r and returns as output an n-element list z of numbers from a set of size r. Each component of the output list is computed as the sum (modulo r) of two elements from list x, where the index of the second element within the list is selected according to an element of list y. As the additions do not require a carry, the entire PDAF function can be executed in parallel. We now give a more formal description.

Parameters for PDAF: positive numbers n, r, where n is even.

Input to PDAF: lists $x = (x[0], \ldots, x[n-1])$ and $y = (y[0], \ldots, y[n-1])$, where $x[i], y[i] \in \{0, \ldots, r-1\}$, for $i = 0, \ldots, n-1$.

Instructions for PDAF:

```
1. For $i = 0, \ldots, n-1$,
       set $j(i) = (i + y(i)) \bmod n$
       set $z[i] = (x[i] + x[j(i)]) \bmod r$
2. Return: $z = (z[0], \ldots, z[n-1])$.
```

Example. Assume $n = 6$ and $r = 16$. If $x = (3, 8, 7, 11, 1, 15)$ and $y = (2, 11, 5, 8, 8, 6)$ then $z = (10, 11, 15, 10, 4, 14)$.

3.2 A Primitive Function: OWC

The second primitive function is called One-Way Cut (briefly, OWC). It takes as input an n-element list x of numbers from a set of size r and returns as output an $n/2$-element list z of numbers from a set of size r. We describe the most basic variant, where the output list is computed by summing modulo r two of the elements from the input list, one from each of two input sublists. In another variant, not described here, the function uses a parameter sv from set $\{1, \ldots, n/2\}$, called a separation number, and used to partition list x into equal-length lists. As the additions do not require a carry, the entire OWC function can be executed in parallel. We now give a more formal description.

Parameters for OWC: positive numbers n, r, where n is even.

Input to OWC: list $x = (x[0], \ldots, x[n-1])$, where $x[i] \in \{0, \ldots, r-1\}$, for $i = 0, \ldots, n-1$.

Instructions for OWC:

```
1. For $i = 0, \ldots, n/2 - 1$,
       set $z[i] = (x[2i] + x[2i+1]) \bmod r$
2. Return: $z = (z[0], \ldots, z[n/2 - 1])$.
```

Example. Assume $n = 6$ and $r = 16$. If $x = (15, 4, 2, 12, 8, 11)$ then $z = (3, 14, 3)$.

3.3 The Stream Cipher rpmSC1

The additive stream cipher rpmSC1=(F, G) is built using primitives PDAF and OWC. Specifically, the next state algorithm F uses PDAF and the next key algorithm G uses OWC. We now give a more formal description.

Parameters for rpmSC1: positive numbers n, r, where n is even.

Input to rpmSC1: the initial state (s_{-1}, s_0), where s_0 is chosen as a random number from $\{0, \ldots, r-1\}^n$, and s_{-1} is the shared key k, also a random number from $\{0, \ldots, r-1\}^n$.

Next state algorithm F: On input the current state (s_{i-1}, s_i), algorithm F returns the next state (s_i, s_{i+1}), where $s_{i+1} = \text{PDAF}\ (s_i, s_{i-1})$, $s_i = s_{i-1}$, and all values s_{i-1}, s_i, s_{i+1} are numbers in $\{0, \ldots, r-1\}^n$, for any $i \geq 1$.

Next key algorithm G: On input the current state (s_{i-1}, s_i), algorithm G returns the next key $k_i = \text{OWC}\ (s_i)$, where value s_i is a number in $\{0, \ldots, r-1\}^n$, and value k_i is a number in $\{0, \ldots, r-1\}^{n/2}$ for any $i \geq 1$.

4 A Second RPM Stream Cipher

In this section we describe the second RPM cipher. We first present its two primitive functions, and then define the stream cipher based on these two functions.

4.1 A Primitive Function: CMBN

The first primitive function is called Combine (briefly, CMBN). It takes as input two n-element lists x and y of numbers from a set of size r and returns as output an n-element list z of numbers from a set of size r. Each component of the output is computed as the sum (modulo r) of one element chosen from list x and one element chosen from list y. For each of these two elements, the index choosing the element from its list is computed according to a recurrence relation involving the next value from the other input list. As the additions do not require a carry, the entire CMBN function can be executed in parallel. We now give a more formal description.

Parameters for CMBN: positive numbers n, r, where n is even.

Input to CMBN: lists $x = (x[0], \dots, x[n-1])$ and $y = (y[0], \dots, y[n-1])$, where $x[h], y[h] \in \{0, \dots, r-1\}$, for $h = 0, \dots, n-1$.

