©Copyrights 2022 Eom, Hyeonsang All Rights Reserved

Contents

1 지갑 (Wallet) 정의

1) 지갑이란?

> 정의

- 블록체인에서의 암호화폐의 잔액을 나타내고, 화폐를 전송 또는 스마트 컨트랙트를 호출할 수 있는 서비스
- 암호화폐는 직접 화폐를 보관하는 것이 아닌 블록체인에서 지갑 주소에 얼마의 잔액를 가지고 있는지에 대한 기록을 받아오는 것

> 특징

- 공개키는 자신의 계좌 번호, 개인키는 계좌 비밀번호 역할
- 실제 저장하고 보호해야 하는 것은 암호 화폐가 아닌 주소에 접근하는 개인키
- 키의 보관 방법은 중요한 요소 => 지갑 관리 애플리케이션 등장

1 지갑 (Wallet) 정의 (Cont'd)

> 지갑 설계에 따른 주의사항

- **프라이버시와 편의성의 Trade-off**
  - 하나의 개인키를 가지고 이를 트랜잭션 서명에 재사용
    - » 간단하게 사용이 가능함
    - » 모든 트랜잭션이 연결되어 추적 및 관찰 가능함 (프라이버시 문제)
  - 하나의 트랜잭션마다 각기 다른 개인키로 서명
    - » 키 생성 및 관리의 어려움 (낮은 편의성)
    - » 모든 트랜잭션들의 관련이 없어 추적 및 관찰 등이 복잡함

1 지갑 (Wallet) 정의 (Cont'd)

2) 지갑 종류

> 비 결정적 지갑 (Non-deterministic Wallet)

- 무작위로 개인키 생성하여 사용함
- 무작위의 개인키 생성으로 인해 지갑의 데이터 백업 및 관리 등이 복잡함

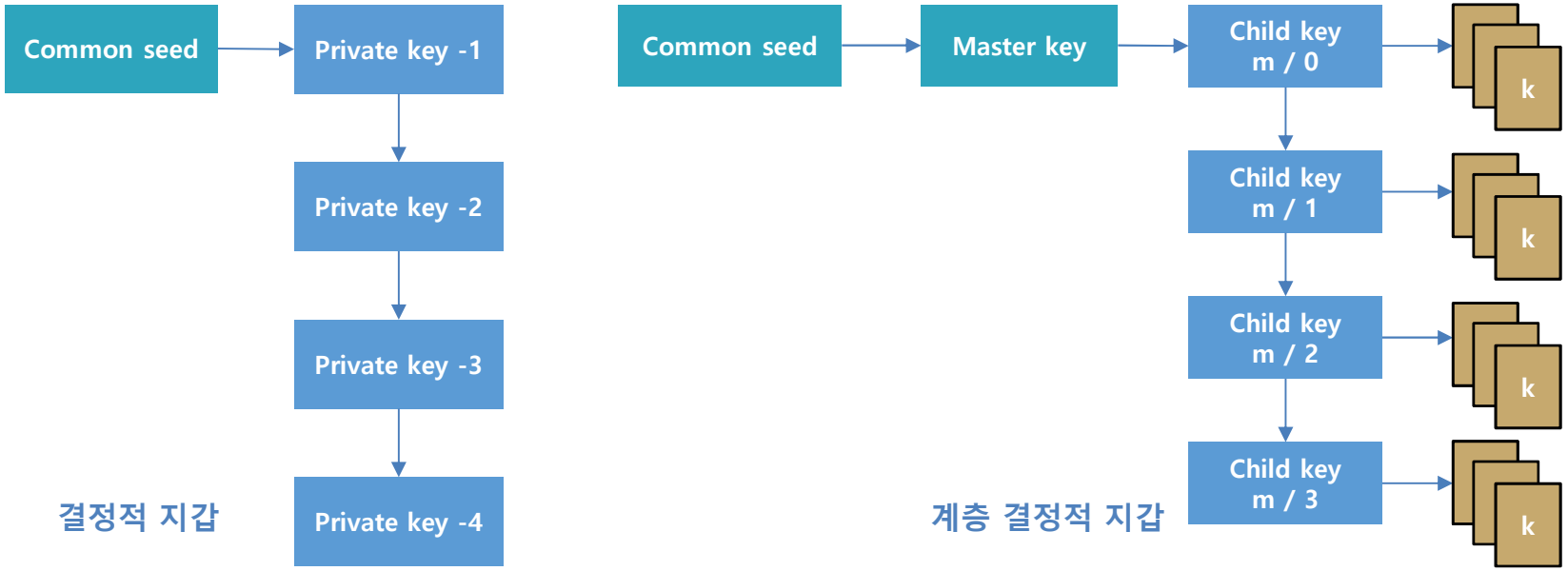
> 결정적 지갑 (Deterministic Wallet)

