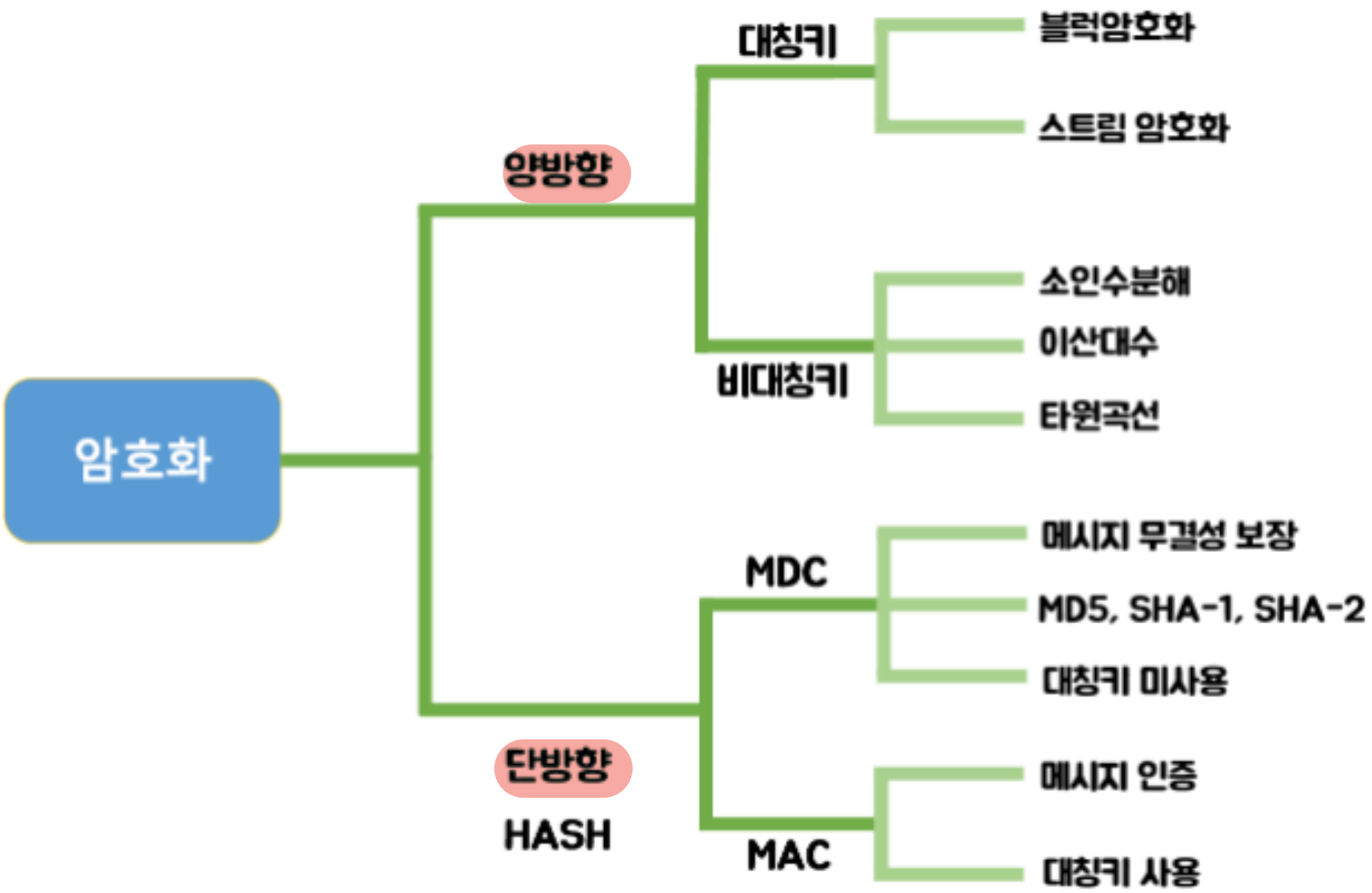
[암호화 알고리즘 종류]



- 양방향 알고리즘: 복호화 가능(대칭키 암호화, 비대칭키 암호화)
- 단방향 알고리즘: 복호화 불가

[대칭키 암호 알고리즘]

대칭키 암호 알고리즘은 송신자와 수신자가 같은 키를 보유하고 그 키를 통해 송신자가 평문을 암호화해서 전송하면
수신자가 같은 키로 복호화하는 방법을 이야기해. 다음 그림을 보면 이해가 더 쉬울거야!

