

Cryptol을 이용한 국내 표준 블록 암호 ARIA 모듈의 자동 정형 검증

(Automated Formal Verification of Korean Standard Block Cipher ARIA using Cryptol)

Security Analysis aNd Evaluation Lab, CIST

최원빈 (Wonbin Choi)

Prof. Dr. Seungjoo (Gabriel) Kim



- 1. Cryptol 개요**
- 2. ARIA 구현 및 검증 프로세스**
- 3. 진행중인 연구**
- 4. Cryptol 환경 구축**
- 5. ARIA 구현 및 검증**

- **What?**

- **Galois, Inc.**
- **High Assurance, Retargetable Crypto Development and Validation**



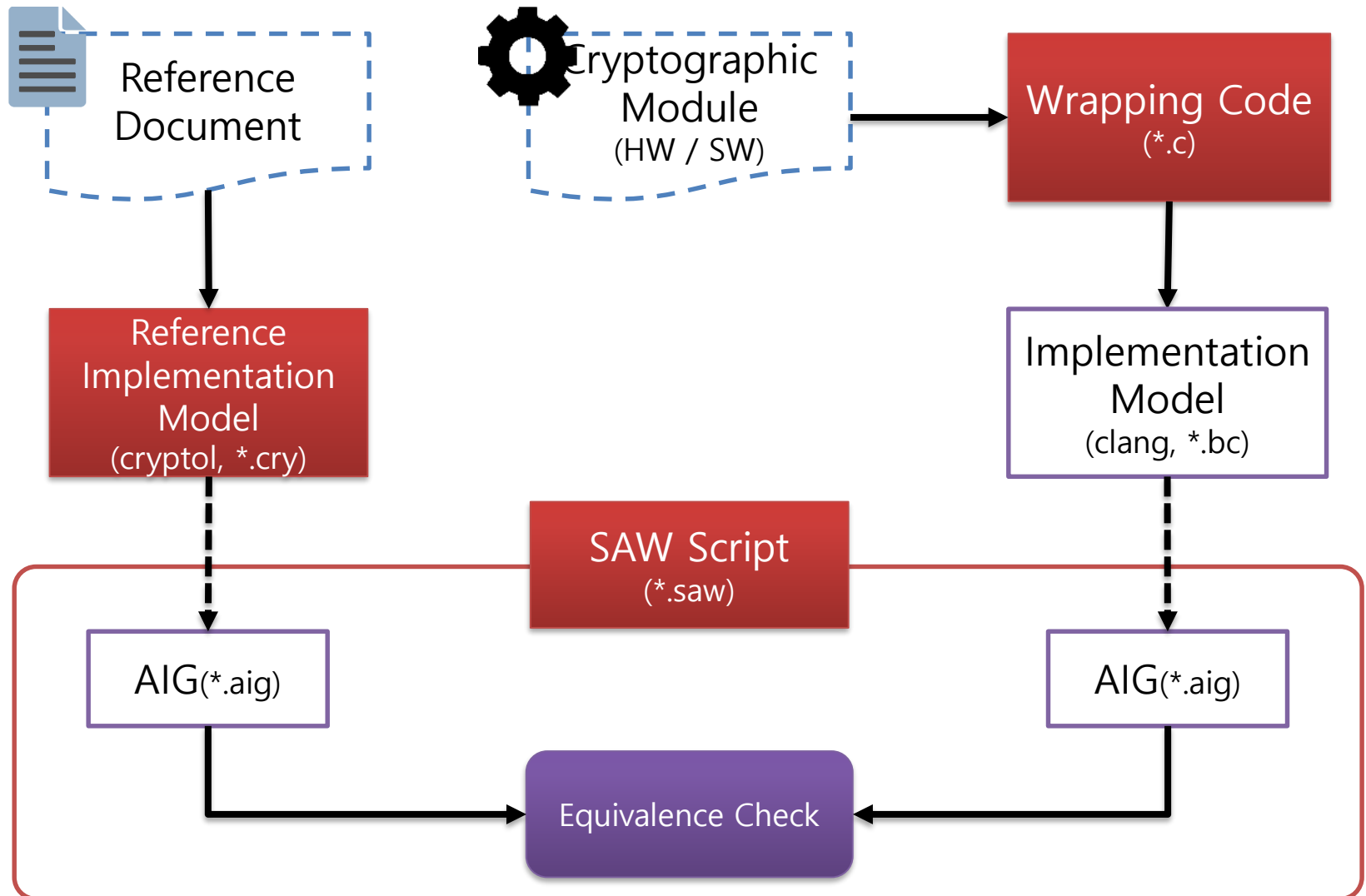
■ What?

- 암호에 대한 **formal-method** 기반 접근
- Galois에서 NSA의 암호 전문가와 설계한 **high-level specification language**
- 하드웨어 플랫폼과 독립적(**independent**)으로 **directly** 및 **formally**하게 표현함
- Haskell 기반의 **Domain-specific language**

■ Why?

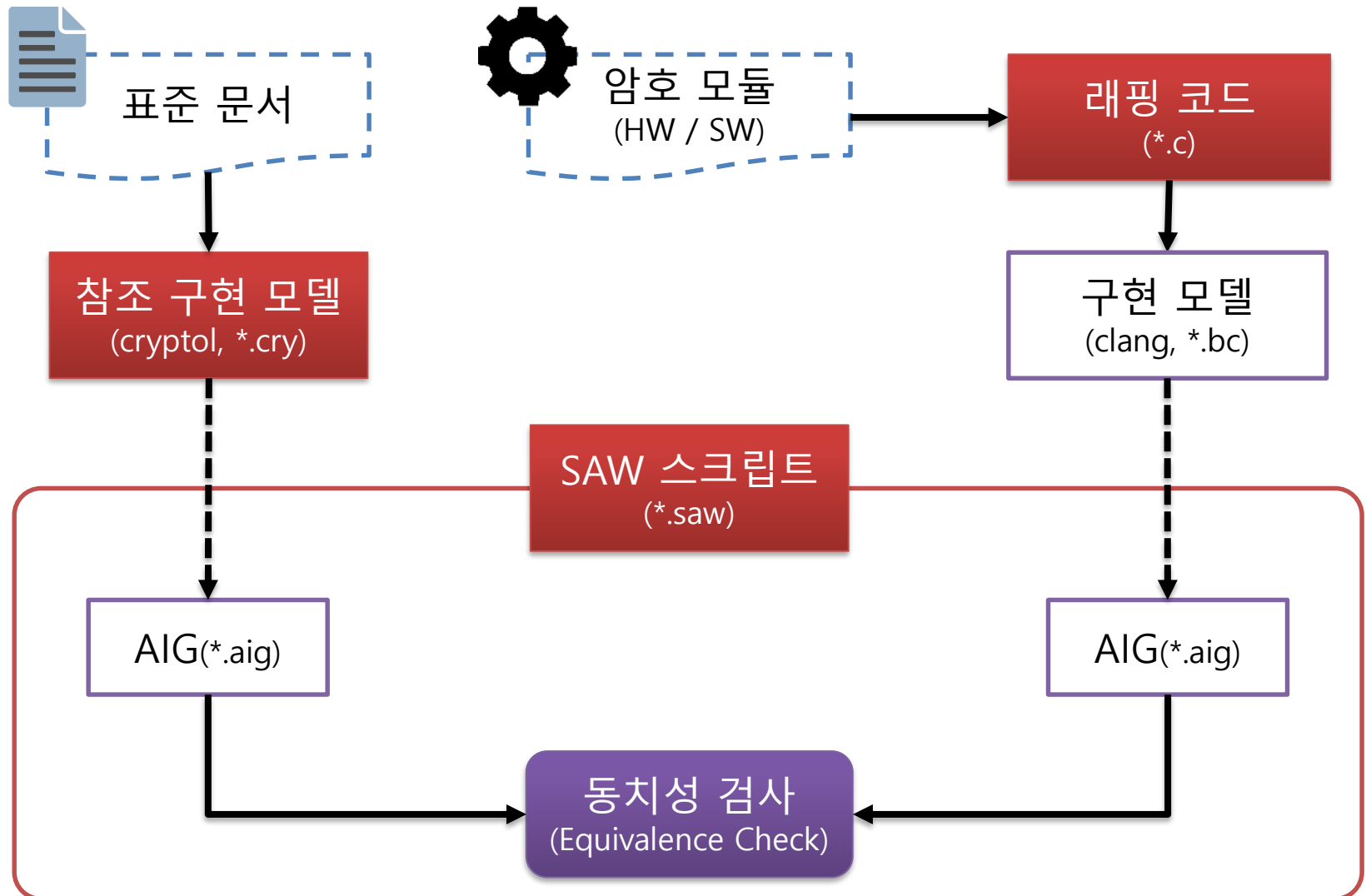
- 기존 암호 모듈 검증은 테스트 벡터 기반
 - KAT(Known Answer Test), MCT(Monte-Carlo Test), MMT(Multi-block Message Test)와 같은 방법이 존재
 - ARIA 알고리즘은 한국인터넷진흥원(KISA)에서 테스트 벡터가 제공되고 있음
- 2^{128} 만큼 전수테스트를 수행하는 것은 현실적으로 불가능
 - AES 알고리즘의 MCT는 NIST에서 암호화 10만회를 기준으로 함
 $100 \text{ multi_block_messages} \times 1,000 \text{ block}$

■ Implementation and verification architecture



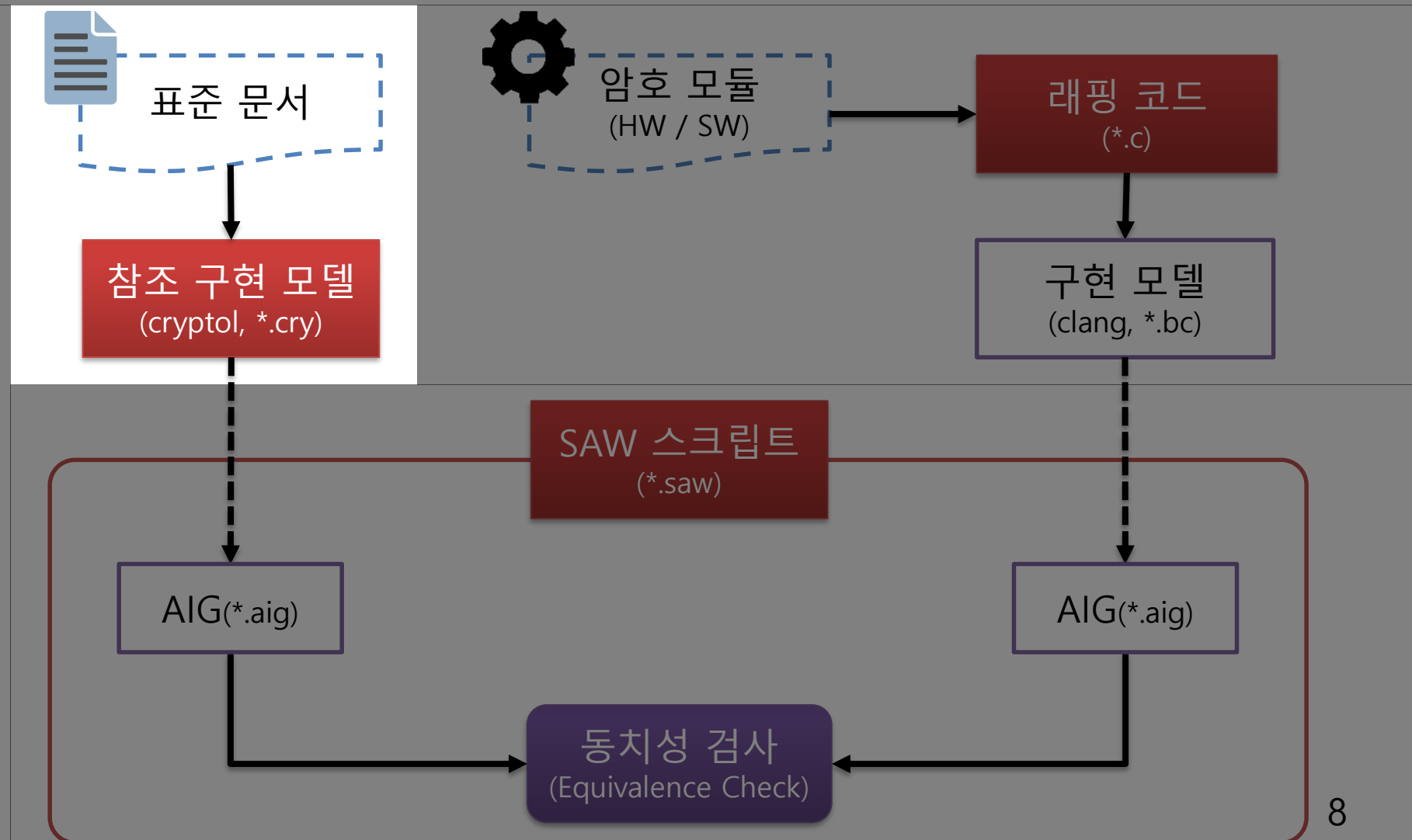
2. ARIA 구현 및 검증 프로세스

■ 구현 및 검증 아키텍처



2. ARIA 구현 및 검증 프로세스

■ 구현 및 검증 아키텍처




2. ARIA 구현 및 검증 프로세스

■ 검증 프로세스(참조 구현 모델 - Cryptol)

