Journal of information and communication convergence engineering
http://jicce.org

J. Inf. Commun. Converg. Eng. 12(4): 237-245, Dec. 2014 Regular paper

Higher-Order Countermeasures against Side-Channel Cryptanalysis on Rabbit Stream Cipher

Jonathan A. P. Marpaung, Bruce Ndibanje, and Hoon Jae Lee*, *Member, KIICE*

Cryptography & Network Security Lab., Department of Ubiquitous IT, Dongseo University, Busan 617-716, Korea

Abstract

In this study, software-based countermeasures against a side-channel cryptanalysis of the Rabbit stream cipher were developed using Moteiv's Tmote Sky, a popular wireless sensor mote based on the Berkeley TelosB, as the target platform. The countermeasures build upon previous work by improving mask generation, masking and hiding other components of the algorithm, and introducing a key refreshment scheme. Our contribution brings improvements to previous countermeasures making the implementation resistant to higher-order attacks. Four functional metrics, namely resiliency, robustness, resistance, and scalability, were used for the assessment. Finally, performance costs were measured using memory usage and execution time. In this work, it was demonstrated that although attacks can be feasibly carried out on unprotected systems, the proposed countermeasures can also be feasibly developed and deployed on resource-constrained devices, such as wireless sensors.

Index Terms: Rabbit stream cipher, Side-channel cryptanalysis, Software countermeasures, Ubiquitous technology

I. INTRODUCTION

Widespread adoption of ubiquitous technologies has led to the production of a variety of tiny devices used for data collection (sensors), enabling network connectivity and performing actions to the environment (actuators). These devices are the building blocks for a complete, ubiquitous ecosystem providing a wide variety of solutions ranging from healthcare monitoring to military battlefield awareness systems. With an increase in the deployment of these solutions, the security level of countermeasures has become an extremely important topic of research. Furthermore, a secure implementation of the security solutions is often overlooked.

Cryptography is the building block of all security systems. Cryptographic primitives are used in various applications to provide confidentiality, integrity, and authentication. Although mathematically strong, sometimes the cryptosystems themselves, if poorly implemented, can contain vulnerabilities, thus posing a risk to the entire system. One of these implementation vulnerabilities is the side-channel leakage of critical security properties such as the secret key. This area of research is called side-channel cryptanalysis and is the focus of this study.

In this study, we develop software-based countermeasures against a side-channel cryptanalysis of the Rabbit stream cipher using a Tmote Sky wireless sensor mote as our target platform. Improvements to previous countermeasures are proposed to make the implementation resistant to higher-order attacks. We formally evaluate our work by using evaluation standards, such as Federal Information Processing Standard (FIPS)-140, Common Criteria for Information Technology Security Evaluation (commonly called Common Criteria or CC), and the Special Publication (SP) 800 series.

Received 25 March 2014, **Revised** 11 April 2014, **Accepted** 27 June 2014
***Corresponding Author** Hoon Jae Lee (E-mail: hjlee@dongseo.ac.kr, Tel: +82-51-320-1730)
Cryptography & Network Security Lab., Department of Ubiquitous IT, Dongseo University, Sasang-Gu, Busan 617-716, Korea.

Open Access http://dx.doi.org/10.6109/jicce.2014.12.4.237 print ISSN: 2234-8255 online ISSN: 2234-8883

We also assess four functional metrics, namely resiliency, robustness, resistance, and scalability. Finally, we look at the performance costs of the proposed implementation by measuring memory usage and execution time.

II. APPROACH

The basic problem that we address is the question of how to securely implement the Rabbit stream cipher on embedded systems with the threat of side-channel cryptanalysis. We assume an attack model where the attack is performed without physically tampering with or removing the targeted node from its deployed location. This characteristic is important as many physical attacks such as fault injection are not only more efficient but also more intrusive. In fact, a simple memory dump attack can obtain sensitive information in less than 1 minute [1]. In this scenario, we assume that an intrusion detection system is disabled or fails to detect side-channel attacks due to the non-intrusive nature of the attack. The scope of this study is the security of individual nodes; therefore, the presence or participation of other nodes is outside the scope of this discussion. Furthermore, we assume that the adversary is sufficiently determined to perform a higher-order differential cryptanalysis and is thus capable of bypassing the previously proposed countermeasures.

Our countermeasures build upon the scheme proposed by Bae et al. by improving it to address higher-order attacks, adding an additional layer of complexity, and evaluating the results using formal verification standards, such as FIPS 140 and Common Criteria. In other words, we want to propose strong solutions to handle a worst-case-scenario side-channel attack. We implement these countermeasures in nesC TinyOS running on the Berkeley TelosB Mote IV.

A. Rabbit

The Rabbit stream cipher was selected for the final eSTREAM portfolio [2] organized by European Network of Excellence for Cryptology in 2003 [3]. It is one of algorithms of the ISO/IEC 18033-4 Stream Ciphers [4] on ISO Security standardization and was evaluated as having a DPA attack complexity of 'medium.'