- Common seed로부터 일방 해시 함수를 이용하여 얻은 개인키를 사용함
- Common seed만 있으면 seed로 인해 생성된 개인키 전부를 복원할 수 있기 때문에 백업을 한 번만 진행함
- 지갑의 export, import도 common seed만 있으면 가능하기 때문에 다른 종류의 지갑 사이에서도 키 전부가 쉽게 이동 가능함

1 지갑 (Wallet) 정의 (Cont'd)

> 계층 결정적 지갑 (Hierarchical Deterministic wallet)

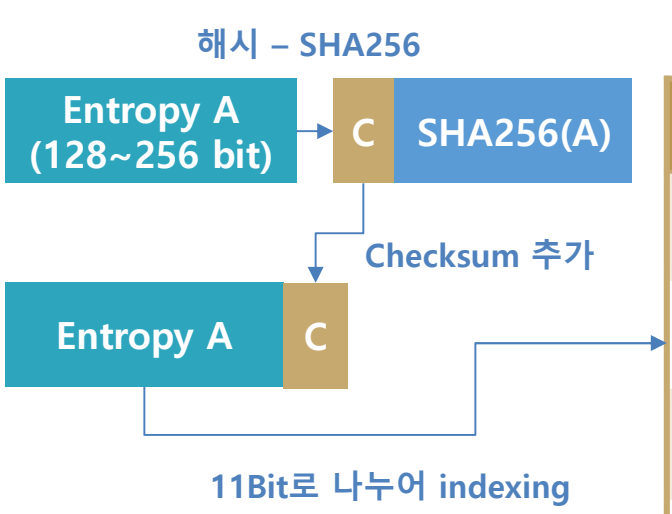
- 계층 결정적 지갑은 트리 구조에서 생성된 키를 담고 있으며,
  부모키가 자식키들을 만들어낼 수 있고, 각각의 자식키가 손자키열을 만듬



1 지갑 (Wallet) 정의 (Cont'd)

> 니모닉 코드 (Mnemonic code)

- **Common seed를 복원하기위해 단어 열을 백업하는 방식**



C : Checksum

Index	Wordlist
0	가격
1	가끔
…	..
2047	힘껏

1. 128~256비트 무작위 난수 생성 (A)
2. A를 SHA256으로 해시한 값의 앞에 비트를 체크섬으로 생성
3. 무작위 난수 A에 끝에 체크섬 추가
4. "무작위 난수 + 체크섬"을 11비트 단위로 분리
5. 각각의 11비트의 값을 미리 정의된 2,048단어와 매핑하여 니모닉 코드 생성

참고 : https://github.com/bitcoin/bips/blob/master/bip-0039.mediawiki

1 지갑 (Wallet) 정의 (Cont'd)

> 니모닉 코드 (Mnemonic code)

- **엔트로피와 단어 길이의 관계도**

Entropy (Bit)	Checksum (Bit)	Entropy + Checksum	Word Number
128	4	132	12
160	5	165	15
192	6	198	18
224	7	231	21
256	8	264	24

참고 : https://github.com/bitcoin/bips/blob/master/bip-0039.mediawiki

1 지갑 (Wallet) 정의 (Cont'd)