5.3.2 라운드 키 생성 과정

$$ek_1 = (W_0) \oplus (W_1 \ggg 19)$$

W_1, W_2, W_3 를 조합하여 암호화 라운드
트 수는 암호키의 크기가 128, 192, 256
라운드  덧셈 계층이 두 번 있으
다. 암호화 라운드 키는 다음과 같다.

$$ek_1 = (W_0) \oplus (W_1 \ggg 19),$$

$$ek_3 = (W_2) \oplus (W_3 \ggg 19),$$

$$ek_5 = (W_0) \oplus (W_1 \ggg 31),$$

$$ek_7 = (W_2) \oplus (W_3 \ggg 31),$$

$$ek_9 = (W_0) \oplus (W_1 \lll 61),$$

$$ek_{11} = (W_2) \oplus (W_3 \lll 61),$$

$$ek_{13} = (W_0) \oplus (W_1 \lll 31),$$

$$ek_{15} = (W_2) \oplus (W_3 \lll 31),$$

$$ek_{17} = (W_0) \oplus (W_1 \ggg 19)$$

위에서 설명한 키 확장 과정을 정리하면 (그림
여기서 ROT()는 각 라운드별로 주어진 순환량에

$$ek_1 = (w_0) \wedge (w_1 \ggg 19)$$

```
파일(F) 편집(E) 보기(V) 삽입(I) 삭제(D) 실행(R) 종료(C)
generateEncKey : [128] -> [13][128]
generateEncKey masterKey = [ek1, ek2, ek3, ek4, ek5, ek6, ek7, ek8, ek9, ek10, e
k11, ek12, ek13]
where
  w0 = masterKey
  w1 = oddFunction(w0, ck @ 0) ^ zero : [128]
  w2 = evenFunction(w1, ck @ 1) ^ w0
  w3 = oddFunction(w2, ck @ 2) ^ w1
  ek1 = (w0) ^ (w1 >>> 19)
  ek2 = (w1) ^ (w2 >>> 19)
  ek3 = (w2) ^ (w3 >>> 19)
  ek4 = (w0 >>> 19) ^ (w3)
  ek5 = (w0) ^ (w1 >>> 31)
  ek6 = (w1) ^ (w2 >>> 31)
  ek7 = (w2) ^ (w3 >>> 31)
  ek8 = (w0 >>> 31) ^ (w3)
  ek9 = (w0) ^ (w1 <<< 61)
  ek10 = (w1) ^ (w2 <<< 61)
  ek11 = (w2) ^ (w3 <<< 61)
  ek12 = (w0 <<< 61) ^ (w3)
  ek13 = (w0) ^ (w1 <<< 31)
```

■ 구현 및 검증 아키텍처



2. ARIA 구현 및 검증 프로세스

- 검증 프로세스(암호 모듈 - C)
 - Cryptography Module
 - KISA 32bit/8bit ARIA Module
 - hi-hee, minjeongJho ARIA Module in Github

```
#include <stdio.h>

void Crypt(const unsigned char *p, int R, const unsigned char *e, unsigned char *c);
int EncKeySetup(const unsigned char *w0, unsigned char *e, int keyBits);
int DecKeySetup(const unsigned char *w0, unsigned char *d, int keyBits);
```

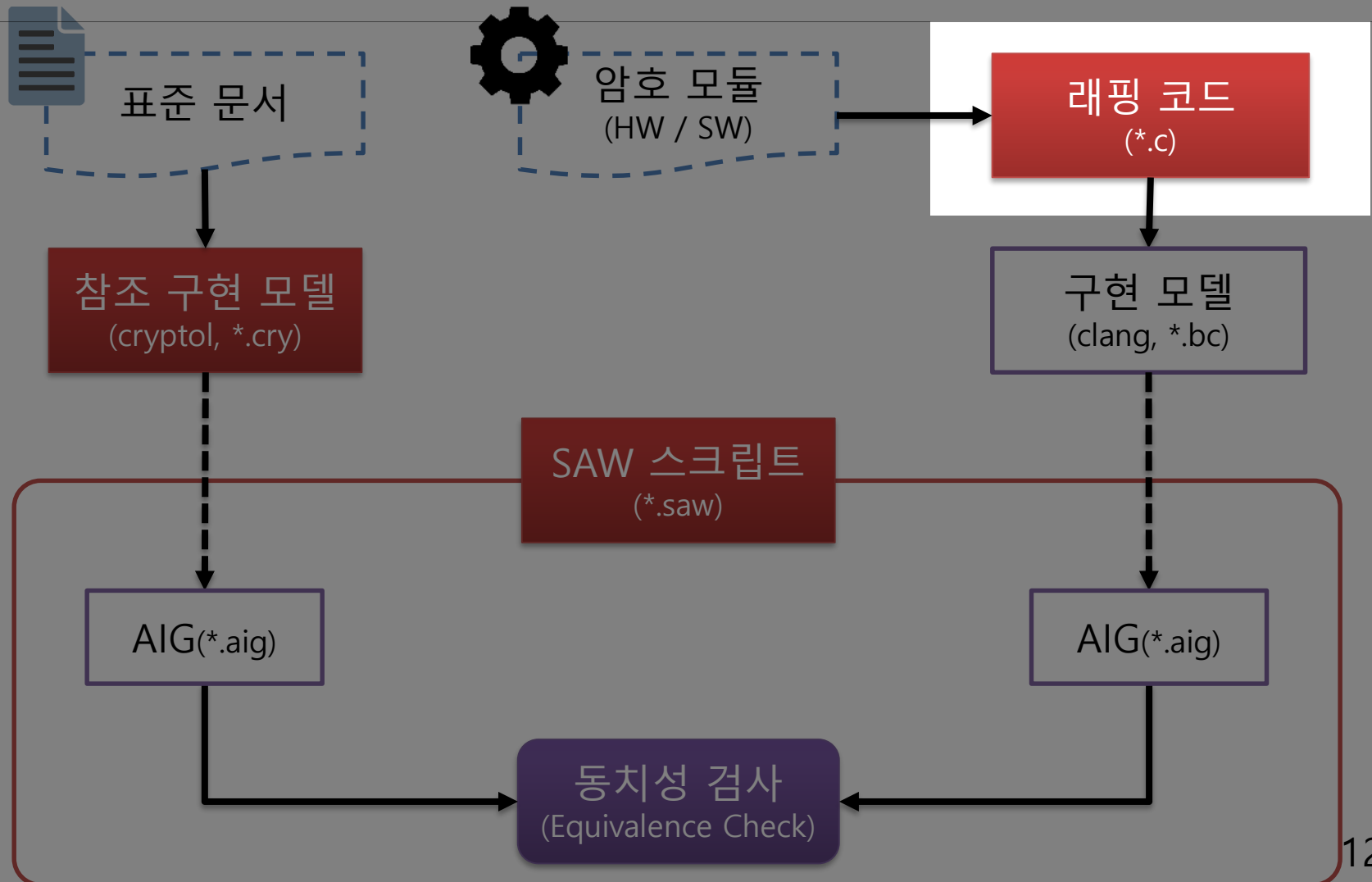
```
/* 암호화 함수.
 * const Byte *i: 입력
 * int Nr: 라운드 수
 * const Byte *rk: 라운드 키들
 * Byte *o: 출력
 */
void Crypt(const Byte *i, int Nr, const Byte *rk, Byte *o) {
    register Word t0, t1, t2, t3;

    WordLoad(WO(i,0), t0); WordLoad(WO(i,1), t1);
    WordLoad(WO(i,2), t2); WordLoad(WO(i,3), t3);

    if (Nr > 12) {KXL FO KXL FE}
    if (Nr > 14) {KXL FO KXL FE}
    KXL FO KXL FE KXL FO KXL FE KXL FO KXL FE
    KXL FO KXL FE KXL FO KXL FE KXL FO KXL
```

