Lighter, Faster, and Constant-Time: WhirlBob, the Whirlpool variant of StriBob

Markku-Juhani O. Saarinen[1] and Billy Bob Brumley[2]

[1] Norwegian University of Science and Technology
mjos@item.ntnu.no
[2] Tampere University of Technology, Finland
billy.brumley@tut.fi

Abstract. WhirlBob is an Authenticated Encryption with Associated Data (AEAD) algorithm derived from the first round CAESAR candidate StriBob and the Whirlpool hash algorithm. As with StriBob, the reduced-size Sponge design has a strong provable security link with a standardized hash algorithm. The new design utilizes only the LPS or ρ keying half of Whirlpool in a flexible domain-separated BLNK Sponge mode and increases the number of rounds from 10 to 12 as a countermeasure against Rebound Distinguishing attacks. The Whirlpool and WhirlBob 8×8 - bit S-Box is constructed from 4×4 - bit "MiniBoxes". We report on a fast constant-time SIMD implementation technique that keeps full miniboxes in registers and accesses them via SIMD shuffles. This is an efficient countermeasure against AES-style cache timing side-channel attacks and we have implemented it on Intel SSSE3 and ARM NEON targets. Another main advantage of WhirlBob over StriBob (and most other AEADs) is its greatly reduced implementation footprint on resource-constrained platforms. On many low-end microcontrollers the total software footprint of π+BLNK = WhirlBob AEAD is less than half a kilobyte. We also report an FPGA implementation of WhirlBob. The implementation requires 4,946 logic units for a single round of WhirlBob, which compares favorably to 7,972 required for Keccak/Keyak on the same platform.The relatively small hardware gate count is also reflected as efficient bitsliced straight-line implementations, especially on pure 64-bit platforms. We finally present some discussion and analysis on the relationships between WhirlBob, Whirlpool, the Russian GOST Streebog hash, and the recent draft Russian Encryption Standard Kuznyechik.

Keywords: Authenticated Encryption, Sponge designs, Timing Attacks, Whirlpool, Streebog, StriBob, CAESAR Project.

1 Introduction

WhirlBob 1.0 is an Authenticated Encryption with Associated Data (AEAD) algorithm based on the CAESAR candidate StriBob [52, 53] and NESSIE Final Portfolio [39] hash function Whirlpool 3.0 [4].

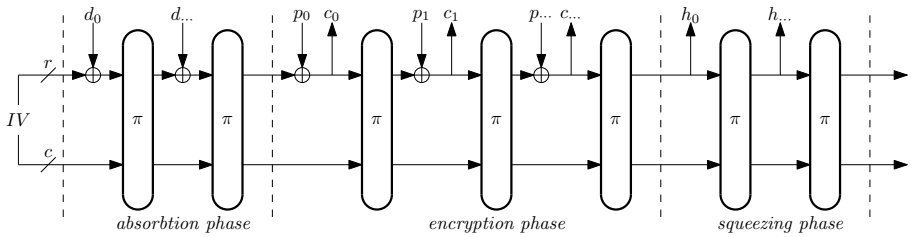

Fig. 1. A simplified view of a Sponge-based AEAD. Padded Secret Key, Nonce, and Associated Authenticated Data - all represented by d_u words - are first "absorbed" into the state. The π permutation is then also used to encrypt data p_i into ciphertext c_i (or vice versa) and finally to "squeeze" out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

AEAD algorithms and modes such as GCM [42] provide both confidentiality and integrity protection, typically in a single pass, thus eliminating the requirement for a MAC algorithm such as HMAC [43]. This has clear advantages for performance and implementation footprint.

2 Motivation: Security Goals and Parameters

WhirlBob uses StriBob's BLNK Sponge AEAD mode and parameters without modification. Outside the CAESAR context, BLNK can be also used in a wider set of applications, even to build entire secure lightweight protocol suites [50].

A sponge mode requires only a single cryptographic component; an unkeyed cryptographic permutation π (See Figure 1). As with other provable Sponge modes, we assume that π is indistinguishable from a random permutation. This work focuses on π permutation design and implementation – for BLNK padding details and analysis we refer to [29, 50, 53].

The StriBob CAESAR [53] candidate is derived from the Russian GOST hash standard Streebog [23]. In close examination Streebog appears to be modeled after the Whirlpool hash [4], with substantial modifications. StriBob and WhirlBob only differ in the particular numerical selections for tables C, S, and L. The code of 64-bit reference implementation is essentially unchanged. These components, $L \circ P \circ S$ or the "LPS permutation" is derived directly from that of Whirlpool for WhirlBob. Both StriBob and WhirlBob have 12 rounds.

One of our aims is to allow the same secure LPS implementation core (such as a special instruction of an SoC CPU in a mobile or IoT device) to be used for unkeyed hashing according to the Whirlpool standard. This is useful in applications that require certificate signature processing. The corresponding standardized, Miyagushi-Preneel hash functions Streebog and Whirlpool require two (or more) times as much as state and processes data in bigger chunks when compared to StriBob and WhirlBob. Our BLNK Sponge mode also supports randomized hashing and MACing without encryption. Our Sponge variants are slightly faster than the original hashes, yet have a provable security relation.

All security parameters remain unmodified from StriBob. As with StriBob, we have an $b = 512$ bit state, which is split to $r = 256$ - bit *rate* "block size" and $c \approx 254$ - bit *capacity*, which is the secret state. According to theorems such as those given in [29, 53] this is sufficient for $k = 192$ - bit secret key security level when less than 2^{64} bits are processed under same key and nonce pair. For the standard variant nonce length is fixed at $n = 128$ bits.

