Opportunities for Genetic Improvement of Cryptographic Code

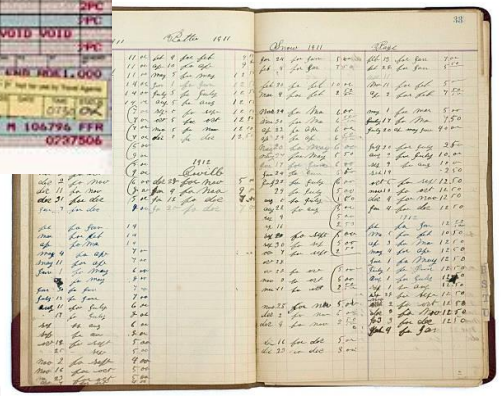
Yuval Yarom

Joint work with

Chitchanok Chuengsatiansup, Markus Wagner

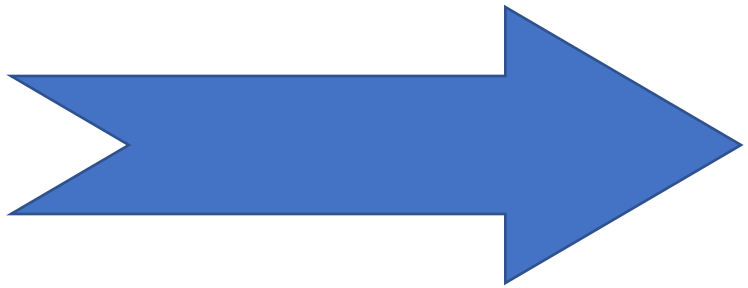
THE UNIVERSITY of ADELAIDE


QANTAS AIRWAYS LIMITED
PAPEETE
GANTIAGO ART MER
BUENOS AIRES EZE
FASTFR ISLAND
PAPEETE

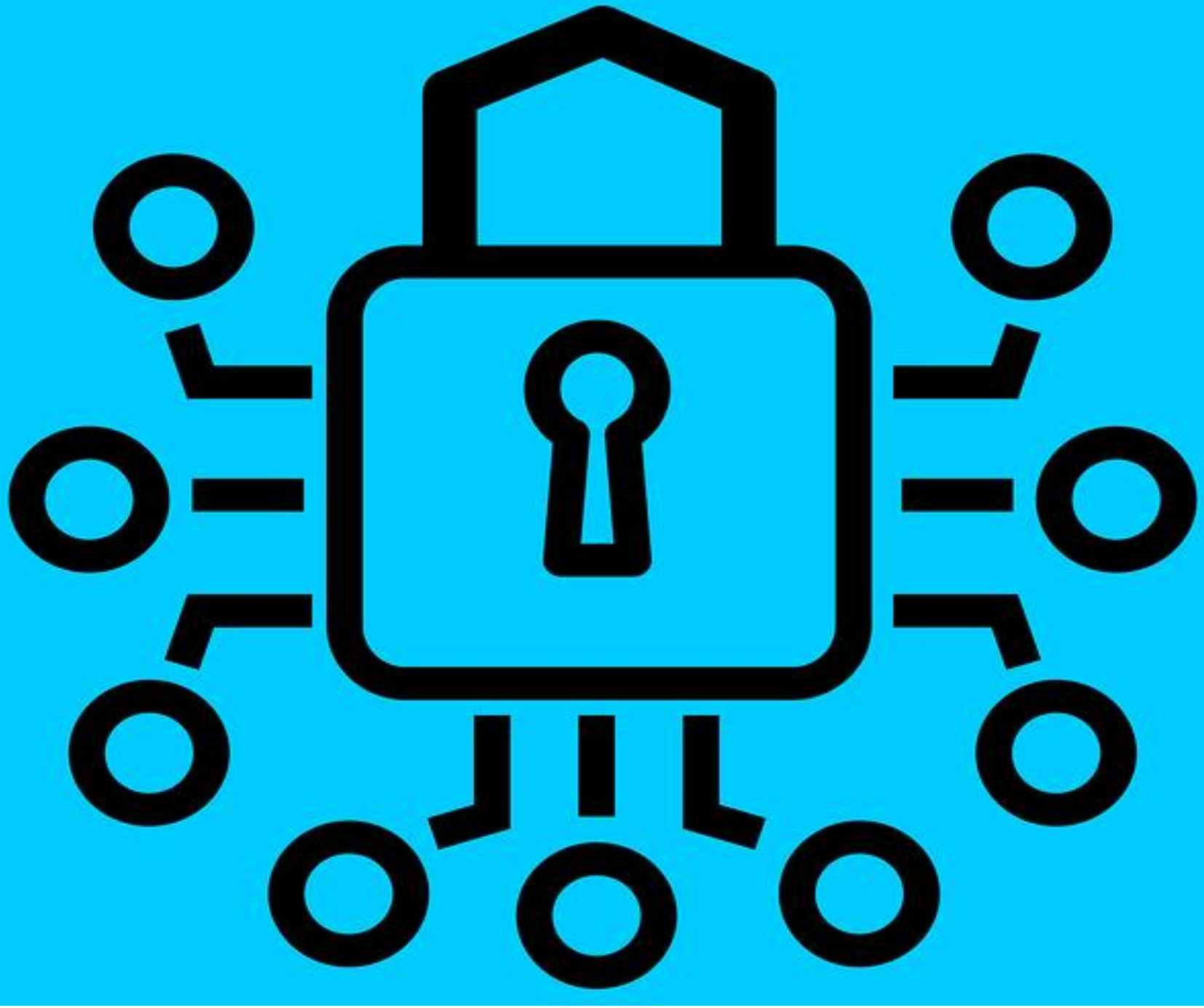



mix tape
the art of cassette culture
Compact Cassette
edited by thurston moore
universe


WALL STREET JOURNAL
FINANCIAL TIMES
What's News
Le Monde
DIE ZEIT
Libération
Frankfurter Allgemeine
Herald Tribune
La Tribune
Yeltsin Dismisses Hard-L




09:45
FaceTime
Calendar
Photos
Camera
Mail
Clock
Maps
Weather
Notes
Stocks
Books
TV
Health
Wallet
Settings