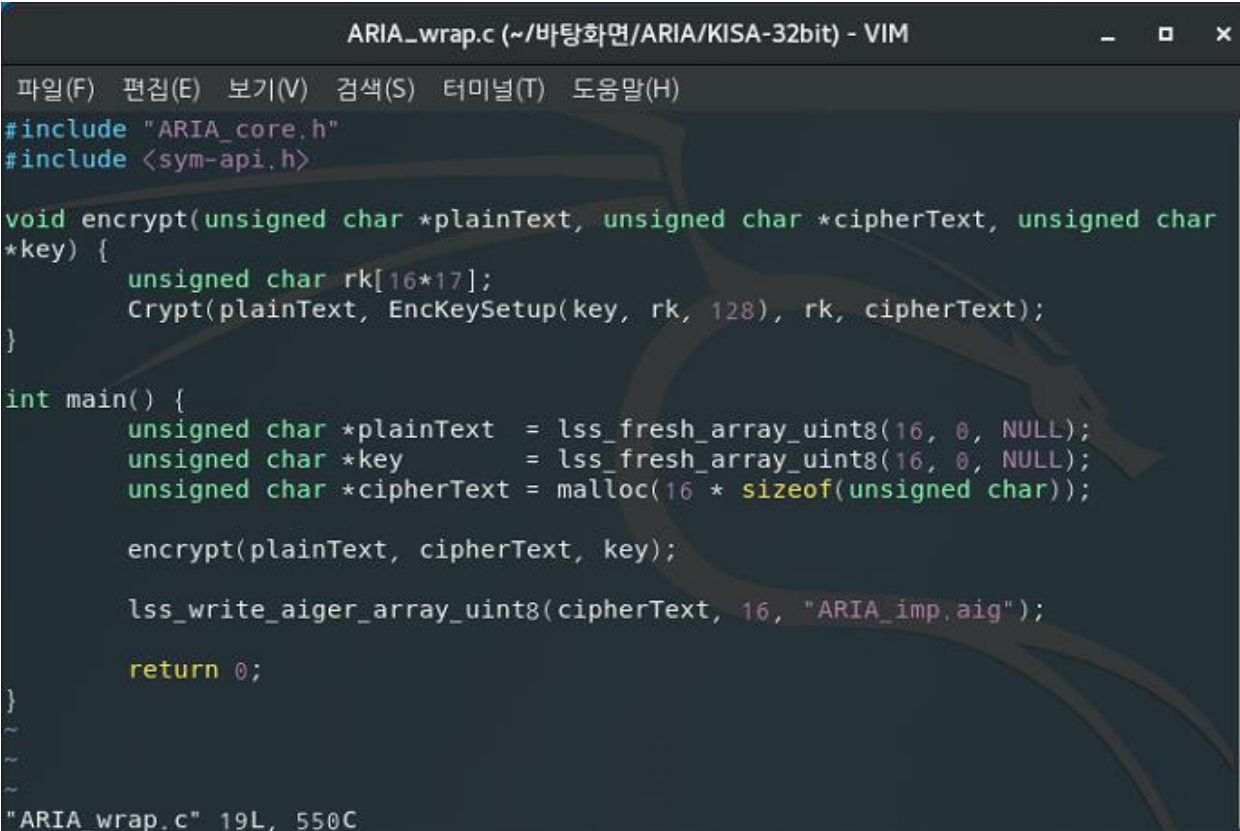
2. ARIA 구현 및 검증 프로세스

■ 구현 및 검증 아키텍처



2. ARIA 구현 및 검증 프로세스

- 검증 프로세스(래핑 코드 - C)
 - Wrap Code
 - LSS(LLVM Symbolic Simulator) - "sym-api.h"



```
ARIA_wrap.c (~/바탕화면/ARIA/KISA-32bit) - VIM
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
#include "ARIA_core.h"
#include <sym-api.h>

void encrypt(unsigned char *plainText, unsigned char *cipherText, unsigned char
*key) {
    unsigned char rk[16*17];
    Crypt(plainText, EncKeySetup(key, rk, 128), rk, cipherText);
}

int main() {
    unsigned char *plainText = lss_fresh_array_uint8(16, 0, NULL);
    unsigned char *key = lss_fresh_array_uint8(16, 0, NULL);
    unsigned char *cipherText = malloc(16 * sizeof(unsigned char));

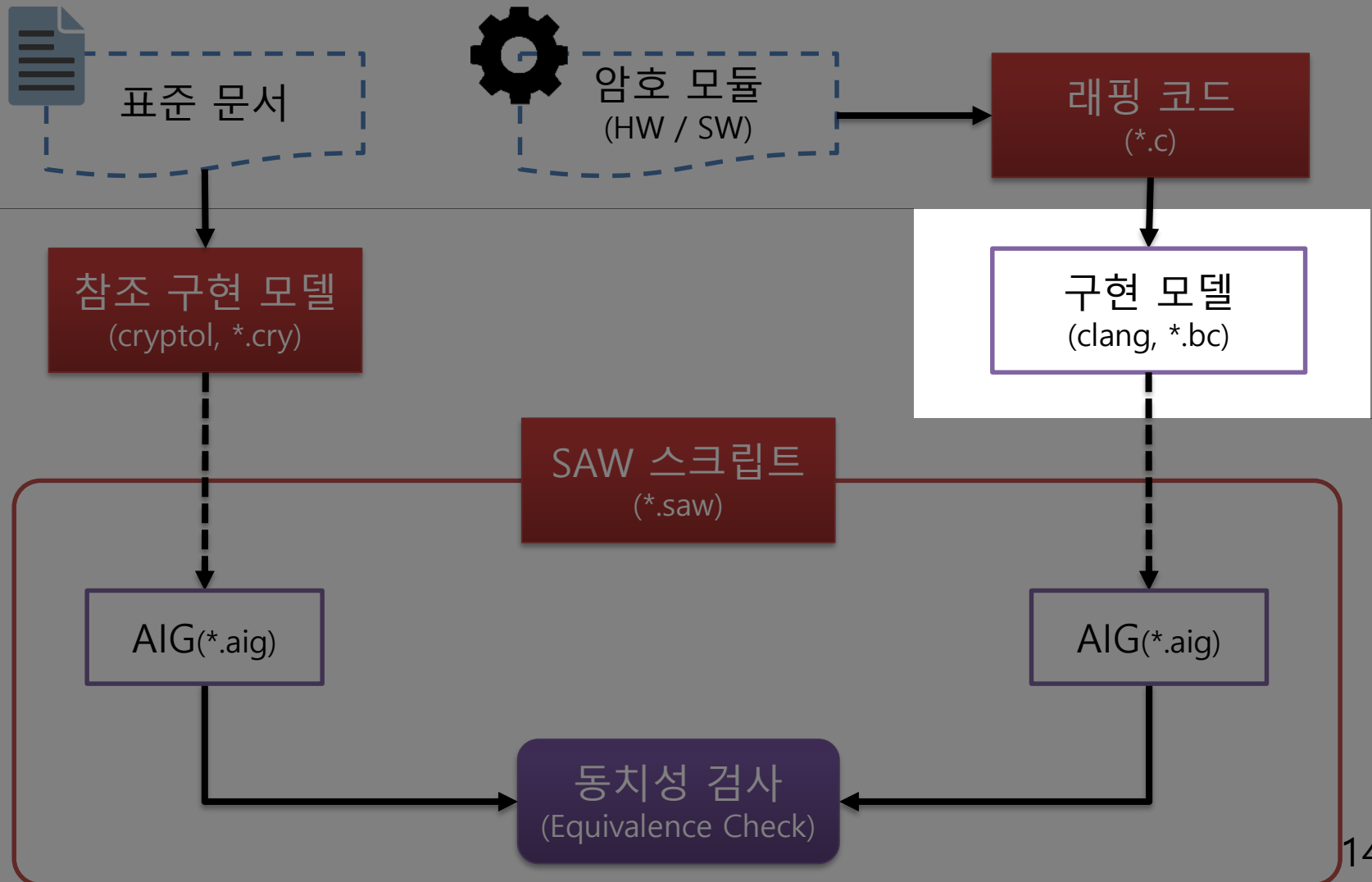
    encrypt(plainText, cipherText, key);

    lss_write_aiger_array_uint8(cipherText, 16, "ARIA_imp.aig");

    return 0;
}
~
~
~
"ARIA_wrap.c" 19L, 550C
```

2. ARIA 구현 및 검증 프로세스

■ 구현 및 검증 아키텍처

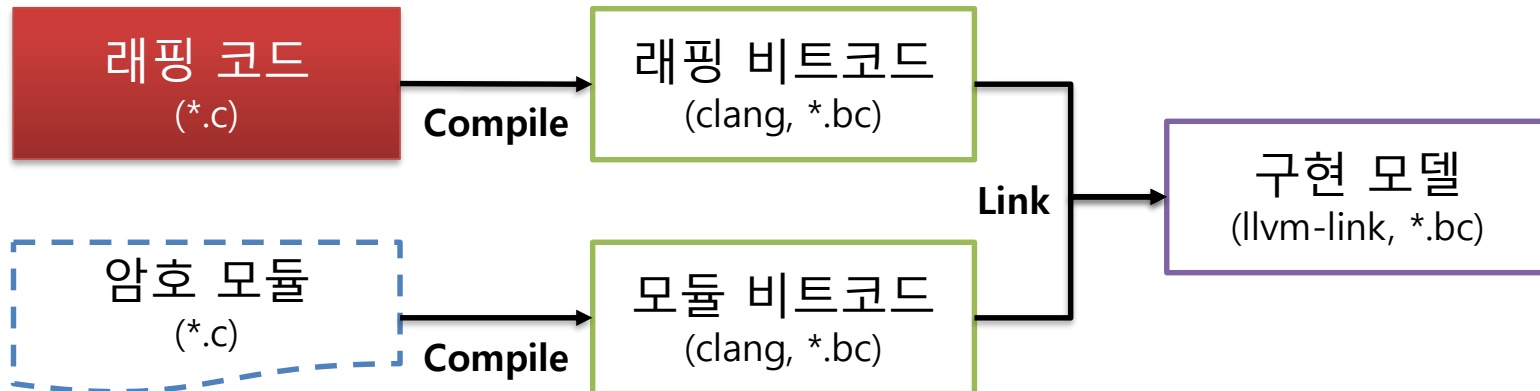


2. ARIA 구현 및 검증 프로세스

■ 검증 프로세스(구현 모델 - Clang)

■ Implementation Model

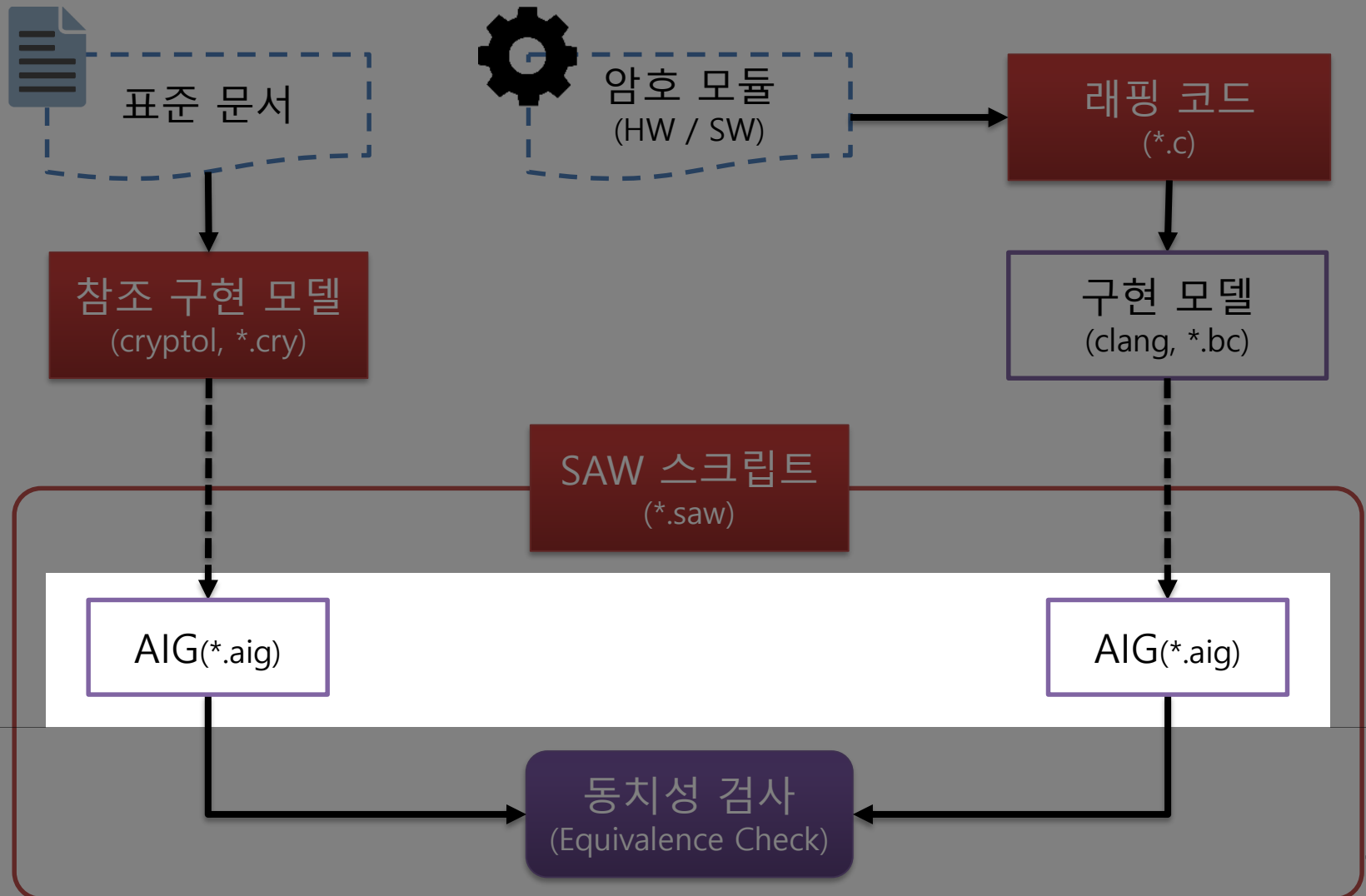
1. compile ALL C PROGRAM using CLANG(LLVM)
2. link ALL BIT CODE using LLVM-LINK



```
bindon@Creator: ~/바탕화면/ARIA/KISA-32bit
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
bindon@Creator:~/바탕화면/ARIA/KISA-32bit$ clang -c -emit-llvm -I/home/bindon/ll
vm-verifier/sym-api -o ARIA_wrap.bc ARIA_wrap.c
bindon@Creator:~/바탕화면/ARIA/KISA-32bit$ clang -c -emit-llvm -I/home/bindon/ll
vm-verifier/sym-api -o ARIA_core.bc ARIA_core.c
bindon@Creator:~/바탕화면/ARIA/KISA-32bit$ llvm-link -o ARIA_imp.bc ARIA_wrap.bc
ARIA_core.bc
bindon@Creator:~/바탕화면/ARIA/KISA-32bit$
```

2. ARIA 구현 및 검증 프로세스

■ 구현 및 검증 아키텍처

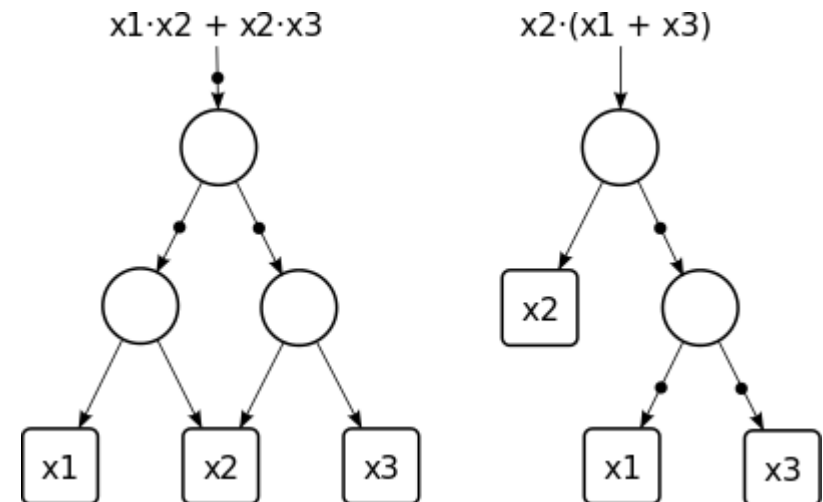
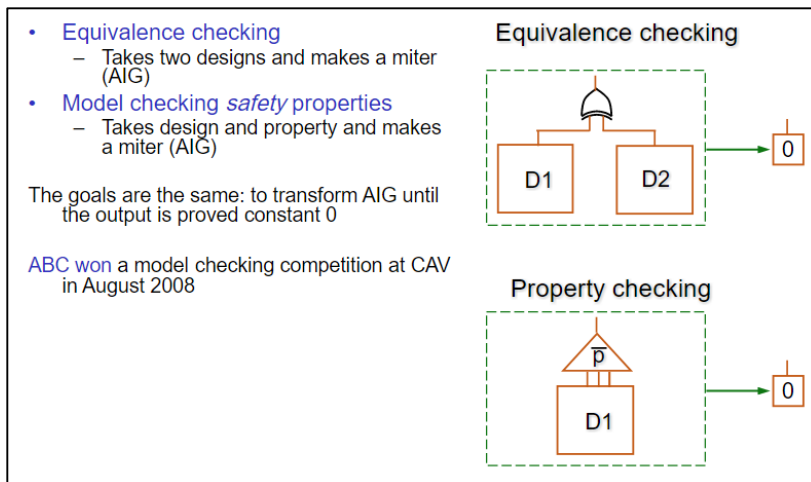


■ 검증 프로세스(AIG)

■ AIG(And-Inverter Graph)

■ Property Checking 및 Equivalence Checking을 위해 AIG 형태의 파일을 생성해야 함

- Brayton, Robert, and Alan Mishchenko. "ABC: An academic industrial-strength verification tool." Computer Aided Verification. Springer Berlin/Heidelberg, 2010.
- Biere, Armin. "The AIGER And-Inverter Graph (AIG) Format Version 20071012." (2007).