Instructions for CMBN:

```
1. Set $i_{-1} = j_{-1} = -1$
2. For $h = 0, \dots, n-1$,
       set $i_h = (i_{h-1} + 1 + x[h]) \bmod n$
       set $j_h = (j_{h-1} + 1 + y[h]) \bmod n$
       set $z[h] = (x[j_h] + y[i_h]) \bmod r$
3. Return: $z = (z[0], \dots, z[n-1])$.
```

Example. Assume $n = 6$ and $r = 16$. If $x = (3, 13, 8, 6, 1, 7)$ and $y = (13, 9, 14, 15, 10, 2)$, then $z = (15, 9, 6, 3, 9, 11)$.

4.2 A Primitive Function: EXTC

The second primitive function is called Extract (briefly, EXTC). It takes as input two n-element lists x and y of numbers from a set of size r and returns as output an n-element list z from a set of size r . Each element of the output list is equal to one element chosen from list x, where the index choosing the element is computed according to a recurrence relation involving the next value from the input list y. In a variant, the output is computed by summing similarly chosen elements from both input lists. As the additions do not require a carry, the entire EXTC function can be executed in parallel. We now give a more formal description.

Parameters for EXTC: positive numbers n, r, where n is even.

Input to EXTC: lists $x = (x[0], \dots, x[n-1])$ and $y = (y[0], \dots, y[n-1])$, where $x[h], y[h] \in \{0, \dots, r-1\}$, for $h = 0, \dots, n-1$.

Instructions for EXTC:

```
1. Set $i_{-1} = -1$
2. For $h = 0, \dots, n-1$,
       set $i_h = (i_{h-1} + 1) + y[h] \bmod n$
       set $z[h] = x[i_h] \bmod r$ (variant : $z[h] = x[i_h] + y[i_h] \bmod r$)
3. Return: $z = (z[0], \dots, z[n-1])$.
```

Example. Assume $n = 6$ and $r = 16$. If $x = (0, 2, 11, 1, 6, 5)$ and $y = (1, 2, 12, 6, 5, 15)$, then $z = (2, 6, 5, 0, 0, 6)$.

4.3 The Stream Cipher rpmSC2

The additive stream cipher rpmSC2=(F, G) is built using primitives CMBN and EXTC. Specifically, both the next state algorithm F and the next key algorithm G use CMBN and EXTC. We now give a more formal description.

Parameters for rpmSC2: positive numbers n, r, where n is even.

Input to rpmSC2: the initial state s_0, the (master) keys mk_0, mk_1, where s_0, mk_0, mk_1 are random numbers from $\{0, \ldots, r-1\}^n$.

Next key function G: On input the current state s_i, and keys mk_0, mk_1, function G does the following

1. set $v[h] = s_i[h] + mk_0[h] \bmod r$, for $h = 0, \ldots, n-1$;
2. set $v = (v[1], \ldots, v[n])$;
3. set $a = CMBN(mk_1, v)$;
4. set $z = EXTC(a, mk_1)$;
5. return: $z = (z[0], \ldots, z[n-1])$.

Next state function F: On input the current state s_i, and keys mk_0, mk_1, function F does the following

1. run steps 1-4 in the computation of function G;
2. set $s_{i+1}(h) = z[h] + v(h) \bmod r$, for $h = 0, \ldots, n-1$;
3. return: $s_{i+1} = (s_{i+1}[1], \ldots, s_{i+1}[n])$.

5 Security Analysis

In this section we provide some considerations on the period of the two described stream ciphers. Specifically, we define idealized variants of the two described stream ciphers, obtained by replacing or augmenting the stream ciphers with a random function, and we analyze the expected period of these two idealized variants. As the determined periods are large, this suggests that so might be the periods for the described stream ciphers, under suitable idealized behavior assumptions (but see the discussion in the introduction for a cautious interpretation of these results).

5.1 On the Period of rpmSC1

We define an idealized version of rpmSC1, denoted as rpmSC1$_r$, by first defining a slightly modified next state function F$'$ and next key function G$'$, and then plugging these functions into rpmSC1.

Next state algorithm F$'$: Let R be a random function. On input the current state (s_{i-1}, s_i), algorithm F$'$ returns the next state (s_i, s_{i+1}), obtained by computing u_{i+1} = PDAF (s_i, s_{i-1}) and $(s_i, s_{i+1}) = R(u_{i+1})$, where all values $s_{i-1}, s_i, s_{i+1}, u_{i+1}$ are numbers in $\{0, \ldots, r-1\}^n$, for any $i \geq 1$.

