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."

speed, speed

ernstein

ty of Illinois at Chicago;
niversity Bochum

---

ng some recent
ric-speed discussions,
y from RWC 2020.

uded in this talk:
LWC.
inputs.
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
and sign

Note tha
not full
Does thi
TCR bre

ed

is at Chicago;
ochum

---

ecent
discussions,
WC 2020.

is talk:

ers.

$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 ran
and sign $(R, H(R,$

Note that H needs
not full collision re
Does this allow fas
TCR breaks how m

ago;

$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 "TC not full collision resistance. Does this allow faster H des TCR breaks how many roun

$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?

$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?

"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."

$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?

"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.

TCR hashing competition

: "I have a problem
need to make some
raphy faster, and I'm
up a \$1000 competition
from my own pocket for
wards the solution."

enough: Signing $H(M)$,
M is a long message.

900MHz Cortex-A7
56] takes 28.86 cpb ...
2b is nearly twice as
However, this is still a
er 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?

"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.

Aumasso

70%, 23
50%, 8%
AES-128
are "bro
"Inconsi

ng competition

a problem
ake some
er, and I'm
0 competition
wn pocket for
solution."

Signing $H(M)$,
message.

Cortex-A7
8.86 cpb ...
y twice as
this is still a
n 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?

"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 r

70%, 23%, 35%, 2
50%, 8%, 25%, 20
AES-128/B2b/Cha
are "broken" or "p
"Inconsistent secu

ition

n

tion

t for

(M),

...

l a

ith."

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 cryp

70%, 23%, 35%, 21% rounc
50%, 8%, 25%, 20% rounds
AES-128/B2b/ChaCha20/S
are "broken" or "practically
"Inconsistent security margi

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".

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".

"Attacks don't really get better".

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".

"Attacks don't really get better".

"Thousands of papers, stagnating
results and techniques".

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".

"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".

choose random R
$(R, H(R, M))$.

at H needs only "TCR",
collision resistance.
is allow faster H design?
eaks how many rounds?

as I know, no-one
proposed a TCR as a
e, designed to be faster
sting hash functions,
t's what I need."

siderata: tree hash,
ak at each vertex,
essage 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".

"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 BL
7-round
parallel
"Much f
SHA-2,

ndom R
M)).

s only "TCR",
esistance.
ster H design?
many rounds?

no-one
a TCR as a
d to be faster
functions,
need."

tree hash,
n vertex,
urity.

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 has
7-round BLAKE2s
parallel XOF + m
"Much faster than
SHA-2, SHA-3, ar

CR",

sign?

ds?

a

ster

,

3

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".

4

New BLAKE3 hash function

7-round BLAKE2s + tree m

parallel XOF + more change

"Much faster than MD5, SH

SHA-2, SHA-3, and BLAKE

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."

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."

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."

on, "Too much crypto"

%, 35%, 21% rounds or
%, 25%, 20% rounds of
3/B2b/ChaCha20/SHA-3
ken" or "practically broken".
stent security margins".

s don't really get better".

nds of papers, stagnating
nd techniques".

we want: More
c and rational approach
sing round numbers,
e 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."

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 inf
Intel cyc

#1	#
0.37	0.6
0.38	0.8
0.38	0.8
1.94	1.9
0.77	0.9
0.74	0.9
0.77	1.0
0.77	1.0
1.71	1.2

much crypto"

21% rounds or
0% rounds of
aCha20/SHA-3
practically broken".
rity margins".

ally get better".

pers, stagnating
ques".

More
onal approach
numbers,
ctions".

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
Intel cycles/byte f

#1	#2	Intel m
0.37	0.68	2018 C
0.38	0.88	2017 C
0.38	0.89	2017 S
1.94	1.90	2016 G
0.77	0.98	2016 K
0.74	0.95	2015 S
0.77	1.01	2014 B
0.77	1.03	2013 H
1.71	1.29	2012 Iv

to”

ls or

of

HA-3

broken”.

ns”.

tter”.

nating

ach

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 cip

#1	#2	Intel microarchite
0.37	0.68	2018 Cannon Lak
0.38	0.88	2017 Cascade La
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

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

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

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

AKE3 hash function =
BLAKE2s + tree mode,
XOF + more changes.
faster than MD5, SHA-1,
SHA-3, and BLAKE2."

: "Android disk crypto is
ight up against the wall
table speed (and battery
diantum uses ChaCha12
till IMHO too slow.
rtex-A7 cycles/byte.] It
es seems like no-one in
to world feels the user's
e; it always looks better
or 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

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

Deck fu

Keccak
0.51 cyc

Deck fu
API to
they "al

h function =

+ tree mode,

ore changes.