2. ARIA 구현 및 검증 프로세스

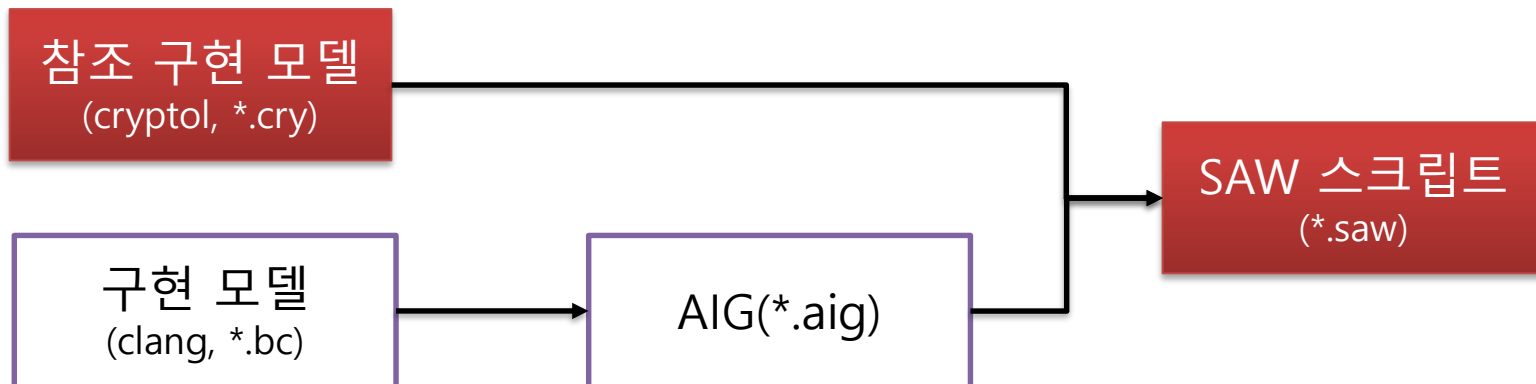
■ 검증 프로세스(AIG)

■ 참조 구현 모델의 AIG 생성

- 따로 생성하지 않아도 **Cryptol**을 사용하여 작성하였으면 추가적인 작업이 필요하지 않음

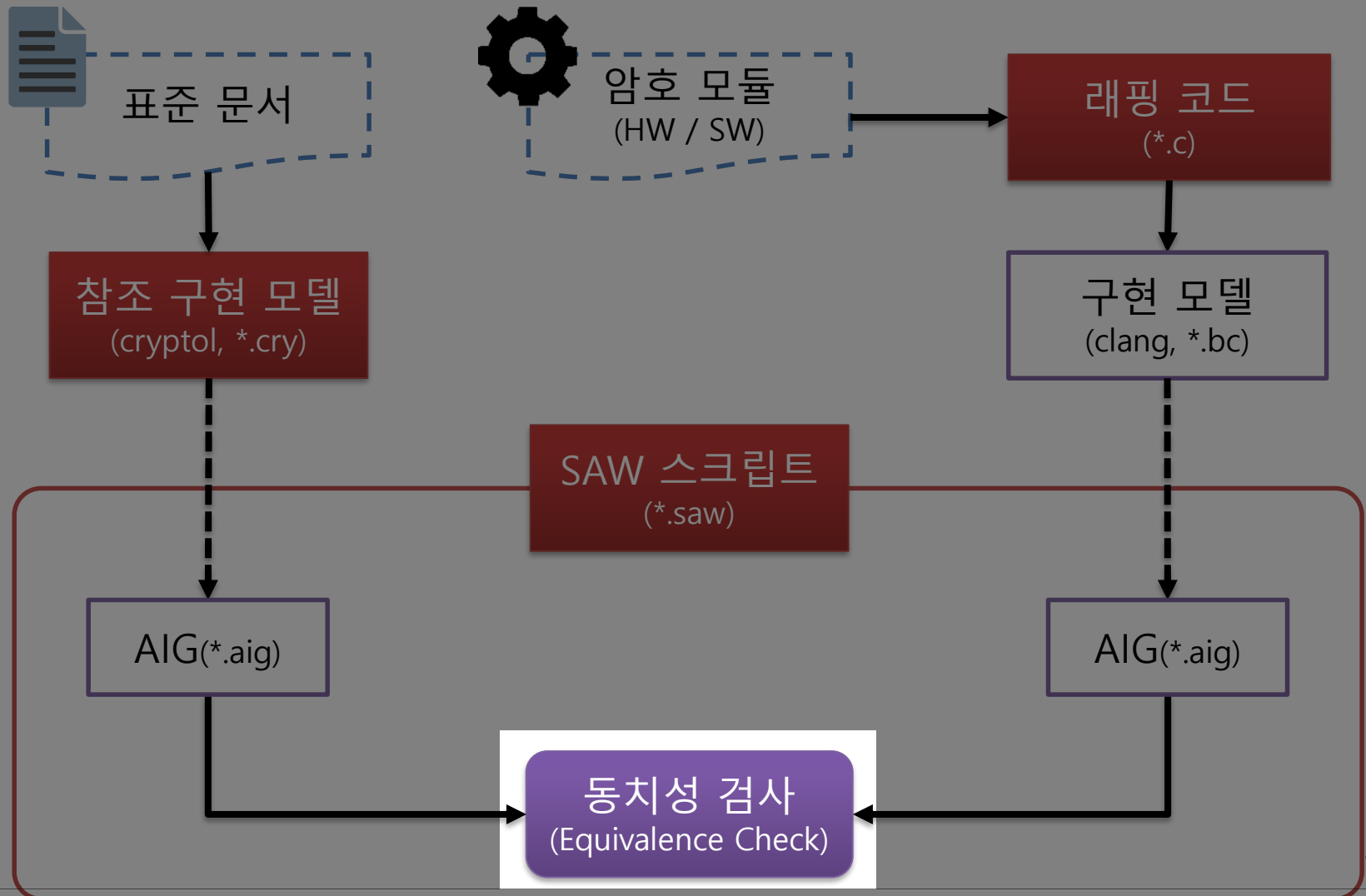
■ 구현 모델의 AIG 생성($AIG_{Impl} - LSS$)

- **LSS(LLVM Symbolic Simulator)**를 사용하여 구현 모델(*.bc)을 AIG파일로 변환
 - Iss ARIA_imp.bc 입력



2. ARIA 구현 및 검증 프로세스

■ 구현 및 검증 아키텍처



2. ARIA 구현 및 검증 프로세스

■ 검증 프로세스(동치성 검사 - SAW)

■ SAW Script

1. 참조 구현 모델 import 및 래핑 코드 작성
2. 구현 모델 AIG 로드
3. CEC(Combinational Equivalence Checking) 수행

```
import "ARIA.cry";

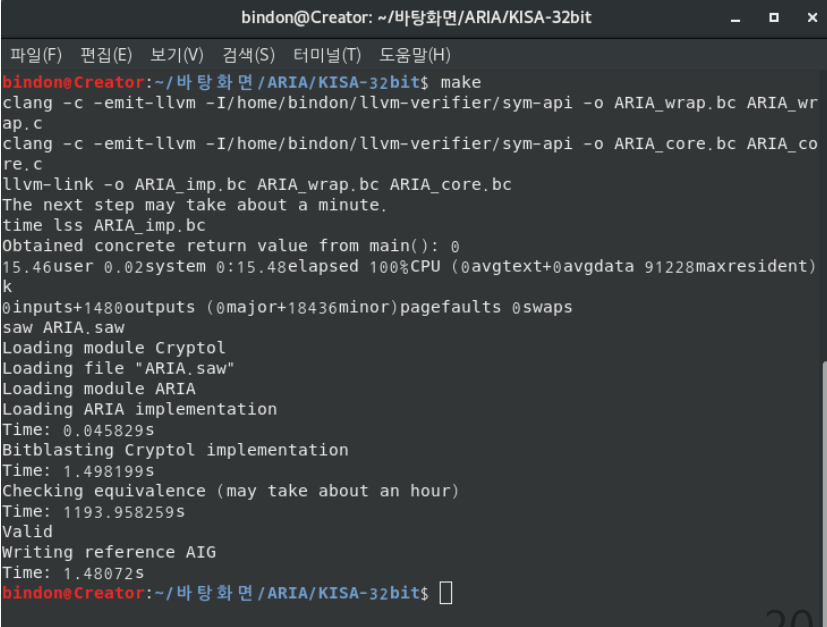
let {{
  ariaExtract x = encrypt (plainText, key)
  where [plainText, key] = split x
}};

print "Loading ARIA implementation";
aes_imp <- time (load_aig "ARIA_imp.aig");

print "Bitblasting Cryptol implementation";
aes_ref <- time (bitblast {{ ariaExtract }});

print "Checking equivalence (may take about an hour)";
res <- time (cec aes_imp aes_ref);
print res;

print "Writing reference AIG";
time (write_aig "ARIA_ref.aig" {{ ariaExtract }});
```



```
bindon@Creator: ~/바탕화면/ARIA/KISA-32bit
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
bindon@Creator:~/바탕화면/ARIA/KISA-32bit$ make
clang -c -emit-llvm -I/home/bindon/llvm-verifier/sym-api -o ARIA_wrap.bc ARIA_wrap.c
clang -c -emit-llvm -I/home/bindon/llvm-verifier/sym-api -o ARIA_core.bc ARIA_core.c
llvm-link -o ARIA_imp.bc ARIA_wrap.bc ARIA_core.bc
The next step may take about a minute.
time ls ARIA_imp.bc
Obtained concrete return value from main(): 0
15.46user 0.02system 0:15.48elapsed 100%CPU (0avgtext+0avgdata 91228maxresident)k
0inputs+1480outputs (0major+18436minor)pagefaults 0swaps
saw ARIA.saw
Loading module Cryptol
Loading file "ARIA.saw"
Loading module ARIA
Loading ARIA implementation
Time: 0.045829s
Bitblasting Cryptol implementation
Time: 1.498199s
Checking equivalence (may take about an hour)
Time: 1193.958259s
Valid
Writing reference AIG
Time: 1.480725s
bindon@Creator:~/바탕화면/ARIA/KISA-32bit$
```

- **Cryptol Ver.2 변경점**
 - **Cryptol Ver.1에서 변경된 사용법**

Cryptol version 2	Cryptol version 1	Summary
<code>[False, True, True] (==3)</code>	<code>[False True True] (== 6)</code>	Big-endian word representation
<code>[1, 1, 2, 3, 5]</code>	<code>[1 1 2 3 5]</code>	Commas separate sequence entries
<code>x = 1</code>	<code>x = 1;</code>	Uses <i>layout</i> instead of ;'s and {'s
<code>[x x <- [1 .. 10]]</code>	<code>[x x <- [1 .. 10]]</code>	Cleaner sequence constructor syntax
<code>f : {a,b} a -> b</code>	<code>f : {a b} a -> b</code>	Commas separate type variables
<code>take' {1} xs</code>	<code>take(1, xs)</code>	First-class type parameters
<code>x ^^ 2</code>	<code>x ** 2</code>	^^ for exponentiation
<code>< x^^2 + 1 ></code>	<code>< x^2 + 1 ></code>	Polynomial exponentiation now uniform
<code>[0 ..]:[_][8]</code>	<code>take(255, [0 ..]:[inf][8])</code>	Both produce [0 .. 255]
<code>[0 ...]:[inf][8]</code>	<code>[0 ..]:[inf][8]</code>	Both produce [0 .. 255] (repeated)
<code>[9, 8 .. 0]</code>	<code>[9 -- 0]</code>	Step defines decreasing sequences
<code>&&, , ^</code>	<code>&, , ^</code>	Boolean operator syntax
<code>property foo xs=...</code>	<code>theorem foo: {xs}. xs==...</code>	Properties replace theorems (see below)

▪ Property Checking

- Haskell 언어로 작성된 CRY 파일의 검증에 사용
- *property* 키워드를 이용하여 검증 방법 작성
- Correctness 검증($D_k(E_k(P)) = P$)

ARIA.cry

```
property ARIACorrect plainText key  
= decrypt (encrypt (plainText, key), key) == plainText
```

Cryptol

```
$ cryptol ARIA.cry  
ARIA> :prove ARIACorrect
```

- 기본 100개의 랜덤 한 값으로 테스트 수행
- :set tests=<개수> 옵션을 통해 변경 가능
- 코드를 작성하면서 신속한 테스트 시 사용
- :prove와는 달리 증명은 불가능 함

23

- **SAT Solver(해당사항 없음)**

- ```
ARIA - cryptol
bindon@Creator:~/bindon/Cryptol$ cryptol sqrt.cry

 _ _ _ _ _
 / \ / \ / \ / \ / \ / \ / \ / \ / \ / \ / \
 () () () () () () () () ()
 \ / \ / \ / \ / \ / \ / \ / \ / \ / \ / \
 _ _ _ _ _ version 2.5.0 (901a1d1)

Loading module Cryptol
Loading module Main
Main> isSqrtOf9(3)
True
Main> :set satNum = 4
Main> :sat isSqrtOf9
isSqrtOf9 0x7d = True
isSqrtOf9 0xfd = True
isSqrtOf9 0x83 = True
isSqrtOf9 0x03 = True
(Total Elapsed Time: 0.000s, using Z3)
```



