Cryptol: The Language of ~~Cryptography~~ Cryptanalysis

Joe Hurd & Sally A Browning | SHARCS | March 2012

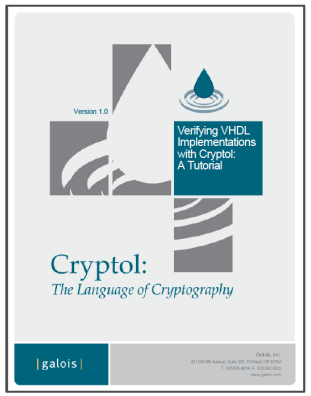
The Cryptol team, past and present:
Sally Browning, Ledah Casburn, Iavor Diatchki, Trevor Elliot, Levent Erkok, Sigbjorn Finne, Adam Foltzer, Andy Gill, Fergus Henderson, Joe Hendrix, Joe Hurd, John Launchbury, Jeff Lewis, Lee Pike, John Matthews, Thomas Nordin, Mark Shields, Joel Stanley, Frank Seaton Taylor, Jim Teisher, Aaron Tomb, Philip Weaver, Adam Wick, Edward Yang

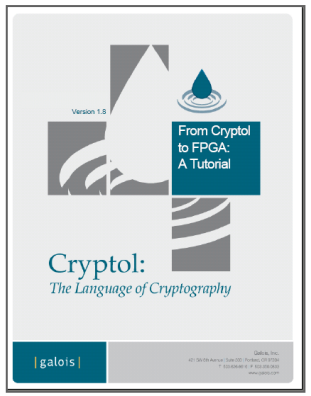
galois

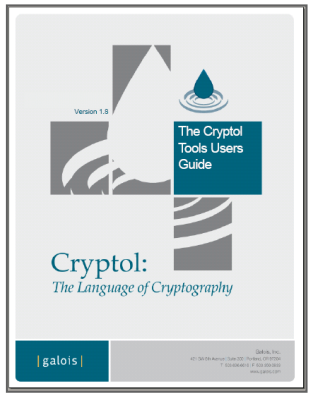
| galois |

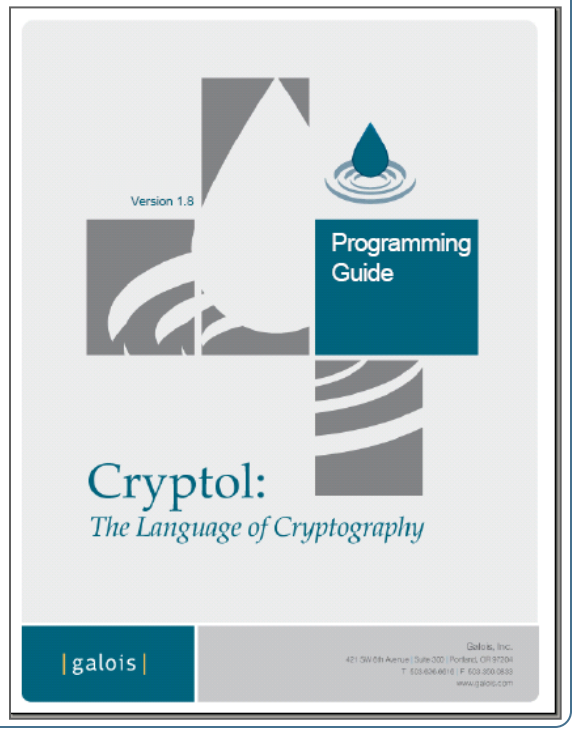
Cryptol: The language of cryptography

- Declarative specification language
  - Language tailored to the crypto domain
  - Designed with feedback from NSA
- Execution and validation tools
  - Tool suite for different implementation and verification applications
  - In use by crypto-implementers









Applying Cryptol to cryptanalysis

- **Thesis:** Cryptol also serves as a domain-specific language for *cryptanalysis*.
  - The same language constructs are useful for both cryptography and cryptanalysis.
  - The back-ends support efficient cryptanalysis on a variety of hardware/software platforms.
  - The verification toolset can be used to probe a cryptographic algorithm for weaknesses.
- This talk will cover these topics and show their use in a cryptanalysis demo using Cryptol.

THE LANGUAGE

A Taste of Cryptol

Cryptol programs

- File of mathematical definitions
  - Has a clean, unambiguous semantics
  - Strong static bit-precise typing
- Definitions are computationally neutral
  - Cryptol tools provide the computational content (interpreters, compilers, code generators, verifiers)

```
x : [4][32];
x = [23 13 1 0];

F : ([16],[16]) -> [16];
F (x,y) = 2 * x + y;
```

galois

Cryptol: Specify interfaces unambiguously

From the Advanced Encryption Standard definition†

3.1 Inputs and Outputs

The **input** and **output** for the AES algorithm each consist of **sequences of 128 bits** (digits with values of 0 or 1). These sequences will sometimes be referred to as **blocks** and the number of bits they contain will be referred to as their length. The **Cipher Key** for the AES algorithm is a **sequence of 128, 192 or 256 bits**. Other input, output and Cipher Key lengths are not permitted by this standard.

