Small cryptographic bytecode

D. J. Bernstein

elaborating on an idea from
Adam Langley

"Line search":
trying to find minimum of
function f defined on x-line.

e.g. "Bisection", trying to find
minimum in interval $[x_0, x_1]$:
Replace interval with either
$[x_0, (x_0+x_1)/2]$ or $[(x_0+x_1)/2, x_1]$;
try to make sensible choice.
Iterate many times.

Small cryptographic bytecode

D. J. Bernstein

elaborating on an idea from
Adam Langley

"Line search":
trying to find minimum of
function f defined on x-line.

e.g. "Bisection", trying to find
minimum in interval $[x_0, x_1]$:
Replace interval with either
$[x_0, (x_0+x_1)/2]$ or $[(x_0+x_1)/2, x_1]$;
try to make sensible choice.
Iterate many times.

Can try to reduce #iterations
using smarter models of f:
see, e.g., "secant method".

Small cryptographic bytecode

D. J. Bernstein

elaborating on an idea from
Adam Langley

"Line search":
trying to find minimum of
function f defined on x-line.

e.g. "Bisection", trying to find minimum in interval $[x_0, x_1]$:
Replace interval with either $[x_0, (x_0+x_1)/2]$ or $[(x_0+x_1)/2, x_1]$;
try to make sensible choice.
Iterate many times.

Can try to reduce #iterations using smarter models of f:
see, e.g., "secant method".

Harder when f varies more.

yptographic bytecode

ernstein

ing on an idea from
angley

“Line search”:
trying to find minimum of
function f defined on x-line.

e.g. “Bisection”, trying to find
minimum in interval $[x_0, x_1]$:
Replace interval with either
$[x_0, (x_0+x_1)/2]$ or $[(x_0+x_1)/2, x_1]$;
try to make sensible choice.
Iterate many times.

Can try to reduce #iterations
using smarter models of f:
see, e.g., “secant method”.

Harder when f varies more.

How to
f define

“Gradier
Starting
try to fi
where f

ic bytecode

idea from

“Line search”:
trying to find minimum of
function f defined on x-line.

e.g. “Bisection”, trying to find
minimum in interval $[x_0, x_1]$:
Replace interval with either
$[x_0, (x_0+x_1)/2]$ or $[(x_0+x_1)/2, x_1]$;
try to make sensible choice.
Iterate many times.

Can try to reduce #iterations
using smarter models of f:
see, e.g., “secant method”.

Harder when f varies more.

How to find minim
f defined on $(x, y$

“Gradient descent
Starting from $(x_0,$
try to figure out d
where f decreases

le

"Line search":
trying to find minimum of
function f defined on x-line.

e.g. "Bisection", trying to find
minimum in interval $[x_0, x_1]$:
Replace interval with either
$[x_0, (x_0+x_1)/2]$ or $[(x_0+x_1)/2, x_1]$;
try to make sensible choice.
Iterate many times.

Can try to reduce #iterations
using smarter models of f:
see, e.g., "secant method".

Harder when f varies more.

How to find minimum of fu
f defined on (x, y)-plane?

"Gradient descent":
Starting from (x_0, y_0),
try to figure out direction
where f decreases fastest.

"Line search":
trying to find minimum of
function f defined on x-line.

e.g. "Bisection", trying to find
minimum in interval $[x_0, x_1]$:
Replace interval with either
$[x_0, (x_0+x_1)/2]$ or $[(x_0+x_1)/2, x_1]$;
try to make sensible choice.
Iterate many times.

Can try to reduce #iterations
using smarter models of f:
see, e.g., "secant method".

Harder when f varies more.

How to find minimum of function
f defined on (x, y)-plane?

"Gradient descent":
Starting from (x_0, y_0),
try to figure out direction
where f decreases fastest.

"Line search":
trying to find minimum of
function f defined on x-line.

e.g. "Bisection", trying to find
minimum in interval $[x_0, x_1]$:
Replace interval with either
$[x_0, (x_0+x_1)/2]$ or $[(x_0+x_1)/2, x_1]$;
try to make sensible choice.
Iterate many times.

Can try to reduce #iterations
using smarter models of f:
see, e.g., "secant method".

Harder when f varies more.

How to find minimum of function
f defined on (x, y)-plane?

"Gradient descent":
Starting from (x_0, y_0),
try to figure out direction
where f decreases fastest.

Could do line search to find
minimum in that direction.
Then find a new direction.

"Line search":
trying to find minimum of
function f defined on x-line.

e.g. "Bisection", trying to find
minimum in interval $[x_0, x_1]$:
Replace interval with either
$[x_0, (x_0+x_1)/2]$ or $[(x_0+x_1)/2, x_1]$;
try to make sensible choice.
Iterate many times.

Can try to reduce #iterations
using smarter models of f:
see, e.g., "secant method".

Harder when f varies more.

How to find minimum of function
f defined on (x, y)-plane?

"Gradient descent":
Starting from (x_0, y_0),
try to figure out direction
where f decreases fastest.

Could do line search to find
minimum in that direction.
Then find a new direction.

Better: Step down that direction.
Then find a new direction.

"Line search":
trying to find minimum of
function f defined on x-line.

e.g. "Bisection", trying to find
minimum in interval $[x_0, x_1]$:
Replace interval with either
$[x_0, (x_0+x_1)/2]$ or $[(x_0+x_1)/2, x_1]$;
try to make sensible choice.
Iterate many times.

Can try to reduce #iterations
using smarter models of f:
see, e.g., "secant method".

Harder when f varies more.

How to find minimum of function
f defined on (x, y)-plane?

"Gradient descent":
Starting from (x_0, y_0),
try to figure out direction
where f decreases fastest.

Could do line search to find
minimum in that direction.
Then find a new direction.

Better: Step down that direction.
Then find a new direction.

Silly: Line search in x direction;
line search in y direction; repeat.

arch”:
o find minimum of
f defined on x-line.

section”, trying to find
n in interval $[x_0, x_1]$:
interval with either
$-x_1)/2]$ or $[(x_0+x_1)/2, x_1]$;
ake sensible choice.
nany times.

to reduce #iterations
narter models of f:
, “secant method”.

vhen f varies more.

How to find minimum of function f defined on (x, y)-plane?

“Gradient descent”:
Starting from (x_0, y_0),
try to figure out direction
where f decreases fastest.

Could do line search to find
minimum in that direction.
Then find a new direction.

Better: Step down that direction.
Then find a new direction.

Silly: Line search in x direction;
line search in y direction; repeat.

Keccak

Goal: Fa
on a Co

You star
impleme

mum of
on x-line.

rying to find
al $[x_0, x_1]$:
ith either
$[(x_0+x_1)/2, x_1]$;
le choice.
s.

#iterations
dels of f:
method”.

ries more.

How to find minimum of function
f defined on (x, y)-plane?

“Gradient descent”:
Starting from (x_0, y_0),
try to figure out direction
where f decreases fastest.

Could do line search to find
minimum in that direction.
Then find a new direction.

Better: Step down that direction.
Then find a new direction.

Silly: Line search in x direction;
line search in y direction; repeat.

Keccak optimizati

Goal: Fastest C co
on a Cortex-M4 C

You start with sim
implementing Kec

nd

$/2, x_1];$

ns

How to find minimum of function f defined on (x, y)-plane?

"Gradient descent": Starting from (x_0, y_0), try to figure out direction where f decreases fastest.

Could do line search to find minimum in that direction. Then find a new direction.