3 WhirlBob

Despite having almost equivalent speed and size on generic 64-bit platforms, the size and performance characteristics of StriBob and WhirlBob differ significantly on various implementation platforms such as FPGA, low-end microcontrollers, SIMD systems, and in bitslicing implementations.

We only give an abbreviated description of WhirlBob's 512×512 - bit keyless π permutation as the computation follows exactly the operation of the internal key schedule of Whirlpool 3.0 [4]. The only modification is that the number of rounds is increased from $R = 10$ to $R = 12$. The key schedule operation is effectively equivalent to the "internal block cipher" W. Blocks of eight bytes from the S-Box are used as partial round keys C_i, as in Whirlpool.

WhirlBob's permutation π is indeed highly similar to AES. In case of StriBob, the "Russian 512-bit block AES" permutation had to be somewhat laboriously uncovered from the structure (see Section 4.3), but the particularities and history of Whirlpool make the connection immediately obvious.

The 512-bit state is typically written as a matrix of 8×8 bytes. To compute $\pi(x_0) = x_{12}$ we iterate

$$x_{i+1} = L(P(S(x_i))) \oplus C_i$$

where, if we use the original AES-style notation, S is equivalent to `SubBytes`, P corresponds to `ShiftColumns`, L to `MixRows`, followed by `AddRoundKey`.

3.1 Lightweight Reference Implementation

The entire byte-oriented implementation of π fits onto a single page; See Appendix A. Remarkably, in addition to π, only the S-Box `wbob_sbox[256]` (See Section 3.2) together with minimal BLNK logic are required for full AEAD implementation. On most microcontrollers WhirlBob's entire software footprint is in the 500B range. Slightly more is required for a shared secret handshake protocol and two-way secure BLINKER protocol [50].

This is a significant improvement over StriBob, which typically needs almost 2kB. StriBob is also much slower and larger on low-end microcontrollers due to the "heavy" MDS matrix. The reference implementation is written for compactness and clarity; it is not optimal when it comes to speed or size. We refer to section 7.3 of [4] for techniques that greatly reduce the number of XORs required, resulting in increased processing speed. Additional tables will be required and this will increase the overall implementation size.

Table 1. Three 4×4 miniboxes that are used to build the 8×8 S-Box in Whirlpool and WhirlBob 1.0. We may revise these for CAESAR Round 2.

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$E(x)$	1	B	9	C	D	6	F	3	E	8	7	4	A	2	5	0
$E^{-1}(x)$	F	0	D	7	B	E	5	A	9	2	C	1	3	4	8	6
$R(x)$	7	C	B	D	E	4	9	F	6	3	8	A	2	5	1	0

Whirlpool ISO Standard trace test vectors have been used to verify the correctness of this π implementation, up to R=10 (Appendix B). One simply observes the keying "line" of these traces and ignores the encryption "line". We offer the listing of Appendix A as WhirlBob v1 π Reference implementation. (Please note the system of algorithm designations at the end of Section 5.)

3.2 Impact of New S-Box Structure on Implementations

Whirlpool's S-Box design utilizes three 4×4 - bit "miniboxes" given in Table 1: E, E^{-1}, and R. Figure 2 shows how these are used to construct the 8×8 - bit S-Box. This computation can even be performed on the fly on 4-bit microcontrollers. FPGA implementations save a significant number of LUTs by explicitly utilizing the 4-bit structure rather than implementing a general 8×8 lookup table.

Note that these small S-Boxes can often fit into a single register and accessed via constant-time shifts, thus enabling constant-time implementation.

The byte-oriented $8 \times 64 = 512$ - bit state can be rapidly split into eight 64-bit registers. The parallelism evident in Figure 2 helps to speed up bitsliced

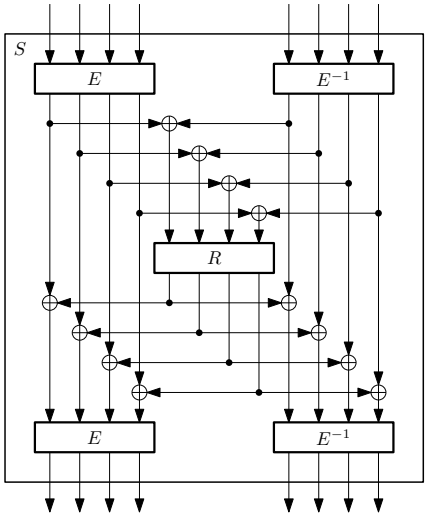

Fig. 2. The 8×8 - bit S-Box is constructed from 4×4 - bit "miniboxes".

implementation. We see that for 2/3 of the time, the S-Box has effectively two independent 4-bit execution paths. Interleaving these may greatly reduce wait states due to the superscalar architecture employed by most modern CPUs.

Appendix B of the current 2003 Whirlpool specification [4] gives listings with 14-16 instructions/gates for each of the miniboxes (if ANDN instruction is allowed).

3.3 Constant-Time Implementation

Due largely in part to Whirlpool's S-Box structure and generous parallelism, it is well suited for high speed, constant-time implementation on Single Instruction Multiple Data (SIMD) architectures. Here we focus on ARM's NEON as the reference architecture since the state layout fits the registers nicely, but also consider Intel's SSSE3 as another explicit example. The goal is to improve performance, while at the same time avoiding memory-resident table lookups that cause execution time to depend on the data cache state and thus algorithm state (the crux of cache timing attacks).