Next key algorithm G$'$: On input the current state (s_{i-1}, s_i), algorithm G$'$ returns the next key $k_i \in \{0, \ldots, r-1\}^{n/2}$ for any $i \geq 1$, computed as follows. First, G$'$ obtains a list u_i of n values $(u_i[0], \ldots, u_i[n-1])$, where $u_i[2j] = s_{i-1}[j]$ and $u_i[2j+1] = s_i[j]$, for $j = 0, \ldots, n/2-1$. Then G$'$ computes k_i = OWC (u_i), where value s_i is a number in $\{0, \ldots, r-1\}^n$, and value k_i is a number in $\{0, \ldots, r-1\}^{n/2}$ for any $i \geq 1$.

The rpmSC1$_r$ stream cipher. Formally, cipher rpmSC1$_r$ is defined starting from rpmSC1, and replacing the next state algorithm F and the next key algorithm G, as described in Section 3, with the previously defined algorithms F$'$ and G$'$, respectively. Intuitively, this idealized stream cipher is defined so to satisfy the following properties: (a) input to the PDAF primitive can be considered random and independent strings; and (b) the current key is derived as a direct OWC computation on input the current state.

We obtain the following

Theorem 3 The expected value of the period of rpmSC1$_r$ is $\geq r^{n/2-0.85\log_2 n-1}$.

The rest of this subsection is dedicated to the proof of Theorem 3.

For every $x, y \in \{0, \ldots, r-1\}^n$, a *left* $(y,$PDAF$)$-*collision* of x is a value $x' \in \{0, \ldots, r-1\}^n$ such that PDAF $(x, y) =$ PDAF (x', y). Our estimate of the period of rpmSC1$_r$ will be directly related to the total number of possible states and the expected number of left $(y,$PDAF$)$-collisions. Thus, in what follows, our main goal becomes to estimate the number of left collisions. The rest of the proof can be summarized in the following high-level steps. First, we define a notion of y-matrix and observe three of its properties, including the fact that a PDAF computation can be seen as a y-matrix product. Second, we define notions of q-cycles for an input y, and expected number of cycles, and relate these three notions by observing five facts. Among other things, in these facts, we relate the expected period of rpmSC1$_r$ to the number of left $(y,$PDAF$)$-collisions, and we bound the number of such collisions.

Definition and properties of y-matrices. Let x be a list in $\{0, \ldots, r-1\}^n$. For each list y in $\{0, \ldots, r-1\}^n$, define the *y-matrix* as the $n \times n$ matrix $M^{(y)}$ obtained by sequentially setting its elements as in the following 3 steps:

1. $M^{(y)}(i, j) = 0$ for all $i, j = 1, \ldots, n$;
2. $M^{(y)}(i, i) = 1$ for all $i = 1, \ldots, n$;
3. $M^{(y)}(i, j) = M^{(y)}(i, j) + 1$ for $i = 1, \ldots, n$ and $j = (i + y[i]) \bmod n$.

For every $x, y \in \{0, \ldots, r-1\}^n$, a *$(y, M)$-collision* of x is a value $x' \in \{0, \ldots, r-1\}^n$ such that $M^{(y)}(x) = M^{(y)}(x')$. As a consequence of the above definitions of y-matrix, (y, M)-collision and left $(y,$PDAF$)$-collision, we observe the following three properties:

1. each row of $M^{(y)}$ has 1 or 2 nonzero entries, one being on the diagonal, and either both nonzero entries are $= 1$ or the one nonzero entry is $= 2$;
2. the computation $z =$ PDAF (x, y) can be written as a matrix product $z = M^{(y)}(x)$; and
3. a (y, M)-collision of x is a left $(y,$PDAF$)$-collision of x.

Definition and properties of q-cycles. Let q be a non-negative integer. We say that the q distinct indices $(i_1, \ldots, i_q) \in \{1, \ldots, n\}$ are a *q-cycle* for input list $y \in \{0, \ldots, r-1\}^q$ if the following holds:

1. $i_h = i_{h-1} + y[h]$ for all $h = 2, \ldots, q$
2. $i_1 = i_q + y[q]$.