Better: Step down that direction. Then find a new direction.

Silly: Line search in x direction; line search in y direction; repeat.

Keccak optimization

Goal: Fastest C code for Ke on a Cortex-M4 CPU core.

You start with simple C cod implementing Keccak.

How to find minimum of function f defined on (x, y)-plane?

"Gradient descent": Starting from (x_0, y_0), try to figure out direction where f decreases fastest.

Could do line search to find minimum in that direction. Then find a new direction.

Better: Step down that direction. Then find a new direction.

Silly: Line search in x direction; line search in y direction; repeat.

Keccak optimization

Goal: Fastest C code for Keccak on a Cortex-M4 CPU core.

You start with simple C code implementing Keccak.

How to find minimum of function f defined on (x, y)-plane?

"Gradient descent":
Starting from (x_0, y_0),
try to figure out direction
where f decreases fastest.

Could do line search to find minimum in that direction.
Then find a new direction.

Better: Step down that direction.
Then find a new direction.

Silly: Line search in x direction;
line search in y direction; repeat.

Keccak optimization

Goal: Fastest C code for Keccak on a Cortex-M4 CPU core.

You start with simple C code implementing Keccak.

You compile it; see how fast it is;
modify it to try to make it faster;
repeat; eventually stop trying.

How to find minimum of function f defined on (x, y)-plane?

"Gradient descent": Starting from (x_0, y_0), try to figure out direction where f decreases fastest.

Could do line search to find minimum in that direction. Then find a new direction.

Better: Step down that direction. Then find a new direction.

Silly: Line search in x direction; line search in y direction; repeat.

Keccak optimization

Goal: Fastest C code for Keccak on a Cortex-M4 CPU core.

You start with simple C code implementing Keccak.

You compile it; see how fast it is; modify it to try to make it faster; repeat; eventually stop trying.

You publish your fastest code.

Maybe lots of people use it, and care about its speed.

find minimum of function
d on (x, y)-plane?

nt descent":
from (x_0, y_0),
gure out direction
decreases fastest.

o line search to find
n in that direction.
nd a new direction.

Step down that direction.
nd a new direction.

ne search in x direction;
ch in y direction; repeat.

Keccak optimization

Goal: Fastest C code for Keccak on a Cortex-M4 CPU core.

You start with simple C code implementing Keccak.

You compile it; see how fast it is; modify it to try to make it faster; repeat; eventually stop trying.

You publish your fastest code.

Maybe lots of people use it, and care about its speed.

Compile
your Kec

mum of function
)-plane?

":

y_0),
irection
fastest.

ch to find
direction.
direction.

that direction.
direction.

in x direction;
rection; repeat.

Keccak optimization

Goal: Fastest C code for Keccak on a Cortex-M4 CPU core.

You start with simple C code implementing Keccak.

You compile it; see how fast it is; modify it to try to make it faster; repeat; eventually stop trying.

You publish your fastest code.

Maybe lots of people use it, and care about its speed.

Compiler writer le
your Keccak Corte

nction

ction.

ion;
peat.

Keccak optimization

Goal: Fastest C code for Keccak on a Cortex-M4 CPU core.

You start with simple C code implementing Keccak.

You compile it; see how fast it is; modify it to try to make it faster; repeat; eventually stop trying.

You publish your fastest code.

Maybe lots of people use it, and care about its speed.

Compiler writer learns about your Keccak Cortex-M4 C c

Keccak optimization

Goal: Fastest C code for Keccak on a Cortex-M4 CPU core.

You start with simple C code implementing Keccak.

You compile it; see how fast it is; modify it to try to make it faster; repeat; eventually stop trying.

You publish your fastest code.

Maybe lots of people use it, and care about its speed.

Compiler writer learns about your Keccak Cortex-M4 C code.

Keccak optimization

Goal: Fastest C code for Keccak on a Cortex-M4 CPU core.

You start with simple C code implementing Keccak.

You compile it; see how fast it is; modify it to try to make it faster; repeat; eventually stop trying.

You publish your fastest code.

Maybe lots of people use it, and care about its speed.

Compiler writer learns about your Keccak Cortex-M4 C code.

Compiles it; sees how fast it is. Modifies *compiler* to try to make the compiled code faster. Repeats; eventually stops trying.

Keccak optimization

Goal: Fastest C code for Keccak on a Cortex-M4 CPU core.

You start with simple C code implementing Keccak.

You compile it; see how fast it is; modify it to try to make it faster; repeat; eventually stop trying.

You publish your fastest code.

Maybe lots of people use it, and care about its speed.

Compiler writer learns about your Keccak Cortex-M4 C code.

Compiles it; sees how fast it is. Modifies *compiler* to try to make the compiled code faster. Repeats; eventually stops trying.

Publishes a new compiler version.

Keccak optimization

Goal: Fastest C code for Keccak on a Cortex-M4 CPU core.

You start with simple C code implementing Keccak.

You compile it; see how fast it is; modify it to try to make it faster; repeat; eventually stop trying.

You publish your fastest code.

Maybe lots of people use it, and care about its speed.

Compiler writer learns about your Keccak Cortex-M4 C code.

Compiles it; sees how fast it is. Modifies *compiler* to try to make the compiled code faster. Repeats; eventually stops trying.

Publishes a new compiler version.

Later: Maybe you try the new compiler. Whole process repeats.

Keccak optimization

Goal: Fastest C code for Keccak on a Cortex-M4 CPU core.

You start with simple C code implementing Keccak.

You compile it; see how fast it is; modify it to try to make it faster; repeat; eventually stop trying.

You publish your fastest code.

Maybe lots of people use it, and care about its speed.

Compiler writer learns about your Keccak Cortex-M4 C code.

Compiles it; sees how fast it is. Modifies *compiler* to try to make the compiled code faster. Repeats; eventually stops trying.

Publishes a new compiler version.

Later: Maybe you try the new compiler. Whole process repeats.

You treat compiler as constant. Compiler treats code as constant.

optimization

astest C code for Keccak
rtex-M4 CPU core.

t with simple C code
nting Keccak.

npile it; see how fast it is;
t to try to make it faster;
eventually stop trying.

lish your fastest code.

ots of people use it,
about its speed.

Compiler writer learns about your Keccak Cortex-M4 C code.

Compiles it; sees how fast it is. Modifies *compiler* to try to make the compiled code faster. Repeats; eventually stops trying.

Publishes a new compiler version.

Later: Maybe you try the new compiler. Whole process repeats.

You treat compiler as constant. Compiler treats code as constant.

Define *f*
code *x* w

on

ode for Keccak
PU core.

nple C code
cak.

e how fast it is;
make it faster;
stop trying.

astest code.

ple use it,
speed.

Compiler writer learns about
your Keccak Cortex-M4 C code.

Compiles it; sees how fast it is.
Modifies *compiler* to try to
make the compiled code faster.
Repeats; eventually stops trying.

Publishes a new compiler version.

Later: Maybe you try the new
compiler. Whole process repeats.

You treat compiler as constant.
Compiler treats code as constant.

Define $f(x, y)$ as t
code x with comp

ccak

e

it is;
aster;
g.

le.

Compiler writer learns about your Keccak Cortex-M4 C code.

Compiles it; sees how fast it is.
Modifies *compiler* to try to make the compiled code faster.
Repeats; eventually stops trying.

Publishes a new compiler version.

Later: Maybe you try the new compiler. Whole process repeats.

You treat compiler as constant.
Compiler treats code as constant.