The Rabbit algorithm is a synchronous stream cipher that can be briefly described as follows: it takes a 128-bit secret key and a 64-bit IV as input, and for each iteration, generates an output block of 128 pseudo-random bits from a combination of the internal state bits. Encryption/decryption is done by XORing the pseudo-random data with the plaintext/ciphertext. The size of the internal state is 513 bits divided between eight 32-bit state variables, eight 32-bit counters, and one counter carry bit. The eight state variables are updated by eight coupled non-linear functions. The counters ensure a lower bound on the period length for the state variables. Rabbit was designed to be faster than the commonly used ciphers and to justify a key size of 128 bits for encrypting up to 2^{64} blocks of plaintext. This means that for an attacker that does not know the key, it will not be possible to distinguish up to 2^{64} blocks of cipher output from the output of a truly random generator, using fewer steps than required for an exhaustive key search of more than 2^{128} keys.

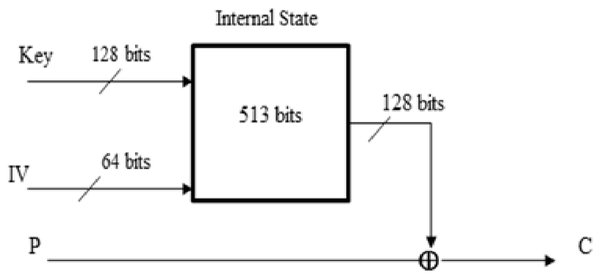


Fig. 1. Rabbit algorithm overview.

Fig. 1 shows the operations of the algorithm and where the size of the internal state is 513 bits divided between eight 32-bit state variables, eight 32-bit counters, and one counter carry bit. The eight state variables are updated by eight coupled non-linear functions. The counters ensure a lower bound on the period length for the state variables.

B. Security Design Goals

Our work has the following security design goals:

1) Limit the window of opportunity for a side-channel cryptanalysis (Resistance).
2) Increase the complexity for carrying out an attack (Robustness).
3) Contain the damage of an attack to only the compromised node (Resilience).
4) Minimize cost and maximize performance (Scalability).

III. PREVIOUS COUNTERMEASURES

Bae et al. [5] were the first to perform a side-channel cryptanalysis on Rabbit and to propose countermeasures according to the attack model. Their countermeasures were based on the masking and hiding processes illustrated in Fig. 2.

A. Masking

A first-order Boolean masking scheme was used in the initialization of the master counter states in the key setup

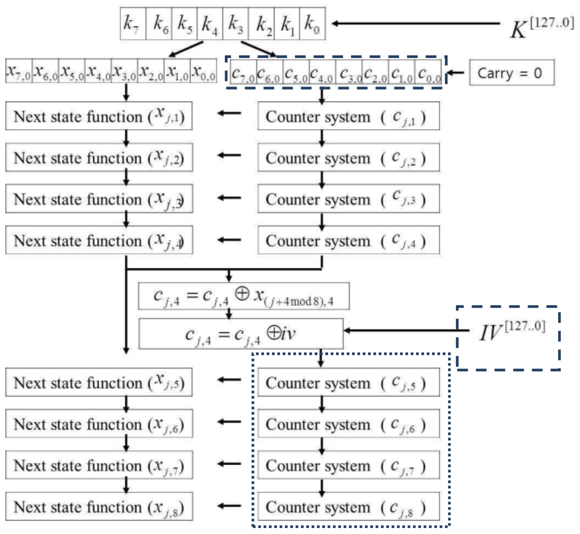

$K^{[127..0]}$
k_7 k_6 k_5 k_4 k_3 k_2 k_1 k_0
$x_{7,0}$ $x_{6,0}$ $x_{5,0}$ $x_{4,0}$ $x_{3,0}$ $x_{2,0}$ $x_{1,0}$ $x_{0,0}$
$c_{7,0}$ $c_{6,0}$ $c_{5,0}$ $c_{4,0}$ $c_{3,0}$ $c_{2,0}$ $c_{1,0}$ $c_{0,0}$
Carry = 0
Next state function ($x_{j,1}$)
Counter system ($c_{j,1}$)
Next state function ($x_{j,2}$)
Counter system ($c_{j,2}$)
Next state function ($x_{j,3}$)
Counter system ($c_{j,3}$)
Next state function ($x_{j,4}$)
Counter system ($c_{j,4}$)
$c_{j,4} = c_{j,4} \oplus x_{(j+4 \bmod 8),4}$
$c_{j,4} = c_{j,4} \oplus iv$
$IV^{[127..0]}$
Next state function ($x_{j,5}$)
Counter system ($c_{j,5}$)
Next state function ($x_{j,6}$)
Counter system ($c_{j,6}$)
Next state function ($x_{j,7}$)
Counter system ($c_{j,7}$)
Next state function ($x_{j,8}$)
Counter system ($c_{j,8}$)

