CryptOpt

Automatic Optimization of Straightline Code

Joel Kuepper, Andres Erbsen, Jason Gross, Owen Conoly, Chuyue Sun, Samuel Tian, David Wu, Adam Chlipala, Chitchanok Chuengsatiansup, Daniel Genkin, Markus Wagner, Yuval Yarom

THE UNIVERSITY of ADELAIDE

THE UNIVERSITY OF MELBOURNE

Georgia Tech

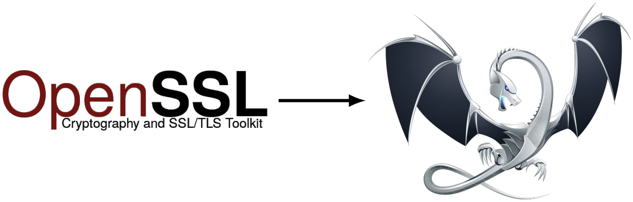
MONASH University

RUB

Motivation

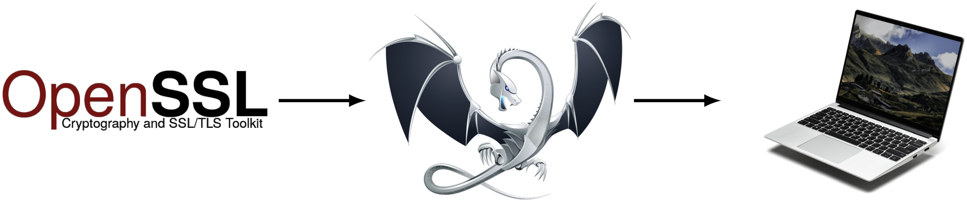
LLVM

Motivation



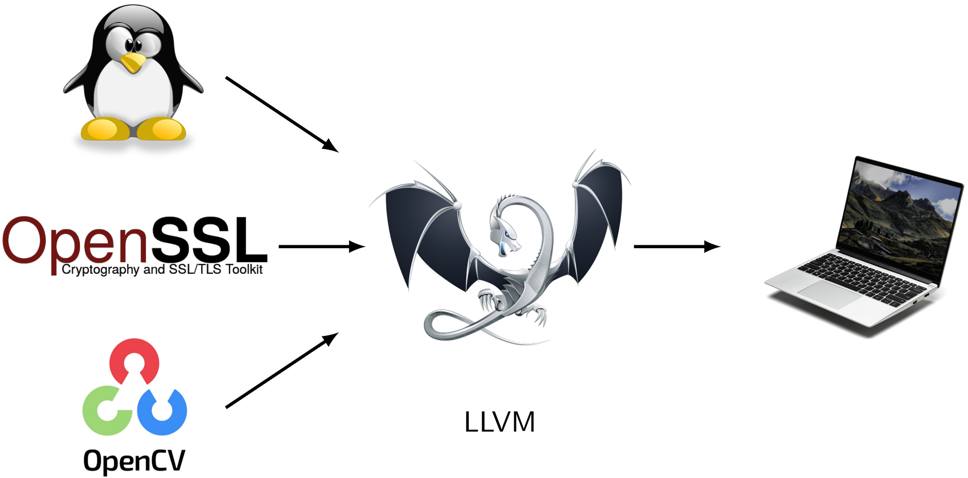
LLVM

Motivation



LLVM

Motivation



LLVM

Idea

Observations:

1. Compilers are general-purpose.

Idea

Observations:

1. Compilers are general-purpose.
2. Cryptographic code is “special”.

Idea

Observations:

1. Compilers are general-purpose.
2. Cryptographic code is “~~special~~ simpler”.

Idea

Observations:

1. Compilers are general-purpose.
2. Cryptographic code is "~~special~~ simpler".

Idea:

1. ~~Compiling to~~ → search for a fast implementation.

