STRIBOB : Authenticated Encryption from GOST R 34.11-2012 LPS or Whirlpool

Markku-Juhani O. Saarinen

mjos@item.ntnu.no

NTNU

Norwegian University of Science and Technology

Directions in Authentication Ciphers '14
24 August 2014, Santa Barbara USA

STRIBOB Ideas

- Security bounds derived from Sponge Theory.
- Well-understood fundamental permutation: Security reduction to Streebog or Whirlpool, with rounds increased $10 \rightarrow 12$.
- Recyclable hardware components.
  - STRIBOBr1: Streebog LPS.
  - *STRIBOBr2d1: Streebog LPS.*
  - *STRIBOBr2d2: Whirlpool LPS - "WhirlBob".*
- Flexible, extensible domain separation with the BLNK Mode ["Beyond Modes: Building a Secure Record Protocol from a Cryptographic Sponge Permutation", CT-RSA 2014.]
  - "Explicit Domain Separation".
  - Fully adjustable security parameters.
  - MAC-then-continue / sessions, Half-duplex protocols..

Fairly conservative design..

History & Real World Crypto

Stewed beef, GOST 5284-84

GOST Spam

a.k.a. Tushonka

- 28149-89 Block Cipher (KGB, 1970s)
- R 34.11-94 was a hash (based on 28149-89) for R 34.10-94 signatures.
- Cryptanalysis by F. Mendel et al (2008): 2^{105} collision, 2^{192} preimage.
- R 34.11-2012 "Streebog" hash algorithm proposed in 2009.
- Since January 1, 2013, the Russian Federation has **mandated** the use of R 34.11-2012 (with R 34.10-2012).
- AES "monoculture" is not universally trusted in some parts of the world.
- **STRIBOB** builds a sponge AEAD algorithm from Streebog, perhaps acceptable in those markets.

GOST R 34.11-2012 "Streebog"

Streebog is a (non-keyed) hash function that produces a 256-bit or 512-bit message digest for a bit string of arbitrary length.

Streebog is Clearly AES & Whirlpool-inspired. Intended for Digital Signatures (R 34.10-2012). Also used in HMAC mode.

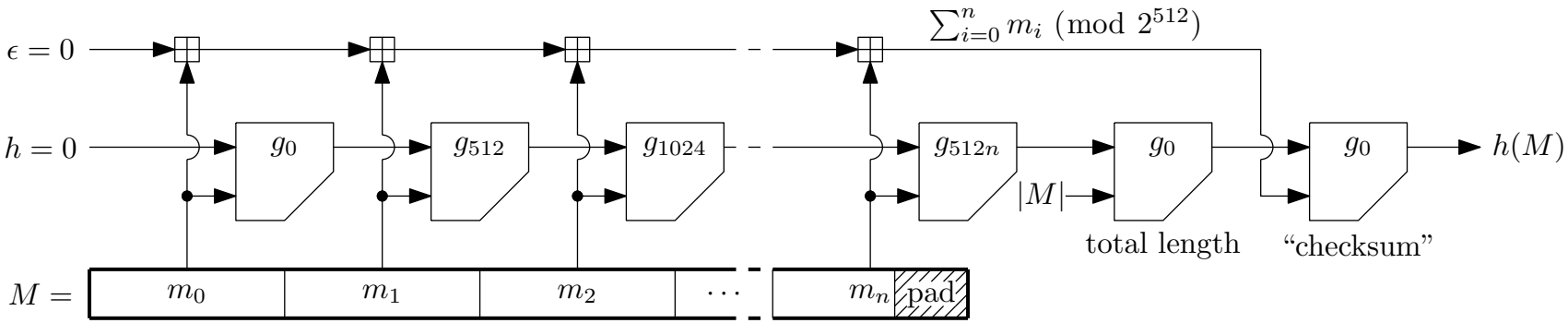
Standard security claims:

- **Collision resistance**:
  m_1 and m_2, $h(m_1) = h(m_2)$ requires $2^{\frac{n}{2}}$ effort.
- **Pre-image resistance**:
  m for given h in $h = H(m)$ requires 2^n effort.
- **Second pre-image resistance**:
  m_2 for given m_1 with $h(m_1) = h(m_2)$ requires $\frac{2^n}{|m_2|}$ effort.