Fig. 2. Bae et al.'s countermeasure scheme (dashed lines, masking; dotted lines, hiding).

phase and the expansion of the IV in the IV setup phase. The mask is generated using a random number generator that takes as input a 16-bit timer and the IV. The masking values are removed before the next-state function is called in the IV setup as the counter variables are used for introducing non-linearity. This scheme removes the correlation between the power traces and the HW hypothesis by changing the internal values during every resynchronization of the algorithm. The authors state that this scheme can prevent the first-order power analysis attacks but is still vulnerable to a higher-order differential cryptanalysis. In order to make such an attack difficult, their scheme is combined with the hiding mechanisms explained below.

B. Hiding

Hiding is performed by executing critical operations in a random order. This method is enhanced by adding dummy operations that shift the execution time making it difficult for an adversary to isolate the desired leakage that corresponds to the target operations. The authors hide the ordering of the counter variable initialization during the IV setup by randomizing the sequence of the eight iterations. They use the shuffling method adapted from the stream cipher RC4, which they acknowledge as being vulnerable to a side-channel cryptanalysis, along with signal processing techniques.

A time-shifting mechanism is implemented by inserting dummy cycles (called 'no operations' or 'nops') during the counter variable initialization in the IV setup phase. Although the number of dummy cycles is randomized, the execution time needs to be regularly in line with the principle of a synchronized stream cipher. Therefore, the authors implemented a total of 50 inserted cycles between the initialization step of the counter variables and the unmasking step. The detailed countermeasure algorithm is shown in Fig. 3.

```
Input : IV, Counter variables, State variables, Masking variables
Output : Recovered Counters
//generation of random masks
for j=0 to 15 do
    Mask[j]=RandomNumberGenerate(IV, GET_TIMER())
end for
// masked master counter states ( 1st order)
for j=0 to 7 do
    master_c[j] = Mask[j] $\oplus$ c[j] $\oplus$ x[(j+4)&0x7]
end for
// masked IV (2nd order)
I[0] = $IV^{[31,...,0]}$ $\oplus$ Mask[8]
I[2] = $IV^{[63,...,32]}$ $\oplus$ Mask[10]
I[4] = $IV^{[31,...,0]}$ $\oplus$ Mask[12]
I[6] = $IV^{[63,...,32]}$ $\oplus$ Mask[14]
I[1] = (( I[0] >> 16 ) | ( I[2] & 0xFFFF0000 )) $\oplus$ Mask[9]
I[3] = (( I[2] << 16 ) | ( I[0] & 0x0000FFFF )) $\oplus$ Mask[11]
I[5] = (( I[0] >> 16 ) | ( I[2] & 0xFFFF0000 )) $\oplus$ Mask[13]
I[7] = (( I[2] << 16 ) | ( I[0] & 0x0000FFFF )) $\oplus$ Mask[15]
// initialization step of masked master-counter states and masked IV
for j=0 to 7 do
    Select j randomly, only one time from j=0 to 7
    Insert random cycles(n) and selected n randomly
    c[j] = master_c[j] $\oplus$ I[j]
end for
carry = master_carry
// recovering of initialized counter variables
for j=0 to 7 do
    Select j randomly, only one time from j=0 to 7
    Insert random cycles(50-n)
    c[j] = (Mask[j] $\oplus$ c[j]) $\oplus$ Mask[j+8]
end for
Return c[j]
```

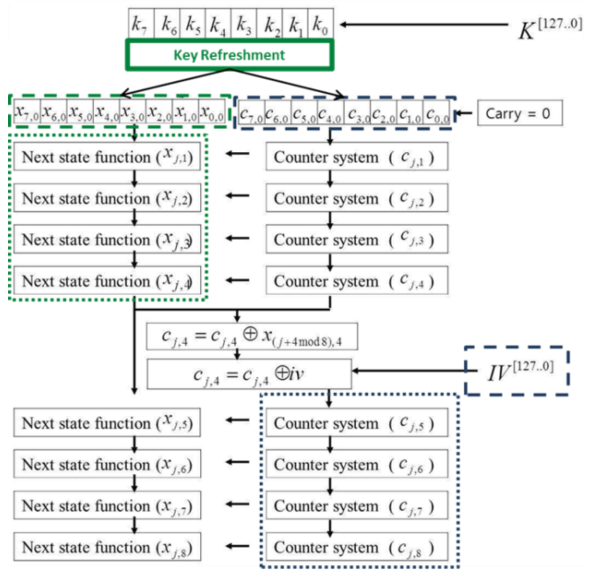
Fig. 3. Bae et al.'s countermeasure algorithm.

C. Evaluation

The countermeasure proposed by Bae et al. successfully removes the relationships between the correlation coefficients and the power traces. The additional costs of running this implementation on an ATmega 128L is summarized in Table 1.