Related work in this area includes simulated ISA extensions to a RISC architecture for parallel table lookups to speed up Whirlpool [25]. These extensions are then used to build essentially a hardware-assisted analogue of the traditional T tables software implementation – storing the state in rows and issuing a single instruction to perform 8 parallel lookups from the 8-bit S-Box input to the 64-bit linear layer output and XOR-summing the results, repeated for each row. AES [24] and Anubis [17] can also take advantage of SSSE3's variable byte shuffle instruction for fast and secure implementations.

NEON has 32×64 - bit SIMD registers and SSSE3 16×128 - bit. We store the state column-wise (one column per NEON register, two columns per SSSE3 register), i.e. byte position j of register i contains the state byte in column i and row j. The `SubBytes` step is not sensitive to this ordering, but both `ShiftColumns` and `MixRows` are. Since both of these architectures feature variable byte shuffle instructions (`vtbl.u8` for NEON and `pshufb` for SSSE3), implementing `SubBytes` is a direct translation of Figure 2 to these instructions. This amounts to 40 NEON shuffles and half as many SSSE3 shuffles. For `ShiftColumns`, NEON uses `vext` for byte-wise register rotation and SSSE3 `pshufb` with constant rotation distances since each register holds two columns. For `MixRows` we use the row formula from the Whirlpool specification [4, Sec. 7.3] where the multiplications by x are a simple left shift (native on NEON, integer addition on SSSE3) and conditional XOR (operand masked by signed right shift on NEON, comparison on SSSE3). The formula is fairly symmetric around even and odd byte positions – while NEON implements it as written with 24 multiplications, SSSE3 slightly rearranges a few registers to parallelize across the full 128-bit register width and use half as many multiplications.

3.4 Implementation Summary

We currently have six implementations of the cryptographic π permutation.

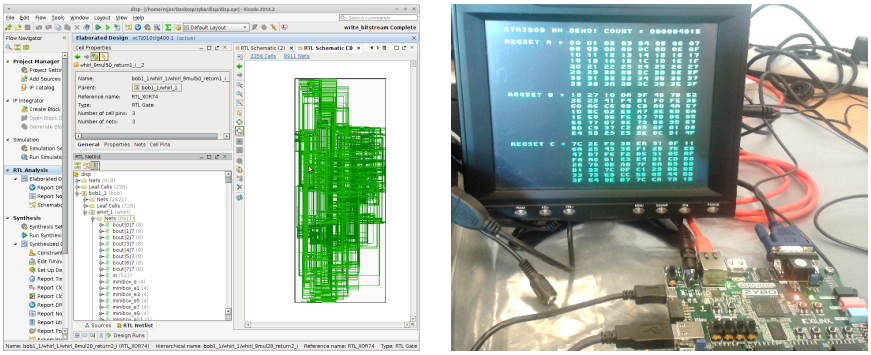

Fig. 3. WhirlBob was implemented on the FPGA logic fabric of Xilix Zynq 7010. The implementation integrates with the AXI bus of ARM Cortex A9 on the SoC chip.

- **C 8-bit**: This is the minimal reference implementation which is optimized for clarity and low-resource platforms, corresponding to Appendix A.
- **C 64-bit**: Standard speed-optimized implementation for most platforms, utilizing large lookup tables. Apart from Whirlpool-derived tables, equivalent to the implementation of STRIBOBr1.
- **C Bitsliced**: Straight-line, fully bitsliced implementation without data-dependent branches or lookups. Resistant to timing attacks.
- **NEON Intrinsics**: Fast constant-time version that avoids table lookups by storing 4×4 - bit miniboxes in SIMD registers.
- **SSSE3 Intrinsics**: Similar but for 128-bit SIMD registers.
- **Verilog 12-cycle**: This is the hardware reference implementation. Source code is about 350 lines. Additional logic is required for AXI Bus integration.

Software Implementations. The first three implementations use only C99, and are hence easily portable. The 64-bit reference implementation is almost exactly as fast as OpenSSL's Whirlpool on the same platform. See Table 2 for implementation metrics. We also have various embedded implementations.

Hardware Implementation. The hardware implementation has been proven on FPGA (Figure 3). The SÆHI proposal reports total post place-and-route utilization on Artix-7 of 4,946 logic units for a single round of WhirlBob, which compares favorably to 7,972 required for Keccak/Keyak [51]. Throughput is roughly 2 MB/s for each MHz.

3.5 Comparison with Other AEAD Schemes.

At the time of writing (Q3/2014) the dominant AEAD scheme is the Galois / Counter Mode (GCM) for the AES block cipher [40, 42], which is recommended for use with TLS, SSH and IPSec protocols by NSA as part of "Suite B" [18,

Table 2. Comparing software implementations of WhirlBob's π.

Metric and Target	**Speed**	**Footprint**		**Source**
	MB/sec	Code	Data	C lines
Single Core of 2.8GHz Core i7 860				
8-bit C99 Reference	4.6311	326	256	97
64-bit C99 Reference	95.368	1942	16512	128
Bitsliced C99 Reference	30.856	4592	768	345
SSSE3 (Constant-Time)	123.15	1290	1152	256
BeagleBone Black 1.0GHz Cortex-A8				
8-bit C99 Reference	0.8288	352	256	97
64-bit C99 Reference	3.3435	6524	16512	128
Bitsliced C99 Reference	1.4353	15704	768	345
NEON (Constant-Time)	9.2084	1528	1072	320

26, 45, 54]. GCM message authentication is based on polynomial evaluation in the finite field $\mathsf{GF}(2^{128})$. The required multiplication can be exceedingly slow on lightweight platforms. An LFSR-style implementation of a 128×128 - bit multiplication will require thousands of cycles on 8-bit targets.

