WG-8: A Lightweight Stream Cipher for Resource-Constrained Smart Devices

Xinxin Fan, Kalikinkar Mandal, and Guang Gong

Department of Electrical and Computer Engineering
University of Waterloo
Waterloo, Ontario, N2L 3G1, Canada
{x5fan,kmandal,ggong}@uwaterloo.ca

Abstract. Lightweight cryptographic primitives are essential for securing pervasive embedded devices like RFID tags, smart cards, and wireless sensor nodes. In this paper, we present a lightweight stream cipher WG-8, which is tailored from the well-known Welch-Gong (WG) stream cipher family, for resource-constrained devices. WG-8 inherits the good randomness and cryptographic properties of the WG stream cipher family and is resistant to the most common attacks against stream ciphers. The software implementations of the WG-8 stream cipher on two popular low-power microcontrollers as well as the extensive comparison with other lightweight cryptography implementations highlight that in the context of securing lightweight embedded applications WG-8 has favorable performance and low energy consumption.

Keywords: Lightweight stream cipher, resource-constrained device, cryptanalysis, efficient implementation.

1 Introduction

The Internet of Things (IoT) is an emerging computing and communication paradigm in which smart devices (e.g., RFID tags, smart cards, wireless sensor nodes, etc.) are linked through both wired and wireless networks to the Internet. Those smart devices interact and cooperate with each other to conduct complicated tasks such as sensing the environment, interpreting the data, and responding to events. While the IoT provides new and exciting experience for end users, it also opens up new avenues to hackers and organized crime. Recent attacks to a wide range of smart devices [13, 39] have emphasized that without adequate security the IoT will only become pervasive nightmare.

The challenges for deploying security solutions for smart devices are threefold: 1) The overhead (i.e., the gate count in hardware or the memory footprint in software) of security solutions should be minimal due to the low-cost nature of smart devices; 2) The power consumption of security solutions should be minimal due to the low-power characteristic of smart devices; and 3) The performance of security solutions should be reasonable to support applications and end-user requirements. To address the aforementioned challenges for securing smart devices, a new research direction called *lightweight cryptography* has

K. Singh, A.K. Awasthi, and R. Mishra (Eds.): QSHINE 2013, LNICST 115, pp. 617–632, 2013.

been established which focuses on designing novel cryptographic algorithms and protocols tailored for implementation in resource-constrained environments.

A host of lightweight symmetric ciphers that particularly target for resource-constrained smart devices have been proposed in the past few years. Early work focuses on optimizing hardware implementations of standardized block ciphers such as AES [17], IDEA [25] and XTEA [22]. Later on, researchers have shown how to modify a classical block cipher like DES [24] for lightweight applications. Recent proposals deal with new low-cost designs, including lightweight block ciphers PRESENT [5], KATAN/KTANTAN [6], PRINTcipher [23], LED [20], and Piccolo [36], lightweight stream ciphers Grain [21], Trivium [7], and MICKEY [3], as well as a lightweight hybrid cipher Hummingbird/Hummingbird-2 [15, 16]. A good research survey about recently published lightweight cryptographic implementations can be found in [14].

In this paper we present the stream cipher WG-8, which is a lightweight variant of the well-known WG stream cipher family [29] as submitted to the eSTREAM project. WG-8 inherits good randomness properties of the WG stream cipher family such as period, balance, ideal two-level autocorrelation, ideal tuple distribution, and exact linear complexity. Moreover, WG-8 is able to resist the most common attacks against stream ciphers including algebraic attack, correlation attack, differential attack, cube attack, distinguish attack, discrete fourier transform attack, and time-memory-data tradeoff attack, thereby providing adequate security for lightweight embedded applications.

We also propose several techniques for efficient implementation of the stream cipher WG-8 on two low-power microcontrollers, including an 8-bit microcontroller ATmega128L from Atmel and a 16-bit microcontroller MSP430 from Texas Instruments. Our experimental results show that WG-8 can achieve high throughput of 185.5 Kbits/s and 95.9 Kbits/s on the above two microcontrollers with energy efficiency of 458 nJ/bit and 125 nJ/bit, respectively. When compared to other lightweight cryptography implementations in the literature, the throughput of the WG-8 is about $2 \sim 15$ times higher and the energy consumption is around $2 \sim 220$ times smaller than those of most previous ciphers.

The remainder of this paper is organized as follows. Section 2 gives a description of the lightweight stream cipher WG-8. Subsequently, in Section 3 we analyze the security of the WG-8 against the most common attacks to stream ciphers. Section 4 describes efficient techniques for implementing the WG-8 stream cipher on low-power microcontrollers and reports our experimental results and comparisons with previous work. Finally, Section 5 concludes this contribution.

2 The Lightweight Stream Cipher WG-8

2.1 Preliminaries

We define the terms and notations that will be used to describe the lightweight stream cipher WG-8 and its architecture as well as to characterize its randomness and cryptographic properties.

- $\mathbb{F}_2 = \{0, 1\}$, the Galois field with two elements 0 and 1.
- $p(x) = x^8 + x^4 + x^3 + x^2 + 1$, a primitive polynomial of degree 8 over $\mathbb{F}_2$.
- $\mathbb{F}_{2^8}$, the extension field of $\mathbb{F}_2$ defined by the primitive polynomial $p(x)$ with 2^8 elements. Each element in $\mathbb{F}_{2^8}$ is represented as an 8-bit binary vector. Let ω be a primitive element of $\mathbb{F}_{2^8}$ such that $p(\omega) = 0$.
- $\mathrm{Tr}(x) = x + x^2 + x^{2^2} + \cdots + x^{2^7}$, the trace function from $\mathbb{F}_{2^8} \mapsto \mathbb{F}_2$.
- $l(x) = x^{20} + x^9 + x^8 + x^7 + x^4 + x^3 + x^2 + x + \omega$, the feedback polynomial of LFSR (which is also a primitive polynomial over $\mathbb{F}_{2^8}$).
- $q(x) = x + x^{2^3+1} + x^{2^6+2^3+1} + x^{2^6-2^3+1} + x^{2^6+2^3-1}$, a permutation polynomial over $\mathbb{F}_{2^8}$.
- $\mathsf{WGP\text{-}8}(x^d) = q(x^d + 1) + 1$, the WG-8 permutation with decimation d from $\mathbb{F}_{2^8} \mapsto \mathbb{F}_{2^8}$, where d is coprime to $2^8 - 1$.
- $\mathsf{WGT\text{-}8}(x^d) = \mathrm{Tr}(\mathsf{WGP\text{-}8}(x^d)) = \mathrm{Tr}(x^9 + x^{37} + x^{53} + x^{63} + x^{127})$, the WG-8 transformation with decimation d from $\mathbb{F}_{2^8} \to \mathbb{F}_2$, where d is coprime to $2^8 - 1$.
- Polynomial basis (PB) of $\mathbb{F}_{2^8}$: A polynomial basis of $\mathbb{F}_{2^8}$ over $\mathbb{F}_2$ is a basis of the form $\{1, \omega, \omega^2, \cdots, \omega^7\}$.
- Normal basis (NB) of $\mathbb{F}_{2^8}$: A normal basis of $\mathbb{F}_{2^8}$ over $\mathbb{F}_2$ is a basis of the form $\{\theta, \theta^2, \cdots, \theta^{2^7}\}$, where $\theta = \omega^5$ (i.e., a normal element) is used in this work.
- Autocorrelation: The autocorrelation of a binary sequence with period T is defined as the difference between the agreements and disagreements when the symbol 0 maps to 1 and 1 maps to -1. If all the out-of-phase autocorrelation is equal to -1, then the sequence is said to have *ideal two-level autocorrelation*.
- Linear span (LS): The linear span or linear complexity of a binary sequence is defined as the length of the smallest linear feedback shift register (LFSR) which generates the entire binary sequence.
- Nonlinearity: The nonlinearity of a function f is defined as the minimum distance from f to any affine function with the same number of variables.
- Algebraic immunity (AI): The algebraic immunity of a function f is defined as the minimum degree of an annihilator Boolean function g such that g is equivalent to either f or the complement of f (i.e., $fg = 0$ or $(f + 1)g = 0$). In the ideal case, the algebraic immunity of a function f is equal to the degree of f, thus making it immune to algebraic attacks.
- $\oplus$, the bitwise addition operator (i.e., XOR).
- $\otimes$, the multiplication operator over $\mathbb{F}_{2^8}$.