> 니모닉 코드로부터 common seed 생성방법

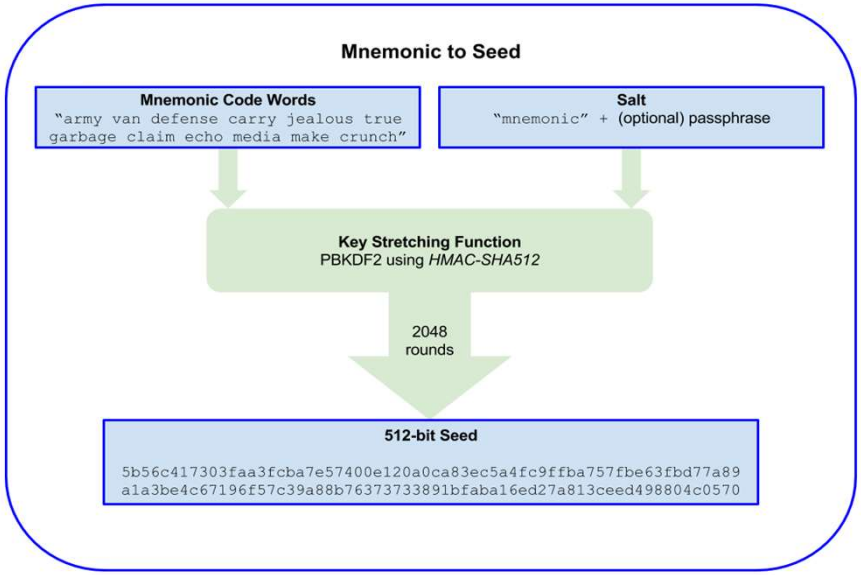
6. PBKDF2 키 스트레칭 함수의 첫번째 파라미터로 니모닉 사용함

- PBKDF2 (Password-Based Key Derivation Function 2)
  - » RSA 연구소의 공개 키 암호화 표준(PKCS)의 #5 2.0 문서에 기술됨
  - » IETF(Internet Engineering Task Force)에서 제공하는 인터넷 표준 문서 RFC 2898에 기술됨
  - » 암호화 해시 함수 표준임
- 키 스트레칭(key stretching)은 입력한 패스워드(니모닉 코드)에 대해 특정 해시 함수를 통해 다이제스트를 생성하는 것을 재귀적으로 반복함을 의미함

7. PBKDF2 키 스트레칭 함수의 두번째 파라미터로 솔트를 사용함

- 솔트 (salt) = "mnemonic" + 암호문(옵션)
  - » 단방향 해시 함수에서 다이제스트를 생성할 때 추가하는 임의의 문자열
  - » Salt 매개변수의 목적은 brute-force attack을 못하게 보호하는 것 (2^512)

지갑 (Wallet) 정의 (Cont'd)



8. **PBKDF2는 최종 출력으로 512비트 값을 만드는 HMAC-SHA512 알고리즘을 통해 2,048 해시 라운드를 사용하여 니모닉과 솔트 파라미터로 확장하며, 해당 결과로 나온 것이 512비트 값이 seed로 사용됨**

```
Derived key = PBKDF2(
  Pseudorandom function,
  Password,
  Salt,
  The number of iteration,
  length of Derived key
)
```

>

```
Seed = PBKDF2(
  HMAC-SHA512,
  {Mnemonic code words},
  "mnemonic" + {optional passphrase},
  2048,
  512
)
```

1 지갑 (Wallet) 정의 (Cont'd)

> 니모닉 단어 == 브레인월렛 (Brainwallet) ?

- **브레인월렛**
  - 사용자(인간)가 고른 단어로 구성됨

- **니모닉 단어**
  - 지갑에 의해 무작위로 생성하여 사용자에게 전달함

인간은 무작위성에 매우 약한 존재임으로
브레인월렛을 사용하는 것은 **보안적으로 매우 취약**할 수 있음

1 지갑 (Wallet) 정의 (Cont'd)

2) 지갑 종류

> **Hot wallet : 온라인에서 동작하는 Wallet**
- 웹, 데스크톱 등 소프트웨어의 클라이언트 형태
- 네트워크에 연결되어 입출금 및 송금이 실시간으로 가능
- 키가 네트워크에 노출될 가능성이 있어 보안에 취약

> **Cold wallet : 오프라인에서 동작하는 Wallet**
- USB wallet, Hardware wallet, Paper wallet

1 지갑 (Wallet) 정의 (Cont'd)

> Hot wallet vs Cold wallet

	Cold wallet	Hot wallet
인터넷 연결	차단(오프라인)	허용(온라인)
보안성	높음	낮음
사용성	불편 / 분실 취약	편리 / 해킹 취약
사용 목적	고액, 장기 투자	소액, 단기 투자

1 지갑 (Wallet) 정의 (Cont'd)

3) Hot wallet

> 데스크톱

- 비트코인 : Bitcoin Core, Bitcoin Knots 등 / 이더리움 : Mist 등

> 웹

- 비트코인 : coin.space 등 / 이더리움 : MyEtherWallet 등

> 모바일

- 비트코인 : Bitcoin wallet 등 / 이더리움 : Jaxx 등

> 참고

- 비트코인 : https://bitcoin.org/en/choose-your-wallet
- 이더리움 : https://coinsutra.com/best-etherum-wallets

1 지갑 (Wallet) 정의 (Cont'd)

4) Cold wallet

> Paper wallet

- 개인키와 공개키를 QR 코드와 텍스트 형식으로 종이에 출력한 형태
- Myetherwallet 등 다양한 사이트에서 Paper wallet 제작 지원
- 분실 시 복구 불가능

> USB wallet

- 개인 키와 공개 키를 USB에 저장한 형태
- 바이러스에 감염 또는 네트워크에 연결된 PC에 사용시 보안 위험

1 지갑 (Wallet) 정의 (Cont'd)

