Speed, speed, speed

D. J. Bernstein

University of Illinois at Chicago;
Ruhr University Bochum

Reporting some recent
symmetric-speed discussions,
especially from RWC 2020.

Not included in this talk:

- NISTLWC.
- Short inputs.
- FHE/MPC ciphers.

$1000 TCR hashing competition

Crowley: "I have a problem where I need to make some cryptography faster, and I'm setting up a $1000 competition funded from my own pocket for work towards the solution."

Not fast enough: Signing $H(M)$, where M is a long message.

"[On a] 900MHz Cortex-A7 [SHA-256] takes 28.86 cpb . . . BLAKE2b is nearly twice as fast . . . However, this is still a lot slower than I'm happy with."

Instead choose random R
and sign $(R, H(R, M))$.

Note that H needs only "TCR",
not full collision resistance.
Does this allow faster H design?
TCR breaks how many rounds?

Instead choose random R
and sign $(R, H(R, M))$.

Note that H needs only "TCR",
not full collision resistance.
Does this allow faster H design?
TCR breaks how many rounds?

"As far as I know, no-one
has ever proposed a TCR as a
primitive, designed to be faster
than existing hash functions,
and that's what I need."

Instead choose random R
and sign $(R, H(R, M))$.

Note that H needs only "TCR",
not full collision resistance.
Does this allow faster H design?
TCR breaks how many rounds?

"As far as I know, no-one
has ever proposed a TCR as a
primitive, designed to be faster
than existing hash functions,
and that's what I need."

More desiderata: tree hash,
new tweak at each vertex,
multi-message security.

Aumasson, "Too much crypto"

70%, 23%, 35%, 21% rounds or
50%, 8%, 25%, 20% rounds of
AES-128/B2b/ChaCha20/SHA-3
are "broken" or "practically broken".
"Inconsistent security margins".

Aumasson, "Too much crypto"

70%, 23%, 35%, 21% rounds or
50%, 8%, 25%, 20% rounds of
AES-128/B2b/ChaCha20/SHA-3
are "broken" or "practically broken".
"Inconsistent security margins".

"Attacks don't really get better".

Aumasson, "Too much crypto"

70%, 23%, 35%, 21% rounds or 50%, 8%, 25%, 20% rounds of AES-128/B2b/ChaCha20/SHA-3 are "broken" or "practically broken".

"Inconsistent security margins".

"Attacks don't really get better".

"Thousands of papers, stagnating results and techniques".

Aumasson, "Too much crypto"

70%, 23%, 35%, 21% rounds or 50%, 8%, 25%, 20% rounds of AES-128/B2b/ChaCha20/SHA-3 are "broken" or "practically broken". "Inconsistent security margins".

"Attacks don't really get better".

"Thousands of papers, stagnating results and techniques".

"What we want: More scientific and rational approach to choosing round numbers, tolerance for corrections".

New BLAKE3 hash function =
7-round BLAKE2s + tree mode,
parallel XOF + more changes.
“Much faster than MD5, SHA-1,
SHA-2, SHA-3, and BLAKE2.”

New BLAKE3 hash function = 7-round BLAKE2s + tree mode, parallel XOF + more changes. "Much faster than MD5, SHA-1, SHA-2, SHA-3, and BLAKE2."

Crowley: "Android disk crypto is always right up against the wall of acceptable speed (and battery use). Adiantum uses ChaCha12 and is still IMHO too slow. [10.6 Cortex-A7 cycles/byte.] It sometimes seems like no-one in the crypto world feels the user's pain here; it always looks better to call for more rounds."

Huge influence of CPU.

Intel cycles/byte for two ciphers:

#1	#2	Intel microarchitecture
0.37	0.68	2018 Cannon Lake
0.38	0.88	2017 Cascade Lake
0.38	0.89	2017 Skylake-X
1.94	1.90	2016 Goldmont
0.77	0.98	2016 Kaby Lake
0.74	0.95	2015 Skylake
0.77	1.01	2014 Broadwell
0.77	1.03	2013 Haswell
1.71	1.29	2012 Ivy Bridge

Huge influence of CPU.

Intel cycles/byte for two ciphers:

#1	#2	Intel microarchitecture
0.37	0.68	2018 Cannon Lake
0.38	0.88	2017 Cascade Lake
0.38	0.89	2017 Skylake-X
1.94	1.90	2016 Goldmont
0.77	0.98	2016 Kaby Lake
0.74	0.95	2015 Skylake
0.77	1.01	2014 Broadwell
0.77	1.03	2013 Haswell
1.71	1.29	2012 Ivy Bridge

#1: ChaCha12. #2: AES-256.

Deck functions: e.g., Xoofff

Keccak team says: Xoofff takes 0.51 cycles/byte on Skylake-X.

Deck functions are "a new useful API to make modes trivial"; they "allow efficient ciphers".