Cryptol

```
blockEncrypt : {k} (k >= 2, 4 >= k) => ([128], [64*k]) -> [128]
```

For all `k`

…between 2 and 4

First input is a sequence of 128 bits

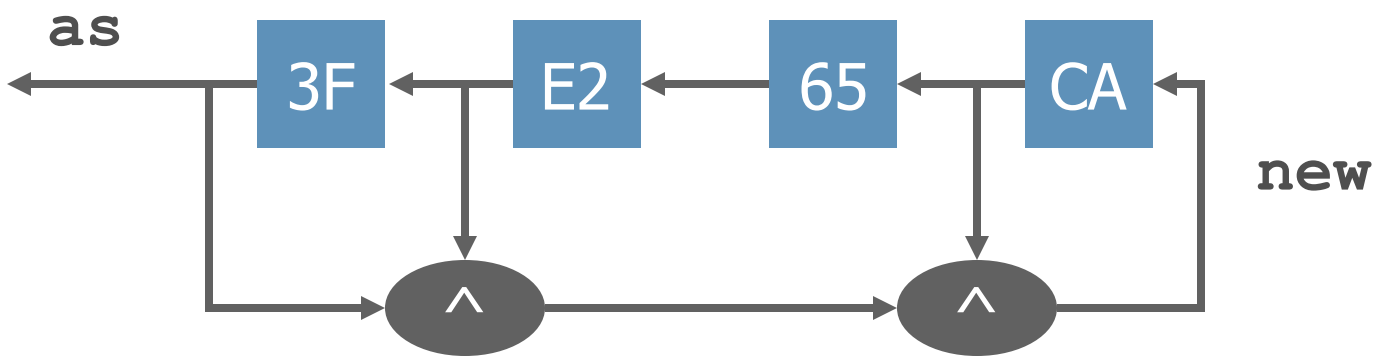
Second input is a sequence of 128, 192, or 256 bits

Output is a sequence of 128 bits

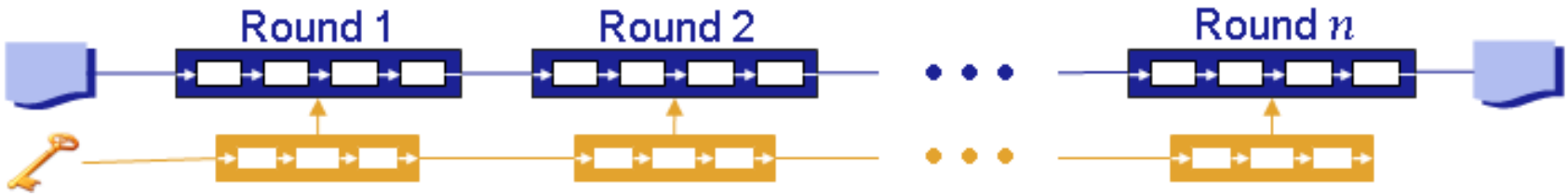
†http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

Cryptol: Natural expression of stream equations

```
as  = [0x3F 0xE2 0x65 0xCA] # new;
new = [| a ^ b ^ c || a <- as
                   || b <- drop(1,as)
                   || c <- drop(3,as)  |];
```



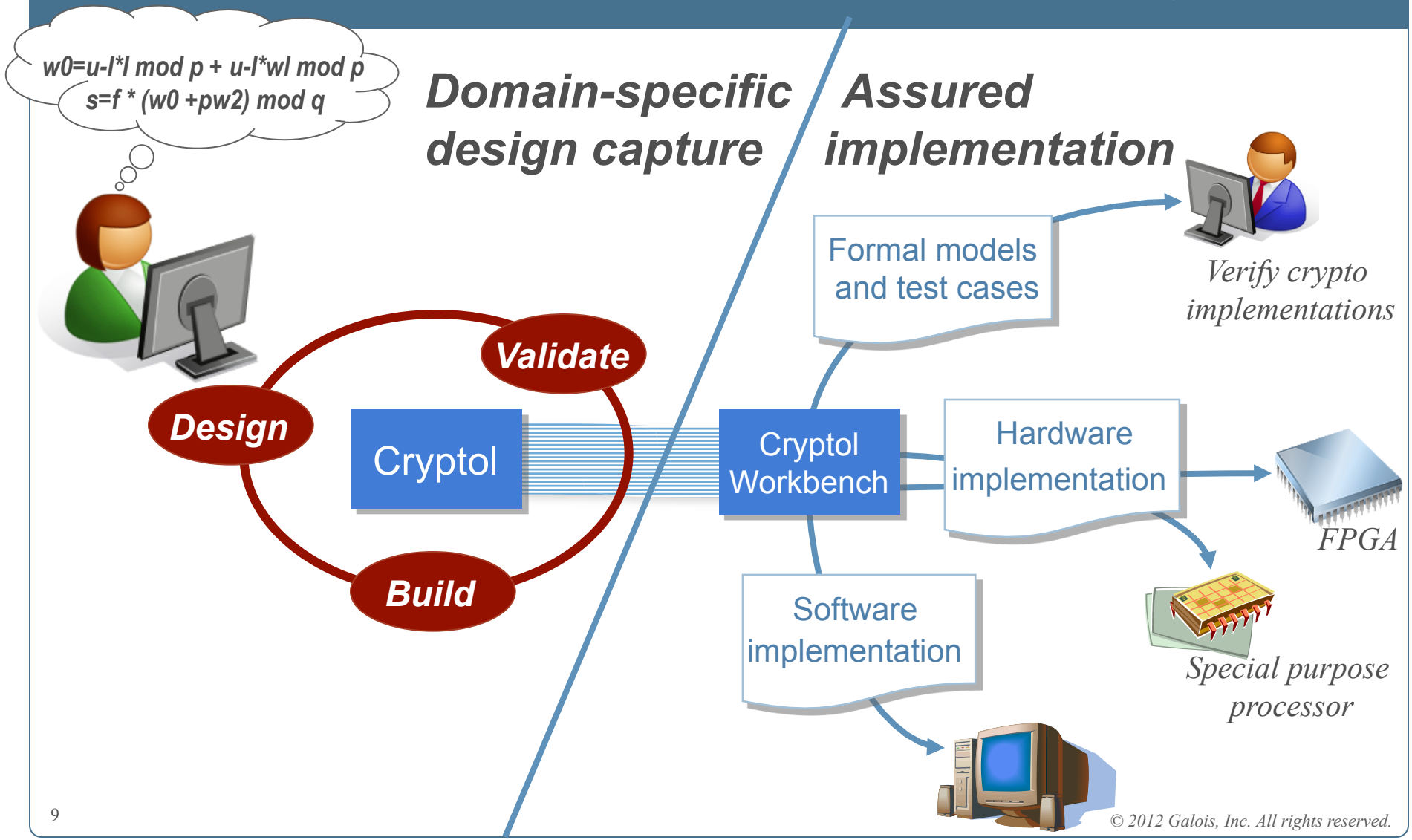
Cryptol: Expressing dataflow dependencies