2.2 The Description of the Stream Cipher WG-8

WG-8 is a lightweight variant of the well-known Welch-Gong (WG) stream cipher family with 80-bit secret key and 80-bit initial vector (IV), which can be regarded as a nonlinear filter generator over finite field $\mathbb{F}_{2^8}$. The stream cipher WG-8 consists of a 20-stage LFSR with the feedback polynomial $l(x)$ followed by a WG-8 transformation module with decimation $d = 19$, and operates in two phases, namely an initialization phase and a running phase.

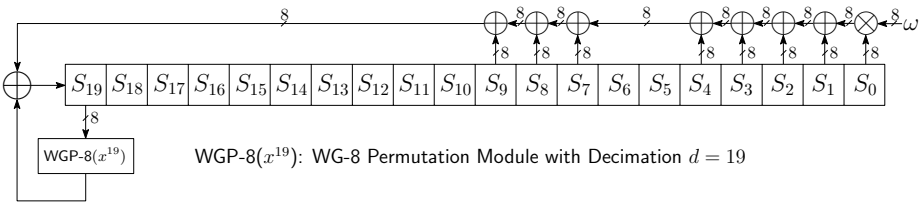

Fig. 1. The Initialization Phase of the Stream Cipher WG-8

Initialization Phase. The key/IV initialization phase of the stream cipher WG-8 is shown in Fig. 1.

Let the 80-bit secret key be $K = (K_{79}, \ldots, K_0)_2$, the 80-bit IV be $IV = (IV_{79}, \ldots, IV_0)_2$, and the internal state of the LFSR be $S_0, \ldots, S_{19} \in \mathbb{F}_{2^8}$, where $S_i = (S_{i,7}, \ldots, S_{i,0})_2$ for $i = 0, \ldots, 19$. The key and IV initialization process is conducted as follows: $S_{2i} = (K_{8i+3}, \ldots, K_{8i}, IV_{8i+3}, \ldots, IV_{8i})_2$ and $S_{2i+1} = (K_{8i+7}, \ldots, K_{8i+4}, IV_{8i+7}, \ldots, IV_{8i+4})_2$ for $i = 0, \ldots, 9$.

Once the LFSR is loaded with the key and IV, the apparatus runs for 40 clock cycles. During each clock cycle, the 8-bit internal state S_{19} passes through the nonlinear WG-8 permutation with decimation $d = 19$ (i.e., the WGP-8(x^{19}) module) and the output is used as the feedback to update the internal state of the LFSR. The LFSR update follows the recursive relation:

$$S_{k+20} = (\omega \otimes S_k) \oplus S_{k+1} \oplus S_{k+2} \oplus S_{k+3} \oplus S_{k+4} \oplus S_{k+7} \oplus S_{k+8} \oplus S_{k+9} \oplus \text{WGP-8}(S_{k+19}^{19}), \quad 0 \le k < 40.$$

After the key/IV initialization phase, the stream cipher WG-8 goes into the running phase and 1-bit keystream is generated after each clock cycle.

Running Phase. The running phase of the stream cipher WG-8 is illustrated in Fig. 2. During the running phase, the 8-bit internal state S_{19} passes through the nonlinear WG-8 transformation with decimation $d = 19$ (i.e., the WGT-8(x^{19}) module) and the output is the keystream. Note that the only feedback in the running phase is within the LFSR and the recursive relation for updating the LFSR is given below:

$$S_{k+20} = (\omega \otimes S_k) \oplus S_{k+1} \oplus S_{k+2} \oplus S_{k+3} \oplus S_{k+4} \oplus S_{k+7} \oplus S_{k+8} \oplus S_{k+9}, k \ge 40.$$

The WG-8 transformation module WGT-8(x^{19}) comprises of two sub-modules: a WG-8 permutation module WGP-8(x^{19}) followed by a trace computation module $\mathrm{Tr}(\cdot)$. While the WGP-8(x^{19}) module permutes elements over $\mathbb{F}_{2^8}$, the $\mathrm{Tr}(\cdot)$ module compresses an 8-bit input to 1-bit keystream.

2.3 Randomness Properties of the WG-8 Keystream

The keystream generated by the stream cipher WG-8 has the following desired randomness properties [8]:

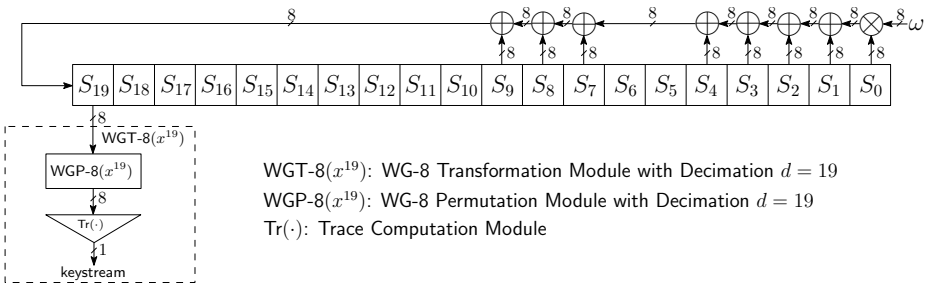

Fig. 2. The Running Phase of the Stream Cipher WG-8

1. The keystream has a period of $2^{160} - 1$.
2. The keystream is balanced, i.e., the number of 0's is only one less than the number of 1's in one period of the keystream.
3. The keystream is an ideal two-level autocorrelation sequence.
4. The keystream has an ideal t-tuple ($1 \leq t \leq 20$) distribution, i.e., every possible output t-tuple is equally likely to occur in one period of the keystream.
5. The linear span of the keystream can be determined exactly, which is $2^{33.32}$.

