CS553 - Term Paper

Group - Cipherfreek

November 25, 2020

1 Abstract

In this paper, we analyse NOEKEON , a block cipher of 128 bit. Subsequently, we study the design choices and address the cipher's resistance to known attacks, followed by the implementation aspects.

2 Introduction

The resistance against cryptanalysis of Noekeon is as good as any block cipher , it can also be implemented efficiently and securely on wide range of platforms and also it's code/circuit compactness is excellent (self-inverse Bit-slice cipher) for which it is especially suitable for platforms, such as Smart Cards, where resources are limited.

3 Design Rationale

NOEKEON is an iterated 128-bit block cipher with 128-bit keys, 16 rounds which consists of several subfunctions.After each round , we add a round constant to prevent round symmetry.

NOEKEON can be used as a one way function , for protection of data integrity, authentication by using it as the core of a MAC algorithm and also for confidentiality by applying some common modes of operation like ECB, CBC, CFB, OFB.

3.1 The Key Schedule

This iterated cipher consists of a round transformation, followed by an output transformation. The round transformation includes XORing a Working Key to the intermediate computation result.Thus, the key schedule consists of the conversion of 128-bit Cipher Key into this Working key.The main agenda behind using this key schedule is to protect the cipher against "related key attacks".The main disadvantage of key schedule is that ,to store the working key, it takes RAM and some cycles to compute .Most importantly, some attacks which use power consumption or radiation reveals the key while the advantage is that , for the computation of cipher, and key schedule ,the same circuit is used .

In order to prevent these , we use a mode called as direct key mode , in which key schedule is not used and raises the fact that the working key is the cipher key , and this is used only where the related key attacks which needs cipher executions with various working Keys that have a minimum of a best-known and in most cases a rigorously chosen relation, can't be mounted.If they can be mounted, we use another mechanism called as key variants in which a key is classified into various keys by XORing different constants to it(if used, we can't use direct key mode). The paradigm to convert the Cipher Key into the Working Key is the cipher only and working key is obtained by providing null string as working key , and NOEKEON to cipher key as input.

3.2 Implementation Attacks

In context of design of any block cipher, the physical protection of key material is difficult. The important feature that allows its secure implementation is that: In every round, Only four of the computations are non-linear , rest of them are linear and the with the use of direct-key mode,the implementation attacks cannot focus on key schedule.

4 Specification

4.1 The State and the Cipher Key

The number of rounds is denoted by Nr.The intermediate result is known as state, on which the transformations are operated, initialized with input , output is obtained from state itself before and after the execution of cipher respectively. The input, output,cipher key consists of one dimensional arrays of 8-bit while the state is of 4 32-bit words, a[0] to a[3].

4.2 The Round Transformation

The pseudo code for different transformations is given below:

```
Round(Key,State,Constant1,Constant2) {
.       State[0] ∧= Constant1;
.       Theta(Key,State);
.       State[0] ∧ = Constant2;
.       Pi1(State);
.       Gamma(State);
.       Pi2(State);
}
```

As we already know, the subfunctions operate on state, In the given two constants in pseudo code, one will always be set to 0;if round transformation is operated on cipher, then Constant2 is set to zero, if it is operated on inverse cipher then Constant1 is set to 0.We will discuss the subfunction in later sections

4.3 Gamma

Gamma is a non-linear function operating on state.

```
Gamma(a){
a[1] ∧ = $\sim a[3] \& \sim a[2]$;
a[0] ∧ = $a[2] \& a[1]$;
tmp = a[3]; a[3] = a[0]; a[0] = tmp;
a[2] ∧ = a[0]∧ a[1]∧ a[3];
a[1] ∧= $\sim a[3] \& \sim a[2]$;
a[0] ∧= a[2]&a[1];
}
```

The Gamma function operates on 32 4-tuples of bits independently which are called as boxes.This operation can be treated as an S-box(given below) which is applied to each of the boxes.

