STRIBOB: Authenticated Encryption from GOST R 34.11-2012 LPS Permutation

(Extended Abstract)

Markku-Juhani O. Saarinen *
Norwegian University of Science and Technology
mjos@item.ntnu.no

Abstract. Authenticated encryption algorithms protect both the confidentiality and integrity of messages with a single processing pass. We show how to utilize the $L \circ P \circ S$ transform of the Russian GOST R 34.11-2012 standard hash "Streebog" to build an efficient, lightweight algorithm for Authenticated Encryption with Associated Data (AEAD) via the Sponge construction. The proposed algorithm "StriBob" has attractive security properties, is faster than the Streebog hash alone, twice as fast as the GOST 28147-89 encryption algorithm, and requires only a modest amount of running-time memory. StriBob is a Round 1 candidate in the CAESAR competition.

Keywords: StriBob, Authenticated Encryption, GOST R 34.11-2012, Streebog, Sponge Construction, SpongeWrap, MonkeyDuplex, CAESAR.

1 Introduction

Since January 1, 2013, the Russian Federation has required the use of new GOST R 34.11-2012 hash algorithm in digital signatures [10, 13]. This hash was designed apparently in response to cryptographic weaknesses reported in the previous hash standard GOST R 34.11-94 [12, 18]. The 2012 standard, dubbed STREEBOG, has superficial similarities to the old 1994 standard but also features clearly AES-inspired design elements [8, 14, 20].

In contrast to the Russian approach, the U.S. NIST selected a novel Sponge-based design, KECCAK, as the basis of future SHA-3 hash function standard [2, 7]. Sponge hashes diverge from more traditional Davies-Mayer [17] (SHA) and derived HAIFA [5] (STREEBOG) constructions in that they are based on a single keyless permutation π rather on a keyed permutation which can seen as a special-purpose block cipher.

Furthermore, Sponge permutations can be used to achieve Authenticated Encryption in straightforward manner (see Figure 1) [1, 3]. Here both the confidentiality and integrity of a

*This work was carried out during the tenure of an ERCIM "Alain Bensoussan" Fellowship Programme.

message can be guaranteed with a single processing pass, without the use of a separate encryption algorithm such as GOST 28147-89 [11]. This has clear advantages for performance and implementation footprint, which are especially useful in limited-resource applications. Even full-featured secure communications suites can be constructed from a single permutation [22].

In this note we show how to construct a modern lightweight AEAD algorithm from the core of the GOST R 34.11-2012 STREEBOG hash. Our proposal, "STRIBOB" is faster than the STREEBOG hash alone, has good security arguments, and runs on low-resource platforms. The proposal is a first round candidate in the U.S. NIST - funded CAESAR Competition [21, 23].

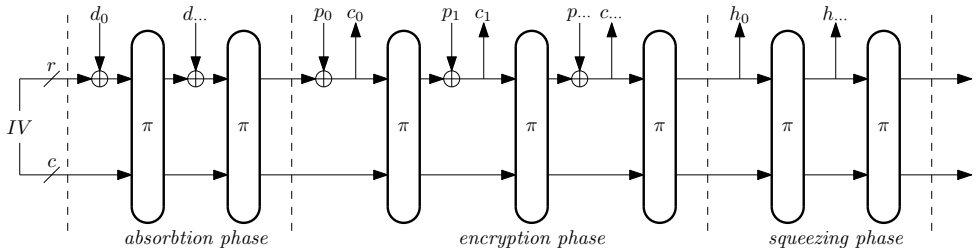

Figure 1: A simplified view of a generic Sponge-based AEAD. First the padded Secret Key, Nonce, and Associated Authenticated Data - all represented by d_u words - are "absorbed" or mixed into the Sponge state. The π permutation is then used to also encrypt data p_i into ciphertext c_i (or vice versa) and finally to "squeeze" out a Message Authentication Code h_i.

2 Structure of GOST R 34.11-2012

STREEBOG produces either a 256-bit or a 512-bit hash from a bit string of arbitrary size using the Merkle-Damgård [9, 19] iterative method without any randomization.

