High-assurance Cryptography

Real World Crypto (RWC 2016)

Dave Archer, Tom DuBuisson, Nathan Collins, Joey Dodds, Trevor Eliott, Iavor Diatchki, Rob Dockins, Adam Foltzer, Joe Hendrix, Brian Huffman, **Joe Kiniry**, John Launchbury, Dylan McNamee, Aaron Tomb, Daniel Wagner, Simon Winwood, Dan Zimmerman, and many other current and past Galois employees who worked on high-assurance cryptography

Galois, Inc.

January 2016

galois

Based in **Portland, Oregon and Washington, D.C.**

Founded in 1999

60+ full time employees

Mission: Create Trustworthiness in Critical Systems.

Areas of expertise:

Programming Languages, Formal Methods, Security.

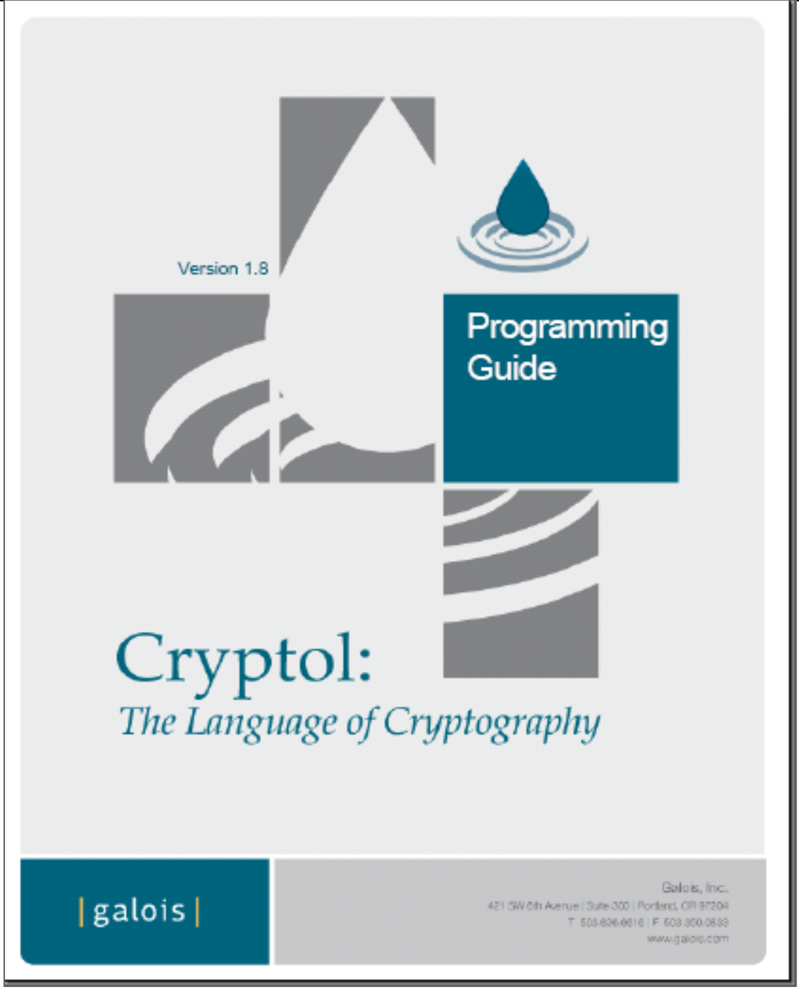
Company Facts

Our Contribution

Galois has developed tools for showing that **different** cryptographic implementations compute the **same** values for all possible keys and inputs.

Uses formal verification techniques including symbolic simulation, rewriting, and third-party SAT and SMT-solvers.

From a user's perspective, our tools act like compilers and perform exhaustive test coverage.

