

블록체인 강의 자료 #1

강사 : 김로이 (해커스홀딩스 대표)

강사 소개

- 현)해커스홀딩스 및 해커스랩 대표
- 현)스토아 네트워크(STA) 대표
- 전)해커스랩 부사장(Security)
- 전)마이크론웨어 대표이사(Embedded)
- ASSIST 경영학 박사 과정
- 고려대 정보보호대학원 석사과정
- 전문 개발 분야
 - 디바이스 드라이버(파일시스템, ndis)
 - 임베디드 커널 포팅
 - 보안 소프트웨어
 - 리버스 엔지니어링
- 주요 이슈
 - finlchain 메인넷 개발(DPOR)
 - 랜섬웨어 사전 차단 솔루션 “Docstory” 지자체 납품 1등 제품
 - 전군 PC보안 솔루션 구축(Whitedefence)
 - 국내 최초의 안드로이드 네비게이션 “k9 안드로이드”

강의 계획

수업 개요	1일차 - 블록체인과 암호화폐 시장 전반의 이해, 기술적 특성의 이해 2일차 - DAO, 다양한 암호화폐 지갑 생성과 블록체인 활용 실습 3일차 - 토큰 생성 및 실습, NFT 민팅, 블록체인 비즈니스 토의
수업 목표	블록체인과 암호화폐 기반 환경을 이해하고, 미래의 새로운 잠재 시장의 가치 형성에 대한 인사이트를 발견하여 블록체인과 관련된 산업에 대한 활용 기회를 모색하고 정보 수용 역량을 높이고자 함
수업 방법	강사 중심으로 수업을 진행, 실습을 최대한 활용 예) 암호화폐 지갑 생성과 거래 정보를 확인하고 블록체인 장부의 기록이 어떻게 이루어지는지를 확인
교재	강사의 수업 자료 중심

2022년 2분기의 Cryptocurrency 상황

- 3억명 이상의 사용자
- 상위 10개의 Cryptocurrency 가 전체 마켓의 약 88% 차지
- 19,000개 이상 존재
 - Global Market cap : \$1.26T (코인마켓캡)
 - 글로벌 증권 마켓 : over \$100T
- 500개 이상의 거래소
- 18,000개 기업이 Cryptocurrency를 거래용 화폐로 활용

Crypto Deep Dive
Blockchain Voting for ElectionsGravity Spotlight - ApolloX
[ApolloX News Station] What is Web3Crypto Espresso
WeChat Bans Crypto!What is TRON?
Learn About "TRON" & Earn \$TRX!Free Airdrop!
Join the "MagicCraft"

시가총액에 의한 최고 100 암호화폐

Highlights 글로벌 암호화폐 시가 총액은 ₩1174.62T, 마지막날 ▼ 2.84% 감소했습니다 [더 읽기](#)

🔥 Trending

More >

- 1 Terra Classic LUNC ▼ 3.83%
- 2 Celsius CEL ▼ 19.00%
- 3 Bitcoin BTC ▼ 2.73%

🕒 Recently Added

More >

- 1 Skate Metaverse Coin SMC ₩21.27
- 2 Got Guaranteed GOTG ₩8,043.05
- 3 CASHTHAI CTHAI ₩0.01236

⭐ Top Gravity Accounts

More >

- BNB Chain @BNBChain [+ Follow](#)
- TRON_DAO @TRON_DAO [+ Follow](#)
- Everscale @Everscale [+ Follow](#)

★ 관심 목록
▣ 포트폴리오
암호화폐
카테고리
DeFi
NFT
Metaverse
플카닷(Polkadot)
BNB Chain
솔라나(Solana)
Avalanche
행 표시하기 100
필터
사용자 정의
⋮

#	▲ 이름	가격	24h %	7d %	시가총액	거래량 (24시간)	유통 공급량	최근 7일
★ 1	Bitcoin BTC 구매하기	₩26,790,039.40	▼ 2.71%	▼ 1.97%	₩510,993,211,609,731	₩37,112,065,603,545 1,385,293 BTC	19,074,000 BTC	
★ 2	Ethereum ETH 구매하기	₩1,430,749.01	▼ 4.45%	▼ 1.38%	₩173,328,630,156,527	₩18,844,234,069,719 13,183,751 ETH	121,263,695 ETH	

구찌에 이어 발렌시아가도 암호화폐 결제를 도입한다

가치 변동에 관한 발렌시아가의 생각은?



Balenciaga

패션

May 24, 2022

글

Eunbo Shim

출처

Wwd

기사 공유

링크

발렌시아가가 비트코인을 비롯한 암호화폐 상품 결제를 도입한다. <WWD>에 따르면 발렌시아가는 뉴욕 메디슨 애비뉴와 벌리힐스 로데오 드라이브에 있는 미국 플래그십 스토어에서 암호화폐 결제를 실험하고 추후 더 많은 지점과 온라인 스토어에 이를 도입할 예정이다.

다만, 암호화폐 결제 게이트웨이 제공 업체로 어떤 회사를 선택할지는 결정되지 않았다. 이 밖에도 발렌시아가는 암호화폐의 가치 변동 위험성에 관해 "암호화폐에 대해 장기적으로 생각하고 있으며 통화 가치 변동은 새로운 것이 아니다"라고 일축했다.

발렌시아가와 함께 케어링 그룹에 속해 있는 구찌 또한 5월 초 암호화폐 결제를 도입한 바 있다. 이에 관한 내용은 [이곳](#)에서 볼 수 있다.

2022년 2분기의 Cryptocurrency 상황

- 매 3초당 비트코인 관련하여 소셜 미디어에 개시됨 – 하루 평균 28,866개
- 2010년 5월 22일 => 피자데이(1만개 첫 거래)
- 4천6백만명의 미국인이 비트코인 투자(지갑은 약 8천만개)
- 비트코인 2021년 하루 40만건 거래 처리
 - 이더리움 하루 거래량 약 130만건
- 새로운 비트코인 생성량 : 새로운 블록 하나당 6.25개
 - 매 4년마다 채굴 보상 반감기를 거침
 - 2009년 50개 보상이 있었고, 현재 3번의 반감기를 거침

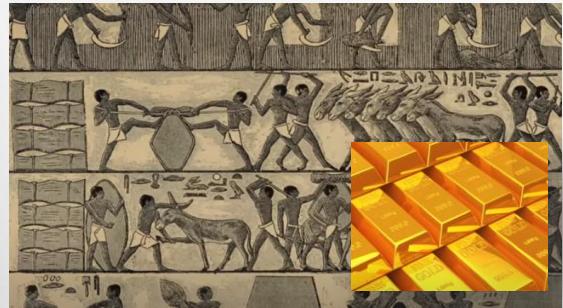
블록체인의 탄생과 발전 과정

- 정부에 의한 사찰과 통제의 만연
- 1980년대 CypherPunk의 탄생
 - Cipher(암호 의미) + Punk(조직 저항 의미)
- 암호 기술의 발전
 - 데이비드 차움 ecash(1980)
 - 아담백의 HashCash(1990)
 - 웨이 다이 B-Money(1998)
 - 뉴 샤보 Big Gold(1998)
 - 에스토니아의 KSI (2007)

화폐의 역사적 흐름



아집트인의 거래를 위한 금 사용



점성과 연성이 높음

원래 상태를 유지할 만큼 잘 부식되지 않음

한정된 자원

가공이 편함

칼륨 칼슘 나트륨 네슘 미늄 아연 철 니켈 주석 납 수소 구리 수은 은 백금 금
K > Ca > Na > Mg > Al > Zn > Fe > Ni > Sn > Pb > H > Cu > Hg > Ag > Pt > Au
← 이온화 경향이 크다. = 반응성이 크다. = 산화되기 쉽다.

희소성

16세기 영국(금보관소)



금 세공업자들은 금고의 금보다 10배나 많은 보관증을 발행했습니다

이자 발생의 명분



은행



전쟁



금본위의 폐지

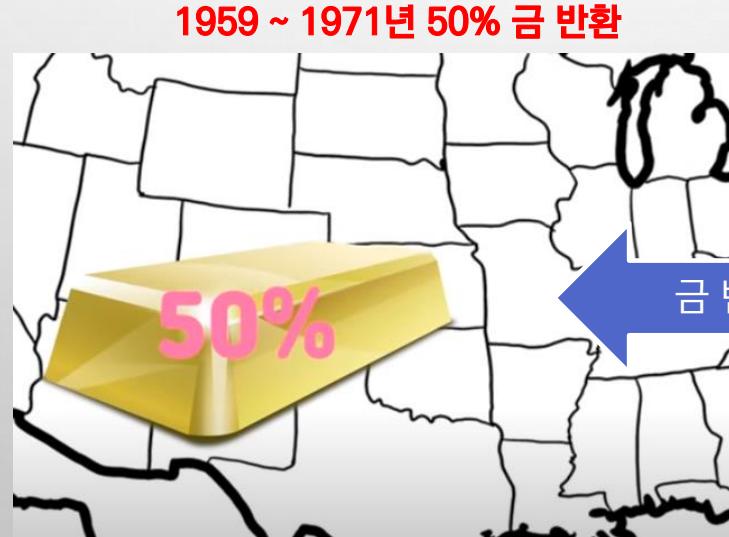


전쟁 자금 마련을 위해 금을 보내고, 화폐를
유통의 결과로 **브레톤우즈 협정**
→ 금 1온스 \$35 고정



1971년 8월 15일
금본위제 폐지

불환화폐(환불균형)



금 지급 준비율 기준이 없어 달러 무한 발행



신용화폐 확장

지급 준비율 : 입금 > 출금보다 많을
거란 사실을 기반으로 출금 가능
수준만큼 유동자금 비율



100만원



271만원



343.9만원

인플레이션 발생

지급준비금 : 10만원, 대출채권 : 90만원



90만원

기업 : 90만원



가계



지급준비금 : 9만원, 대출채권 : 81만원



81만원

기업 : 81만원



가계



81만원 저축

지급준비금 : 8.1만원, 대출채권 : 72.9만원



72.9만원

기업 : 72.9만원



가계



비트코인의 핵심 키워드

- 금본위제 기반 설계
 - 금 => 희소성, 공급 안정성 (금본위제를 기반으로 오랜기간 화폐로 발행)
 - 총량 2100만개
 - 10분 단위의 발행 => 공급 안정성
 - 발행량, 발행시간 즉 결정된 정보는 예측이 가능
- 탈중앙시스템
 - 중앙에 의해서 신뢰가 훼손되지 않는 설계 구조
 - 제약없는 참여, 검증과 합의에 참여
- 분산장부
 - 동일 장부를 모두가 소유 => 투명성과 신뢰성 보장

블록체인의 개념

블록체인을 어떻게 정의하는가?

- 블록을 분산저장하고 체인처럼 연결한 것
- 분산된 암호장부

Peer-to-Peer

- 계층 혹은 기능을 의미
- Peer 는 Node (마디, 교점)
 - 데이터 통신망에서, 데이터를 전송하는 통로에 접속되는 하나 이상의 기능 단위

블록의 전산학적 개념

- 하나 이상의 논리 레코드가 모여서 구성, 각종 저장 매체와의 입.출력 단위
- 비트 → 바이트 → 워드 → 필드 → 레코드 → 블록 → 파일 → 데이터베이스

물리레코드

암호화폐의 개념

암호화폐란 무엇인가?

암호화된 디지털 화폐

암호화된 디지털(데이터)에 가치를 부여

암호의 종류

SHA-1, SHA256, MD5, RSA, DES

스마트 컨트랙트

Nick Szabo가 1994년 최초 제안한 개념

프로그래머블한 디지털 계약

암호화 방식

- 암호화 방식 구분
 - 대칭 키 암호화 방식

암호화 키와 복호화 키가 같은 방식으로 암호화/복호화 속도가 공개키 암호화보다 빠르다.

암호문의 크기가 평문보다 크지 않다. 키의 개수가 증가되면 키 관리가 어려워진다. 키를 교환해야 하는 문제와 탈취 관리의 문제가 있다. 기밀성을 제공하나, 무결성/인증/부인방지를 보장하지 않는다.

- 비대칭 키 암호화 방식(공개키 암호 방식)

암호화 키와 복호화 키가 다른 방식으로 공개키와 비밀키로 구분된다. 대칭 키 암호화 방식보다 보안강도가 높다.

연산이 복잡하고 느리다. 키 분배 필요가 없고, 기밀성/인증/부인방지를 보장한다.

