NORX8 and NORX16: Authenticated Encryption for Low-End Systems

Jean-Philippe Aumasson[1], Philipp Jovanovic[2] and Samuel Neves[3]

[1]Kudelski Security, Switzerland
jeanphilippe.aumasson@gmail.com

[2]University of Passau, Germany
jovanovic@fim.uni-passau.de

[3]University of Coimbra, Portugal
sneves@dei.uc.pt

***Abstract*—This paper presents NORX8 and NORX16, the 8-bit and 16-bit versions of the authenticated cipher NORX, one of the CAESAR candidates. These new versions are better suited for low-end systems—such as "internet of things" devices—than the original 32-bit and 64-bit versions: whereas 32-bit NORX requires 64 bytes of RAM or cache memory, NORX8 and NORX16 require just 16 and 32 bytes, respectively. Both of the low-end variants were designed to retain the security properties of the initial NORX and be fast on small CPUs.**

***Keywords*-authenticated encryption, lightweight, CAESAR**

I. Introduction

NORX [1] is a family of authenticated ciphers: given a secret key, a nonce, a message, and (optionally) associated data, a NORX cipher returns an encrypted message and an authentication tag. NORX was submitted to the CAESAR competition [2] in 2014, and has been analysed with respect to its core algorithm as well as to its mode of operation [3], [4], and is not known to have cryptographic weaknesses.

The submission to CAESAR included algorithms based either on 32-bit or on 64-bit word arithmetic, denoted as NORX32 and NORX64, respectively. This paper describes two new variants of NORX designed for low-end systems: NORX8 and NORX16. These are based on 8-bit and 16-bit words, and require respectively $1/4$ and $1/2$ the memory as the previously smallest NORX instance. We designed NORX8 and NORX16 to offer a low-memory, secure, and fast enough authenticated cipher for resource-constrained systems.

II. Specification

This section is a succinct specification of the new NORX instances and although it is basically self-contained, we refer to the CAESAR submission document [1] for detailed documentation and design rationale.

A. Generalities

In this work, we introduce two new classes of the NORX family:

1) NORX8, with word size $w = 8$, tag size $t \leq 80$, and number of rounds $1 \leq l \leq 63$.
2) NORX16, with word size $w = 16$, tag size $t \leq 96$, and number of rounds $1 \leq l \leq 63$.

Instances from these new classes are denoted by NORXw-l-p-t. We assume that the parallelism degree p is set to 1 (fully serial versions)—although parallel versions could be constructed, we do not expect relevant use cases.

1) Encryption Interface: NORX8 and NORX16 encryption take as input keys K of $k = 80$ and $k = 96$ bits, respectively, a nonce N of $n = 32$ bits, and a datagram $A \parallel M \parallel Z$ where, A is a *header*, M a *message*, and Z a *trailer*. $|A|$, $|M|$, and $|Z|$ are allowed to be 0. NORX encryption produces a ciphertext C, with $|C| = |M|$, and an *authentication tag* T.

2) Decryption Interface: NORX decryption is similar to encryption: Besides K and N, it takes as input a datagram $A \parallel C \parallel Z$, where A and Z denote header and trailer, and C the *encrypted payload*, with $|A|$, $|C|$, and $|Z|$ may be 0. The last input is an authentication tag T. Decryption either returns failure, upon failed verification of the tag, or produces a plaintext M of the same size as C if the tag verification succeeds.

B. Layout Overview

Like the original NORX versions, the new variants are based on the *monkeyDuplex construction* [5], [6]. An overview of the layout is given in Figure 1.

The round function F is a permutation of $b = r + c$ bits, where b is called the *width*, r the *rate* (or block length), and c the *capacity*. We call F a *round* and F^l denotes its l-fold iteration. The internal state S of

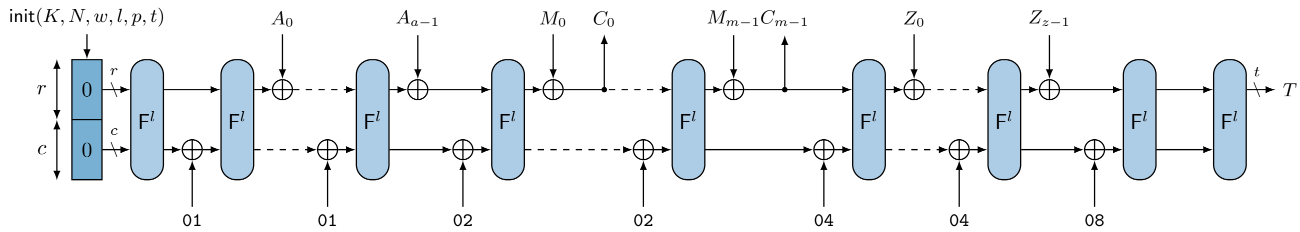

Fig. 1. Layout of NORX.