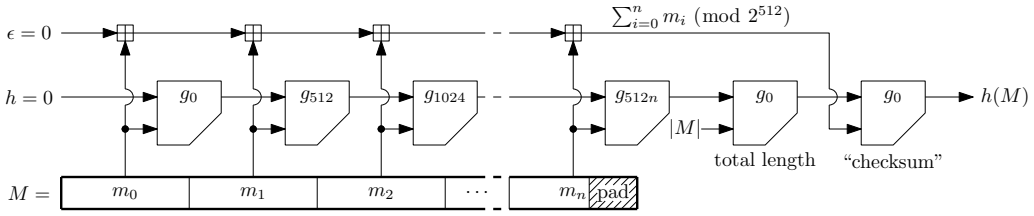

Figure 2: Operation of STREEBOG with 512-bit output. For 256-bit hashes, the initial h value is changed to `0x010101..01` and the output $h(M)$ is truncated to 256 bits.

Figure 2 gives an overview of the hashing process. Padded message M is processed in 512-bit blocks $M = m_0 \mid m_1 \mid \cdots \mid m_n$ by a compression function $h' = g_N(h, m_i)$. The chaining variable h also has 512 bits and N denotes the index bit offset of the input block. After the last message block, there are finalization steps involving two invocations of the compression function, first on the total bit length of input, and then on checksum ϵ, which is computed over all input blocks $\bmod 2^{512}$.

2.1 STREEBOG Compression Function $g_N(h, m)$

The compression function $h' = g_N(h, m)$ takes in a chaining variable h, message block m, a position index variable N, and produces a new chaining value h'. The compression function is built from a keyless 512-bit nonlinear permutation LPS and 512-bit vector XOR operations. The compression function has 12 rounds and performs a total of 25 invocations of LPS :

$$\begin{aligned}
[K_1, X_1] &= [\ \mathsf{LPS}(h \oplus N),\ m\ ] \\
[K_{i+1}, X_{i+1}] &= [\ \mathsf{LPS}(K_i \oplus C_i), \mathsf{LPS}(X_i \oplus K_i)\ ] \ \text{ for } \ 1 \le i \le 12 \\
g_N(h, m) &= K_{13} \oplus X_{13} \oplus h \oplus m.
\end{aligned}$$

Figure 3 shows the structure of g. We can view it as a two-track substitution-permutation network where input value $h \oplus N$ and a set of 12 round constants C_i is used to key (via K_i) another substitution-permutation network operating on h . The outputs of the two tracks are finally XOR'ed together with original values of h and m. We note that h together with offset N uniquely defines all K_i subkey values for each invocation of g.

Computation of $g_N(h, m)$ requires at least 3×512 bits or 192 bytes of temporary storage, which may be preventive for ultra light-weight applications. Furthermore the $\mod 2^{512}$ summation for ϵ must be performed concurrently to the compression function.

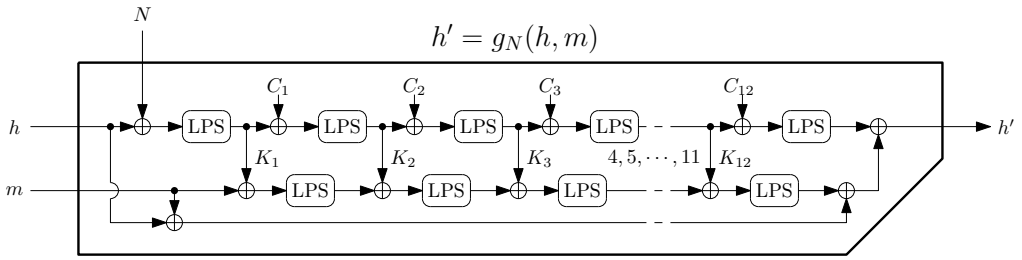

Figure 3: STREEBOG compression function. All data paths, inputs, and outputs are 512-bit vectors. Here the $\oplus$ symbol denotes the XOR operation between two 512-bit vectors.

2.2 The LPS Transform

The LPS transform is a 512-bit keyless permutation, and forms the cryptographic core of STREEBOG and STRIBOB. It is depicted in Figure 4 and consists of three stages. We abbreviate the composite function $L(P(S(x))) = (L \circ P \circ S)(x)$ as LPS . The components are:

S **Nonlinear substitution.** An 8×8 - bit S-Box is applied to each of the 64 bytes of data.

P **Permutation.** A byte transpose where the 8×8 byte matrix is reflected over its main diagonal (rows written as columns or columns written as rows).