- **SHA-1(SECURE HASH ALGORITHM 1)**

- SHA 함수들 중 가장 많이 쓰이는 함수로, 임의의 길이의 입력데이터(최대 2^{64})를 160비트의 출력데이터(해시값)로 바꾼다.
- 인터넷 보안 프로토콜과 공개키 인증에서도 적용되고 있는 매우 중요한 암호 알고리즘이다.
- 160비트의 메시지 디제스트를 생성하는데 무차별 대입 공격으로 동일한 해시를 만들 수 있는 취약점으로 인해 많은 기업들이 SHA-2로 전환

기밀성
무결성
가용성
부인방지

상호운용성
경제성
명활용성



BGP	chord	PGP	message queue		Share vs Incentive	reward
inode	SSL	RPC	metric	Responsibility	HD Wallet	dijkstra
tapestry/pasty			Multi-dimensional	Unstructured overlay		Scale-free network
hole punching			Smart Contract	governance	cryptography	steganography
oid	socket		Decentralize	Consensus algorithm		vxlan
Uni-dimensional		file tree	security	Peer to Peer		overlay network
vlan	vpn	blk	Byzantine Fault Tolerance		merkle	CAN
POW	flt	offchain		Storage	elliptic curves	hash
POS	MFT	rip	speed		sysctl	Page rank
base58	DPOS		EdDSA	ECDSA		crud
				IX	HFT	dN1/d logbN
			transaction		dht	
				cli	ipsec	b logN + b
		Scale-out NAS	DMA Transaction			

비트코인: P2P 전자 거래 시스템

Bitcoin: A Peer-to-Peer Electronic Cash System

Satoshi Nakamoto
satoshin@gmx.com
www.bitcoin.org

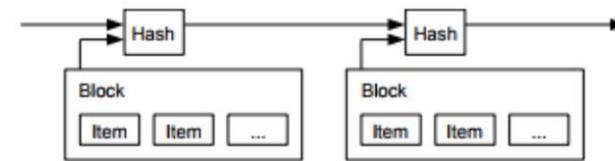
- P2P 버전의 전자화폐로 금융기관 없이 당사자(Peer)간 결제가 가능
- 이중지불을 저지할 '믿을 수 있는 제 3자'는 필요하지 않음
- 네트워크의 TIMESTAMP로 거래를 기록
 - Proof-of-Work
 - Zero bit 가 되는 nonce 찾기
- 가장 긴 정보의 연결을 훼손할 방법이 없다
 - 발생한 사건의 순서를 증명
 - 참여하는 노드의 신뢰성을 훼손할 만큼의 연산을 내는 것은 무모한 일
- 네트워크를 통해서만 공유

Abstract. A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution. Digital signatures provide part of the solution, but the main benefits are lost if a trusted third party is still required to prevent double-spending. We propose a solution to the double-spending problem using a peer-to-peer network.

The network timestamps transactions by hashing them into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing

3. Timestamp Server

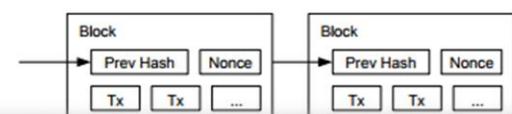
The solution we propose begins with a timestamp server. A timestamp server works by taking a hash of a block of items to be timestamped and widely publishing the hash, such as in a newspaper or Usenet post [2-5]. The timestamp proves that the data must have existed at the time, obviously, in order to get into the hash. Each timestamp includes the previous timestamp in its hash, forming a chain, with each additional timestamp reinforcing the ones before it.



4. Proof-of-Work

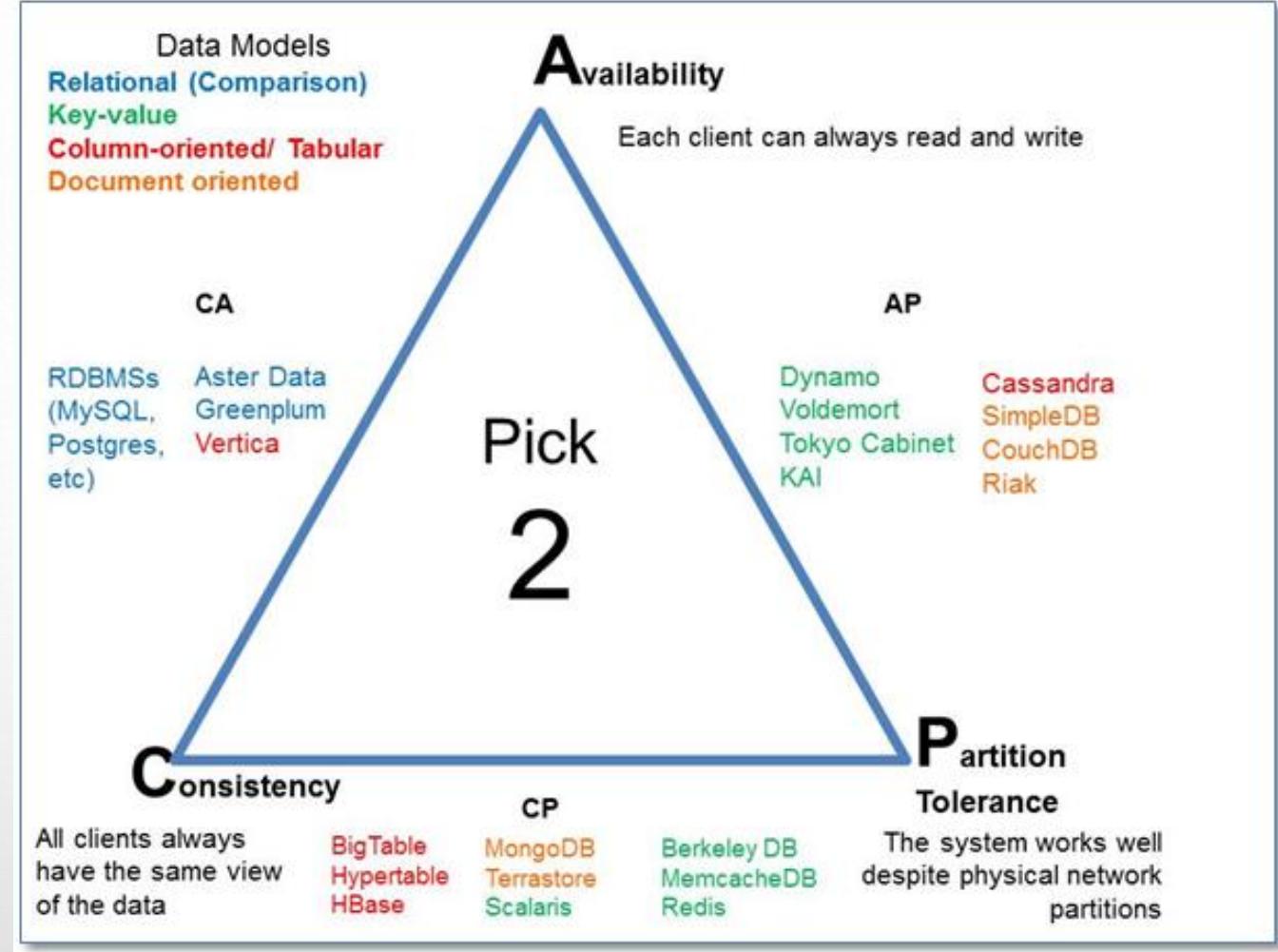
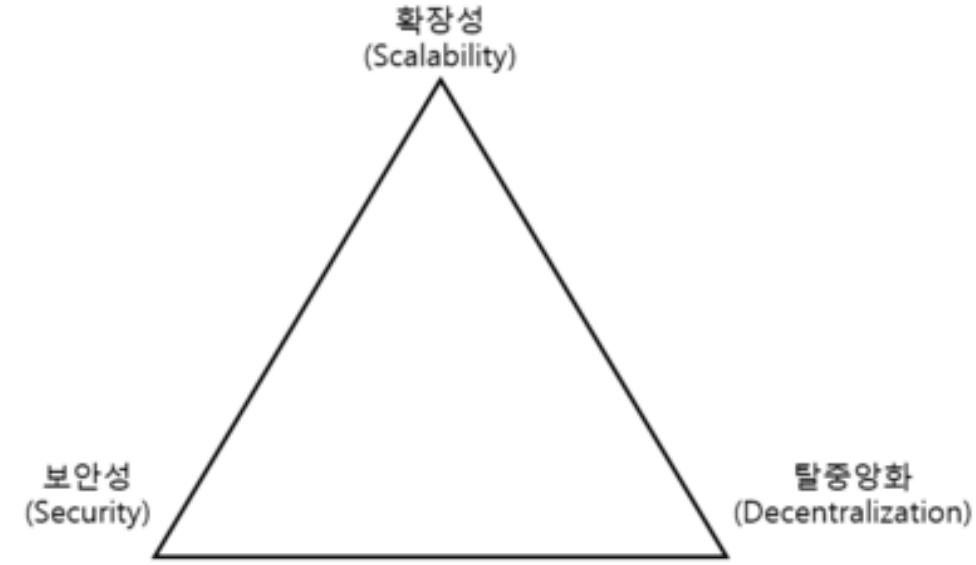
To implement a distributed timestamp server on a peer-to-peer basis, we will need to use a proof-of-work system similar to Adam Back's Hashcash [6], rather than newspaper or Usenet posts. The proof-of-work involves scanning for a value that when hashed, such as with SHA-256, the hash begins with a number of zero bits. The average work required is exponential in the number of zero bits required and can be verified by executing a single hash.

For our timestamp network, we implement the proof-of-work by incrementing a nonce in the block until a value is found that gives the block's hash the required zero bits. Once the CPU effort has been expended to make it satisfy the proof-of-work, the block cannot be changed without redoing the work. As later blocks are chained after it, the work to change the block would include redoing all the blocks after it.



블록체인 메인넷 정의

- **변조 불가능한 분산 저장소**
- 독자적인 **탈중앙화(비제어기반)** 네트워크
- 신뢰 **프로세스**를 운영하는 네트워크
- P2P 오버레이 네트워크의 신뢰 프로세스 제공
- 부인 방지 네트워크
- 디지털의 **VALUE 이동** 네트워크
- CAP에 목적에 **ACID** 성질을 만족(낮은 포맷 수준)
- 테스트넷
 - 메인넷의 프로토 타입



확장성(scability)

사용자 수의 증대에 유연하게 대응할 수 있는 정도(일반적으로 처리속도인 TPS를 의미)

블록체인에서는 사용자수의 증가에 따라 거래 건수가 늘어나더라도 무리 없이 전송 처리 용량을 증대시킬 수 있는 능력

탈중앙화(decentralization)

중앙 집중화를 벗어나 분산된 소규모 단위로 자율적으로 운영되는 것(분산화 정도, 노드의 개수를 의미)

신뢰된 제 3자를 별도로 두지 않고 분산형 네트워크(P2P)환경에서의 거래

보안성(security)

블록체인 기밀성, 무결성, 가용성, 프라이버시 등 다양한 요소를 총족

트릴레마 사례와 TPS

- **블록체인 트릴레마의 사례**
 - 비트코인 : 높은 탈중앙화와 보안성은 확보, 확장성의 한계로 트릴레마 발생
네트워크 확장에 따른 트랜잭션 속도에 대한 확장성 문제 발생
 - 이더리움 : 확장성 한계로 트릴레마 발생, 확장성 한계를 극복하기 위해
작업증명 방식의 대안으로 지분증명과 샤팅 기술 활용한 이더리움 2.0을 발표
작업증명 기반 이더리움 채굴 난이도를 변경해 지분증명으로의 자연스러운 전환을 유도하는
난이도 폭탄 처리와 관련해 네트워크 지연 등의 이슈가 발생하여 업데이트가 연기되었다.
 - 이오스 : 위임지분증명(DPOS) 합의 알고리즘을 통해 탈중앙화와 확장성 해결
위임지분증명 합의 알고리즘의 특성상 소수의 노드를 통해 합의가 이루어지기 때문에 보안성 저하
- **TPS(TRANSACTION PER SECOND, 초당 처리할 수 있는 거래내역의 수)**

블록체인 네트워크가 1초에 얼마나 많은 거래를 처리할 수 있는지 나타내는 지표
단순히 TPS로만 해당 프로젝트의 좋고 나쁨을 판단할 수는 없음, TPS에만 집중하는 경우
탈중앙화와 보안성이 취약해질 가능성이 높다.