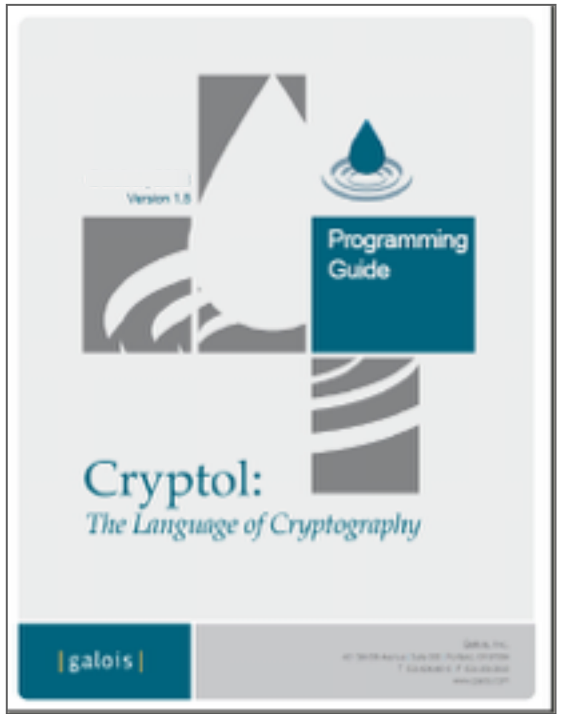


ABC

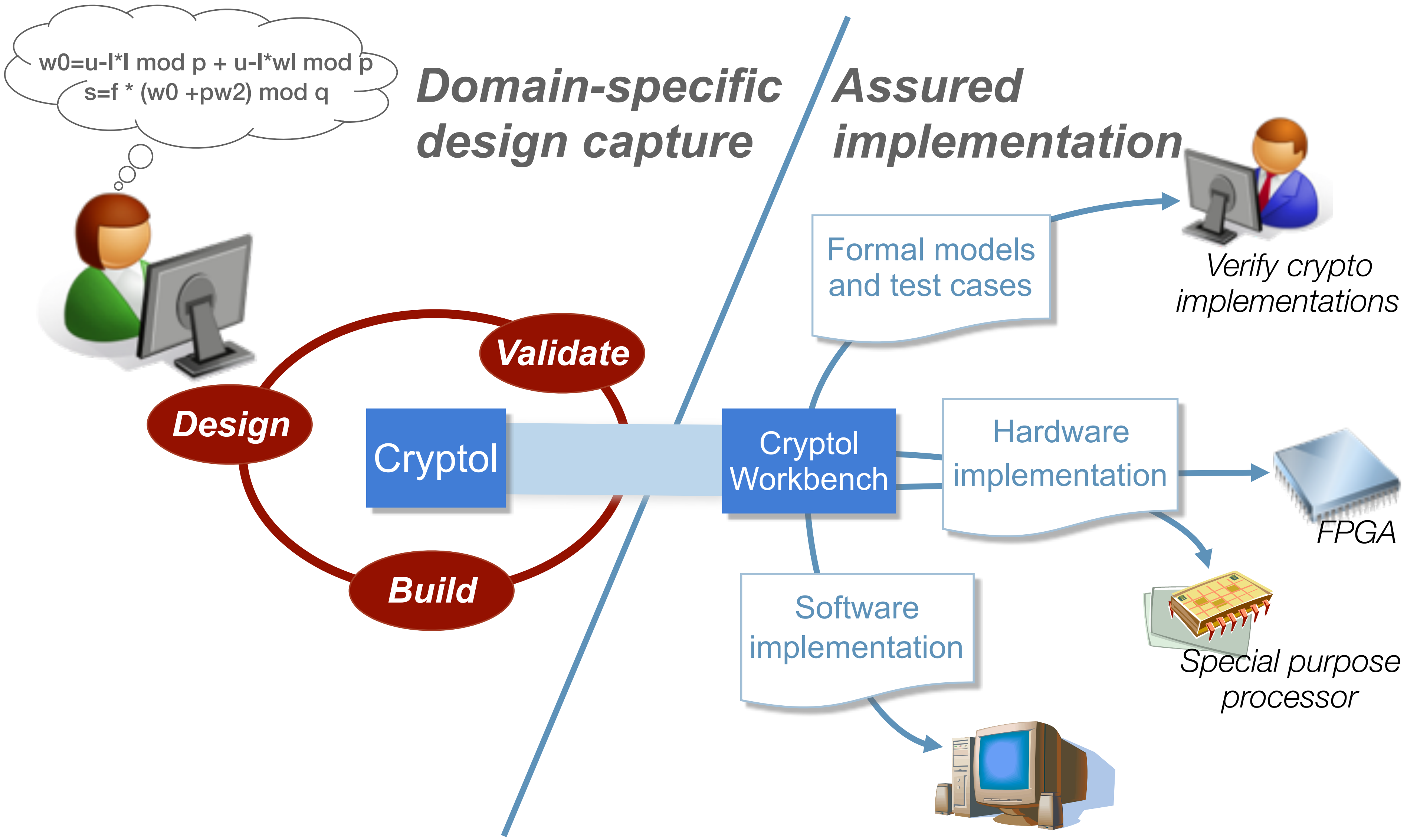
Yices

Cryptol: The Language of Cryptography

- Declarative specification language, tailored to the crypto domain, designed with feedback from cryptographers, and is dependently typed, pure functional language
- Cryptol 2 is open source, BSD licensed, on GitHub, has seen several releases over the past two years, and runs on all major platforms (check out http://cryptol.net/)
- Cryptol includes a REPL, support for literate programming, a parser, type checker, symbolic evaluator, quickcheck-style runtime validation, and SAT and SMT-based verification
- Cryptol specifications exist for nearly every standardized or proposed cryptographic algorithm, including many curves and some post-quantum algorithms



One specification - Many uses



SAW:
The Software Assurance Workbench

- capable of reasoning about the equivalency of Cryptol, LLVM, and JVM specifications and implementations
- highly tuned toward bit-centric computing (e.g., crypto, compression, codecs, etc.)
- works in tandem with Cryptol
- is open source, non-commercial licensed, on GitHub, was released mid-last year, and runs on all major platforms (see http://saw.galois.com/)
- includes a REPL and a proof specification language that supports compositional proof techniques spanning platforms and solvers
- SAWcore is the IR for semantic representation (dependently-

VHDL

C

Language

LLVM

Java

Cryptol

SAW

System Verilog

Verification Ecosystem

Verification of Suite B Algorithms and more

Suite B Verification Efforts

	Role	Implementation	Lines of Code
AES-128	Symmetric Key Cipher	BouncyCastle (Java)	817
SHA-384	Secure Hash Function	libgcrypt (C)	423
ECDSA (P-384)	Digital Signature Scheme	galois (Java)	2348