L **Linear transform.** Finally the eight 64-bit words are individually subjected to a vector-matrix multiplication with a 64×64 - bit matrix in $\mathbb{F}_2$

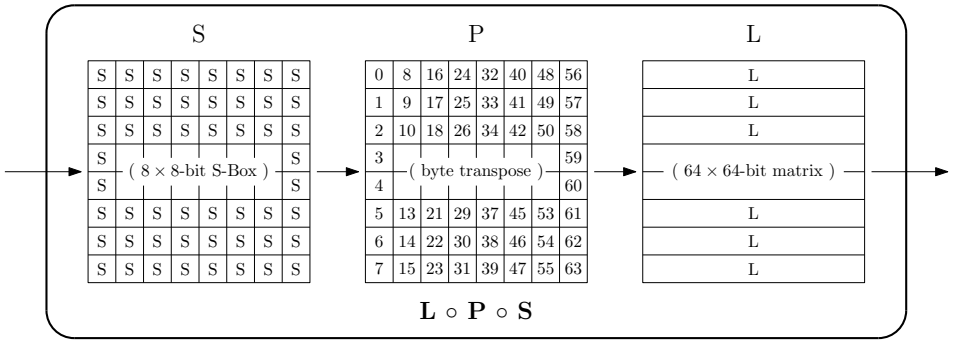

Figure 4: LPS consists of a byte substitution layer S, byte transpose P, and linear layer L.

STREEBOG gets all of its nonlinearity from 8-bit S-box S, which seems to have been designed to offer resistance against classical methods of cryptanalysis. Its differential bound [6] is $P = \frac{8}{256}$ and best linear approximation [16] holds with $P = \frac{28}{128}$.

The linear transform L is not randomly constructed even though it is expressed without explanation as a 64×64 binary matrix in [13]. L in fact has a byte-oriented MDS matrix structure with $\mathbb{F}_{2^8}$ arithmetic in a similar fashion as AES. This is not mentioned in the standard specification [14, 20].

S and L are effective in mixing the bits of the eight 64-bit rows. P swaps rows and columns and after two rounds each input bit affects each output bit of the 512-bit state. LPS has similar per-round avalanche to AES and similar resistance to Square attacks[1] [15].

3 Authenticated Encryption Algorithm STRIBOB

In a sponge function only a single keyless permutation π is required. We utilize the LPS transform and twelve round constants C_i of GOST R 34.11-2012 in our new design. For some vector of twelve 512-bit subkeys K_i we define a 512-bit permutation $\pi_K(X_1) = X_{13}$ with iteration

$$x_{i+1} = \mathsf{LPS}(X_i \oplus K_i) \text{ for } 1 \leq i \leq 12.$$

Theorem 1. *If $\pi_K(x)$ can be effectively distinguished from a random permutation for any C, so can $g_N(h, x)$ for any h and N.*

Proof. If h is known, so are all of the subkeys K_i as those are a function of h alone. We have the equivalence

$$g_N(h, x) \oplus x \oplus h = \pi_K(x \oplus N).$$

Assuming that the round constants C_i offer no advantage over known round keys K_i, π_C is as secure as π_K and any distinguisher should have the same complexity. □

[1] STRIBOBhas 6-round "Squarepants", as this is the best theoretical Square attack we know of.

Theorem 1 indicates that a generic powerful attack against π is also an attack on g. A distinguishing attack against g of course does not imply a collision attack against STREEBOG as a whole. However as the security level expected of STRIBOBis lower than that of STREEBOG, Theorem 1 a significant level of confidence for our construction.

Structure of π is shown in Figure 5. One may find it helpful to compare it with Figure 3 while considering the first input block which always has $N = h = 0$; the subkey values K_i are always the same for the first imput block, regardless of the input message block m. The chaining value h after processing the first block (but before final XORs) is $h = \pi_K(m)$, which is equivalent to the STRIBOBπ_C function, with different random round constants.

The output truncation after the last invocation of g of STREEBOG-256 indicates that collision resistance is expected of half of the output as well, which is exactly what we need in a $r = 256$ Sponge mode (Section 3.1).

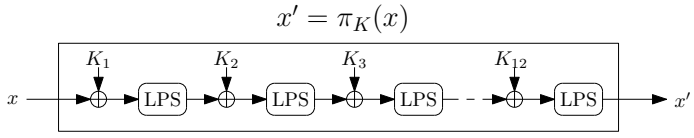

Figure 5: The 512-bit permutation π used by STRIBOB. For the CAESAR version we use round constants $K_i = C_i$.