그림 출처: My Ether Wallet

1 지갑 (Wallet) 정의 (Cont'd)

> Hardware wallet

- 자체 보안 솔루션을 가지고 있으며, 개인키를 분리 시켜놓은 형태
- 지갑에 접근을 위해서 기기에서 고유 PIN 번호로 숫자를 매번 입력하도록 설계
- 물리적 탈취를 제외한 다른 해킹에 안전
- 분실을 대비하여 지갑을 최초 생성시 기기 화면에 출력되는 니모닉 코드 (Mnemonic Code) 12~24개의 단어들을 별도로 보관
  (분실 시 해당 단어들을 가지고 키 복구 가능)

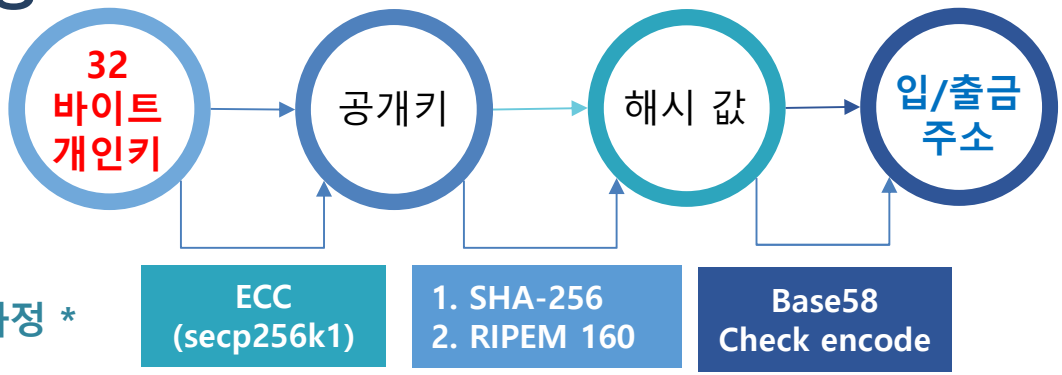
1 지갑 (Wallet) 정의 (Cont'd)

그림 출처: https://cryptalker.com/bitcoin-affiliate/

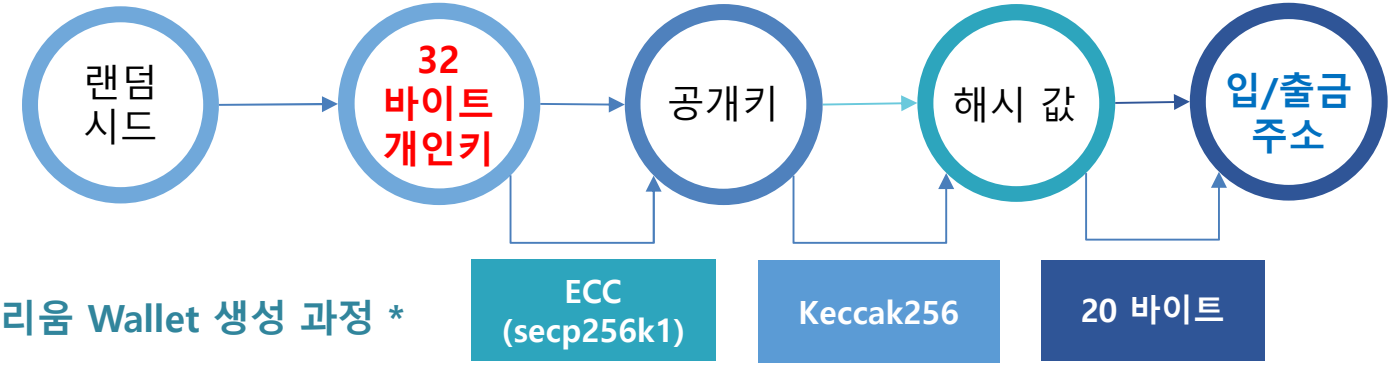
2 지갑 동작 과정

1) 지갑 주소 생성과정

* ECC(Elliptic Curve Cryptography) :
타원곡선 이론에 기반한 공개 키 암호 방식



*** 비트코인 Wallet 생성 과정 ***



*** 이더리움 Wallet 생성 과정 ***

2 지갑 동작 과정 (Cont'd)

1) 지갑 주소 생성과정

2 지갑 동작 과정 (Cont'd)

2) 트랜잭션 서명 및 전송 과정

> 거래 생성

- 거래 생성에서 돈을 송금하기 위한 의사를 표시함
- 거래 생성자와 거래 서명자는 동일인일 필요가 없음
- 암호화폐의 소유주가 해당 거래에 서명하고 나면, 해당 거래는 유효화되고 암호화폐 송금에 필요한 모든 정보가 담기게 됨

