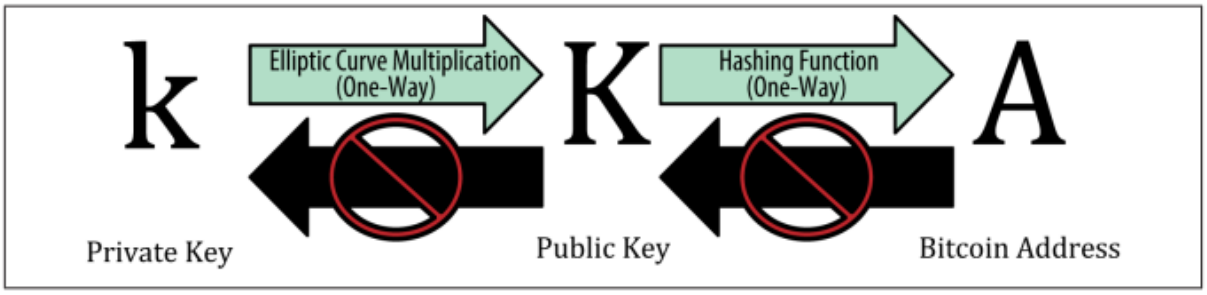
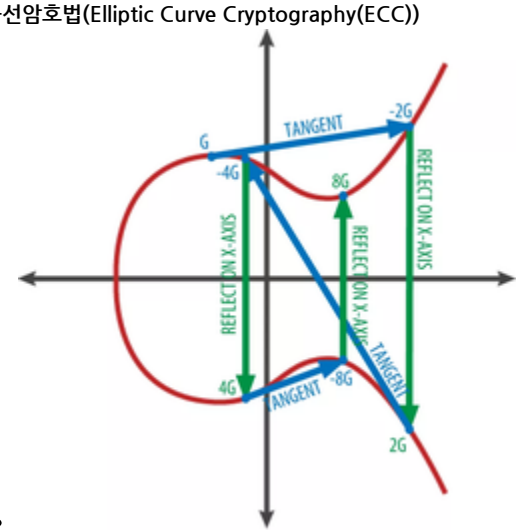
4. 키, 주소, 지갑

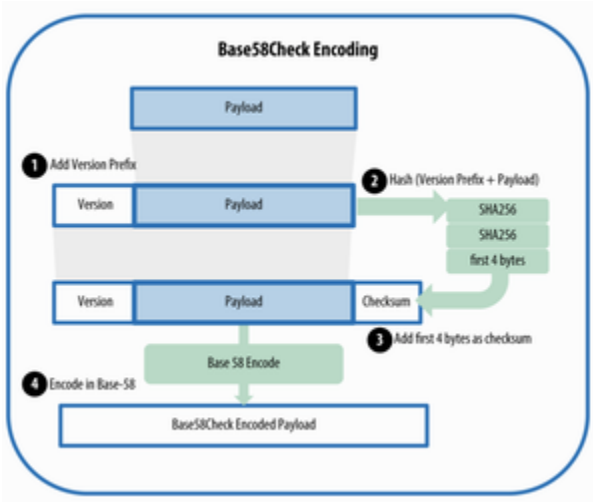
- 키 - 개인키, 공개키, 비트코인 주소



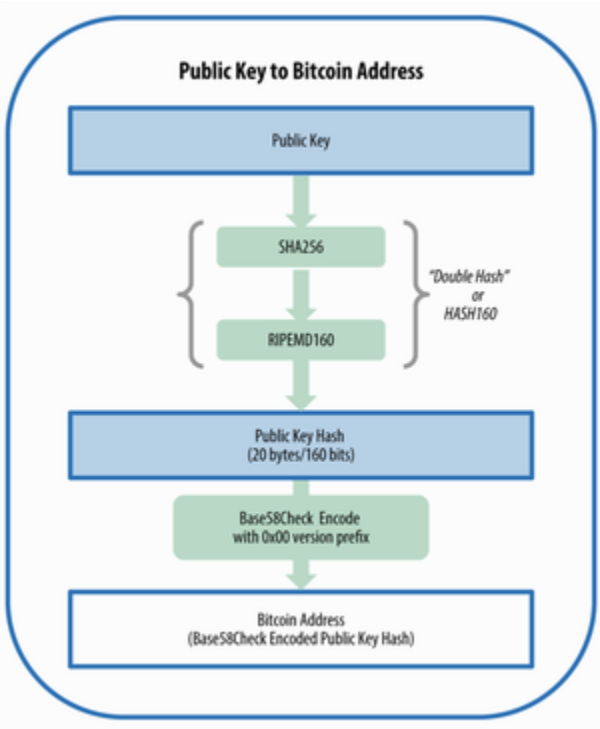
-
- 먼저 알아야 할 개념
  - 타원곡선암호법(Elliptic Curve Cryptography(ECC))