Cryptol

```
encrypt128 : ([4][32],[4][4][8]) -> [4][4][8];
encrypt128 (initialKey, plainText) = cipherText where {
   roundKeys = [ initialKey ] # [| nextKey (round, prev)
                                 || round <- [1..10]
                                 || prev <- roundKeys
                                 |];
   initialState = first(roundKeys) ^ plainText;
   rounds = [ initialState ] # [| nextState (prev, roundKey, round)
                                || round <- [1..10]
                                || prev <- rounds
                                || roundKey <- drop (1, roundKeys)
                                |];
   cipherText = last(rounds);
};
```

One specification - Many uses



CRYPTOL APPLICATIONS

Cryptol at Work

Cryptol invites high-level exploration of the design space

- Explore the implementation design space at a very high level
- Experiment with several radically different designs in Cryptol in the course of a few hours, covering ground that would take weeks by traditional methods
- Each design can be modeled and characterized quickly

GENERAL DYNAMICS
C4 Systems

ALGORITHM DEVELOPMENT

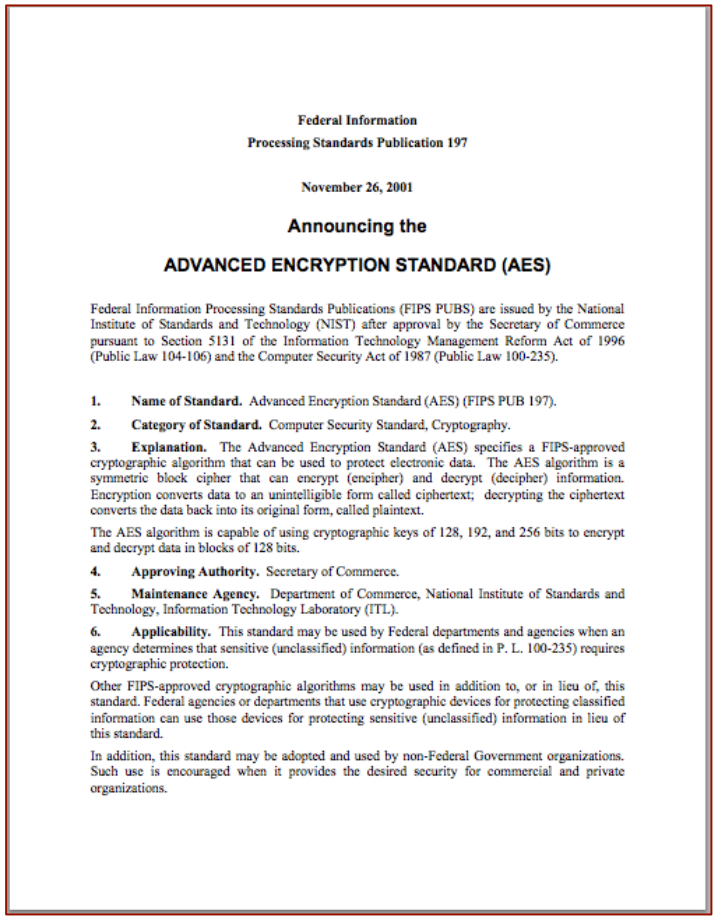
AIM: Advanced INFOSEC Machine

"...an experienced Cryptol programmer, given a new crypto program specification and a soft copy of test vectors, can be expected to learn the algorithm and have a fully functional and verified Cryptol model in a few days to a week."

Alan Newman, GD C4 Systems

What to do?

- You get a specification for an encryption algorithm. It's a mixture of English, pseudo-code, diagrams, and test vectors, and voluminous!
- You need to build the algorithm into a piece of hardware, and more quickly than your competition
- You want to sell the hardware to the government



Federal Information
Processing Standards Publication 197

November 26, 2001

Announcing the

ADVANCED ENCRYPTION STANDARD (AES)

Federal Information Processing Standards Publications (FIPS PUBS) are issued by the National Institute of Standards and Technology (NIST) after approval by the Secretary of Commerce pursuant to Section 5131 of the Information Technology Management Reform Act of 1996 (Public Law 104-106) and the Computer Security Act of 1987 (Public Law 100-235).

1. **Name of Standard.** Advanced Encryption Standard (AES) (FIPS PUB 197).

2. **Category of Standard.** Computer Security Standard, Cryptography.

3. **Explanation.** The Advanced Encryption Standard (AES) specifies a FIPS-approved cryptographic algorithm that can be used to protect electronic data. The AES algorithm is a symmetric block cipher that can encrypt (encipher) and decrypt (decipher) information. Encryption converts data to an unintelligible form called ciphertext; decrypting the ciphertext converts the data back into its original form, called plaintext.

The AES algorithm is capable of using cryptographic keys of 128, 192, and 256 bits to encrypt and decrypt data in blocks of 128 bits.

4. **Approving Authority.** Secretary of Commerce.