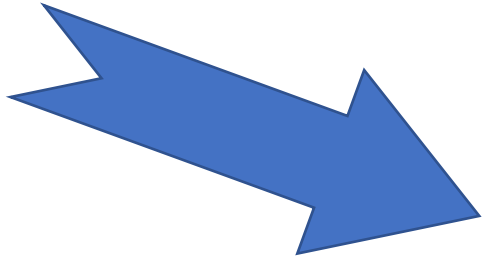


Cryptographic Code - Challenges

- Secure

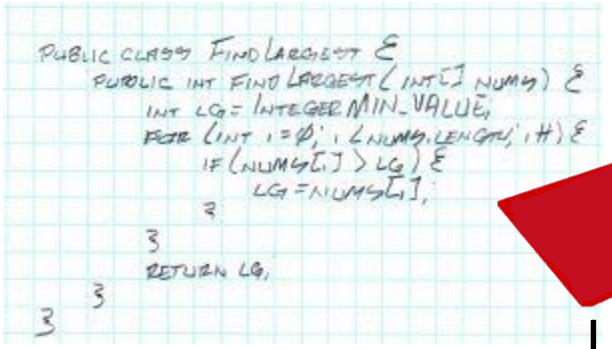
Provably correct

Leak free

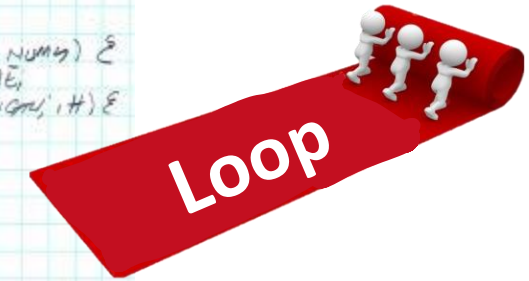


High expertise

- Efficient

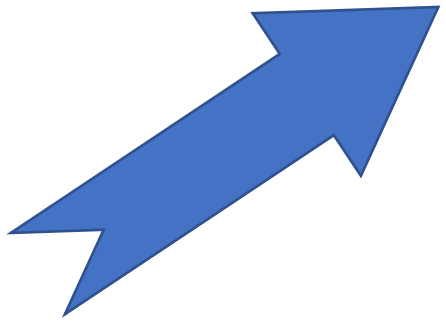


Hand tuned

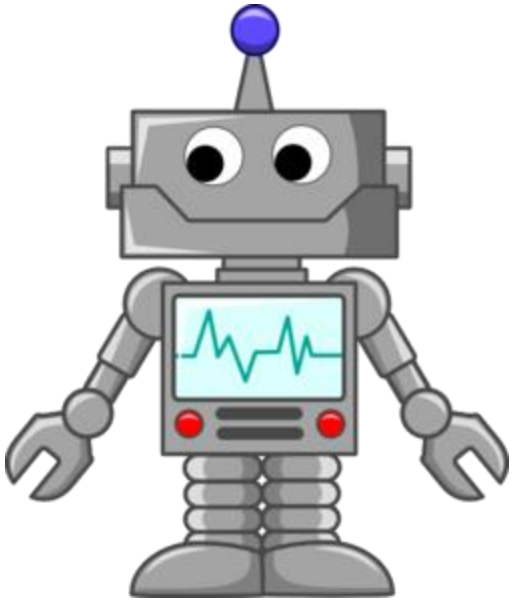


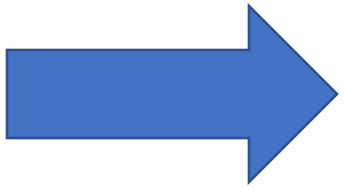
Loop unrolling

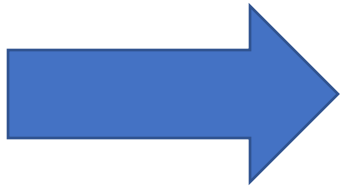
Assembly



Microarchitectural Side Channels





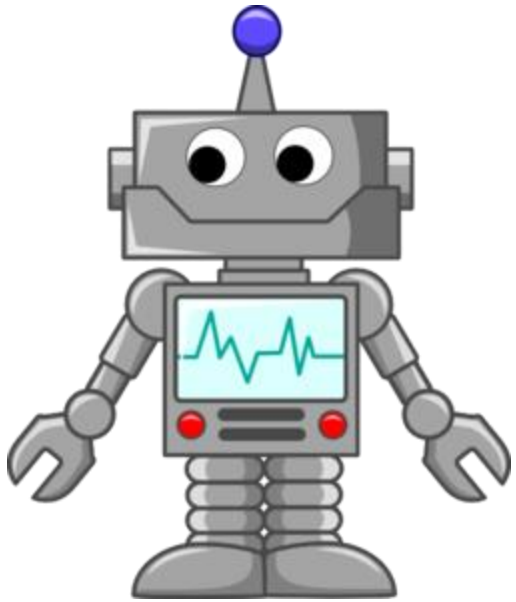


Program History

Cache

Execution Time

Microarchitectural Side Channels



Program History

Cache

Execution Time

Constant-time Programming

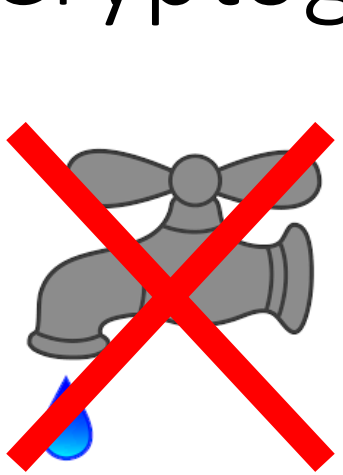
- A programming paradigm that mitigates microarchitectural side-channels
- No flow of secrets to
  - Branch conditions