Idea

Observations:

1. Compilers are general-purpose.
2. Cryptographic code is “~~special~~ simpler”.

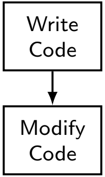
Idea:

1. ~~Compiling to~~ → search for a fast implementation.
2. Prove it correct. (→ in Full Paper, PLDI’23, Distinguished Paper)

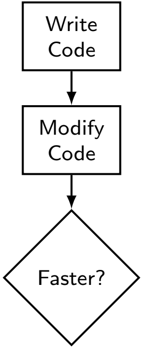
Search for Fast Implementation

Search for Fast Implementation

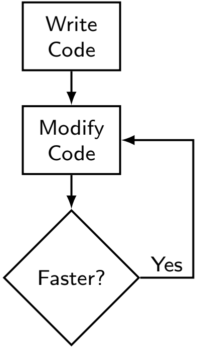
Search for Fast Implementation



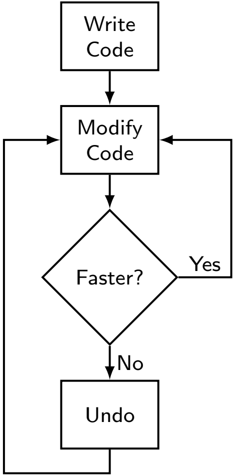
Search for Fast Implementation



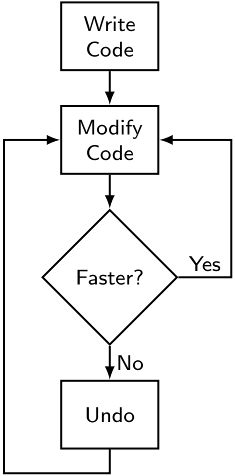
Search for Fast Implementation



Search for Fast Implementation

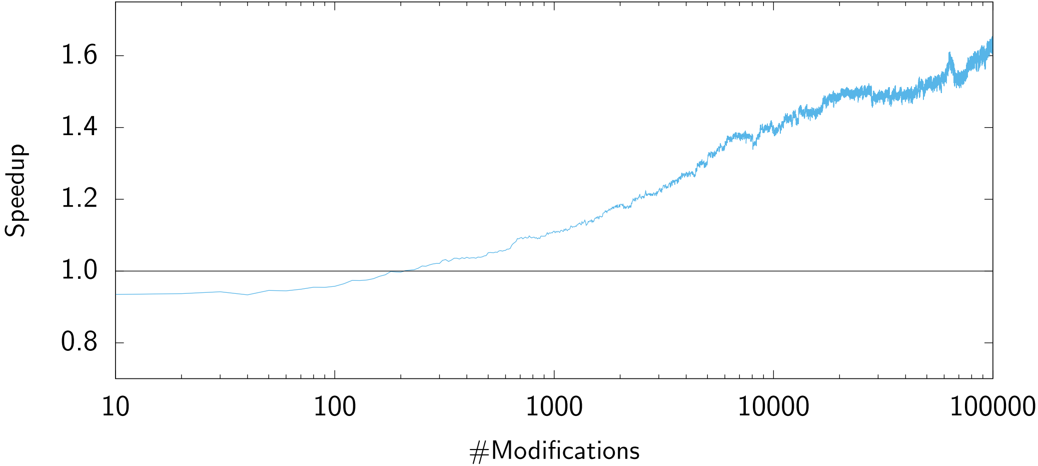


Search for Fast Implementation



"Random Local Search"

Performance



Fiat Cryptography

Fiat Cryptography
[Erbsen et al. 2019, IEEE S&P]