It is often more efficient to use the CCM [41, 58] double-mode of operation on lightweight platforms, since implementing a full extra AES operation can be faster than the finite field multiplication operation. CCM and GCM are currently the only two FIPS - standardized authenticated modes. The performance characteristics of AES-CCM AEAD can be expected to be very similar to WhirlBob due to their structural similarities and relative data bandwidth:

- WhirlBob: 12 rounds with 64 S-Boxes for 256 bits of data.
- AES-192-CCM: 2×12 rounds with 16 S-Boxes for 128 bits of data.

There are additional (patented) AES modes which will be faster on 8-bit platforms – such as AES-OTR [37] and AES-OCB [31], and dozens of others. Virtually all block cipher modes offer lower levels of integrity protection (2^{64} level even for 128-bit tags) and are not directly usable in wider Sponge applications such as non-randomized hashing.

At the time of writing (Q3/2014) only unoptimized reference implementations are available for most CAESAR candidates [19], making fair performance comparisons difficult. Furthermore, no other CAESAR candidate is targeted at 192-bit security level apart from AES modes. Little attention has been paid to 8-bit or hardware implementations.

We note that leading full-featured Sponge candidates, directly SHA3 / Keccak - based Ketje [11] and Keyak [12] have significantly slower reference implementation than StriBob and Whirlbob (Table 3). WhirlBob falls very significantly from candidates such as NORX [3] and MORUS [59], which have been designed specifically with 64-bit targets in mind. Our proposal can claim a more conservative security margin when compared to these candidates, however.

Table 3. Relative performance of some CAESAR candidates on the AMD64 reference system in SUPERCOP testing (smaller number indicates faster speed).

MORUS 1280 - 128 [59]	0.09
NORX 64-4-1 [3]	0.19
ASCON-128n [22]	0.89
WhirlBob Intel SSSE3 Constant-Time	**1.00**
WhirlBob and StriBob 64-bit Reference	1.26
Lake Keyak [12]	2.23
Ketje Sr. [11]	4.25
PRIMATES (HANUMAN, GIBBON, APE) [2]	50+

Source: `http://bench.cr.yp.to/web-impl/amd64-titan0-crypto_aead.html`

4 Security Analysis and Design Notes

Most of the security arguments and proofs offered for StriBob in [53] also apply unmodified to the new proposal, as those proofs are based on an indistinguishably arguments of the π permutation and a simple theorem (Thm. 1, Sec. 3.3. in [53]) that loosely ties the Miyagushi-Preneel mode [38, 48] with the indistinguishably of π. A random-indistinguishable π and appropriate padding rules are sufficient to construct Sponge-based hashes [6], Tree Hashes [10], MACs [9], Authenticated Encryption (AE) algorithms [8, 12], and pseudorandom extractors (SHAKEs, PRFs, and PRNGs) [7, 44].

4.1 Side-Channel and Implementation Attacks

Due to the minibox structure, we may load the 4×4 - bit tables in registers and access them via constant-time shuffles on Intel SSSE3 and ARM NEON SIMD targets (as noted in Section 3.3). Whirlpool is also relatively well suited for bitsliced implementation due to its particular S-Box and MDS design (as noted in Section 3.2).

Being unconditional straight-line code without data-dependent table lookups, bitsliced and byte shuffling implementations are effective countermeasures against cache timing attacks, which are effective against cryptographic primitives with large tables such as AES [1, 5, 47, 57].

A non-bitsliced implementation of the S-Box on Whirlpool, Streebog, or StriBob on 64-bit platforms typically requires lookup tables of up to $8 \times 256 \times 8 =$ 16384B. Even though this size easily fits into the Level 2 cache of any 64-bit system, one may see that timing attacks are possible as L2 caches are not always shared even between different execution cores within a single CPU unit. This is due to the process switching operation of most 64-bit operating systems.

4.2 Historical Modifications to Whirlpool

Whirlpool has received a significant amount of analysis in the almost 15 years since its original publication. Whirlpool was the only hash function in the final

NESSIE portfolio in addition to SHA-2 hashes [39]. Whirlpool has also been standardized by ISO as part of ISO/IEC 10118-3:2004 [27].

Our design is based on Whirlpool 3.0. The amended MDS matrix used by current ('03) Whirlpool is also used by WhirlBob as a countermeasure to the structural observations given in [55].

Whirlpool was found to be vulnerable to a Rebound Distinguisher [32, 33, 36]. That 2^{188} attack applies to the 10-round variant; our 12-round version should offer a comfortable security margin, especially as our security target is 2^{192}. The way the round constants are derived from the S-Box allows this change to be made in a straightforward manner.

4.3 Notes on the origins of Streebog, Kuznyechik, and StriBob

The GOST R 34.11-2012 "Streebog" standard text [23] does not describe the linear step as a 8×8 matrix-vector multiplication with $\mathsf{GF}(2^8)$ elements like the StriBob spec [53], but as a 64×64 binary matrix multiplication. One can see that $8 \times 8 \times 8 = 512$ bits are required to describe the former, but $64 \times 64 = 4096$ bits are required for the latter. The more effective description was discovered by Kazymorov and Kazymorova in [30] by exhaustively testing all 30 irreducible polynomial basis, revealing an AES-like MDS structure. The origin of the particular numerical values of that MDS matrix and round constants is still a mystery. They do not appear to offer avenues for size or performance optimization like those in Whirlpool 3.0 and WhirlBob do.