3 Cryptanalysis of the Stream Cipher WG-8

In this section, we analyze the security of the stream cipher WG-8 under the context of lightweight embedded applications.

3.1 Algebraic Attack

The algebraic attack is a powerful attack against LFSR based filtering sequence generators [11]. The goal of the algebraic attack is to form a lower degree multivariate equation by multiplying the filtering function by a low-degree multivariate polynomial. This gives an overdefined system of nonlinear equations for sufficiently many keystreams, which can be solved to recover the internal state of the LFSR. The algebraic immunity of the WGT-8(x^{19}) is equal to 4. According to the algebraic attack, the time complexity and the data complexity for recovering the internal state of the LFSR are about $\frac{7}{64} \cdot \binom{160}{4}^{\log_2 7} = 2^{66.0037}$ and $\binom{160}{4} = 2^{24.65}$, respectively. For applying the fast algebraic attacks [10] to the stream cipher WG-8, one needs to respectively find two multivariate polynomials g and h of degree e and d ($e < d$) such that $f \cdot g = h$. For the WGT-8(x^{19}) and $e = 1$, there does not exist a multivariate polynomial h in 8 variables with degree less than 7. Hence, in order to launch the fast algebraic attack one needs to obtain more keystream bits with a higher complexity. For lightweight embedded applications, it is hard for an attacker to obtain about $2^{24.65}$ keystream bits. Even if the attacker can get those many bits for a fixed key and IV, he needs to perform the operations with the time complexity $2^{66.0037}$, which completely defeats this attack.

3.2 Correlation Attack

In the correlation attack, the objective of an attacker is either to find the correlation between a keystream and an output sequence of an LFSR or to find the correlation among the keystreams [9,27,37]. The stream cipher WG-8 is secure against the correlation among the keystreams as it produces keystreams with 2-level autocorrelation. We now consider the fast correlation attack in which the keystream of the stream cipher is considered as a distorted version of the LFSR output. In the fast correlation attack, the linear approximation of WGT-8(x^{19}) can be used to derive a generator matrix of a linear code that can be decoded by a maximum likelihood decoding (MLD) algorithm. Letting $f(x)$ be a linear function in 8 variables, we have $\Pr(\text{WGT-8}(x^{19})(x) = f(x)) = \frac{(2^8-108)}{2^8} = 0.578125$. Applying the results of [9] for $t = 3$, the amount of keystream (denoted by N) required for the attack to be successful is given by $N \approx (k \cdot 12 \cdot \ln 2)^{\frac{1}{3}} \cdot \epsilon^{-2} \cdot 2^{\frac{160-k}{3}}$ and the decoding complexity is given by $C_{dec} = 2^k \cdot k \cdot \frac{2 \ln 2}{(2\epsilon)^6}$, where $\epsilon = (\Pr(\text{WGT-8}(x^{19}) = f(x)) - 0.5) = 0.078125$ and k is the number of LFSR internal state bits recovered. If we choose a small value of k (e.g., $k = 7$), the number of bits required to launch the attack is about $2^{60.31}$, which is not possible in practice. Similarly, if we choose a large value of k (e.g., $k = 80$), the number of bits required to mount the attack is about $2^{37.15}$. However, the decoding complexity of the attack is approximately $2^{102.68}$, which is worse than the exhaustive search. Hence, the stream cipher WG-8 is secure against the fast correlation attack.

3.3 Differential Attack

The initialization phase in the first design of the WG stream cipher was vulnerable to the chosen IV attack [40], where an attacker can distinguish several output bits by building a distinguisher based on the differential cryptanalysis. This weakness has been fixed in the later design by placing the WG permutation module at the last position of the LFSR [29]. For the proposed stream cipher WG-8, the differential distribution of the WGP-8(x^{19}) is 8-uniform, which provides a maximum 2^{-5} possibility for differential characteristic. During the initialization phase the WGP-8(x^{19}) is applied for 40 times. Thus, after the initialization phase, it would be quite hard for an attacker to distinguish the output keystream since the differentials become complex and contain most key/IV bits.

3.4 Cube Attack

Cube attack [12] is a generic key-recovery attack that can be applied to any cryptosystem, provided that the attacker can obtain a bit of information that can be represented by a low-degree decomposition multivariate polynomial in Algebraic Normal Form (ANF) of the secret and public variables of the target cryptosystem. Note that the nonlinearity of WGP-8(x^{19}) is 92 and the algebraic degrees of the component functions of WGP-8(x^{19}) are 7. Moreover, the ANF representations of 8 component functions contain 133, 113, 146, 124, 137, 109,

122, and 120 terms, respectively, and only the ANF of the second component contains 7 linear terms and other terms are of degree greater than or equal to 2. In the WG-8 stream cipher, after 40 rounds of the initialization phase, the degree of the output polynomial can be very high. As a result, it would be hard for an attacker to collect low-degree relations among the secret key bits.

3.5 Distinguishing Attack

Recently, a distinguishing attack has been proposed against the stream cipher WG-7 [30]. Due to the small number of tap positions in the LFSR of the WG-7, the characteristic polynomial of the LFSR allows an attacker to build a distinguisher for distinguishing a keystream generated by WG-7 from a truly random keystream. For the WG-8 cipher, the characteristic polynomial of the LFSR consists of 8 tap positions and a similar distinguisher as in [30] can be built as

$$F(S_i, ..., S_{i+4}, S_{i+7}, ..., S_{i+9}) = \mathsf{WGT\text{-}8}(\omega \otimes S_i \oplus S_{i+1} \oplus S_{i+2} \oplus S_{i+3} \oplus S_{i+4} \oplus S_{i+7} \oplus S_{i+8} \oplus S_{i+9}) \oplus \mathsf{WGT\text{-}8}(S_i) \oplus \mathsf{WGT\text{-}8}(S_{i+1}) \oplus \mathsf{WGT\text{-}8}(S_{i+2}) \oplus \mathsf{WGT\text{-}8}(S_{i+3}) \oplus \mathsf{WGT\text{-}8}(S_{i+4}) \oplus \mathsf{WGT\text{-}8}(S_{i+7}) \oplus \mathsf{WGT\text{-}8}(S_{i+8}) \oplus \mathsf{WGT\text{-}8}(S_{i+9}),$$

which is a Boolean function in 64 variables. For the distinguisher F, the probability $\Pr(F(x) = 0) = \frac{1}{2} \pm \epsilon$, where $x = (a_0, ..., a_7), a_i \in \mathbb{F}_{2^8}$. Note that the value of ϵ will be quite small due to a huge number of variables in the distinguisher, which requires an attacker to obtain more keystream bits for distinguishing the keystream. However, the computation of the exact value of ϵ is infeasible in this case because the number of possible values of x is 2^{64}. Hence the WG-8 stream cipher is resistant to the distinguishing attack. Note that this type of distinguishing attacks can also be extended to the case in which a distinguisher can be built using a linear relation of a remote term of the LFSR, say S_τ for not large τ, and the sequences addressed in a subset of tap positions of the LFSR, denoted by $I = \{i_1, \cdots, i_t\} \subset \{0, 1, \cdots, 19\}$. In other words, a distinguisher could be built using the linear relation $S_\tau = S_{i_1} + \cdots + S_{i_t}$. Since this property is controlled by the characteristic polynomial of the LFSR, it can be easily teared done by a proper selection of the characteristic polynomial of the LFSR. For our selection of the characteristic polynomial $l(x)$, there is no remote term S_τ for $20 \leq \tau \leq 2^{34}$ for which the size of set I is less than 5. Thus, the WG-8 stream cipher is also resistant to this general distinguishing attack.