3.1 Sponge Mode and Security Parameters

The sponge function π is operated in a MULTIPLEX - like mode in order to achieve maximum flexibility [22]. This way STRIBOBcan be used for plain hashing, PRNG generation and in two-party protocols in addition to Authenticated Encryption with Associated Data (AEAD).

Theorem 2 (Theorem 4 from [4])**.** *The* DUPLEXWRAP *and BLNK authenticated encryption modes satisfy the following privacy and authentication security bounds:*

$$\mathrm{Adv}^{\mathsf{priv}}_{\mathsf{sbob}}(\mathcal{A}) < (M+N)2^{-k} + \frac{M^2+4MN}{2^{c+1}}$$
$$\mathrm{Adv}^{\mathsf{auth}}_{\mathsf{sbob}}(\mathcal{A}) < (M+N)2^{-k} + \frac{M^2+4MN}{2^{c+1}}$$

against any single adversary $\mathcal{A}$ if $K \xleftarrow{\$} \{0,1\}^k$, tags of $l \geq t$ bits are used, π is a randomly chosen permutation, M is the data complexity (i.e., the total number of blocks queried to the keyed sponge function or duplex object) and N is the time complexity (total number of times π or its inverse is called).

Proof. Can be derived from Theorem 4 from [4] and Theorem 1 of [1]. □

Since $b = 512$, we choose a Sponge rate of $r = 256$ bits, which leaves capacity $c = b - r = 256$. We choose key size $k = 192$ and limit $M < 2^{56}$ (2^{64} bits) and $N < 2^{k-1}$. As our actual effective capacity is $c \approx 254$ ($\delta \leq 2$ effective capacity bits are lost due to domain separation bits [22]), a 192-bit security level is comfortably reached.

3.2 Padding Example

STRIBOBuses BLNK padding, which is a variant of [22]. The "payload rate bytes" are the first 32 bytes of the state and byte 32 is used as padding and domain indicator. Each element is padded with a `0x01` byte and zeros to full r-bit block length so that π is called between different domains. If the domain data length is an exact multiple of r, `BLNK_END` is set at the domain indicator byte. The domain flags used in the CAESAR implementation are:

Flag name	Value	Padding bit or Domain identifier
BLNK_END	0x01	Padding marker bit
BLNK_FIN	0x02	Data element final block marker bit
BLNK_KEY	0x10	Secret key (in)
BLNK_NPUB	0x20	Public sequence number (in)
BLNK_NSEC	0x30	Secret sequence number (in / out)
BLNK_AAD	0x40	Authenticated Associated Data (in)
BLNK_MSG	0x50	Confidential Message Payload (in/out)
BLNK_MAC	0x60	Message Authentication Code (out)

Example. To illustrate the operation with CAESAR parameters, we use the 192-bit secret key "`192-bit Secret Key value`" and public nonce "`Nonces Used Once`" (16 bytes) to authenticate Associated Data "`AAD Test Vector Exact Block 32 B`" (32 bytes) and to encrypt plaintext "`This is a Test Vector for stribob192r1`" (38 bytes).

S1 STRIBOBuses an all-zero initial state. The first input to π is the padded secret key value:
3139322D626974205365637265742042B65792076616C7565**01**0000000000000000**12**00···00

S2 Nonce is XORed into the state before second π:
4E6F6E6365732055736564204F6E6365**01**00000000000000000000000000000000**22**00···00

S3 Associated data length equals rate (32 bytes) so padding is in domain separation byte:
414144205465737420566563746F7220457861637420426C6F636B203322042**43**00···00

Info. The state before encryption is:
39E876FD1FA6DB05FC681ECAC803A2A48B6CB30E6B47D9FEC94FE1E8CB3E02D4
734803FB16F36A5653DEFEBC7012C28C949172CAEC1274E19A7C5132AFE58EAC

S4 Padded plaintext blocks for π invocations 4 and 5:
5468697320697320612054657374205665637 46F7220666F722073747269626F**50**00···00
623139327231**01**00000000000000000000000000000000000000000000000000000000**52**00···00

Corresponding ciphertext bytes:
6D801F8E3FCFA8259D484AAFBB7782F2EE0FC7611967BF91BB6F929CB95760BB
A808DE292F8B

S5 Authentication tag extraction, 128 bits. No need for state after this with CAESAR.
165BD9D62B3C7B7D6DC423446BE76082

4 Implementation and Performance Notes

Since the new construct requires 12 invocations of LPS per 256 bits processed in comparison to 25 invocations per 512 bits with STREEBOG, we see that the new construct is faster. Furthermore, the operational memory requirement is shrunk to approximately 25 % of the original.