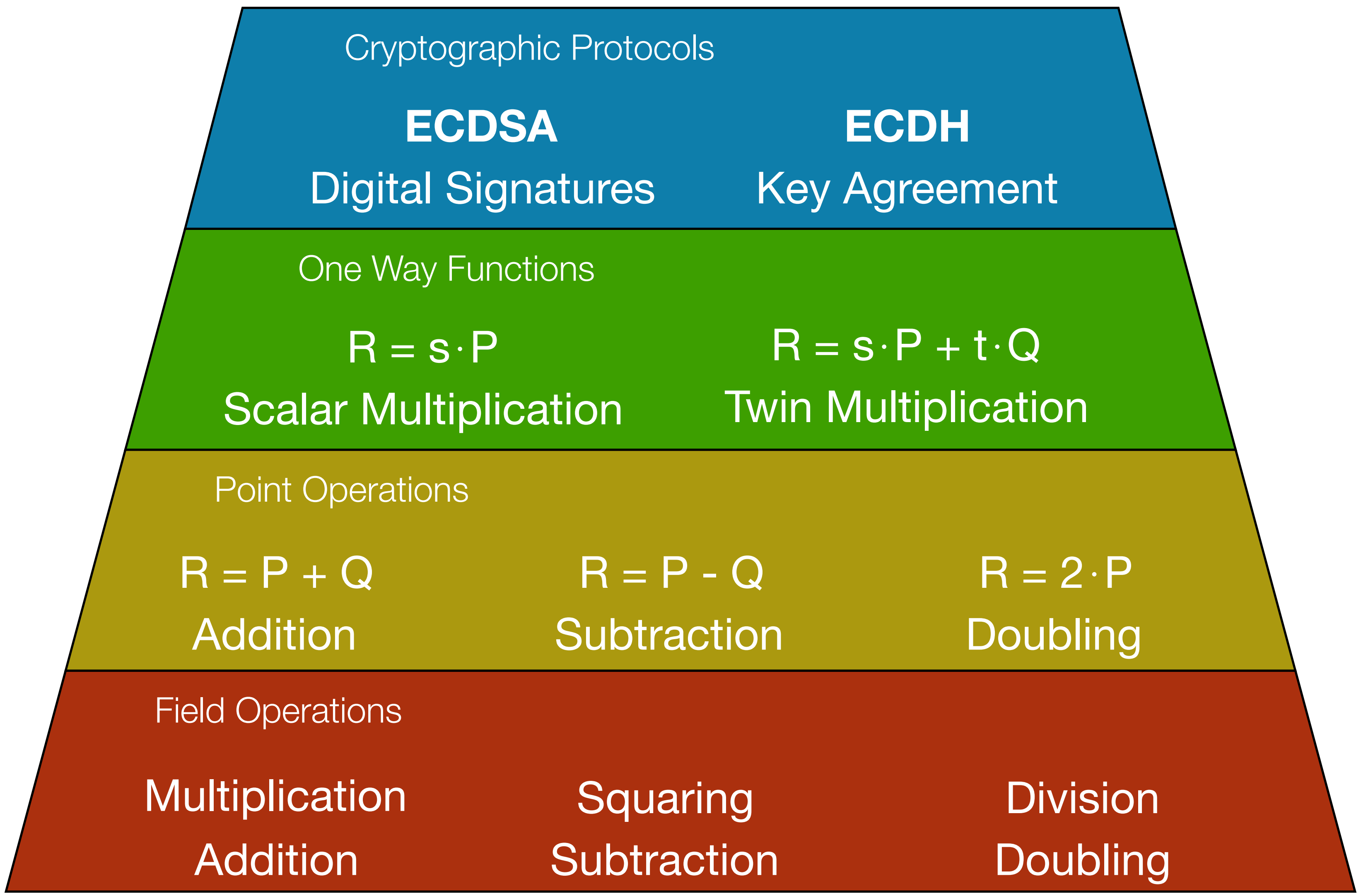
Some Suite B Problem Sizes

	Lines of Code	AIG Size	Decomposition Steps Required	Verification Time
AES-128 BouncyCastle AESFastEngine	817	1MB	None needed **Fully automatic**	40 min
SHA-384 libgcrypt	423	3.2MB	10 steps **All solved via SAT**	160 min
ECDSA (P-384) (galois)	2348	More than 5GB	48 steps **Multiple tactics required**	10 min