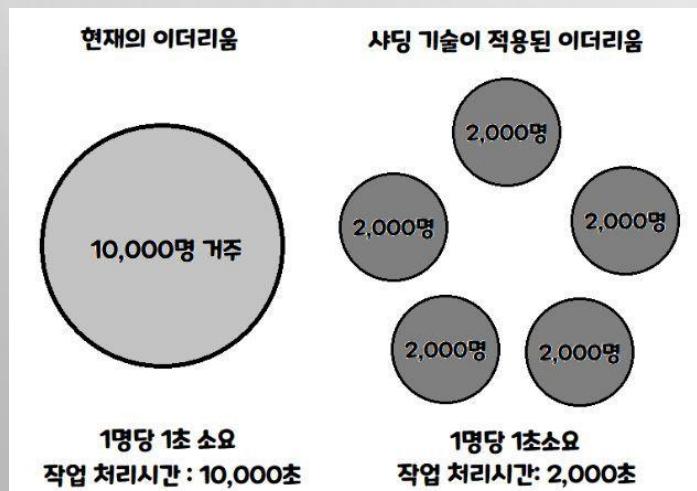
Sharding

- 이더리움의 새로운 기술 도입 배경

- 현재 이더리움의 거래 처리 속도는 15TPS 수준으로 매우 느려 거래를 처리하는데 항상 대기열을 가지고 있는 모습이 많다. 사용화가 되기 위해서는 이 느린 확장성(Scalability) 문제를 해결해야만 한다.
- 기존 PoW 방식에서 PoS방식으로 합의 알고리즘을 전환하고자 한다.
- 확장성 문제를 해결하기 위해 이더리움은 샤딩(Sharding), 라이덴 네트워크(Riden Network), 플라즈마(Plasma) 기술을 통해 해결하려 한다.

- 샤딩

- 샤드는 조각/파편을 의미한다. 샤딩은 블록체인이 나오기 이전에도 있던 기술로, 거대한 데이터베이스를 효과적으로 관리하기 위해 데이터를 분할하여 관리하는 방식이다. 온체인 솔루션이다.



샤딩의 장단점

- 장점 : 대용량 정보를 처리함에 있어 노드별 용량 부담이 저하되고 네트워크에 과부하가 걸릴 가능성이 낮아진다. 일반적인 블록체인 노드보다 체계적인 정보 처리가 가능하므로 접근성이 향상된다. 결과적으로 속도가 크게 향상된다.
- 단점 : 너무 많이 분산하게 될 경우 정보 손실 문제나 블록체인 무결성에 문제가 생길 수 있다.

플라즈마(Plasma)

- 하위체인인 차일드체인을 이용하여 이더리움의 확장성을 해결하려는 오프체인 프로젝트
- 차일드체인에서 거래들을 취합한 후 중개자를 통해 메인체인인 이더리움 네트워크에 반영하는 방식
- 하위체인인 차일드체인을 쓰기 때문에 차일드체인 자체에 문제가 생기더라도 메인체인인 이더리움의 네트워크에는 문제가 발생하지 않음
- 플라즈마는 사용자가 모든 플라즈마 체인의 블록을 다운받아서 유효성 검사를 수행해야 한다는 번거로움
 - 하위체인에 문제가 생겼다고 하면 그 하위체인의 모든 사용자들이 메인체인으로 이동 필요
- 메인체인에서 모든 인출을 처리한다고 하였을 때, 사용자가 많을 수록 모든 인출 처리 시간은 증가하게 되고 그 시간 동안 하위체인에 있는 코인들은 탈취 가능성

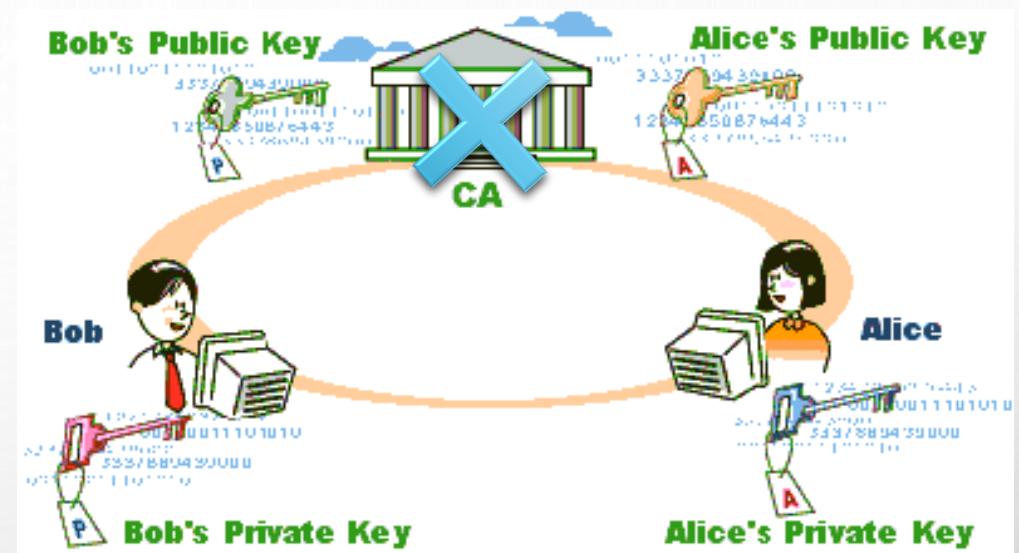
인증 기관

- 제3의 신뢰자

- Certificate Authority
 - Bank is right or not ?

- 탈중앙화의 이점

- 금융 시장의 의도성(LP) 탈피
 - 거래의 투명성 증대(블록탐색기)
 - 금융 IT 의 한계 극복(베이시스)
 - 금융 네트워크의 글로벌화에 기여(분산 저장)
 - 금융 서비스의 상향 평준화에 기여(가용성 높은 거래 네트워크 제공)



- ❖ CA : 디지털서명을 이용한 전자상거래 등에 있어서 누구나가 객관적으로 **신뢰할 수 있는 제3자(Trusted Third Party)**를 의미함
 - 전자서명 및 암호화를 위한 디지털 인증서를 발급, 관리하는 서비스 제공 기관/서버

CERTIFICATION AUTHORITY LIST

A W3Techs survey from May 2018 shows that IdenTrust, a cross-signer of Let's Encrypt intermediates,^[14] has risen to be the most popular SSL certificate authority, while Symantec has dropped out of the chart, due to its security services being acquired by DigiCert.^{[15][16]}

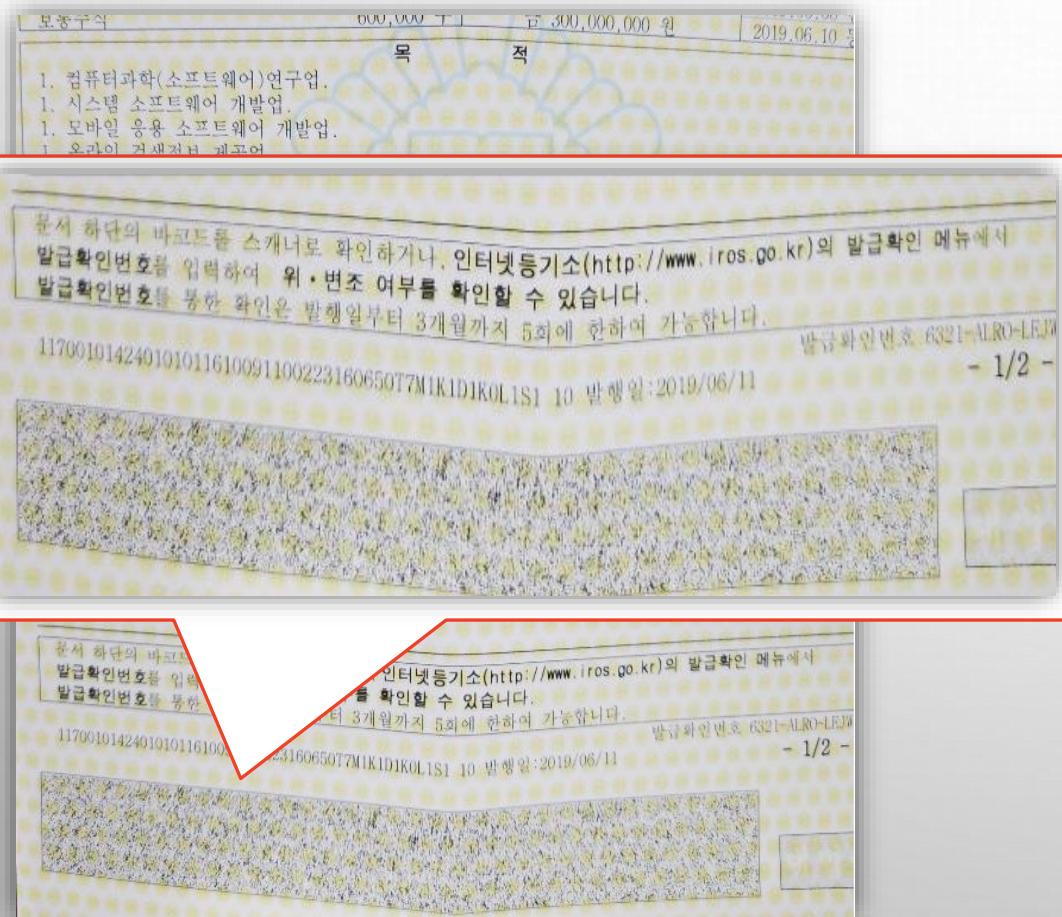
Rank	Issuer	Usage	Market share
1	IdenTrust	20.4%	39.7%
2	Comodo	17.9%	34.9%
3	DigiCert	6.3%	12.3%
4	GoDaddy	3.7%	7.2%
5	GlobalSign	1.8%	3.5%
6	Certum	0.4%	0.7%
7	Actalis	0.2%	0.3%
8	Entrust	0.2%	0.3%
9	Secom	0.1%	0.3%
10	Let's Encrypt	0.1%	0.2%
11	Trustwave	0.1%	0.1%
12	WISeKey Group	< 0.1%	0.1%
13	StartCom	< 0.1%	0.1%
14	Network Solutions	< 0.1%	0.1%

- CA의 역할과 기능

- 디지털서명의 서명자의 신원을 확인
- 서명자로부터 그의 공개키를 맡아 보관
- 대외적으로 서명자와 그의 공개키의 귀속관계를 보장
- 인증서 및 인증서폐기목록(CRL)을 발행할 수 있음
- 1 이상의 등록기관(RA,Registration Authority)을 지정할 수 있음

서명 기술 적용의 예

- 전자서명 : 서명자를 확인하고 서명자가 당해 전자문서에 서명했다는 사실을 나타내는 데 이용하고, 특정 전자문서에 첨부되거나 논리적으로 결합된 전자적 형태의 정보



구분	공인전자서명 인증체계	행정전자서명 인증체계
명칭	국가 공개 키 기반 구조 (National Public Key Infrastructure, NPKI)	정부 공개 키 기반구조 (Government Public Key Infrastructure, GPKI)
주관기관	과학기술정보통신부 한국인터넷진흥원	행정안전부 행정전자서명인증관리센터
근거법령	전자서명법(1999년 2월 제정)	전자정부법(2001년 3월 제정)
최상위 인증기관	한국인터넷진흥원(KISA)	행정전자서명인증관리센터
인증기관	한국정보인증(KICA) 코스콤(KOSCOM) 금융결제원(KFTC) 한국전자인증(KECA) 한국무역정보통신(KTNET) 한국정보사회진흥원 (구 전산원, 2008년에 한국정보인증으로 업무 이관)	교육과학기술부 국방부 행정안전부 대검찰청 병무청 대법원(법원 행정처)
등록기관	은행, 증권회사 등 등록대행기관	29개 기관: 대통령비서실 등 13개 중앙 행정기관, 서울특별시 등 16개 지방자치단체
발급대상	<u>자연인</u> 또는 <u>법인</u> , 서버 및 기계장치	행정기관, 보조기관, 보좌기관 및 공무원
용도	전자 상거래, <u>전자정부</u>	<u>행정행위</u>

전자서명법

- 전자서명법
 - 제 1조(목적) : 전자문서의 안전성과 신뢰성을 확보하고 그 이용을 활성화하기 위하여 전자서명에 관한 기본적인 사항을 정함으로써 국가와 사회의 정보화를 촉진하고 국민생활의 편의 증진을 목적
- 전자서명법 개정(2020.05.20 개정안 국회통과 => 2020.12.10 개정안 시행)
 - 기존의 공인인증서는 사용하는데 있어 ACTIVEX등 프로그램 설치, 복잡한 비밀번호, 짧은 유효기간으로 소비자들의 불편함이 존재하였다.
 - 때문에 공인인증서의 법적 지위를 없애고 공동인증서로 바뀌면서 민간인증서와 구분이 없어지게 되었다.
 - 금융결제원 등 정부 인정기관만 발급 가능했던 인증서를 민간 기업도 발급이 가능 해졌다.
 - 민간 전자서명을 이용하는 소비자들은 간편비밀번호, 지문과 홍채, 모바일 앱 등 다양한 수단 사용이 가능하고 유효기간이 2~3년으로 비교적 길고, 별도 플러그인이 필요 없다는 장점이 있다.
- 전자서명이 제공하는 기능
 - 위조 불가(UNFORGEABLE), 인증(AUTHENTICATION), 재사용 불가(NOT REUSABLE),
변경 불가(UNALTERABLE), 부인 방지(NON REPUDIATION)

전자서명의 종류

- 전자서명의 종류

- RSA

전자상거래에서 가장 흔히 쓰는 알고리즘. 합성수의 소인수분해문제가 어렵다는 데에 기초한 비대칭형
공개키 암호화방식이다. 암호화 뿐 아니라 전자서명의 용도로도 사용된다. 공개키로 메시지를 암호화하고,
개인키로 메시지를 복호화한다.