Low-resource software platforms. For a software implementation on a low-resource 8 or 16-bit CPUs and SoCs (e.g. RFID, Smart Card, Sensor, Ubiquitous / IoT category systems) it is advantageous to realize the linear layer L as a matrix multiplication in $\mathbb{F}_{2^8}$. Multiplication in a small finite field can be implemented via discrete logarithm and exponentiation tables: $AB = \exp(\log A + \log B)$. Note that STREEBOG and therefore STRIBOBuses a special bit-inverted representation for field elements [14].

One can combine the S-Box lookup and discrete logarithm table into a single 8×8 - bit lookup table $\log(S(x))$. The 8×8 matrix over $\mathbb{F}_{2^8}$ M (representing L) can be stored in $\log$ form. Required addition $x + y \pmod{2^8 - 1}$ can be implemented by adding carry bit $\lfloor \frac{x+y}{2^8} \rfloor$ of addition $x + y \pmod{2^8}$ to the 8-bit sum itself – set $\exp(255) = \exp(0)$ in this case.

As the transpose P can be coded into the loops (switching the column and row indexes), the implementation of LPS requires a total of $256 + 256 + 8 \times 8 = 576$ bytes for storage. Unfortunately C_i round constants still require $12 \times 64 = 768$ bytes. One may consider a variant that uses a fast pseudorandom generator such as some Fibonacci-based sequence or linear congruential generator instead of a truly random C to further compress the implementation.

Medium- to high-resource software platforms. A software implementation on system with a medium- or high-performance CPUs (e.g. server, desktop, laptop, or tablet category systems) can utilize $8 \times 8 \times 64$ - bit lookup tables that combine S and L, requiring a total of 16 kB and 768 B for round constants. The compression function code itself is very compact.

Results of wall-clock throughput measurements on a typical desktop system: [2]

Algorithm	**Throughput**
AES - 128 / 192 / 256	109.2 / 90.9 / 77.9 MB/s
SHA - 256 / 512	212.7 / 328.3 MB/s
GOST 28147-89	53.3 MB/s
GOST R 34.11-1994	20.8 MB/s
GOST R 34.11-2012	109.4 MB/s
STRIBOB	115.7 MB/s

Hardware. Use of AES instruction set significantly boosts AES performance, but so would similar hardware optimizations for STREEBOG and STRIBOB. Since the rate of STRIBOBis twice that of AES and there are 12 rounds (indicating roughly equivalent critical path), we can expect STRIBOBhardware implementations to be significantly faster than AES.

[2]Measurements were made on a single core of an Intel Core i7 860 @ 2.80 GHz system running Ubuntu Linux 13.10 (amd64) with gcc 4.8.1. The AES, SHA, GOST 28147-89 and R 34.11-1994 timings with were measured with Ubuntu default OpenSSL (1.0.1e). A. Degtyarev's implementation (0.11) was used for the GOST 34.11-2012 benchmark. The STRIBOBreference implementation is by author.

5 Conclusions

We propose STRIBOB, an Authenticated Encryption with Associated Data (AEAD) algorithm based on the GOST R 34.11-2012 hash standard. The new algorithm is faster than the hash standard alone, twice as fast as the GOST 27147-89 encryption algorithm, and is competitive against AES. STRIBOBis a first round candidate in the CAESAR competition of the U.S. National Institute of Standardization and Technology [21, 23].

A strong security relation exists between STRIBOB's π function to the compression function g of GOST R 34.11-2012, giving us a significant level of confidence in its security. Furthermore the underlying Sponge mode of operation is provably secure. We feel that our proposal offers a viable alternative to present GOST standards.

References

[1] BERTONI, G., DAEMEN, J., PEETERS, M., AND ASSCHE, G. V. Duplexing the sponge: Single-pass authenticated encryption and other applications. In *SAC 2011* (2011), A. Miri and S. Vaudenay, Eds., vol. 7118 of *LNCS*, Springer, pp. 320–337.

[2] BERTONI, G., DAEMEN, J., PEETERS, M., AND ASSCHE, G. V. The Keccak reference, version 3.0. NIST SHA3 Submission Document, January 2011.