We say that the q distinct indices $(i_1, \ldots, i_q) \in \{1, \ldots, n\}$ are a *minimal q-cycle* for y if no sublist of $(i_1, \ldots, i_q)$ is a q'-cycle, for $q' < q$. We say that a tuple of indices *is a (minimal) cycle* for y if it is a (minimal) q-cycle for y, for some $q \in \{1, \ldots, n\}$. Two cycles are *disjoint* if all indices in $\{1, \ldots, n\}$ in one cycle are different from all indices in $\{1, \ldots, n\}$ in the other cycle. If y is uniformly chosen from $\{0, \ldots, r-1\}^n$, the number of q-cycles for y and the number of cycles of y are random variables, with a well-defined expectation. We finally observe the following five facts relating the above notions of cycles, collisions and period:

1. if $(i_1, \ldots, i_q)$ is a q-cycle for y, the matrix entries $M^{(y)}(i_1, i_2), \ldots, M^{(y)}(i_{q-1}, i_q), M^{(y)}(i_q, i_1)$ are $= 0$

2. if $(i_1,\ldots,i_q)$ is a minimal q-cycle for y, the rank of the $q\times n$ submatrix $S^{(y),q}$ containing the q rows indexed as $i_1,\ldots,i_q$ in $M^{(y)}$ is $= q-1$;
3. for any $t\geq 1$, if there are t disjoint cycles for y, then for each $x\in\{0,\ldots,r-1\}^n$, there exist r^t left (y,PDAF)-collisions of x;
4. the expected number of cycles for a uniformly chosen $y\in\{0,\ldots,r-1\}^n$ is $\leq 1.7\log_2 n+2$;
5. if the expected number of cycles for a uniformly chosen $y\in\{0,\ldots,r-1\}^n$ is $\leq t$ then the expected value of the period of stream cipher rpmSC1 is $\geq r^{n-t}$.

Note that Theorem 3 directly follows from above Facts 4 and 5, which, in turn, are established using Facts 1-3. Therefore, to end the proof of Theorem 3, it suffices to prove all of the above Facts 1-5.

Proof of Fact 1. The fact directly follows by the definition of q-cycle, implying that all entries $(i_1,i_2),\ldots,(i_{q-1},i_q),(i_q,i_1)$ of y-matrix $M^{(y)}$ are either set to 1 in step 2 or incremented by 1 in step 3.

Proof of Fact 2. If $(i_1,\ldots,i_q)$ is a minimal q-cycle for y, by definition of q-cycle, all indices $i_1,\ldots,i_q$ are distinct. Therefore, by definition of y-matrix, the q rows indexed as $i_1,\ldots,i_q$ in $M^{(y)}$ contain 2 entries $=1$, one such entry being on the matrix diagonal. By Fact 1, we have that the sum of all q rows is the all-zero row, and thus the rank of the y-matrix is $\leq q-1$. Because of the cycle's minimality, any $q-1$ among these q rows are linearly independent and thus the rank of the y-matrix is $=q-1$.

Proof of Fact 3. By property 2 of y-matrices, a PDAF computation can be written as $z=M^{(y)}x$, where z,x are n-component vectors with elements in $\{0,\ldots,r-1\}$, and $M^{(y)}$ is a $n\times n$ matrix with elements in $\{0,1,2\}$. Then, every disjoint cycle for y introduces a new linear dependency between a different subset of rows in $M^{(y)}$. Thus, t disjoint cycles for y imply that $M^{(y)}$ has rank $n-t$. This implies, by Fact 2 and linear independence, that for each z, there are r^t vectors x such that $z=M^{(y)}(x)$. Therefore, every x has r^t (y,M)-collisions, and, by property 3 of y-matrices, r^t left (y,PDAF)-collisions.

Proof of Fact 4. Let $p_{t,q,n}$ denote the probability of having t disjoint q-cycles in a uniformly chosen, n-element, input list y, and let $M^{(y)}$ be the associated y-matrix. We obtain that

$$p_{1,q,n}=\frac{\binom{n}{q}(q-1)!}{n^q}\quad\text{and, for } t>1,\ p_{t,q,n}=\frac{\binom{n}{tq}(q-1)!^t}{n^{tq}}\leq\frac{n^{tq}(q-1)!^t}{(tq)!n^{tq}}\leq\frac{1}{qt^q},$$

where the inequalities follows by known bounds on the involved binomial coefficient and factorials. We then obtain that the expected number of cycles in a uniformly chosen, n-element, input list y, is

$$\sum_{q=1}^{n}\sum_{t=1}^{n/q}\frac{1}{qt^q}=\sum_{q=1}^{n}\frac{1}{q}\sum_{t=1}^{n/q}\frac{1}{t^q}\leq\ln n+1+\sum_{q=2}^{n}\frac{1}{q}\sum_{t=1}^{n/q}\frac{1}{t^q}\leq\ln n+1+1.44\ln n+1=1.7\log_2 n+2,$$

where the first inequality follows from known results on Harmonic series and the second inequality follows from known bounds on the Riemann's zeta function on inputs >1 (see, e.g., [AbSt72]).