Table 1. Performance cost of Bae et al.'s countermeasure on ATmega 128L

	Basic cipher	Bae et al.'s countermeasure	Additional cost
Flash memory	8.934 kB	10.218 kB	12.3%
SRAM	59 bytes	59 bytes	0%
Execution time	6.6 ms	8.2 ms	24%



k_7 k_6 k_5 k_4 k_3 k_2 k_1 k_0
$K^{[127..0]}$
Key Refreshment
Carry = 0
Next state function ($x_{j,1}$)
Counter system ($c_{j,1}$)
$c_{j,4} = c_{j,4} \oplus x_{(j+4 \bmod 8),4}$
$c_{j,4} = c_{j,4} \oplus iv$
$IV^{[127..0]}$

Fig. 4. Proposed side-channel attacks (SCA) countermeasures (dashed lines, masking; dotted lines, hiding).

IV. PROPOSED COUNTERMEASURES

We build upon the work described in the previous section by proposing additional and improved masking and hiding schemes, as well as a new key management component. The new countermeasure scheme is designed to provide resilience and resistance against higher-order attacks. The complete countermeasure scheme is illustrated in Fig. 4.

A. Key Management

Most cryptanalytic attacks depend on the ability to obtain many encryptions performed using a single key. Birthday attacks on a cipher with a key size of k, for example, require only $2^{k/2}$ encryptions performed using the same key. However, differential cryptanalysis attacks typically have a higher threshold, requiring many more encryptions. Therefore, it can be inferred that there have to be a certain maximum threshold number of encryptions before a key can no longer be safely used. In this section, we discuss how if appropriately defined, key management can be used for defending against a side-channel cryptanalysis.

Key management deals with the generation, exchange, storage, use, and refreshment of keys. We focus on the idea of key refreshment and briefly discuss several possible key exchange mechanisms as we limit our countermeasure proposal to individual nodes.

1) Key Refreshment

The standard Rabbit encryption scheme using a 128-bit key can be used for encrypting up to 2^{64} blocks (128 bit) of plaintext. Bae et al.'s power analysis attack on Rabbit, is performed during the key and IV setup phase. In a real-world scenario, if the attack is performed non-intrusively, without the ability to control when the device initiates the key setup process, an adversary would have to wait for the encryption of 2^{71} bits of data before the next setup phase. Furthermore, the attack model requires 1,000 traces of the setup phase with the same key but random IVs. Therefore, an adversary would need to know when the setup phase is being executed, wait between 2^{71} bits of encryptions to capture each power trace, and do this 1,000 times assuming that the same key is used throughout. Further, assuming that there exists an adversary sufficiently determined to carry out this attack, we add more complexity to this attack by introducing a key refreshment mechanism.

The effectiveness of a key is determined not only by its length but also by the number of encryptions performed using the same key. Abdalla and Bellare [6] argued that re-keying (key refreshment) provides a provable increase in the security of an application. They defined the encryption threshold Q as the number of k-bit messages that can be safely encrypted, with $Q \approx 2^{k/2}$ for the single-key scheme. By using parallel or serial re-keying methods, they showed that re-keying every set of $2^{k/3}$ encryptions increases the encryption threshold to $Q \approx 2^{2k/3}$. The sub-key lifetime to $l = 2^{k/3}$ allows significantly more data to be safely encrypted. We extend this idea as a countermeasure to the side-channel cryptanalysis attack on Rabbit.

We propose a key refreshment scheme that generates sub-keys that are used as session keys for communication between nodes in a wireless sensor network deployment. We assume that an initial shared master key K_M is pre-distributed in a node and that one or more of the other nodes depending on the key exchange mechanism are used. A key derivation function F is a pseudo-random number generator that takes as input the 128-bit master key, 64-bit IEEE EUI-64 unique node identifier, 64-bit values of a timer, and 64 bits of entropy. The secure hash function SHA-256 is used for processing the aforementioned input, essentially deriving a sub-key by using a keyed-hash message authentication code (HMAC). The use of system time and extra entropy ensures that the same sub-key is not generated twice. To prevent adversaries from obtaining a sufficient number of encryptions, we limit the lifetime of each sub-key to $l = 2^{43}$ encryptions.

