Microprocessors and Microsystems 45 (2016) 45–55

Contents lists available at ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

QTL: A new ultra-lightweight block cipher

Lang Li[*], Botao Liu, Hui Wang

Department of Computer Science, Hengyang Normal University, Hengyang, 421002, China

ARTICLE INFO

Article history:
Received 6 January 2015
Revised 14 January 2016
Accepted 11 March 2016
Available online 21 March 2016

Keywords:
Lightweight block cipher
Feistel-type structures
Generalized Feistel network
SPNs

ABSTRACT

We propose a new ultra-lightweight block cipher, QTL. The 64 bits block cipher QTL supports 64 and 128 bits keys. To solve the slow diffusion of the traditional Feistel-type structures we have used a new variant of generalized Feistel network structure in design of the QTL. Traditional Feistel-type structures change only half of block messages in an iterative round, but our structure overcomes this disadvantage and changes all block messages. Thus, our structure has the fast diffusion of the Substitution Permutation Networks (SPNs) structures, which improves the security of lightweight block cipher in Feistel-type structures. Moreover, QTL algorithm has the same encryption and decryption processes, so it will occupy less area in resource-constrained applications. Furthermore, to reduce the cost of energy consumption in hardware implementation of the cipher while maintaining security, we decide not to use a key schedule. We show that QTL offers an adequate security level against classic analyses. Our hardware implementation for the 64 and the 128 bits keys modes only require 1025.52 and 1206.52 gate equivalents, respectively. QTL achieves high security and compact implementation in hardware. QTL is one of the most competitive ultra-lightweight block ciphers, which is suitable for extremely constrained devices.

1. Introduction

1.1. Background and motivation

In cryptography, a block cipher is a deterministic algorithm operating on fixed-length groups of bits. Essentially, by limited iteration with block cipher, fixed plaintext is converted to ciphertext. Block cipher algorithm is fast and easy to standardize and facilitate the software and hardware implementations. Hence, in information security block cipher has core values and is a main constrained environment cipher. Recently, with the development of resource-constrained embedded systems, embedded block ciphers are essential primitives for cryptographic applications, such as wireless sensor networks (WSNs), RFID tags, and sensor nodes. These devices manufactures have lower maintenance cost, higher network robustness, stronger self-organization and broader applicability features, which have become a key part of the networking industry. But WSNs and RFID tags are based on wireless networks to transmit information, and this information can easily be acquired, interposed or even destroyed. Under resource-constrained environment, the block cipher has extremely restricted hardware resources. A basic RFID tag chip area generally needs around 1000 to 10,000 gate equivalents (GE) [1]. However, the block cipher algorithm only takes RFID tag chip area around 200 to 2000 GE. Traditional block ciphers use a lot of GE in hardware implementation. For example, low-cost implementation of the AES [2] requires around 3600 GE, which is far more than 2000 GE. Thus, traditional block ciphers are not suitable for resource-constrained environment equipment. In this case, low-cost and efficient lightweight block ciphers became a hot topic of the research.

Nowadays, in the area of lightweight block cipher some of the lightweight block ciphers are proposed, such as PRESENT [3], LBlock [4], TWINE[5], KLEIN [6], MIBS [7], LED [8], PRINCE [9], Piccolo [10], ITUbee [11], EPCBC [12], PRINTcipher [13] and RECTANGLE[14]. Particularly, PRESENT is the most representative of the block cipher. Structures of these lightweight ciphers as like traditional block ciphers are generally developed into two main classical structures: SPNs and Feistel-type structures.

On the one hand, PRESENT was proposed in 2007, based on a SPN, and its hardware requirement is less than 2000 GE. But PRESENT is susceptible to threats of side-channel attacks [15, 16]. Recently, LED was designed for RFID tags, which is also an instantiation of a SPN. It achieves remarkably compact hardware implementation, and explores the role of a non-existent key schedule. KLEIN also adopts a SPN. KLEIN and LED balance the tradeoffs between hardware and software implementations [17].