Input:	$a_3a_2a_1a_0$	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Output:	$b_3b_2b_1b_0$	7	A	2	C	4	8	F	0	5	9	1	E	3	D	B	6

4.4 Theta

Theta is a linear function which takes working key k as an argument and operates on state a :

```
Theta(k,a){
temp = a[0]∧ a[2];
temp ∧= temp>>>8 ∧ temp<<<8;
a[1] ∧ = temp;
a[3] ∧ = temp;
a[0] ∧= k[0];
a[1] ∧= k[1];
a[2] ∧= k[2];
a[3] ∧= k[3];
temp = a[1]∧a[3];
temp ∧= temp>>>8 ∧ temp<<<8;
a[0] ∧= temp;
a[2] ∧= temp;
}
```

This comes to the reality that the first and last component mapping of Theta commute.The inverse of Theta is expressed as Theta(k^1, a).Here k^1 is the working key obtained after computing Theta(Null vector,k).

4.5 The Shift Operations

Pi1 and Pi2 are known as shift operations and are inverse to each other.These carry out cyclic shifts of 3 of the 4 phrases over specific offsets.
Here, a is the state.

```
Pi1(a){ a[1] <<<= 1; a[2] <<<= 5; a[3] <<<= 2;}
Pi2(a){ a[1] >>> = 1; a[2] >>>= 5; a[3] >>>= 2;}
```

4.6 The Cipher

Here we discuss the Cipher and its inverse.
The most effective distinction among NOEKEON and its inverse is withinside the computation of the Working Key and the application of the round constants.
In the indirect-key mode, the working key is obtained using the cipher key while in the

direct-key mode, the working key is the cipher key itself.

$$\text{WorkingKey} = \text{CipherKey};$$
$$\text{Noekeon(NullVector,WorkingKey); \{indirect-key mode\}}$$

$$\text{WorkingKey} = \text{CipherKey; \{direct-key mode\}}$$

As we know, the number of rounds=16 $\implies$ Nr=16
The pseudo code for cipher and its inverse is given below:

```
Noekeon(WorkingKey,State) {
For( i=0 ; i<Nr ; i++) Round(WorkingKey,State,Roundct[i],0);
State[0] ∧= Roundct[Nr];
Theta(WorkingKey,State);
}

    InverseNoekeon(WorkingKey,State) {
Theta(NullVector,WorkingKey);
For( i=Nr ; i>0 ; i–) Round(WorkingKey,State,0,Roundct[i]);
Theta(WorkingKey,State);
State[0] ∧= Roundct[0];
}
```

4.7 Round Constants

In the above pseudo code we have used "Roundct" which means the Round Constants which are the form :

$$\text{Roundct[i]} = (\text{‘00’,‘00’,‘00’,RC[i]})$$

We can compute the round constants in two ways:

- We can use the following recursion method to find the value of the round constants RC[1] to RC[Nr]:

$$\text{RC[0]} = \text{0x80};$$
$$\text{if (RC[i]\&0x80 != 0) RC[i+1]} = \text{RC[i]<<1} \wedge \text{0x1B else RC[i+1]} = \text{RC[i]<<1};$$

- the round constants can be computed in reversed order which is similar to linear feedback shift register
  if (RC[i]&0x01 != 0) RC[i-1] = RC[i]>>1 ∧ 0x8D else RC[i-1] = RC[i]>>1;

5 Motivation of design choices

Here, we will motivate the choice of specific transformations and constants:-

5.1 Theta operation

This operations was decided on the following criteria:
i) if the key is zero, then it should be able to use same round transformation in NOEKEON and its inverse.
ii) It should have symmetry and relevant diffusion power.
iii)Description should be simple and should have small number of operation.