2) Key Exchange

Perrig et al. [7] proposed a suite of security protocols for sensor networks and named it SPINS. Understanding that public key cryptography protocols consume a considerable number of resources for sensor nodes, a symmetric protocol that uses a base station as a trusted agent for key setup was proposed. In the trust setup, two nodes (A and B) share a pre-distributed master secret key with base stations X_{AS} and X_{BS}, respectively. A shared secret session key SK_{AB} is established. Strong key freshness is ensured by using the nonces N_A and N_B. The related key agreement protocol is as follows:

$A \rightarrow B$: N_A, A,

$B \rightarrow S$: N_A, N_B, A, B, $\mathrm{MAC}(K'_{BS}, N_A|N_B|A|B)$,

$S \rightarrow A$: $\{SK_{AB}\}K_{SA}$, $\mathrm{MAC}(K'_{SA}, N_A|B|\{SK_{AB}\}K_{SA})$,

$S \rightarrow B$: $\{SK_{AB}\}K_{SB}$, $\mathrm{MAC}(K'_{SB}, N_A|B|\{SK_{AB}\}K_{SB})$,

Another pre-distributed scheme is the have each node have a unique pairwise key for communication between the nodes. However, this scheme is costly when the number of sensors increases beyond the amount of memory available, which is already limited. Moreover, scalability is difficult to achieve when new nodes need to be introduced and another mechanism is required for updating the existing nodes with the keys of the new nodes.

A random key pre-distribution scheme proposed in [8], wherein each node receives a randomly chosen 'ring of keys,' is applied to each sensor prior to deployment. Since a node only knows a subset of potentially hundreds of keys, a shared-key discovery protocol for key distribution, revocation, and node re-keying is introduced. This scheme costs fewer resources on individual nodes and limits the damage done if a node is compromised. The authors in [9] worked on improving this scheme by exploiting deployment knowledge to improve network performance and resilience against node capture.

Chan and Perrig [10] built upon the work in SPINS by developing a key distribution scheme using peer intermediaries for key establishment (PIKE). The scheme uses peer sensor nodes as trusted intermediaries for establishing keys between any two nodes irrespective of the network topology or density. Basically, this protocol decentralizes the role of the trusted base station in SPINS.

B. Masking

Masking the master counter variables and IV expansion provided a layer of defense against the first-order attacks. We improve the previous masking countermeasures by also masking the master state variables, thereby preventing higher-order attacks. The masking method applied generally follows the same principle of masking the counter variables during the key setup phase and then unmasking them before calling the next-state function. However, we propose improvements to the method used for mask generation, also making higher-order attacks difficult to perform.

The first improvement is to use different masks for the counter variables, IV expansion, and master state variables. This will make it difficult for the adversaries to obtain each of these values even if they manage to find one of the masks. The second improvement is to use a stronger pseudo-random number generator for creating the masks. While the previous scheme used an unnamed random number generator that took as input a 16-bit timer and the counter register of the chip, we propose the use of the SHA-256 hash function taking as input a 64-bit timer, 64-bit IEEE EUI-64 unique node identifier, and 256-bit entropy. Code efficiency is achieved by reusing the same SHA-256 function in the key refreshment mechanism. The modification to Bae et al.'s techniques is described in Fig. 5.

C. Hiding

The same technique for time shifting and random ordering of initialization operations is applied to hide the recovery of the masked state variables and the execution of the next-state function. The complete masking and hiding technique used in this improved countermeasure scheme is summarized in Figs. 6 and 7.

```
Input :IV, Counter variables State variable, Masking variables(seed)
Output: Recovered Counters

1.  CounterMask=SHA256(64bitTimer,64bitIEEE-EUI-UNI,
    256bits-Entropy)
2.  //masked master counter states
3.  for j=0 to 7 do
4.  master_c[j]=CounterMask[j]⊕c[j]⊕x[(j+4)&0x7
5.  end for
6.  IVMAsk= SHA256(64bitTimer,64bitIEEE-EUI-UNI,256bits-
    Entropy)
7.  //masked IV expansion
8.  I[0]=IV[31,…0]⊕IVMask[0]
9.  I[2]=IV[63,…32]⊕IVMask[2]
10. I[4]=IV[31,…0]⊕IVMask[4]
11. I[6]=IV[63,…32]⊕IVMask[6]
12. I[1]=((I[0]>>16)|(I[2]&0xFFFF0000))⊕ IVMask[1]
13. I[3]=((I[2]<<16)|(I[0]&0x0000FFFF))⊕ IVMask[3]
14. I[5]=((I[0]>>16)|(I[2]&0xFFFF0000))⊕ IVMask[5]
15. I[7]=((I[2]<<16)|(I[0]&0x0000FFFF))⊕ IVMask[7]
16. Initialization step of masked-counter states and masked IV
17. For j=0to7 do
18. Select randomly, only one time from j=0to 7
19. Insert random cycles(n) and select n randomly
20. c[j]=master_c[j] ⊕ I[j]
21. end for
22. carry=master_carry
23. //hidden recovery of initialized counter variables
24. for j=0to 7 do
25. Select j randomly, only one time from j=0 to 7
26. Insert random cycles(50-n)
27. C[j]=(CounterMask[j]⊕c[j]⊕IVMask[j]
28. end for
29. Return c[j]
```

Fig. 5. Improved countermeasures for IV and counter variables.

```
Input: State variables, Counter variables

1. // hidden next operations
2. While next counter variables being instantiated
3.     Insert random cycles
4. end while
5. While next state variables being instantiated
6.     Insert random cycles
7. end while
```