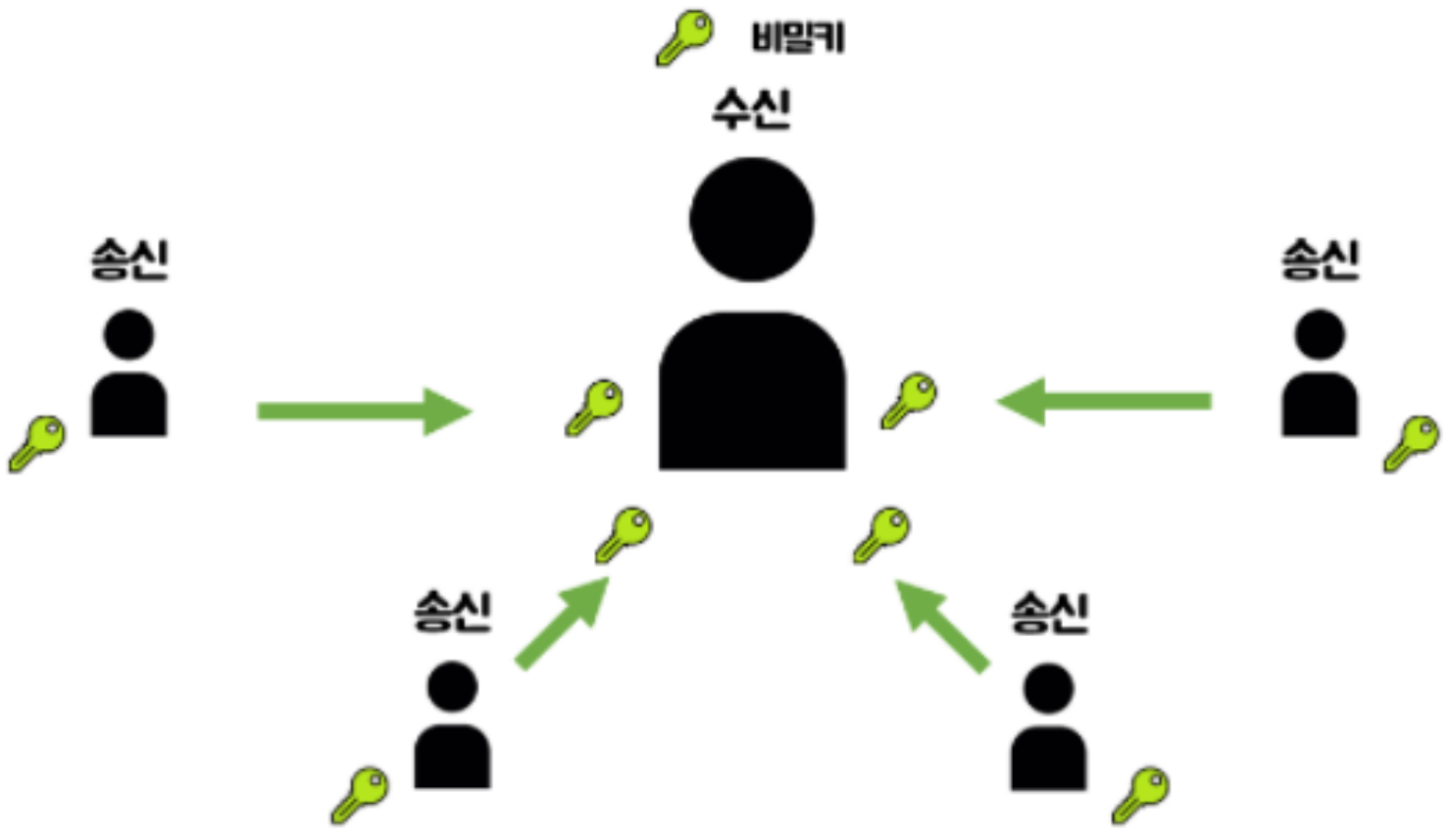
대칭키 작동순서

1. 송신자는 수신자와 같은 비밀키로 평문을 암호화 한다.
2. 송신자는 암호화된 메시지를 상대방에게 전달한다.
3. 수신자는 송신자와 같은 비밀키로 암호화된 메시지를 해독해서 평문을 얻는다.

특징으로는,
- **암호화키와 복호화키가 같다.**
- **속도가 빠르다.**

정도가 있지.