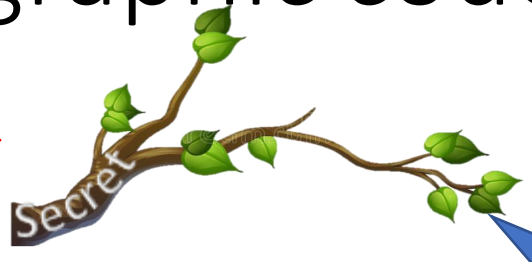
  - Memory addresses

  - Variable-time instructions

- De facto requirement for cryptographic code.

Cryptographic code - Opportunity





Secret

Large Basic Blocks

Mainstream compilers are not designed for large basic blocks

Simplicity is promising for combinatorial search

CryptOpt

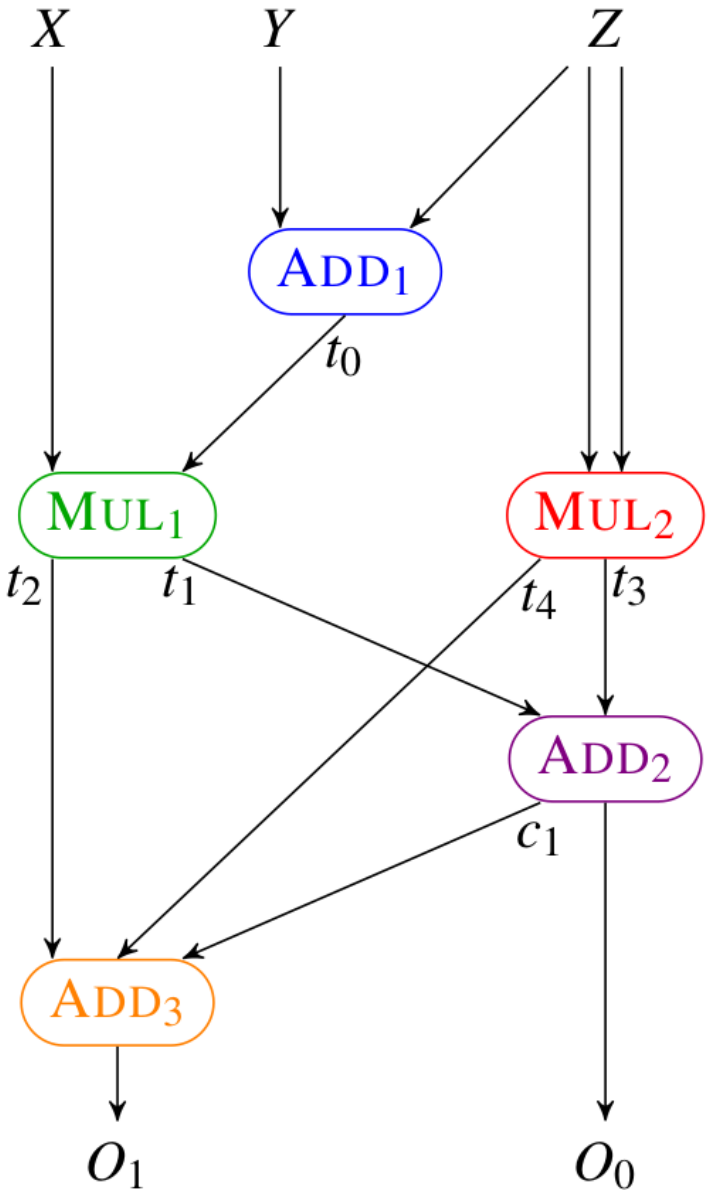
- Search-based code generator
- Currently tested on finite-field operations
  - e.g. arithmetic modulo 2^{255}-19
- Input: IR of a basic block from
  - Fiat Cryptography (Erbsen et al., IEEE SP 2019)
  - LLVM
- Output: X86 assembly
- Approach: Random Local Search