Not a Sponge, but a Miyaguchi–Preneel - inspired construction:

$$h_i = E_{g(H_{i-1})}(m_i) \oplus h_{i-1} \oplus m_i.$$

GOST Streebog: Computing $h(M)$

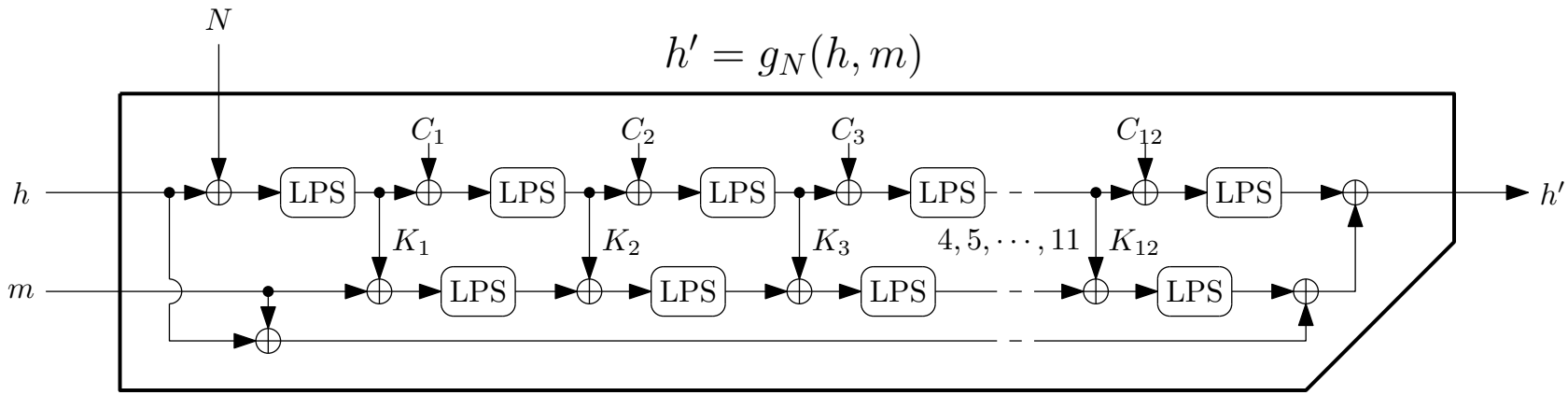


Padded message M is processed in 512-bit blocks $M = m_0 \mid m_1 \mid \cdots \mid m_n$ by a compression function $h' = g_N(h, m_i)$.

Chaining variable h has 512 bits. N is the bit offset of the block.

There are finalization steps involving two invocations of g, first on the total bit length of M, and then on checksum ϵ, which is computed over all input blocks mod 2^{512}.

