Valgrind를 통한 알고리즘 안전성 검사

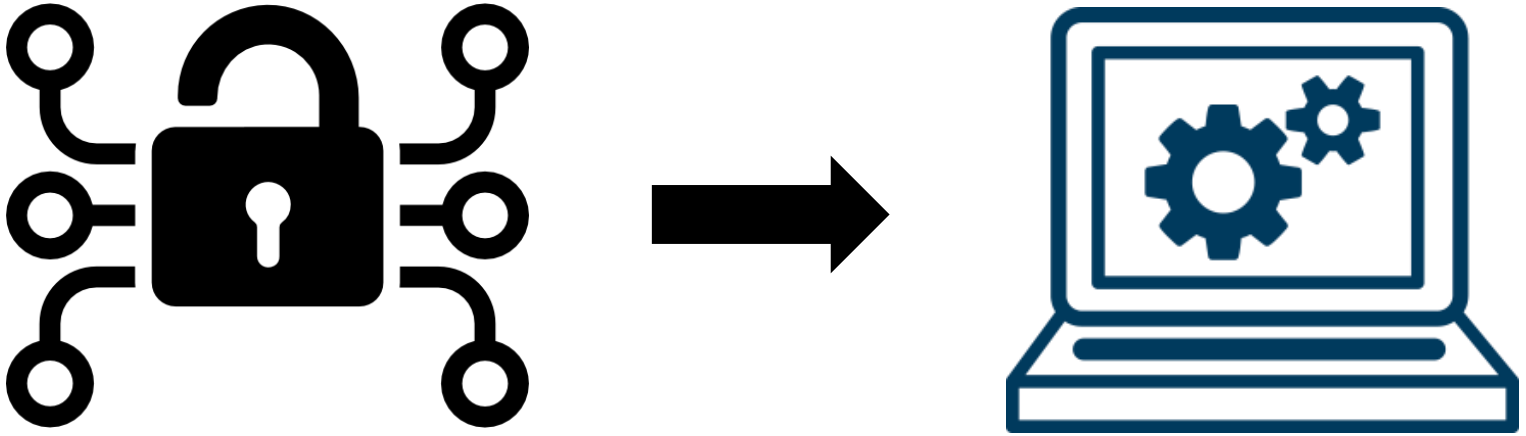
정보컴퓨터공학과 권혁동

한성대학교 HANSUNG UNIVERSITY

CryptoCraft LAB

암호 알고리즘

- 암호 알고리즘은 수학적 원리에 기반해서 안전성 확보
- 컴퓨터 언어로 **구현하는 과정은 별개**의 문제
  - 구현이 잘못될 경우 알고리즘이 위험해질 수 있음
- 따라서 **안전하게 구현 되었는지 확인**하는 과정이 필요



메모리 누수

- 프로그램이 사용한 메모리를 반환하지 않는 증상
  - **사용이 끝난 메모리는 반환해서 다시 비워두어야 함**
- 메모리를 자동으로 반환하는 기능이 존재
  - **쓰레기 수집(Garbage Collection)**
- 그러나 C, C++ 같은 **일부 언어는 쓰레기 수집 기능이 없음**
  - 프로그래머의 코드 구현을 전적으로 따름

메모리 누수

- Valgrind 툴을 사용해서 메모리 누수를 확인할 수 있음
  - 사용자가 **메모리 할당을 하고 free를 하지 않으면 발생**
  - 할당을 해제 해주면 정상으로 판단

```c
#include <stdlib.h>

int main(void)
{
    int *a = (int *)malloc(sizeof(int) * 8);

    return 0;
}
```