Define $f(x, y)$ as time taken
code x with compiler y.

Compiler writer learns about your Keccak Cortex-M4 C code.

Compiles it; sees how fast it is. Modifies *compiler* to try to make the compiled code faster. Repeats; eventually stops trying.

Publishes a new compiler version.

Later: Maybe you try the new compiler. Whole process repeats.

You treat compiler as constant. Compiler treats code as constant.

Define $f(x, y)$ as time taken by code x with compiler y.

Compiler writer learns about your Keccak Cortex-M4 C code.

Compiles it; sees how fast it is. Modifies *compiler* to try to make the compiled code faster. Repeats; eventually stops trying.

Publishes a new compiler version.

Later: Maybe you try the new compiler. Whole process repeats.

You treat compiler as constant. Compiler treats code as constant.

Define $f(x, y)$ as time taken by code x with compiler y.

x_0: initial code.

y_0: initial compiler.

Compiler writer learns about your Keccak Cortex-M4 C code.

Compiles it; sees how fast it is. Modifies *compiler* to try to make the compiled code faster. Repeats; eventually stops trying.

Publishes a new compiler version.

Later: Maybe you try the new compiler. Whole process repeats.

You treat compiler as constant. Compiler treats code as constant.

Define $f(x, y)$ as time taken by code x with compiler y.

x_0: initial code.

y_0: initial compiler.

You try to minimize $f(x, y_0)$.
x_1: new code from this line search in x direction.

Compiler writer learns about your Keccak Cortex-M4 C code.

Compiles it; sees how fast it is. Modifies *compiler* to try to make the compiled code faster. Repeats; eventually stops trying.

Publishes a new compiler version.

Later: Maybe you try the new compiler. Whole process repeats.

You treat compiler as constant. Compiler treats code as constant.

Define $f(x, y)$ as time taken by code x with compiler y.

x_0: initial code.

y_0: initial compiler.

You try to minimize $f(x, y_0)$.
x_1: new code from this line search in x direction.

Compiler writer: $f(x_1, y)$.
y_1: new compiler from this line search in y direction.

Compiler writer learns about your Keccak Cortex-M4 C code.

Compiles it; sees how fast it is. Modifies *compiler* to try to make the compiled code faster. Repeats; eventually stops trying.

Publishes a new compiler version.

Later: Maybe you try the new compiler. Whole process repeats.

You treat compiler as constant. Compiler treats code as constant.

Define $f(x, y)$ as time taken by code x with compiler y.

x_0: initial code.

y_0: initial compiler.

You try to minimize $f(x, y_0)$. x_1: new code from this line search in x direction.

Compiler writer: $f(x_1, y)$. y_1: new compiler from this line search in y direction.

This whole approach is silly.

r writer learns about
ccak Cortex-M4 C code.

s it; sees how fast it is.
s *compiler* to try to
e compiled code faster.
; eventually stops trying.

es a new compiler version.

Maybe you try the new
r. Whole process repeats.

nt compiler as constant.
r treats code as constant.

Define $f(x, y)$ as time taken by
code x with compiler y.

x_0: initial code.

y_0: initial compiler.

You try to minimize $f(x, y_0)$.
x_1: new code from this
line search in x direction.

Compiler writer: $f(x_1, y)$.
y_1: new compiler from this
line search in y direction.

This whole approach is silly.

min$\{f(x$
fastest K

arns about
ex-M4 C code.

how fast it is.
to try to
d code faster.
y stops trying.

ompiler version.

try the new
process repeats.

r as constant.
ode as constant.

Define $f(x, y)$ as time taken by code x with compiler y.

x_0: initial code.

y_0: initial compiler.

You try to minimize $f(x, y_0)$.
x_1: new code from this line search in x direction.

Compiler writer: $f(x_1, y)$.
y_1: new compiler from this line search in y direction.

This whole approach is silly.

$\min\{f(x, y)\}$ is th
fastest Keccak Co

t
ode.
is.
ter.
ying.
rsion.
ew
eats.
ant.
stant.

Define $f(x, y)$ as time taken by code x with compiler y.

x_0: initial code.

y_0: initial compiler.

You try to minimize $f(x, y_0)$.
x_1: new code from this line search in x direction.

Compiler writer: $f(x_1, y)$.
y_1: new compiler from this line search in y direction.

This whole approach is silly.

$\min\{f(x, y)\}$ is the time tak
fastest Keccak Cortex-M4 as

Define $f(x, y)$ as time taken by code x with compiler y.

x_0: initial code.

y_0: initial compiler.

You try to minimize $f(x, y_0)$.
x_1: new code from this line search in x direction.

Compiler writer: $f(x_1, y)$.
y_1: new compiler from this line search in y direction.

This whole approach is silly.

$\min\{f(x, y)\}$ is the time taken by fastest Keccak Cortex-M4 asm.

Define $f(x, y)$ as time taken by code x with compiler y.

x_0: initial code.

y_0: initial compiler.

You try to minimize $f(x, y_0)$.
x_1: new code from this line search in x direction.

Compiler writer: $f(x_1, y)$.
y_1: new compiler from this line search in y direction.

This whole approach is silly.

$\min\{f(x, y)\}$ is the time taken by fastest Keccak Cortex-M4 asm.

Slowly bouncing between x-line searches, y-line searches is a silly way to approach this min.

Define $f(x, y)$ as time taken by code x with compiler y.

x_0: initial code.

y_0: initial compiler.

You try to minimize $f(x, y_0)$.
x_1: new code from this line search in x direction.

Compiler writer: $f(x_1, y)$.
y_1: new compiler from this line search in y direction.

This whole approach is silly.

$\min\{f(x, y)\}$ is the time taken by fastest Keccak Cortex-M4 asm.

Slowly bouncing between x-line searches, y-line searches is a silly way to approach this min.

Clearly min can be achieved by many different pairs (x, y).
Which pair is easiest to find?

Define $f(x, y)$ as time taken by code x with compiler y.

x_0: initial code.

y_0: initial compiler.

You try to minimize $f(x, y_0)$.
x_1: new code from this line search in x direction.

Compiler writer: $f(x_1, y)$.
y_1: new compiler from this line search in y direction.

This whole approach is silly.

$\min\{f(x, y)\}$ is the time taken by fastest Keccak Cortex-M4 asm.

Slowly bouncing between x-line searches, y-line searches is a silly way to approach this min.

Clearly min can be achieved by many different pairs (x, y).
Which pair is easiest to find?

Generalize from C to other languages: which language makes min easiest to find?
Why did goal say "C code"?
End user doesn't need C.

(x, y) as time taken by
with compiler y.

al code.

al compiler.

to minimize $f(x, y_0)$.
code from this
ch in x direction.

r writer: $f(x_1, y)$.
compiler from this
ch in y direction.

ole approach is silly.

$\min\{f(x, y)\}$ is the time taken by fastest Keccak Cortex-M4 asm.

Slowly bouncing between x-line searches, y-line searches is a silly way to approach this min.

Clearly min can be achieved by many different pairs (x, y). Which pair is easiest to find?

Generalize from C to other languages: which language makes min easiest to find? Why did goal say “C code”? End user doesn’t need C.

Does en

time taken by
iler y.

r.

ze $f(x, y_0)$.
n this
rection.

$f(x_1, y)$.
from this
rection.

ch is silly.

$\min\{f(x, y)\}$ is the time taken by fastest Keccak Cortex-M4 asm.

Slowly bouncing between x-line searches, y-line searches is a silly way to approach this min.