# 3. 진행중인 연구

- **Property Checking(prove)**
  - **SAT/SMT Solver 사용**
  - 기본적으로 **z3**를 사용하지만 외부 **SAT/SMT Solver**를 사용할 수 있음

```
bindon@Creator: ~/bindon/Cryptol/proveTest
bindon@Creator:~/bindon/Cryptol/proveTest$ cryptol proveTest.cry

┌──────────┐
│ Cryptol │
│ version 2.5.0 (901a1d1) │
└──────────┘

Loading module Cryptol
Loading module Main
Main> :prove
:prove FG
 Q.E.D.
(Total Elapsed Time: 0.008s, using Z3)
:prove FH
 FH 0x00 = False
(Total Elapsed Time: 0.006s, using Z3)
Main>
```

```
bindon@Creator: ~/bindon/Cryptol/proveTest
f, g, h : [8] -> [8]
f x = (x-1)*(x+1)
g x = x*x - 1
h x = x*x + 1
property FG x = f x == g x
property FH x = f x == h x
```

## ▪ Safety Checking(Previous Version)

- 코드에 구현 상의 취약점이 존재하는지 확인
- 기존에는 :safe 키워드를 통한 검사가 가능 했었음
- lookup 함수
  - 매개변수 xs는 4개의 배열, i는 2비트 숫자
    - i는 2비트이므로 0~3의 값을 가질 수 있음
    - xs는 4개의 배열이기 때문에 모든 i에 대해 안전(safe)

lookup.cry

```
lookup : ([4], [2]) -> Bit;
lookup(xs, i) = xs @ i;
```

Example

```
lookup> :set sbv
lookup> :safe lookup
"lookup" is safe; no safety violations exist.
```

## ▪ Safety Checking(Previous Version)

### ▪ lookup2 함수

- 매개변수 **xs**는 4개의 배열, **i**는 **3비트** 숫자
  - **i**는 3비트이므로 0~7의 값을 가질 수 있음
  - **xs**는 4개의 배열이기 때문에 4 이상의 **i**에 대해 안전하지 않음(**unsafe**)

lookup2.cry

```
lookup2 : ([4], [3]) -> Bit;
lookup2(xs, i) = xs @ i;
```

Example

```
lookup2> :safe lookup2
*** 1 safety condition to be checked.
*** Violation detected:
lookup (0, 4) = "lookup2.cry", line 2, col 20: index of 4 is out of
bounds (valid range is 0 thru 3).
*** 1 problem found.
```

## ▪ Safety Checking(Previous Version)

### ▪ lookup3 함수

- 매개변수 **xs**는 4개의 배열, **i**는 **3비트** 숫자
  - **i**는 3비트이므로 0~7의 값을 가질 수 있음
  - **xs**는 4개의 배열이기 때문에 4 이상의 **i**에 안전하지 않았으나 예외처리를 수행하여 안전(**safe**)

lookup3.cry

```
lookup3 : ([4], [3]) -> Bit;
lookup3 (xs, i) = if i >= 3 then False else xs @ i;
```

Example

```
lookup3> :safe lookup3
*** 1 safety condition to be checked.
*** Verified safe.
*** All safety checks pass, safe to execute.
```

# 3. 진행중인 연구

## ▪ Safety Checking(Current Version)

- 현재는 **safe** 키워드가 삭제됨
- **lookup** 함수
  - 매개변수 **xs**는 4개짜리 배열, **i**는 **2비트** 숫자
    - **i**는 2비트이므로 0~3의 값을 가질 수 있음
    - **xs**는 4개의 배열이기 때문에 모든 **i**에 대해 안전(**safe**)
  - **check**를 이용해 모든 경우의 수 테스트
    - 배열 2비트 4개, index 2비트 :  $(2^2)^4 \times 2^2 = 1,024$

lookup.cry

```
lookup : ([4], [2]) -> Bit;
lookup(xs, i) = xs @ i;
```

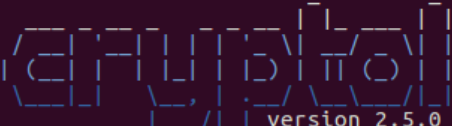
Example

```
lookup> :set sbv
lookup> :safe lookup
"lookup" is safe; no safety violations exist.
```

```
lkup : ([4][2], [2]) -> [2]
lkup (xs, i) = xs @ i
property safeTest(xs, i) = lkup (xs, i) == xs @ i
```

Main - cryptol

```
bindon@Creator:~/bindon/Cryptol/safeTest$ cryptol safeTest.cry
```

 version 2.5.0 (901a1d1)

Loading module Cryptol

Loading module Main

```
Main> :set tests=1024
```

```
Main> :check
```

property safeTest Using exhaustive testing.

passed 1024 tests.

Q.E.D.

```
Main> █
```

## ▪ Safety Checking(Current Version)

### ▪ lookup2 함수

- 매개변수 **xs**는 4개짜리 배열, **i**는 **3비트** 숫자
  - **i**는 3비트이므로 0~7의 값을 가질 수 있음
  - **xs**는 4개의 배열이기 때문에 4 이상의 **i**에 대해 안전하지 않음(**unsafe**)

```
lookup2 : ([4][2], [3]) -> [2]
lookup2 (xs, i) = xs @ i

property safeTest(xs, i) = lookup2 (xs, i) == xs @ i
```

#### lookup2.cry

```
lookup2 : ([4], [3]) -> Bit;
lookup2(xs, i) = xs @ i;
```

#### Example

```
lookup2> :safe lookup2
*** 1 safety condition to be checked.
*** Violation detected:
lookup (0, 4) = "lookup2.cry", line 2, col 20: index of 4 is out of
bounds (valid range is 0 thru 3).
*** 1 problem found.
```

# 3. 진행중인 연구

## ▪ Safety Checking(Current Version)

### ▪ lookup2 함수

#### ▪ check를 이용해 모든 경우의 수 테스트

- 배열 2비트 4개, index 3비트 :  $(2^2)^4 \times 2^3 = 2,048$
- array index out of bounds exception 발생
- but, 테스트케이스가 너무 적으면 검증이 성공하는 일도 있음(i가 4 미만일 경우) -> **현재 암호 검증의 문제점**

```
bindon@Creator:~/bindon/Cryptol/safeTest$ cryptol safeTest.cry
```

```
version 2.5.0
```

```
Loading module Cryptol
Loading module Main
Main> :check
property safeTest Using random testing.
ERROR for the following inputs:
([0x0, 0x2, 0x3, 0x1], 0x7)
invalid sequence index: 7
Main> :set tests=1
Main> :check
property safeTest Using random testing.
passed 1 tests.
Coverage: 0.05% (1 of 2048 values)
```

인덱스가 0~3인 배열에 7을 참조하여  
오류 발생

인덱스가 0~3인 배열에 0~3의 어떤  
값이 임의로 선택되어 통과  
(Coverage가 100%가 아니기 때문에  
Q.E.D가 아님)

## ▪ Safety Checking(Current Version)

### ▪ lookup2 함수

#### ▪ prove를 이용한 증명(SMT/SAT Solver)

- Symbolic Execution을 수행하면 lkup2 (xs, i)가 xs @ i와 심볼이 동일하기 때문에 검증 완료됨(but, unsafe)

```
lkup2 : ([4][2], [3]) -> [2]
lkup2 (xs, i) = xs @ i

property safeTest(xs, i) = lkup2 (xs, i) == xs @ i
```

[PC : true] xs=XS, i=1



[PC : true] result = xs[i]

```
causality
version 2.5.0 (901a1d1)

Loading module Cryptol
Loading module Main
Main> :prove
:prove safeTest
 Q.E.D.
(Total Elapsed Time: 0.005s, using Z3)
Main>
```



## ▪ Safety Checking(Current Version)

### ▪ lookup3 함수

- 매개변수 **xs**는 4개의 배열, **i**는 **3비트** 숫자
  - **i**는 3비트이므로 0~7의 값을 가질 수 있음
  - **xs**는 4개의 배열이기 때문에 4 이상의 **i**에 안전하지 않았으나 예외처리를 수행하여 안전(safe)

```
lookup3 : ([4][2], [3]) -> [2]
lookup3 (xs, i) = if i > 3 then (xs @ (i % 4)) else (xs @ i)

property safeTest(xs, i) = lookup3 (xs, i) == (if i > 3 then (xs @ (i % 4)) else (xs @ i))
```

#### lookup3.cry

```
lookup3 : ([4], [3]) -> Bit;
lookup3 (xs, i) = if i >= 3 then False else xs @ i;
```

#### Example

```
lookup3> :safe lookup3
*** 1 safety condition to be checked.
*** Verified safe.
*** All safety checks pass, safe to execute.
```



## ▪ Safety Checking(Current Version)

### ▪ lookup3 함수

#### ▪ prove를 이용한 검증

- **Symbolic Execution**을 수행하면 심볼이 동일하기 때문에 검증 완료됨

```
lookup3 : ([4][2], [3]) -> [2]
lookup3 (xs, i) = if i > 3 then (xs @ (i % 4)) else (xs @ i)

property safeTest(xs, i) = lookup3 (xs, i) == (if i > 3 then (xs @ (i % 4)) else (xs @ i))
```

[PC : true] xs=XS, i=1

[PC : true] i>3

true

[PC : i>3] result=xs[i%4]

false

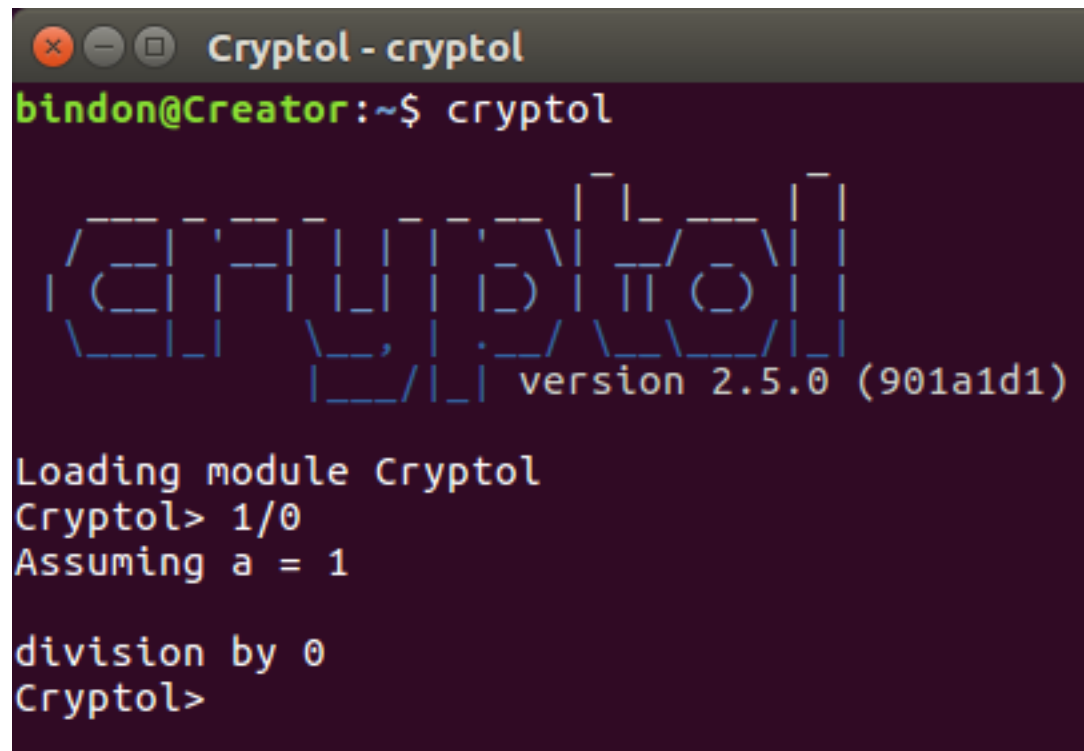
[PC : i<=3] result=xs[i]

```
cauda
version 2.5.0 (901a1d1)

Loading module Cryptol
Loading module Main
Main> :prove
:prove safeTest
Q.E.D.
(Total Elapsed Time: 0.004s, using Z3)
Main>
```

## ■ Safety Checking(Current Version)

- **division by zero**의 경우 아래와 같이 자동으로 검사됨
- 변수에 의한 오류 발생도 동일하게 처리



```
Cryptol - cryptol
bindon@Creator:~$ cryptol

 _ _ _ _ _
 _
 | (C) | F | U | L | L | I | D | I | T | O | R | |
 |_____|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|
 version 2.5.0 (901a1d1)

Loading module Cryptol
Cryptol> 1/0
Assuming a = 1

division by 0
Cryptol>
```