On the other hand, MIBS and LBock are designed for low resource devices, and ITUbee is a software oriented lightweight block cipher. Those are an instantiation of the Feistel networks. Piccolo is an instantiation of the generalized Feistel networks (GFNs), which

[*] Corresponding author at: 16 henghua Rd.,Zhuihui District, Hengyang, Hunan 421002, China, Tel:+86 15873438955
E-mail address: lilang911@126.com (L. Li).

http://dx.doi.org/10.1016/j.micpro.2016.03.011
0141-9331/© 2016 Elsevier B.V. All rights reserved.

has low-cost in hardware implementation, but the Biclique cryptanalysis can reduce security of Piccolo [18].

The SPN structure is developed using round function on the whole data block [19]. The slow diffusion of the traditional Feistel-type structures will result in some security problems. Therefore, to solve these problems the ciphers in traditional Feistel-type structures commonly demand a lot of rounds in contrast to the ciphers based on SPNs; thus, this increases energy consumption. Nevertheless, compared to SPNs, the traditional Feistel-type structures have more features. Firstly, it has a small and simple round function. Secondly, it has the same program for encryption and decryption processes to reduce decryption implementation cost. Thus, the traditional Feistel-type structures are more suitable for designing lightweight block ciphers.

In this paper, we aim to address the slow diffusion of the lightweight block cipher in traditional Feistel-type structures. It is significant to design a new variant of generalized Feistel network structures with the fast diffusion of the SPNs.

1.2. Our contributions

We propose a new ultra-lightweight block cipher called QTL, which is optimized for extremely resource-constrained devices. QTL is a new variant of generalized Feistel network structure algorithm, which supports 64 bits block with 64 or 128 bits keys. QTL has the fast diffusion of the SPNs, which improves the security of lightweight block cipher in Feistel-type structures. QTL has many numbers of active S-boxes on certain bounds during encryption process. We decide not to use a key schedule to reduce the cost of energy consumption in hardware implementation of the cipher. The ciphers without key schedule are from SPNs structure such as LED and PRINCE, and the only cipher in Feistel networks structure without key schedule is ITUbee.

However, if a lightweight block cipher in Feistel-type structures has no key schedule, it is susceptible to related-key attacks [11]. We have analyzed related-key attacks in Section 4.3. Meanwhile, we demonstrate that QTL offers an adequate security level against classic analyses. Moreover, the area for the 64 and 128 bits keys modes only require 1025.52 and 1206.52 GE. Therefore, QTL achieves high security and compact implementation in hardware.

The organization of the rest paper is as the following. We describe the specification of QTL in Section 2, explain the design rationale for QTL in Section 3, present the result of security analysis in Section 4 along with the performance results of hardware implementation in Section 5, and conclude the paper in Section 6.

2. Specification of QTL

QTL is a 64 bits lightweight block cipher, and the key lengths are 64 bits or 128 bits. The 64 and 128 bits keys modes are referred as QTL-64 and QTL-128, and QTL is a new variant of generalized Feistel network structure block cipher. We choose the number of iterative rounds NR as 16/20 for QTL-64/QTL-128. We will describe QTL encryption algorithm and decryption algorithm in details.

2.1. Notations

We use the following notations in the paper:

X	64 bits plaintext
Y	64 bits ciphertext
$K_{(64)}$	64 bits master key
$K_{(128)}$	128 bits master key
F_1	F_1-function
F_2	F_2-function
AC	AddConstants
P	Permutation operate on 16 bits
S-box	4×4 S-box
S	S-box layer
RT	Round transposing
||	Concatenation of two binary strings
$\oplus$	Bitwise exclusive-OR operation
CON	Round constants