Clearly min can be achieved by many different pairs (x, y). Which pair is easiest to find?

Generalize from C to other languages: which language makes min easiest to find? Why did goal say “C code”? End user doesn’t need C.

Does end user nee

by

).

$\min\{f(x, y)\}$ is the time taken by fastest Keccak Cortex-M4 asm.

Slowly bouncing between x-line searches, y-line searches is a silly way to approach this min.

Clearly min can be achieved by many different pairs (x, y). Which pair is easiest to find?

Generalize from C to other languages: which language makes min easiest to find? Why did goal say "C code"? End user doesn't need C.

Does end user need Cortex-M

$\min\{f(x, y)\}$ is the time taken by fastest Keccak Cortex-M4 asm.

Slowly bouncing between x-line searches, y-line searches is a silly way to approach this min.

Clearly min can be achieved by many different pairs (x, y). Which pair is easiest to find?

Generalize from C to other languages: which language makes min easiest to find? Why did goal say "C code"? End user doesn't need C.

Does end user need Cortex-M4?

$\min\{f(x, y)\}$ is the time taken by fastest Keccak Cortex-M4 asm.

Slowly bouncing between x-line searches, y-line searches is a silly way to approach this min.

Clearly min can be achieved by many different pairs (x, y). Which pair is easiest to find?

Generalize from C to other languages: which language makes min easiest to find? Why did goal say "C code"? End user doesn't need C.

Does end user need Cortex-M4?

CPU designer learns about your Keccak Cortex-M4 asm.

$\min\{f(x, y)\}$ is the time taken by fastest Keccak Cortex-M4 asm.

Slowly bouncing between x-line searches, y-line searches is a silly way to approach this min.

Clearly min can be achieved by many different pairs (x, y). Which pair is easiest to find?

Generalize from C to other languages: which language makes min easiest to find? Why did goal say "C code"? End user doesn't need C.

Does end user need Cortex-M4?

CPU designer learns about your Keccak Cortex-M4 asm.

Modifies the CPU design to try to make this code faster. Repeats; eventually stops trying.

$\min\{f(x,y)\}$ is the time taken by fastest Keccak Cortex-M4 asm.

Slowly bouncing between x-line searches, y-line searches is a silly way to approach this min.

Clearly min can be achieved by many different pairs (x,y). Which pair is easiest to find?

Generalize from C to other languages: which language makes min easiest to find? Why did goal say "C code"? End user doesn't need C.

Does end user need Cortex-M4?

CPU designer learns about your Keccak Cortex-M4 asm.

Modifies the CPU design to try to make this code faster. Repeats; eventually stops trying.

Years later, sells a new CPU. You reoptimize for this CPU.

$\min\{f(x,y)\}$ is the time taken by fastest Keccak Cortex-M4 asm.

Slowly bouncing between x-line searches, y-line searches is a silly way to approach this min.

Clearly min can be achieved by many different pairs (x, y). Which pair is easiest to find?

Generalize from C to other languages: which language makes min easiest to find? Why did goal say "C code"? End user doesn't need C.

Does end user need Cortex-M4?

CPU designer learns about your Keccak Cortex-M4 asm.

Modifies the CPU design to try to make this code faster. Repeats; eventually stops trying.

Years later, sells a new CPU. You reoptimize for this CPU.

Sometimes CPUs try extending or replacing instruction set, but this is poorly coordinated with programmers, compiler writers.

$, y)\}$ is the time taken by
Keccak Cortex-M4 asm.

ouncing between
earches, y-line searches is
ay to approach this min.

min can be achieved by
fferent pairs (x, y).
air is easiest to find?

ze from C to other
es: which language
nin easiest to find?
goal say "C code"?
r doesn't need C.

Does end user need Cortex-M4?

CPU designer learns about your Keccak Cortex-M4 asm.

Modifies the CPU design to try to make this code faster. Repeats; eventually stops trying.

Years later, sells a new CPU. You reoptimize for this CPU.

Sometimes CPUs try extending or replacing instruction set, but this is poorly coordinated with programmers, compiler writers.

Generali
$f(x, y)$ i
code x c

If compi
asm $y(x$
$f(x, y) =$

e time taken by
rtex-M4 asm.

etween
line searches is
roach this min.

e achieved by
rs (x, y).
est to find?

to other
language
to find?
"C code"?
need C.

Does end user need Cortex-M4?

CPU designer learns about your Keccak Cortex-M4 asm.

Modifies the CPU design to try to make this code faster. Repeats; eventually stops trying.

Years later, sells a new CPU. You reoptimize for this CPU.

Sometimes CPUs try extending or replacing instruction set, but this is poorly coordinated with programmers, compiler writers.

Generalize $f(x, y)$
$f(x, y)$ is time tak
code x on platforn

If compiler y on c
asm $y(x)$ for Corte
$f(x, y) = f(y(x),$

ken by
sm.

nes is
min.

by

?

?

Does end user need Cortex-M4?

CPU designer learns about your Keccak Cortex-M4 asm.

Modifies the CPU design to try to make this code faster. Repeats; eventually stops trying.

Years later, sells a new CPU. You reoptimize for this CPU.

Sometimes CPUs try extending or replacing instruction set, but this is poorly coordinated with programmers, compiler writers.

Generalize $f(x, y)$ definition

$f(x, y)$ is time taken by code x on platform y.

If compiler y on code x proc

asm $y(x)$ for Cortex-M4:

$f(x, y) = f(y(x), \text{Cortex-M4}$

Does end user need Cortex-M4?

CPU designer learns about your Keccak Cortex-M4 asm.

Modifies the CPU design to try to make this code faster. Repeats; eventually stops trying.

Years later, sells a new CPU. You reoptimize for this CPU.

Sometimes CPUs try extending or replacing instruction set, but this is poorly coordinated with programmers, compiler writers.

Generalize $f(x, y)$ definition: $f(x, y)$ is time taken by code x on platform y.

If compiler y on code x produces asm $y(x)$ for Cortex-M4: $f(x, y) = f(y(x), \text{Cortex-M4})$.

Does end user need Cortex-M4?

CPU designer learns about your Keccak Cortex-M4 asm.

Modifies the CPU design to try to make this code faster. Repeats; eventually stops trying.

Years later, sells a new CPU. You reoptimize for this CPU.

Sometimes CPUs try extending or replacing instruction set, but this is poorly coordinated with programmers, compiler writers.

Generalize $f(x, y)$ definition: $f(x, y)$ is time taken by code x on platform y.

If compiler y on code x produces asm $y(x)$ for Cortex-M4: $f(x, y) = f(y(x), \text{Cortex-M4})$.

Without the CPU changing: Minimize $f(a, \text{Cortex-M4})$. Search for (x, y) with $y(x) = a$.

Does end user need Cortex-M4?

CPU designer learns about your Keccak Cortex-M4 asm.

Modifies the CPU design to try to make this code faster. Repeats; eventually stops trying.

Years later, sells a new CPU. You reoptimize for this CPU.

Sometimes CPUs try extending or replacing instruction set, but this is poorly coordinated with programmers, compiler writers.

Generalize $f(x, y)$ definition: $f(x, y)$ is time taken by code x on platform y.

If compiler y on code x produces asm $y(x)$ for Cortex-M4: $f(x, y) = f(y(x), \text{Cortex-M4})$.

Without the CPU changing: Minimize $f(a, \text{Cortex-M4})$. Search for (x, y) with $y(x) = a$.

