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

Markku-Juhani O. Saarinen

Norwegian University of Science and Technology
mjos@item.ntnu.no

Abstract. WhirlBob is a new Authenticated Encryption with Associated Data (AEAD) algorithm derived from the first round CAESAR candidate StriBob and the Whirlpool hash algorithm. The main advantage of WhirlBob over StriBob is its greatly reduced implementation footprint on resource-constrained platforms. Remarkably, the entire C reference implementation of WhirlBob 1.0 π fits onto a single page of the Appendix. On most low-end microcontrollers the total software footprint of π+BLNK = WhirlBob AEAD is less than half a kilobyte. The greatly reduced hardware gate count is also reflected as efficient bitsliced straight-line implementations, especially on 64-bit platforms. Bitslicing works as an efficient countermeasure against AES-style cache timing side-channel attacks. The new design utilizes only the LPS or ρ keying line of Whirlpool in a flexible domain-separated Sponge mode BLNK and adds the number of rounds in π permutation from 10 to 12 as a countermeasure against Rebound Distinguishing attacks. As with StriBob, the reduced-size Sponge design has a strong provable security link with the original hash algorithm. We finally present some discussion and analysis on differences between Whirlpool, the Russian GOST Streebog hash, and the recently proposed draft Russian Encryption Standard Kuznyechik.

Keywords: Authenticated Encryption, Sponge designs, Whirlpool, Streebog, GOST R 34.11-2012, StriBob, CAESAR.

1 Introduction

WhirlBob 1.0 is an Authenticated Encryption with Associated Data (AEAD) algorithm based on the CAESAR candidate StriBob [43, 44] and NESSIE Final Portfolio [31] hash function Whirlpool 3.0 [3]. AEAD algorithms and modes such as GCM [34] provide both confidentiality and integrity protection in a single pass, thus eliminating the requirement for an MAC algorithm such as HMAC [35]. This has clear advantages for performance and implementation footprint. Extensible-Output Functions (XOFs) such as SHA3 SHAKE [36] offer similar features to AEADs in some Sponge modes [10].

Document version 20140731103800.

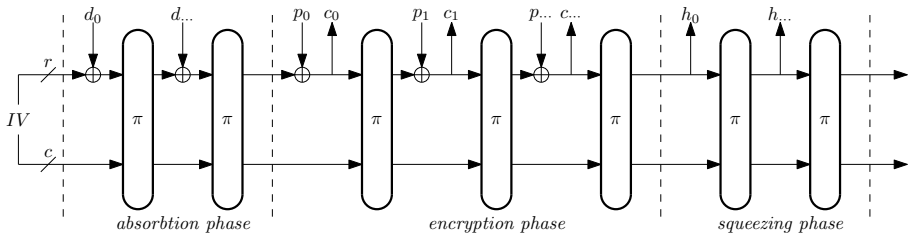

Fig. 1. A simplified view of a Sponge-based AEAD. First the padded Secret Key, Nonce, and Associated Authenticated Data - all represented by d_u words - are "absorbed" or mixed into the Sponge state. The π permutation is then used to also encrypt data p_i into ciphertext c_i (or vice versa) and finally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

2 Motivation: Security Goals and Parameters

WhirlBob uses StriBob's versatile BLNK Sponge AEAD mode 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 [42].

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 – for BLNK padding details and analysis we refer to [23, 42, 44].

As it is clear that the Russian GOST hash standard Streebog [19] was closely modeled after Whirlpool [3], the only difference between StriBob and WhirlBob is in the particular numerical selections for the round constants C_i, the 8 - bit S-Box S, byte permutation P, and the 8×8 MDS matrix L, which is defined over a finite field $\mathsf{GF}(2^8)$. These components, $L \circ P \circ S$ or the "LPS permutation" is derived almost unmodified from that of Whirlpool in present work. Both StriBob and WhirlBob have 12 rounds and the same state size.

The aim is to allow the same secure LPS implementation core (such as a special instruction of a SoC CPU in a mobile or IoT device) to be used for unkeyed hashing according to the Whirlpool standard. This is useful in applications such as certificate processing. The corresponding standardized, Miyagushi-Preneel hash functions require two (or more) times as much as state and processes data in bigger chunks. Our BLNK Sponge mode naturally also supports hashing and MACing without encryption. The Sponge variants are slightly faster.

All of the security parameters remain unmodified. 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 [23, 44] 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. Nonce size is largely arbitrary, but in the standard variant we adopt $n = 128$ bits. See Section 5 for further security analysis.