The 8-bit S-Box used by StriBob was directly lifted from Streebog so that hardware and software components developed for Streebog could be shared or recycled when implementing StriBob. The same S-Box is also used by the very recently proposed Russian Encryption Standard "Kuznyechik" [56].

Not much about the particular design criteria of the Streebog S-Box has been published. That S-Box was apparently selected at least 5 years ago as Streebog already appeared in RusCrypto '10 proceedings [35]. We can easily observe that it offers reasonable resistance against basic methods of cryptanalysis. Its differential bound [13] is $P = \frac{8}{256}$ and best linear approximation [34] holds with $P = \frac{28}{128}$. There does not seem to be any exploitable algebraic weaknesses. These are the exact same bounds as can be found for the Whirlpool S-Box, but fall clearly short from the bounds of the AES S-Box.

The Rijndael AES S-Box is constructed from finite field inversion x^{-1} operation in $\mathsf{GF}(2^8)$ (inspired by the Nyberg construction [46]) and an affine bit transform that serves as a countermeasure against, among other things, Interpolation Attacks [28] on the AES predecessor SHARK [49]. We refer to [21] for more information about the AES design process.

We had brief informal discussions with some members of the Streebog and Kuznyechik design team at the CTCrypt '14 workshop (05-06 June 2014, Moscow RU). Their recollection was that the aim was to choose a "randomized" S-Box that meets the basic differential, linear, and algebraic requirements. Randomization was simply iterated until a "good enough" permutation was found. This was seen as an effective countermeasure against yet-unknown attacks. At the

time of Streebog S-Box selection (before 2010's) the emergence of allegedly effective AES Algebraic Attacks such as [20] was a major concern for much of the symmetric cryptographic community. Hence it was felt appropriate to avoid too much algebraic structure in either the S-Box or MDS matrix while also ensuring necessary resistance against known attacks such as DC and LC. Algebraic attack attempts of this type against AES have since largely fizzled out, so we feel confident that the Whirlpool S-Box should be sufficient for our claimed security level, especially as it offers significantly better speeds in bitsliced implementations.

One is left with the impression that Streebog is a "whitened" or randomized copy of the original Whirlpool design. Despite its partially unknown origins and relative shortcomings on some implementation targets, we consider StriBob to be a more secure algorithm than WhirlBob if appropriately implemented. Indeed some of the more successful attacks on AES and Whirlpool have been based on their deep structural self-similarities and simplistic key schedules [14–16].

5 Conclusions

We have introduced the WhirlBob 1.0 authenticated encryption algorithm, a variant of the StriBob first round CAESAR candidate. The new proposal loans its key components from the Whirlpool 3.0 hash function, modifying it into a Sponge AEAD. WhirlBob has extremely small implementation footprint on resource-limited software and hardware platforms – typically under half a kilobyte. The reference implementation fits onto a single page of Appendix A.

The hardware-optimized design of Whirlpool components also gives WhirlBob efficient bitsliced and SIMD byte shuffling implementations. These are effective countermeasure against cache timing attacks, which are a concern against AES. The $b = 8 \times 64$ - bit state size is particularly suitable for bitslicing of a byte-oriented algorithm on 64-bit platforms and byte slicing for SIMD platforms.

We also discussed the design choices for the S-Box and other components used in the Streebog hash and Kuznyechik cipher, which are standards or becoming standards for the Russian security market.

However WhirlBob has superb implementation characteristics on SIMD and lightweight platforms. Furthermore WhirlBob offers provable security assurance through its security reduction to the well-analyzed Whirlpool hash. Furthermore, the RAM requirement of WhirlBob AEAD is only half of that required by Whirlpool.

Note on designations. This document describes WhirlBob 1.0, which corresponds to Whirlpool 3.0's components. Should STRIBOB be selected for the second round of the CAESAR competition, a WhirlBob tweak will be designated STRIBOBr2d2 (Round 2, Design 2) a.k.a. WhirlBob 2.0 and may differ from this description. The original StriBob based on Streebog components will be designated STRIBOBr2d1 (Round 2, Design 1.) The current official first round algorithm designation is STRIBOBr1 [53].

Acknowledgements

We thank Oleksandr Kazymyrov, Vasily Shishkin, Bart Preneel, and Paulo Barreto for their helpful comments. Dr. Saarinen carried out his research during the tenure of an ERCIM "Alain Bensoussan" Fellowship Programme.