그러나 키를 교환해야한다는게 문제인 것 같아.
키를 교환하는 중 키가 탈취될 수 있는 문제도 있어.



수신자가 관리해야할 키가 방대하게 많아짐

게다가 사람이 증가할수록 전부 따로따로 키교환을 해야하기 때문에 관리해야 할 키가 방대하게 많아진다는 점도 문제점이라고 생각해.

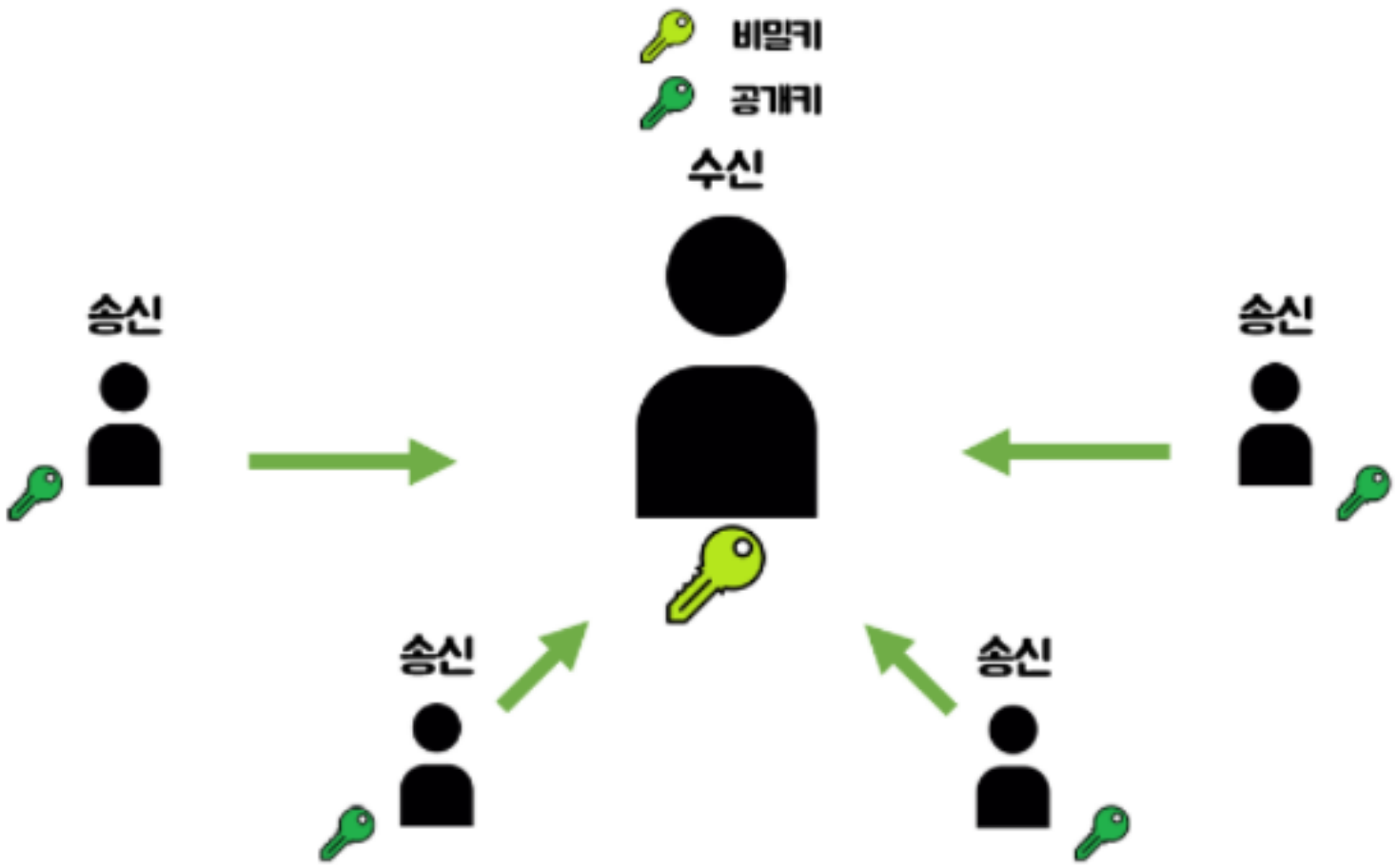
[비대칭키 암호 알고리즘]