Streebog: The Compression Function $g_N(h, m)$



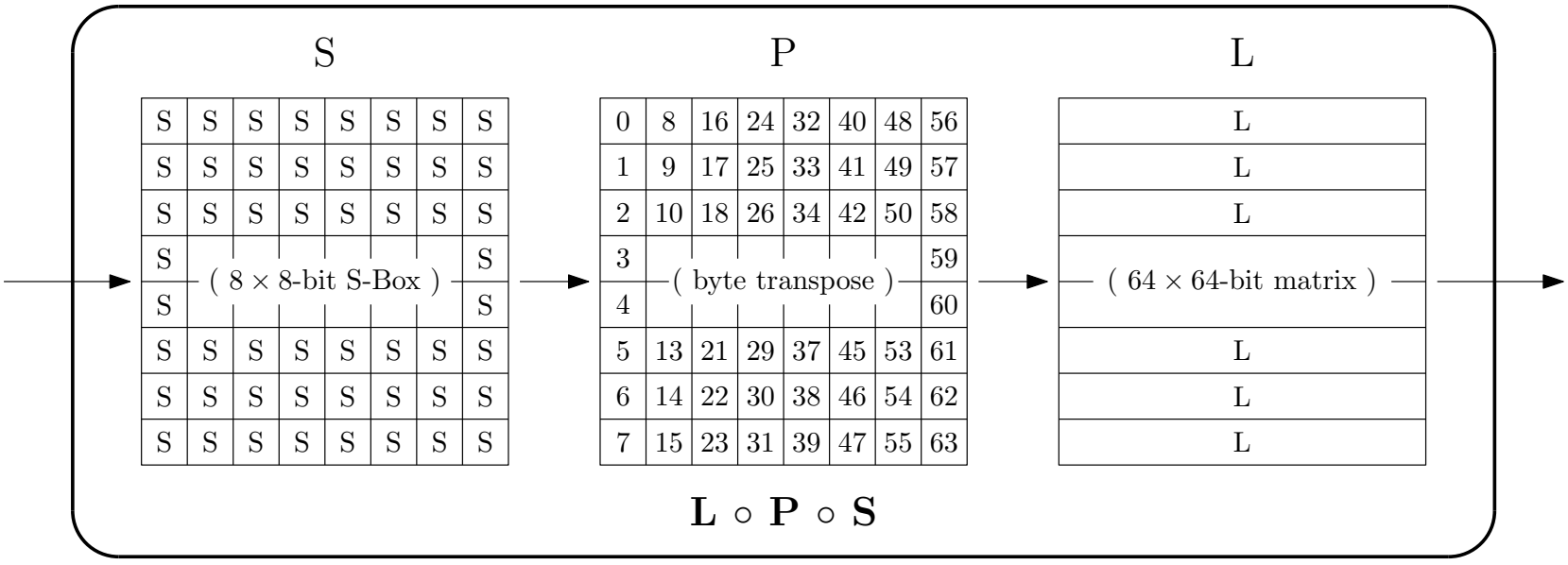
N: bit offset h: chaining value m: 512-bit message block

The compression function is built form a 512×512 - bit **keyless** permutation LPS and XOR operations. All data paths are 512 bits.

The 12 random round constants C_i are given in the standard spec.

One can see the upper "line" (kinda) keying the lower line via K_i.

Streebog: LPS $= L \circ P \circ S = L(P(S(x)))$



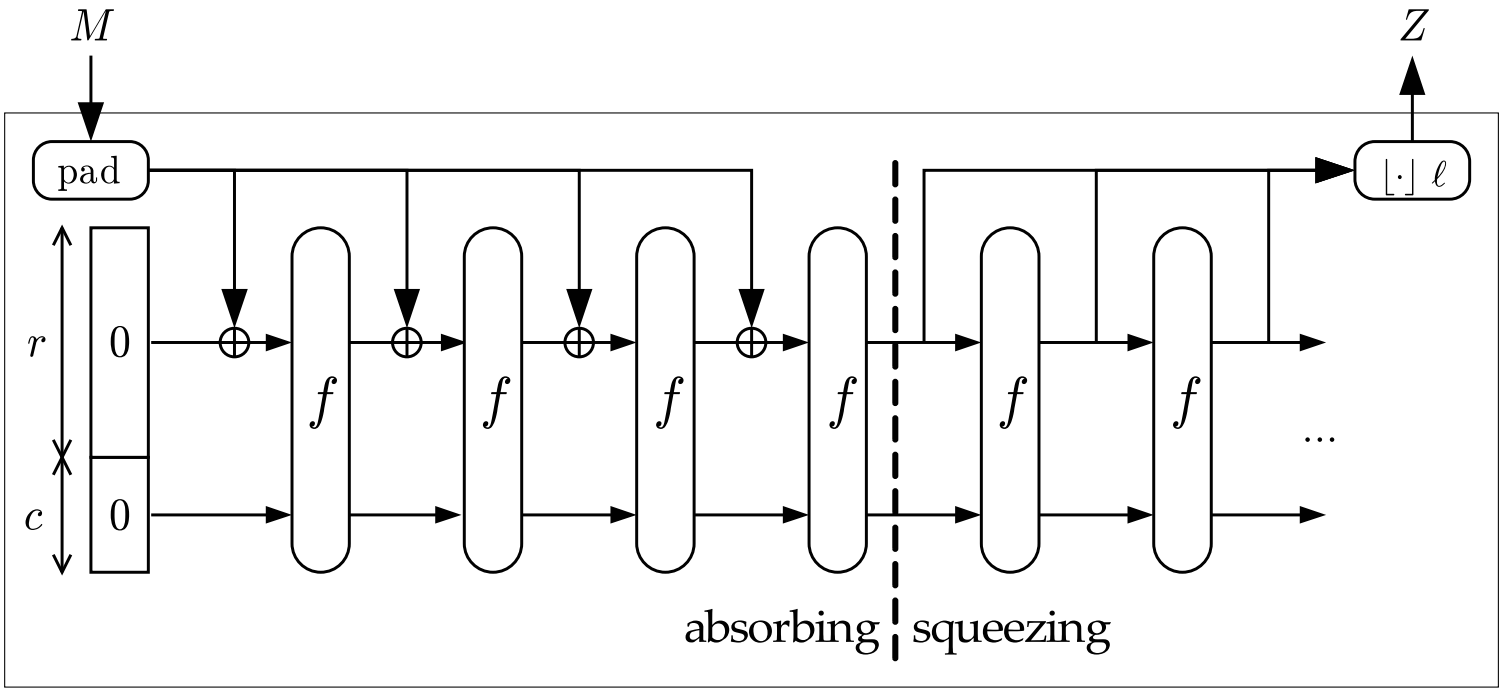
S : ("Substitution") An 8×8 - bit S-Box applied to each one of 64 bytes ($8 \times 64 = 512$ bits).