3.6 Discrete Fourier Transform Attack

The Discrete Fourier Transform (DFT) attack is a new type of attack to recover the internal state of a filtering generator, which was first proposed by Rønjom and Helleseth in [34] and extended to attacking filtering generators over $\mathbb{F}_{2^n}$ by Gong *et al.* in [19]. For mounting the DFT attack against the WG-8 stream cipher, an attacker needs to obtain $2^{33.32}$ (i.e., the linear complexity) consecutive keystream bits. Hence, the online complexity of this attack for recovering the internal state is $2^{33.32}$, after an offline computation with complexity $2^{48.49}$. For

typical lightweight embedded applications like RFID systems, a reader and a tag only exchange 32-bit random numbers in each communication session. Hence, an attacker can never obtain $2^{33.32}$ consecutive keystream bits.

3.7 Time-Memory-Data Tradeoff Attack

The Time-Memory-Data (TMD) tradeoff attack [4] is a generic cryptanalytic attack that is applicable to any stream cipher, especially those with low sampling resistance. The complexity of the TMD tradeoff attack is $O(2^{\frac{n}{2}})$, where n is the size of the internal state. For the WG-8 stream cipher, the size of the internal state is 160-bit and thus the complexity of launching a TMD attack is at least 2^{80}. Moreover, the sampling resistance of the WG-8 stream cipher is high due to the usage of the WGT-8(x^{19}) as the filtering function. The ANF representation of the WGT-8(x^{19}) contains 109 terms, among which only four terms are linear and other terms have degree greater than 2 and less than 8. Hence, only by fixing 7 out of 8 variables can one obtain a linear equation.

4 Efficient Implementation of the Stream Cipher WG-8

In this section, we address efficient implementation of the WG-8 cipher on low-power microcontrollers. For each platform we provide three implementation variants that deal with trade-offs among speed, code size, and energy consumption.

4.1 Implementation of the WG-8 Permutation Module WGP-8(x^{19})

The most complicated WGP-8(x^{19}) module can be implemented using three different methods: a) a 256-byte direct look-up table; b) a 34-byte coset leader based look-up table; or c) tower field (TF) arithmetic.

Directly Look-up Table (DLT) Approach. Depending on the bases used, one can precompute the WG-8 permutation with decimation $d = 19$ by

$$\text{WGP-8}(x^{19}) = q(x^{19} + 1) + 1$$

for all elements $x \in \mathbb{F}_{2^8}$. Hence, a 256-byte look-up table $T_{\text{WGP-8}}$ can be generated to compute WGP-8(x^{19}).

Coset Leader Based Look-up Table (CLT) Approach. This approach assumes that a normal basis is used to represent elements in $\mathbb{F}_{2^8}$ and uses the essential property of the WG-8 permutation with decimation d below:

$$\begin{aligned} &\text{WGP-8}\left((x^{2^i})^d\right) = q\left((x^{2^i})^d + 1\right) + 1 = q\left((x^d)^{2^i} + 1\right) + 1 \\ &= \left(q(x^d + 1)\right)^{2^i} + 1 = \left(q(x^d + 1) + 1\right)^{2^i} = \left(\text{WGP-8}(x^d)\right)^{2^i} \end{aligned} \quad (1)$$

for $x \in \mathbb{F}_{2^8}$ and $i = 0, 1, \ldots, 7$. According to the Equation (1), if we know the WG-8 permutation WGP-8(x^d) for an element $x \in \mathbb{F}_{2^8}$, we can easily obtain the WG-8 permutation WGP-8($(x^{2^i})^d$) for the entire conset $\{x^2, x^{2^2}, \ldots, x^{2^7}\}$ of x by cyclically shifting WGP-8(x^d) to the right by i positions, provided that a normal basis is employed to represent finite field elements. The complete cosets and coset leaders of $\mathbb{F}_{2^8}$ (in hexadecimal notation) are shown in Table 1. We note that under the normal basis representation the elements in $\mathbb{F}_{2^8}$ have been grouped into 34 different cosets except for 0 and 1. Since WGP-8(0) = 0x00 and WGP-8(1) = 0xFF, we only need to generate a 34-byte look-up table $T_{\text{Co-WGP-8}}$ for storing the WG-8 permutation results for each coset leader. Here we present the following Algorithm 1 that uses the table $T_{\text{Co-WGP-8}}$ to compute WGP-8(x^d) for any $x \in \mathbb{F}_{2^8}$.