공개키 암호 알고리즘이라고도 불리우는 방법이야.
이것도 우선 그림을 먼저 보고 이야기해보자.

비대칭키 작동순서

1. 송신자는 수신자의 공개키를 구한다.
2. 송신자는 수신자의 공개키로 평문을 암호화한다.
3. 송신자는 암호화된 메시지를 상대방에게 전달한다.
4. 수신자는 "자신의 비밀키로" 암호화된 메시지를 해독해서 평문을 얻는다.

특징으로는,
암호화와 복호화에 사용하는 **키가 서로 다르다.**
키를 교환할 필요가 없다.
↑ 비밀키를 교환할 필요가 없음.



비대칭키 형식은 암호화 할 때 수신자의 공개키를 가져다 사용하므로 각각 비밀키를 다 갖고 있지 않아도 된다는 장점이 있는 것 같아!

다만 단점으로는 대칭키 암호화에 비해서 속도가 굉장히 느리다는 점이 있어!

다들 비밀번호를 잊어버려서 찾아본적 있겠지?
그럴 때 대체 비밀번호는 발급해주는데, 왜 원래 비밀번호를 알려주진 않을까?

그건 바로 비밀번호는 단방향 알고리즘으로 암호화해서 저장하기 때문이야.
데이터베이스에 저장된 비밀번호는 사실, 입력한 그대로의 번호가 아닌
암호화 된 문자로 저장되어있다는 거지.

이 말은 곧, 서버도 사용자의 비밀번호가 무엇인지 알 수 없다는 말이야.

서론이 좀 길었네. 이번포스팅은 단방향 암호화 알고리즘, 해시함수에 대한 글이야.
앞서 양방향 알고리즘에 대해서 알아봤었지? 잠깐 단방향 알고리즘을 보고 가자.

양방향 알고리즘과는 다르게 단방향 알고리즘은 복호화가 불가능해.
따라서 암호화한 데이터의 원본 값을 알 수가 없지.

먼저 해시함수가 무엇인지부터 살펴보자!

[해시 함수 Hash Function]

해시함수는 임의의 길이 데이터를 고정된 길이의 데이터로 매핑하는 함수야.
해시함수에 의해 얻는 값을 해시값, 해시 코드, 해시 체크섬 등 이라고 부르며, 간단하게 해시라고 도 해.
이런 해시함수는 결과값을 가지고 원래의 원본 데이터를 알아내기 힘들다는 특징이 있는데
이런 특징을 이용해서 암호화할 때 사용하지.