> 거래 전송

- 해당 거래가 블록체인에 포함되기 위해서는 거래가 블록체인 네트워크에 전달되어야 함
- 블록체인 거래에는 기밀 정보, 개인키, 인증서 등이 없고 거래의 서명만이 존재함

2 지갑 동작 과정 (Cont'd)

> 거래 전파

- **해당 거래가 블록체인 네트워크에 연결된 하나의 노드로 전송되면 해당 노드에 의해 거래가 검증이 진행됨**
  - 거래의 유효성 검증을 마치고 나면, 해당 노드와 연결된 다른 노드들로 거래를 전파하고 동시에 성공 메시지가 거래 전송자에게 전달됨
  - 거래가 유효하지 않은 경우, 해당 노드는 거래 실패 메시지를 생성자에게 전달됨
- **유효한 거래는 주변 3~4개 노드들로 전송되고 이 과정이 반복되어 수 초안에 노드들 전체에 해당 거래가 전파됨**
- **모든 노드들은 독립적으로 모든 거래를 검증하며, 유효하지 않은 거래는 한 노드 이상 전파될 수 없음**

3 블록체인 지갑 문제점

1) 지갑 (개인키) 분실

> 개인 개인키 분실

- **확률상 거의 복구 불가능**

2) 해킹

> 거래소 Hot wallet 해킹

- **2018.6 - 국내 '빗썸' 350억원 규모 해킹**
  - 외주 직원 PC에 악성 코드를 활용한 해킹 (Hot wallet 개인키 탈취 추정)
- **2018.1 - 일본 '코인 체크' 5648억원 규모 해킹**
  - 거래소의 보안 관리 소홀로 인한 Hot wallet 해킹 사건

참고 뉴스 : https://platum.kr/archives/101362

3 블록체인 지갑 문제점 (Cont'd)

가상통화 거래소 주요 해킹 피해 사례 자료: 한국은행, 금융보안원

시기	국가	거래소	피해 규모
2014년 2월	일본	마운트곡스	470억엔 비트코인 분실로 파산, 고객 아직 보상 못 받아
2015년 1월	슬로베니아	비트스탬프	500만달러 비트코인 도난
2016년 8월	홍콩	비트파이넥스	약 7000만달러 비트코인 도난, 고객 36.1% 손실
2017년 4월	한국	야피존	전자지갑 해킹으로 55억 비트코인 도난, 고객 37.1% 손실
2017년 6월	한국	빗썸	직원 컴퓨터 해킹으로 3만6천명 고객 정보 유출
2018년 1월	한국	유빗(야피존 후신)	자산 17% 손실, 파산선언 뒤 영업재개 검토
2018년 1월	일본	코인체크	580억엔 넴(NEM) 도난, 엔화로 보상 추진

대부분 Hot Wallet 이 해킹 당한 사건

그림 출처 : 한국은행, 금융보안원

3 블록체인 지갑 문제점 (Cont'd)

3) 지갑 문제점 해결 방안

> 한국블록체인협회 자율 규제안 발표 (2017년 12월)

- 각 거래소는 보유한 암호 화폐 자산 중 70% 이상을 네트워크가 차단된 Cold wallet에 보관하도록 권고됨

> Cold wallet 현황

빗썸	확인 불가
업비트, 바이낸스	확인 불가
후오비	거래소의 98%에 해당하는 암호 화폐를 Cold wallet 보관
코인원	주요 거래소 해킹 사건 이전부터 Cold wallet을 사용 중이며, Cold wallet 보관 비중을 꾸준히 상향 중
코빗	고객 자산의 최대 80%까지 Cold wallet에 보관 중

4 개선된 지갑 구조 (Cont'd)

1) BitGo Wallet

> P2SH (Pay To Script Hash) Safe address

- P2SH는 2012년 Bitcoin Improvement Proposal 16 (BIP 16) 에서 소개된 비트코인 주소의 새로운 타입
- 하나의 키를 PC에 보관하고 두번째 키를 서버에 보관
- 해커는 사용자의 PC와 서버를 모두 해킹해야 사용자의 코인에 접근 가능
- 서비스가 다운되어도 손쉽게 복구 가능

> 2-of-3 address (Multi-sig)

- 3개의 Elliptic Curve Digital Signature Algorithm (ECDSA) key를 사용
- 입금은 기존과 동일한 방법을 사용하며, 출금 시 최소 2개 이상의 ECDSA key-pair들을 사용하여 보안 강화

4 개선된 지갑 구조 (Cont'd)