- ELGAMAL

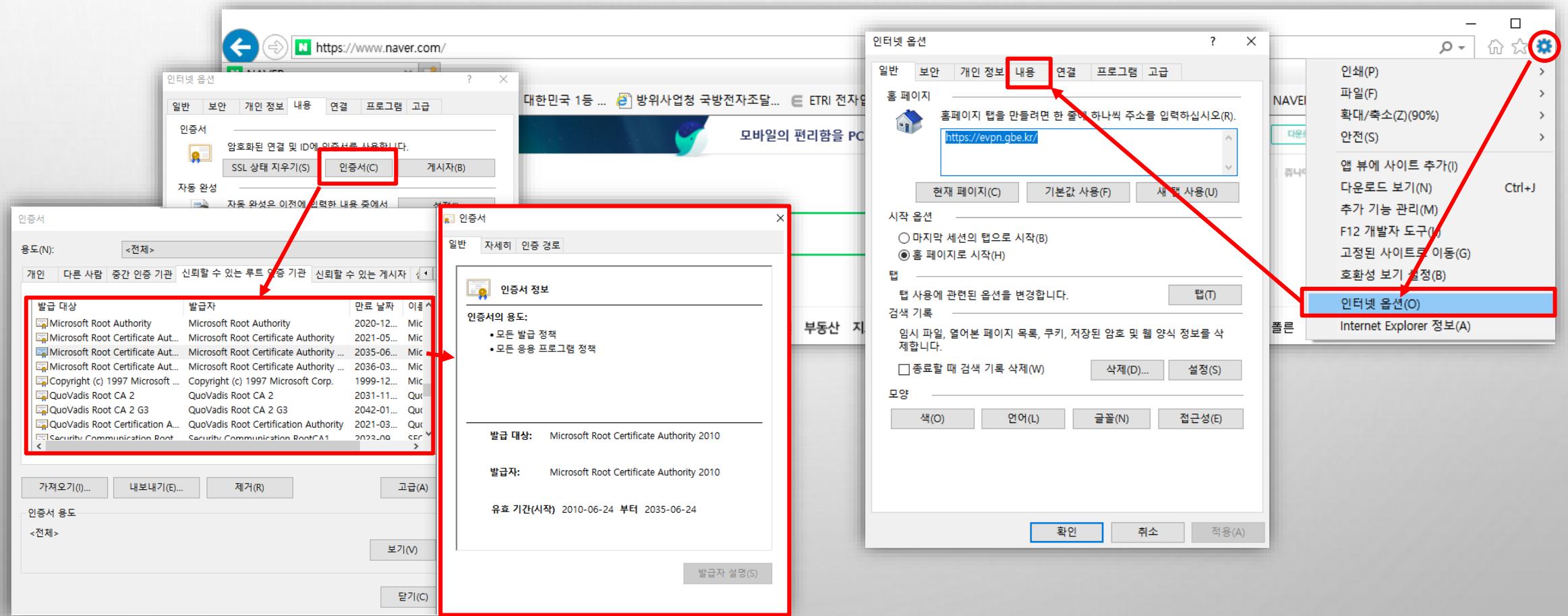
이산대수 문제의 어려움(P 가 소수이고 G 가 원시원소일 때 G, X, P 를 이용하여 $Y=G^X \text{ MOD } P$ 를 구하긴
쉽지만 G, Y, P 값을 이용하여 지수승의 X 를 구하기는 어렵다.)에 기초하여 X 를 개인키, Y 를 공개키로 사용
다른 알고리즘보다 속도가 느린다.

- DSA

ELGAMAL 전자서명을 개량하였다(이산대수 문제), ELGAMAL보다 서명과 검증에 소요되는 계산양이 적다.
서명 길이는 320BIT이며, ELGAMAL에 대한 공격 일부를 해결하였다. 오직 전자서명 기능만을 제공하도록 설계되었다.
RSA에 비해 키 생성이 빠르지만 검증속도는 더 느린다. 해시 함수로는 SHA-1을 사용한다.

IE의 서명 정보

- 루트 인증 기관(CA) : 관리하는 공개 키 인증서나 자체 서명 인증서이다.[1] 루트 인증서는 공개 키 기반계획의 일부이다. 가장 상업적으로 공통되는 ITU-t X.509 표준형, 일반적으로 루트인증기관(CA)에서 나온 디지털 서명을 포함



“최상위인증기관 안전성 검증할 제3자 감사 필요”

이민철 기자 기자 2013.07.19 06:20:42

가+ 가-

- 공인인증 단일 체계 문제점 지적, “사설인증체계도 사용 환경 만들어야”

[디지털데일리 이민철기자] 국내 공인인증체계를 전반적으로 관리하는 최상위인증기관(root CA)의 안전성과 신뢰성을 확보하기 위해 ‘공인 인증 정책 위원회’를 설치, 운영해온다는 수령이 나눴다.

영흥열 순청향대 교수<사진>는 18일 서울 엘타워에서 열린 ‘전자인증서비스 발전을 위한 컨퍼런스’ 기조연설을 통해 “현재 우리나라 최상위인증기관 한국인터넷진흥원(KISA)은 자체적인 ‘공인인증 정책위원회’를 운영하거나 제3자 감사를 통해 투명성과 안전성 우려를 불식할 필요가 있다”고 말했다.

최상위인증기관은 국내 공인인증기관들을 관리, 감독하고 각 기관들이 발급한 공인인증서의 상호연동성을 보장해주는 기관이다. 국내에서는 KISA가 유일하게 그 역할을 담당하고

일각에서는 KISA가 단일 최상위인증기관으로 지정돼 있어 공인인증 시장의 경쟁이 제한되고, 제3자에 의한 감사를 받지 않기 때문에 투명한 운영이 미뤄지지 않는다고 지적하고

미려한 지적을 근거로 지난 5월 국회서 발의된 전자서명법도 못한 채 계류됐다.

또 영 교수는 공인인증체계와 사설인증체계를 구별해 사용하는 부분”이라고 말했다.

공인인증체계와 사설인증체계를 구별해 사설인증체계 기반시

미와 관련 오승곤 미래부 정보보호정책과장은 이날 기조연설에서 “사용할 수 있도록 개선돼야 하기 때문”이라는 입장을 나타냈다.

공인인증서 사용 강제화로 인해 다양한 인증 수단 도입이 법률로 공인인증서의 구체적 활용은 개별영역에서 결정될

영 교수는 공인인증기관의 보안관리와 운영보안을 향상시키기

그는 “인증서비스의 침해사고 대응, 재해 대응 체계 개선을

650억원 규모 공인인증 시장 '대변혁'

[이슈진단+] 공인인증제도 폐지정책(上)

정부가 추진 중인 공인인증제도 폐지정책이 지난 3월 입법예고된 전자서명법 전부개정안을 통해 구체화됐다. 개정안에는 사설인증서와 구분된 ‘공인인증서’ 개념이 빠졌다. 대신 명시적인 전자서명 법적효력 및 전자서명인증업무 평가제 도입 조항이 들어갔다. 현재의 공인인증 시장에 혁성이 불가피하다. 어떤 변화가 오는지를 상, 하편으로 살펴봤다. [편집자주]

공인인증제도는 현행 전자서명법으로 지정된 ‘공인인증기관’만이 공인인증서를 발급 및 갱신할 수 있도록 정한 제도다. 그간 공인인증제도로 공인인증서 외의 전자서명 기술 및 서비스 발전과 시장경쟁이 저해됐다는 판단에 따라 정부가 제도 폐지를 추진하고 있다.

핵심은 공인인증서라는 전자서명 수단을 없애는 게 아니라, 해당 기술을 과정해 온 공인인증기관이라는 법정지위를 없애는 거다. 한국정보보호산업협회(KISIA)가 2017년 정보보호산업실태조사에서 655억원 규모로 파악한 시장에 지각변동이 예상된다.

현재 공인인증기관은 6곳이다. 앞서 2000~2002년 사이에 한국증권전산(현 코스콤), 한국정보인증, 금융결제원, 한국전산원(현 한국정보화진흥원), 한국전자인증, 한국무역정보통신, 6곳이 공인인증기관으로 지정됐다.

그간 변화도 있었다. 한국정보화진흥원이 2001년부터 맡아 온 공인인증업무를 2008년 한국정보인증에 넘기며 공인인증기관 대열에서 빠졌고, 지난달 이니텍이 16년만에 신규 공인인증기관으로 지정됐다.

과학기술정보통신부는 현행 공인인증제도의 근거가 담긴 전자서명법의 전부개정안을 입법예고했다. 개정안의 특징은 4가지다.

구글, 최상위 인증기관(Root CA) 운영한다

HTTPS는 암호 구글 제품 발전 기반기술

임민철 기자 | 입력: 2017/02/02 11:57 | 컴퓨팅



웹브라우저가 방문한 웹사이트 서버와 HTTPS 암호화 통신을 수행하고 있을 때 주소창에 이런 자를통 아이콘이 표시된다. 서버에 적용된 SSL인증서의 신뢰성이 확인돼야 한다. [사진]

구글이 HTTPS 암호화 통신에 쓰는 SSL 인증서를 직접 만들어 쓰기로 했다.

기존 최상위 인증기관(Root CA) 2곳을 인수해 ‘구글트러스트서비스’

Digital Signature Market

Global Digital Signature Market (US\$ Mn),
2018 to 2026



현재 1조원 규모

← 2026년 7조원 규모로 성장

Global Digital Signature Market Share,
By Component, 2018



North America Digital Signature Market
(US\$ Mn), 2018



North America
US\$ 297.3 Mn





Digital Signature Market Size, Share And Industry Analysis By Component (Software, Hardware, Services), By Deployment (On-Premises, Cloud), By Organization Size (SMEs, Large Enterprises), By End-User (BFSI, IT And Telecommunication, Government, HealthCare And Life Science, Education, Retail, Real Estate, And Others) And Regional Forecast 2019-2026

Region : Global | Published Date: Oct, 2019 | Report ID: FBI100356 | Status : Published

Share

[Summary](#)

[Table of Content](#)

[Market Segmentation](#)

[Methodology](#)

[Infographics](#)

[Request Sample PDF](#)

KEY INDUSTRY INSIGHTS



[Listen to this report overview](#)

The global digital signature market valued at US\$ 879.6 Mn in 2018 and is projected to reach US\$ 6,128.0 Mn by the end of 2026, exhibiting a CAGR of 28.77% during the forecast period (2019 – 2026).

Owing to the rapid advances in e-business, there is an increasing need for online security and authentication. Many advanced technologies are developing to serve internet authentication. One of the main issues in e-business operations is the need to replace the hand-written signature with the digital signature.

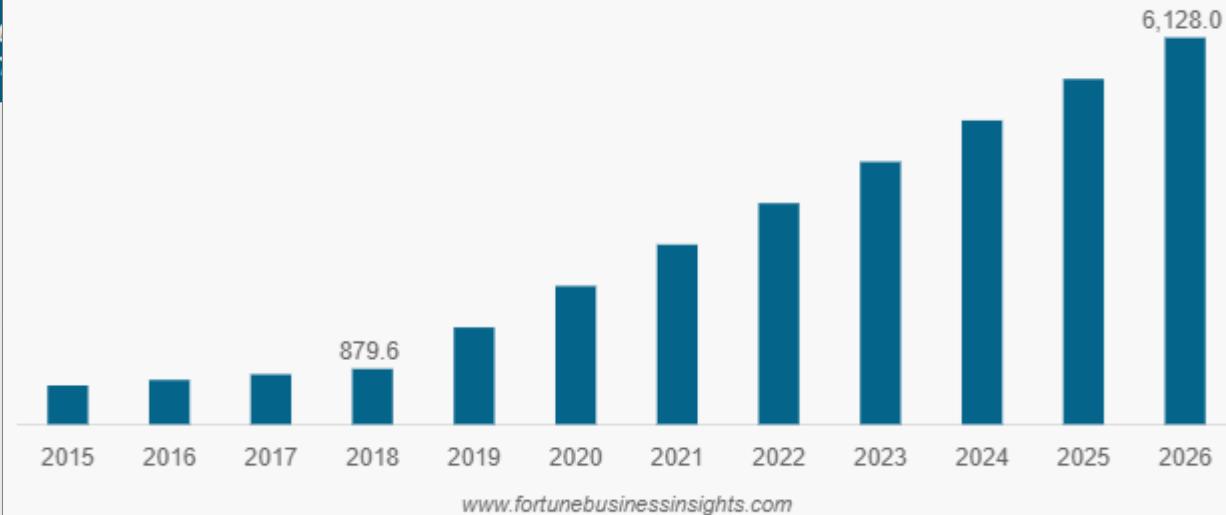
During online communication, digital signature enables organizations to authenticate and assure the client's identity and records. Digital signatures offer highly precious possibilities for organizations to enter into noncommercial contracts. For example, if the law recognizes digital signatures as valid signatures, an e-mailed job offer would be legal, just as if it were sent in hardcopy and signed by the employer.