References

1. Aciiçmez, O., Schindler, W., and Ç. K. Koç. Cache based remote timing attack on the AES. In *CT-RSA 2007* (2007), M. Abe, Ed., vol. 4377 of *LNCS*, Springer, pp. 271–286.
2. Andreeva, E., Bilgin, B., Bogdanov, A., Luykx, A., Mendel, F., Mennink, B., Mouha, N., Wang, Q., and Yasuda, K. PRIMATEs v1 – Submission to the CAESAR Competition. `competitions.cr.yp.to/round1/primatesv1.pdf`, March 2014.
3. Aumasson, J.-P., P.Jovanovic, and Neves, S. CAESAR submission: NORX v1. `competitions.cr.yp.to/round1/norxv1.pdf`, March 2014.
4. Barreto, P. S. L. M., and Rijmen, V. The Whirlpool hashing function. NESSIE Algorithm Specification `www.larc.usp.br/~pbarreto/WhirlpoolPage.html`, 2000, Revised May 2003.
5. Bernstein, D. J. Cache-timing attacks on AES. Tech. rep., University of Chigaco, 2005.
6. Bertoni, G., Daemen, J., Peeters, M., and Assche, G. V. Sponge functions. In *Ecrypt Hash Workshop 2007* (May 2007).
7. Bertoni, G., Daemen, J., Peeters, M., and Assche, G. V. Sponge-based pseudo-random number generators. In *CHES 2010* (2010), S. Mangard and F.-X. Standaert, Eds., vol. 6225 of *LNCS*, Springer, pp. 33–47.
8. 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.
9. Bertoni, G., Daemen, J., Peeters, M., and Assche, G. V. On the security of the keyed sponge construction. In *SKEW 2011 Symmetric Key Encryption Workshop* (February 2011).
10. Bertoni, G., Daemen, J., Peeters, M., and Assche, G. V. Sakura: a flexible coding for tree hashing. IACR ePrint 2013/231, `eprint.iacr.org/2013/231`, April 2013.
11. Bertoni, G., Daemen, J., Peeters, M., Assche, G. V., and Keer, R. V. CAESAR submission: Ketje v1. `competitions.cr.yp.to/round1/ketjev1.pdf`, March 2014.
12. Bertoni, G., Daemen, J., Peeters, M., Assche, G. V., and Keer, R. V. CAESAR submission: Keyak v1. `competitions.cr.yp.to/round1/keyakv1.pdf`, March 2014.
13. Biham, E., and Shamir, A. *Differential Cryptanalysis of the Data Encryption Standard*. Springer, 1993.
14. Biryukov, A., and Khovratovich, D. Related-key cryptanalysis of the full AES-192 and AES-256. In *ASIACRYPT '09* (2009), M. Matsui, Ed., vol. 5912 of *LNCS*, Springer, pp. 1–18.
15. Biryukov, A., Khovratovich, D., and Nikolić, I. Distinguisher and related-key attack on the full AES-256. In *CRYPTO '09* (2009), S. Halevi, Ed., vol. 5677 of *LNCS*, Springer, pp. 231–249.

16. Bogdanov, A., Khovratovich, D., and Rechberger, C. Biclique cryptanalysis of the full AES. In *ASIACRYPT '11* (2011), D. Lee and X. Wang, Eds., vol. 7073 of *LNCS*, Springer, pp. 344–371.
17. Brumley, B. B. Secure and fast implementations of two involution ciphers. In *NordSec '10* (2012), T. Aura, K. Järvinen, and K. Nyberg, Eds., vol. 7127 of *LNCS*, Springer, pp. 269–282.
18. Burgin, K., and Peck, M. Suite B Profile for Internet Protocol Security (IPsec). IETF RFC 6380, October 2011.
19. CAESAR. CAESAR first round submissions. `competitions.cr.yp.to/caesar-submissions.html`, March 2014.
20. Courtois, N. How fast can be algebraic attacks on block ciphers? IACR ePrint 2006/168, `eprint.iacr.org/2006/168`, May 2006.
21. Daemen, J., and Rijmen, V. *The Design of Rijndael: AES - the Advanced Encryption Standard*. Springer, 2002.
22. Dobraunig, C., Eichlseder, M., Mendel, F., and Schläffer, M. Ascon v1 – Submission to the CAESAR Competition. `competitions.cr.yp.to/round1/asconv1.pdf`, March 2014.
23. GOST. Information technology. cryptographic protection of information, hash function. GOST R 34.11-2012, 2012. (In Russian).
24. Hamburg, M. Accelerating AES with vector permute instructions. In *CHES '09* (2009), C. Clavier and K. Gaj, Eds., vol. 5747 of *LNCS*, Springer, pp. 18–32.
25. Hilewitz, Y., Yin, Y. L., and Lee, R. B. Accelerating the Whirlpool hash function using parallel table lookup and fast cyclical permutation. In *FSE '08* (2008), K. Nyberg, Ed., vol. 5086 of *LNCS*, Springer, pp. 173–188.
26. Igoe, K. Suite B Cryptographic Suites for Secure Shell (SSH). IETF RFC 6239, May 2011.
27. ISO/IEC. Information technology – security techniques – hash-functions – part 3: Dedicated hash-functions. ISO/IEC 10118-3:2004, 2004.
28. Jakobsen, T., and Knudsen, L. The interpolation attack on block cipers. In *FSE '97* (1996), E. Biham, Ed., vol. 1267 of *LNCS*, Springer, pp. 99–112.
29. Jovanovic, P., Luykx, A., and Mennink, B. Beyond $2^{c/2}$ security in sponge-based authenticated encryption modes. IACR ePrint 2014/373, `eprint.iacr.org/2014/373`, May 2014.
30. 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 2013/556 `eprint.iacr.org/2013/556`.
31. Krovetz, T., and Rogaway, P. OCB (v1). `competitions.cr.yp.to/round1/ocbv1.pdf`, March 2014.
32. Lamberger, M., Mendel, F., Rechberger, C., Rijmen, V., and Schläffer, M. Rebound distinguishers: Results on the full whirlpool compression function. In *ASIACRYPT '09* (2009), M. Matsui, Ed., vol. 5912 of *LNCS*, Springer, pp. 126–143.
33. Lamberger, M., Mendel, F., Schläffer, M., Rechberger, C., and Rijmen, V. The rebound attack and subspace distinguishers: Application to Whirlpool. *J. Cryptology* (2013). DOI: 10.1007/s00145-013-9166-5.
34. Matsui, M. Linear cryptoanalysis method for DES cipher. In *EUROCRYPT '93* (1994), T. Helleseth, Ed., vol. 765 of *LNCS*, Springer, pp. 386–397.
35. Matyuhin, D. V., Rudskoy, V. I., and Shishkin, V. A. Promising hashing algorithm. RusCrypto '10 Workshop, 02 April 2010, 2010. (In Russian).