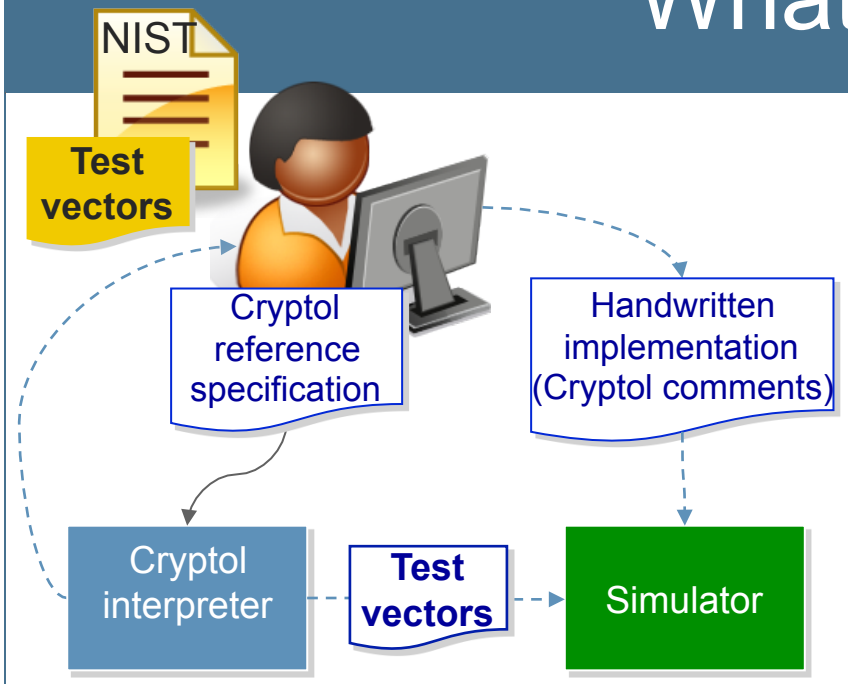
5. **Maintenance Agency.** Department of Commerce, National Institute of Standards and Technology, Information Technology Laboratory (ITL).

6. **Applicability.** This standard may be used by Federal departments and agencies when an agency determines that sensitive (unclassified) information (as defined in P. L. 100-235) requires cryptographic protection.

Other FIPS-approved cryptographic algorithms may be used in addition to, or in lieu of, this standard. Federal agencies or departments that use cryptographic devices for protecting classified information can use those devices for protecting sensitive (unclassified) information in lieu of this standard.

In addition, this standard may be adopted and used by non-Federal Government organizations. Such use is encouraged when it provides the desired security for commercial and private organizations.

What GD C4 Systems does



- The specification is translated into a fully executable, unambiguous notation
- Cryptol fragments annotate the AIM micro sequencer code, greatly increasing the readability of the extremely dense assembly language
- Component testing, from small snippets through major subroutines, is greatly facilitated with Cryptol-generated test vectors derived from the end-to-end test vectors provided in algorithm source specifications
- The Cryptol models directly support the certification effort

Rockwell Collins

A RESEARCH EXPERIMENT

A High Speed Encryptor

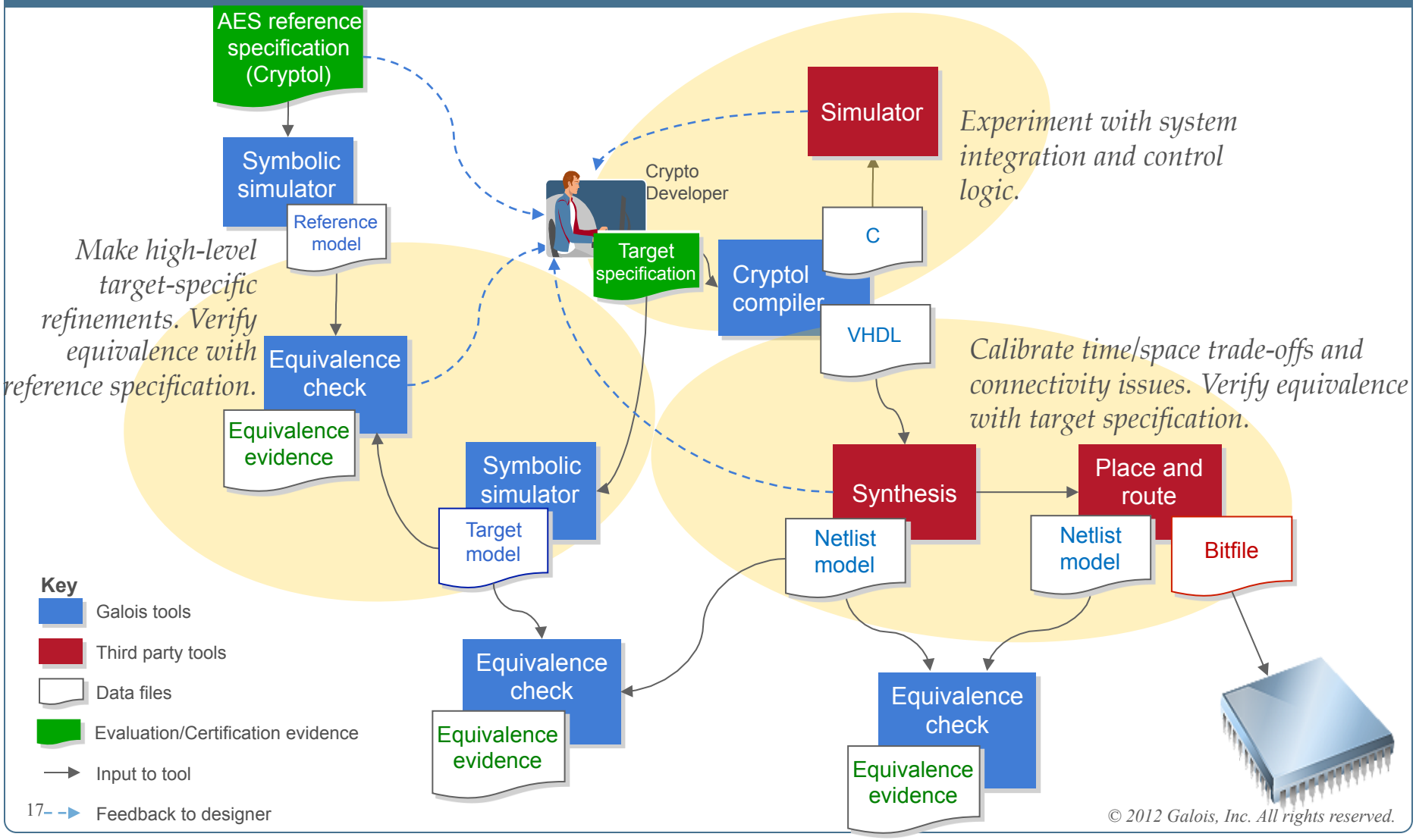
"…the Rockwell Collins/Galois team was able to design, implement, simulate, integrate, analyze, and test a complex CEA on the new hardware, including AES-256 and Galois Counter Mode (GCM), in less than 3 months."