Digital signatures authenticate electronic documents the same way as the printed handwritten signatures verify documents. The digital signature market growth is driven by the growing use of digital signatures to eliminate fraud, encourage technological innovations, and improve data integrity, scalability, and transparency.

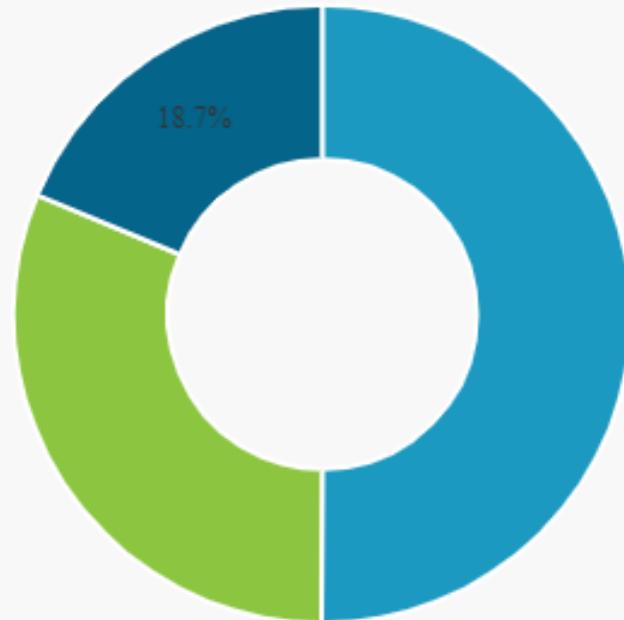
Note)

<https://www.fortunebusinessinsights.com/industry-reports/digital-signature-market-100356>

Global Digital Signature Market Size, 2015-2026 (US\$ Million)



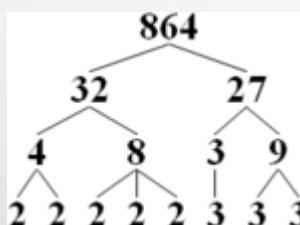
Global Digital Signature Market Share, By Component, 2018



공개키 기반

- 공개키 = 비대칭 암호키
 - 2개의 키를 사용
 - 서명 방식 : RSA, DES, DSA, ECDSA, EdDSA
- 개인키
 - 송신사 식별용
 - 부인 방지(Non-Repudiation)
 - 개인키 암호화 => 전자 서명(Digital Signature)
- 공개키 암호화
 - 수신자의 공개키로 암호화
 - 수신자의 개인키로 해석이 가능

- RSA
 - 소인수분해 기반
 - 결정론적 알고리즘
 - 암호, 서명 모두 이용 가능
- DSA(Digital Signature Algorithm)
 - 이산대수 기반(logarithm)
 - 확율론적 알고리즘
 - 서명만 이용 가능

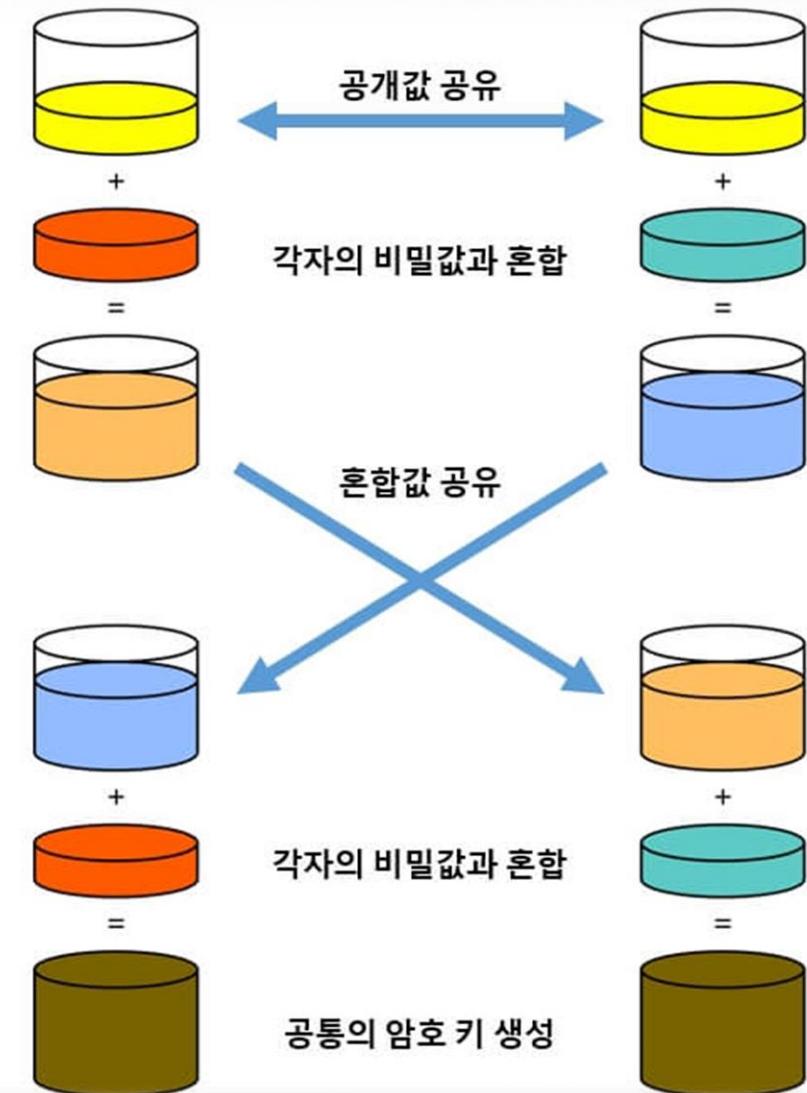


$$\log_2 8 = 3 \Rightarrow 2^3 = 8$$

$$\text{ind}_2 3 = 4 \pmod{13}$$

2를 4제곱하고 13으로 나누면 나머지는 3
 $16 = 13 \times 1 + 3$

- DSA와 RSA의 비교
 - 암호화 속도는 DSA와 RSA가 유사.
 - 키 생성은 DSA가 빠르나, 검증속도는 RSA가 빠름



on-chain, off-chain

• 온체인

- 블록체인 위에 기록이 된다고 하여 온체인이라 한다.
- 온체인은 블록체인 네트워크 상 발생하는 트랜잭션이다. 온체인 상의 모든 거래 정보가 블록에 포함되며 기록된 거래는 영구적으로 제거할 수 없다. 대표적인 예로 비트코인과 이더리움이 있다.

• 오프체인

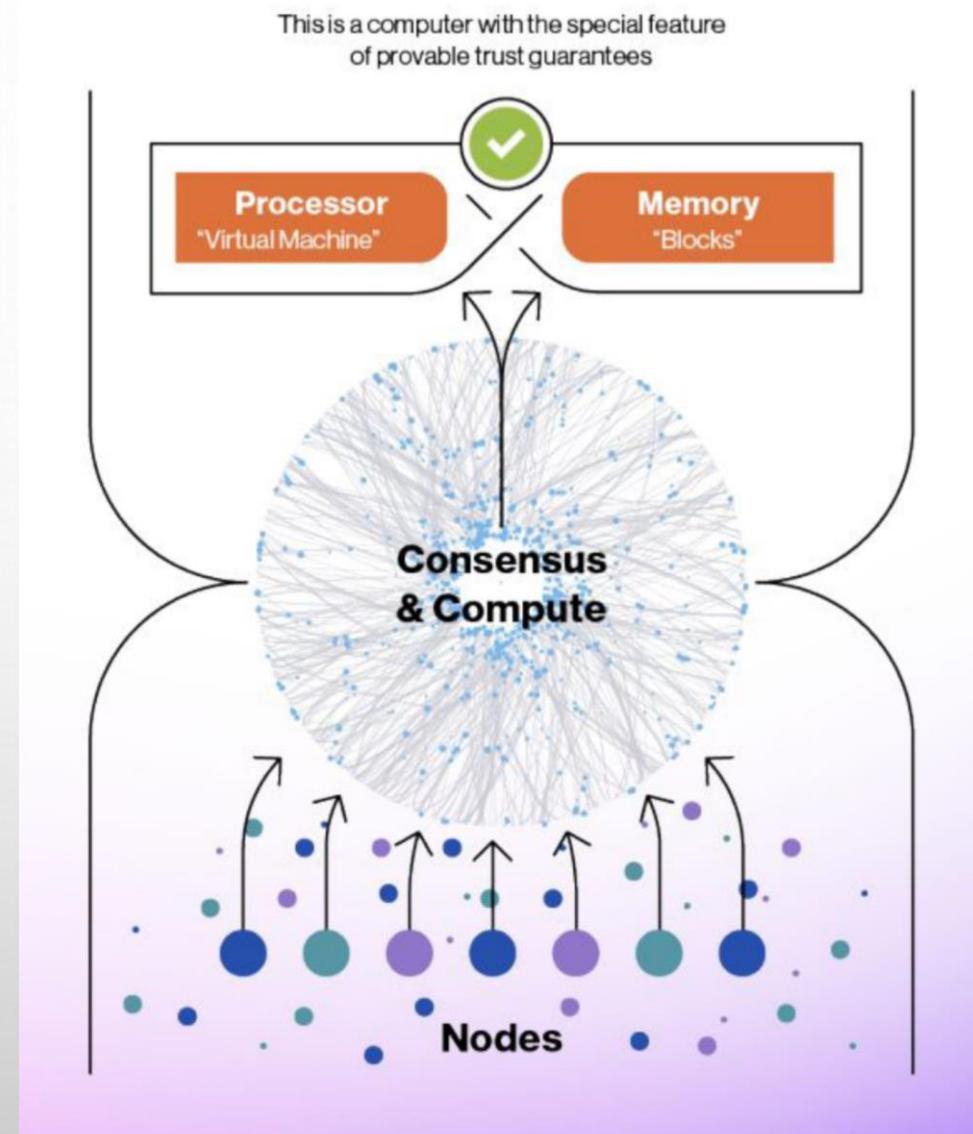
- 온체인과 반대로 블록체인 외에 기록이 된다고 하여 오프체인이라 한다.
- 블록체인에 직접 기록하는 방식이 아닌 특정한 거래 내역을 블록체인이 아닌 독립된 외부에 기록한다.
- 합의과정이나 검증이 필요 없어서 온체인과는 다르게 빠른 처리가 가능하다.
- 핵심 데이터만을 블록체인에 기록하고 빠른 속도가 중요한 데이터는 블록체인이 아니라 DAPP 중앙 서버에 기록된다. 온체인과는 다르게 오프체인은 속도도 빠르고 비용도 저렴하지만 오프체인에 기록된 트랜잭션을 신뢰할 수 있다는 문제가 있다.