Table 1. The Cosets and Coset Leaders of $\mathbb{F}_{2^8}$

Coset Leader	Coset							Coset Leader	Coset						
0x00	–	–	–	–	–	–	–	0x27	0x4E	0x9C	0x39	0x72	0xE4	0xC9	0x93
0x01	0x02	0x04	0x08	0x10	0x20	0x40	0x40	0x2B	0x56	0xAC	0x59	0xB2	0x65	0xCA	0x95
0x03	0x06	0x0C	0x18	0x30	0x60	0xC0	0x81	0x2D	0x5A	0xB4	0x69	0xD2	0xA5	0x4B	0x96
0x05	0x0A	0x14	0x28	0x50	0xA0	0x41	0x82	0x2F	0x5E	0xBC	0x79	0xF2	0xE5	0xCB	0x97
0x07	0x0E	0x1C	0x38	0x70	0xE0	0xC1	0x83	0x33	0x66	0xCC	0x99	–	–	–	–
0x09	0x12	0x24	0x48	0x90	0x21	0x42	0x84	0x35	0x6A	0xD4	0xA9	0x53	0xA6	0x4D	0x9A
0x0B	0x16	0x2C	0x58	0xB0	0x61	0xC2	0x85	0x37	0x6E	0xDC	0xB9	0x73	0xE6	0xCD	0x9B
0x0D	0x1A	0x34	0x68	0xD0	0xA1	0x43	0x86	0x3B	0x76	0xEC	0xD9	0xB3	0x67	0xCE	0x9D
0x0F	0x1E	0x3C	0x78	0xF0	0xE1	0xC3	0x87	0x3D	0x74	0xF4	0xE9	0xD3	0xA7	0x4F	0x9E
0x11	0x22	0x44	0x88	–	–	–	–	0x3F	0x7E	0xFC	0xF9	0xF3	0xE7	0xCF	0x9F
0x13	0x26	0x4C	0x98	0x31	0x62	0xC4	0x89	0x55	0xAA	–	–	–	–	–	–
0x15	0x2A	0x54	0xA8	0x51	0xA2	0x45	0x8A	0x57	0xAE	0x5D	0xBA	0x75	0xEA	0xD5	0xAB
0x17	0x2E	0x5C	0xB8	0x71	0xE2	0xC5	0x8B	0x5B	0xB6	0x6D	0xDA	0xB5	0x6B	0xD6	0xAD
0x19	0x23	0x64	0xC8	0x91	0x23	0x46	0x8C	0x5F	0xBE	0x7D	0xFA	0xF5	0xEB	0xD7	0xAF
0x1B	0x36	0x6C	0xD8	0xB1	0x63	0xC6	0x8D	0x6F	0xDE	0xBD	0x7B	0xF6	0xED	0xDB	0xB7
0x1D	0x3A	0x74	0xE8	0xD1	0xA3	0x47	0x8E	0x77	0xEE	0xDD	0xBB	–	–	–	–
0x1F	0x3E	0x7C	0xF8	0xF1	0xE3	0xC7	0x8F	0x7F	0xFE	0xFD	0xFB	0xF7	0xEF	0xDF	0xBF
0x25	0x4A	0x94	0x29	0x52	0xA4	0x49	0x92	0xFF	–	–	–	–	–	–	–

Algorithm 1. Coset Leader Based Look-up Table Approach

Input: $x \in F_{2^8}$, a decimation d, a look-up table $T_{\text{Co-WGP-8}}$
Output: WGP-8(x^d)

1: **if** $x =$ 0x00 or $x =$ 0xFF **then**
2: **return** x
3: **end if**
4: Find the coset leader x_c of x by cyclically shifting x to the right by i positions, where $0 \leq i \leq 7$ (i.e., x_c is the smallest odd integer in the coset containing x.)
5: Find the position j of x_c being in the table $T_{\text{Co-WGP-8}}$
6: $a \leftarrow T_{\text{Co-WGP-8}}[j]$
7: **return** $a \lll i$

Tower Field Arithmetic (TFA) Based Approach. The software implementation of the WGP-8(x^{19}) module involves the arithmetic (i.e., addition, multiplication, and exponentiation) over finite field $\mathbb{F}_{2^8}$. Although we can directly

implement all the operations over $\mathbb{F}_{2^8}$, it is well known that using the isomorphic tower constructions of $\mathbb{F}_{2^8}$ might save the memory consumption. Therefore, we investigate the tower construction $\mathbb{F}_{(2^4)^2}$ in this work.

Tower Construction $\mathbb{F}_{(2^4)^2}$ and Its Arithmetic. To obtain the tower construction $\mathbb{F}_{(2^4)^2}$, we first construct $\mathbb{F}_{2^4}$ by using an irreducible polynomial $e(X)$ of degree 4 over $\mathbb{F}_2$, and then construct $\mathbb{F}_{(2^4)^2}$ by using a certain irreducible polynomial $f(X)$ of degree 2 over $\mathbb{F}_{2^4}$. In our tower construction, we use $e(X) = X^4 + X^3 + 1$ with its polynomial basis $\{1, \alpha, \alpha^2, \alpha^3\}$ for $\mathbb{F}_{2^4}$ and $f(X) = X^2 + X + \alpha$ with its normal basis $\{\beta, \beta^{16}\}$ for $\mathbb{F}_{(2^4)^2}$, where $\alpha = \omega^{119} \in \mathbb{F}_{2^4}$ and $\beta = \omega^7 \in \mathbb{F}_{(2^4)^2}$ are zeros of the polynomials $e(X)$ and $f(X)$, respectively.

Arithmetic operations in $\mathbb{F}_{2^4}$. The arithmetic in $\mathbb{F}_{2^4}$ is conducted with the aid of a 4×4 exponentiation table T_{exp} and a 4×4 logarithm table T_{log}. While the table T_{exp} stores exponentiation $\alpha^i, i = 0, 1, \ldots, 14$, the table T_{log} keeps the exponent i for each $\alpha^i, i = 0, 1, \ldots, 14$. Let $A = a_0 + a_1\alpha + a_2\alpha^2 + a_3\alpha^3$ and $B = b_0 + b_1\alpha + b_2\alpha^2 + b_3\alpha^3$ be two non-zero elements in $\mathbb{F}_{2^4}$, where $a_i, b_i \in \mathbb{F}_2, i = 0, 1, 2, 3$. We can perform the arithmetic in $\mathbb{F}_{2^4}$ as follows:

$$\begin{aligned} AB &= T_{exp}[(T_{log}[(a_0, a_1, a_2, a_3)] + T_{log}[(b_0, b_1, b_2, b_3)]) \bmod 15], \\ A^2 &= T_{exp}[(T_{log}[(a_0, a_1, a_2, a_3)] \ll 1) \bmod 15], \\ \alpha A &= T_{exp}[(T_{log}[(a_0, a_1, a_2, a_3)] + 1) \bmod 15]. \end{aligned}$$

Arithmetic operations in $\mathbb{F}_{(2^4)^2}$. Let $A = a_0\beta + a_1\beta^{16}$ and $B = b_0\beta + b_1\beta^{16}$, where $a_0, a_1, b_0, b_1 \in \mathbb{F}_{2^4}$. A multiplication AB in $\mathbb{F}_{(2^4)^2}$ is computed as follows:

$$AB = (a_0\beta + a_1\beta^{16})(b_0\beta + b_1\beta^{16}) = (c\alpha \oplus a_0b_0)\beta + (c\alpha \oplus a_1b_1)\beta^{16},$$

where $c = (a_0 \oplus a_1)(b_0 \oplus b_1)$. For a non-zero element $A \in \mathbb{F}_{(2^4)^2}$, the squaring of A is calculated as follows:

$$A^2 = (a_0\beta + a_1\beta^{16})^2 = [(a_0 \oplus a_1)^2\alpha \oplus a_0^2]\beta + [(a_0 \oplus a_1)^2\alpha \oplus a_1^2]\beta^{16}.$$

The Frobenius mapping of A with respect to $\mathbb{F}_{2^4}$, which is the 16^{th} power operation, is computed as follows:

$$A^{2^4} = (a_0\beta + a_1\beta^{16})^{16} = a_0\beta^{16} + a_1\beta^{256} = a_1\beta + a_0\beta^{16}.$$

Implementation of WGP-8(x^{19}) Module. For an element $x \in \mathbb{F}_{2^8}$, the WGP-8(x^{19}) can be computed as follows:

$$\text{WGP-8}(x^{19}) = q(x^{19} + 1) + 1 = y + y^{2^3+1} + y^{2^6}(y^{2^3+1} + y^{2^3-1}) + y^{2^3(2^3-1)+1} + 1,$$

where $y = x^{19} + 1 = x^{2^4} \cdot x^2 \cdot x + 1$. Note that for the tower construction $\mathbb{F}_{(2^4)^2}$, 1 can be denoted by the vector $(1, 0, 0, 0, 1, 0, 0, 0)$. Therefore, the addition with 1 under the TF representation is equivalent to XORing with a constant 0x88.