Table 2 shows the input output hamming weight distribution table of theta. The distribution of the 2^{16} input-output pairs for one column over the weight of the Hamming input and the weight of the Hamming output is shown in this table.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1							16								
2		8				48				48				16	
3					112				448						
4				348				1152				224			
5			112				3136				1120				
6		48				3928				3984				48	
7	16				3136				7840				448		
8				1152				10556				1152			
9			448				7840				3136				16
10		48				3984				3928				48	
11					1120				3136				112		
12				224				1152				348			
13							448				112				
14		16				48				48				8	
15									16						

5.2 The shift offsets

For both Pi1 and Pi2, the shift offsets have been selected with the following criteria:

- If Pi2 is the inverse of Pi1, then it should be able to use the same round transformations both in NOEKEON and its inverse.
- The offset for 0 is 0.
- Various modulo 8 are the four offsets of Pi1.
- From the set of arrays that maximize diffusion over a single round according to a simple criterion, the array of offsets is selected. The first one, in lexicographical order, was chosen from the resulting set of arrays.

5.3 Gamma

This operation has been designed with the following criteria:

- If key is Zero, then it should be able to use the same round transformation in both NOEKEON and its inverse.
- There should be low number of bitwise boolean operaions.
- The overall non-trivial correlation between input-output is 1/2
- and the overall difference propagation probability is 1/4.

Table below shows the difference propagation probability of Gamma s-box.

	0	1	2	4	8	3	5	6	9	A	C	7	B	D	E	F
0	16	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
1	.	.	.	.	.	.	.	.	.	.	4	.	.	4	4	4
2	.	.	.	2	4	.	2	2	.	.	.	2	4	.	.	.
4	.	.	2	2	2	2	.	2	.	2	2	.	.	2	.	.
8	.	.	4	2	.	4	2	2	.	.	.	2	.	.	.	.
3	.	.	.	2	4	.	2	2	.	.	.	2	4	.	.	.
5	.	.	2	.	2	2	2	.	.	2	.	2	.	.	2	2
6	.	.	2	2	2	2	.	2	.	2	.	.	.	.	2	2
9	.	.	.	.	.	.	.	.	4	.	2	.	4	2	2	2
A	.	.	.	2	.	.	2	2	.	.	2	2	.	2	2	2
C	.	4	.	2	.	.	.	.	2	2	.	2	.	.	4	.
7	.	.	2	.	2	2	2	.	.	2	2	2	.	2	.	.
B	.	.	4	.	.	4	.	.	4	.	.	.	4	.	.	.
D	.	4	.	2	.	.	.	.	2	2	.	2	.	4	.	.
E	.	4	.	.	.	.	2	2	2	2	4	.	.	.	.	.
F	.	4	.	.	.	.	2	2	2	2	.	.	.	.	.	4

Table below shows input output correlations in Gamma s-box.

	0	1	2	4	8	3	5	6	9	A	C	7	B	D	E	F
0	16	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
1	.	-8	-4	.	8	4	.	-4	.	4	.	-4	-4	.	-4	-4
2	.	-4	4	.	.	.	4	-4	-4	-4	-8	.	8	-4	-4	.
4	.	.	.	.	.	-8	.	.	-8	.	.	-8	.	8	.	.
8	.	8	.	.	.	8	8	.	.	.	.	-8	.	.	.	.
3	.	4	.	-8	8	4	-4	.	-4	.	.	4	4	4	.	4
5	.	.	4	.	8	-4	8	4	.	-4	.	4	-4	.	4	-4
6	.	-4	-4	.	.	.	4	4	4	4	-8	.	.	4	4	-8
9	.	.	-4	-8	.	-4	.	4	8	-4	.	-4	4	.	-4	-4
A	.	4	-4	.	.	.	-4	4	-4	4	-8	.	.	-4	4	-8
C	.	.	-8	.	.	.	.	-8	.	-8	.	.	.	.	8	.
7	.	-4	.	-8	-8	4	4	.	-4	.	.	4	-4	4	.	-4
B	.	-4	8	.	.	4	-4	.	4	.	.	-4	4	4	8	-4
D	.	.	-4	8	.	4	.	4	.	-4	.	4	4	8	-4	-4
E	.	-4	-4	.	.	.	4	4	-4	4	8	.	8	-4	4	.
F	.	-4	.	.	.	4	-4	8	-4	-8	.	-4	-4	-4	.	4

5.4 Round Constants

It is added to remove the following symmetry:

- The round transformation process the different boxes in the same way.
- The round transformation is same for all rounds.