MD5, SHA-1,

nd BLAKE2."

d disk crypto is

gainst the wall

ed (and battery

ses ChaCha12

too slow.

ycles/byte.] It

like no-one in

eels the user's

s looks better

unds."

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

Keccak team says:

0.51 cycles/byte o

Deck functions are

API to make mode

they "allow efficie

=
ode,
es.
HA-1,
2."

pto is
wall
ttery
a12

.] It
e in
ser's
etter

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 ta
0.51 cycles/byte on Skylake-

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

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".

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".

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

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".

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

Security goal: PRF.

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".

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.

fluence of CPU.

cles/byte for two ciphers:

=2	Intel microarchitecture
58	2018 Cannon Lake
38	2017 Cascade Lake
39	2017 Skylake-X
00	2016 Goldmont
08	2016 Kaby Lake
05	2015 Skylake
01	2014 Broadwell
03	2013 Haswell
29	2012 Ivy Bridge

aCha12. #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".

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-St

CPU.

or two ciphers:

icroarchitecture

annon Lake

ascade Lake

kylake-X

oldmont

aby Lake

kylake

roadwell

aswell

vy Bridge

#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".

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: $F_k($

hers:

ecture

ke

ke

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".

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

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-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 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-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 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-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-WBC: wide-block cipher.

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

nctions: e.g., Xoofff

team says: Xoofff takes
les/byte on Skylake-X.

nctions are "a new useful
make modes trivial";
low efficient ciphers".

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

goal: PRF.

cy goal: quickly compute
g of $F_k(X_0)$, then
g of $F_k(X_0, X_1)$, then
g of $F_k(X_0, X_1, X_2)$, 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-WBC: wide-block cipher.

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

MAC sp

2014 Be
29 bit o
using m

(I've sta
bit ops f

g., Xoofff

: Xoofff takes
n Skylake-X.

e "a new useful
es trivial";
nt ciphers".

nction:
$\{0,1\}^\infty$.

F.

uickly compute
$_0$), then
$_0, X_1$), then
$_0, X_1, X_2$), etc.

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–Ch
29 bit ops per mes
using mults in fiel

(I've started invest
bit ops for integer

akes
-X.
useful
'.
pute
en
etc.

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 Auth2
29 bit ops per message bit,
using mults in field of size 2

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

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.)

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-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.)

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

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-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.)

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?

ream: $F_k(N)$.

AC: 128 bits of $F_k(M)$.

NE session:
of $F_k(N) \rightarrow$ tag;
e bits of $F_k(N)$
$m \rightarrow$ ciphertext C_1;
of $F_k(N, A_1, C_1) \rightarrow$ tag;

NSE: misuse resistance.

BC: wide-block cipher.

ed, the wide-block cipher
s Xoofff and Xoofffie,
) 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.)

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 oper
(assumin

key	ops
256	54
256	78
128	88
128	100
128	117
256	126
256	144
128	147
256	156
128	162
128	202
256	283

(N).

its of $F_k(M)$.

n:

$\rightarrow$ tag;

$_k(N)$

ertext C_1;

$A_1, C_1) \rightarrow$ tag;

use resistance.

block cipher.

le-block cipher

nd Xoofffie,

n 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.)

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
(assuming precom

key	ops/bit	ciph
256	54	Cha
256	78	Cha
128	88	Simo
128	100	NOE
128	117	Skin
256	126	Cha
256	144	Simo
128	147.2	PRE
256	156	Skin
128	162.75	Picc
128	202.5	AES
256	283.5	AES

M).

tag;

nce.

er.

pher

e,

.

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 pla
(assuming precomputed sub

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

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

eed

rnstein–Chou Auth256:
ps per message bit,
ults in field of size 2^{256}.

rted investigating
for integer mults.)

on sounds slower, but
PRF or PRP or SPRP.
ny rounds are needed
ontext of a MAC?

c. try to skip MAC,
these modes safely use
ounds 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 vi
- Easy
- Binary
  code-b
- Intege
  lattice
- Use ex

nou Auth256:
ssage bit,
d of size 2^{256}.

tigating
mults.)

s slower, but
RP or SPRP.
are needed
a MAC?

kip MAC,
des safely use
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 m

- Easy masking.
- Binary mults: S
  code-based cryp
- Integer mults: S
  lattice-based cry
- Use existing CP

56:

^{256}.

ut

RP.

ed

use

ode?

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

- Easy masking.
- Binary mults: Share area
  code-based crypto.
- Integer mults: Share area
  lattice-based crypto and E
- Use existing CPU multipli

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.

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.

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

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.

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.