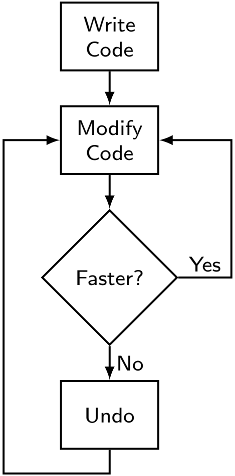
Fiat Cryptography

Functional Program

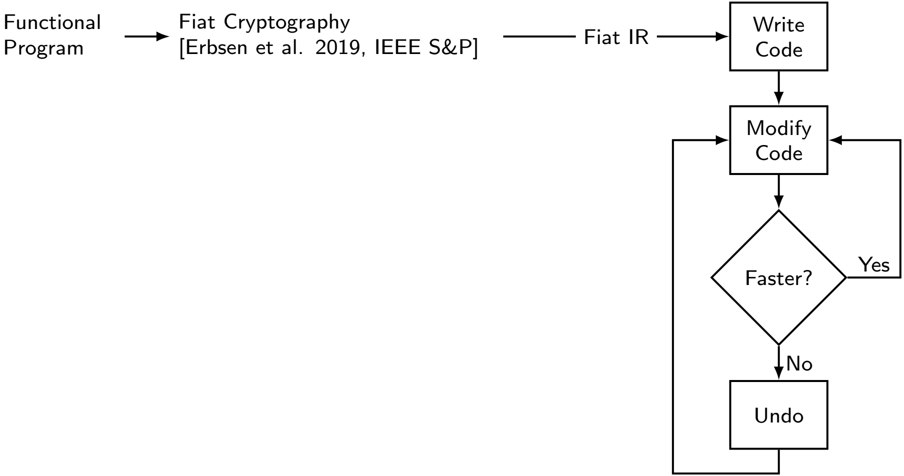
Fiat Cryptography [Erbsen et al. 2019, IEEE S&P]

Fiat Cryptography

Functional Program → Fiat Cryptography [Erbsen et al. 2019, IEEE S&P]



Fiat Cryptography



Functional Program

Fiat Cryptography [Erbsen et al. 2019, IEEE S&P]

Fiat IR

Write Code

Modify Code

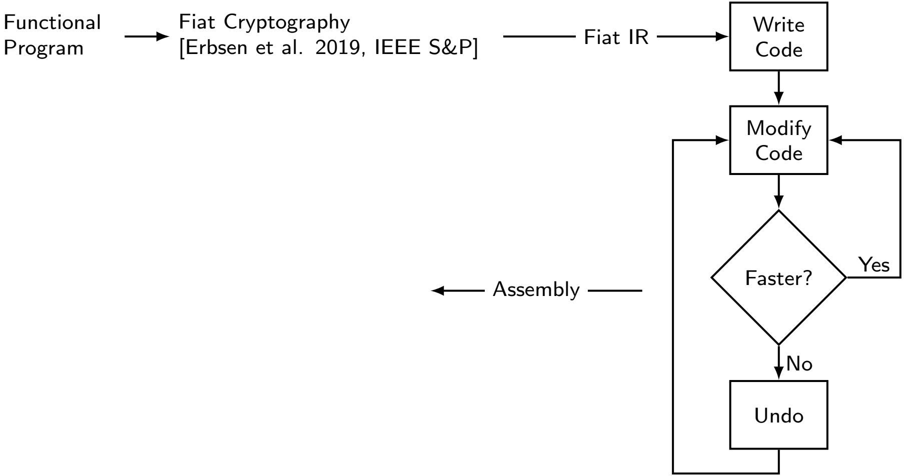
Faster?

Yes

No

Undo

Fiat Cryptography



Functional Program

Fiat Cryptography [Erbsen et al. 2019, IEEE S&P]

Fiat IR

Write Code

Modify Code

Faster?

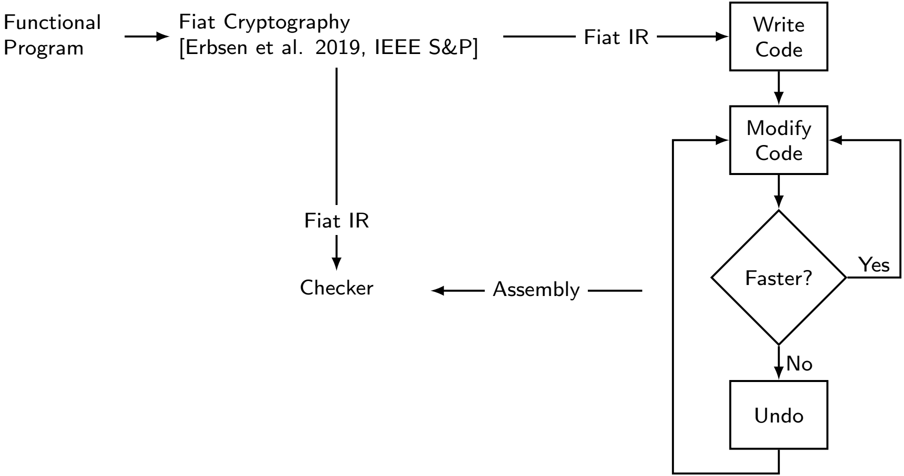
Yes

No

Undo

Assembly

Fiat Cryptography



Performance: Field Arithmetic

Geometric Mean (4x AMD, 6x Intel)