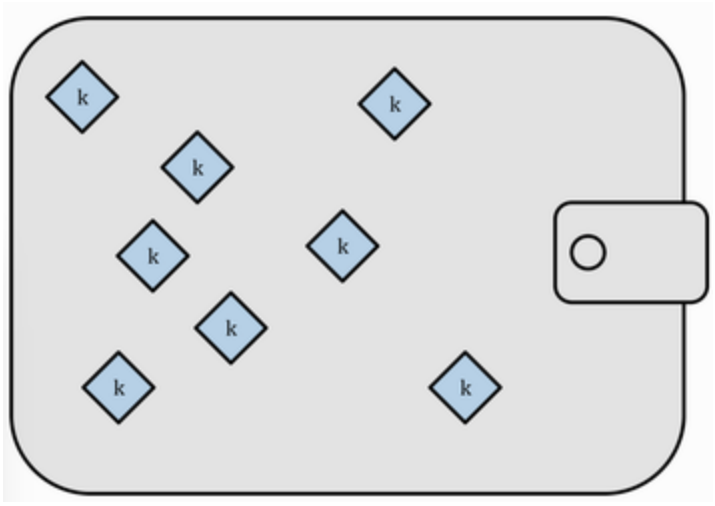
    -
    - y^2 mod p = (x^3 +) mod p
    - 타원곡선함수위의 생성 포인트 G로부터 미분한 직선과 타원곡선함수가 만나는 점에서 X축에 대해서 reflect하면 2G를 구할 수 있다.
    - 이와 같은 방법을 k만큼 반복하여 K = k * G를 통하여 단방향으로 암호화된 K를 얻을 수 있다.
  - Base58Check

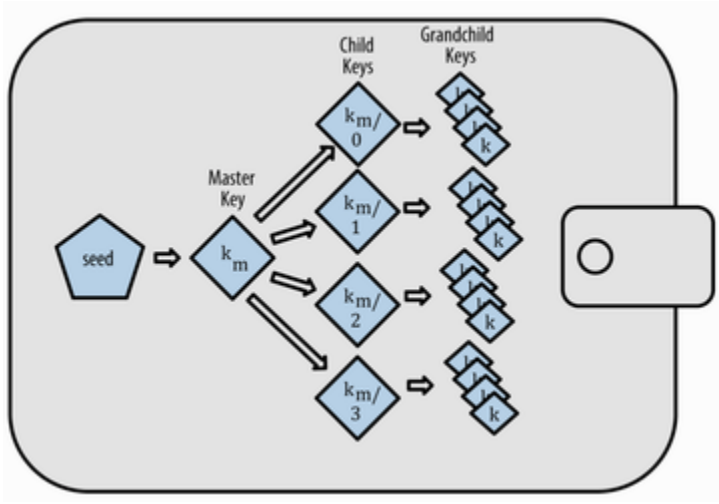


    -
    - 공개키는 그대로 쓰이지 않고, 압축해서 사용한다.
    - Base58 - Base64에서 0, O, l, I, +, / 6개의 문자를 뺀 나머지 58개의 문자열로 압축하는 표현법
    - Base58Check - Base58 포맷으로 압축할 때, 버전과 체크섬을 붙여 데이터의 유효성을 검증하기 위한 압축법
      - payload에 버전 prefix(1바이트)를 붙인다.
      - prefix + payload를 두번 SHA256으로 해싱하여 체크섬을 구한뒤 앞의 4바이트를 prefix + payload + checksum(4바이트) 형태로 붙인다.
      - prefix + payload + checksum 을 Base58 형태로 인코딩한다.
    - 전송 받은 쪽에서는 데이터를 Base58 형태에서 디코딩 한 후에 뒤의 4바이트를 떼고 앞부분을 두번 해싱하여 나온 결과값의 처음 4바이트가 체크섬과 같은지 확인할 수 있다.