[안전한 해시함수] (Secure Hash Function)

암호화된 데이터는 쉽게 복호화되면 안되겠지?
다음은 안전한 해시함수가 되기위해서 충족해야하는 조건들이야.

1. **역상 저항성**
   해시 값을 생성하는 입력값을 찾는 것이 계산상 어려워야한다.

2. **제 2역상 저항성**
   동일한 해시값(y)이 나오는 다른 입력값(x')을 찾는 것은 계산적으로 불가능해야한다.

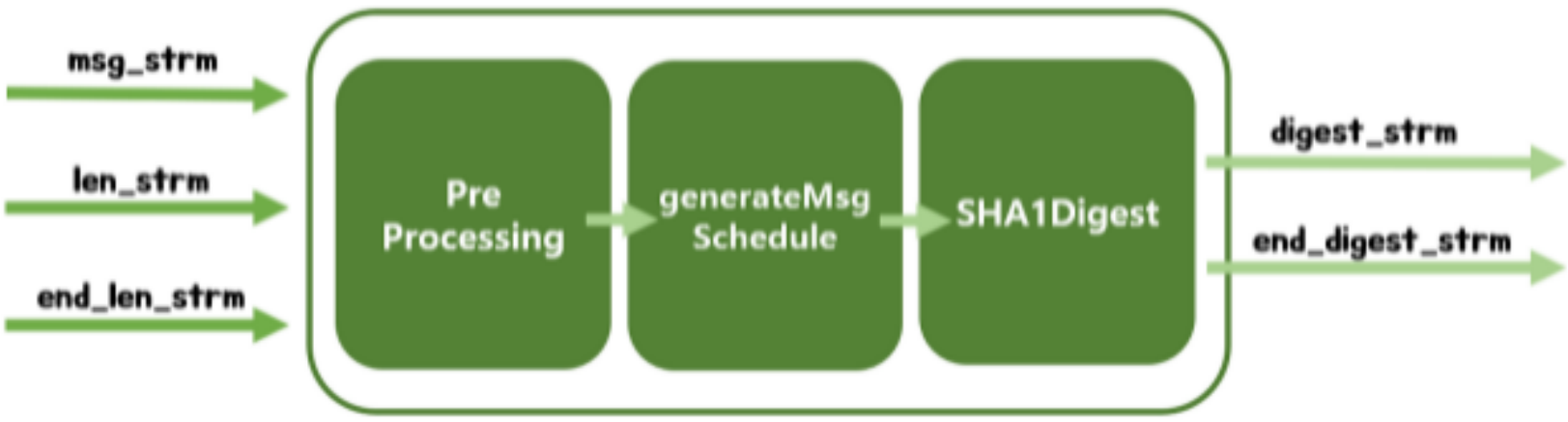
3. **충돌 저항성**
   동일한 해시 값이 나와서는 안된다.

다음으로는 SHA해시함수를 좀 알아볼까해.
다만 암호화하는 방법을 쓰기에는 너무 복잡하기도 하고,
실제로 암호화 원리를 아는 것이 소용있을까싶어서 소개 정도만 해볼거야!

[SHA-1]

Secure Hash Algorithm-1

> SHA-0을 변형한 SHA 함수들 중 하나이며, 1995년 발표되었고 SHA 함수들 중 가장 많이 쓰이고 있는 해시함수



SHA-1 내부구조(digest는 암호문이라고 이해하면 돼)

SHA-1은 1995년 미국 국가안보국 설계한 암호학적 해시 함수들의 모음이야.
264비트의 메시지로부터 160비트의 해시값을 만들어 내고
입력 메시지는 512bit *패딩을 적용하는 방식을 사용해.
*패딩: 100개의 공간에 문자 1개가 들어오면 나머지 99개의 공간을 숫자 0 등으로 메우는 작업

문제점

- 충돌쌍 문제: 두 데이터에 대한 해시를 계산하였더니 같은 결과값이 도출되는 것.