Proof of Fact 5. This follows by combining Fact 3-4 with the observation that the sequence of keys repeats at every iteration where the random oracle R returns a state that results in a left (y,PDAF)-collision of x, for any of the previous x input to PDAF.

5.2 On the Period of rpmSC2

We define an idealized version of rpmSC2, denoted as rpmSC2$_r$, by replacing the CMBN primitive function in the rpmSC2 stream cipher with a random permutation, so that the inputs to the EXTC primitive function can be considered random and independent strings.

The rpmSC2$_r$ stream cipher. Let R denote a random permutation. Formally, cipher rpmSC2$_r$ is defined starting from rpmSC2, and replacing step 3 of its next key function G, as described in Section 4, there reading as "$a = CMBN(mk_1, v)$", with the step "$a = R(v)$".

We obtain the following

Theorem 4 The expected value of the period of rpmSC2$_r$ is $\geq r^{0.316\,n}$.

Proof. Since R is a random permutation, the output of R, on input the current state s_i, has no collisions. Then, to estimate the period of rpmSC2$_r$, we observe that the next key output by rpmSC2$_r$ cycles whenever the output z at the i-th execution of EXTC is equal to the output z at the j-th execution of EXTC, for some $j < i$. We note that in these two executions of EXTC, the second input is the same value mk_1, but the first input differs and is based on different states s_i, s_j, respectively. We then define the notion of *left collision* in the output of EXTC; that is, two values a, a' such that EXTC(a, k) =EXTC(a', k), for any random (master) key k. Our estimate of the period of rpmSC2$_r$ will be directly related to the total number of possible states and the number of left collisions in EXTC. Thus, in what follows, our main goal becomes to estimate the number of left collisions.

We start by observing that, by the definition of EXTC, the output z returned by EXTC satisfies $z[h] = a[i_h]$, for $h = 0, \ldots, n-1$. Furthermore, we observe that

- $i_0 = mk_1[0]$;
- $i_1 = 1 + mk_1[0] + mk_1[1]$;
- $i_2 = 2 + mk_1[0] + mk_1[1] + mk_1[2]$; ...
- $i_h = h + mk_1[0] + \ldots + mk_1[h]$; ...
- $i_{n-1} = n - 1 + mk_1[0] + \ldots + mk_1[n-1]$.

In matrix notation, we have that $I_n = b_n + T_{n\times n}k_n$, where I_n is the index vector $(i_0, \ldots, i_{n-1})^T$, b_n is the constant vector $(0, 1, \ldots, n-1)^T$, k_n is the key vector $(mk_1[0], \ldots, mk_1[n-1])^T$, and $T_{n\times n}$ is a lower left triangular matrix with all 1's in the lower left triangle.

Because $T_{n\times n}$ is a full-rank matrix, and k_n is a vector with random and independent components, by linear independence properties, we obtain that I_n is a vector with random and independent components. This implies that the output z returned by EXTC is the concatenation of randomly chosen entries of the input array a; that is, $a[i_0], \ldots, a[i_{n-1}]$ for random $i_0, \ldots, i_{n-1}$. This fact allows us to find all left collisions in EXTC as the pairs of strings a, a' that only differ in the positions (if any) in $L = \{0, \ldots, n-1\} \setminus \{i_0, \ldots, i_{n-1}\}$. Estimating the number of such pairs is a variation of the following balls-into-bins occupancy problem that has been studied in the algorithms literature: imagine throwing n balls into n bins; what is the expected number of empty bins? Each bin remains empty with probability $p_{eb} = (1 - 1/n)^n$, which is $\leq 1/e$ for any $n \geq 1$, where $e = 2.718\ldots$ is Neper's constant. Thus, by linearity of expectation, the expected number of empty bins is $= np_{eb} \leq n/e$, which implies that the expected cardinality of set L is $= np_{eb} \leq n/e$. This, in turn, implies that the expected number of strings a' that creates a left collision together with a fixed string a is $\leq r^{n/e}$. Finally, by a birthday-type probability argument (similarly as in the proof of Theorem 3), we obtain that the expected value of the period of rpmSC2$_r$ is $\geq r^{n(1/2-1/2e)} \geq r^{0.316\,n}$.

Remark. We observe that the above technique to find left collisions continues to work, with a few minor changes, for the variant of EXTC where the output vector z is computed as the sum modulo r of an element of the input x and an element of the other input y, as described in Section 4.2. This follows from the fact that this different computation of vector z does not modify the definition of left collisions for EXTC, or the definitions of set L and quantity p_{eb}.