P : ("Permutation") Transpose of 8×8 - byte matrix.

L : ("Linear") Mixing of **rows** with a 64×64 binary matrix.

[KaKa13] L is actually an 8×8 MDS Matrix in $\mathrm{GF}(2^8)$

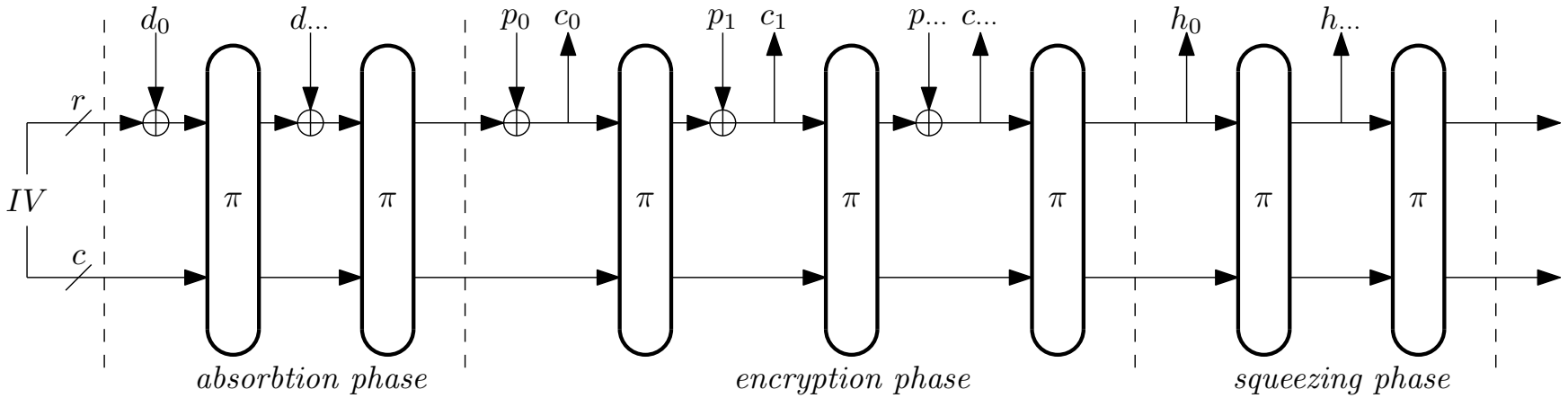
vs.. Sponge Construction for Hashing (SHA3)



sponge

- Built from a b-bit permutation f (π) with $b = r + c$
  - r bits of rate, related to hashing speed
  - c bits of capacity, related to security
- More general than traditional hash: arbitrary-length output

vs.. Sponge-based Authenticated Encryption Æ



1. **Absorption.** Key, nonce, and associated data (d_i) are mixed.
2. **Encryption.** Plaintext p_i is used to produce ciphertext c_i.
3. **Squeezing.** Authentication Tag h_i is squeezed from the state.
4. Why not use that final state as IV for reply and go straight to Step 2 ? *(feature called "sessions" in Ketje and Keyak)*

[Sa14a] **BLNK** mode defines "explicit domain separation" and applies that to build ultra-light weight half-duplex protocols.