# 4. Cryptol 환경 구축

## ■ Cryptol 설치

### ■ Cryptol 다운로드

- <https://cryptol.net/downloads.html>
- 작성 시점으로 cryptol 2.5.0 Ubuntu 14.04-64 사용

```
bindone@Creator: ~/다운로드
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
bindone@Creator:~/다운로드$ wget https://github.com/GaloisInc/cryptol/releases/download/2.5.0/cryptol-2.5.0-Ubuntu1404-64.tar.gz
--2017-08-16 08:09:12-- https://github.com/GaloisInc/cryptol/releases/download/2.5.0/cryptol-2.5.0-Ubuntu1404-64.tar.gz
Resolving github.com (github.com)... 192.30.253.113, 192.30.253.112
Connecting to github.com (github.com)|192.30.253.113|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://github-production-release-asset-2e65be.s3.amazonaws.com/18895427/0efce6a-7123-11e7-9c9c-82f892e0829f?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWNJYAX4CSVEH53A%2F20170815%2Fus-east-1%2Faws4_request&X-Amz-Date=20170815T230932Z&X-Amz-Expires=300&X-Amz-Signature=0df508ec07756ca8937da46e9173f7c5909b895831583c1a6a6c7ae0d3247bf2&X-Amz-SignedHeaders=host&actor_id=0&response-content-disposition=attachment%3B%20filename%3Dcryptol-2.5.0-Ubuntu1404-64.tar.gz&response-content-type=application%2Foctet-stream [following]
--2017-08-16 08:09:13-- https://github-production-release-asset-2e65be.s3.amazonaws.com/18895427/0efce6a-7123-11e7-9c9c-82f892e0829f?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWNJYAX4CSVEH53A%2F20170815%2Fus-east-1%2Faws4_request&X-Amz-Date=20170815T230932Z&X-Amz-Expires=300&X-Amz-Signature=0df508ec07756ca8937da46e9173f7c5909b895831583c1a6a6c7ae0d3247bf2&X-Amz-SignedHeaders=host&actor_id=0&response-content-disposition=attachment%3B%20filename%3Dcryptol-2.5.0-Ubuntu1404-64.tar.gz&response-content-type=application%2Foctet-stream
Resolving github-production-release-asset-2e65be.s3.amazonaws.com (github-production-release-asset-2e65be.s3.amazonaws.com)... 54.231.32.99
Connecting to github-production-release-asset-2e65be.s3.amazonaws.com (github-production-release-asset-2e65be.s3.amazonaws.com)|54.231.32.99|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 6039366 (5.8M) [application/octet-stream]
Saving to: 'cryptol-2.5.0-Ubuntu1404-64.tar.gz'

cryptol-2.5.0-Ubuntu 100%[=====] 5.76M 2.41MB/s in 2.4s

2017-08-16 08:09:17 (2.41 MB/s) - 'cryptol-2.5.0-Ubuntu1404-64.tar.gz' saved [6039366/6039366]

bindone@Creator:~/다운로드$
```

# 4. Cryptol 환경 구축

## ■ Cryptol 설치

### ■ Cryptol 압축 해제 및 환경변수 등록

#### ■ tar -zxvf cryptol-2.5.0-Ubuntu1404-64.tar.gz

```
bindon@Creator: ~
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
bindon@Creator:~/다운로드$ tar -zxvf cryptol-2.5.0-Ubuntu1404-64.tar.gz
cryptol-2.5.0-Ubuntu14.04-64/
cryptol-2.5.0-Ubuntu14.04-64/bin/
cryptol-2.5.0-Ubuntu14.04-64/bin/cryptol
cryptol-2.5.0-Ubuntu14.04-64/share/
cryptol-2.5.0-Ubuntu14.04-64/share/doc/
cryptol-2.5.0-Ubuntu14.04-64/share/doc/cryptol/
cryptol-2.5.0-Ubuntu14.04-64/share/doc/cryptol/Version2Table.pdf
cryptol-2.5.0-Ubuntu14.04-64/share/doc/cryptol/Version2Table.md
cryptol-2.5.0-Ubuntu14.04-64/share/doc/cryptol/Version2Changes.md
cryptol-2.5.0-Ubuntu14.04-64/share/doc/cryptol/Syntax.pdf
cryptol-2.5.0-Ubuntu14.04-64/share/doc/cryptol/ProgrammingCryptol.pdf
cryptol-2.5.0-Ubuntu14.04-64/share/doc/cryptol/CryptolPrims.pdf
cryptol-2.5.0-Ubuntu14.04-64/share/doc/cryptol/LICENSE
cryptol-2.5.0-Ubuntu14.04-64/share/doc/cryptol/Version2Changes.pdf
cryptol-2.5.0-Ubuntu14.04-64/share/doc/cryptol/examples/
cryptol-2.5.0-Ubuntu14.04-64/share/doc/cryptol/examples/Cipher.cry
cryptol-2.5.0-Ubuntu14.04-64/share/doc/cryptol/examples/SIV-rfc5297.md
cryptol-2.5.0-Ubuntu14.04-64/share/doc/cryptol/examples/Salsa20.cry
cryptol-2.5.0-Ubuntu14.04-64/share/doc/cryptol/examples/contrib/
cryptol-2.5.0-Ubuntu14.04-64/share/doc/cryptol/examples/contrib/mkrand.cry
cryptol-2.5.0-Ubuntu14.04-64/share/doc/cryptol/examples/contrib/RC4.cry
cryptol-2.5.0-Ubuntu14.04-64/share/doc/cryptol/examples/contrib/EvenMansour.cry
cryptol-2.5.0-Ubuntu14.04-64/share/doc/cryptol/examples/contrib/speck.cry
cryptol-2.5.0-Ubuntu14.04-64/share/doc/cryptol/examples/contrib/simon.cry
cryptol-2.5.0-Ubuntu14.04-64/share/doc/cryptol/examples/contrib/README.md
cryptol-2.5.0-Ubuntu14.04-64/share/doc/cryptol/examples/SHA1.cry
cryptol-2.5.0-Ubuntu14.04-64/share/doc/cryptol/examples/DES.cry
cryptol-2.5.0-Ubuntu14.04-64/share/doc/cryptol/examples/MiniLock/
cryptol-2.5.0-Ubuntu14.04-64/share/doc/cryptol/examples/MiniLock/prim/
cryptol-2.5.0-Ubuntu14.04-64/share/doc/cryptol/examples/MiniLock/prim/Salsa20.cry
cryptol-2.5.0-Ubuntu14.04-64/share/doc/cryptol/examples/MiniLock/prim/Poly1305.m
d
cryptol-2.5.0-Ubuntu14.04-64/share/doc/cryptol/examples/MiniLock/prim/LittleEndi
```

```
bindon@Creator: ~
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
fi
else
if ["id -u" -eq 0]; then
 PS1='# '
else
 PS1='$ '
fi
fi
fi
if [-d /etc/profile.d]; then
 for i in /etc/profile.d/*.sh; do
 if [-r $i]; then
 . $i
 fi
 done
 unset i
fi

bindon Home Directory Variable
BINDON_HOME=/home/bindon

Java Environment Variable
JAVA_ROOT=/usr
JAVA7_HOME=$JAVA_ROOT/jdk1.7.0_80
JAVA8_HOME=$JAVA_ROOT/jdk1.8.0_121
JAVA_HOME=$JAVA8_HOME
PATH=$PATH:$JAVA_HOME/bin

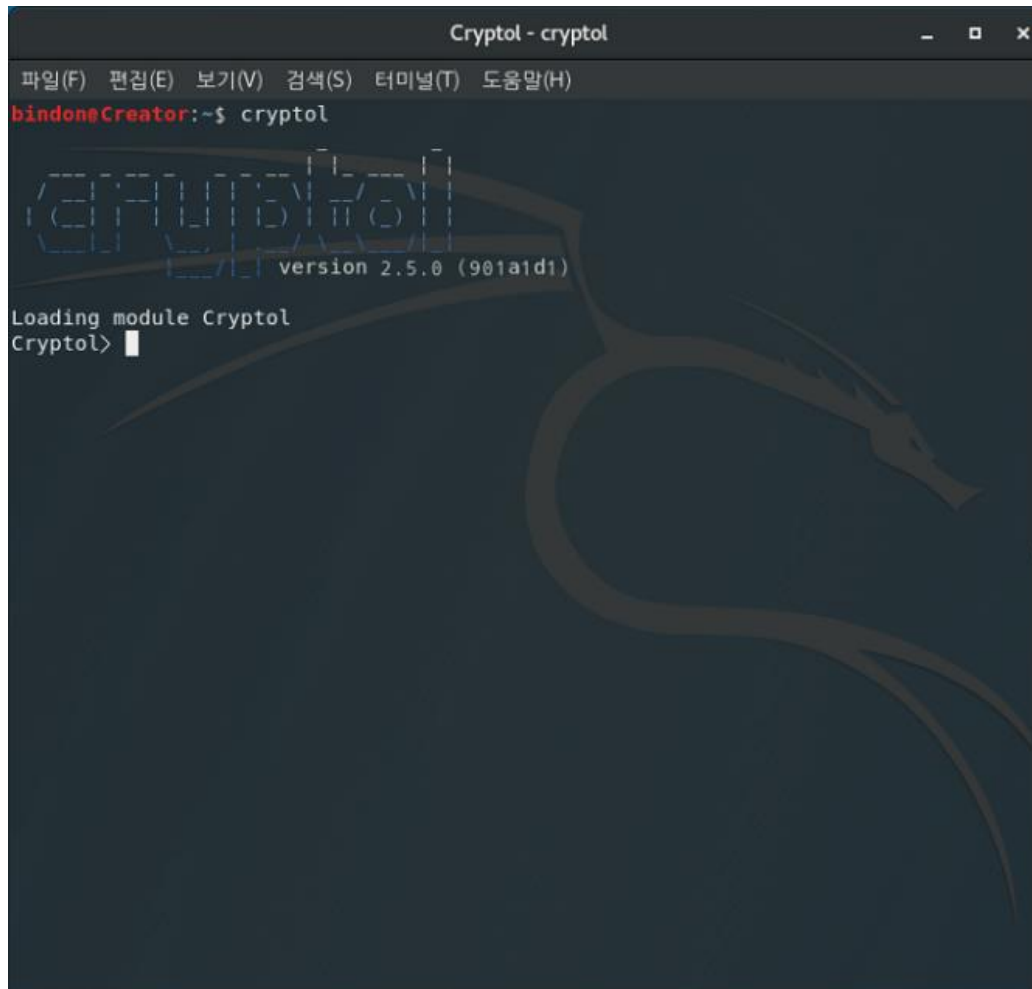
Cryptol, SAW Environment Variable
CRYPTOL_HOME=$BINDON_HOME/cryptol-2.5.0-Ubuntu14.04-64
SAW_HOME=
LLVM_HOME=
PATH=$PATH:$CRYPTOL_HOME/bin:$SAW_HOME/bin:$LLVM_HOME/bin
"/etc/profile" 50L, 1158C 50,1 바닥
```

## ■ Cryptol 설치

- Cryptol을 사용하기 위해 **Dependency Package**인 **Z3 SMT Solver** 설치 필요
- **Package Manager**를 이용한 **z3** 설치
  - **/usr/bin**에 자동으로 설치되는 편리한 방법
  - **apt-get install z3**
  - **debian** 계열이 아닐 경우 각 **Package Manager**를 통해 설치 수행
- 직접 다운로드를 통한 **Z3** 설치
  - 환경변수 등록 필요
  - <https://github.com/Z3Prover/z3/releases>

# 4. Cryptol 환경 구축

- Cryptol 설치
  - Cryptol 실행



```
Cryptol - cryptol
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
bindone@Creator:~$ cryptol
Cryptol
version 2.5.0 (901a1d1)
Loading module Cryptol
Cryptol> |
```



# 4. Cryptol 환경 구축

## ■ SAW 설치

### ■ SAW 다운로드

- <https://saw.galois.com/downloads.html>
- 작성 시점으로 saw 0.2 Ubuntu 14.04-64 사용

```
bindon@Creator: ~/다운로드
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
bindon@Creator:~/다운로드$ wget https://github.com/GaloisInc/saw-script/releases
/download/v0.2/saw-0.2-2016-04-12-Ubuntu14.04-64.tar.gz
--2017-08-16 08:34:41-- https://github.com/GaloisInc/saw-script/releases/downlo
ad/v0.2/saw-0.2-2016-04-12-Ubuntu14.04-64.tar.gz
Resolving github.com (github.com)... 192.30.253.112, 192.30.253.113
Connecting to github.com (github.com)|192.30.253.112|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://github-production-release-asset-2e65be.s3.amazonaws.com/340820
65/c990e900-00d8-11e6-9bab-1ceb7b00a449?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-C
redential=AKIAIWNJYAX4CSVEH53A%2F20170815%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-
Date=20170815T233500Z&X-Amz-Expires=300&X-Amz-Signature=4b4ab05fad1f1fe1f712bd16
7e2b7c4c8b53f26e730bdc34c35230003a622f2c&X-Amz-SignedHeaders=host&actor_id=0&res
ponse-content-disposition=attachment%3B%20filename%3Dsaw-0.2-2016-04-12-Ubuntu14
.04-64.tar.gz&response-content-type=application%2Foctet-stream [following]
--2017-08-16 08:34:42-- https://github-production-release-asset-2e65be.s3.amazo
naws.com/34082065/c990e900-00d8-11e6-9bab-1ceb7b00a449?X-Amz-Algorithm=AWS4-HMAC
-SHA256&X-Amz-Credential=AKIAIWNJYAX4CSVEH53A%2F20170815%2Fus-east-1%2Fs3%2Faws4
_request&X-Amz-Date=20170815T233500Z&X-Amz-Expires=300&X-Amz-Signature=4b4ab05fa
d1f1fe1f712bd167e2b7c4c8b53f26e730bdc34c35230003a622f2c&X-Amz-SignedHeaders=host
&actor_id=0&response-content-disposition=attachment%3B%20filename%3Dsaw-0.2-2016
-04-12-Ubuntu14.04-64.tar.gz&response-content-type=application%2Foctet-stream
Resolving github-production-release-asset-2e65be.s3.amazonaws.com (github-produc
tion-release-asset-2e65be.s3.amazonaws.com)... 52.216.97.123
Connecting to github-production-release-asset-2e65be.s3.amazonaws.com (github-pr
oduction-release-asset-2e65be.s3.amazonaws.com)|52.216.97.123|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 42473727 (41M) [application/octet-stream]
Saving to: 'saw-0.2-2016-04-12-Ubuntu14.04-64.tar.gz'

saw-0.2-2016-04-12- 100%[=====] 40.51M 5.73MB/s in 8.4s

2017-08-16 08:34:51 (4.82 MB/s) - 'saw-0.2-2016-04-12-Ubuntu14.04-64.tar.gz' sav
ed [42473727/42473727]

bindon@Creator:~/다운로드$
```