What to do?

- You want to quickly and cheaply build a research prototype to strengthen your proposal to participate in a research program
- You need some encryption in the FPGA prototype, but you aren't sure how fast it needs to be or how much space will be available
- You want to simulate the complete system before you build it to nail down the requirements for the crypto

galois

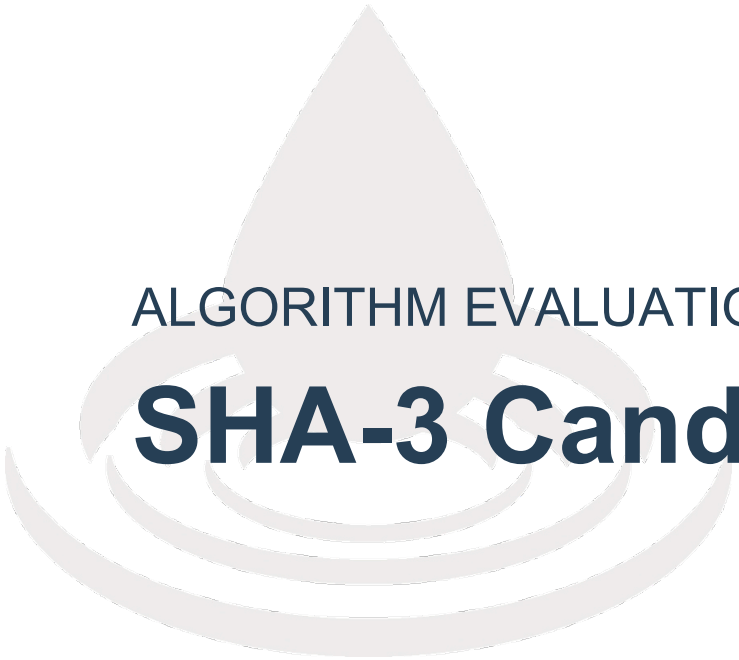
Producing a family of designs



Experimental results

- High-level exploration of the design space yields huge benefits
  - A Cryptol developer can experiment with multiple designs in a short amount of time
  - Detailed implementation refinements can be effectively modeled in the high-level specification
    - A heavily pipelined AES written in Cryptol outperforms commercial cores
- System integration issues are important
  - In HSE, space considerations were more important than performance

AES	Resource Utilization			Performance		
Optimization	LUTs	FF	Block RAM	Latency	Clockrate (MHz)	Throughput (Gbps)
Small size	1124	461	10	12	144	1.67
High performance	6336	12576	100	53	385	47.4



ALGORITHM EVALUATION

SHA-3 Candidates

What to do?

- You are developing a tool to build models in the form of AIG graphs for crypto circuits described in VHDL
- None of your colleagues design crypto circuits in VHDL
- You need to find a source for VHDL designs for crypto algorithms