DuplexWrap (basic Sponge Æ Scheme) Bounds

Theorem

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, and π is a randomly chosen permutation. M is the data complexity (total number of blocks queried) and N is the time complexity (in equivalents of π).

Proof.

Theorem 4 of [KeyakV1]. See also [AnMePr10,BeDaPeAs11]. □

STRIBOB: Sponge Permutation π

For some vector of twelve 512-bit subkeys C_i we define a 512-bit permutation $\pi_C(X_1) = X_{13}$ with iteration

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

We adopt 12 rounds of LPS as the Sponge permutation with:

- b Permutation size $b = r + c = 512$, the LPS permutation size.
- r Rate $r = 256$ bits.
- c Capacity $c = 256$ bits.

As π satisfies the indistinguishability criteria, we may choose:

- k Key size $k = 192$ bits.
- t Authentication tag (MAC) size $t = 128$ bits.
- k Nonce (IV) size $t = 128$ bits.

Easy Security Reduction

Theorem

If $\pi_C(x)$ can be effectively distinguished from a random permutation for some C_i, 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

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

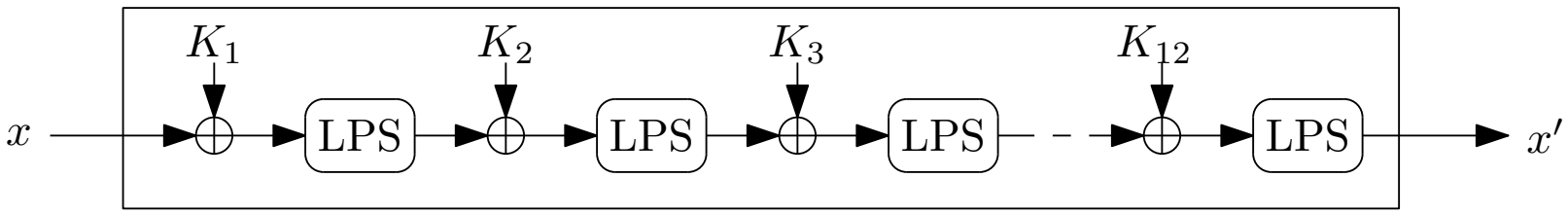
Assuming that the round constants C_i offer no advantage over known round keys K_i, π_C is as secure as π_K and any distinguisher should have the same complexity. □

We see that a generic powerful attack against π is also an attack on g. A distinguishing attack against g does not imply a collision attack against Streebog as a whole.

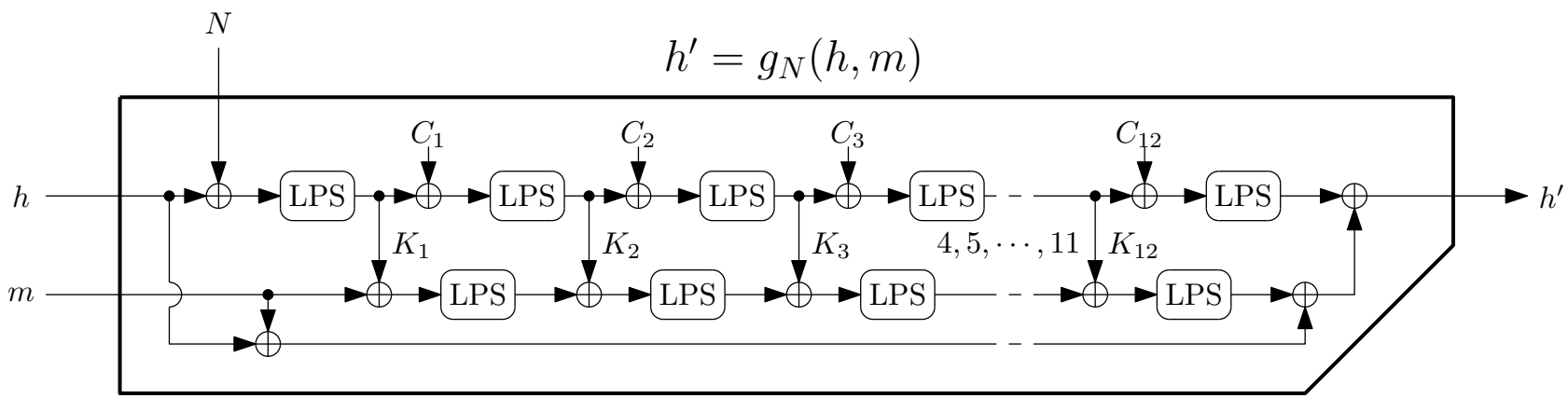
Security Reduction Explained

STRIBOB: Just replace C with K in π:

$$x' = \pi_K(x)$$



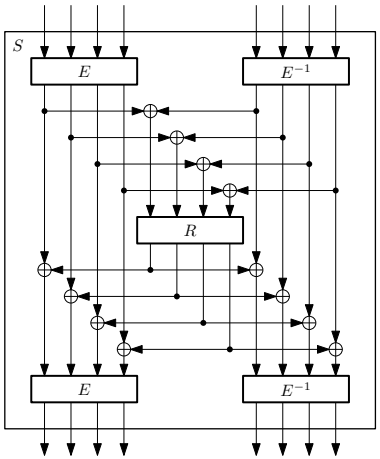
Streebog: We have $g_N(h, x) \oplus x \oplus h = \pi_K(x \oplus N)$:

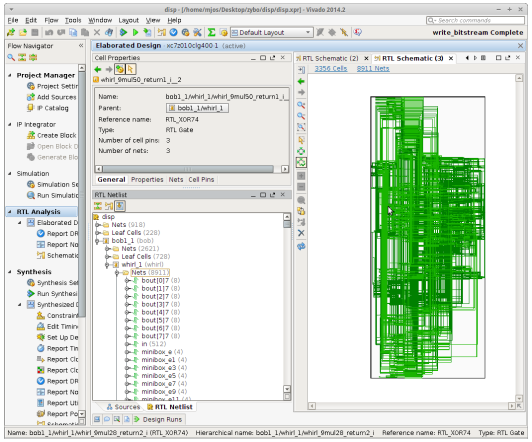


WHIRLBOB Variant (STRIBOBr2d2)

Whirlpool is a NESSIE final portfolio algorithm and an ISO standard. If STRIBOB is accepted to R2, we will add a variant which is more directly based on Whirlpool [RiBa00] v3.0 [RiBa03].

- STRIBOBr1
- *STRIBOBr2d1 = STRIBOBr1*
- *STRIBOBr2d2 a.k.a.* **WHIRLBOB**





S-Box structure saves hardware gates & makes bitslicing faster. Current constant-time (timing attack resistant) bitsliced version runs at about 35 % of table lookup -based implementation.

STRIBOB Software Performance

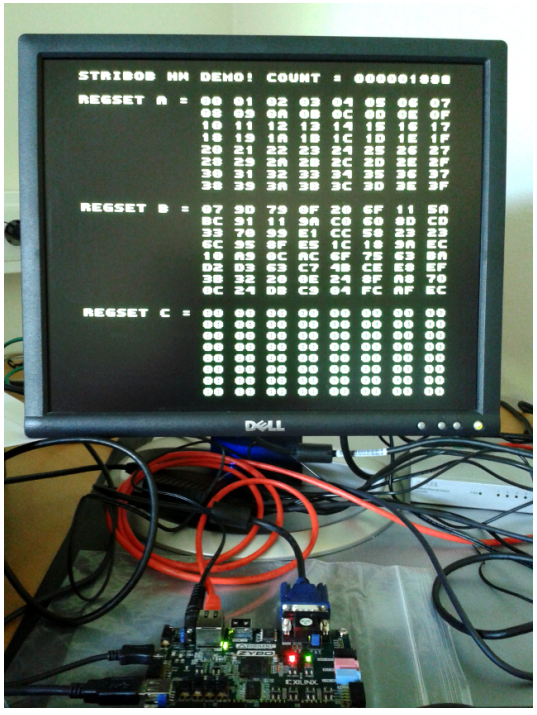
STRIBOB requires 12 LPS invocations per 256 bits processed whereas Streebog requires 25 LPS invocations per 512 bits: STRIBOB is faster. Also the runtime memory requirement is cut down to 25 %. WHIRLBOB performance is equal to STRIBOB.