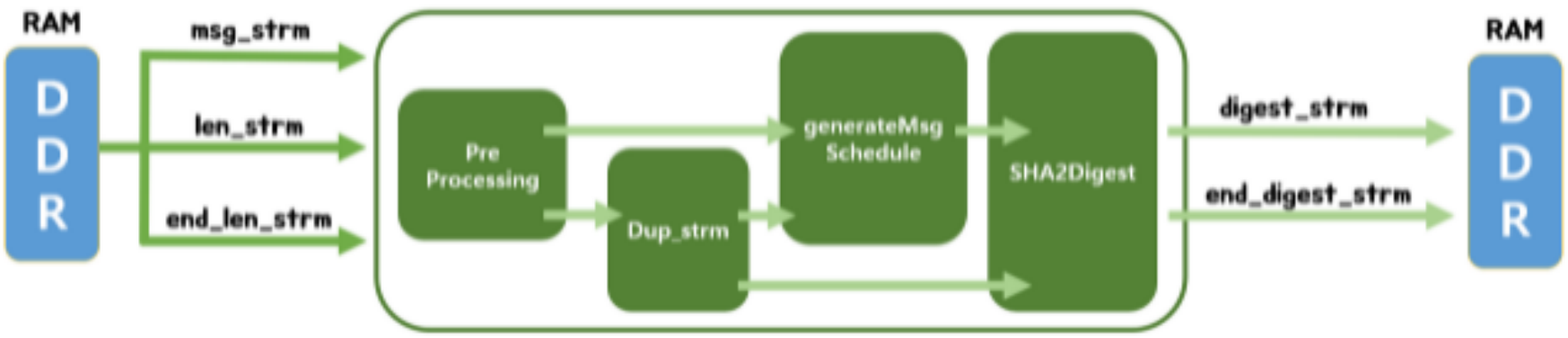
충돌쌍 문제를 악용한다면 시스템 내부의 파일을 제멋대로 교체해버리고서는 마치 기존에 존재하던 파일인 것처럼 둔갑시킬 수 있어.
보안상 상당히 치명적인 문제라고 볼 수 있는 것이지.

이러한 문제점 때문에 SHA1사용이 지양되고 SHA2를 권장하고 있어.

[SHA-2]

Secure Hash Algorithm-2

> 미국국가안전보장국(NSA)이 2001년에 설계한 암호화 해시 함수들의 집합



SHA-2 내부구조

SHA1을 대체하는 해시암호야. SHA-2에는 여러 종류 알고리즘이 있어.
SHA-224, SHA-256, SHA-384, SHA-512...
이것들을 통칭해서 SHA-2라고 명칭하고 있지.
수학적으로는 충돌 가능성이 존재하지만 공학적으로는 그 가능성은 없다고 봐도 무방해.

특징으로는,
해시값에 대해 계산된 해시를 비교함으로써 사람이 **데이터의 무결성을 파악**할 수 있다는 점이야.
이게 무슨 말이냐면,

데이터의 정확성과 일관성을 유지하고 보증하는 것

다운로드한 파일의 해시를 계산한 다음 이전에 게시한 해시 결과물의 결과와 비교하면
다운로드한 파일이 수정 또는 조작되었는지 알 수 있다라는 거지.

종류

SHA-2 암호의 종류 몇 가지만 살펴볼게.

SHA-256

- 현재 블록체인에서 가장 많이 채택하여 사용되고 있는 암호 방식
- SHA-512보다 훨씬 빠르게 64개의 해시 생성
- 안정성 문제에서도 큰 단점이 발견되지 않음
- 256 비트(bits)의 축약된 메시지
- 속도가 빠름