NORX is viewed as a concatenation of 16 words, i.e. $S = s_0 \parallel \cdots \parallel s_{15}$, which are conceptually arranged in a 4×4 matrix. The so-called *rate words* are used for data block injection which are $s_0, \ldots, s_4$ for NORX8 and $s_0, \ldots, s_7$ for NORX16. The *capacity words* on the other hand are unchanged during data processing and ensure the security of the scheme. These are $s_5, \ldots, s_{15}$ and $s_8, \ldots, s_{15}$ for NORX8 and NORX16, respectively. Proposals for concrete parameters are given in Table I. We consider NORX8 and NORX16 with $l = 4$ to be the default instances.

TABLE I
PROPOSED PARAMETER COMBINATIONS OF THE LIGHTWEIGHT NORX VARIANTS.

w	l	b	r	c	k	n	t
8	4 or 6	128	40	88	80	32	80
16	4 or 6	256	128	128	96	32	96

C. The Round Function F

F processes a state S by first transforming its four columns with

$$\mathsf{G}(s_0, s_4, s_8, s_{12}) \quad \mathsf{G}(s_1, s_5, s_9, s_{13})$$
$$\mathsf{G}(s_2, s_6, s_{10}, s_{14}) \quad \mathsf{G}(s_3, s_7, s_{11}, s_{15})$$

and then transforming its four diagonals with

$$\mathsf{G}(s_0, s_5, s_{10}, s_{15}) \quad \mathsf{G}(s_1, s_6, s_{11}, s_{12})$$
$$\mathsf{G}(s_2, s_7, s_8, s_{13}) \quad \mathsf{G}(s_3, s_4, s_9, s_{14})$$

Those two operations are called *column step* and *diagonal step*, as in BLAKE2 [7] and NORX [1], [8]. The permutation G transforms four words a, b, c, d by computing (top-down, left-to-right):

1. $a \longleftarrow (a \oplus b) \oplus ((a \wedge b) \ll 1)$
2. $d \longleftarrow (a \oplus d) \ggg r_0$
3. $c \longleftarrow (c \oplus d) \oplus ((c \wedge d) \ll 1)$
4. $b \longleftarrow (b \oplus c) \ggg r_1$
5. $a \longleftarrow (a \oplus b) \oplus ((a \wedge b) \ll 1)$
6. $d \longleftarrow (a \oplus d) \ggg r_2$
7. $c \longleftarrow (c \oplus d) \oplus ((c \wedge d) \ll 1)$
8. $b \longleftarrow (b \oplus c) \ggg r_3$

The rotation offsets (r_0, r_1, r_2, r_3) are $(1, 3, 5, 7)$ for NORX8, and $(8, 11, 12, 15)$ for NORX16. They were chosen such that similar performance and security goals are achieved as in the case of NORX32 and NORX64 [1], [8]. In particular, full diffusion is provided after F^2 in both cases.

D. Encryption and Tag Generation

NORX encryption can be divided into three main phases: *initialisation*, *message processing*, and *tag generation*. Processing of a datagram $A \parallel M \parallel Z$ is done in up to three steps: *header processing*, *payload processing*, and *trailer processing*. The number of steps depends on whether A, M, or Z are empty or not. NORX skips processing phases of empty message parts. For example, in the simplest case when $|A| = |Z| = 0$, $|M| > 0$, message processing is done in one step, since only the payload M needs to be encrypted and authenticated.

Below, we first describe the padding and domain separation rules, then each of the aforementioned phases.

1) Padding: NORX uses the *multi-rate padding* [6], defined by $\mathsf{pad}_r : X \longmapsto X \parallel \mathtt{1}\mathtt{0}^q\mathtt{1}$ with bitstrings X and $\mathtt{1}\mathtt{0}^q\mathtt{1}$, and $q = (-|X| - 2) \bmod r$. This extends X to a multiple of the rate r and guarantees that the last block of $\mathsf{pad}_r(X)$ differs from the all-zero block $\mathtt{0}^r$.