Cryptopt

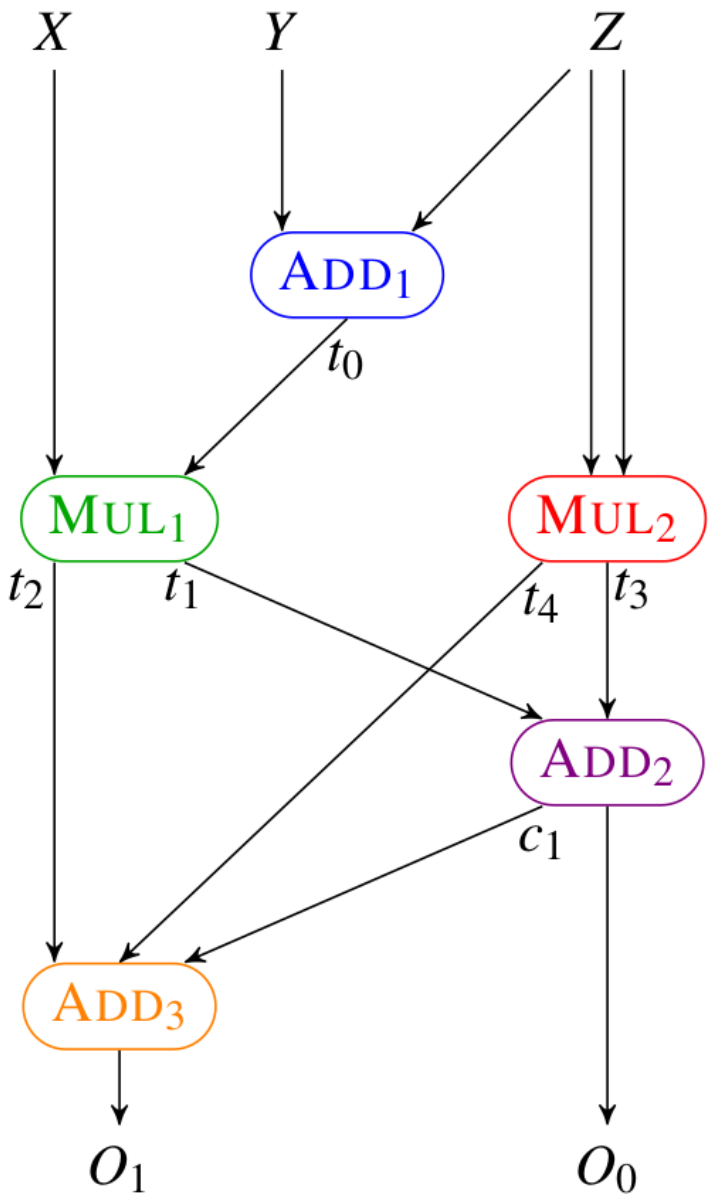
Algorithm 1: An Example Function

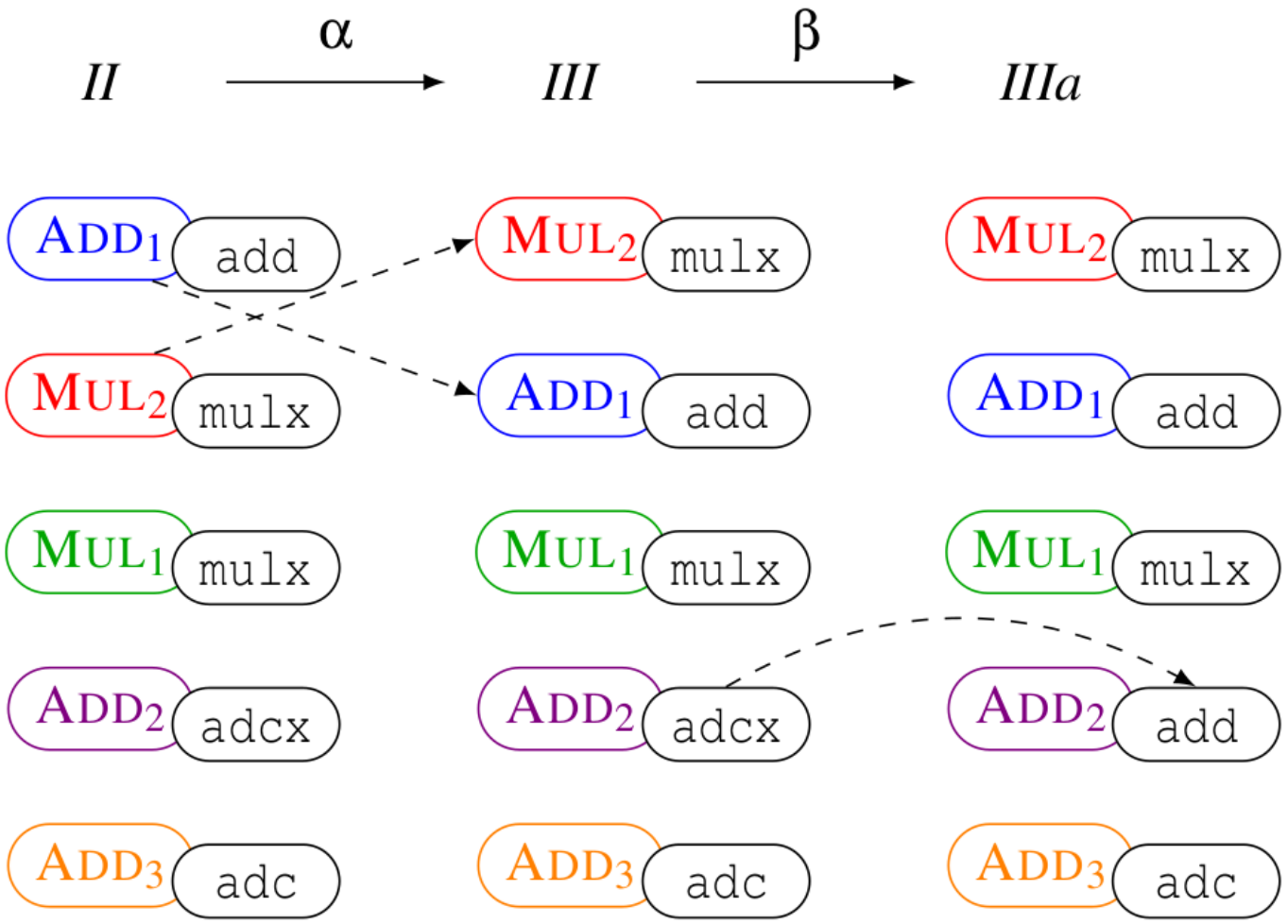
input : X, Y, Z such that $0 \leq X, Y, Z < 2^{63}$
output : $O = 2^{64}O_1 + O_0 = X \cdot (Y + Z) + Z^2$

function *example*$(X,\ Y,\ Z)$
begin
  $c_0,\ t_0 \leftarrow \text{ADD}_1(Y,\ Z,\ 0)$
  $t_2,\ t_1 \leftarrow \text{MUL}_1(t_0,\ X)$
  $t_4,\ t_3 \leftarrow \text{MUL}_2(Z,\ Z)$
  $c_1,\ O_0 \leftarrow \text{ADD}_2(t_1,\ t_3,\ 0)$
  $c_2,\ O_1 \leftarrow \text{ADD}_3(t_2,\ t_4,\ c_1)$
  return $O_1,\ O_0$
end