Fig. 6. Proposed hiding of next-state function.

```
Input: State variables, Masking variables(seed)
Output : Recovered State

1.  // generation of random mask
2.  Mask=SHA256(64bitTimer,64bitIEEE-EUI-UNI, 256bits-Entropy)
3.  //masked master states
4.  master_x[0]=SK[0]⊕Mask[0]
5.  master_x[2]=SK[1]⊕Mask[2]
6.  master_x[4]=SK[2]⊕Mask[4]
7.  master_x[6]=SK[3]⊕Mask[6]
8.  master_x[1]=((SK[3]<<16)|(SK[2]>>16))⊕Mask[1]
9.  master_x[3]=((SK[0]<<16)|(SK[3]>>16))⊕Mask[3]
10. master_x[5]=((SK[1]<<16)|(SK[0]>>16))⊕Mask[5]
11. master_x[7]=((SK[2]<<16)|(SK[1]>>16))⊕Mask[7]
12. //hidden recovery of initialized state variables
13. For j=0to 7 do
14.     Select j randomly, only one time from j=0 to 7
15.     Insert random cycles(50-n)
16.     x[j]=(Mask[j] ⊕ master_x[j])
17. end for
18. Return x[j]
```

Fig. 7. Proposed masking of state variables.

V. RESULTS

We evaluated the proposed countermeasure scheme by using formal evaluation standards, functional metrics, and performance metrics.

A. Formal Evaluation Framework Management

1) FIPS PUB 140-3 Draft

Currently, under public review, the FIPS PUB 140-3 Draft (2009) specifies the security requirements for cryptographic modules [11]. Also published under ISO/IEC 19790:2012, the standard defines four increasing levels of security that cover a range of applications and environments in which the crypto modules can be deployed. Although under this standard only the National Institute of Standards and Technology (NIST)-approved cryptographic algorithms are allowed, we apply the rest of the requirements to evaluate the Rabbit countermeasures. A side-channel cryptanalysis on Rabbit falls under the category of non-invasive attacks that are governed by Security Levels 3 and 4.

At Security Level 3, the cryptographic module shall protect the module's critical security parameters (e.g., secret keys) against all of the applicable non-invasive attacks specified in Annex F of the abovementioned draft. Documentation is required to specify the mitigation techniques employed against these attacks, how the techniques work, and their effectiveness. Annexure F specifies the definitions of the non-invasive attack methods covered under this standard including correlation power analysis (CPA), differential power analysis (DPA), differential electro-magnetic analysis (DEMA), simple power analysis (SPA), simple electro-magnetic analysis (SEMA), and timing analysis (TA). Based on these definitions, the countermeasures proposed for Rabbit fulfill the requirements for Security Level 3.

At Security Level 4, the module shall undergo testing and shall meet the requirements defined by the validation authority. Since we did not perform a higher-order attack to test the new countermeasures, our proposal does not meet the requirements for this security level. However, the basic countermeasures proposed by Bae et al. have been tested against first-order attacks, thereby permitting the verification of Security Level 4 up to the first-order attacks.

2) Common Criteria

The Common Criteria for Information Technology Security Evaluation (Common Criteria or CC) is an international standard for computer security certification [12]. Published as ISO/IEC 15408, it provides a framework for specifying security functional requirements (SFR) and security assurance requirements (SAR) by defining and using protection profiles (PPs) for a class of security devices and security targets (STs) to identify the security properties for a specific target of evaluation (TOE). Details of specific cryptographic algorithms and implementations are outside of the scope of CC. Further, the evaluation assurance levels (EALs) provide an increasing metric of the level of assurance obtained and range from EAL1 (Functionally Tested) to EAL7 (Formally Verified Design and Tested).

The cryptographic module, security level "enhanced" PP written by the German government's Federal Office for Information Security (BSI) describes the security requirements for cryptographic modules [13]. A security requirement for handling side-channel attacks is defined under the assurance family FPT_EMSEC:

"*This family defines requirements to mitigate intelligible emanations. The requirements address the level of resistance of the cryptographic module against side-channel attacks such as timing analysis, simple power analysis (SPA), differential power analysis (DPA), electromagnetic emanation analysis (EMEA), and template attacks. If the cryptographic* *module applies masking, the requirements also address the level of resistance of the cryptographic module against a higher-order side-channel analysis.*”

Based on the definition above, the proposed countermeasures fulfill the assurance family FPT_EMSEC requirements for a side-channel cryptanalysis. A general vulnerability assessment of possible covert channels (side-channels) can also be investigated under class AVA, but the specification from BSI adequately covers the requirements in detail.