5.5 Number of rounds

It is estimated that 16 rounds are enough to prevent from any known shortcut attacks.

6 Strength Against Known Attacks

6.1 Differential and Linear Cryptanalysis

6.1.1 Differential Cryptanalysis

There must be a predictable differential propagation in order for a DC attack to exist:

- over all but few(typically 2 or 3) rounds.
- With a propagation ratio (the relative amount of all input pairs that give rise to the output difference for the given input difference) substantially greater than 2^{-127}.

A differential propagation from a pattern of difference a' to a pattern of difference b' consists of differential trails, where its prop ratio is the sum of the prop ratios of all differential trails with an initial difference a' and the final pattern of difference b'. So as to resist against DC, it is required not to have differential trails with prop ratio higher than 2^{-127}.
For Noekeon, it is guarranted that there are no 4-round differential trials with a prop ratio above 2^{-48}.

6.1.2 Linear Cryptanalysis

If there are stable values of input-output correlation over all but a few (typically 2 or 3) rounds significantly larger than 2^{-64}, LC attacks are possible. Linear trails are composed of an input-output correlation, where their correlation is the sum of the correlation coefficients of all linear trails that have the initial and final selection patterns defined. Linear trail correlation coefficients are signed and their signature depends on the value of the working key bits. It is a necessary requirement that there are no linear trails with a correlation coefficient greater than 2^{-64} to be immune to LC.
For NOEKEON, it is guaranteed that there are no 4-round linear trials with a correlation coefficient above 2^{-24}.

6.1.3 Weight of differential and linear trail

- The prop ratio of a differential trail may be approximated by the product of its active S-boxes' prop ratios.
- The correlation of a linear trail can be approximated by the product of its active S-boxes' input-output correlations.

6.2 Symmetry properties and weak keys as in DES

Care has been taken to eliminate symmetry in the behavior of the cipher, despite the large amount of symmetry. This has been done by adding round constants which are different for each round. This eliminates the possibility of weak keys and semi-weak keys.
For iterative ciphers in which the round transformation is the same for each round, Slide Attacks are applicable. If the round transformation didn't require the inclusion of a round constant, this would be the case for NOEKEON. So, to prevent slide attacks, we should rely on these constants which are different for each round.

6.3 Truncated Differentials

We should not expect truncated differentials because of the bit-oriented nature of NOEKEON.

6.4 Related key attacks

NOEKEON has a key schedule that basically consists of applying the cipher to the Cipher Key as an input and using the output of this operation as a working key to protect against related-key attacks. We expect this to make known Cipher Keys relationships unusable. In addition, the colliding of Cipher Keys giving rise to the same Working Key does not occur due to the bijective nature of the block cipher transformation. In addition, with the same circuit/code used for the cipher itself the key schedule has the advantage that it can be performed.
The key schedule in NOEKEON, however is optional and we assume that by intelligently designing the protocol in which the block cipher is used, related key attacks can easily be avoided. In circumstances where it is not possible to use a similar key, we should use NOEKEON in direct-key mode.

7 Implementation Aspects

NOEKEON is ideal for efficient implementation on 32-bit processors and on dedicated hardware. On 8-bit processors, however it is also fairly successful.

7.1 Flexibility and Parallelism

The round transformation of NOEKEON, with its three stages in Gamma, its three stages in Theta and the two stages of rotation Pi1 and Pi2, has a sequential existence at first sight. The linearity of most of the transformations, however, enables a great deal of flexibility in their implementation:

- The key addition, carried out in the middle of Theta, may be transferred to the beginning of Theta, before Pi2, after Theta, or even after Pi1, at the expense of the corresponding working key measurement, which must be done only once for each use of the Cipher Key.