# 4. Cryptol 환경 구축

## ■ SAW 설치

- SAW 압축 해제 및 환경변수 등록
- `tar -zxvf saw-0.2-2016-04-12-Ubuntu14.04-64.tar.gz`

```
bindon@Creator: ~/다운로드
파임(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
bindon@Creator:~/다운로드$ tar -zxvf saw-0.2-2016-04-12-Ubuntu14.04-64.tar.gz
saw-0.2-2016-04-12-Ubuntu14.04-64/
saw-0.2-2016-04-12-Ubuntu14.04-64/lib/
saw-0.2-2016-04-12-Ubuntu14.04-64/lib/Cryptol/
saw-0.2-2016-04-12-Ubuntu14.04-64/lib/Cryptol/Extras.cry
saw-0.2-2016-04-12-Ubuntu14.04-64/lib/Cryptol.cry
saw-0.2-2016-04-12-Ubuntu14.04-64/lib/galois.jar
saw-0.2-2016-04-12-Ubuntu14.04-64/bin/
saw-0.2-2016-04-12-Ubuntu14.04-64/bin/jss
saw-0.2-2016-04-12-Ubuntu14.04-64/bin/bcdump
saw-0.2-2016-04-12-Ubuntu14.04-64/bin/lss
saw-0.2-2016-04-12-Ubuntu14.04-64/bin/llvm-disasm
saw-0.2-2016-04-12-Ubuntu14.04-64/bin/saw
saw-0.2-2016-04-12-Ubuntu14.04-64/LICENSE
saw-0.2-2016-04-12-Ubuntu14.04-64/doc/
saw-0.2-2016-04-12-Ubuntu14.04-64/doc/sawScriptTutorial.pdf
saw-0.2-2016-04-12-Ubuntu14.04-64/doc/extcore.txt
saw-0.2-2016-04-12-Ubuntu14.04-64/doc/code/
saw-0.2-2016-04-12-Ubuntu14.04-64/doc/code/Cipher.cry
saw-0.2-2016-04-12-Ubuntu14.04-64/doc/code/ffs.cry
saw-0.2-2016-04-12-Ubuntu14.04-64/doc/code/NQueens.cry
saw-0.2-2016-04-12-Ubuntu14.04-64/doc/code/basic.c
saw-0.2-2016-04-12-Ubuntu14.04-64/doc/code/double.bc
saw-0.2-2016-04-12-Ubuntu14.04-64/doc/code/dotprod.bc
saw-0.2-2016-04-12-Ubuntu14.04-64/doc/code/ffs_compare.saw
saw-0.2-2016-04-12-Ubuntu14.04-64/doc/code/ffs_llvm.saw
saw-0.2-2016-04-12-Ubuntu14.04-64/doc/code/dotprod.cry
saw-0.2-2016-04-12-Ubuntu14.04-64/doc/code/picosat.saw
saw-0.2-2016-04-12-Ubuntu14.04-64/doc/code/double.saw
saw-0.2-2016-04-12-Ubuntu14.04-64/doc/code/DES.cry
saw-0.2-2016-04-12-Ubuntu14.04-64/doc/code/basic.bc
saw-0.2-2016-04-12-Ubuntu14.04-64/doc/code/java_symexec.saw
saw-0.2-2016-04-12-Ubuntu14.04-64/doc/code/Makefile
saw-0.2-2016-04-12-Ubuntu14.04-64/doc/code/nqueens.saw
saw-0.2-2016-04-12-Ubuntu14.04-64/doc/code/dotprod.saw
```

```
bindon@Creator: ~
파임(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
fi
else
 if ["`id -u`" -eq 0]; then
 PS1='# '
 else
 PS1='$ '
 fi
fi
fi
if [-d /etc/profile.d]; then
 for i in /etc/profile.d/*.sh; do
 if [-r $i]; then
 . $i
 fi
 done
 unset i
fi

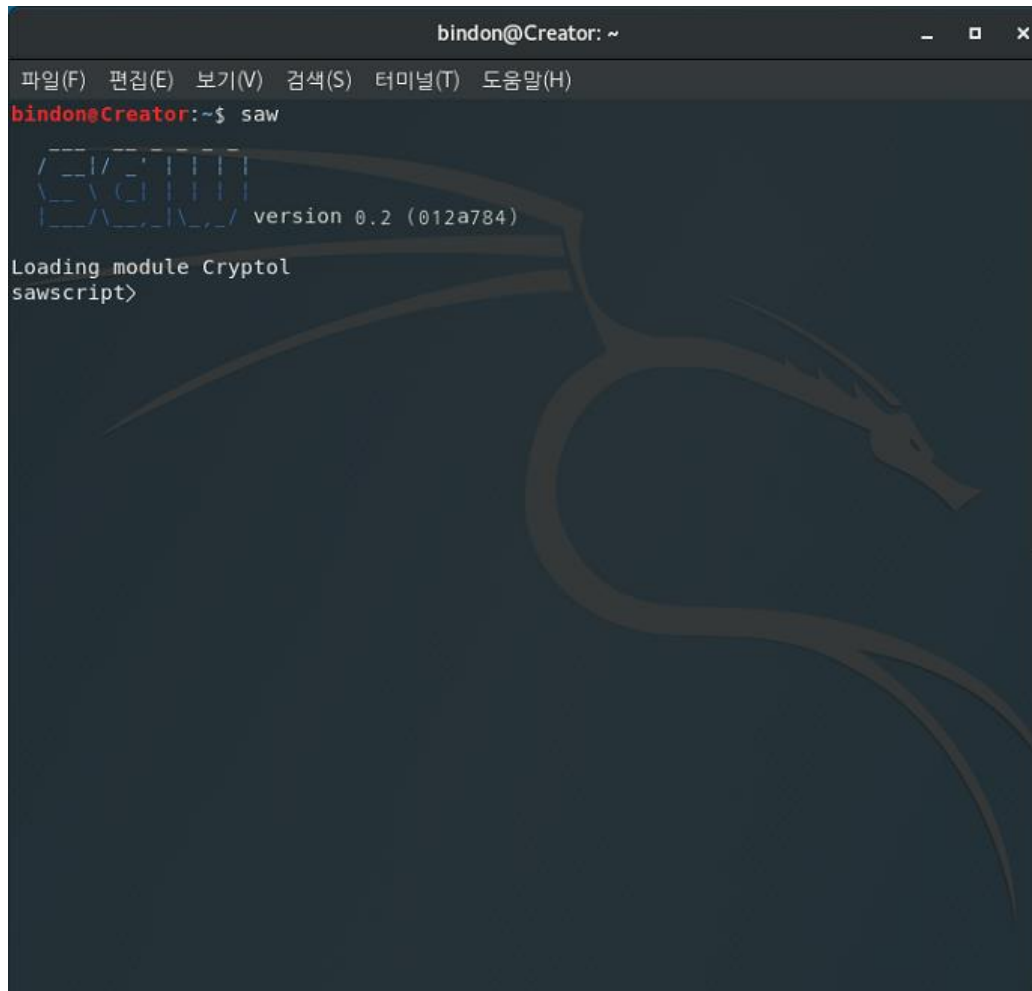
bindon Home Directory Variable
BINDON_HOME=/home/bindon

Java Environment Variable
JAVA_ROOT=/usr
JAVA7_HOME=$JAVA_ROOT/jdk1.7.0_80
JAVA8_HOME=$JAVA_ROOT/jdk1.8.0_121
JAVA_HOME=$JAVA8_HOME
PATH=$PATH:$JAVA_HOME/bin

Cryptol, SAW Environment Variable
CRYPTOL_HOME=$BINDON_HOME/cryptol-2.5.0-Ubuntu14.04-64
SAW_HOME=$BINDON_HOME/saw-0.2-2016-04-12-Ubuntu14.04-64
LLVM_HOME=
PATH=$PATH:$CRYPTOL_HOME/bin:$SAW_HOME/bin:$LLVM_HOME/bin
"/etc/profile" 소L, 1204C 저장 했습니다
48,55 바닥
```

# 4. Cryptol 환경 구축

- SAW 설치
  - SAW 실행

A terminal window titled 'bindon@Creator: ~' with a menu bar containing '파일(F)', '편집(E)', '보기(V)', '검색(S)', '터미널(T)', and '도움말(H)'. The terminal shows the command 'saw' being executed, which outputs 'version 0.2 (012a784)' and 'Loading module Cryptol'. The prompt then changes to 'sawscript>'. A large, faint dragon logo is visible in the background of the terminal window.

```
bindon@Creator: ~
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
bindon@Creator:~$ saw
version 0.2 (012a784)
Loading module Cryptol
sawscript>
```

# 4. Cryptol 환경 구축

## ■ Clang 설치

### ■ Clang 다운로드

- 최신 버전이 아닌 **clang-llvm-3.6.2-x86\_64** 다운로드
- 최신 버전의 경우 **saw** 연동 시 오류가 발생할 수 있음

```
bindon@Creator: ~/다운로드
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
bindoneCreator:~/다운로드$ wget http://releases.llvm.org/3.6.2/clang+llvm-3.6.2-x86_64-linux-gnu-ubuntu-14.04.tar.xz
--2017-08-16 09:04:34-- http://releases.llvm.org/3.6.2/clang+llvm-3.6.2-x86_64-linux-gnu-ubuntu-14.04.tar.xz
Resolving releases.llvm.org (releases.llvm.org)... 151.101.26.49
Connecting to releases.llvm.org (releases.llvm.org)|151.101.26.49|:80... connect
ed.
HTTP request sent, awaiting response... 200 OK
Length: 71861184 (69M) [application/x-tar]
Saving to: 'clang+llvm-3.6.2-x86_64-linux-gnu-ubuntu-14.04.tar.xz'

clang+llvm-3.6.2-x8 100%[=====] 68.53M 5.60MB/s in 13s

2017-08-16 09:04:48 (5.37 MB/s) - 'clang+llvm-3.6.2-x86_64-linux-gnu-ubuntu-14.0
4.tar.xz' saved [71861184/71861184]

bindoneCreator:~/다운로드$
```