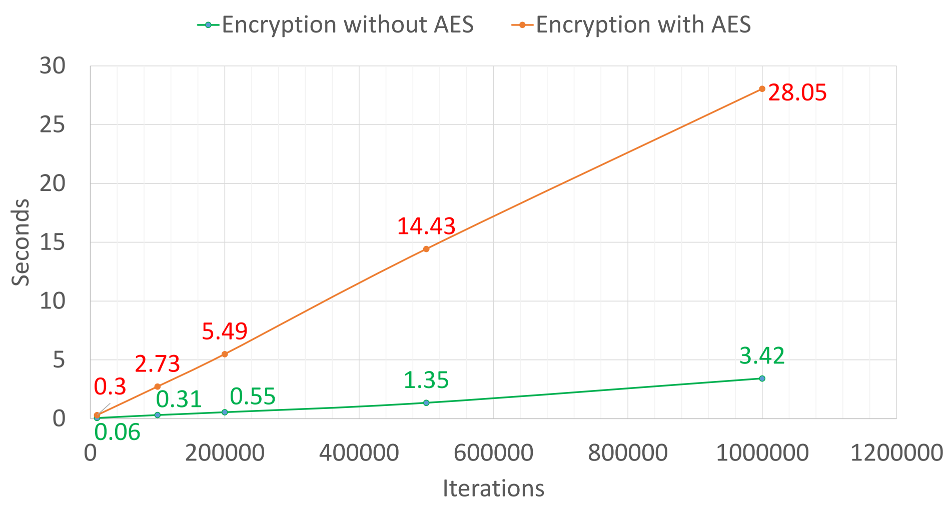

Figure 1: Comparison between encryption without AES and with AES of 1056 bits per iteration.

6 Performance Analysis

Two sets of performance tests have been performed on the techniques presented in Section 3, 4 and the associated symmetric encryption schemes.

A first set was performed in February 2010, with machine specifications: HP Pavilion with Intel Core Duo running XP 32-bit, with 1GB RAM running CentOS. These tests were performed for the stream cipher rpmSC2, The result was a performance of 5.2 seconds for $1,000,000$ iterations, each processing 1056 bits.

A second set was performed in July 2010, with machine specifications: Intel Core i3-2330M 2.2 GHz with 4GB RAM. These tests were performed for symmetric encryption based on the additive stream cipher rpmSC2, in two cases: (1) encryption is performed using modular sum between the stream cipher's next key and the message block; and (2) encryption is performed using the AES block cipher, whose input key was the stream cipher's next key. As seen in Figure 1, the performance of encryption without AES is almost one order or magnitude faster.

Empirically, it has been observed that rpmSC1 has similar performance as rpmSC2, even though it is somewhat slower.

7 Application Use Cases

One example application use case for the presented stream ciphers and their associated secure communication solutions is for securing Low Power Wide Area (LPWA) Networks with the following features:

1. utilize lightweight computing devices,
2. have minimal power requirements,
3. operate at low data rates; e.g., 1Kbps,
4. use 150 MHz to 1 GHz transmission power with unlimited duty cycles,

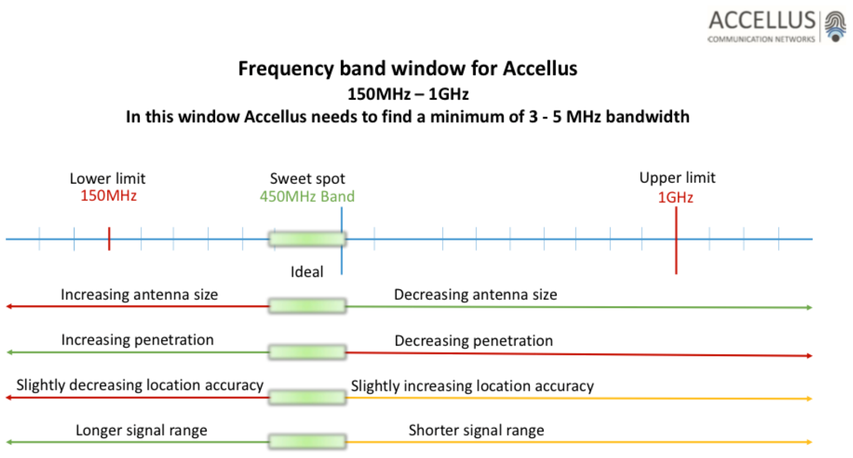

Figure 2: Frequency band window

5. can deliver Medium, High quality of service
6. have mission critical performance capability, and
7. can provide packet level security.