36. Mendel, F., Rechberger, C., Schläffer, M., and Thomsen, S. S. The rebound attack: Cryptanalysis of reduced Whirlpool and Grøstl. In *FSE 2009* (2009), O. Dunkelman, Ed., vol. 5665 of *LNCS*, Springer, pp. 260–276.
37. Minematsu, K. AES-OTR v1. `competitions.cr.yp.to/round1/aesotrv1.pdf`, March 2014.
38. Miyaguchi, S., Ohta, K., and Iwata, M. 128-bit hash function (n-hash). *NTT Review*, 2 (1990), 128–132.
39. NESSIE. *Final report of European project number IST-1999-12324, named New European Schemes for Signatures, Integrity, and Encryption*. April 2004. Draft available at `https://www.cosic.esat.kuleuven.be/nessie/Bookv015.pdf`.
40. NIST. Advanced Encryption Standard (AES). FIPS 197, 2001.
41. NIST. Counter with Cipher Block Chaining - Message Authentication Code (CCM). NIST Special Publication 800-38C, May 2004.
42. NIST. Recommendation for block cipher modes of operation: Galois/counter mode (GCM) and GMAC. NIST Special Publication 800-38D, 2007.
43. NIST. The keyed-hash message authentication code (HMAC). FIPS 198-1, July 2008.
44. NIST. DRAFT SHA-3 standard: Permutation-based hash and extendable-output functions. DRAFT FIPS 202, May 2014.
45. NSA. Suite B Cryptography. `www.nsa.gov/ia/programs/suiteb_cryptography`, June 2014.
46. Nyberg, K. Differentially uniform mappings for cryptography. In *EUROCRYPT '93* (1993), T. Helleseth, Ed., vol. 765 of *LNCS*, Springer, pp. 55–64.
47. Osvik, D. A., Shamir, A., and Tromer, E. Cache attacks and countermeasures: The case of AES. In *CT-RSA 2006* (2006), D. Pointcheval, Ed., vol. 3860 of *LNCS*, Springer, pp. 1–20.
48. Preneel, B. *Analysis and Design of Cryptographic Hash Functions*. PhD thesis, K. U. Leuven (Belgium), 1993.
49. Rijmen, V., Daemen, J., Preneel, B., Bosselaers, A., and Win, E. D. The cipher SHARK. In *FSE '96* (1996), vol. 1008 of *LNCS*, Springer, pp. 99–111.
50. 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.
51. Saarinen, M.-J. O. Simple AEAD Hardware Interface (SÆHI) in a SoC: Implementing an On-Chip Keyak/WhirlBob Coprocessor. In *TrustED '14, 03 November 2014 Scottsdale AZ, USA* (November 2014), ACM (to appear).
52. Saarinen, M.-J. O. StriBob: Authenticated encryption from GOST R 34.11-2012 LPS permutation (extended abstract). In *CTCrypt '14, 05-06 June 2014, Moscow, Russia. Preproceedings* (June 2014), pp. 170–182.
53. Saarinen, M.-J. O. The STRIBOBr1 authenticated encryption algorithm. CAESAR, 1st Round `www.stribob.com`, March 2014.
54. Salter, M., and Housley, R. Suite B Profile for Transport Layer Security (TLS). IETF RFC 6460, January 2012.
55. Shirai, T., and Shibutani, K. On the diffusion matrix employed in the Whirlpool hashing function. NESSIE Public Report, 2003.
56. Shishkin, V., Dygin, D., Lavrikov, I., Marshalko, G., Rudskoy, V., and Trifonov, D. Low-weight and hi-end: Draft Russian Encryption Standard. In *CTCrypt '14, 05-06 June 2014, Moscow, Russia. Preproceedings.* (June 2014), pp. 183–188.

57. Weiss, M., Heinz, B., and Stumpf, F. A cache timing attack on AES in virtualization environments. In *FC 2012* (2013), A. Keromytis, Ed., vol. 7397 of *LNCS*, Springer, pp. 314–328.
58. Whiting, D., Housley, R., and Ferguson, N. Counter with CBC-MAC (CCM). IETF RFC 3610, September 2003.
59. Wu, H., and Huang, T. The Authenticated Cipher MORUS (v1). `competitions.cr.yp.to/round1/morusv1.pdf`, March 2014.