2) Domain Separation: NORX performs domain separation by XORing a *domain separation constant* to the least significant byte of s_{15} each time before the state is transformed by the permutation F^l. Distinct constants are used for the different phases of message processing and tag generation. Table II gives the specification of those constants and Figure 1 illustrates their integration into the state of NORX.

TABLE II
DOMAIN SEPARATION CONSTANTS.

header	payload	trailer	tag
01	02	04	08

3) Initialisation: This phase processes a secret key K, a nonce N and the parameters w, l, p, and t. For $w = 8$, we have $K = k_0 \parallel \cdots \parallel k_9$ and $N = n_0 \parallel \cdots \parallel n_3$ of sizes 80 and 32 bits, respectively. For $w = 16$, we have $K = k_0 \parallel \cdots \parallel k_5$ and $N = n_0 \parallel n_1$ of sizes 96 and 32 bits, respectively. First, the state $S = s_0 \parallel \cdots \parallel s_{15}$ is initialised. For $\mathsf{NORX}8$ this is done by

$$\begin{pmatrix} s_0 & s_1 & s_2 & s_3 \\ s_4 & s_5 & s_6 & s_7 \\ s_8 & s_9 & s_{10} & s_{11} \\ s_{12} & s_{13} & s_{14} & s_{15} \end{pmatrix} \longleftarrow \begin{pmatrix} n_0 & n_1 & n_2 & n_3 \\ k_0 & k_1 & k_2 & k_3 \\ k_4 & k_5 & k_6 & k_7 \\ k_8 & k_9 & u_{14} & u_{15} \end{pmatrix}$$

For $\mathsf{NORX}16$ state initialisation is

$$\begin{pmatrix} s_0 & s_1 & s_2 & s_3 \\ s_4 & s_5 & s_6 & s_7 \\ s_8 & s_9 & s_{10} & s_{11} \\ s_{12} & s_{13} & s_{14} & s_{15} \end{pmatrix} \longleftarrow \begin{pmatrix} n_0 & n_1 & k_0 & k_1 \\ k_2 & k_3 & k_4 & k_5 \\ u_8 & u_9 & u_{10} & u_{11} \\ u_{12} & u_{13} & u_{14} & u_{15} \end{pmatrix}$$

The constants can be computed in both cases through

$$(u_0, \ldots, u_{15}) \longleftarrow \mathsf{F}^2(0, \ldots, 15)$$

using the respective variants of F^2. Note, however that only a particular subset of the generated constants is used: u_{14} and u_{15} for $\mathsf{NORX}8$ and $u_8, \ldots, u_{15}$ for $\mathsf{NORX}16$. Afterwards, the parameters w, l, p, and t are integrated into the state S by XORing them to s_{12}, s_{13}, s_{14}, and s_{15}, respectively. Finally, S is updated with F^l.

4) Message Processing: Message processing is the main phase of NORX encryption or decryption.

a) Header Processing: If $|A| = 0$, this step is skipped, otherwise let $\mathsf{pad}_r(A) = A_0 \parallel \cdots \parallel A_{a-1}$ denote the padded header data, with r-bit sized header blocks $A_i = a_{i,0} \parallel \cdots \parallel a_{i,r/w-1}$ and $0 \leq i \leq a-1$. Then A_i is processed by:

$$\begin{aligned} s_{15} &\longleftarrow s_{15} \oplus \mathtt{01} \\ S &\longleftarrow \mathsf{F}^l(S) \\ s_j &\longleftarrow s_j \oplus a_{i,j}, \quad \text{for} \quad 0 \leq j \leq r/w - 1 \end{aligned}$$

b) Payload Processing: If $|M| = 0$, this step is skipped. Otherwise, payload data is padded using the multi-rate padding and then encrypted. Let $\mathsf{pad}_r(M) = M_0 \parallel \cdots \parallel M_{m-1}$. To encrypt $M_i = m_{i,0} \parallel \cdots \parallel m_{i,r/w-1}$ and get a new ciphertext block $C_i = c_{i,0} \parallel \cdots \parallel c_{i,r/w-1}$ the following steps are executed

$$\begin{aligned} s_{15} &\longleftarrow s_{15} \oplus \mathtt{02} \\ S &\longleftarrow \mathsf{F}^l(S) \\ s_j &\longleftarrow s_j \oplus m_{i,j}, \text{ for } 0 \leq j \leq r/w - 1 \\ c_{i,j} &\longleftarrow s_j \end{aligned}$$

for $0 \leq i < m-1$. For $i = m-1$, the procedure is almost the same, but only a truncated ciphertext block is created such that C has the same length as (unpadded) M. In other words, padding bits are never written to C.

c) Trailer Processing: Trailer data Z is processed in a similar way as header data. The only difference is that the domain separation constant `04` is used instead of `02`, see Table II.