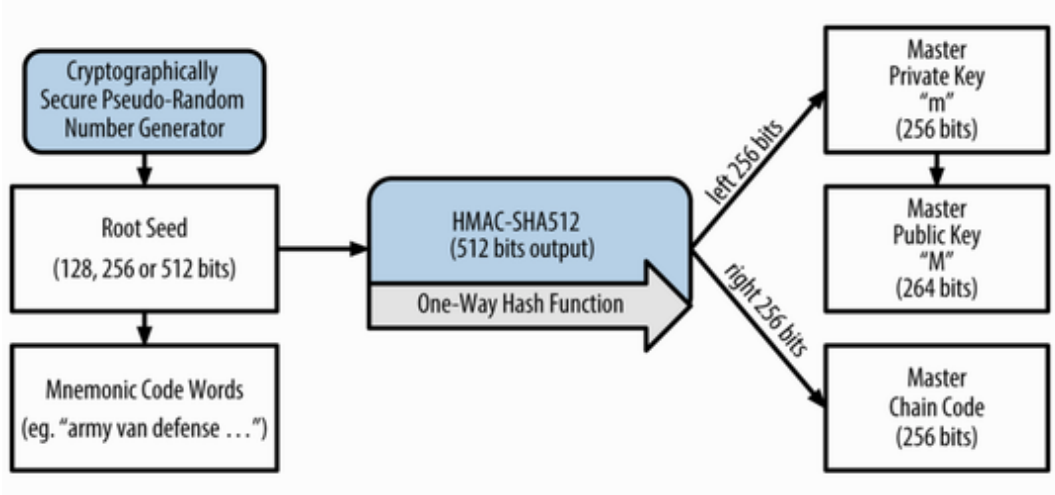
        - 버전 prefix는 고정적이기 때문에 데이터가 무엇을 담고 있는지 확인하기 쉽다. Ex) 1 - 비트코인 주소, 5 - 개인키 WIF
    - 개인키
        - 1 ~ 2^256의 무작위 숫자. 가능하면 seed를 이용해서 생성
        - Wallet Import Format(WIF)
            - 개인키를 위의 Base58Check 인코딩 방법으로 포매팅.
        - 압축형 WIF
            - 실제로는 '압축'되지 않지만, 밑에서 설명할 압축 공개키에 대한 호환성을 위하여 개인키의 뒤에 01이라는 postfix를 붙여서 인코딩
    - 공개키
        - K = k * G
        - K - 공개키, k - 개인키, G - 생성포인트
        - 타원곡선 암호법(ECC)를 이용하여 생성포인트 G로부터 k만큼 계산하여 K를 구한다.
        - 단방향으로는 계산이 쉬우나, 그 반대로는 계산 불가능하다.
        - 공개키 포맷 - 압축/비압축 공개키
            - 비압축 공개키
                - 공개키는 한쌍의 (x, y)좌표 - 접두부(04) 256비트(x) 256비트(y)
            - 압축 공개키
                - 타원곡선 암호법에 의해서 x좌표로부터 y좌표를 정확하게 계산할 수 있으므로 x좌표만 기록한다. 단, x좌표에 대한 y좌표가 최대 2개씩 있으므로 prefix 02, 03을 추가한다.
                - 520비트(8 + 256 + 256) → 264비트(8 + 256) 로 압축
    - 비트코인 주소
        - A = RIPEMD160(SHA256(K))
        - 