A WhirlBob 1.0 π "8-bit" Reference Implementation

This ANSI C function implements the WhirlBob 512×512-bit π permutation.

```
void wbob_pi(uint8_t st[64])            // WhirlBob Pi
{
    int r, i, j;
    uint8_t t[64], x, *pt;

    for (r = 0; r < 12; r++) {          // 12 rounds
        for (i = 0; i < 64; i++) {
            t[(i & 7) + ((i + (i << 3)) & 070)] = // P
                wbob_sbox[st[i]];                 // S
        }
        // The round constants C comes from the S-box
        pt = (uint8_t *) &wbob_sbox[8 * r];
        for (i = 0; i < 8; i++)
            st[i] = pt[i];              // C in first 8
        for (i = 8; i < 64; i++)
            st[i] = 0;                  // zero the rest

        // Apply the circular, low weight MDS matrix
        for (i = 0; i < 64; i += 8) {
            pt = &st[i];                // start of row
            for (j = 0; j < 8; j++) {
                x = t[i + j];           // Circular MDS
                pt[j & 7] ^= x;         // 01
                pt[(j + 1) & 7] ^= x;   // 01
                pt[(j + 3) & 7] ^= x;   // 01
                pt[(j + 5) & 7] ^= x;
                pt[(j + 7) & 7] ^= x;
                // x <- 02
                x = (x << 1) ^ (x & 0x80 ? 0x1D : 0x00);
                pt[(j + 6) & 7] ^= x;   // 02
                // x <- 04
                x = (x << 1) ^ (x & 0x80 ? 0x1D : 0x00);
                pt[(j + 2) & 7] ^= x;   // 04
                pt[(j + 5) & 7] ^= x;   // 01 + 04 = 05
                // x <- 08
                x = (x << 1) ^ (x & 0x80 ? 0x1D : 0x00);
                pt[(j + 4) & 7] ^= x;   // 08
                pt[(j + 7) & 7] ^= x;   // 01 + 08 = 09
            }
        }
    }
}
```

B Test Vectors

B.1 The 12-round π transform

These are derived from ISO Test vectors of Whirlpool 3.0. We give the input x_0 and results after 1, 10 (as in Whirlpool), and full 12 rounds of processing.

$$x_0 = \begin{pmatrix} 77\ 38\ E1\ B5\ 41\ A0\ 36\ EA \\ 45\ 8D\ 50\ F8\ 0F\ A0\ 1C\ 44 \\ 72\ 88\ CE\ 97\ D1\ A0\ DC\ F0 \\ 16\ 95\ FF\ D6\ E7\ 1D\ 09\ 25 \\ 33\ BE\ 30\ 9F\ 01\ 2A\ 59\ 09 \\ 72\ 91\ 14\ 59\ 5F\ 08\ 6E\ 76 \\ 07\ 18\ AF\ E3\ 65\ BC\ 09\ DE \\ B6\ AF\ A1\ 80\ BC\ EC\ 2A\ 98 \end{pmatrix} \quad x_1 = \begin{pmatrix} 1A\ 78\ 4D\ 7D\ BD\ 4C\ 17\ E6 \\ 27\ 31\ 10\ AA\ 63\ C5\ 9E\ 25 \\ 7A\ 2E\ B7\ 48\ C4\ 5D\ E0\ 23 \\ 6D\ 0D\ 61\ 9F\ 6C\ 1D\ 80\ AE \\ 01\ A2\ D5\ 6E\ DB\ 41\ D9\ A0 \\ E9\ 06\ 4C\ D1\ 27\ 95\ FA\ 86 \\ 77\ 62\ 31\ BC\ B4\ 4E\ C6\ 01 \\ 6F\ CD\ BC\ 98\ 10\ 78\ 6F\ EC \end{pmatrix}$$

$$x_{10} = \begin{pmatrix} B4\ 74\ E1\ 56\ 96\ 31\ B9\ 6C \\ 21\ A1\ B6\ 33\ CC\ 89\ 68\ 1A \\ B1\ 97\ 25\ 86\ 7B\ 2B\ 3F\ 09 \\ 4C\ 73\ C7\ 62\ 93\ A8\ 15\ CF \\ 55\ 15\ C0\ C0\ 9A\ 05\ 05\ 16 \\ 23\ 44\ 8D\ 8D\ D3\ 5F\ B3\ 6E \\ 7E\ 6C\ 2D\ 37\ 12\ D0\ F3\ 3E \\ CE\ B8\ 04\ F2\ 8D\ 9F\ C9\ 99 \end{pmatrix} \quad x_{12} = \begin{pmatrix} 3F\ 72\ C2\ 60\ EE\ 28\ EF\ EA \\ 42\ 8E\ B5\ 3A\ FB\ 8A\ 33\ A2 \\ 03\ E4\ 72\ 31\ 90\ A5\ 1A\ D3 \\ 3E\ 68\ E6\ 46\ FC\ 94\ 3C\ C7 \\ 80\ 42\ 9E\ 2E\ CB\ 32\ 75\ 93 \\ 30\ AA\ E2\ 21\ 21\ C8\ 99\ ED \\ 86\ 1E\ 06\ 9E\ 91\ 1F\ 89\ 6C \\ D2\ 99\ EC\ 7E\ E9\ 0B\ 01\ 10 \end{pmatrix}$$

The last entry corresponds to the final output $\pi(x_0) = x_{12}$.

B.2 Authenticated Encryption

Inputs are plain ASCII.

K	= "192-bit Secret Key value"	(24 Bytes)
N	= "Nonces Used Once"	(16 Bytes)
A	= "AAD Test Vector Exact Block 32 B"	(32 Bytes)
P	= "2 Block Test Vector for stribob192r2d2"	(38 Bytes)

Authenticated ciphertext has 38 message bytes + 16 for MAC = 54 (0x36) bytes:

```
     x0 x1 x2 x3  x4 x5 x6  x7 x8 x9 xA xB  xC xD xE xF
C =  59 9C 5F 69  7F 16 30 07  B4 D5 52 30  24 0C 2B 7B |0x
     0A 93 4E 4C  63 19 4F AC  EA 2D D5 4E  BD 05 61 2C |1x
     19 92 47 FC  A1 97 AE AE  71 0F 0D ED  3E 56 5B D0 |2x
     26 FE 20 F6  4A 4F                                 |3x
```

For BLNK padding technical implementation details, see [53].