[3] BERTONI, G., DAEMEN, J., PEETERS, M., AND ASSCHE, G. V. Permutation-based encryption, authentication and authenticated encryption. In *DIAC 2012* (2012). `http://keccak.noekeon.org/KeccakDIAC2012.pdf`.

[4] BERTONI, G., DAEMEN, J., PEETERS, M., ASSCHE, G. V., AND KEER, R. V. CAESAR submission: Keyak v1. CAESAR `http://competitions.cr.yp.to/round1/keyakv1.pdf`, March 2014.

[5] BIHAM, E., AND DUNKELMAN, O. A framework for iterative hash functions - HAIFA. IACR ePrint 2007/278, `http://eprint.iacr.org/2007/278`, July 2007.

[6] BIHAM, E., AND SHAMIR, A. *Differential Cryptanalysis of the Data Encryption Standard*. Springer, 1993.

[7] CHANG, S., R.PERLNER, BURR, W. E., TURAN, M. S., KELSEY, J. M., PAUL, S., AND BASSHAM, L. E. Third-round report of the SHA-3 cryptographic hash algorithm competition. Tech. Rep. NISTIR 7896, National Institute of Standards and Technology, November 2012.

[8] DAEMEN, J., AND RIJMEN, V. *The Design of Rijndael: AES - the Advanced Encryption Standard*. Springer, 2002.

[9] DAMGÅRD, I. A design principle for hash functions. In *CRYPTO '89* (1989), G. Brassard, Ed., vol. 435 of *LNCS*, Springer, pp. 416–427.

[10] DOLMATOV, V., AND DEGTYAREV, A. *GOST R 34.11-2012: Hash Function*. No. RFC 6986. Internet Engineering Task Force, August 2013.

[11] GOST. *Cryptographic Protection for Data Processing System*. No. GOST 28147-89. Gosudarstvennyi Standard of USSR, 1989. (In Russian).

[12] GOST. *Cryptographic Protection of Information, Hash Function*. No. GOST R 34.11-94. Gosudarstvennyi Standard of Russian Federation, 1994. (In Russian).

[13] GOST. *Cryptographic Protection of Information, Hash Function*. No. GOST R 34.11-2012. Gosudarstvennyi Standard of Russian Federation, 2012. (In Russian).

[14] KAZYMYROV, O., AND KAZYMYROVA, V. Algebraic aspects of the russian hash standard GOST R 34.11-2012. In *CTCrypt '13, June 23-24, 2013, Ekaterinburg, Russia* (2013). IACR ePrint 2013/589 `http://eprint.iacr.org/2013/589`.

[15] KNUDSEN, L., AND WAGNER, D. Integral cryptanalysis (extended abstract). In *FSE 2002* (2002), J. Daemen and V. Rijmen, Eds., vol. 2365 of *LNCS*, Springer, pp. 112–127.

[16] MATSUI, M. Linear cryptoanalysis method for DES cipher. In *EUROCRYPT '93* (1994), T. Helleseth, Ed., vol. 765 of *LNCS*, Springer, pp. 386–397.

[17] MATYAS, S., MEYER, C., AND OSSAS, J. Generating strong one-way functions with cryptographic algorithm. *IBM Technical Disclosure Bulletin*, 27 (1985), 5658–5659.

[18] MENDEL, F., PRAMSTALLER, N., RECHBERGER, C., KONTAK, M., AND SZMIDT, J. Cryptanalysis of the GOST hash function. In *CRYPTO 2008* (2008), D. Wagner, Ed., vol. 5157 of *LNCS*, Springer, pp. 162–128.

[19] MERKLE, R. *Secrecy, Authenticatication, and public key systems*. PhD thesis, Stanford University, 1979.

[20] NIST. *Advanced Encryption Standard (AES)*. No. FIPS-197. National Institute of Standards and Technology, 2001.

[21] NIST, AND BERNSTEIN, D. CAESAR call for submissions, final. `http://competitions.cr.yp.to/caesar-call.html`, January 2014.

[22] SAARINEN, M.-J. O. Beyond modes: Building a secure record protocol from a cryptographic sponge permutation. In *CT-RSA 2014* (2014), J. Benaloh, Ed., vol. 8366 of *LNCS*, Springer, pp. 270–285.

[23] SAARINEN, M.-J. O. The STRIBOBr1 authenticated encryption algorithm. CAESAR First Round Candidate, `http://www.stribob.com`, March 2014.

Document version 20140512224100.