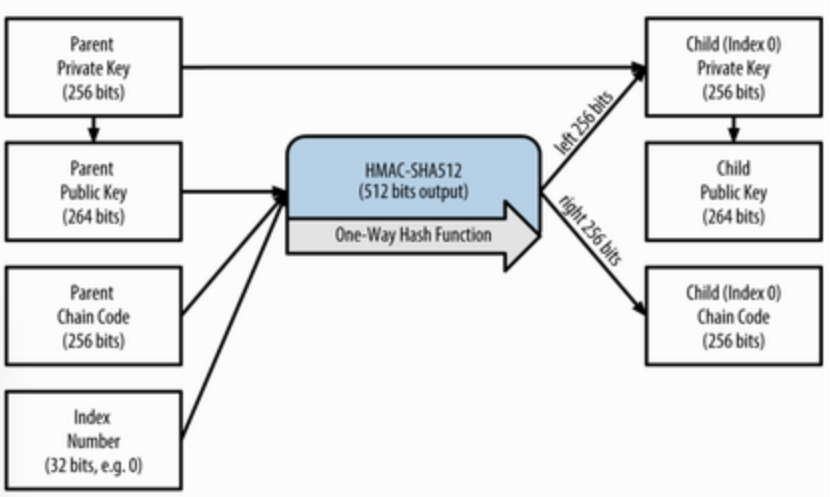
        - 공개키를 두번의 해싱 함수를 돌려서 나온 결과값이 공개키 해시 값이다.
        - 그대로 쓰지 않고, 위에서 설명한 Base58Check 인코딩을 거쳐서 나온 결과값을 비트코인 주소로 활용한다.
- 지갑
    - 비 결정적 지갑 - 각 개인키를 독립적으로 생성하고, 키간에는 연관성이 없다. 지갑을 자주 백업해야 하며, 여러 건의 거래에 대한 주소가 연관된 것으로 보여 프라이버시에 문제가 있을 수 있다.
    - 결정적(종자) 지갑 - 하나의 seed를 기반으로 연관된 개인키들을 생성하는 방법. 초기에 시드를 한번만 백업하면 된다.





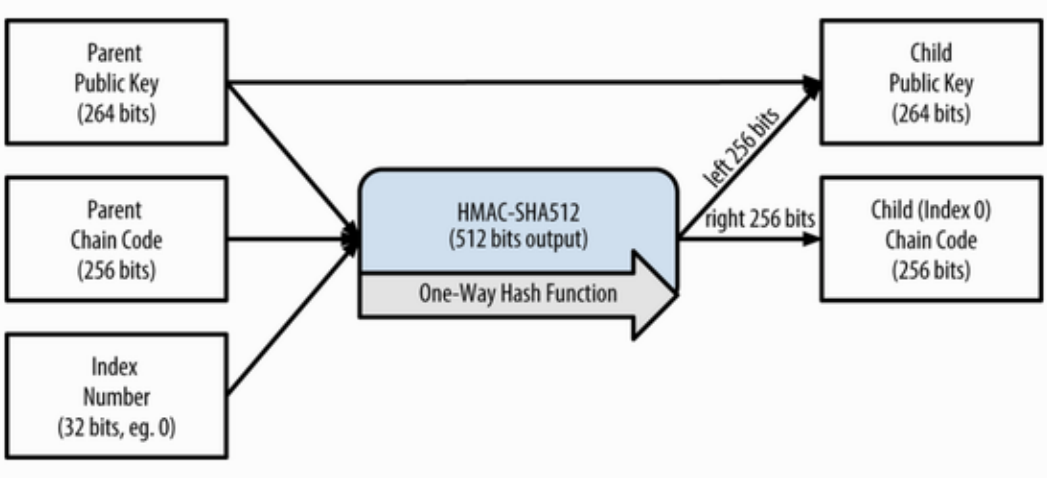
- 연상기호 코드 워드
  - 난수를 인코딩하는 단어 리스트. 단어열을 알고 있으면, 시드를 구할 수 있고 이를 통해 지갑의 키를 모두 복원할 수 있다.
  - 알고리즘
    - 128 ~ 256비트의 무작위열을 해싱하여 첫 몇 비트를 무작위열의 끝에 붙인다.
    - 11비트씩 나누어 미리 정의한 2048개의 단어로 구성된 사전의 문자열의 인덱스로 사용한다.
    - 총 12 ~ 24단어를 생성한다.
- 계층 결정적 지갑(HD wallet(Hierarchical Deterministic wallet) (BIP0032)
  - 개인 자식 키 유도하기