Typical CPU designer: View a as a constant; try to minimize $f(a, y)$. Silly optimization approach.

d user need Cortex-M4?

signer learns about your
Cortex-M4 asm.

s the CPU design to
ake this code faster.
; eventually stops trying.

ter, sells a new CPU.
ptimize for this CPU.

nes CPUs try extending
cing instruction set, but
oorly coordinated with
mers, compiler writers.

Generalize $f(x, y)$ definition:
$f(x, y)$ is time taken by
code x on platform y.

If compiler y on code x produces
asm $y(x)$ for Cortex-M4:
$f(x, y) = f(y(x), \text{Cortex-M4})$.

Without the CPU changing:
Minimize $f(a, \text{Cortex-M4})$.
Search for (x, y) with $y(x) = a$.

Typical CPU designer:
View a as a constant;
try to minimize $f(a, y)$.
Silly optimization approach.

“I know
I’ve deve
that con
This circ

ed Cortex-M4?

ns about your
4 asm.

design to
ode faster.
y stops trying.

new CPU.
r this CPU.

try extending
ction set, but
dinated with
npiler writers.

Generalize $f(x, y)$ definition:
$f(x, y)$ is time taken by
code x on platform y.

If compiler y on code x produces
asm $y(x)$ for Cortex-M4:
$f(x, y) = f(y(x), \text{Cortex-M4})$.

Without the CPU changing:
Minimize $f(a, \text{Cortex-M4})$.
Search for (x, y) with $y(x) = a$.

Typical CPU designer:
View a as a constant;
try to minimize $f(a, y)$.
Silly optimization approach.

“I know the minim
I’ve developed the
that computes Ke
This circuit is my

M4?

your

ying.

ling

but

ith

ers.

Generalize $f(x, y)$ definition:
$f(x, y)$ is time taken by
code x on platform y.

If compiler y on code x produces
asm $y(x)$ for Cortex-M4:
$f(x, y) = f(y(x), \text{Cortex-M4})$.

Without the CPU changing:
Minimize $f(a, \text{Cortex-M4})$.
Search for (x, y) with $y(x) = a$.

Typical CPU designer:
View a as a constant;
try to minimize $f(a, y)$.
Silly optimization approach.

"I know the minimum!
I've developed the fastest ci
that computes Keccak.
This circuit is my CPU."

Generalize $f(x, y)$ definition:
$f(x, y)$ is time taken by
code x on platform y.

If compiler y on code x produces
asm $y(x)$ for Cortex-M4:
$f(x, y) = f(y(x), \text{Cortex-M4})$.

Without the CPU changing:
Minimize $f(a, \text{Cortex-M4})$.
Search for (x, y) with $y(x) = a$.

Typical CPU designer:
View a as a constant;
try to minimize $f(a, y)$.
Silly optimization approach.

"I know the minimum!
I've developed the fastest circuit
that computes Keccak.
This circuit is my CPU."

Generalize $f(x, y)$ definition:
$f(x, y)$ is time taken by
code x on platform y.

If compiler y on code x produces
asm $y(x)$ for Cortex-M4:
$f(x, y) = f(y(x), \text{Cortex-M4})$.

Without the CPU changing:
Minimize $f(a, \text{Cortex-M4})$.
Search for (x, y) with $y(x) = a$.

Typical CPU designer:
View a as a constant;
try to minimize $f(a, y)$.
Silly optimization approach.

"I know the minimum!
I've developed the fastest circuit
that computes Keccak.
This circuit is my CPU."

Wait a minute: "CPU" concept
is more restrictive than "chip".

Perspective of CPU designer:
This chip can do anything!

People want this chip to support
SHA-1, SHA-2, SHA-3, SHAmir;
all sorts of block ciphers;
public-key cryptosystems;
non-cryptographic computations.

ze $f(x, y)$ definition:
s time taken by
on platform y.

ler y on code x produces
) for Cortex-M4:
$= f(y(x), \text{Cortex-M4})$.

the CPU changing:
e $f(a, \text{Cortex-M4})$.
or (x, y) with $y(x) = a$.

CPU designer:
as a constant;
inimize $f(a, y)$.
imization approach.

"I know the minimum!
I've developed the fastest circuit
that computes Keccak.
This circuit is my CPU."

Wait a minute: "CPU" concept
is more restrictive than "chip".

Perspective of CPU designer:
This chip can do anything!

People want this chip to support
SHA-1, SHA-2, SHA-3, SHAmir;
all sorts of block ciphers;
public-key cryptosystems;
non-cryptographic computations.

Adding f
("Kecca
adds are

Adding f
for desir
adds eve

definition:
ken by
n y.

ode x produces
ex-M4:
Cortex-M4).

changing:
tex-M4).
with $y(x) = a$.

gner:
ant;
a, y).
approach.

"I know the minimum!
I've developed the fastest circuit that computes Keccak.
This circuit is my CPU."

Wait a minute: "CPU" concept is more restrictive than "chip".

Perspective of CPU designer:
This chip can do anything!

People want this chip to support SHA-1, SHA-2, SHA-3, SHAmir; all sorts of block ciphers; public-key cryptosystems; non-cryptographic computations.

Adding fast Kecca
("Keccak coproces
adds area to CPU.

Adding fast coproc
for desired mix of
adds even more ar

:

duces

4).

$= a$.

"I know the minimum!
I've developed the fastest circuit that computes Keccak.
This circuit is my CPU."

Wait a minute: "CPU" concept is more restrictive than "chip".

Perspective of CPU designer: This chip can do anything!

People want this chip to support SHA-1, SHA-2, SHA-3, SHAmir; all sorts of block ciphers; public-key cryptosystems; non-cryptographic computations.

Adding fast Keccak circuit ("Keccak coprocessor") to C adds area to CPU.

Adding fast coprocessors for desired mix of operations adds even more area to CPU

"I know the minimum!
I've developed the fastest circuit
that computes Keccak.
This circuit is my CPU."

Wait a minute: "CPU" concept
is more restrictive than "chip".

Perspective of CPU designer:
This chip can do anything!

People want this chip to support
SHA-1, SHA-2, SHA-3, SHAmir;
all sorts of block ciphers;
public-key cryptosystems;
non-cryptographic computations.

Adding fast Keccak circuit
("Keccak coprocessor") to CPU
adds area to CPU.

Adding fast coprocessors
for desired mix of operations
adds even more area to CPU.

"I know the minimum!
I've developed the fastest circuit
that computes Keccak.
This circuit is my CPU."

Wait a minute: "CPU" concept
is more restrictive than "chip".

Perspective of CPU designer:
This chip can do anything!

People want this chip to support
SHA-1, SHA-2, SHA-3, SHAmir;
all sorts of block ciphers;
public-key cryptosystems;
non-cryptographic computations.

Adding fast Keccak circuit
("Keccak coprocessor") to CPU
adds area to CPU.

Adding fast coprocessors
for desired mix of operations
adds even more area to CPU.

For same CPU area,
obtain much better throughput
by building many copies
of original CPU core
without these coprocessors.

"I know the minimum!
I've developed the fastest circuit
that computes Keccak.
This circuit is my CPU."

Wait a minute: "CPU" concept
is more restrictive than "chip".

Perspective of CPU designer:
This chip can do anything!

People want this chip to support
SHA-1, SHA-2, SHA-3, SHAmir;
all sorts of block ciphers;
public-key cryptosystems;
non-cryptographic computations.

Adding fast Keccak circuit
("Keccak coprocessor") to CPU
adds area to CPU.