In Figure 2 we show the optimal bandwidth factors of Accellus LPWA [Spl14].

To identify the market for LPWA networks, AEgis Systems Limited and Machina Research [Aeg14, Mor13] have defined a set of 8 application categories, based on range, bandwidth and quality of service, but reflecting the predominantly narrow band nature of most Market-to-Market (M2M) applications.

The categories are shown in Figure 3 along with examples of typical applications in each case. Furthermore, they conclude that a sizeable proportion of the narrow band connections fall into the medium quality of service category. This is largely a reflection of the M2M market itself, which is mainly accounted for by applications in sectors such as automotive, manufacturing, smart metering and building automation, which (whilst not being mission critical in life or death terms) nevertheless may have significant financial or public policy implications, if they should not perform in the required way.

By design, the Accellus LPWA Network technology fits in groups 2, 4 and 6 (shown in orange in the figure), with the emphasis on groups 4 and 6.

8 Conclusions

Lightweight cryptographic functions are proposed in RPM for constrained environments like LPWA sensor networks, to efficiently target important security properties, including: confidentiality via encryption, implicit mutual and constant authentication, and continuous key management by key refreshing. We reported performance results showing that encryption based on some RPM stream ciphers is almost 1 order of magnitude faster than AES-based encryption. We also studied RPM primitives to prove security of idealized versions of associated stream ciphers against a certain class of attacks (i.e., attacks based on low period). With respect to the RPM stream ciphers, it remains of interest to reduce

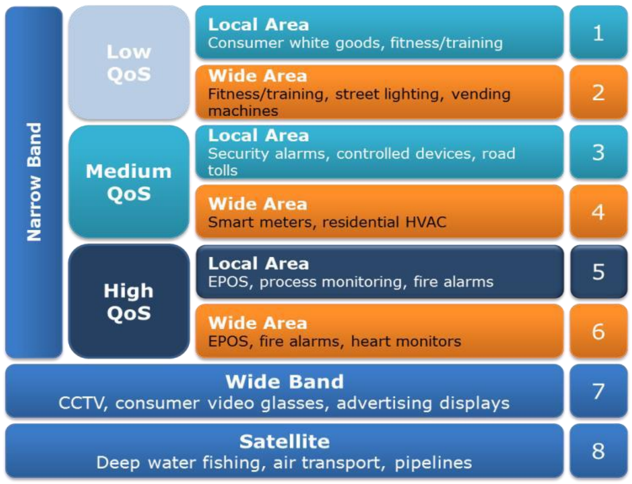
Narrow Band
Low QoS
Local Area
Consumer white goods, fitness/training
1
Wide Area
Fitness/training, street lighting, vending machines
2
Medium QoS
Local Area
Security alarms, controlled devices, road tolls
3
Wide Area
Smart meters, residential HVAC
4
High QoS
Local Area
EPOS, process monitoring, fire alarms
5
Wide Area
EPOS, fire alarms, heart monitors
6
Wide Band
CCTV, consumer video glasses, advertising displays
7
Satellite
Deep water fishing, air transport, pipelines
8

Figure 3: Application categories, based on range, bandwidth and QoS.

the idealization in the primitive analysis done in this paper or obtain similar results with respect to other classes of attacks, as well as finding successful attacks. With respect to provable security for lightweight cryptography constructions, it remains of interest to study what classes of attacks can be provably shown to be ineffective under suitable assumptions.

References

[AbSt72] Abramowitz, Milton; Stegun, Irene A., eds. (1972), Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables, New York: Dover Publications, ISBN 978-0-486-61272-0

[Adc00] Adcock, Jamison M., David M. Balenson, David W. Carman, Michael Heyman, and Alan T. Sherman, Trading off strength and performance in network authentication: Experience with the ACSA Project, Proceedings of the DARPA Information Survivability Conference & Exposition (DISCEX 00), IEEE Computer Society (Jan. 25-27, 2000), 127139.

[Aeg14] AEgis Systems Limited and Machina Research, M2M application characteristics and their implications for spectrum, (Surrey, United Kingdom, 13 May 2014)

[Bdjr97] Mihir Bellare, Anand Desai, Eron Jokipii, and Phillip Rogaway, "A Concrete Security Treatment of Symmetric Encryption", 38th Annual Symposium on Foundations of Computer Science, FOCS '97, Miami Beach, Florida, USA, October 19-22, 1997.