Deck functions: e.g., Xoofff

Keccak team says: Xoofff takes 0.51 cycles/byte on Skylake-X.

Deck functions are "a new useful API to make modes trivial"; they "allow efficient ciphers".

Syntax of deck function: $F_k : (\{0,1\}^*)^* \to \{0,1\}^\infty$.

Deck functions: e.g., Xoofff

Keccak team says: Xoofff takes 0.51 cycles/byte on Skylake-X.

Deck functions are "a new useful API to make modes trivial"; they "allow efficient ciphers".

Syntax of deck function: $F_k : (\{0,1\}^*)^* \to \{0,1\}^\infty$.

Security goal: PRF.

Deck functions: e.g., Xoofff

Keccak team says: Xoofff takes 0.51 cycles/byte on Skylake-X.

Deck functions are "a new useful API to make modes trivial"; they "allow efficient ciphers".

Syntax of deck function: $F_k : (\{0,1\}^*)^* \to \{0,1\}^\infty$.

Security goal: PRF.

Efficiency goal: quickly compute substring of $F_k(X_0)$, then substring of $F_k(X_0, X_1)$, then substring of $F_k(X_0, X_1, X_2)$, etc.

Deck-Stream: $F_k(N)$.

Deck-Stream: $F_k(N)$.

Deck-MAC: 128 bits of $F_k(M)$.

Deck-Stream: $F_k(N)$.

Deck-MAC: 128 bits of $F_k(M)$.

Deck-SANE session:
128 bits of $F_k(N) \to$ tag;
use more bits of $F_k(N)$
as stream $\to$ ciphertext C_1;
128 bits of $F_k(N, A_1, C_1) \to$ tag;
etc.

Deck-Stream: $F_k(N)$.

Deck-MAC: 128 bits of $F_k(M)$.

Deck-SANE session:
128 bits of $F_k(N) \to$ tag;
use more bits of $F_k(N)$
as stream $\to$ ciphertext C_1;
128 bits of $F_k(N, A_1, C_1) \to$ tag;
etc.

Deck-SANSE: misuse resistance.

Deck-Stream: $F_k(N)$.

Deck-MAC: 128 bits of $F_k(M)$.

Deck-SANE session:
128 bits of $F_k(N) \rightarrow$ tag;
use more bits of $F_k(N)$
as stream $\rightarrow$ ciphertext C_1;
128 bits of $F_k(N, A_1, C_1) \rightarrow$ tag;
etc.

Deck-SANSE: misuse resistance.

Deck-WBC: wide-block cipher.

For speed, the wide-block cipher
combines Xoofff and Xoofffie,
(sort of) built from Xoodoo.

MAC speed

2014 Bernstein–Chou Auth256:
29 bit ops per message bit,
using mults in field of size 2^{256}.

(I've started investigating
bit ops for integer mults.)

MAC speed

2014 Bernstein–Chou Auth256:
29 bit ops per message bit,
using mults in field of size 2^{256}.

(I've started investigating
bit ops for integer mults.)

Encryption sounds slower, but
aims for PRF or PRP or SPRP.
How many rounds are needed
in the context of a MAC?

MAC speed

2014 Bernstein–Chou Auth256:
29 bit ops per message bit,
using mults in field of size 2^{256}.

(I've started investigating
bit ops for integer mults.)

Encryption sounds slower, but
aims for PRF or PRP or SPRP.
How many rounds are needed
in the context of a MAC?

OCB etc. try to skip MAC,
but can these modes safely use
as few rounds as counter mode?

Bit operations per bit of plaintext (assuming precomputed subkeys):

key	ops/bit	cipher
256	54	ChaCha8
256	78	ChaCha12
128	88	Simon: 62 ops broken
128	100	NOEKEON
128	117	Skinny
256	126	ChaCha20
256	144	Simon: 106 ops broken
128	147.2	PRESENT
256	156	Skinny
128	162.75	Piccolo
128	202.5	AES
256	283.5	AES

More virtues of mult-based MACs:

- Easy masking.
- Binary mults: Share area with code-based crypto.
- Integer mults: Share area with lattice-based crypto and ECC.
- Use existing CPU multipliers.

More virtues of mult-based MACs:

- Easy masking.
- Binary mults: Share area with code-based crypto.
- Integer mults: Share area with lattice-based crypto and ECC.
- Use existing CPU multipliers.

If int mults are available anyway, should we renew attention to ciphers that use some mults?

More virtues of mult-based MACs:

- Easy masking.
- Binary mults: Share area with code-based crypto.
- Integer mults: Share area with lattice-based crypto and ECC.
- Use existing CPU multipliers.

If int mults are available anyway, should we renew attention to ciphers that use some mults?

e.g. `x *= 0xdf26f9` is same as `x-=x<<3; x-=x<<8; x+=x<<13`. Mix with `^`, `>>>16`, maybe `+`. Try 16-bit mults for Intel, ARM.