3) SP 800-90A

NIST SP 800-90A specifies the recommendation for random number generation using deterministic random bit generators (DRBGs) [14]. We evaluate the DRBGs used in our countermeasures according to this standard. For hash-based DRBGs, the document recommends the use of an NIST-approved hash function such as the SHA family of secure hashes. The document states that the input to this function should primarily consist of entropy and other inputs in order to provide a security cushion. The minimum entropy required for instantiation and reseeding should have the same length as the desired security strength. Ideally, the input entropy should be equal to or greater than 3/2 of the desired security strength (in bits). Furthermore, the Recommendation strongly advises the use of a personalization string.

The proposed countermeasures employ the SHA-256 function and takes as input 256 bits of entropy along with a 64-bit timer and a 64-bit unique node identifier. As the required security strength of the mask is 256 bits, the recommended entropy input length is met. Therefore, on the basis of this Recommendation, the DRBGs used in the proposed countermeasures fulfill the security requirements specified.

4) SP 800-108

NIST SP 800-108 specifies the recommendation for key derivation using pseudo-random functions [15]. The document identifies a pseudo-random function as the basic building block of key derivation functions. The Recommendation approves the use of the keyed hash message authentication code (HMAC) or the cipher-based message authentication code as the pseudo-random function. The key refreshment procedure defined in the proposed countermeasure fulfills this basic requirement. The document further specifies a family of key derivation functions based on several modes that enable different parties to obtain the same keys from the derived keying material. Since key exchange is beyond the scope of this study, it is unnecessary to evaluate the key refreshment proposal against these techniques.

B. Functional Metrics

Earlier, we defined the research problem as how to securely implement the Rabbit stream cipher against a higher-order side-channel cryptanalysis while assuming that other defense mechanisms, such as intrusion detection systems cannot prevent or detect such attacks. We evaluate the achievement of the security design goals by using the functional metrics of resilience, resistance, scalability, and robustness.

1) Resistance

The first security design goal is to provide resistance against a side-channel cryptanalysis by limiting an adversary’s window of opportunity. This goal is achieved by employing a key refreshment scheme and limiting the sub-key lifetime to $l = 2^{43}$ encryptions. By limiting the number of encryptions performed under each key, a sufficient number of traces cannot be obtained to carry out the attack.

2) Robustness

The robustness of the countermeasures can be defined as the complexity for carrying out an attack. We increased this complexity by masking the master state, applying different masks for different variables, and randomizing the execution of critical operations. By using these techniques, we achieved the second security design goal.

3) Resilience

By applying the proposed countermeasures, even if an adversary manages to optimize the attack by reducing the number of traces needed, obtaining one key is not enough to gain access to the entire network. Different nodes and communication sessions use different sub-keys; therefore, the damage is limited to the compromised node. Moreover, the next session key cannot be derived from only the current session key, creating an additional time constraint.

4) Scalability

Being software-based, the countermeasures can be implemented with ease even across the existing systems. Furthermore, since only the key and IV setup phase is targeted, minimum cost is incurred as most these operations are not performed during normal encryption/decryption. Therefore, it can be concluded that this solution can be widely deployed without incurring a significant overhead cost as the network grows.

C. Performance Metrics

The overall costs are summarized in Table 2. The countermeasures consume mostly more flash memory (ROM) with negligible overhead of SRAM and execution time.

Table 2. Performance cost of countermeasures on Tmote Sky

	Basic cipher	With proposed countermeasures	Additional costs
Flash memory	10,334 bytes	14,749 bytes	43%
SRAM	5,449 bytes	5,618 bytes	3%
Execution time	35 ms	36 ms	2%

VI. DISCUSSION AND CONCLUSIONS

A. Summary of Contributions

In this work, we proposed and developed software-based countermeasures for the Rabbit stream cipher to defend against higher-order side-channel attacks. The countermeasures build upon previous work by improving the mask generation, masking and hiding other components of the algorithm, and introducing a key refreshment scheme. We ported Rabbit to TinyOS as to the best of our knowledge, no one has written an implementation in nesC thus far.

Through experiments using the Berkeley Telos B mote as the target platform, we evaluated the performance of the proposed scheme. Furthermore, we evaluated the security strength of the scheme by using FIPS PUB 140-3, the SP 800 Series, and Common Criteria. The overall scheme fulfills the security design goals that we set of being resistant to higher-order attacks, resilient against a single-node compromise, robust against determined adversaries, and scalable for deployment in wireless sensor networks.

B. Concluding Remarks

Ubiquitous computing applications are prone to a side-channel cryptanalysis if the design and implementation of cryptosystems do not take this class of attack into consideration. Since the use of ubiquitous technologies is becoming more widespread, adversaries can be expected to emerge and take advantage of these vulnerabilities. In this work, however, we demonstrated that although the attacks can be feasibly carried out on unprotected systems, countermeasures can be feasibly developed and deployed on resource-constrained devices such as wireless sensors.