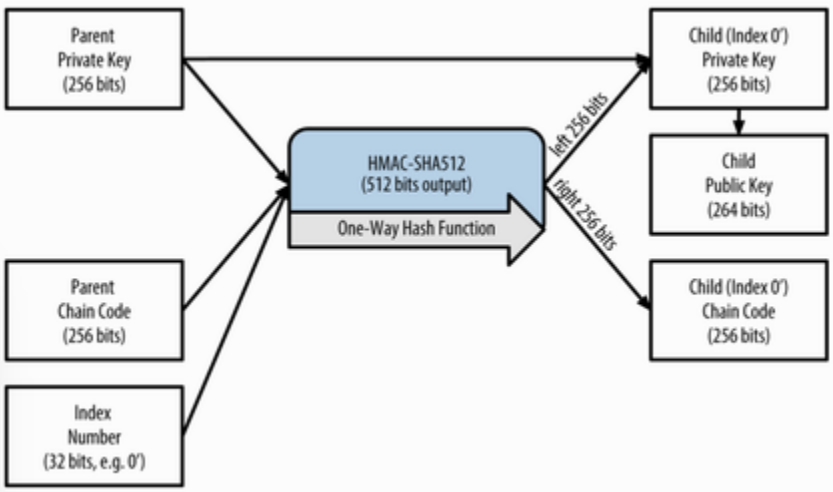
- 임의의 난수를 생성하여 루트 시드로 한다. (이 난수는 위에서 설명한 연상기호 코드 워드로 인코딩 될 수 있다.)
- 루트 시드를 HMAC-SHA512 알고리즘을 통해서 512 비트의 해쉬값을 구한다.
- 해쉬 값의 좌측 256 비트는 마스터 개인키, 우측 256은 마스터 체인코드이다.
- 이후 자식키는 부모의 개인키, 부모의 체인코드, 인덱스 번호를 조합으로 다시 한번 HMAC-SHA512 해싱 알고리즘을 통해서 자식의 개인키와 체인코드를 얻을 수 있다.
    - 자식의 개인키는 부모의 개인키에 위에서 구한 해싱값의 좌측 256비트에 더해진다.

- 공개 자식 키 유도하기



-
    - 개인키를 노출하지 않고도 공개키를 전부 얻을 수 있다.
    - EX) 웹서버, 하드웨어 지갑(종이지갑)

- 단절된 자식 키 유도하기



-
    - 위의 공개 자식키 유도하기를 위해서 체인코드를 필요로 하기 때문에 어떤 경로에서든 개인키를 얻게 되면 모든 개인키가 유출될 수 있다.

        - 부모의 공개키, 부모의 체인코드는 알려져 있음
        - 이 때, 자식의 개인키가 유출 된다면, 부모의 개인키를 찾아낼 수 있음
    - 부모 개인키를 이용해서 자식 개인키를 생성한다.
    - 부모와 자식 간의 트리 구조에서 갭을 만든다.
- 색인 번호
    - 색인 번호는 32비트. 따라서 0 ~ 2^32 -1 까지 가능하다.
    - 0 ~ 2^31 -1 (0x0 ~ 0x7FFFFFFF)는 정규유도. 2^31 ~ 2^32 -1 (~0xFFFFFFFF)는 단절유도에 사용하도록 약속. 단절 유도의 색인은 '로 표기
    - 경로 - 마스터 개인키(m), 마스터 공개키(M)으로부터 유도되어 나온 키의 색인 경로
        - m/0
        - m/0/0
        - m/0'/0
        - m/1/0
        - M/23/17/0/0
    - 미리 규정된 트리 레벨 (BIP0044)
        - m / purpose' / coin_type' / account' / change / adress_index
        - M / 44' / 0' / 0' / 0 / 2 - 비트코인의 첫번째 계좌의 지불에 대한 세번째 수신 공개키
        - M / 44' / 0' / 3' / 1 / 14 - 비트코인의 네번째 계좌의 15번째 잔액을 받는 공개키
- 암호화된 개인키(BIP0038)
    - 고급암호표준(AES)로 암호화
    - 패스워드를 이용하여 암호화 하고 6P로 시작하는 개인키가 됨
- P2SH, 다중서명주소 - 5장에서 자세히 다룸
- 꾸미기주소
    - 1Lovexxxxxxx와 같이 앞의 특정 글자수를 원하는 글자로 만드는 주소
    - 해당하는 주소를 만들기 위해서 임의로 대입해봐야 한다. 글자수가 늘어날수록 58배의 연산이 필요하다.
- 종이지갑
    - 비트코인 주소와 개인키를 물리적으로 인쇄한 종이
    - 가능하면 패스프레이즈로 암호화 하는게 좋다.