3 WhirlBob

Despite having almost equivalent speed and size on typical 64-bit platforms, the size and performance characteristics of StriBob and WhirlBob differ significantly in hardware, low-end microcontrollers, and in bitslicing implementations. We therefore suggest using WhirlBob especially in those cases.

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 [3]. The only modification is that the number of rounds is increased from $R = 10$ to $R = 12$. The key schedule operation is also effectively equivalent to the "internal block cipher" W. Eight bytes from the S-Box are used as partial round keys C_i.

WhirlBob's permutation π is indeed highly similar to AES. In case of StriBob, the "Russian 512-bit block AES" permutation had to be uncovered from the structure (See Section 5.3), but the particularities and history of Whirlpool make it immediately obvious.

The 512-bit state is typically seen 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 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 less than 500 bytes. Only slightly more is required for a shared secret handshake protocol and two-way secure BLINKER protocol [42].

This is a significant improvement over StriBob, which typically needs almost 2kB. StriBob is also much slower and larger 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 [3] for techniques that greatly reduce the number of XORs required.

Whirlpool ISO Standard trace test vectors have been used to verify the correctness of this π implementation, up to R=10. 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 6.)

3.2 S-Box Structure, FPGA and Bitsliced Implementation

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.

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

FPGA implementations save a significant number of LUTs by explicitly utilizing the 4-bit structure rather than implementing a general 8×8 lookup table.

The byte-oriented $8 \times 64 = 512$ - bit state can be rapidly split into eight 64-bit registers (the first register representing the bit 0 of each one of the 64 bytes etc). This bitsliced mode can be maintained for the entire 12-round computation. Bitsliced versions of the round constants are naturally used.

The parallelism evident in Figure 2 helps to speed up bitsliced 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. Execution time of a bitsliced software implementation not linearly dependent on the number of instructions.

Appendix B of current 2003 Whirlpool specification [3] gives listings with 14-16 instructions/gates for each of the miniboxes (if ANDN instruction is allowed). Note that we reserve the option to make revisions based on developments in gate optimization that have occurred during the last decade.

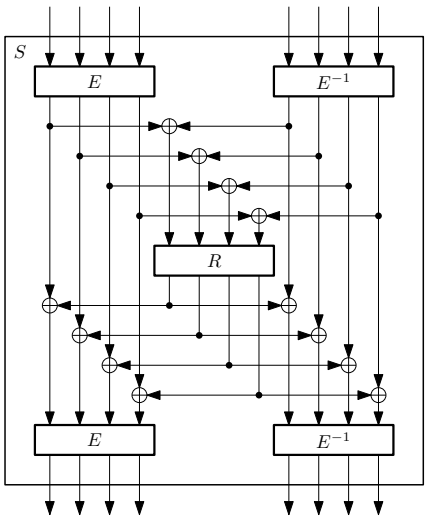

Fig. 2. The 8×8 - bit S-box is constructed from 4×4 - bit "miniboxes".

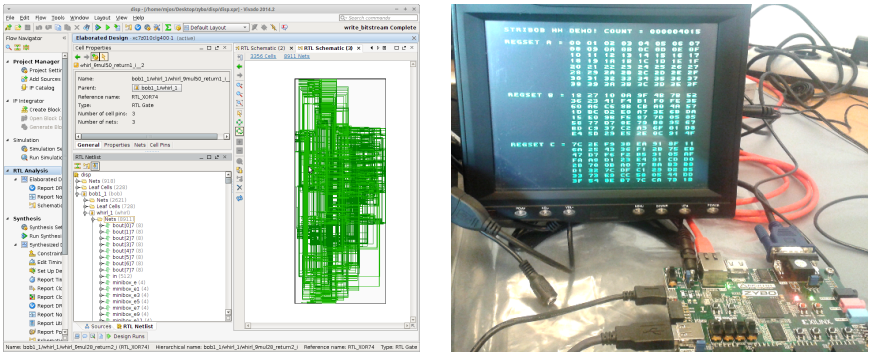

Fig. 3. WhirlBob was implemented in Verilog on the FPGA logic fabric of Xilix Zynq 7010 with Vivado 2014.2 design tools (left). The implementation integrates with the AXI bus of ARM Cortex A9 on the SoC chip, but can also run independently (right).

4 Footprint and Performance

We currently four implementations of the cryptographic transformation π:

- **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.
- **Verilog 12-cycle**: This is the hardware reference implementation. Source code is about 350 lines. Additional logic is required for AXI Bus integration.