Table 2. Trace Computation of an Element $x \in F_{2^8}$ Using Different Bases

Basis	Element Representation	Tr(x)
Polynomial Basis (PB)	$x_0 + x_1\omega + \cdots + x_7\omega^7$	x_5
Normal Basis (NB)	$x_0\theta + x_1\theta^2 + \cdots + x_7\theta^{2^7}$	$\bigoplus_{i=0}^{7} x_i$
Tower Field (TF)	$(x_0 + x_1\alpha + x_2\alpha^2 + x_3\alpha^3)\beta +$ $(x_4 + x_5\alpha + x_6\alpha^2 + x_7\alpha^3)\beta^{16}$	$x_1 \oplus x_2 \oplus x_3 \oplus x_5 \oplus x_6 \oplus x_7$

4.2 Implementation of the Trace Computation Module Tr(·)

Depending on the bases chosen, the trace of an element $x \in F_{2^8}$ can be computed as shown in Table 4.2.

4.3 Implementation of the Multiplication by ω Module

The multiplication by ω module can be implemented using either finite field arithmetic or an 8×8 look-up table.

Multiplication by ω Using Finite Field Arithmetic. We consider the following three cases when the PB, NB, and TF are used to represent finite field elements, respectively. With the PB representation, the multiplication of an element $x \in \mathbb{F}_{2^8}$ by ω can be computed as follows:

$$\begin{aligned} x \cdot \omega &= x_0\omega + x_1\omega^2 + \cdots + x_6\omega^7 + x_7\omega^8 \\ &= x_7 + x_0\omega + (x_1 \oplus x_7)\omega^2 + (x_2 \oplus x_7)\omega^3 + \\ &\quad (x_3 \oplus x_7)\omega^4 + x_4\omega^5 + x_5\omega^6 + x_6\omega^7. \end{aligned} \tag{2}$$

Therefore, the result of $x \cdot \omega$ is represented as an 8-bit vector $(x_7, x_0, x_1 \oplus x_7, x_2 \oplus x_7, x_3 \oplus x_7, x_4, x_5, x_6)$ with respect the PB.

With the NB representation, the multiplication of an element $x \in \mathbb{F}_{2^8}$ by ω can be calculated as follows:

$$x \cdot \omega = (x_0\theta + x_1\theta^2 + \cdots + x_6\theta^{2^6} + x_7\theta^{2^7}) \cdot \omega = \mathbf{M} \cdot (x_0, x_1, \cdots, x_6, x_7)^T, \tag{3}$$

where the matrix $\mathbf{M}$ is given below.

With the TF representation, the multiplication of an element $x \in \mathbb{F}_{2^8}$ by ω can be calculated as follows:

$$\begin{aligned} x \cdot \omega &= [(x_0 + x_1\alpha + x_2\alpha^2 + x_3\alpha^3)\beta + (x_4 + x_5\alpha + x_6\alpha^2 + x_7\alpha^3)\beta^{16}] \cdot \omega \\ &= \mathbf{M}' \cdot (x_0, x_1, \cdots, x_6, x_7)^T, \end{aligned} \tag{4}$$

where the matrix $\mathbf{M}'$ is given below.

$$\mathbf{M} = \begin{pmatrix} 1&1&1&0&1&0&1&1 \\ 0&0&0&0&1&1&1&0 \\ 1&0&1&0&1&0&0&1 \\ 1&0&1&1&1&0&0&0 \\ 0&0&1&0&1&1&1&0 \\ 0&1&1&0&0&1&1&1 \\ 1&0&1&1&1&1&0&0 \\ 0&1&1&0&1&0&1&1 \end{pmatrix} \quad \text{and} \quad \mathbf{M}' = \begin{pmatrix} 1&0&1&1&1&0&0&1 \\ 0&1&0&1&1&1&0&0 \\ 1&0&1&0&0&1&1&0 \\ 0&1&1&0&0&0&1&0 \\ 1&0&0&1&0&1&1&1 \\ 1&1&0&0&0&0&1&1 \\ 0&1&1&0&0&0&0&1 \\ 0&0&1&0&1&1&1&1 \end{pmatrix}.$$

Multiplication by ω Using Look-Up Tables. Based on the Equations (2)–(4), one can generate 256-byte look-up tables with respect to the chosen bases.

4.4 Implementation Platforms and Development Tools

In this section, we briefly describe two low-power microcontrollers for implementing the WG-8 stream cipher as well as the corresponding development tools.

8-Bit Microcontroller ATmega128L and Development Tool. The low-power 8-bit microcontroller ATmega128L [1] from Atmel is based on the AVR enhanced RISC architecture with 128 Kbytes of In-System Self-Programmable Flash, 4 Kbytes EEPROM and 8 Kbytes Internal SRAM. It is equipped with 133 highly-optimized instructions and most of them can be executed within one clock cycle. Moreover, the clock frequency of the ATmega128L can run from 0 to 8 MHz and the power supplies can go from 2.7 to 5.5 V. We use the latest integrated development environment Atmel Studio 6.0 [2] from Atmel for implementing and testing the performance of the WG-8 on the target platform.

16-Bit Microcontroller MSP430F1611 and Development Tool. The 16-bit microcontroller MSP430F1611 [38] from Texas Instruments has a traditional von-Neumann architecture with 48 Kbytes Flash memory and 10 Kbytes RAM. All special function registers, peripherals, RAM and Flash/ROM share the same address space. The clock frequency of the MSP430F1611 ranges from 0 to 8 MHz and the power supplies can go from 1.8 to 3.6 V. The MSP430F1611 features 27 instructions and 7 different addressing modes that provide great flexibility in data manipulation. To implement and simulate the WG-8 on the target platform, we use the CrossWorks for MSP430 Version 2.1 from Rowley Associates [35].

4.5 Experimental Results and Comparisons

In this section, we report our experimental results for implementing the stream cipher WG-8 on the low-power microcontrollers ATmega128L and MSP430F1611 and compare our results with other lightweight-cryptography implementations on the same or similar platforms. We focus on three major performance criteria for implementing cryptographic primitives on resource-constrained environments, namely throughput, code size, and energy consumption (i.e., energy/bit). Table 3 compares our implementation results with previous work in terms of the aforementioned three performance criteria. Note that we estimate the per bit energy consumptions by the formula: $\text{energy/bit} = \frac{\text{Supply Voltage}\times\text{Current}\times\text{Cycles}}{\text{Clock Frequency}\times\text{Number of Bits}}$, which is based on the typical current consumption of a low-power microcontroller for the given clock frequency and supply voltage.