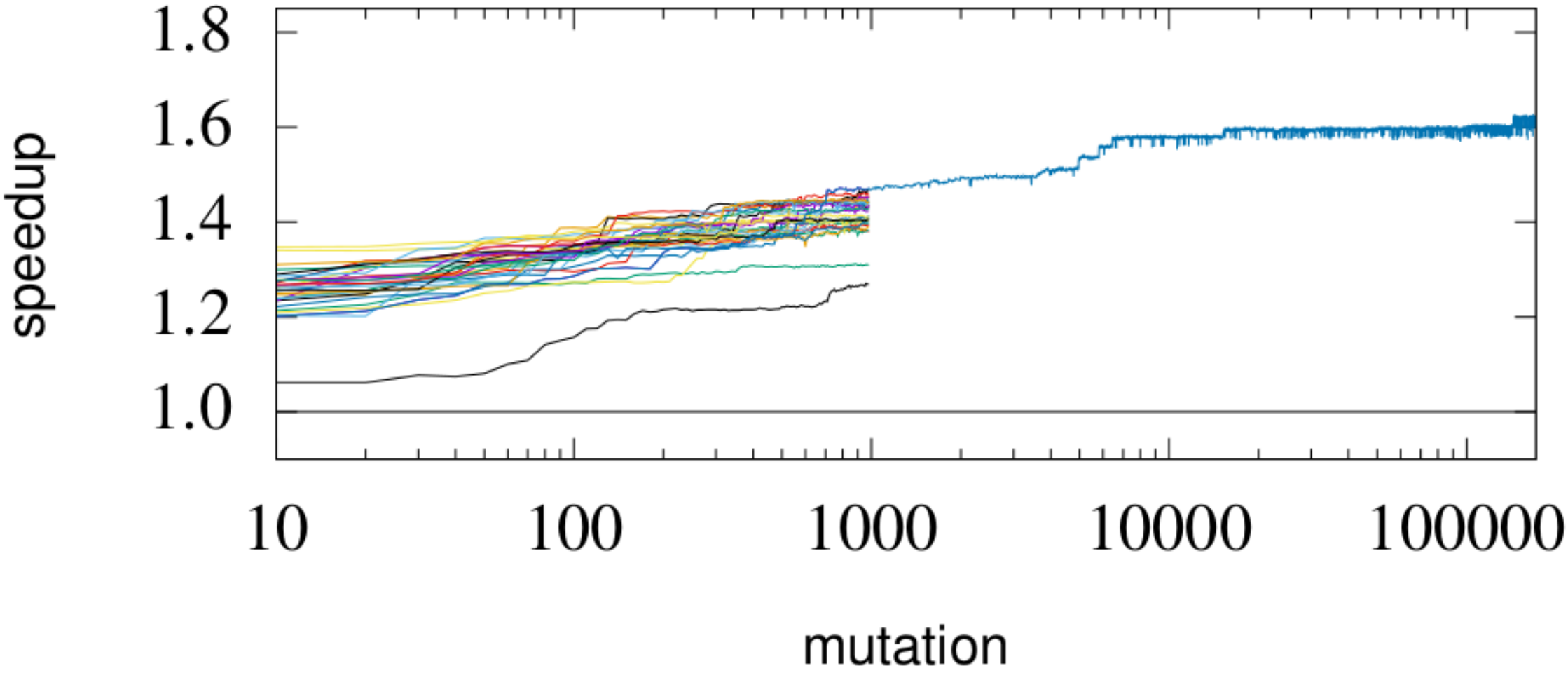


Mutations


X
Y
Z
ADD1
t0
MUL1
MUL2
t2
t1
t4
t3
ADD2
c1
ADD3
O1
O0


II
α
III
β
IIIa
ADD1 add
MUL2 mulx
MUL2
MUL2 mulx
ADD1 add
ADD1
MUL1 mulx
MUL1 mulx
MUL1
ADD2 adcx
ADD2 adcx
ADD2 add
ADD3 adc
ADD3 adc
ADD3 adc