Adding fast coprocessors
for desired mix of operations
adds even more area to CPU.

For same CPU area,
obtain much better throughput
by building many copies
of original CPU core
without these coprocessors.

Fast Keccak chip is special case.
Doesn't reflect general case.

the minimum!
eloped the fastest circuit
nputes Keccak.
cuit is my CPU."

minute: "CPU" concept
restrictive than "chip".

tive of CPU designer:
p can do anything!

vant this chip to support
SHA-2, SHA-3, SHAmir;
of block ciphers;
ey cryptosystems;
otographic computations.

Adding fast Keccak circuit
("Keccak coprocessor") to CPU
adds area to CPU.

Adding fast coprocessors
for desired mix of operations
adds even more area to CPU.

For same CPU area,
obtain much better throughput
by building many copies
of original CPU core
without these coprocessors.

Fast Keccak chip is special case.
Doesn't reflect general case.

CPU des
What is
for a spe
within a

num!
fastest circuit
ccak.
CPU."

CPU" concept
than "chip".

U designer:
anything!

chip to support
HA-3, SHAmir;
ciphers;
ystems;
computations.

Adding fast Keccak circuit ("Keccak coprocessor") to CPU adds area to CPU.

Adding fast coprocessors for desired mix of operations adds even more area to CPU.

For same CPU area, obtain much better throughput by building many copies of original CPU core without these coprocessors.

Fast Keccak chip is special case. Doesn't reflect general case.

CPU designer's m
What is best perfo
for a specified mix
within a particular

rcuit

cept
p".

r:

pport
Amir;

tions.

Adding fast Keccak circuit ("Keccak coprocessor") to CPU adds area to CPU.

Adding fast coprocessors for desired mix of operations adds even more area to CPU.

For same CPU area, obtain much better throughput by building many copies of original CPU core without these coprocessors.

Fast Keccak chip is special case. Doesn't reflect general case.

CPU designer's metric: What is best performance for a specified mix of operat within a particular CPU area

Adding fast Keccak circuit
("Keccak coprocessor") to CPU
adds area to CPU.

Adding fast coprocessors
for desired mix of operations
adds even more area to CPU.

For same CPU area,
obtain much better throughput
by building many copies
of original CPU core
without these coprocessors.

Fast Keccak chip is special case.
Doesn't reflect general case.

CPU designer's metric:
What is best performance
for a specified mix of operations
within a particular CPU area?

Adding fast Keccak circuit ("Keccak coprocessor") to CPU adds area to CPU.

Adding fast coprocessors for desired mix of operations adds even more area to CPU.

For same CPU area, obtain much better throughput by building many copies of original CPU core without these coprocessors.

Fast Keccak chip is special case. Doesn't reflect general case.

CPU designer's metric: What is best performance for a specified mix of operations within a particular CPU area?

CPU designer is much more likely to consider incorporating a **small** Keccak coprocessor.

Adding fast Keccak circuit
("Keccak coprocessor") to CPU
adds area to CPU.

Adding fast coprocessors
for desired mix of operations
adds even more area to CPU.

For same CPU area,
obtain much better throughput
by building many copies
of original CPU core
without these coprocessors.

Fast Keccak chip is special case.
Doesn't reflect general case.

CPU designer's metric:
What is best performance
for a specified mix of operations
within a particular CPU area?

CPU designer is much more likely
to consider incorporating a
small Keccak coprocessor.

"So we should design the
smallest Keccak circuit?"

Adding fast Keccak circuit
("Keccak coprocessor") to CPU
adds area to CPU.

Adding fast coprocessors
for desired mix of operations
adds even more area to CPU.

For same CPU area,
obtain much better throughput
by building many copies
of original CPU core
without these coprocessors.

Fast Keccak chip is special case.
Doesn't reflect general case.

CPU designer's metric:
What is best performance
for a specified mix of operations
within a particular CPU area?

CPU designer is much more likely
to consider incorporating a
small Keccak coprocessor.

"So we should design the
smallest Keccak circuit?"

—Maybe, but will this extreme
be faster than using existing CPU
instructions without coprocessor?

fast Keccak circuit
k coprocessor") to CPU
a to CPU.

fast coprocessors
ed mix of operations
en more area to CPU.

e CPU area,
nuch better throughput
ing many copies
al CPU core
these coprocessors.

ccak chip is special case.
reflect general case.

CPU designer's metric:
What is best performance
for a specified mix of operations
within a particular CPU area?

CPU designer is much more likely
to consider incorporating a
small Keccak coprocessor.

"So we should design the
smallest Keccak circuit?"

—Maybe, but will this extreme
be faster than using existing CPU
instructions without coprocessor?

Intel typ
quite lar
32KB L1
32KB L1
several f
many di
out-of-o

"So it's
to add i
for my f

k circuit
ssor") to CPU
.

cessors
operations
ea to CPU.

a,
r throughput
copies
re
rocessors.

is special case.
neral case.

CPU designer's metric:
What is best performance
for a specified mix of operations
within a particular CPU area?

CPU designer is much more likely
to consider incorporating a
small Keccak coprocessor.

"So we should design the
smallest Keccak circuit?"

—Maybe, but will this extreme
be faster than using existing CPU
instructions without coprocessor?

Intel typically desi
quite large CPU c
32KB L1 data cac
32KB L1 instructi
several fast multip
many different ins
out-of-order unit,

"So it's small cost
to add instruction
for my favorite cry

CPU

s
U.

put

case.

CPU designer's metric:
What is best performance
for a specified mix of operations
within a particular CPU area?

CPU designer is much more likely
to consider incorporating a
small Keccak coprocessor.

"So we should design the
smallest Keccak circuit?"

—Maybe, but will this extreme
be faster than using existing CPU
instructions without coprocessor?

Intel typically designs
quite large CPU cores:
32KB L1 data cache,
32KB L1 instruction cache,
several fast multipliers,
many different instructions,
out-of-order unit, etc.

"So it's small cost for Intel
to add instruction-set extens
for my favorite crypto!"

CPU designer's metric:
What is best performance
for a specified mix of operations
within a particular CPU area?

CPU designer is much more likely
to consider incorporating a
small Keccak coprocessor.

"So we should design the
smallest Keccak circuit?"

—Maybe, but will this extreme
be faster than using existing CPU
instructions without coprocessor?

Intel typically designs
quite large CPU cores:
32KB L1 data cache,
32KB L1 instruction cache,
several fast multipliers,
many different instructions,
out-of-order unit, etc.

"So it's small cost for Intel
to add instruction-set extension
for my favorite crypto!"

CPU designer's metric:
What is best performance
for a specified mix of operations
within a particular CPU area?

CPU designer is much more likely
to consider incorporating a
small Keccak coprocessor.

"So we should design the
smallest Keccak circuit?"

—Maybe, but will this extreme
be faster than using existing CPU
instructions without coprocessor?

Intel typically designs
quite large CPU cores:
32KB L1 data cache,
32KB L1 instruction cache,
several fast multipliers,
many different instructions,
out-of-order unit, etc.

"So it's small cost for Intel
to add instruction-set extension
for my favorite crypto!"

—Yes, but even smaller benefit
for Intel's mix of operations.

signer's metric:
best performance
ecified mix of operations
particular CPU area?

signer is much more likely
der incorporating a
eccak coprocessor.

should design the
Keccak circuit?"

e, but will this extreme
r than using existing CPU
ons without coprocessor?