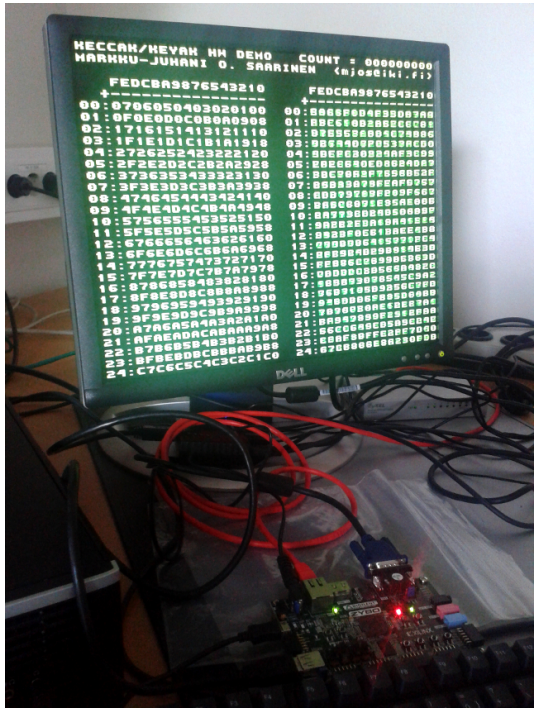
Implementation techniques are similar to AES. 64-bit "rows" are better suited for 64-bit architectures (AES is from 90s, 32-bit era).

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
(bitsliced WHIRLBOB)	> 40 MB/s -- w. current S-Boxes

..as measured on my few years old Core i7 @ 2.80.

Briefly about FPGA Implementations





Total logic on Xilinx Artix-7: WHIRLBOB: 4,946, Keyak 7,972

Report on these & a Proposal for CAESAR HW/SW API:

"Simple AEAD Hardware Interface (SÆHI) in a SoC: Implementing an On-Chip Keyak/WhirlBob Coprocessor", ePrint 2014/575.

Mikko Hypponen, CRO of F-Secure, 29 Apr 2014.

- Implementation of **secure links** over TCP using the BLNK protocol. Can be used as a secure replacement for **`netcat`**.
- File **encryption** and **decryption** using an authenticated chunked file format; you can efficiently encrypt a backup stream up to terabytes in size.
- **Hashing** of files and streams. StriCat can also do 256- and 512-bit standard-compliant GOST **Streebog** hashes.
- Portable, self-contained, **open source**, POSIX compliant, relatively small (couple of thousand lines).

Originally written to debug real-world BLNK..


```
$ ./stricat -h
stricat: STRIBOB / Streebog Cryptographic Tool.
(c) 2013-4 Markku-Juhani O. Saarinen <mjos@iki.fi>. See LICENSE.

stricat [OPTION].. [FILE]..
 -h          This help text
 -t          Quick self-test and version information

Shared secret key (use twice to verify):
 -q          Prompt for key
 -f <file>   Use file as a key
 -k <key>    Specify key on command line

Files:
 -e          Encrypt stdin or files (add .sb1 suffix)
 -d          Decrypt stdin or files (must have .sb1 suffix)
 -s          Hash stdin or files in STRIBOB BNLK mode (optionally keyed)
 -g          GOST R 34.11-2012 unkeyed Streebog hash with 256-bit output
 -G          GOST R 34.11-2012 unkeyed Streebog hash with 512-bit output

Communication via BLNK protocol:
 -p <port>   Specify TCP port (default 48879)
 -c <host>   Connect to a specific host (client)
 -l          Listen to incoming connection (server)
```

http://www.stribob.com/stricat

References..

Sa14a "Beyond Modes: Building a Secure Record Protocol from a Cryptographic Sponge Permutation" *CT-RSA 2014, IACR ePrint 2013/772*.

Sa14b "STRIBOB: Authenticated Encryption from GOST R 34.11-2012 LPS Permutation (Extended Abstract)" *CTCrypt '14, IACR ePrint 2014/271*.

Sa14c "Lighter, Faster, and Constant-Time: WHIRLBOB, the Whirlpool variant of STRIBOB", *Submitted for publication, ePrint 2014/501*.

Sa14d "Simple AEAD Hardware Interface (SÆHI) in a SoC: Implementing an On-Chip Keyak/WhirlBob Coprocessor", *Submitted for publication, IACR ePrint 2014/575*.

`http://www.stribob.com` `http://www.mjos.fi`