From Table 3, we note that on 8-bit ATmega microcontrollers the throughput of WG-8 is about $2 \sim 15$ times higher than that of stream ciphers Grain, Trivium, Salsa20, and WG-7, block ciphers PRESENT-80 and XTEA as well as the hybrid cipher Hummingbird, whereas the energy consumption of WG-8 is around

Table 3. Performance Comparison of Lightweight-Cryptography Implementations on Low-Power Microcontrollers

Low-Power Microcontroller	Cryptographic Primitive	Clock Freq. [MHz]	Opt. Goal/ Method	Memory Usage [byte]		Setup [cycle]	Throughput [Kbits/sec]	Energy/Bit [nJ]
				Flash	SRAM			
ATmega	AES [31]	8 MHz	RAM	1,912	176	789	475.6	179
			Speed	1,912	256	747	513.8	165
	PRESENT-80 [33]		Size	1,474	32	–	0.99	85,819
			Speed	2,398	528	–	66.7	1,274
	Hummingbird [15]		Size	1,308	–	14,735	34.9	2,433
			Speed	10,918	–	8,182	91.5	929
	Hummingbird-2 [16]		RAM	3,600	114	2,970	171.8	495
			Speed	3,200	1,500	1,800	258.6	329
	XTEA [32]		Speed	820	–	–	51.7	1,645
	Grain [32]		Speed	778	20	107,336	12.9	6,556
	Trivium [32]		Speed	424	36	775,726	12.0	7,066
	Salsa20 [28]		Speed	3,842	258	318	83.7	101,564
	WG-7 [26]		Size	938	–	20,917	34.0	2,497
	WG-8		TFA	2,450	20	99,702	3.58	23,739
			CLT	2,238	148	10,683	31.7	2,683
			DLT	**1,984**	**20**	**1,379**	**185.5**	**458**
MSP430	PRINTcipher-48 [18]	8 MHz	Speed	6,424	48	–	4.5	153
	AES [18]		Speed	10,898	218	–	78.0	154
	PRESENT-80 [18]		Speed	6,424	288	–	19.4	619
	KLEIN-64 [18]		Speed	6,424	288	–	65.0	185
	Hummingbird [15]		Size	1,064	–	9,667	53.0	226
			Speed	1,360	–	4,824	104.9	114
	Hummingbird-2 [16]		Size	770	50	5,984	84.2	143
			Speed	3,648	114	1,361	356.5	34
	WG-7 [26]		Size	1,050	–	18,379	21.0	572
	WG-8		TFA	2,110	20	127,944	2.44	4,926
			CLT	2,628	148	15,265	10.8	1,107
			DLT	**1,558**	**20**	**3,604**	**95.9**	**125**

$2 \sim 220$ times smaller than that of those ciphers. Moreover, WG-8 has the comparable throughput and energy efficiency with the hybrid cipher Hummingbird-2 (optimized with assembly language). On the 8-bit platform, WG-8 is less efficient than AES in terms of throughput and energy consumption. The main reason is that WG-8 is a bit-oriented stream cipher whereas AES is a block cipher with block size 128-bit. Furthermore, the code size of WG-8 is medium and the SRAM usage of WG-8 is small among all the lightweight implementations.

On 16-bit MSP430 microcontrollers, the throughput of WG-8 is about $1 \sim 20$ times higher than that of the stream cipher WG-7 as well as block ciphers PRINTcipher-48, AES, PRESENT-80, and KLEIN-64, whereas the energy efficiency is comparable with that of those ciphers. While WG-8 has similar throughput and energy efficiency as the hybrid cipher Hummingbird, it is less efficient when compared to the Hummingbird-2 cipher. The main reason comes from the optimization with the assembly language in the speed-optimized Hummingbird-2 implementation. Furthermore, the code size of WG-8 is about $2 \sim 7$ times smaller than block ciphers PRINTcipher-48, AES, PRESENT-80, and KLEIN-64 as well as the hybrid cipher Hummingbird-2, and is comparable with the Hummingbird cipher. Regarding to the SRAM usage, the stream cipher WG-8 is superior to other block cipher and stream ciphers.

In addition, for the three implementation variants, we note that on both 8-bit and 16-bit platforms the DLT method is consistently better than both CLT and TFA methods with respect to throughput and energy consumption. The reason lies in the efficient memory access for look-up tables on both microcontrollers.

5 Conclusion

In this paper, we present a lightweight stream cipher WG-8 targeted for resource-constrained devices like RFID tags, smart cards, and wireless sensor nodes, which inherits all the good randomness and cryptographic properties of the well-known WG stream cipher family. A detailed cryptanalysis shows that WG-8 is resistant to the most common attacks against stream ciphers. Moreover, the software implementations on low-power microcontrollers demonstrate the high performance and low energy consumption of the WG-8 stream cipher, when compared to most of previous block ciphers and stream ciphers. Therefore, the stream cipher WG-8 is a competitive candidate for securing pervasive embedded applications.

References

1. Atmel Corporation, ATmega128(L): 8-bit Atmel Microcontroller with 128 KBytes In-System Programmable Flash (2011), http://www.atmel.com/Images/doc2467.pdf
2. Atmel Corporation, Atmel Studio 6 – The Integrated Development Environment (2012), http://www.atmel.com/microsite/atmel_studio6/
3. Babbage, S., Dodd, M.: The Stream Cipher MICKEY 2.0, ECRYPT Stream Cipher (2006), http://www.ecrypt.eu.org/stream/p3ciphers/mickey/mickey_p3.pdf
4. Biryukov, A., Shamir, A.: Cryptanalytic Time/Memory/Data Tradeoffs for Stream Ciphers. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 1–13. Springer, Heidelberg (2000)
5. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M., Seurin, Y., Vikkelsoe, C.: PRESENT: An ultra-lightweight block cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007)
6. De Cannière, C., Dunkelman, O., Knežević, M.: KATAN and KTANTAN — A Family of Small and Efficient Hardware-Oriented Block Ciphers. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 272–288. Springer, Heidelberg (2009)
7. De Cannière, C., Preneel, B.: Trivium – A Stream Cipher Construction Inspired by Block Cipher Design Principles. ECRYPT Stream Cipher (2005), http://www.ecrypt.eu.org/stream/papersdir/2006/021.pdf
8. Chen, L., Gong, G.: Communication System Security. Chapman & Hall/CRC, Boca Raton (2012)
9. Chepyzhov, V.V., Johansson, T., Smeets, B.: A Simple Algorithm for Fast Correlation Attacks on Stream Ciphers. In: Schneier, B. (ed.) FSE 2000. LNCS, vol. 1978, pp. 181–195. Springer, Heidelberg (2001)
10. Courtois, N.T.: Fast Algebraic Attacks on Stream Ciphers with Linear Feedback. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 176–194. Springer, Heidelberg (2003)