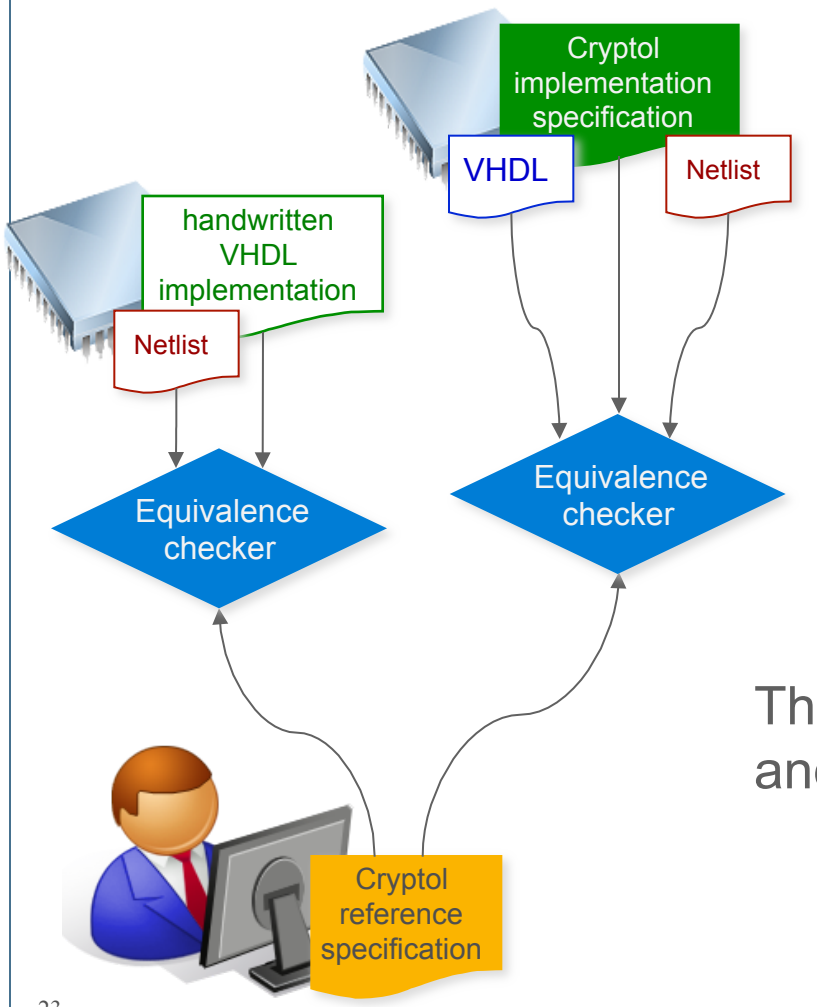
The SHA-3 competition

- “NIST has opened a public competition to develop a new cryptographic hash algorithm, which converts a variable length message into a short “message digest” that can be used for digital signatures, message authentication and other applications.”
- 51 original submissions
- 5 finalists

The SHA-3 finalists

- Cryptol specifications are available for all 5 finalists
- Galois wrote specs for three of the finalists
  - Skein
    - We have verified two third-party VHDL implementation
  - Blake
    - Verification of a third-party VHDL implementation is described in a downloadable tutorial
  - JH
- Two students at U.Minho in Portugal wrote a Cryptol specification for Grøstl
  - They also generated a respectable FGPA implementation and verified it against the Cryptol specification
- A client provided us with a Cryptol spec for Keccak

Cryptol in the evaluation process



A crypto-device evaluator:

- Creates a reference specification and associated formal model
- Checks the equivalence of the implementation models with the specification models

The process works for both hand-written and Cryptol-generated designs

The verification process

1. Develop a specification
2. Understand the implementation
3. Import the VHDL into Cryptol
4. Match up the inputs and outputs
5. Create AIG representations of both circuits and reduce one to the other

Verifying two implementations of Skein

- Partial implementation by Men Long in VHDL
  - **Bug found!** Men Long's concise cyclic rotation was interpreted differently by GHDL, Simili and Xilinx.
  - **Resolution:** Replaced by call to standard library function.
- Full implementation by Stefan Tillich
  - http://www.iaik.tugraz.at/content/research/

	Cryptol spec AIG	VHDL implementation AIG	Verification time
Men Long	118,156 nodes	653,963 nodes	~ 1 hr
Stefan Tillich	301,342 nodes	900,496 nodes	~ 17.5 hrs

CRYPTOL USE CASE

Finding van de Waerden numbers

What to do?

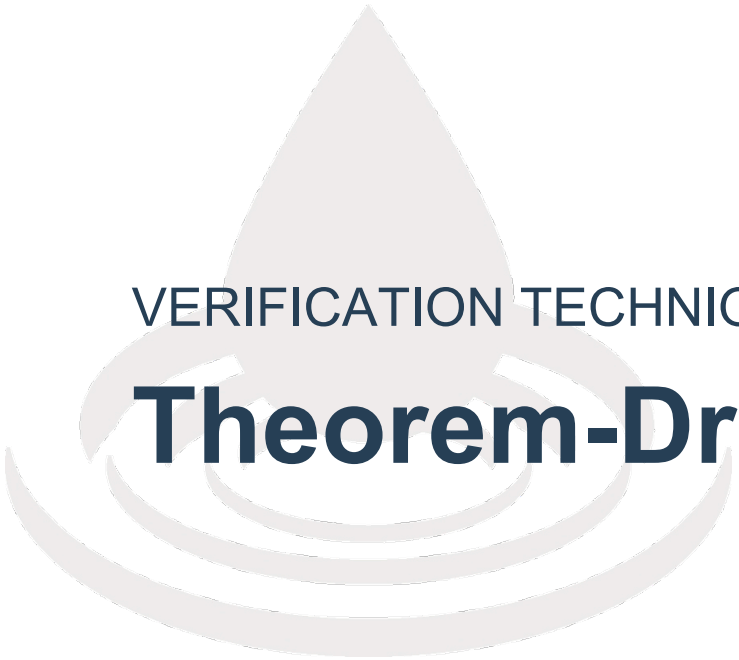
- You think you found a van der Waerden number
  - For all positive integers r and k there exists a positive integer N such that if the integers {1 2 ... N} are colored, each with one of r different colors, then there are at least k integers in arithmetic progression all of the same color. For any r and k, the smallest such N is the van der Waerden number W(r,k).
- Van de Waerden numbers are difficult to compute; the last discovery was more than 30 years ago.* How can you be sure you found one?

* W(2,5)=178. In 2007, Dr. Michal Kouril of the University of Cincinnati established that W(2,6)=1132

Gaining confidence in an implementation

- Kouril employed a special SAT-solver and clever techniques to bound the search and programmed FPGAs to speed up the search.
- To convince himself that the FPGA ensemble was doing what he expected, he:
  - Wrote a Cryptol specification for the algorithm running in the FPGA ensemble
  - Generated formal models for both the Cryptol specification and his VHDL implementation
  - Verified that the two were equivalent

Searching a large key space might employ similar techniques



VERIFICATION TECHNIQUES

Theorem-Driven Analysis

Verification using formal methods

- Formal Methods is a body of verification techniques that work by building a mathematical model of an artifact and proving properties about it
- Formal methods are complementary to testing
  - Testing techniques generate weak evidence about the real artifact [Worry: Have I tested enough?]
  - Formal methods generate strong evidence about a model of the artifact [Worry: Is the model faithful enough?]

