What output size resists collisions in a xor of independent expansions?

D. J. Bernstein
University of Illinois at Chicago

Hashes from stream ciphers?

Salsa20 stream cipher
allows fast random access:
expands 32-byte secret key,
8-byte nonce,
8-byte block counter
into 64-byte output block.

Alternative: Reduced rounds.

Alternative: ChaCha20,
newer variant of Salsa20.

Alternative: Traditional
LFSR-based stream ciphers.
Random access via exp.

Reuse this 48-byte-to-64-byte function in a SHA replacement?

Different security goals, but some sharing of (e.g.) differential cryptanalysis; some sharing of software; some sharing of hardware.

Analogous to classic study of reusing block ciphers. Presumably better speed by reusing stream ciphers.

Rumba20, a compression function

Salsa20 : $(\mathbf{Z}/256)^{48} \to (\mathbf{Z}/256)^{64}$.

Tweak the "diagonal constants" to make four new functions

$f_1 : (\mathbf{Z}/256)^{48} \to (\mathbf{Z}/256)^{64}$,
$f_2 : (\mathbf{Z}/256)^{48} \to (\mathbf{Z}/256)^{64}$,
$f_3 : (\mathbf{Z}/256)^{48} \to (\mathbf{Z}/256)^{64}$,
$f_4 : (\mathbf{Z}/256)^{48} \to (\mathbf{Z}/256)^{64}$.

Define function Rumba20 : $(\mathbf{Z}/256)^{192} \to (\mathbf{Z}/256)^{64}$ by Rumba20$(m_1, m_2, m_3, m_4) = f_1(m_1) \oplus f_2(m_2) \oplus f_3(m_3) \oplus f_4(m_4)$.

Main question in this talk:
How cheaply can we find
collisions in Rumba20?

In the absence of collisions,
many reasonable ways
to build SHA replacement
using Rumba20.
Use output filter
to hide linear structure.
Optionally truncate after filter.

Rumba20 cycles/compressed byte
$\approx 2 \cdot$ Salsa20 cycles/byte.
Generally faster than SHA-256.
Can reduce rounds to save time.

Some literature

1996 Bellare/Micciancio:

Compress $(m_1, m_2, \ldots)$
to $f_1(m_1) \oplus f_2(m_2) \oplus \cdots$.
Good “incremental” speed.

Collision resistant?

Easy collisions for long inputs.
Not so easy if $\oplus$ is replaced by
$+$, vector $+$, modular $\cdot$, etc.

Shorter inputs seem ok.

1999 van Oorschot/Wiener: Parallel collision search.

2002 Wagner, "generalized birthday attack": impressively fast collisions for $\oplus$, $+$, vector $+$ for medium-length inputs.

Speed not so impressive for short inputs.
Also, heavy memory use.

Open questions from Wagner:
Smaller memory use?
Parallelization "without enormous communication complexity"?

Review of Wagner's attack

Wagner says:
Choose 2^{128} values of m_1
and 2^{128} values of m_2.

Sort all pairs $(f_1(m_1), m_1)$
into lexicographic order.
Sort all pairs $(f_2(m_2), m_2)$
into lexicographic order.

Merge sorted lists to find
$\approx 2^{128}$ pairs (m_1, m_2)
such that first 128 bits
of $f_1(m_1) \oplus f_2(m_2)$ are 0.

Compute $\approx 2^{128}$ vectors $(f_1(m_1) \oplus f_2(m_2), m_1, m_2)$ where first 128 bits are 0.

Sort into lexicographic order.

Similarly $f_3(m_3) \oplus f_4(m_4)$.

Merge to find $\approx 2^{128}$ vectors (m_1, m_2, m_3, m_4) such that first 256 bits of $f_1(m_1) \oplus f_2(m_2) \oplus f_3(m_3) \oplus f_4(m_4)$ are 0.

Sort to find ≈ 1 collision in all 512 bits of $f_1(m_1) \oplus f_2(m_2) \oplus f_3(m_3) \oplus f_4(m_4)$.

Generalize 128 to b. Sorting:
"$O(n \log n)$ time" where $n = 2^b$.
"A lot of memory":
huge machine storing 2^b vectors.

Compare van Oorschot/Wiener:
Similar time, $\approx 2^b$, using
$\approx 2^b$ parallel search units.
Similar machine cost.
Much more flexibility:
easily use smaller machines.

Normally want collisions in
truncation(scrambling($4b$ bits)).
Truncation reduces b for van
Oorschot/Wiener; not Wagner.

Improving Wagner's attack

1. Search in parallel for m_i's having many 0 bits in $f_i(m_i)$.

2. Use mesh sorting:
sort n items on n parallel cells
in a $\sqrt{n} \times \sqrt{n}$ mesh
in time $\approx \sqrt{n}$.

3. Adapt and optimize parameters to use smaller machine.

4. Streamline everything to save constant factor.

Speed of improved attack,
ignoring constant factors:

Machine cost $2^{8b/9}$, time $2^{4b/9}$.

More generally, for $0 < c \le 8b/9$:
Machine cost 2^c, time 2^{4b-4c}. For
$c < 2b/3$, van Oorschot/Wiener
is better: time 2^{2b-c}.

More generally, for $0 < c \le 8b/9$
and $c/2 \le t \le 4b - 4c$:
Machine cost 2^c, time 2^t,
success chance $2^{t+4c-4b}$.
For $t > 2c$, van Oorschot/Wiener
chance $2^{2t+2c-4b}$ is better.

Status of Rumba20

Best attack at this point:

For small machine costs,
van Oorschot/Wiener.
Price-performance ratio
$AT \approx 2^{256}$.

For large machine costs
(above $\approx 2^{85}$ parallel cells),
this improvement of Wagner.
Best $AT \approx 2^{171}$
with $\approx 2^{114}$ parallel cells.

In theory: can compute AT
given gate/wire costs, speeds.

Cryptanalyst needs
much smaller AT!

Better attack on 4-xor?
Better attack on Rumba20?
On the ChaCha20 variant?
On reduced-round variants?
Quickly generate leading 0's?

I offer \$1000 prize for
the public Rumba20 cryptanalysis
that I consider most interesting.
Awarded at the end of 2007.

Send URLs of your papers to
`snuffle6@box.cr.yp.to`.