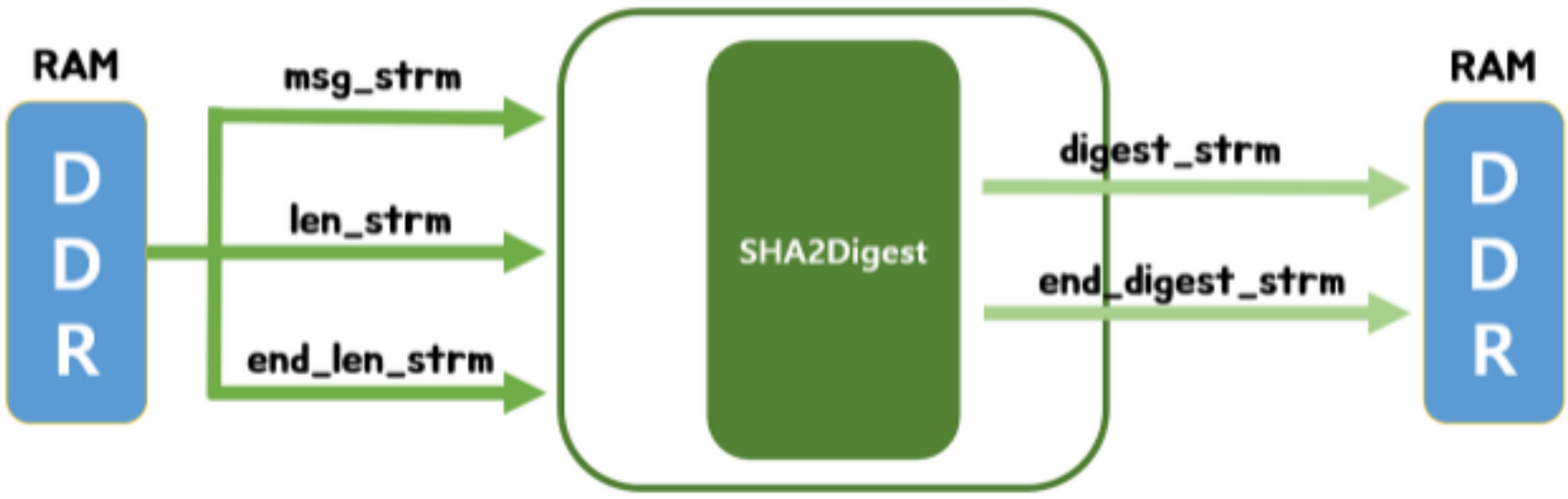
SHA-512

- 512비트(64바이트) 해시 값을 생성
- 길이 확장 공격에 대해 취약
- 결과 값이 512비트로만 나와서 용량을 많이 차지

2004년부터 암호 해시 알고리즘에 대한 공격이 시작되었어. 이에 따라 더 강력한 알고리즘을 개발하기 시작했지.
바로 **SHA-3** 이야.

[SHA-3]

SHA-1과 SHA-2를 대체하기 위해 미국 국립표준기술연구소(NIST)가 2015년 8월 5일에 발표한 암호화 해시함수



SHA-3 내부구조

사실 SHA-1과 SHA-2에는 동일한 수학적 오류가 있었어. 그 오류를 해결한게 바로 SHA-3 암호야.

특징으로는,

- SHA-2가 출력할 수 있는 메시지 해시값의 크기를 모두 출력
- 충돌 저항성, 역상 저항성, 제2역상 저항성을 모두 갖춤
- SHA-2가 사용되는 곳에 SHA-3를 바로 적용 가능
- 높은 수준의 병렬 구조를 가지고 메모리 접근해서 효율성이 아주 좋음

문제점

이렇게보면 SHA-3는 기존 해시함수들 중 가장 뛰어나고, 안전해보여.
그런데 생각보다 많은 시스템이 SHA-2 함수에서 SHA-3로 마이그레이션을 하지 않았더라구.
SHA-3는 다음과 같은 문제가 있었어.

1. SHA-3을 지원하는 소프트웨어 또는 하드웨어가 사실상 없음
2. SHA-1,2 보다 느린 속도

따라서 아직까지도 SHA-2를 보편적으로 사용하고 있어.