Intel typically designs
quite large CPU cores:
32KB L1 data cache,
32KB L1 instruction cache,
several fast multipliers,
many different instructions,
out-of-order unit, etc.

"So it's small cost for Intel
to add instruction-set extension
for my favorite crypto!"

—Yes, but even smaller benefit
for Intel's mix of operations.

Intel did
for 1 rou

How ma
in an AE

Can be
8: small
4: even
. . . 1: p
compare
and usin

etric:
ormance
of operations
CPU area?

uch more likely
orating a
rocessor.

sign the
ircuit?"

this extreme
ng existing CPU
ut coprocessor?

Intel typically designs
quite large CPU cores:
32KB L1 data cache,
32KB L1 instruction cache,
several fast multipliers,
many different instructions,
out-of-order unit, etc.

"So it's small cost for Intel
to add instruction-set extension
for my favorite crypto!"

—Yes, but even smaller benefit
for Intel's mix of operations.

Intel did add instr
for 1 round of AES

How many paralle
in an AES-round c

Can be 16: big; fa
8: smaller but slov
4: even smaller bu
. . . 1: probably no
compared to skipp
and using other C

tions
a?
likely

me
g CPU
essor?

Intel typically designs
quite large CPU cores:
32KB L1 data cache,
32KB L1 instruction cache,
several fast multipliers,
many different instructions,
out-of-order unit, etc.

“So it’s small cost for Intel
to add instruction-set extension
for my favorite crypto!”

—Yes, but even smaller benefit
for Intel’s mix of operations.

Intel did add instruction
for 1 round of AES.

How many parallel S-boxes a
in an AES-round coprocesso

Can be 16: big; fast.
8: smaller but slower.
4: even smaller but slower.
. . . 1: probably not worthwh
compared to skipping coproc
and using other CPU instruc

Intel typically designs
quite large CPU cores:
32KB L1 data cache,
32KB L1 instruction cache,
several fast multipliers,
many different instructions,
out-of-order unit, etc.

"So it's small cost for Intel
to add instruction-set extension
for my favorite crypto!"

—Yes, but even smaller benefit
for Intel's mix of operations.

Intel did add instruction
for 1 round of AES.

How many parallel S-boxes are
in an AES-round coprocessor?

Can be 16: big; fast.
8: smaller but slower.
4: even smaller but slower.
. . . 1: probably not worthwhile
compared to skipping coprocessor
and using other CPU instructions.

Intel typically designs
quite large CPU cores:
32KB L1 data cache,
32KB L1 instruction cache,
several fast multipliers,
many different instructions,
out-of-order unit, etc.

"So it's small cost for Intel
to add instruction-set extension
for my favorite crypto!"

—Yes, but even smaller benefit
for Intel's mix of operations.

Intel did add instruction
for 1 round of AES.

How many parallel S-boxes are
in an AES-round coprocessor?

Can be 16: big; fast.
8: smaller but slower.
4: even smaller but slower.
. . . 1: probably not worthwhile
compared to skipping coprocessor
and using other CPU instructions.

An instruction for 4 rounds of
SHA-256 is in a few Intel CPUs.

ically designs
ge CPU cores:
1 data cache,
1 instruction cache,
ast multipliers,
fferent instructions,
rder unit, etc.

small cost for Intel
nstruction-set extension
avorite crypto!"

ut even smaller benefit
's mix of operations.

Intel did add instruction for 1 round of AES.

How many parallel S-boxes are in an AES-round coprocessor?

Can be 16: big; fast.
8: smaller but slower.
4: even smaller but slower.
. . . 1: probably not worthwhile compared to skipping coprocessor and using other CPU instructions.

An instruction for 4 rounds of SHA-256 is in a few Intel CPUs.

Lightwei

Frequent
where X

- Kecca
- any se
- a secu

"Resour
need the

gns
ores:
he,
on cache,
liers,
tructions,
etc.

for Intel
-set extension
pto!"

maller benefit
perations.

Intel did add instruction for 1 round of AES.

How many parallel S-boxes are in an AES-round coprocessor?

Can be 16: big; fast.
8: smaller but slower.
4: even smaller but slower.
. . . 1: probably not worthwhile compared to skipping coprocessor and using other CPU instructions.

An instruction for 4 rounds of SHA-256 is in a few Intel CPUs.

Lightweight crypto

Frequent claim in
where X might be
• Keccak;
• any secure hash;
• a secure cipher;
"Resource-constra
need the smallest

sion

efit

Intel did add instruction
for 1 round of AES.

How many parallel S-boxes are
in an AES-round coprocessor?

Can be 16: big; fast.
8: smaller but slower.
4: even smaller but slower.
. . . 1: probably not worthwhile
compared to skipping coprocessor
and using other CPU instructions.

An instruction for 4 rounds of
SHA-256 is in a few Intel CPUs.

Lightweight crypto

Frequent claim in literature,
where X might be

- Keccak;
- any secure hash;
- a secure cipher; . . . :

"Resource-constrained IoT d
need the smallest circuit for

Intel did add instruction
for 1 round of AES.

How many parallel S-boxes are
in an AES-round coprocessor?

Can be 16: big; fast.
8: smaller but slower.
4: even smaller but slower.
. . . 1: probably not worthwhile
compared to skipping coprocessor
and using other CPU instructions.

An instruction for 4 rounds of
SHA-256 is in a few Intel CPUs.

Lightweight crypto

Frequent claim in literature,
where X might be

- Keccak;
- any secure hash;
- a secure cipher; . . . :

"Resource-constrained IoT devices
need the smallest circuit for X."

Intel did add instruction
for 1 round of AES.

How many parallel S-boxes are
in an AES-round coprocessor?

Can be 16: big; fast.
8: smaller but slower.
4: even smaller but slower.
. . . 1: probably not worthwhile
compared to skipping coprocessor
and using other CPU instructions.

An instruction for 4 rounds of
SHA-256 is in a few Intel CPUs.

Lightweight crypto

Frequent claim in literature,
where X might be

- Keccak;
- any secure hash;
- a secure cipher; . . . :

"Resource-constrained IoT devices
need the smallest circuit for X."

—Even if speed is acceptable,
who will use smallest X circuit?

Intel did add instruction for 1 round of AES.

How many parallel S-boxes are in an AES-round coprocessor?

Can be 16: big; fast.
8: smaller but slower.
4: even smaller but slower.
. . . 1: probably not worthwhile compared to skipping coprocessor and using other CPU instructions.

An instruction for 4 rounds of SHA-256 is in a few Intel CPUs.

Lightweight crypto

Frequent claim in literature, where X might be

- Keccak;
- any secure hash;
- a secure cipher; . . . :

"Resource-constrained IoT devices need the smallest circuit for X."

—Even if speed is acceptable, who will use smallest X circuit?

Why should minimum area for X give minimum area for IoT+X?

add instruction
und of AES.

ny parallel S-boxes are
ES-round coprocessor?

16: big; fast.
er but slower.
smaller but slower.
robably not worthwhile
ed to skipping coprocessor
g other CPU instructions.

uction for 4 rounds of
6 is in a few Intel CPUs.

Lightweight crypto

Frequent claim in literature,
where X might be
- Keccak;
- any secure hash;
- a secure cipher; . . . :

"Resource-constrained IoT devices
need the smallest circuit for X."

—Even if speed is acceptable,
who will use smallest X circuit?

Why should minimum area for X
give minimum area for IoT+X?