	Multiply		Square	
Curve	Clang	GCC	Clang	GCC
Curve25519				
P-224				
P-256				
P-384				
SIKEp434				
Curve448				
P-521				
Poly1305				
secp256k1				

Performance: Field Arithmetic

Geometric Mean (4x AMD, 6x Intel)

	Multiply		Square	
Curve	Clang	GCC	Clang	GCC
Curve25519	1.19	1.14	1.14	1.18
P-224	1.31	1.87	1.24	1.84
P-256	1.27	1.79	1.30	1.85
P-384	1.12	1.66	1.08	1.60
SIKEp434	1.30	1.70	1.29	1.83
Curve448	1.02	0.95	1.00	0.99
P-521	1.20	1.06	1.25	1.11
Poly1305	1.10	1.15	1.09	1.16
secp256k1	1.34	1.73	1.32	1.74

Summary

Summary

Compilation of straightline code
$\Longrightarrow$ Search

Summary

Compilation of straightline code
$\implies$ Search

Random Local Search + Runtime

Summary

Compilation of straightline code
$\Longrightarrow$ Search

Random Local Search + Runtime

Proven-correct assembly for field arithmetic by Fiat Cryptography now with on-par performance to hand-optimized assembly. (see full Paper)

Summary

Compilation of straightline code
$\implies$ Search

Random Local Search + Runtime

Proven-correct assembly for field arithmetic by Fiat Cryptography now with on-par performance to hand-optimized assembly. (see full Paper)

Summary

GitHub Project
(with links to papers)

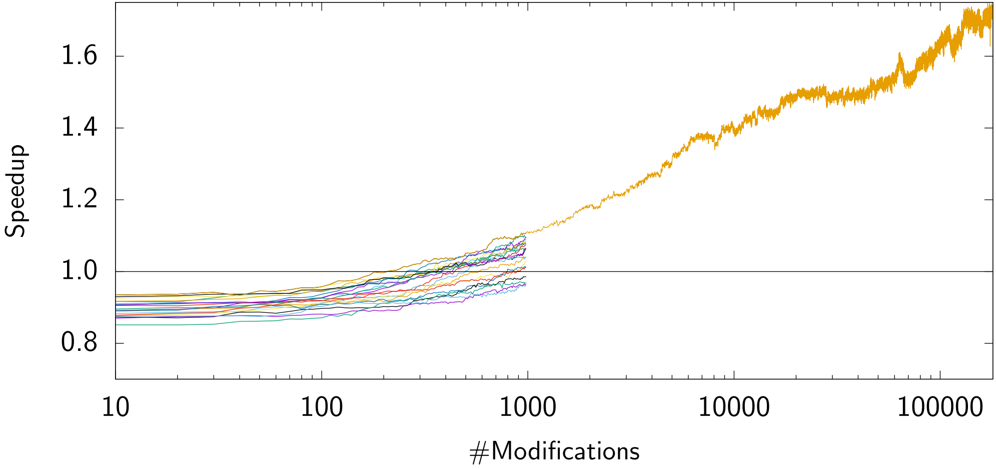
Compilation of straightline code
$\implies$ Search

Random Local Search + Runtime

Proven-correct assembly for field arithmetic by Fiat Cryptography now with on-par performance to hand-optimized assembly. (see full Paper)

`https://0xade1a1de.github.io/CryptOpt`

Bet and Run


Speedup
1.6
1.4
1.2
1.0
0.8
10
100
1000
10000
100000
#Modifications