5) Tag Generation: Computation of the authentication tag T is handled slightly different for $\mathsf{NORX}8$ and $\mathsf{NORX}16$. Both variants first execute the following steps:

$$\begin{aligned} s_{15} &\longleftarrow s_{15} \oplus \mathtt{08} \\ S &\longleftarrow \mathsf{F}^l(S) \\ S &\longleftarrow \mathsf{F}^l(S) \\ T &\longleftarrow T \parallel s_0 \parallel \cdots \parallel s_{r/w-1} \end{aligned}$$

While the tag can be extracted at once for $\mathsf{NORX}16$, this is not possible for $\mathsf{NORX}8$ in most cases as the rate only has a size of 40 bits. Thus, $\mathsf{NORX}8$ performs the following operations for each block required after the first:

$$\begin{aligned} s_{15} &\longleftarrow s_{15} \oplus \mathtt{08} \\ S &\longleftarrow \mathsf{F}^l(S) \\ T &\longleftarrow T \parallel s_0 \parallel \cdots \parallel s_{r/w-1} \end{aligned}$$

In summary, $3l$ rounds are necessary to extract an 80-bit tag for $\mathsf{NORX}8$.

E. Decryption and Tag Verification

NORX decryption mode is similar to the encryption mode. The only two differences are described below.

1) Message Processing: Processing header A and trailer Z of $A \parallel C \parallel Z$ is done in the same way as for encryption. Decryption of the encrypted payload C is achieved as follows:

$$\begin{aligned} s_{15} &\longleftarrow s_{15} \oplus \mathtt{02} \\ S &\longleftarrow \mathsf{F}^l(S) \\ m_{i,j} &\longleftarrow s_j \oplus c_{i,j}, \quad \text{for} \quad 0 \leq j \leq r/w - 1 \\ s_j &\longleftarrow c_{i,j} \end{aligned}$$

Like in encryption, as many bits are extracted and written to M as unpadded encrypted payload bits.

2) Tag Verification: This step is executed after tag generation. Let T and T' denote the *received* and the *generated tag*. If $T = T'$, tag verification succeeds; otherwise it fails, the decrypted payload is discarded and an error is returned.

III. HARDWARE REQUIREMENTS

In this section, we present preliminary estimates of the required gate-equivalents (GE) when realising NORX8 and NORX16 in hardware. We assume that the ciphers are implemented for a technology that needs 7 GE per D-flip-flop, 3 GE per XOR and 2 GE per AND. To store the states of NORX8 and NORX16 a total of 128 and 256 D-flip-flops are necessary amounting to $7 \cdot 128 = 896$ GE and $7 \cdot 256 = 1792$ GE, respectively. Implementing G requires 12 XORs, 4 ANDs, and some bit shifts for $\ll 1$ and cyclic rotations $\ggg r$. Bit shifts can be ignored for GE estimations since they are realised through re-wiring. The 8- and 16-bit G functions therefore need $(3 \cdot 12 + 2 \cdot 4) \cdot 8 = 44 \cdot 8 = 352$ GE and $44 \cdot 16 = 704$ GE, respectively. The difference between the column and diagonal steps is also just a re-wiring of state elements and therefore requires no additional GE. Absorption of r-bit data blocks is realised through bitwise XOR. Thus, an additional number of $3 \cdot 40 = 120$ GE (NORX8) and $3 \cdot 128 = 384$ GE (NORX16) are necessary. In summary, the lower bounds for hardware implementations can be estimated by $896 + 352 + 120 = 1368$ GE for NORX8 and $1792 + 704 + 384 = 2880$ GE for NORX16.