- In this process, the first and last steps of Theta can be performed simultaneously and the rotations can be combined.

7.2 Hardware Suitability

The fact that NOEKEON consists of a relatively simple round transformation being repeatedly applied and the fact that the main schedule makes use of the same circuit makes it possible to implement it in a surprisingly limited number of gates. In addition, NOEKEON's inverse operation can be done using the same circuit.

640 XOR, 64 AND and 64 NOR (two-input) gates are the gate count of the round transformation:

1. Theta: 416 XOR gates
   - (a) First part of Theta: 32+16+32+32+32: 144 XOR gates
   - (b) Key addition: 128 XOR gates
   - (c) Last part of Theta: 144 XOR gates
2. Round constant addition and computation (2 LFSRs): less than 32 XOR gates
3. Pi1 and Pi2: none
4. Gamma: 224 XORs, 64 AND and 64 NOR gates
   - (a) First part of Gamma: 32 AND, 32 NOR and 64 XOR gates
   - (b) Middle part of Gamma: 96 XOR gates
   - (c) Last part of Gamma: 32 AND, 32 NOR and 64 XOR gates

By directly hard-wiring it or by feeding the intermediate result between the application of Theta and Pi1 to the output, the additional Theta that forms the output transformation can be implemented. This "hardware Theta" can be used in both cases to compute the key for the inverse cipher needed.

The number of gates in the critical path defines the speed of the clock in a hardware circuit. Theta can be implemented with three XOR gates in depth, Gamma can be implemented with four XOR gates in depth, and one AND (or NOR) gate. It is possible to hardwire the Pi1 and Pi2, resulting in a gate delay of 7 XOR gates, 1 AND gate and possibly a multiplexer to allow input loading.

The fact that the circuit is so lightweight leaves more space to defend against execution attacks for potential additional circuitry.

7.3 Software Suitability

7.3.1 8-bit Processors

It is easy to divide all bit-wise Boolean operations operating on 32-bit words into four operations on 8-bit words. The combination of shift operations of the 8-bit words and bit-wise Boolean operations will enforce the cyclic shifts of 32-bit words. Theta's cyclic shift operations do not require explicit coding on an 8-bit processor, so only the 6 Pi1 and Pi2 rotations have to be implemented. The code size is very small due to the fact that the round transformation is the same for each round and that the main schedule uses the same round transformation.

A direct-key mode implementation needs only 16 bytes to keep the state, one byte to keep track of the round number, and a few bytes to keep temporary variables (say 3). If the indirect-key mode is introduced, the Working Key requires 16 more bytes to be stored.

7.3.2 32-bit Processors

7.3.2.1 ARM7

Using the ARM Software Development Toolkit from Advanced RISC Machine Ltd, we programmed NOEKEON in direct-key mode on the ARM7 RISC microprocessor.

The ARM processor is especially suitable for the NOEKEON implementation. Indeed, it has an orthogonal instruction set and accepts one target register and two source registers for all binary operations, one of which can be rotated in the same loop. In addition, a lot of job records are open.This then enables a very compact and effective implementation of a round in which either Gamma or Theta are combined with Pi1 and Pi2 in the computation. At the start of Theta, we also transfer the application of the key, and hence all encryption and decryption processes have the same execution time.

The statistics obtained for a simple implementation and two optimised NOEKEON implementations in direct-key mode in the ARM7 assembly language are shown in Table below:

	code size	NOEKEON	$NOEKEON^{-1}$	bit rate @ 28.56MHz
Straightforward	428 bytes	1247 cycles	1271 cycles	2.9 Mbit/s
Optimised for size	332 bytes	712 cycles	712 cycles	5.1 Mbit/s
Optimised for speed	3688 bytes	475 cycles	475 cycles	7.7 Mbit/s

The key set-up time can be calculated as being equivalent to the encryption time for NOEKEON in indirect-key mode. The last column gives the bit rate for smart cards with a clock multiplier, assuming a frequency of 28.56MHz obtained from the normal frequency of 3.57MHz.