[BeRo93] Mihir Bellare, Phillip Rogaway: Random Oracles are Practical, "A Paradigm for Designing Efficient Protocols", ACM Conference on Computer and Communications Security 1993: 62-73

[CaGoHa04] Ran Canetti, Oded Goldreich, Shai Halevi: The random oracle methodology, revisited. J. ACM 51(4): 557-594 (2004), also in Proc. of ACM STOC 1998.

[DiC11] Di Crescenzo, Giovanni, "Cryptography Assessment of RSCorp′s Real Privacy Management (RPM) System", Telcordia Technologies, Inc., (Piscataway, NJ, 2011).

[McG99] McGough, Paul, "Method and system for performing secure electronic messaging", U.S. Patent 6,002,769, (December 14, 1999).

[McG00] McGough, Paul, “Method and system for performing secure electronic monetary transactions”, U.S. Patent 6,058,189, (May 2, 2000).

[McG02] McGough, Paul, “Method and system for performing secure electronic digital streaming”, U.S. Patent 6,445,797, (September 3, 2002).

[McG06] McGough, Paul, “Method and system for performing perfectly secure key exchange and authenticated messaging”, U.S. Patent Application 11/108,347, (February 16, 2006).

[McG11] McGough, Paul, Real Privacy Management Authentication System, U.S. Patent 7,899,185, (March 1, 2011).

[McG12] McGough, Paul, Method for obtaining key for use in secure communications over a network and apparatus for providing same, U.S. Patent 8,144,874, (March 27, 2012).

[McG12a] McGough, Paul, Method and system for establishing real-time authenticated and secured communications channels in a public network, U.S. Patent 8,144,875, (March 27, 2012).

[McG14] McGough, Paul, Method and system for establishing real-time trust in a public network, U.S. Patent 8,649,520, (February 11, 2014).

[Mor13] Morrish, Jim, The Emergence of M2M/IoT Application Platforms, Machina Research, (London, September 2013).

[Rel14] Relevant Security Corp., Real Privacy Management (RPM) Cryptographic Description, Version 3.3, (Denver, Colorado, 2014).

[Rel14b] Relevant Security Corp., Real Privacy Management (RPM) RPM Properties Description for Analysis, Version 2.2, (Denver, Colorado, 2014).

[Rel14c] Relevant Security Corp., Real Privacy Management (RPM) Recommendations, Considerations and Architectures for Initial Key Establishment (IKE), Version 2.4, (Denver, Colorado, 2014).

[Rel14d] Relevant Security Corp., Real Privacy Management (RPM) Reference Guide, Version 3.2, (Denver, Colorado, 2014).

[RiS82] Rivest, Ronald L., and Alan T. Sherman, Randomized encryption techniques in Advances in Cryptology: Proceedings of Crypto 82, David Chaum, Ronald L. Rivest, and Alan T. Sherman, eds., Plenum Press (New York, 1983), 145 - 163.

[Sh05] Sherman, Alan T., An initial assessment of the 2factor authentication and key-management system: highlights, unpublished memorandum, (May 27, 2005), 6 pages.

[ShM03] Sherman, Alan T., and David A. McGrew, David A., Key establishment in large dynamic groups using one-way function trees, IEEE Transactions on Software Engineering, 29:5 (May 2003), 444 - 458.

[Spl14] Splinter, Erik and Guido van Alphen, ”Accellus Communication Networks - Low Power Wide Area Network Technology Provider”, (Zwolle, the Netherlands, 10 December 2014)

[Tan06] Tanaka, Hatsukazu, Security-function integrated simple cipher communication system, The 2006 Symposium on Cryptography and Information Security (SCIS 2006), The Institute of Electronics, Information and Communication Engineers, (Hiroshima, Japan, January 17 - 20, 2006).

[Tan07] Tanaka, Hatsukazu, Generation of cryptographic random sequences and its application to secure enciphering, The 2007 Symposium on Cryptography and Information Security (SCIS 2007), The Institute of Electronics, Information and Communication Engineers, (Sasebo, Japan, January 23 - 26, 2007).

[Tan09] Tanaka, Hatsukazu, A New Generation Scheme of Random Number Sequences and Its Application to Enhancing the Security of DES, The Institute of Electronics, Information and Communication Engineers, IEICE Transactions, Manuscript submission (January 1, 2009).

[Tan10] Tanaka, Hatsukazu, Informationally Secure Secret-Key Sharing Scheme Using the Singularity of Source Coding, The 2007 Symposium on Cryptography and Information Security (SCIS 2010), The Institute of Electronics, Information and Communication Engineers, (Takamatsu, Japan, Jan. 19 – 22, 2010).