Algorithm 1
The encryption routine of QTL.

```
ENC_NR
Input: $X_{(64)}$, $K_{0(16)}$, $K_{1(16)}$, $K_{2(16)}$, $K_{3(16)}$
Output: $Y_{(64)}$
1:$X_{(64)} \rightarrow X_{0(16)} || X_{1(16)} || X_{2(16)} || X_{3(16)}$
2:for i=1 to (NR-1) do the following
3:  for j=1 to 2 do the following
4:    if(j= =1) then
5:      $X_{1(16)} \leftarrow X_{1(16)} \oplus F_1(K_{0(16)}, X_{0(16)})$, $X_{3(16)} \leftarrow X_{3(16)} \oplus F_2(K_{1(16)}, X_{2(16)})$
6:      $X_{0(16)} || X_{1(16)} || X_{2(16)} || X_{3(16)} \leftarrow X_{1(16)} || X_{0(16)} || X_{3(16)} || X_{2(16)}$
7:    end if
8:    if(j= =2) then
9:      $X_{1(16)} \leftarrow X_{1(16)} \oplus F_1(K_{2(16)}, X_{0(16)})$, $X_{3(16)} \leftarrow X_{3(16)} \oplus F_2(K_{3(16)}, X_{2(16)})$
10:     end if
11:  end for
12:  $X_{0(16)} || X_{1(16)} || X_{2(16)} || X_{3(16)} \leftarrow RT(X_{2(16)} || X_{1(16)} || X_{0(16)} || X_{3(16)})$
13:end for
14:for j=1 to 2 do the following
15:  if(j= =1) then
16:    $X_{1(16)} \leftarrow X_{1(16)} \oplus F_1(K_{0(16)}, X_{0(16)})$, $X_{3(16)} \leftarrow X_{3(16)} \oplus F_2(K_{1(16)}, X_{2(16)})$
17:    $X_{0(16)} || X_{1(16)} || X_{2(16)} || X_{3(16)} \leftarrow X_{1(16)} || X_{0(16)} || X_{3(16)} || X_{2(16)}$
18:  end if
19:  if(j= =2) then
20:    $X_{1(16)} \leftarrow X_{1(16)} \oplus F_1(K_{2(16)}, X_{0(16)})$, $X_{3(16)} \leftarrow X_{3(16)} \oplus F_2(K_{3(16)}, X_{2(16)})$
21:  end if
22:end for
23:$Y_{0(16)} || Y_{1(16)} || Y_{2(16)} || Y_{3(16)} \leftarrow X_{0(16)} || X_{1(16)} || X_{2(16)} || X_{3(16)}$
24:$Y_{(64)} \leftarrow Y_{0(16)} || Y_{1(16)} || Y_{2(16)} || Y_{3(16)}$
25:Return $Y_{(64)}$
```

2.2. Encryption algorithm

The encryption algorithm of QTL has the number of iterative rounds NR. Fig. 1 illustrates the encryption procedure. Then we will explain encryption algorithm steps of QTL in details as below.

The data processing part of QTL consists of NR rounds. ENC_{NR} inputs a 64 bits plaintext data $X \in \{0, 1\}^{64}$, four 16 bits round subkeys $K_{i(16)}$ $(0 \le i \le 3)$, and outputs a 64 bits ciphertext data $Y \in \{0, 1\}^{64}$ (see Algorithm 1). Specifically, the AddRoundKey illustrates the structure of round sub-keys $K_{i(16)}$ in details. ENC_{NR} is defined as below:

$$ENC_{NR}: \begin{Bmatrix} \{0,1\}^{64} \times \{\{0,1\}^{16}\}^4 \rightarrow \{0,1\}^{64} \\ (X_{(64)}, K_{0(16)}, K_{1(16)}, K_{2(16)}, K_{3(16)}) \rightarrow Y_{(64)} \end{Bmatrix}$$

where F_1 and F_2 are 16 bits F-functions, and RT is a 64 bits permutation defined in the following sections. **F-functions.** Round function of QTL consists of two different 16 bits F-functions F_1 and F_2. F_1 and F_2 have the same composition of the structure. We define the input and output of F_1 and F_2 as follows:

$$\begin{Bmatrix} \{0,1\}^{16} \rightarrow \{0,1\}^{16} \\ F_1: (X_i, K_i)| \rightarrow Y_i = S_1(P(S_1(X_i \oplus CON_1 \oplus K_j))) \quad (i = 0 \text{ or } 1) \quad (j = 0 \text{ or} 2) \\ F_2: (X_i, K_i)| \rightarrow Y_i = S_2(P(S_2(X_i \oplus CON_2 \oplus K_j))) \quad (i = 2 \text{ or } 3) \quad (j = 1 \text{ or } 3) \end{Bmatrix}$$

The F_1 and F_2 consist of AddConstants and AddRoundKey where the S_1s and S_2s are separated by a P. And the F-function process is illustrated as: AddConstants → AddRoundKey → S-box layer → P permutation layer → S-box layer (see Fig. 2).

The AddConstants updates a 16 bits data $X_{(16)}$ as follows:

The round constants (*CON*) consist of CON_1 and CON_2. We define the *n*th for $(0 \le n \le NR)$ round constants as the CON_1^n and