> 사용자 2개의 key-pair들, 서버 1개의 key-pair를 생성

- **Backup key-pair**
  - 자금 복구에만 사용되며, 개인키는 인쇄되고 공개키는 서비스에 보관
- **User key-pair**
  - 개인키는 사용자의 암호로 암호화되고, 암호화된 개인키와 공개키는 서비스에 보관
- **Service key-pair**
  - 개인키는 서비스가 알고있는 암호로 암호화되어 서버에 보관

4 개선된 지갑 구조 (Cont'd)

> 2-of-3 address (Multi-sig) 특징

- **안전성 (Safety)**
  - 이 서비스는 스스로 트랜잭션 생성 불가능

- **편의성 (Convenience)**
  - 사용자는 어떤 PC로든 자신의 fund 접근 가능

- **복원성 (Recovery)**
  - 사용자는 서비스에 문제가 생길지라도 fund 복구 가능

- **복원성 (Recovery)**
  - 사용자 fund의 프라이버시 준수 필수

4 개선된 지갑 구조 (Cont'd)

> 2-of-3 address (Multi-sig) 구현 가정

- **TLS를 통한 모든 통신을 포함하는 서비스 (e.g., 웹사이트)**
- **서비스와 브라우저간의 Coordination**
- **2단계 인증 (2-factor authentication)**
- **강력한 비밀번호 사용**

4 개선된 지갑 구조 (Cont'd)

> 2-of-3 address (Multi-sig) fund 출금

- 서비스 통한 2단계 인증 로그인 (ID-PASSWORD & 모바일 인증)
- 사용자 PC의 암호화된 private key를 사용자의 암호로 복호화
- 출금 transaction을 생성하며 단일 서명으로 사인 후 서비스로 전송
- 서비스에서 유효성 검사, 서비스의 private key 사용 2번째 서명

4 개선된 지갑 구조 (Cont'd)

2) Shamir Secret Sharing

> 의미

- 메시지(S)를 N개의 조각들로 나누고, 그 중 K조각들을 조합하여 메시지(S)를 복원할 수 있는 방식

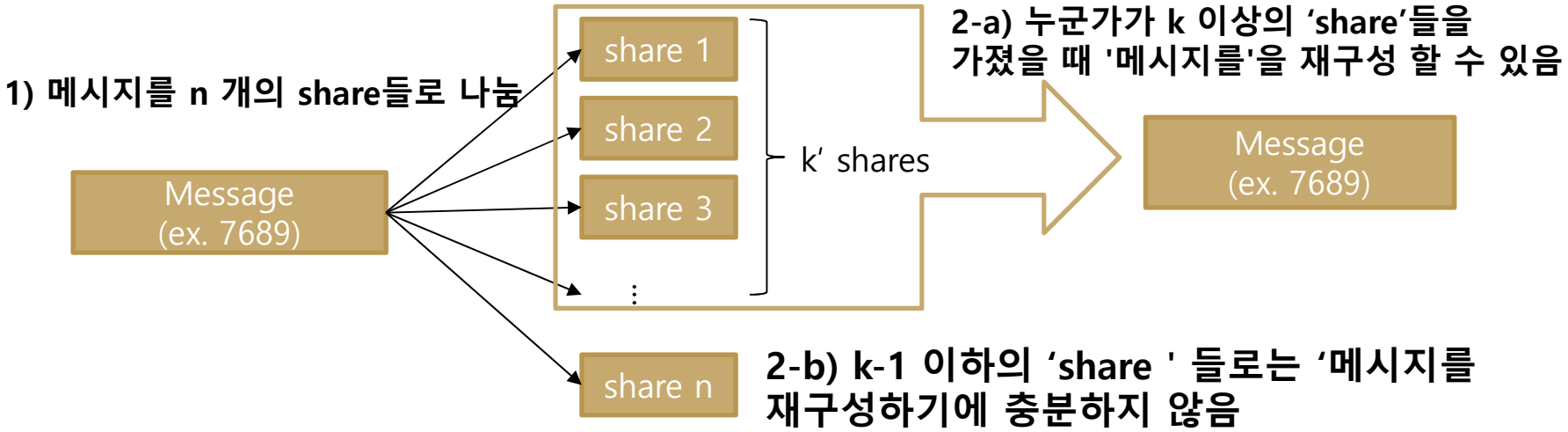
> 방식

- 메시지(S)를 N개의 share들로 분리
- K개의 share들을 가지고 메시지(S) 복원

> 장점

- 여러 개의 키들을 분산시켜 보안성 강화
- 최대 N – K개까지 키들을 분실해도 복원 가능

4 개선된 지갑 구조 (Cont'd)