Bet-and-Run in Action



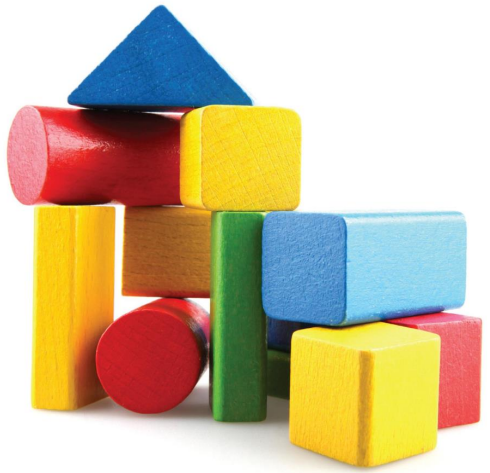
(a) NIST P-256-mul on i7 10G

Results

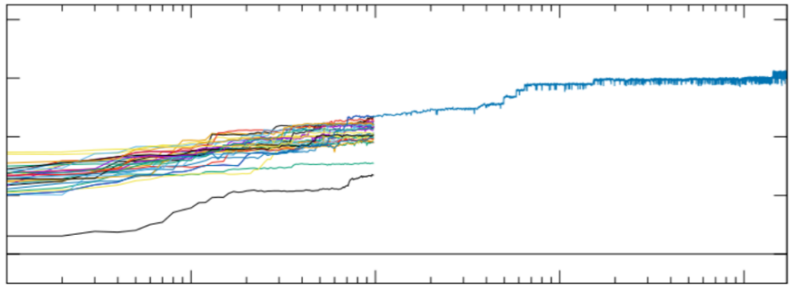
	Multiply		Square	
Curve	GCC	Clang	GCC	Clang
Curve25519	1.16	1.25	1.20	1.16
P-224	2.55	1.51	2.47	1.34
P-256	2.65	1.63	2.53	1.52
P-384	2.07	1.21	2.15	1.17
SIKEp434	2.10	1.46	2.13	1.44
Curve448	0.82	0.96	0.89	0.88
P-521	1.00	1.24	1.11	1.42
Poly1305	1.17	1.11	1.18	1.03
secp256k1	2.30	1.61	2.24	1.51

Summary

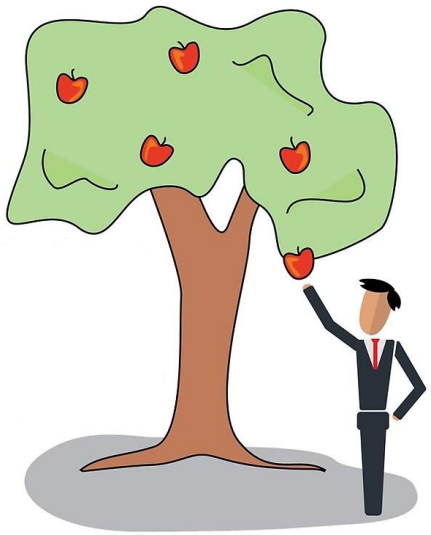
- Cryptographic code tends to have large basic blocks



- These are good for combinatorial search



- CryptOpt uses RLS to optimise finite-field operations



- Many more low hanging fruits

	Multiply		Square	
Curve	GCC	Clang	GCC	Clang
Curve25519	1.16	1.25	1.20	1.16
P-224	2.55	1.51	2.47	1.34
P-256	2.65	1.63	2.53	1.52
P-384	2.07	1.21	2.15	1.17
SIKEp434	2.10	1.46	2.13	1.44
Curve448	0.82	0.96	0.89	0.88
P-521	1.00	1.24	1.11	1.42
Poly1305	1.17	1.11	1.18	1.03
secp256k1	2.30	1.61	2.24	1.51