노드의 역할

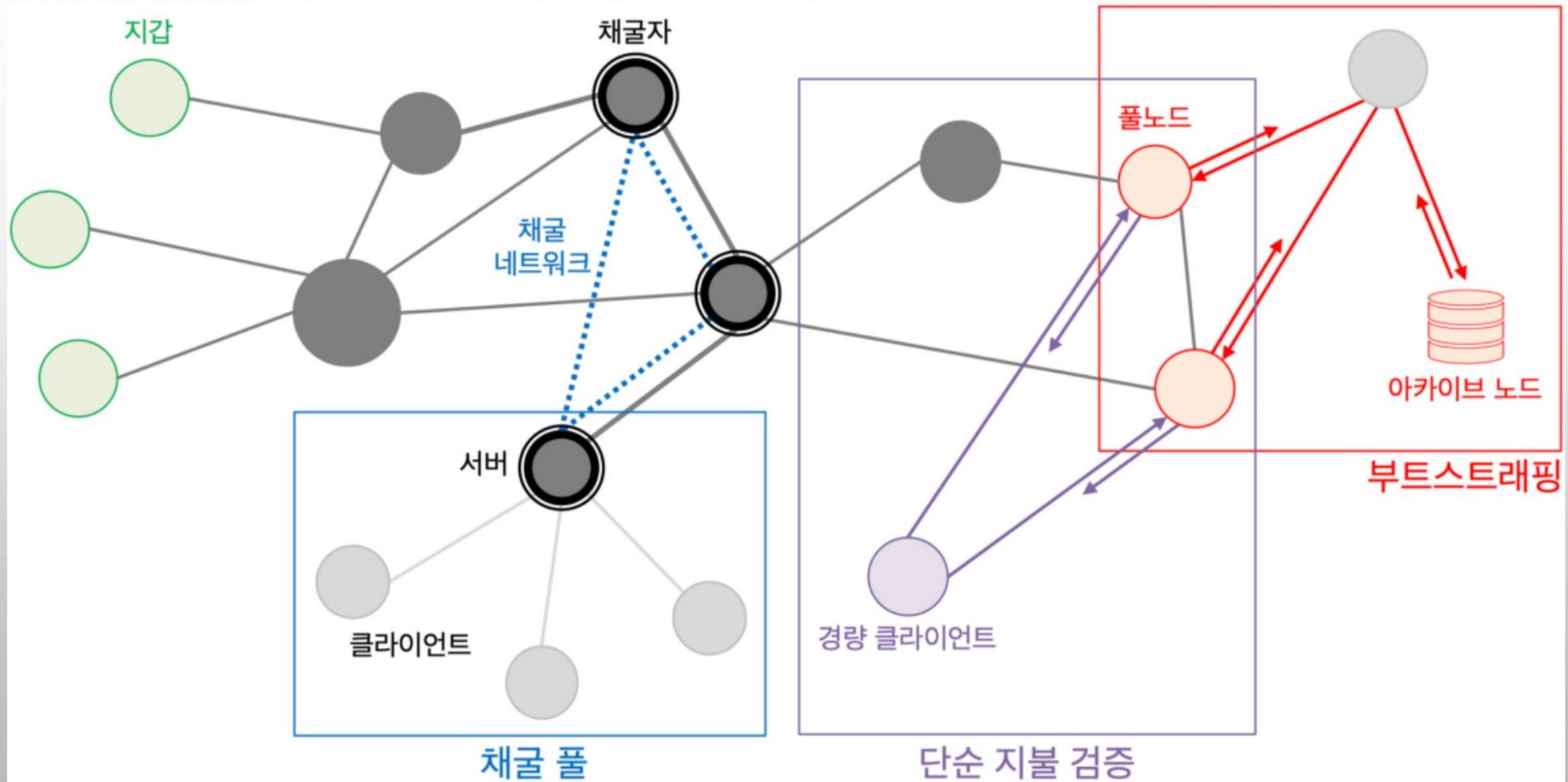
- 거래를 위한 검증
 - 송금자가 송금하고자 하는 금액 확인
 - 수신자 월렛 주소와 송금액 입력
 - 송금자의 잔고가 송금액 보다 크다는 것을 검증
 - 송금자 계좌에서 송금액 만큼 공제
 - 수신자 계좌에 송금액 만큼 증액

노드

- 노드: 블록체인 네트워크에 참여한 모든 컴퓨터 또는 사용자. 중앙 집중형 서버에 거래 기록을 보관, 관리하지 않고 네트워크에 참여하는 개개인의 서버들이 네트워크를 유지 및 관리
- 네트워크에 참여하기 위해서는 해당 네트워크의 프로그램을 설치 해야함
 - 비트코인: [Bitcoin Core](#)
- 풀노드, 라이트노드, 채굴노드, 랜덤노드 등 시간이 지나면서 다양한 유형의 노드가 생겨남



Node



블록

Block Hash : 블록의 대표값으로 해당 블록을 식별하는 기준

Header : 트랜잭션들에 대한 정보 요약 및 블록 관련 정보를 포함하며 header의 모든 정보를 하나의 값으로 표현한 것이 block hash

Body : 거래내역인 트랜잭션들을 모두 저장되어 있는 영역
개별 트랜잭션은 각각의 해시값을 가지고 이 해시값들을 연결하여 하나의 해시값으로 표현 => 이 값을 Merkle Root
라고 하며 이것이 Header에 저장

Hash of Block		
Version	Pre Block Hash	Merkle Hash
Time	Bits 난이도 설정 값	Nonce 임의의 값
Transaction Count		