The Verilog implementation has been proven on FPGA (Figure 3). Post place-and-route utilization on Zynq 7010 was 3795 LUTs, 1060 registers, and 90 MUXes. Throughput with interface overhead is roughly 2 MB/s for each MHz.

All C implementations are in pure ANSI C99, and hence easily portable (Table 2). The 64-bit implementation is almost exactly as fast as OpenSSL's Whirlpool on the same platform. We additionally have native embedded implementations.

Table 2. Comparing C99 software implementations of WhirlBob's π on a single core of 2.8GHz Core i7 860. No architecture-dependent intrinsics or techniques are used.

Metric	**8-bit ref.**	**64-bit std.**	**Bitsliced**
Throughput (MBytes / sec)	4.533	93.564	30.046
Footprint (code + data bytes)	326 + 256	1942 + 16512	4592 + 768
C Source lines (without tables)	50	125	350

4.1 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 [32, 34], which is recommended for use with TLS, SSH and IPSec protocols by NSA as part of "Suite B" [15, 20, 37, 45]. 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 be more efficient to use the CCM [33, 49] 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 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.

Currently only unoptimized reference implementations are available for most CAESAR candidates [16], making fair performance comparisons difficult. Little attention has been paid to 8-bit or hardware implementations, and the speed difference between some "reference" and "target optimized" implementations can be several orders of magnitude. However, we note that the reference implementations of StriBob and WhirlBob are about 75 % faster than the "Lake Keyak" reference instance of SHA3 / Keccak - based candidate Keyak [10]. WhirlBob falls significantly (by a factor of 3 or more) from candidates such as NORX [2] and MORUS [50], which have been specifically designed for 64-bit targets.

5 Security Analysis

Most of the security arguments and proofs offered for StriBob in [44] 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 [44]) that loosely ties the Miyagushi-Preneel mode [30, 40] with the indistinguishably of π. A random-indistinguishable π and appropriate padding rules are sufficient to construct Sponge-based hashes [5], Tree Hashes [9], MACs [8], Authenticated Encryption (AE) algorithms [7, 10], and pseudorandom extractors (SHAKEs, PRFs, and PRNGs) [6, 36].

5.1 Side-Channel and Implementation Attacks

Whirlpool is better suited for bitsliced implementation due to its particular S-Box and MDS design (as noted in Section 3.2). As an unconditional straight-line code without data-dependent table lookups, a bitsliced implementation is an effective countermeasure against cache timing attacks, which have been found

to be effective against cryptographic primitives with large tables such as AES [1, 4, 39, 48].

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.

5.2 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 [31]. Whirlpool has also been standardized by ISO as part of ISO/IEC 10118-3:2004 [21].

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 [46].

Whirlpool was found to be vulnerable to a Rebound Distinguisher [25, 26, 29]. 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.

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

The GOST R 34.11-2012 "Streebog" standard text [19] does not describe the linear step as a 8×8 matrix-vector multiplication with $\mathsf{GF}(2^8)$ elements like the StriBob spec [44], 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 [24] 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" [47].

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 [28]. We can easily observe that it offers reasonable resistance against basic methods of cryptanalysis. Its differential bound [11] is $P = \frac{8}{256}$ and best linear approximation [27] holds with

$P = \frac{28}{128}$. There does not seem to be any exploitable algebraic weaknesses. These are the exactly same bounds as can be found for Whirlpool S-Box, but fall clearly short from the bounds of the AES S-Box.

The Rijndael AES S-box is constructed of from finite field inversion x^{-1} operation in $\mathsf{GF}(2^8)$ (inspired by the Nyberg construction [38]) and an affine bit transform that serves as a countermeasure against, among other things, Interpolation Attacks [22] on AES' predecessor SHARK [41]. We refer to [18] for more information about the AES design process.

The author 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 [17] 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 [12–14].

6 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 implementation. A bitsliced implementation is an effective countermeasure against cache timing attacks, which have been a concern against AES. The $b = 8 \times 64$ - bit state size is particularly suitable for bitslicing of an byte-oriented algorithm on 64-bit 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. StriBob uses this S-Box as well, and

we feel that it offers better long-term security than Whirlpool, if appropriately implemented.

However WhirlBob has superb implementation characteristics on lightweight platforms and 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 [44].