# 4. Cryptol 환경 구축

## ■ Clang 설치

### ■ Clang 압축 해제 및 환경변수 등록

- `tar -xvf clang+llvm-3.6.2-x86_64-linux-gnu-ubuntu-14.04.tar.xz`

```
bindon@Creator: ~/다운로드
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)

bindon@Creator:~/다운로드$ tar -xvf clang+llvm-3.6.2-x86_64-linux-gnu-ubuntu-14.04.tar.xz
clang+llvm-3.6.2-x86_64-linux-gnu-ubuntu-14.04/
clang+llvm-3.6.2-x86_64-linux-gnu-ubuntu-14.04/share/
clang+llvm-3.6.2-x86_64-linux-gnu-ubuntu-14.04/share/man/
clang+llvm-3.6.2-x86_64-linux-gnu-ubuntu-14.04/share/man/man1/
clang+llvm-3.6.2-x86_64-linux-gnu-ubuntu-14.04/share/man/man1/clang.1
clang+llvm-3.6.2-x86_64-linux-gnu-ubuntu-14.04/share/llvm/
clang+llvm-3.6.2-x86_64-linux-gnu-ubuntu-14.04/share/llvm/cmake/
clang+llvm-3.6.2-x86_64-linux-gnu-ubuntu-14.04/share/llvm/cmake/ChooseMSVCCRT.cmake
clang+llvm-3.6.2-x86_64-linux-gnu-ubuntu-14.04/share/llvm/cmake/AddOCaml.cmake
clang+llvm-3.6.2-x86_64-linux-gnu-ubuntu-14.04/share/llvm/cmake/GetSVN.cmake
clang+llvm-3.6.2-x86_64-linux-gnu-ubuntu-14.04/share/llvm/cmake/TableGen.cmake
clang+llvm-3.6.2-x86_64-linux-gnu-ubuntu-14.04/share/llvm/cmake/HandleLLVMOptions.cmake
clang+llvm-3.6.2-x86_64-linux-gnu-ubuntu-14.04/share/llvm/cmake/LLVMProcessSources.cmake
clang+llvm-3.6.2-x86_64-linux-gnu-ubuntu-14.04/share/llvm/cmake/LLVMExports.cmake
clang+llvm-3.6.2-x86_64-linux-gnu-ubuntu-14.04/share/llvm/cmake/LLVM-Config.cmake
clang+llvm-3.6.2-x86_64-linux-gnu-ubuntu-14.04/share/llvm/cmake/CrossCompile.cmake
clang+llvm-3.6.2-x86_64-linux-gnu-ubuntu-14.04/share/llvm/cmake/AddLLVMDefinitions.cmake
clang+llvm-3.6.2-x86_64-linux-gnu-ubuntu-14.04/share/llvm/cmake/LLVMParseArguments.cmake
clang+llvm-3.6.2-x86_64-linux-gnu-ubuntu-14.04/share/llvm/cmake/FindOCaml.cmake
clang+llvm-3.6.2-x86_64-linux-gnu-ubuntu-14.04/share/llvm/cmake/AddLLVM.cmake
clang+llvm-3.6.2-x86_64-linux-gnu-ubuntu-14.04/share/llvm/cmake/AddSphinxTarget.cmake
clang+llvm-3.6.2-x86_64-linux-gnu-ubuntu-14.04/share/llvm/cmake/LLVMConfigVersion.cmake
```

```
bindon@Creator: ~
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)

fi
else
if ["`id -u`" -eq 0]; then
PS1='# '
else
PS1='$ '
fi
fi
fi
if [-d /etc/profile.d]; then
for i in /etc/profile.d/*.sh; do
if [-r $i]; then
. $i
fi
done
unset i
fi

bindon Home Directory Variable
BINDON_HOME=/home/bindon

Java Environment Variable
JAVA_ROOT=/usr
JAVA7_HOME=$JAVA_ROOT/jdk1.7.0_80
JAVA8_HOME=$JAVA_ROOT/jdk1.8.0_121
JAVA_HOME=$JAVA8_HOME
PATH=$PATH:$JAVA_HOME/bin

Cryptol, SAW Environment Variable
CRYPTOL_HOME=$BINDON_HOME/cryptol-2.5.0-Ubuntu14.04-64
SAW_HOME=$BINDON_HOME/saw-0.2-2016-04-12-Ubuntu14.04-64
LLVM_HOME=$BINDON_HOME/clang+llvm-3.6.2-x86_64-linux-gnu-ubuntu-14.04
PATH=$PATH:$CRYPTOL_HOME/bin:$SAW_HOME/bin:$LLVM_HOME/bin
"/etc/profile" 50L, 1263C 저장 했습니다
50,1
바닥
```

# 4. Cryptol 환경 구축

- Clang 설치
  - Clang, LLVM 설치 확인
    - clang --version
    - llvm-link --version

```
bindon@Creator: ~/clang+llvm-3.6.2-x86_64-linux-gnu-ubuntu-14.04/bin
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
bindoneCreator:~/clang+llvm-3.6.2-x86_64-linux-gnu-ubuntu-14.04/bin$ clang --version
clang version 3.6.2 (tags/RELEASE_362/final)
Target: x86_64-unknown-linux-gnu
Thread model: posix
bindoneCreator:~/clang+llvm-3.6.2-x86_64-linux-gnu-ubuntu-14.04/bin$ llvm-link --version
LLVM (http://llvm.org/):
 LLVM version 3.6.2
 Optimized build.
 Default target: x86_64-unknown-linux-gnu
 Host CPU: corei7-avx
bindoneCreator:~/clang+llvm-3.6.2-x86_64-linux-gnu-ubuntu-14.04/bin$
```

# 4. Cryptol 환경 구축

## ■ llvm-verifier 설정

- llvm-verifier 다운로드 및 압축 해제
  - c언어로 작성된 모듈을 wrapping하기 위한 library
  - <https://github.com/GaloisInc/llvm-verifier>

```
bindon@Creator: ~/다운로드
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
bindon@Creator:~/다운로드$ unzip llvm-verifier-master.zip
Archive: llvm-verifier-master.zip
777ba398aeb2442d5539274ee8dd4ad9dea6154
 creating: llvm-verifier-master/
 extracting: llvm-verifier-master/.gitignore
 inflating: llvm-verifier-master/ABC_LICENSE
 inflating: llvm-verifier-master/LICENSE
 inflating: llvm-verifier-master/Memory.hs
 inflating: llvm-verifier-master/README.md
 extracting: llvm-verifier-master/Setup.hs
 creating: llvm-verifier-master/bcdump/
 inflating: llvm-verifier-master/bcdump/Main_bcdump.hs
 creating: llvm-verifier-master/doc/
 inflating: llvm-verifier-master/doc/lss-api.md
 creating: llvm-verifier-master/doc/lss-tutorial/
 inflating: llvm-verifier-master/doc/lss-tutorial/Makefile
 creating: llvm-verifier-master/doc/lss-tutorial/bib/
 inflating: llvm-verifier-master/doc/lss-tutorial/bib/lss-tutorial.bib
 creating: llvm-verifier-master/doc/lss-tutorial/code/
 inflating: llvm-verifier-master/doc/lss-tutorial/code/AES.cry
 inflating: llvm-verifier-master/doc/lss-tutorial/code/Makefile
 inflating: llvm-verifier-master/doc/lss-tutorial/code/aes.bc
 inflating: llvm-verifier-master/doc/lss-tutorial/code/aes.saw
 inflating: llvm-verifier-master/doc/lss-tutorial/code/aes128BlockEncrypt.bc
 inflating: llvm-verifier-master/doc/lss-tutorial/code/aes128BlockEncrypt.c
 inflating: llvm-verifier-master/doc/lss-tutorial/code/aes128BlockEncrypt.h
 inflating: llvm-verifier-master/doc/lss-tutorial/code/aes128BlockEncrypt_drive
r.bc
 inflating: llvm-verifier-master/doc/lss-tutorial/code/aes128BlockEncrypt_drive
r.c
 creating: llvm-verifier-master/doc/lss-tutorial/images/
 inflating: llvm-verifier-master/doc/lss-tutorial/images/Galois_logo_blue_box.p
df
 inflating: llvm-verifier-master/doc/lss-tutorial/images/Galois_print_logo_blue
.pdf
```

## ■ 참조 구현 모델 작성(ARIA.cry)

### ■ ARIA 표준 문서를 바탕으로 참조 구현 모델 작성

```
bindon@Creator: ~/bindon/Cryptol/aria/KISA-32bit

encrypt : ([128], [128]) -> [128]
encrypt (plainText, key) = cipherText
 where
 ek = generateEncKey(key)
 r1 = oddFunction (plainText, ek @ 0)
 r2 = evenFunction(r1, ek @ 1)
 r3 = oddFunction (r2, ek @ 2)
 r4 = evenFunction(r3, ek @ 3)
 r5 = oddFunction (r4, ek @ 4)
 r6 = evenFunction(r5, ek @ 5)
 r7 = oddFunction (r6, ek @ 6)
 r8 = evenFunction(r7, ek @ 7)
 r9 = oddFunction (r8, ek @ 8)
 r10 = evenFunction(r9, ek @ 9)
 r11 = oddFunction (r10, ek @ 10)
 cipherText = finalFunction (r11, ek @ 11, ek @ 12)

decrypt : ([128], [128]) -> [128]
decrypt (plainText, key) = cipherText
 where
 dk = generateDecKey(key)
 r1 = oddFunction (plainText, dk @ 0)
 r2 = evenFunction(r1, dk @ 1)
 r3 = oddFunction (r2, dk @ 2)
 r4 = evenFunction(r3, dk @ 3)
 r5 = oddFunction (r4, dk @ 4)
 r6 = evenFunction(r5, dk @ 5)
 r7 = oddFunction (r6, dk @ 6)
 r8 = evenFunction(r7, dk @ 7)
 r9 = oddFunction (r8, dk @ 8)
 r10 = evenFunction(r9, dk @ 9)
 r11 = oddFunction (r10, dk @ 10)
 cipherText = finalFunction (r11, dk @ 11, dk @ 12)

property ARIACorrect plainText key = decrypt (encrypt (plainText, key), key) == plainText
```



- 래핑 코드 작성(ARIA\_wrap.c)
  - ARIA 암호 모듈을 호출하는 래핑 코드 작성
  - llvm-verifier 라이브러리 import를 통해 변수 할당

```
bindon@Creator: ~/bindon/Cryptol/aria/KISA-32bit
#include "ARIA_core.h"
#include <sym-api.h>

void encrypt(unsigned char *plainText, unsigned char *cipherText, unsigned char *key) {
 unsigned char rk[16*17];
 Crypt(plainText, EncKeySetup(key, rk, 128), rk, cipherText);
}

int main() {
 unsigned char *plainText = lss_fresh_array_uint8(16, 0, NULL);
 unsigned char *key = lss_fresh_array_uint8(16, 0, NULL);
 unsigned char *cipherText = malloc(16 * sizeof(unsigned char));

 encrypt(plainText, cipherText, key);

 lss_write_aiger_array_uint8(cipherText, 16, "ARIA_imp.aig");

 return 0;
}
```

- **SAW 스크립트 작성(ARIA.saw)**
  - 참조 구현 모델과 구현 모델 **AIG**를 이용하여 동치성 검사를 수행하는 **SAW** 스크립트 작성

```
bindon@Creator: ~/bindon/Cryptol/aria/KISA-32bit
import "ARIA.cry";

let {{
 ariaExtract x = encrypt (plainText, key)
 where [plainText, key] = split x
}};

print "Loading ARIA implementation";
aes_imp <- time (load_aig "ARIA_imp.aig");

print "Bitblasting Cryptol implementation";
aes_ref <- time (bitblast {{ ariaExtract }});

print "Checking equivalence (may take about an hour)";
res <- time (cec aes_imp aes_ref);
print res;

print "Writing reference AIG";
time (write_aig "ARIA_ref.aig" {{ ariaExtract }});
```

## ▪ Makefile 작성(Makefile)

- 작성된 코드들을 이용하여 배치 작업을 한 번에 수행할 수 있도록 Makefile을 작성
- 현재 환경에 맞게 수정 필요

```
bindon@Creator: ~/bindon/Cryptol/aria/KISA-32bit
CLANG?=clang
LLVM_LINK?=llvm-link
SAW?=saw
LSS?=lss
LLVM_VERIFIER=/home/bindon/llvm-verifier/sym-api

all: ARIA_imp.aig

ARIA_all.bc:
 ${CLANG} -c -emit-llvm -I${LLVM_VERIFIER} -o ARIA_wrap.bc ARIA_wrap.c
 ${CLANG} -c -emit-llvm -I${LLVM_VERIFIER} -o ARIA_core.bc ARIA_core.c

ARIA_imp.bc: ARIA_all.bc
 ${LLVM_LINK} -o ARIA_imp.bc ARIA_wrap.bc ARIA_core.bc

ARIA_imp.aig: ARIA_imp.bc
 @echo "The next step may take about a minute."
 time ${LSS} ARIA_imp.bc
 ${SAW} ARIA.saw

clean:
 rm -f ARIA_wrap.bc ARIA_imp.aig
```

# 5. ARIA 구현 및 검증

## ■ 검증 수행(make)

### ■ Makefile이 존재하는 폴더에서 make 수행

```
bindon@Creator: ~/바탕화면/ARIA/KISA-32bit
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
bindon@Creator:~/바탕화면/ARIA/KISA-32bit$ make
clang -c -emit-llvm -I/home/bindon/llvm-verifier/sym-api -o ARIA_wrap.bc ARIA_wrap.c
clang -c -emit-llvm -I/home/bindon/llvm-verifier/sym-api -o ARIA_core.bc ARIA_core.c
llvm-link -o ARIA_imp.bc ARIA_wrap.bc ARIA_core.bc
The next step may take about a minute.
time lss ARIA_imp.bc
Obtained concrete return value from main(): 0
15.46user 0.02system 0:15.48elapsed 100%CPU (0avgtext+0avgdata 91228maxresident)k
0inputs+1480outputs (0major+18436minor)pagefaults 0swaps
saw ARIA.saw
Loading module Cryptol
Loading file "ARIA.saw"
Loading module ARIA
Loading ARIA implementation
Time: 0.045829s
Bitblasting Cryptol implementation
Time: 1.498199s
Checking equivalence (may take about an hour)
Time: 1193.958259s
Valid
Writing reference AIG
Time: 1.48072s
bindon@Creator:~/바탕화면/ARIA/KISA-32bit$
```

# Cryptol을 이용한 국내 표준 블록 암호 ARIA 모듈의 자동 정형 검증

## (Automated Formal Verification of Korean Standard Block Cipher ARIA using Cryptol)

---

**Security Analysis aNd Evaluation Lab, CIST**

**최원빈 (Wonbin Choi), [bindon@hanmir.com](mailto:bindon@hanmir.com)**

**Prof. Dr. Seungjoo (Gabriel) Kim**