Various methods can be used for increasing the complexity of performing a side-channel cryptanalysis, thus making it an undesirable attack vector for the adversaries. We focused on software-based countermeasures that altered specific implementation issues and cryptosystem design considerations. Other potential methods include physical aspects such as improving the design of microcontroller boards that make it difficult to isolate a clear power trace from GND or VCC. The basic goal of a countermeasure is to introduce mechanisms that reduce the feasibility of carrying out an attack, be it preventing the capture of clear power traces or hiding meaningful information in the power traces.

C. Future Work

To further assess the threat of a side-channel cryptanalysis against Rabbit, we recommend that future work be focused on testing higher-order attacks against the proposed countermeasure scheme and on performing attacks on real-world cryptosystems, such as CyaSSL or SSH.

ACKNOWLEDGMENTS

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology. (NRF, 2014) This research was also supported by the BB21 project of Busan Metropolitan City.

REFERENCES

[1] C. Hartung, J. Balasalle, and R. Han, "Node compromise in sensor networks: the need for secure systems," Department of Computer Science, University of Colorado at Boulder, *Technical Report CU-CS-990-05*, 2005.

[2] Ecrypt stream cipher project [Internet], Available: http://www.ecrypt.eu.org/stream/.

[3] European Network of Excellence for Cryptology II (ECRYPT II) [Internet], Available: http://www.ecrypt.eu.org/.

[4] Information technology - Security techniques - Encryption algorithms - Part 4: Stream ciphers, ISO/IEC 18033-4:2011, 2011.

[5] K. Bae, M. Ahn, H. Lee, J. Ha, and S. Moon, "Power analysis attack and countermeasure on the Rabbit Stream Cipher," in *Proceedings of the 7th International Workshop on Software Engineering for Secure Systems*, Honolulu, HI, pp. 50-56, 2011.

[6] M. Abdalla and M. Bellare, "Increasing the lifetime of a key: a comparative analysis of the security of re-keying techniques," in *Advances in Cryptology (ASIACRYPT 2000).* Heidelberg: Springer, pp. 546-559, 2000.

[7] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler, "SPINS: security protocols for sensor networks," *Wireless Networks*, vol. 8, no. 5, pp. 521-534, 2002.

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

[9] W. Du, J. Deng, Y. S. Han, S. Chen, and P. K. Varshney, "A key management scheme for wireless sensor networks using deployment knowledge," in *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies*, Hong Kong, China, 2004.

[10] H. Chan and A. Perrig, "PIKE: peer intermediaries for key establishment in sensor networks," in *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies,* Miami, FL, pp. 524-535, 2005.

[11] National Institute of Standards and Technology, "Security requirements of cryptographic modules," *FIPS 140-3*, 2009.

[12] Common Criteria for Information Technology Security Evaluation [Internet], Available: http://www.commoncriteriaportal.org/cc/.

[13] German Federal Office for Information Security, *Common Criteria - Protection Profile Cryptographic Modules, Security Level 'Enhanced'*. Bonn: German Federal Office for Information Security, 2008.

[14] National Institute of Standards and Technology, *Recommendation for Random Number Generation Using Deterministic Random Bit Generators (SP 800-90A)*. Gaithersburg, MD: National Institute of Standards and Technology, 2012.

[15] National Institute of Standards and Technology, *Recommendation for Key Derivation Using Pseudorandom Functions (SP 800-108)*. Gaithersburg, MD: National Institute of Standards and Technology, 2009.

Jonathan A. P. Marpaung
received his B.Sc. in Computer Science from the University of Indonesia in 2010, and completed his master's degree working in the Cryptography & Network Security Lab of Dongseo University in 2013. He is currently affiliated with Spentera Security of Jakarta. His research interests include cryptographic engineering, side-channel cryptanalysis, software and systems security, and malware engineering.

Bruce Ndibanje
received his B.Sc. in Computer Sciences from Ngozi University, Burundi, in 2006, and his M.S. in Ubiquitous IT from Dongseo University in 2013. He has worked with many companies in the ICT domain, including Huawei, United Nations, and Econet Wireless Burundi. In 2013, he joined the Cryptography and Network Security Lab at Dongseo University, Busan, Korea, as Doctorate Researcher. His research interests include wireless and sensor networks; authentication protocol; security in e-healthcare systems, cloud computing, and cellular networks; side-channel attacks, and countermeasures. He is a member of the IEEE Computer Society.

Hoon Jae Lee
received his B.S., M.S., and Ph.D. in Electrical Engineering from Kyungpook National University in 1985, 1987, and 1998, respectively. He was engaged in the research on cryptography and network security at Agency for Defense Development from 1987 to 1998. In 2002, he joined Department of Computer Engineering at Dongseo University as an associate professor and is now a full professor. He has published more than 250 papers and 50 patents. He has served as a reviewer for many international conferences and journals. His current research interests include security communication systems, side-channel attacks, USN, and RFID security. He is a member of the Korea Institute of Information Security and Cryptology, IEEE Computer Society, and IEEE Information Theory Society, among others.