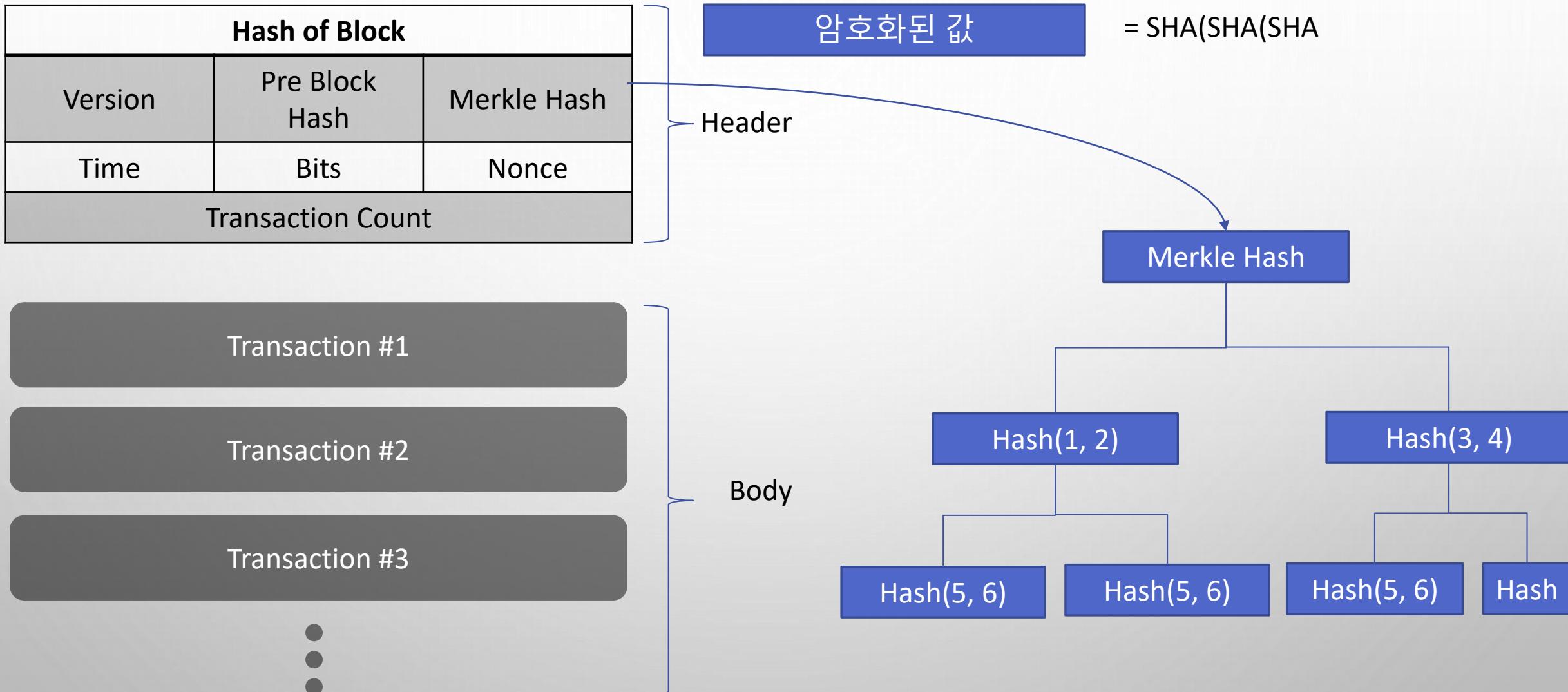
Transaction #1

Transaction #2

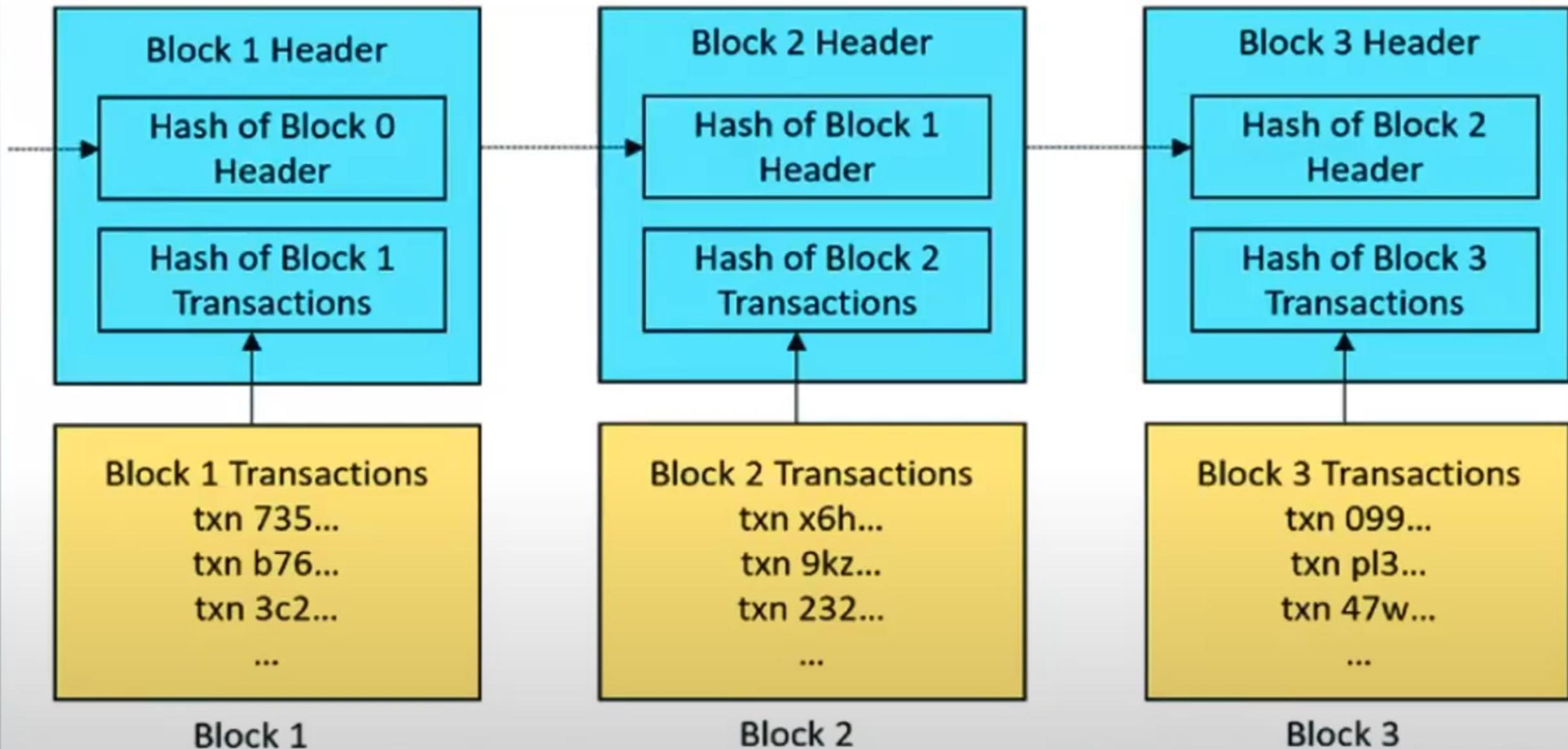
Transaction #3

⋮

블록의 구조

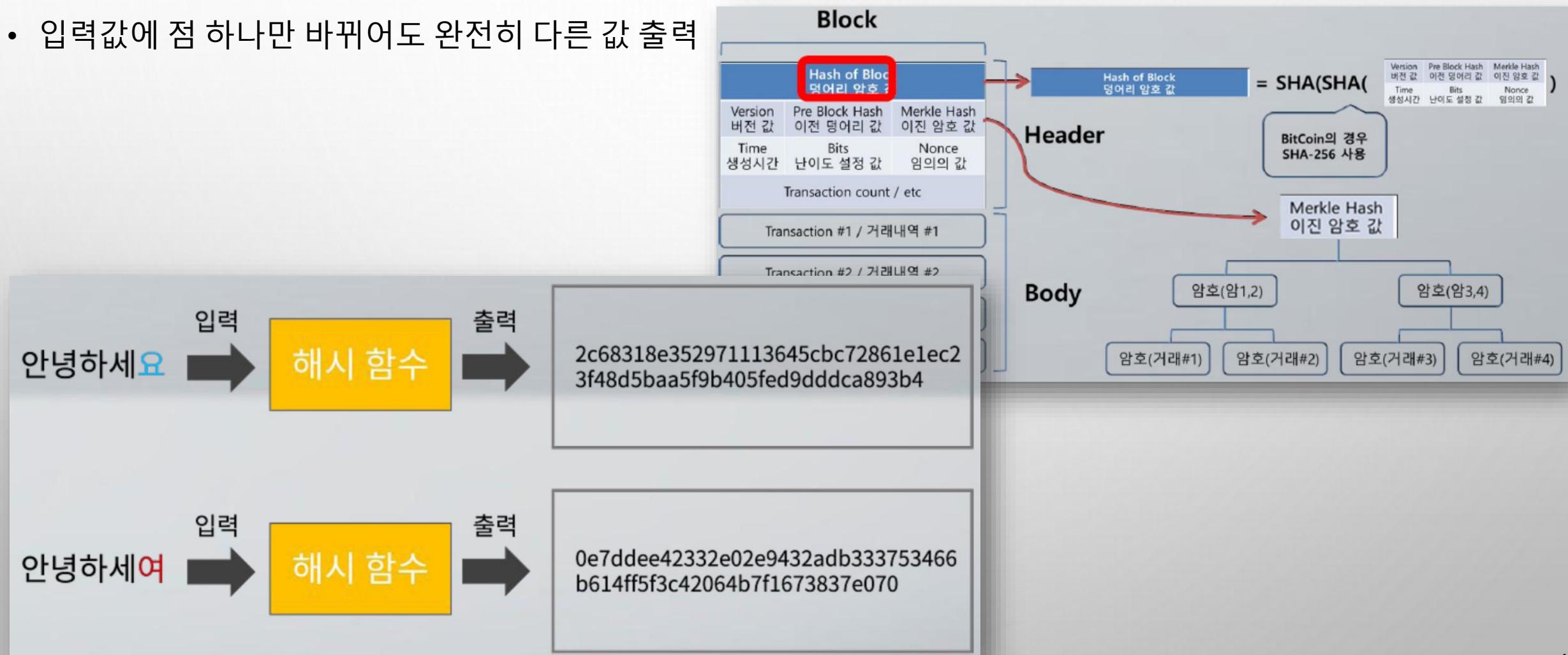


체인화



해시함수

- 임의 길이의 데이터를 해시함수에 입력하면 고정된 길이의 랜덤값을 출력하는 것
- 특정 입력값에 대해 어떤 값이 출력될지는 아무도 예측할 수 없으며 출력값을 가지고 입력값을 유추할 수 없음
- 입력값에 점 하나만 바뀌어도 완전히 다른 값 출력



Hash function test

- <https://emn178.github.io/online-tools/sha256.html>
- <http://hash-functions.online-domain-tools.com/>

Online Tools

SHA256

SHA256 online hash function

123

Input type: Text

Auto Update

a665a45920422f9d417e4867efdc4fb8a04a1f3fff1fa07e998e86f7f7a27ae3

OnlineDomainTools

Anonymous

Network Tools Web and Browser Tools Domain Tools Security and Privacy Tools Data and

Hash Functions Online

Jetpack Compose helped Monzo quickly build higher quality screens [Learn More](#)

Input type: Text

Input text:
123

Hash functions: MD4
 MD5
 NTLM
 SHA1
 SHA256
 SHA384

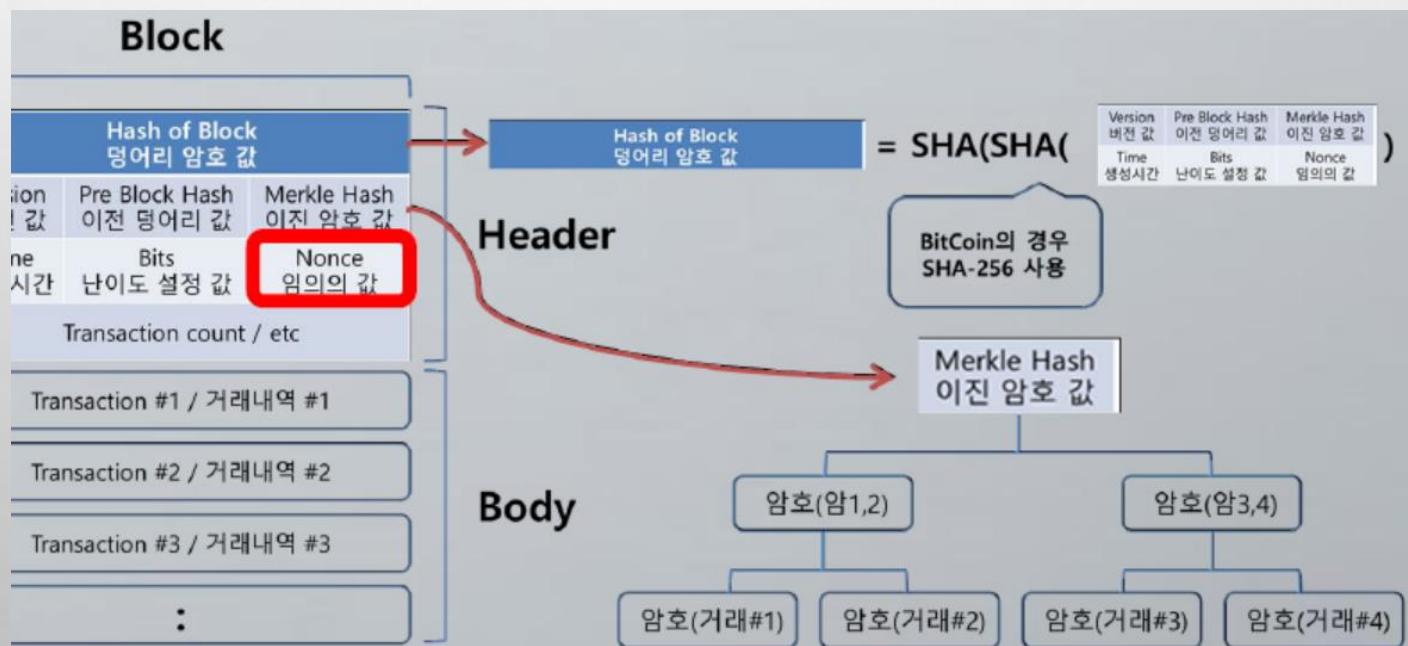
> Hash! [Share](#) [Copy](#)

Results

MD5: 202cb962ac59075b964b07152d234b70
SHA1: 40bd001563085fc35165329ea1ff5c5ecbdbbeef
CRC32: 9b0ead26

Nonce

- 논스: 아직 결정되지 않은 임의의 수
- 조건을 충족하는 논스 값을 가장 먼저 찾으면 최종적으로 블록의 해시값이 결정되고 블록이 생성됨(=채굴)
- 조건을 충족하는 논스 값은 컴퓨터의 복잡한 연산과정을 거쳐 찾게 되며, 논스 값을 찾았다는 것은 어려운 작업을 수행했다는 증명
- POW
- 논스 값이 결정되면 해당 네트워크의 메인 화폐가 발행되어 채굴자(블록생성자)에게 대가로 지급



비잔티움 장군 문제를 풀다!

(Proof-Of-Work 관점)

1. 300명의 병력이 있는 비잔틴성을 100명씩의 병력을 가진 장군 5명이 공격해야 함
2. 이기려면 적 병력보다 많은 병력이 공격해야 함
3. 장군들은 연락병을 보내 소통 가능함
4. 장군들 중에는 배신자가 있어 서로 신뢰 불가함

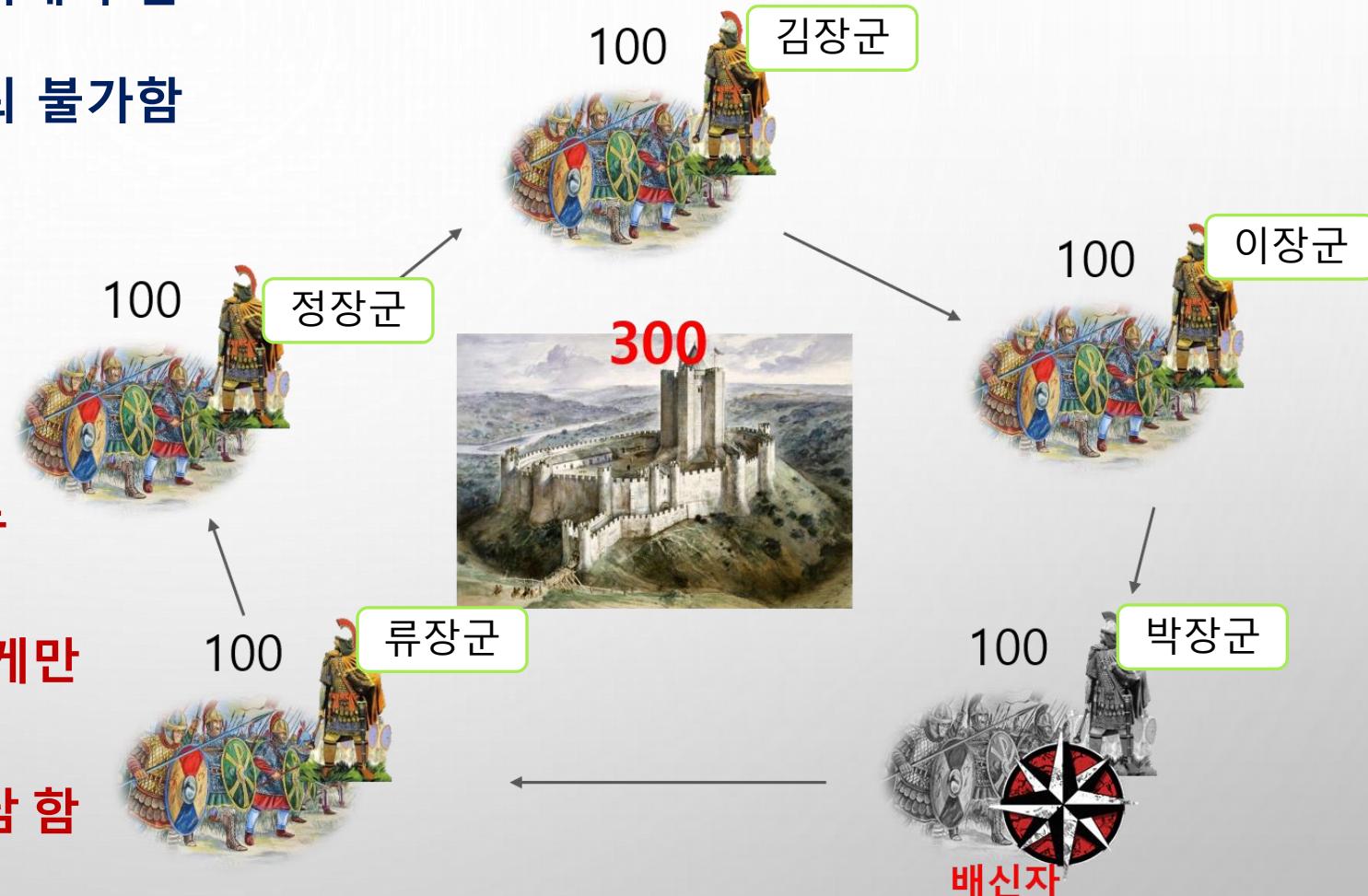
문제) 서로 신뢰할 수 없는데, 어떻게 공격 시각을 합의할 수 있는가?