IV. SECURITY GOALS

NORX8 and NORX16 follow the same security paradigms like their bigger siblings NORX32 and NORX64, i.e. it is assumed that adversaries are *nonce-respecting* and that nothing but an error is returned on a tag verification failure. The security of the schemes is limited by key and tag sizes of $k = t = 80$ bits (NORX8) and of $k = t = 96$ bits (NORX16) for our proposed instances, see Table I. We set the *usage exponent* e to 24 (NORX8) and 32 (NORX16) which limits the number of initialisations to 2^e before a given key has to be changed. According to the results for keyed sponge constructions presented in [9], NORX8 and NORX16 are expected to indeed achieve the generic security bounds of 80 and 96 bits, respectively.

V. PRELIMINARY CRYPTANALYSIS

We conducted a preliminary analysis of certain properties of NORX with $w \in \{8, 16\}$, and used similar techniques as presented in [3] for $w \in \{32, 64\}$. The rotation offsets $(1, 3, 5, 7)$, for $w = 8$, and $(8, 11, 12, 15)$, for $w = 16$, of the round function were chosen such that F^2 provides full diffusion, as already mentioned in Section II-C. Additionally, the above rotation offsets ensure that the function G has no fixed points, which are values that satisfy $\mathsf{G}(a, b, c, d) = (a, b, c, d)$, except for the trivial one $\mathsf{G}(0, 0, 0, 0) = (0, 0, 0, 0)$. As a consequence, F^l also has no fixed points except for the all-zero point. Our SMT/SAT-solver-based differential cryptanalysis of F^2 showed that the probabilities of differential characteristics are upper-bounded by 2^{-29} ($w = 8$) and 2^{-37} ($w = 16$). Moreover, the differential probabilities for F during initialisation are upper-bounded by 2^{-31} ($w = 8$) and 2^{-53} ($w = 16$) for the case where only the nonce words can be modified. In this scenario, 7 rounds of initialisation can be roughly upper-bounded by $2^{-31+3\cdot(-29)} = 2^{-118}$ (NORX8) and $2^{-53+3\cdot(-37)} = 2^{-164}$ (NORX16), respectively.

VI. CONCLUSION

In this work, we presented NORX8 and NORX16, the two latest members of the NORX family of authenticated encryption schemes targeted at low-end systems. The reference source code for both new variants of NORX will be released on the official NORX website [10]. The NORX family of authenticated encryption algorithms is free for everyone to use and we have neither filed nor have we planned to file a patent application for the algorithm.

REFERENCES

[1] J.-P. Aumasson, P. Jovanovic, and S. Neves, "NORX," in *CAESAR Proposal*, 2014.

[2] "CAESAR — Competition for Authenticated Encryption: Security, Applicability, and Robustness," 2014, http://competitions.cr.yp.to/caesar.html.

[3] J.-P. Aumasson, P. Jovanovic, and S. Neves, "Analysis of NORX," in *LATINCRYPT 2014*, 2014, to appear.

[4] P. Jovanovic, A. Luykx, and B. Mennink, "Beyond $2^{c/2}$ Security in Sponge-Based Authenticated Encryption Modes," in *Advances in Cryptology — ASIACRYPT 2014*, ser. Lecture Notes in Computer Science, T. Iwata and P. Sarkar, Eds., vol. 8873. Springer, 2014, pp. 85–104.

[5] G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche, "Permutation-based Encryption, Authentication and Authenticated Encryption," presented at DIAC 2012, 05–06 July 2012, Stockholm, Sweden.

[6] ——, "Duplexing the Sponge: Single-Pass Authenticated Encryption and Other Applications," in *SAC 2011*, ser. LNCS, A. Miri and S. Vaudenay, Eds., vol. 7118. Springer, 2011, pp. 320–337.

[7] J.-P. Aumasson, S. Neves, Z. Wilcox-O'Hearn, and C. Winnerlein, "BLAKE2: Simpler, Smaller, Fast as MD5," in *ACNS 2013*, ser. LNCS, M. Jacobson, M. Locasto, P. Mohassel, and R. Safavi-Naini, Eds., vol. 7954. Springer, 2013, pp. 119–135.

[8] J.-P. Aumasson, P. Jovanovic, and S. Neves, "NORX: Parallel and Scalable AEAD," in *ESORICS 2014*, ser. LNCS, M. Kutylowski and J. Vaidya, Eds., vol. 8713. Springer, 2014, pp. 19–36.

[9] E. Andreeva, J. Daemen, B. Mennink, and G. V. Assche, "Security of Keyed Sponge Constructions Using a Modular Proof Approach," in *FSE 2015*, to appear.

[10] "Official website of NORX," 2014, https://www.norx.io.