4 개선된 지갑 구조 (Cont'd)

> 메시지(S)를 N개의 share로 분리

- **2차 다항식에서 계수와 상수를 모두 구해해서 2차 다항식을 찾으려면**
  ⇒ 다항식을 지나는 점들의 좌표가 필요함
  ⇒ 또한, N차 다항식의 계수를 모두 알아내려면 N+1의 점들의 좌표가 필요함

$$f(x) = a_1x + a_2x + S$$

구하려는 계수가 3개 이기 때문에 3개의 점들의 좌표가 필요함

$$f(1) = a_1 + a_2 + S$$

$$f(2) = 4a_1 + 2a_2 + S$$

$$f(3) = 9a_1 + 3a_2 + S$$

이를 3개의 좌표를 대입 후, 연립방정식을 활용하여 a_1, a_2,S를 구할 수 있음

4 개선된 지갑 구조 (Cont'd)

> 메시지(S)를 N개의 share로 분리

- 메시지(S)의 다항식으로 표현하고 이를 N개의 좌표들로 분리하고,
  이 메시지(S)를 가지는 임의의 K-1차 다항식을 생성하면,
  K개의 점들의 좌표를 이용해서 S를 구할 수 있음
- 상수항을 메시지(S)를 가지는 임의의 K-1차 다항식을 생성하려면,
  K개의 좌표들 필요 ⇒ 상수 S를 구할 수 있음 (이후 메시지 복원에서 사용됨)

$$f(x) = a_1 x^{k-1} + a_2 x^{k-2} + \cdots a_{k-1} x + S$$

다항식의 상수항은 메시지 S를 나타냄

즉, $f(0) = S$

$$\mathbf{shares} = \{(1, f(1)), (2, f(2)), \ldots, (n, f(n))\}$$

4 개선된 지갑 구조 (Cont'd)

> K개의 share를 가지고 메시지(S) 복원

- **연립방정식**
- **랑그랑주 보간법 (Lagrange interpolation)**
  - 연립 방정식을 사용하지 않고 다항식을 구하는 방법
    ⇒ 계산 시간이 비교적 짧고 간단하며 자료의 구간에 관계없이 사용 가능함

$$L(x) = \sum_{k=0}^{k-1} y_i l_j(x)$$

여기서 $l_j(x)$는 Lagrange basis polynomial (라그랑주 기초 다항식) 이라고 함

$$l_j(x) := \prod_{f=0, f\neq j}^{k-1} \frac{(x - x_f)}{x_j - x_f}$$ **로 정의됨**

$L(0) = S$ **메시지(S) 복원**

랑그랑주 보간법 참고 자료 : https://en.wikipedia.org/wiki/Lagrange_polynomial

5 분산키 서명

> MPC (Multi-Party Computation) 기반 분산키 서명

보안

공동의 개인키 없이
서명 가능합니다.
한명이 서명한 것과
동일합니다.

복구

분실 시 전체키를 새로
생성하지 않고 나머지
분산 개인키들을 사용해
복구할 수 있습니다.