An idea

Consider
public ke
receives
under th
verifies t

e.g. an S

Painful h
all client
to suppo
since old

uction
S.

l S-boxes are
coprocessor?

nst.
wer.
ut slower.
ot worthwhile
bing coprocessor
PU instructions.

4 rounds of
ew Intel CPUs.

Lightweight crypto

Frequent claim in literature, where X might be

- Keccak;
- any secure hash;
- a secure cipher; . . . :

"Resource-constrained IoT devices need the smallest circuit for X."

—Even if speed is acceptable, who will use smallest X circuit?

Why should minimum area for X give minimum area for IoT+X?

An idea from Adar

Consider a device
public keys from t
receives data supp
under these public
verifies these signa

e.g. an SSL client.

Painful historical e
all clients needed u
to support new ha
since old functions

are

r?

hile

cessor

ctions.

of

PUs.

Lightweight crypto

Frequent claim in literature, where X might be

- Keccak;
- any secure hash;
- a secure cipher; . . . :

"Resource-constrained IoT devices need the smallest circuit for X."

—Even if speed is acceptable, who will use smallest X circuit?

Why should minimum area for X give minimum area for IoT+X?

An idea from Adam Langley

Consider a device that recei
public keys from trusted sou
receives data supposedly sig
under these public keys;
verifies these signatures.

e.g. an SSL client.

Painful historical event:
all clients needed upgrades
to support new hash functio
since old functions were bro

Lightweight crypto

Frequent claim in literature,
where X might be
- Keccak;
- any secure hash;
- a secure cipher; . . . :

"Resource-constrained IoT devices need the smallest circuit for X."

—Even if speed is acceptable, who will use smallest X circuit?

Why should minimum area for X give minimum area for IoT+X?

An idea from Adam Langley

Consider a device that receives public keys from trusted sources; receives data supposedly signed under these public keys; verifies these signatures.

e.g. an SSL client.

Painful historical event:
all clients needed upgrades
to support new hash functions
since old functions were broken.

ght crypto

t claim in literature,
' might be
k;
cure hash;
re cipher; . . . :
ce-constrained IoT devices
e smallest circuit for X."

if speed is acceptable,
use smallest X circuit?

ould minimum area for X
imum area for IoT+X?

An idea from Adam Langley

Consider a device that receives public keys from trusted sources; receives data supposedly signed under these public keys; verifies these signatures.

e.g. an SSL client.

Painful historical event: all clients needed upgrades to support new hash functions since old functions were broken.

A public
signatur
in a limi

Langley'
Replace
a full pro
Then ca
(or upgr
signatur
keys, wit

o

literature,

e

;

. . . :

ined IoT devices

circuit for X.”

acceptable,

est X circuit?

num area for X

a for IoT+X?

An idea from Adam Langley

Consider a device that receives public keys from trusted sources; receives data supposedly signed under these public keys; verifies these signatures.

e.g. an SSL client.

Painful historical event: all clients needed upgrades to support new hash functions since old functions were broken.

A public key is a signature-verificati in a limited langua

Langley's idea: Replace this langu a full programmin Then can upgrade (or upgrade to pos signatures!) by cha keys, with no char

devices
X.”

e,
uit?

for X
-X?

An idea from Adam Langley

Consider a device that receives public keys from trusted sources; receives data supposedly signed under these public keys; verifies these signatures.

e.g. an SSL client.

Painful historical event:
all clients needed upgrades to support new hash functions since old functions were broken.

A public key is a signature-verification progra
in a limited language.

Langley’s idea:
Replace this language with
a full programming languag
Then can upgrade hash func
(or upgrade to post-quantu
signatures!) by changing pu
keys, with no changes to cli

An idea from Adam Langley

Consider a device that receives public keys from trusted sources; receives data supposedly signed under these public keys; verifies these signatures.

e.g. an SSL client.

Painful historical event: all clients needed upgrades to support new hash functions since old functions were broken.

A public key is a signature-verification program in a limited language.

Langley's idea: Replace this language with a full programming language. Then can upgrade hash function (or upgrade to post-quantum signatures!) by changing public keys, with no changes to clients.

An idea from Adam Langley

Consider a device that receives public keys from trusted sources; receives data supposedly signed under these public keys; verifies these signatures.

e.g. an SSL client.

Painful historical event: all clients needed upgrades to support new hash functions since old functions were broken.

A public key is a signature-verification program in a limited language.

Langley's idea: Replace this language with a full programming language. Then can upgrade hash function (or upgrade to post-quantum signatures!) by changing public keys, with no changes to clients.

Same for public-key encryption systems: public key is program.

from Adam Langley

r a device that receives
eys from trusted sources;
data supposedly signed
hese public keys;
these signatures.

SSL client.

historical event:
ts needed upgrades
ort new hash functions
d functions were broken.

A public key is a
signature-verification program
in a limited language.

Langley's idea:
Replace this language with
a full programming language.
Then can upgrade hash function
(or upgrade to post-quantum
signatures!) by changing public
keys, with no changes to clients.

Same for public-key encryption
systems: public key is program.

Say veri
is a chip
How sm

Have to
size of a
size of a

m Langley

that receives
rusted sources;
osedly signed
keys;
atures.

event:
upgrades
sh functions
were broken.

A public key is a
signature-verification program
in a limited language.

Langley's idea:
Replace this language with
a full programming language.
Then can upgrade hash function
(or upgrade to post-quantum
signatures!) by changing public
keys, with no changes to clients.

Same for public-key encryption
systems: public key is program.

Say verification de
is a chip of area A
How small can pu

Have to consider,
size of a SHA-256
size of a Keccak p

k

ves
rces;
ned

ns
ken.

A public key is a
signature-verification program
in a limited language.

Langley's idea:
Replace this language with
a full programming language.
Then can upgrade hash function
(or upgrade to post-quantum
signatures!) by changing public
keys, with no changes to clients.

Same for public-key encryption
systems: public key is program.

Say verification device
is a chip of area A.
How small can public keys b

Have to consider, e.g.,
size of a SHA-256 program,
size of a Keccak program, et

A public key is a
signature-verification program
in a limited language.

Langley's idea:
Replace this language with
a full programming language.
Then can upgrade hash function
(or upgrade to post-quantum
signatures!) by changing public
keys, with no changes to clients.

Same for public-key encryption
systems: public key is program.

Say verification device
is a chip of area A.
How small can public keys be?

Have to consider, e.g.,
size of a SHA-256 program,
size of a Keccak program, etc.

A public key is a
signature-verification program
in a limited language.

Langley's idea:
Replace this language with
a full programming language.
Then can upgrade hash function
(or upgrade to post-quantum
signatures!) by changing public
keys, with no changes to clients.

Same for public-key encryption
systems: public key is program.

Say verification device
is a chip of area A.
How small can public keys be?

Have to consider, e.g.,
size of a SHA-256 program,
size of a Keccak program, etc.

Similar question to optimizing
total size of a CPU with
a SHA-256 instruction,
a Keccak instruction, etc.

A public key is a
signature-verification program
in a limited language.

Langley's idea:
Replace this language with
a full programming language.
Then can upgrade hash function
(or upgrade to post-quantum
signatures!) by changing public
keys, with no changes to clients.

Same for public-key encryption
systems: public key is program.

Say verification device
is a chip of area A.
How small can public keys be?

Have to consider, e.g.,
size of a SHA-256 program,
size of a Keccak program, etc.

Similar question to optimizing
total size of a CPU with
a SHA-256 instruction,
a Keccak instruction, etc.

Not the usual code-size question.
Change the language!