ChaCha,
a variant of Salsa20

D. J. Bernstein
University of Illinois at Chicago

NSF ITR–0716498

Note: ChaCha is
not an eSTREAM candidate!
“Post-eSTREAM cryptography.”

The Salsa20 cipher family

Salsa20/20 is faster than AES.
I recommend Salsa20/20
as an AES replacement
for typical applications.

What if the users value speed
more highly than confidence?
Reduce rounds: Salsa20/12 or /8.

Attacks: Crowley, SASC 2006;
Fischer et al., Indocrypt 2006;
Tsunoo et al., SASC 2007;
Aumasson et al., FSE 2008.
Against Salsa20/8, slightly faster
than 256-bit exhaustive search.

The ChaCha cipher family

ChaCha8, ChaCha12, ChaCha20
are like Salsa20/8 etc.:
same key size, same block size,
same arithmetic operations.

But there are some changes
in the order of operations.

Aumasson et al. attack
a preliminary "ChaCha"
with one of these changes.
The change makes the attack
fail one round sooner.

Perhaps the other changes
are also helpful. Need analysis!

Changed order of operations
often saves some CPU time.

Cycles/byte, 576-byte packet,
for ChaCha8 and Salsa20/8:

C8	S8	
2.00	2.14	ppc32 PowerPC G4
2.09	2.07	amd64 Core 2 Duo
3.47	3.66	amd64 Athlon 64 X2
4.06	5.57	amd64 Pentium D f64
5.05	5.54	x86 Pentium M 695
6.11	5.98	x86 Pentium 4 f12
6.27	7.06	x86 Pentium 4 f41
6.71	6.76	sparc UltraSPARC III
7.02	7.20	ppc64 Cell PPE

Salsa20 quarter-round

4 modifications per quarter-round:

```
x4 ^= (x0+x12) <<< 7
x8 ^= (x4+x0) <<< 9
x12 ^= (x8+x4) <<< 13
x0 ^= (x12+x8) <<< 18
```

16 modifications per round.
128 modifications in Salsa20/8.
320 modifications in Salsa20/20.

Key words kept separate? No!
("I also see each use of a k word as a missed opportunity to spread changes through the n words.")
Constants kept separate? No!

ChaCha quarter-round

Salsa20 temporarily allocates
separate word for storing sum.
Missed diffusion opportunity!

ChaCha takes this opportunity:

```
x0+=x12; x4^=x0; x4<<<=16
x8+=x4; x12^=x8; x12<<<=12
x0+=x12; x4^=x0; x4<<<=8
x8+=x4; x12^=x8; x12<<<=7
```

In absence of carries,
average 1-bit differential
affects 12.5 output bits.
Salsa20: only 8 output bits.

Salsa20 round switch

Salsa20 sweeps down columns:

`x4`, `x8`, `x12`, `x0`;

`x9`, `x13`, `x1`, `x5`;

`x14`, `x2`, `x6`, `x10`;

`x3`, `x7`, `x11`, `x15`.

Then across rows:

`x1`, `x2`, `x3`, `x0`;

`x6`, `x7`, `x4`, `x5`;

`x11`, `x8`, `x9`, `x10`;

`x12`, `x13`, `x14`, `x15`.

Modifications in these two rounds:
below-diagonal, . . ., . . ., diagonal,
. . ., . . ., below-diagonal, diagonal.

ChaCha round switch

ChaCha sweeps *up* columns:

`x0`, `x12`, `x8`, `x4`;

`x5`, `x1`, `x13`, `x9`;

`x10`, `x6`, `x2`, `x14`;

`x15`, `x11`, `x7`, `x3`.

Then across rows:

`x0`, `x1`, `x2`, `x3`;

`x5`, `x6`, `x7`, `x4`;

`x10`, `x11`, `x8`, `x9`;

`x15`, `x12`, `x13`, `x14`.

Diagonal, . . ., . . ., below-diagonal,
diagonal, . . ., . . ., below-diagonal.
Intuition: better diffusion.

Salsa20 input positions

Only 128 bits of input
are controlled by attacker:
x6, x7, x8, x9.

2^{256} possible input pairs.
$\approx 2^{264}$ possible state pairs.
Wishful-thinking differentials
impose ≈ 512 bit conditions
and will never be entered.
See "Salsa20 security," 2005.04.

Successful reduced-round attacks
have traced 1-bit differential
starting from "best" bit in
x6, x7, x8, x9.

ChaCha input positions

New choice of input positions:
`x1`, `x6`, `x11`, `x12`.

All of these words are modified at second position in quarter-round.

Intuition: less variability
in "goodness" of these bits;
"best" bit for ChaCha isn't
as good as "best" bit for Salsa20.

Also: Output array is permuted.
Better speed; identical security.
Convenient to describe states
with the same permutation.
See paper for details.