Early Reference:

Alan M. Turing. Checking a large routine. In Report of a Conference on High Speed Automatic Calculating Machines, pages 67–69, Cambridge, England, June 1949. University Mathematical Laboratory.

Equivalence checking

- Given two Cryptol functions *f*, *g*
  - Either prove they agree on all inputs:
    - ∀*x*. *f x* == *g x*
  - Or, provide a counter example *x* such that
    - *f x* != *g x*
- Typical use-case:
  - *f*: Specification, written for clarity
  - *g*: Implementation, optimized for speed/space on the target platform

Simple equivalence checking example

```
f, g, h : [64] -> [64];
f x = 2*x;
g x = x << 1;
h x = x <<< 1;
```

```
Cryptol> :eq f g
True
```

```
Cryptol> :eq f h
False
f 0xffffffffffffffff
    = 0xfffffffffffffffe
h 0xffffffffffffffff
    = 0xffffffffffffffff
```

Property verification

- Equivalence checking shows functional equivalence
  - The input/output behaviors are “precisely the same”
- Property verification goes further
  - Allows “correctness” properties to be specified and proved automatically
- Counter-examples are extremely useful for debugging
  - or deriving concrete exploits
- **Example:** For all values of key and plaintext, encryption followed by the decryption returns the plaintext:

```
theorem encDec: {key pt}.
  dec (key, enc(key, pt)) == pt;
```

Theorem declarations

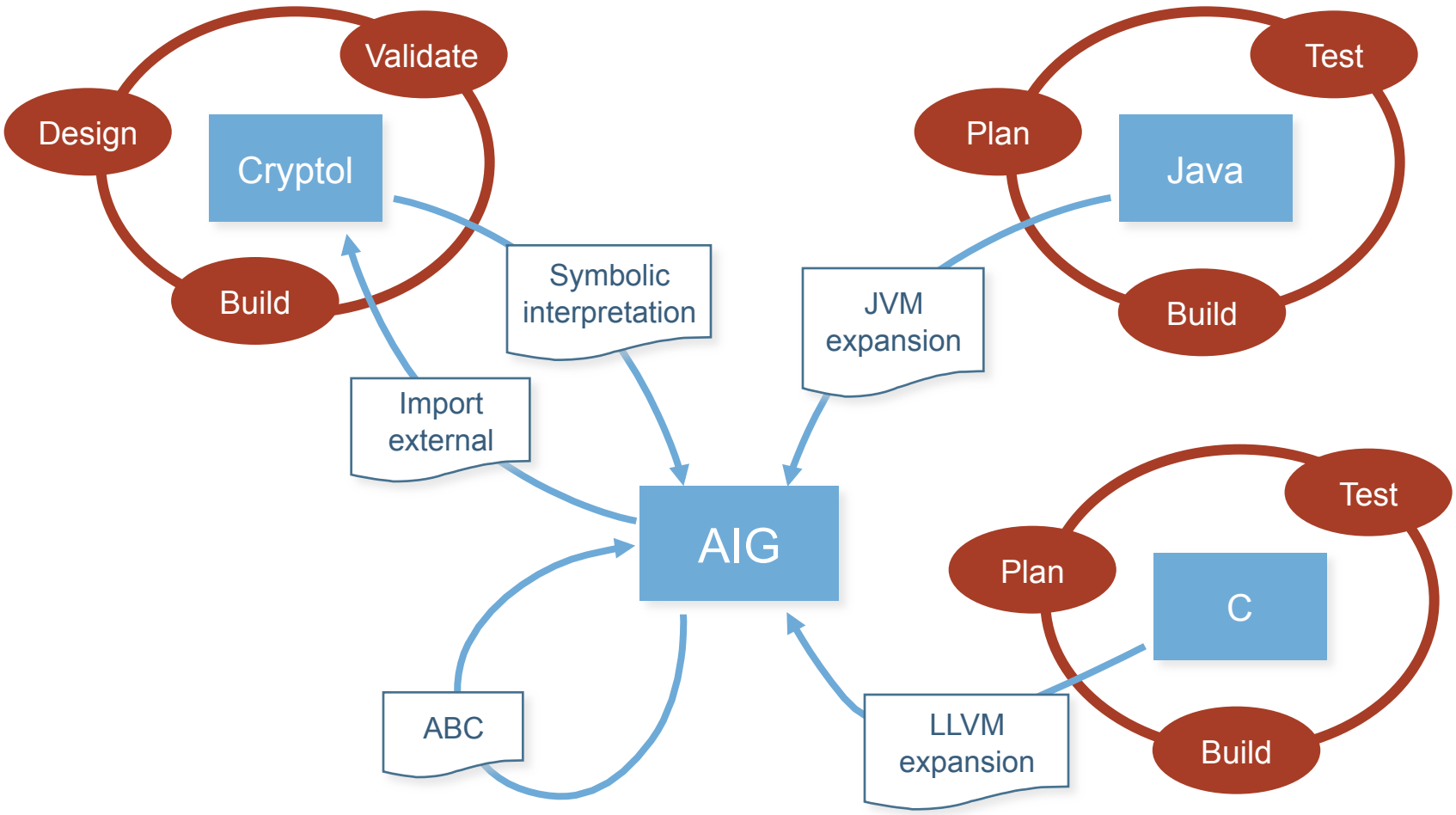
- To prove a sort function correct, we need to show:
  1. Output is in non-decreasing order
  2. Output is a permutation of the input
- Define these conditions as “predicates” in Cryptol
  - Predicates are just functions returning true/false
- Write a “theorem declaration” that captures correctness:

```
theorem sortIsCorrect: {xs}.
    nonDecreasing(ys) & isPermutationOf(xs, ys)
  where ys = sort(xs);
```