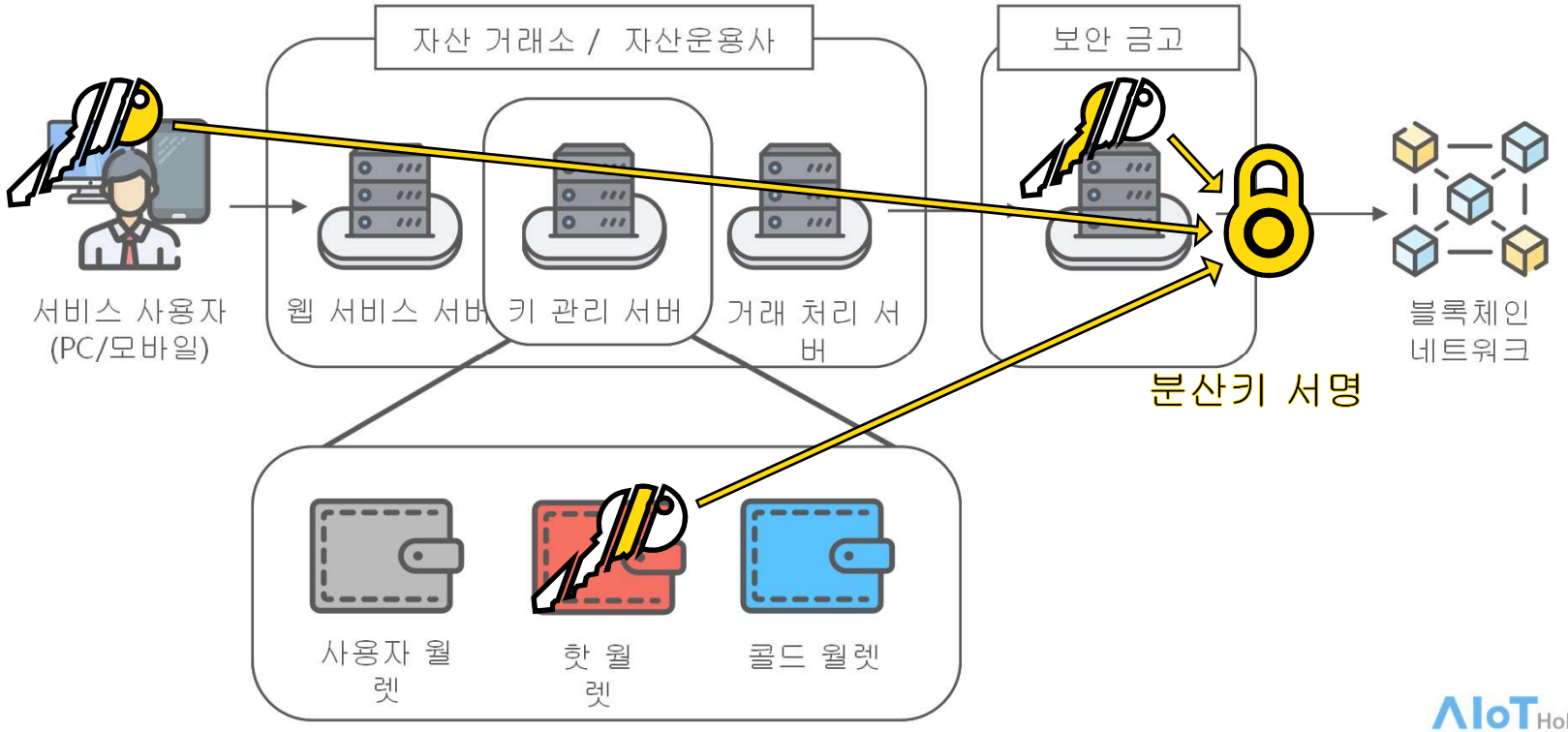
비용 절감

프로세스, 시스템 도입에
필요한 금전적, 시간적
비용 절감할 수 있습니다.

AIoT Holdings

5 분산키 서명 (Cont'd)

> 분산키 기반 암호자산 커스터디 서비스



AIoT Holdings

5 서명 방식 비교

> 단일 서명 vs 분산키 서명

- **단일 지점 암호키 보안의 취약성**
  - 분산키 기술을 통해 단일 암호키가 존재하지 않아 **단일 지점 보안 취약성 해결**
  - 일부 분산 암호키 분실시 **다른 참여자들의 분산 암호키를 활용하여 자산 (암호키) 복구** 가능

> 다중 서명 vs 분산키 서명

- **다중 서명의 이슈**
  - 일반 단일 서명과 동일한 서명을 얻어서 **트랜잭션 크기가 커지는 문제 해결**
  - 단일 서명 정보만을 출력하여 서명에 참여한 **참여자 정보 보호 가능 및 서명 사용자 판단 가능**

AIoT Holdings

4 서명 방식 비교 (Cont'd)

구분	단일 서명	다중 서명	샤미르 비밀 보장 서명
생성되는 키	1	N	N
트랜잭션 요구 서명 수	1	T	T
최종 발생 서명의수	1	T	1
서명데이터의 크기	(상대적으로) 작음	(상대적으로) 큼	(상대적으로) 작음
서명자 식별 여부	O	O	X
플랫폼 종속 여부	독립적	종속적	독립적
트랜잭션 서명 성능	빠름	상대적으로 빠름	상대적으로 느림
보안성	낮음	높음	높음
복구 가능 여부	X	O	O
			

AIoT Holdings

6 정리

> 지갑이란 블록체인에서의 암호화폐의 잔액을 나타내고, 화폐를 전송 또는 스마트 컨트랙트를 호출할 수 있는 서비스임

> 지갑의 종류는 비결정적, (계층) 결정적 지갑이 존재함

> 지갑의 개인키는 해당 지갑 주소의 암호화폐의 전송에 주요 요소이므로 보안성이 매우 중요함

> 지갑의 보안성을 강화하기 위한 방법으로는 Multi-sig, Secret share, Secure MPC (Multi-Party Computation) 등이 존재함

Thank You!

Eom, Hyeonsang (엄현상)

hseom@snu.ac.kr
Department of Computer Science & Engineering
Seoul National University