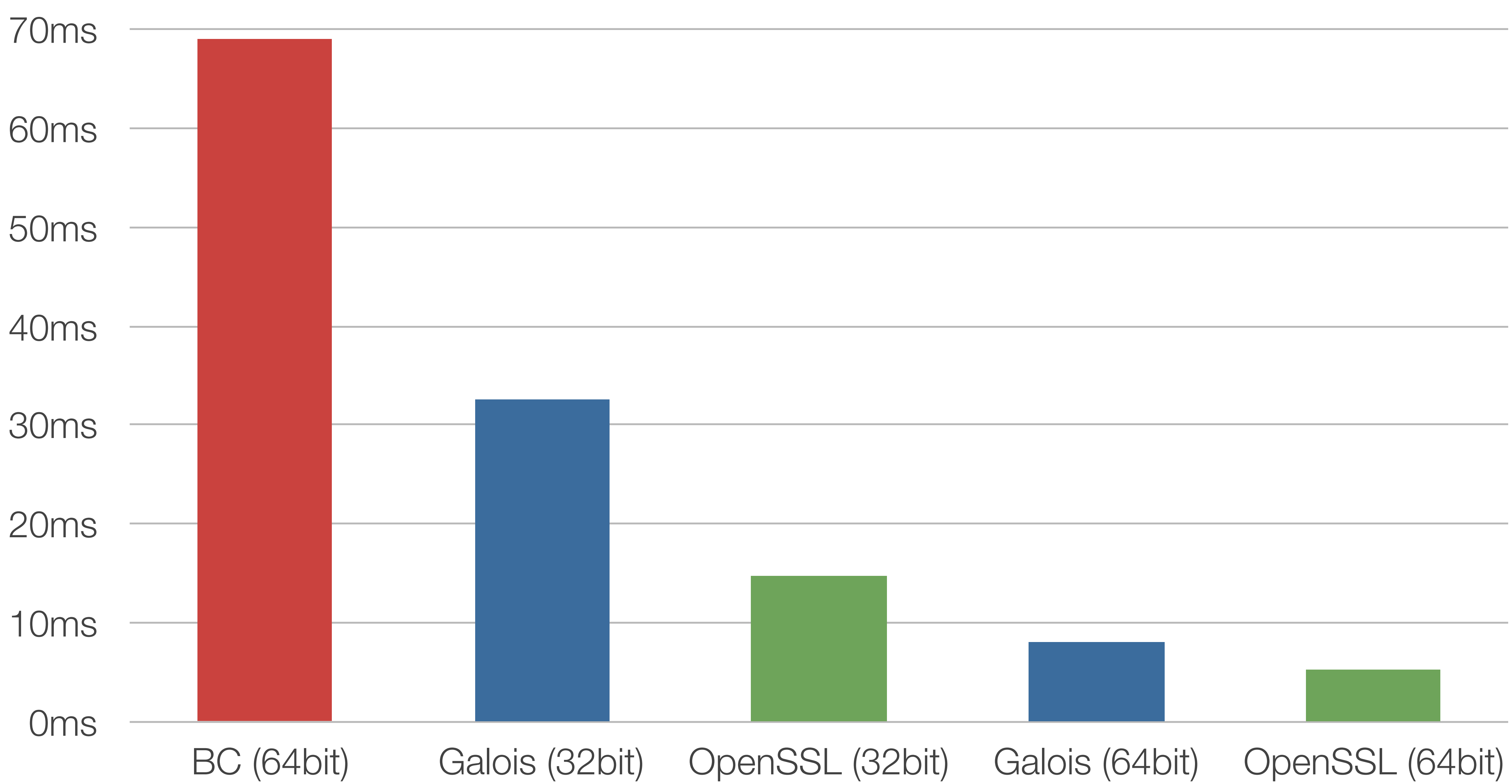
ECC Verification Target

- verify an efficient implementation of ECDSA over NIST P-384 curve in Java (to our knowledge, the fastest in existence)
  - use known optimizations such as twin multiplication, projective coordinates, optimized field arithmetic
  - specification can use the same high-performance published algorithms as the implementation
  - implementation uses many low-level tricks for improving efficiency, often verified via a refinement style of verification

Implementing ECC



ECC Benchmarks Sign & Verify



Verification Statistics

- 48 method specifications in total
  - 2 protocol specifications (verify & sign)
  - 8 scalar multiplication specifications
  - 3 point specifications (add, subtract, double)
  - 20 field specifications
  - 15 bitvector specifications
- total verification time is under 10 minutes

Found Three Bugs

- sign & verify failed to clear all intermediate results
- boundary condition due to use of less-than where less-than-or-equal was needed
- modular reduction failed to propagate one overflow

Automatic Synthesis of High-Performance Software and Hardware Implementations

Synthesis

- our flagship product in the synthesis space was Cryptol version 1
- it is capable of generating verifiable C, JVM, VHDL, and Verilog implementations of Cryptol specs
- implementations witness decent performance
- implementations can be verified with other toolchains
- synthesis goals previously focused exclusively on code as the target artifact, not validation or verification artifacts like test benches or proofs, resp.

Synthesis

- we are forward porting ideas and code from our synthesis tools into Cryptol version 2 and SAW
- we now have prototypes of fully automatic synthesis of rigorously engineered C, LLVM, and SystemVerilog[-CSP] implementations
- artifacts included in the development method for synthesis include a domain model, requirements, correctness and security policies, and an architecture specification
- artifacts synthesized in addition to implementations include source documentation, validation artifacts (unit and system runtime verification harness), and theorems for other verification systems

Galois High-Assurance Crypto Tool Suite