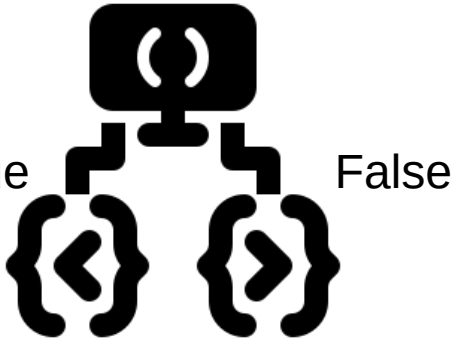
```
==119082== Memcheck, a memory error detector
==119082== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==119082== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==119082== Command: ./a.out
==119082==
==119082==
==119082== HEAP SUMMARY:
==119082==     in use at exit: 32 bytes in 1 blocks
==119082==   total heap usage: 1 allocs, 0 frees, 32 bytes allocated
==119082==
==119082== 32 bytes in 1 blocks are definitely lost in loss record 1 of 1
==119082==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-
amd64-linux.so)
==119082==    by 0x10915E: main (in /home/hd/a.out)
==119082==
==119082== LEAK SUMMARY:
==119082==    definitely lost: 32 bytes in 1 blocks
==119082==    indirectly lost: 0 bytes in 0 blocks
==119082==      possibly lost: 0 bytes in 0 blocks
==119082==    still reachable: 0 bytes in 0 blocks
==119082==         suppressed: 0 bytes in 0 blocks
==119082==
==119082== For lists of detected and suppressed errors, rerun with: -s
==119082== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
```

```c
#include <stdlib.h>

int main(void)
{
    int *a = (int *)malloc(sizeof(int) * 8);

    free(a);

    return 0;
}
```

```
==119142== Memcheck, a memory error detector
==119142== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==119142== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==119142== Command: ./a.out
==119142==
==119142==
==119142== HEAP SUMMARY:
==119142==     in use at exit: 0 bytes in 0 blocks
==119142==   total heap usage: 1 allocs, 1 frees, 32 bytes allocated
==119142==
==119142== All heap blocks were freed -- no leaks are possible
==119142==
==119142== For lists of detected and suppressed errors, rerun with: -s
==119142== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

상수 시간 구현

- 상수 시간 구현(Constant Time)은 타이밍 공격에 중요한 부분
  - 연산 시간이 일정치 않으면, **연산 시간을 통해서 중요 정보 노출** 가능
- 조건문을 사용하면 상수 시간 구현이 깨질 수 있음
  - 분기마다 연산 속도가 다르기 때문
  - **조건문에 사용된 값을 알 수 있음**



연산시간: 1ms　　연산시간: 2ms

상수 시간 구현

- Valgrind에 체크하고자 하는 값을 플래그로 준 다음 검사
  - 해당 값을 사용한 분기가 있다면 그 자리를 문제로 점검
  - Valgrind는 **분기문의 뜻은 해석하지 못함**, 때문에 예외가 있을 수 있음

```c
#include <stdio.h>
#include <valgrind/memcheck.h>

int main(void)
{
    int a[8] = {0, 1, 2, 3, 4, 5, 6, 7};

    VALGRIND_MAKE_MEM_UNDEFINED(a, 8);
    if(a[0] < 4)
    {
        printf("A is less than 4 \n");
        printf("A is less than 4 \n");
        printf("A is less than 4 \n");
        printf("A is less than 4 \n");
    }

    return 0;
}
```

```
==119412== Memcheck, a memory error detector
==119412== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==119412== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==119412== Command: ./a.out
==119412==
==119412== Conditional jump or move depends on uninitialised value(s)
==119412==    at 0x109495: main (in /home/hd/a.out)
==119412==  Uninitialised value was created by a client request
==119412==    at 0x109484: main (in /home/hd/a.out)
==119412==
A is less than 4
A is less than 4
A is less than 4
A is less than 4
==119412==
==119412== HEAP SUMMARY:
==119412==     in use at exit: 0 bytes in 0 blocks
==119412==   total heap usage: 1 allocs, 1 frees, 1,024 bytes allocated
==119412==
==119412== All heap blocks were freed -- no leaks are possible
==119412==
==119412== For lists of detected and suppressed errors, rerun with: -s
==119412== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
```

```
ditional jump or move depends on uninitialised value(s)
at 0x10D6EA: aimer_instance_get (in /home/hd/kpqclean-Round_valgrind/dsa-AIMer_revised0623/AIMer-l1-param1/PQC_be
by 0x109888: crypto_sign (in /home/hd/kpqclean-Round_valgrind/dsa-AIMer_revised0623/AIMer-l1-param1/PQC_bench)
```

```c
const aimer_instance_t *aimer_instance_get(aimer_params_t param)
{
  if (param <= PARAMETER_SET_INVALID || param >= PARAMETER_SET_MAX_INDEX)
  {
    return NULL;
  }

  return &instances[param];
}
```

Q & A