Acknowledgements

The author wishes to thank Oleksandr Kazymyrov, Vasily Shishkin, Bart Preneel, and Paulo Barreto for their helpful comments. This work was carried out 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. Aumasson, J.-P., P.Jovanovic, and Neves, S. CAESAR submission: NORX v1. `competitions.cr.yp.to/round1/norxv1.pdf`, March 2014.
3. Barreto, P. S. L. M., and Rijmen, V. The Whirlpool hashing function. NESSIE Algorithm `www.larc.usp.br/~pbarreto/WhirlpoolPage.html`, 2000, Revised May 2003.
4. Bernstein, D. J. Cache-timing attacks on AES. Tech. rep., University of Chigaco, 2005.
5. Bertoni, G., Daemen, J., Peeters, M., and Assche, G. V. Sponge functions. In *Ecrypt Hash Workshop 2007* (May 2007).
6. 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.
7. 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.
8. 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).
9. 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.

10. 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.
11. Biham, E., and Shamir, A. *Differential Cryptanalysis of the Data Encryption Standard*. Springer, 1993.
12. 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.
13. 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.
14. 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.
15. Burgin, K., and Peck, M. Suite B Profile for Internet Protocol Security (IPsec). IETF RFC 6380, October 2011.
16. CAESAR. CAESAR submissions. competitions.cr.yp.to/caesar-submissions.html, March 2014.
17. Courtois, N. How fast can be algebraic attacks on block ciphers? IACR ePrint 2006/168, eprint.iacr.org/2006/168, May 2006.
18. Daemen, J., and Rijmen, V. *The Design of Rijndael: AES - the Advanced Encryption Standard*. Springer, 2002.
19. GOST. *Information Technology. Cryptographic Protection of Information, Hash Function*. No. R 34.11-2012 in GOST. Gosudarstvennyi Standard of Russian Federation, 2012. (In Russian).
20. Igoe, K. Suite B Cryptographic Suites for Secure Shell (SSH). IETF RFC 6239, May 2011.
21. ISO/IEC. Information technology – security techniques – hash-functions – part 3: Dedicated hash-functions. ISO/IEC 10118-3:2004, 2004.
22. 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.
23. 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.
24. 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.
25. 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.
26. 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.
27. Matsui, M. Linear cryptoanalysis method for DES cipher. In *EUROCRYPT '93* (1994), T. Helleseth, Ed., vol. 765 of *LNCS*, Springer, pp. 386–397.
28. Matyuhin, D. V., Rudskoy, V. I., and Shishkin, V. A. Promising hashing algorithm. RusCrypto '10 Workshop, 02 April 2010, 2010. (In Russian).
29. 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.

30. Miyaguchi, S., Ohta, K., and Iwata, M. 128-bit hash function (n-hash). *NTT Review*, 2 (1990), 128–132.
31. 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`.
32. NIST. Advanced Encryption Standard (AES). FIPS 197, November 2001.
33. NIST. Counter with Cipher Block Chaining - Message Authentication Code (CCM). NIST Special Publication 800-38C, May 2004.
34. NIST. Recommendation for block cipher modes of operation: Galois/counter mode (GCM) and GMAC. NIST Special Publication 800-38D, November 2007.
35. NIST. The keyed-hash message authentication code (HMAC). FIPS 198-1, July 2008.
36. NIST. DRAFT SHA-3 standard: Permutation-based hash and extendable-output functions. DRAFT FIPS 202, May 2014.
37. NSA. Suite B Cryptography. `www.nsa.gov/ia/programs/suiteb_cryptography`, June 2014.
38. Nyberg, K. Differentially uniform mappings for cryptography. In *EUROCRYPT '93* (1993), T. Helleseth, Ed., vol. 765 of *LNCS*, Springer, pp. 55–64.
39. 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.
40. Preneel, B. *Analysis and Design of Cryptographic Hash Functions*. PhD thesis, K. U. Leuven (Belgium), 1993.
41. 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.
42. 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.
43. 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.
44. Saarinen, M.-J. O. The STRIBOBr1 authenticated encryption algorithm. CAESAR First Round Candidate, `www.stribob.com`, March 2014.
45. Salter, M., and Housley, R. Suite B Profile for Transport Layer Security (TLS). IETF RFC 6460, January 2012.
46. Shirai, T., and Shibutani, K. On the diffusion matrix employed in the Whirlpool hashing function. NESSIE Public Report, 2003.
47. 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.
48. 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.
49. Whiting, D., Housley, R., and Ferguson, N. Counter with CBC-MAC (CCM). IETF RFC 3610, September 2003.
50. 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
            }
        }
    }
}
```