Regarding the use of RAM, if both the state and the key are forwarded through registers, no byte of memory is needed for both optimised implementations.

7.3.2.2 Pentium

Using Microsoft Visual C/C++ 6.0 under Windows NT 4.0, we programmed NOEKEON in direct-key mode on the Intel Pentium II processor.

The statistics obtained for early implementation in assembly language are shown in Table below using an interface adapted to a low level of endianness and implementing only simple rules to improve intrinsic parallelism. It is likely that more optimizations are possible (e.g. using the MMX instruction set) but have not been explored here. These statistics were obtained on a 500Mhz-Pentium II processor, but for the Pentium 200MHz reference platform, the speed figures are shown.

	NOEKEON	$NOEKEON^{-1}$	bit rate @ 200MHz
Pentium II – 200 MHz	525 cycles	525 cycles	49 Mbit/s

In the case of the indirect-key mode, the key setup efficiency is close to that of the encryption mode.

7.4 Protection against Implementation Attacks

We have taken into account in the design of the cipher the fact that it must be feasible to enforce it in such a way that it offers security against implementation attacks such as timing attacks and attacks from power analysis.

By coding the cipher as a fixed sequence of instructions, one can defend against timing attacks and simple power analysis. This is straightforward, as NOEKEON only consists of a set sequence of simple operations.

Differential Power Analysis (DPA) security [KoJaJu98] can be obtained through the implementation of multiple mechanisms, ideally at the same time. State splitting is considered one of the processes. In [GoPa99, CJRR99], this approach was suggested and consists of splitting the state into two pieces (one of which is supposed to be random), giving the actual state value to XORed. This approach removes any association between values within the machine and any intermediate computational outcome of the cypher if properly implemented, removing the possibility of first-order DPA. We have shown in [DaPeVa2000] how this can be extended to BASEKING. It turns out that the calculations can be performed separately on the two parts of the state for linear operations. In non-linear operations, there is an extra overhead in which operations must be done using parts of the two split states.

The mechanisms mentioned in [DaPeVa2000] apply equally well to NOEKEON's non-linear pieces, which are actually of the same type: by means of an XOR, the addition of an AND or NOR) of two terms to a third word. The number of such operations is only 4 per round, reducing to a minimum the extra overhead.

We programmed an anti-DPA variant of NOEKEON in direct-key mode that implements the discussed mechanism on the ARM7 RISC microprocessor. We have not configured this execution as deeply as we did in Section 7.3.2.1. The table below compares the results obtained for this version with the straightforward implementation of NOEKEON given in Section 7.3.2.1 in order to provide a meaningful comparison of the implementation cost.

	code size	NOEKEON	$NOEKEON^{-1}$	bit rate @ 28.56MHz
Straightforward	428 bytes	1247 cycles	1271 cycles	2.9 Mbit/s
Anti-DPA	792 bytes	2145 cycles	2254 cycles	1.7 Mbit/s

We see that for the anti-DPA version, the implementation cost is 364 bytes for the code size, and an average of 940 cycles for the execution time. Although this almost doubles the cost of the classical implementation, we find this to be a very appealing approach for countering DPA attacks.

In order to counter higher-order DPA attacks, the mechanisms described above can be implemented multiple times, say n times. For small n, as a function of n, the implementation cost increases approximately linearly.

8 Strengths and Advantages of NOEKEON

NOEKEON has a range of strengths and benefits that make it very special compared to other block ciphers.
NOEKEON:

- In specialised hardware implementations, it is ultra compact and quick;
- Enables successful program implementations that are DPA-resistant;
- In program implementations, it has very low RAM requirements;
- Has compact code in implementations of software;
- Runs on a large range of platforms efficiently;
- It has a style that is so clean and clear that an average person will memorise it.

Reference 1 Reference 2 Reference 3