Proving theorem declarations

- "Quickcheck" property-based testing
  - User gives a property, Cryptol automatically tests it on random inputs.
- Use of SMT-based property checkers
  - SAT: Checks for satisfiability of large Boolean formulas
  - SMT extends SAT with higher-level constraint solvers (linear arithmetic, arrays, functions, etc.)
- Semi-automatic theorem proving
  - Translator from Cryptol to Isabelle theorem prover
  - User can specify arbitrary Cryptol properties, but proof may need human guidance

AIG forms a common interchange



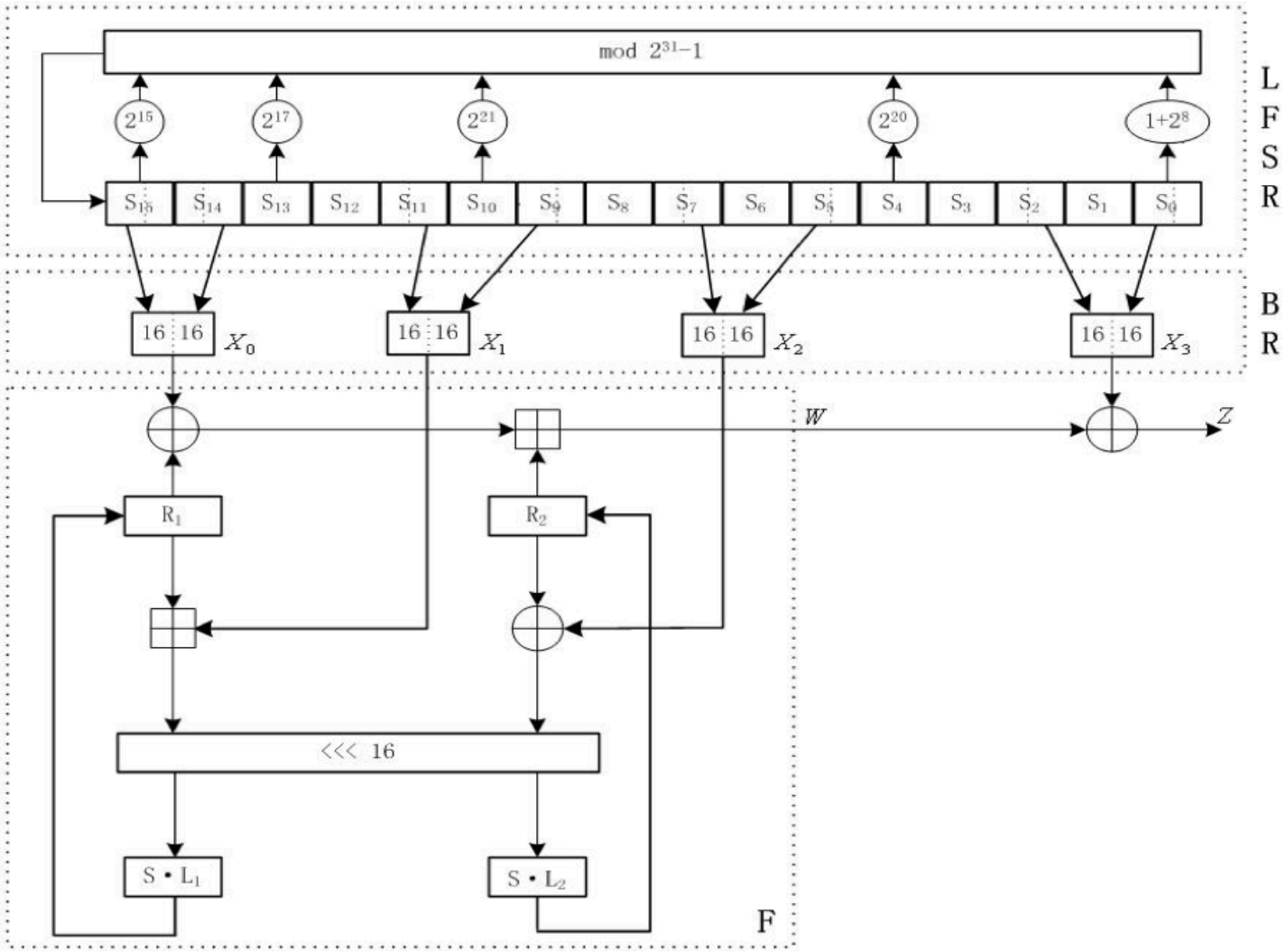
CRYPTOL FOR CRYPTANALYSIS

Demo

ZUC stream cipher

- Initiative by 3GPP for securing mobile networks
- Word-oriented stream cipher
  - 128-bit key, 128-bit initialization vector
  - Generates keystream of 32-bit words
- Forms the heart of two 3GPP crypto algorithms:
  - The 128-EEA3 confidentiality algorithm
  - The 128-EIA3 integrity algorithm
- A severe vulnerability was discovered in ZUC version 1.4
  - “ZUC initialization process does not preserve key entropy”
  - Led to a chosen IV attack [Wu et. al., ASIACRYPT2010]

ZUC design



Summary

- Cryptol is the *language of cryptanalysis.*
- Cryptol language constructs are useful for expressing cryptanalysis algorithms.
- Different back-ends support efficient cryptanalysis on a variety of hardware/software platforms.
- The verification toolset can be used to probe cryptographic algorithms for weaknesses.

Questions?

- For more information on Cryptol visit

  http://www.cryptol.net

- Free download of the interpreter with QuickCheck
- Also comes with language manuals and toolset tutorials
- Evaluation license available for FPGA back-end and verification tools

SUPPORTING MATERIAL

Backup Slides

About Galois, Inc.

- high assurance research and development
- Creating trustworthiness in critical systems

Galois [gal-wah]

Named after French mathematician Évariste Galois

| galois |

Company facts