11. Courtois, N., Meier, W.: Algebraic Attacks on Stream Ciphers with Linear Feedback. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 345–359. Springer, Heidelberg (2003)
12. Dinur, I., Shamir, A.: Cube Attacks on Tweakable Black Box Polynomials. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 278–299. Springer, Heidelberg (2009)
13. Driessen, B., Hund, R., Willems, C., Paar, C., Holz, T.: Don't Trust Satellite Phones: A Security Analysis of Two Satphone Standards. In: The 33th IEEE Symposium on Security and Privacy - S&P 2012, pp. 128–142 (2012)
14. Eisenbarth, T., Kumar, S., Paar, C., Poschmann, A., Uhsadel, L.: A Survey of Lightweight-Cryptography Implementations. IEEE Design & Test of Computers 24(6), 522–533 (2007)
15. Engels, D., Fan, X., Gong, G., Hu, H., Smith, E.M.: Hummingbird: Ultra-Lightweight Cryptography for Resource- Constrained Devices. In: Sion, R., Curtmola, R., Dietrich, S., Kiayias, A., Miret, J.M., Sako, K., Sebé, F. (eds.) RLCPS, WECSR, and WLC 2010. LNCS, vol. 6054, pp. 3–18. Springer, Heidelberg (2010)
16. Engels, D., Saarinen, M.-J.O., Schweitzer, P., Smith, E.M.: The Hummingbird-2 Lightweight Authenticated Encryption Algorithm. In: Juels, A., Paar, C. (eds.) RFIDSec 2011. LNCS, vol. 7055, pp. 19–31. Springer, Heidelberg (2012)
17. Feldhofer, M., Wolkerstorfer, J., Rijmen, V.: AES Implementation on a Grain of Sand. IEE Proceedings Information Security 15(1), 13–20 (2005)
18. Gong, Z., Nikova, S., Law, Y.: KLEIN: A New Family of Lightweight Block Ciphers. In: Juels, A., Paar, C. (eds.) RFIDSec 2011. LNCS, vol. 7055, pp. 1–18. Springer, Heidelberg (2012)
19. Gong, G., Rønjom, S., Helleseth, T., Hu, H.: Fast Discrete Fourier Spectra Attacks on Stream Ciphers. IEEE Transactions on Information Theory 57(8), 5555–5565 (2011)
20. Guo, J., Peyrin, T., Poschmann, A., Robshaw, M.: The LED Block Cipher. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 326–341. Springer, Heidelberg (2011)
21. Hell, M., Johansson, T., Meier, W.: Grain: A Stream Cipher for Constrained Environments. International Journal of Wireless and Mobile Computing 2(1), 86–93 (2007)
22. Kaps, J.-P.: Chai-tea, Cryptographic Hardware Implementations of xTEA. In: Chowdhury, D.R., Rijmen, V., Das, A. (eds.) INDOCRYPT 2008. LNCS, vol. 5365, pp. 363–375. Springer, Heidelberg (2008)
23. Knudsen, L., Leander, G., Poschmann, A., Robshaw, M.J.B.: PRINTcipher: A Block Cipher for IC-Printing. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 16–32. Springer, Heidelberg (2010)
24. Leander, G., Paar, C., Poschmann, A., Schramm, K.: New Lightweight DES Variants. In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 196–210. Springer, Heidelberg (2007)
25. Liu, D., Yang, Y., Wang, J., Min, H.: A Mutual Authentication Protocol for RFID Using IDEA, Auto-ID Labs White Paper, WP-HARDWARE-048 (March 2009), http://www.autoidlabs.org/uploads/media/AUTOIDLABS-WP-HARDWARE-048.pdf
26. Luo, Y., Chai, Q., Gong, G., Lai, X.: WG-7: A Lightweight Stream Cipher with Good Cryptographic Properties. In: IEEE Global Communications Conference – GLOBECOM 2010, pp. 1–6 (2010)
27. Meier, W., Staffelbach, O.: Fast Correlation Attacks on Certain Stream Ciphers. Journal of Cryptology 1(3), 159–176 (1989)

28. Meiser, G., Eisenbarth, T., Lemke-Rust, K., Paar, C.: Efficient Implementation of eSTREAM Ciphers on 8-bit AVR Microcontrollers. In: International Symposium on Industrial Embedded Systems – SIES 2008, pp. 58–66 (2008)
29. Nawaz, Y., Gong, G.: WG: A Family of Stream Ciphers with Designed Randomness Properties. Information Science 178(7), 1903–1916 (2008)
30. Orumiehchiha, M.A., Pieprzyk, J., Steinfeld, R.: Cryptanalysis of WG-7: A Lightweight Stream Cipher. Cryptography and Communications 4(3-4), 277–285 (2012)
31. Osvik, D.A., Bos, J.W., Stefan, D., Canright, D.: Fast Software AES Encryption. In: Hong, S., Iwata, T. (eds.) FSE 2010. LNCS, vol. 6147, pp. 75–93. Springer, Heidelberg (2010)
32. Otte, D.: AVR-Crypto-Lib (2012), http://www.das-labor.org/wiki/AVR-Crypto-Lib/en
33. Poschmann, A.: Lightweight Cryptography – Cryptographic Engineering for a Pervasive World, Ph.D. Thesis, Department of Electrical Engineering and Information Science, Ruhr-Universitäet Bochum, Bochum, Germany (2009)
34. Rønjom, S., Helleseth, T.: A New Attack on the Filtering Generator. IEEE Transactions on Information Theory 53(5), 1752–1758 (2007)
35. Rowley Associates, CrossWorks for MSP430 (2012), http://www.rowley.co.uk/msp430/
36. Shibutani, K., Isobe, T., Hiwatari, H., Mitsuda, A., Akishita, T., Shirai, T.: *Piccolo*: An Ultra-Lightweight Blockcipher. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 342–357. Springer, Heidelberg (2011)
37. Siegenthaler, T.: Decrypting a Class of Stream Ciphers Using Ciphertext Only. IEEE Transactions on Computers 34(1), 81–85 (1985)
38. Texas Instuments Inc., MSP430F15x, MSP430F16x, MSP430F161x Mixed Signal Microcontroller (2011), http://www.ti.com/lit/ds/symlink/msp430f1611.pdf
39. Verdult, R., Garcia, F.D., Balasch, J.: Gone in 360 Seconds: Hijacking with Hitag2. In: The 21st USENIX Security Symposium - USENIX Security 2012, pp. 237–252. USENIX Association (2012)
40. Wu, H., Preneel, B.: Chosen IV Attack on Stream Cipher WG, ECRYPT Stream Cipher Project Report, 2005/045, http://cr.yp.to/streamciphers/wg/045.pdf