가정 1 - 공격 시각은 상관 없음. 모두 한번에 공격할 수 있는 합의된 시각이면 됨

가정 2 - 배신자는 병력을 분산시키려 하므로 이전 장군의 메시지와 다른 시각을 제시함

가정 3 - 각 장군은 자신의 바로 다음 장군에게만 연락할 수 있음

가정 4 - 배신자도 지정한 시간에 공격에 가담 함



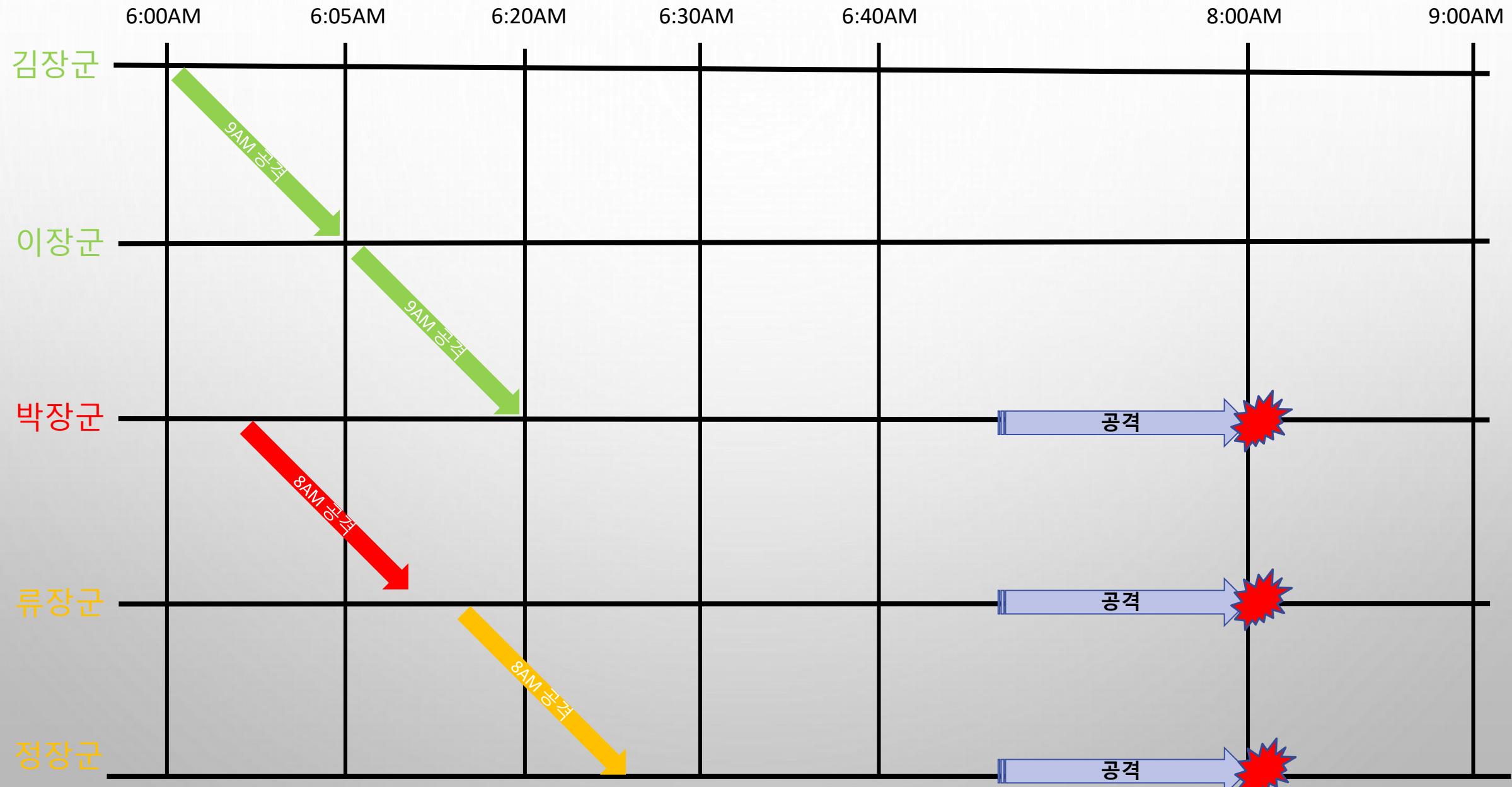
문제 발생

- 김장군이 "9 AM" 공격시각을 적어 서명과 함께 이장군에게 보냄
- 이장군은 김장군의 "9 AM" 공격시각을 보고 기억한 후 박장군에게 전달함
- 박장군은 배신자로, "9 AM" 메시지를 찢어버리고 "8 AM"으로 고쳐서 류장군에게 전달함
- 류장군은 "8AM"을 기억한 후 정장군에게 전달
- 정장군은 "8AM"을 기억
- 8AM에 박자군, 류장군, 정장군가 쳐들어가지만 300명이므로 지게 됨

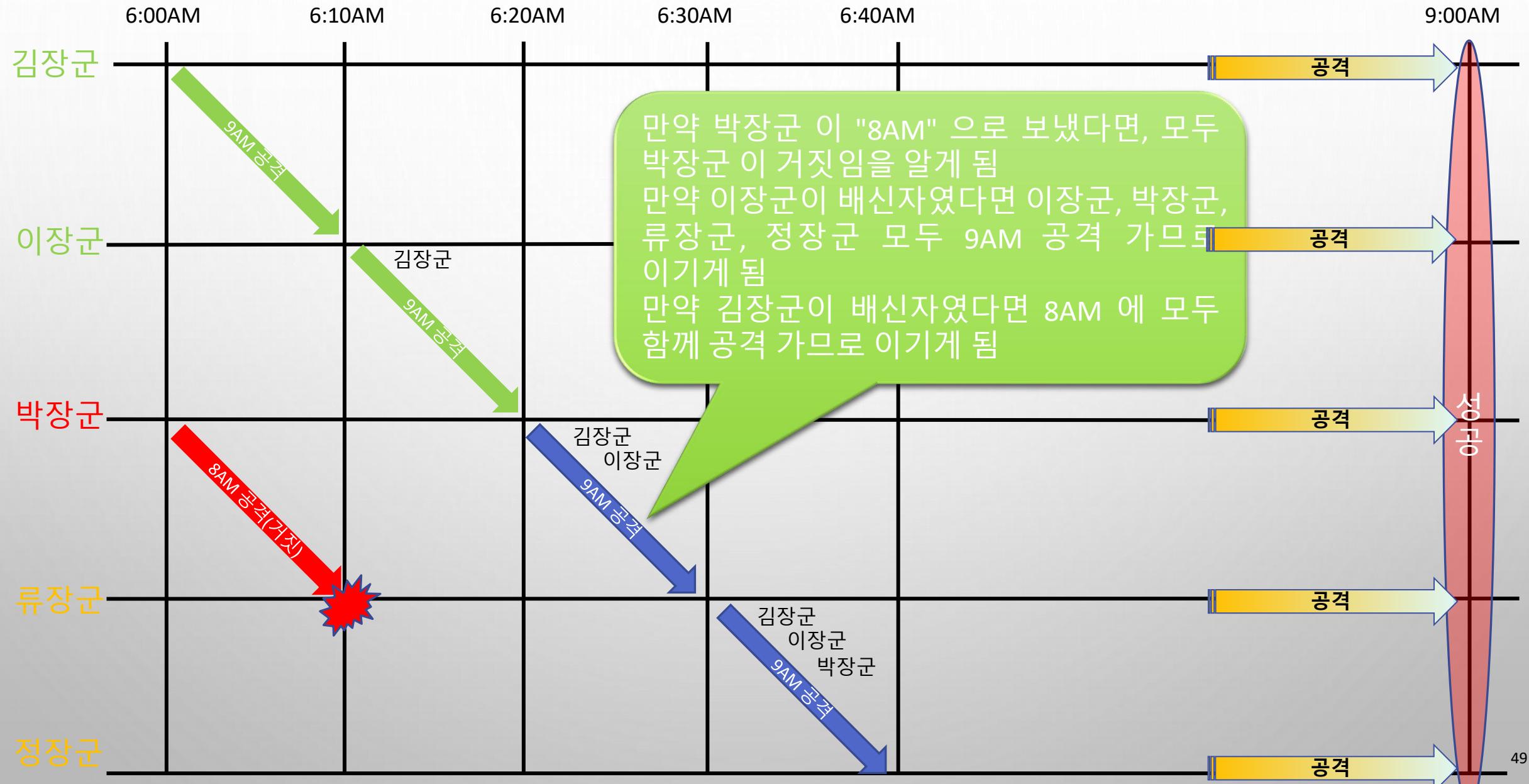
솔루션

- 규칙을 정의
 1. 장군은 메시지를 보내기 위해 반드시 **10분의 시간**을 들여야 함
 2. 메시지는 모든 이전 장군의 메시지와 **10분의 시간을 들였다는 증거를 포함**하여 보내야 함
- 김장군이 "9 AM" 공격시각을 적어 **10분간 작업하여 증거와 함께** 이장군에게 보냄
- 이장군은 "9 AM" 메시지와 김장군의 **10분 작업 증거를 보고 확신 후**, 박장군에게 "9 AM" 메시지 보냄
 - **이장군도 10분 간 작업 함**
 - **김장군, 이장군의 메시지와 작업내용 모두를 포함하여 보냄**
- 박장군은 배신자로 "8AM"으로 메시지를 수정하여 보내고 싶으나 그냥 보낼 수 없음. 아래 중 택 1
 - **속일 수 있는 유일한 방법**
 - **박장군은 10분보다 빠르게 작업을 하여 "8 AM" 메시지를 만들어 냄**
 - **김장군, 이장군의 총 20분 작업에 해당하는 메시지 모두를 남은 시간 내에 만들어 포함 시켜 보냄**
 - **들키지 않으려면 반드시 " 9 AM " 으로만 보내야 함**
- 류장군, 정장군 모두 동일

기존 TIMESTAMP



TIMESTAMP



비잔티움 장군 문제 해결과 이슈

대표적인 증명 : POW

- POW = Proof-Of-Work = 작업 증명 = Consensus Algorithm(합의 알고리즘)
- 과반 이상의 참여를 통해 비잔티움 장군 문제를 해결
- 증명에 참여자들에게 보상 근거를 마련 : BTC, BTH, LTC, ETH, ETC

한계점

- 51% 공격 가능 = 51%의 해시 파워로 공격

극복방안

- 해시 파워 과반 유지 = 팬덤 유지

비잔티움 장군 문제 해결 통해 얻고자 주요 성질

신뢰성

트랜잭션 정보를 저장장치에
저장시 훼손 방지

무결성

인위적인 개입으로 정보
변경 불가

- 데이터(디지털 코드 혹은 정보)의 훼손과 변경을 방지
- (연산 + 네트워크)*상호작용 => 변질되지 않은 디지털 정보를 얻고자

암호화폐를 포괄하는 블록체인만의 강력한 특징

익명성

개인정보 없는 참여와 거래
기존 지불수단보다 **높은 익명성**
제공

분산성

제3자의 **개입이 없는** 거래 가능
시스템 운용 및 유지보수 **불필요**

확장성

개방된 소스로 인한 연결과
확장이 용이

투명성

모든 거래 기록의 접근 가능
거래 양성화 및 규제 비용 절감

보안성

거래 데이터 조작 방지 및 무결성
보장

안정성

분산 구조로 인한 장애와 오류
제한적

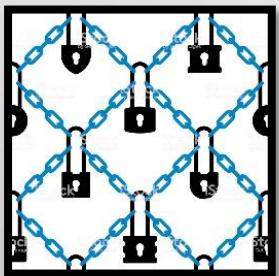
▪ 신뢰성 측면에서 블록체인 기술이 **압도적**

보안의 3요소로 이해하는 블록체인의 특징

기밀성

암호, 패스워드, 방화벽 등
비인가 정보접근의 **분산 보안**

블록체인 자물쇠



무결성

인위적인 개입으로 정보
변경 불가

거짓 정보 접근, 거짓된 처리 판별



가용성

상시적인 정보 자산 접근 가능,
장애, 오류 상태 확인에 **지연 오류를 낮춤**

블록체인 활용
상시 검증 가능



기존
오류, 장애 확인불가



▪**보안 측면에서 블록체인 기술이 압도적**

비잔티움장군 문제

분산된 작업을 통해서 무결성과 신뢰를 얻기 위한 솔루션을 찾는 문제

작업 증명 = POW = 합의 알고리즘(모델)

작업을 통해서 증명하고자 하는 것은 블록의 무결성(신뢰성)

블록을 생성하기 위한 연산을 통해 블록 생성에 합의하는 알고리즘(모델)

- 몇 가지의 블록체인 자체만으로는 장점과 단점이 공존

다양한 합의 모델

POW VS POS

POW

보상

해시 파워

장점

대부분의 코인
안정성이 높음

단점

높은 전력 소비
해시 유지
ASIC, GPU, CPU 활용
비효율

VS

POS

보상

- 코인의 지분
- 이자 : 에어드랍

장점

실질적 네트워크 분산
네트워크 자원 소비

단점

지분 쓸림
POW에 비해 효율적

POS VS DPOS

POS

대표 코인

퀀텀, 네오, 스트라티스

장점

해시 파워 불필요.
블록 생산자의 탈중앙화
지분을 담보

단점

에어드랍(코인이 묶임)
보안성에 대한 의심
지분 쓸림

VS

DPOS

대표 코인

스팀, 이오스, 아크, 라이즈

장점

소규모 참여에 대한 이익 공유
속도가 빠름

단점

탈중앙화는 아님
보안 취약

PBFT VS LFT

PBFT

대표 코인

리플, 하이퍼레저

장점

빠른 트랜잭션 처리
특정 노드(검증자)에 의한 검증

단점

제한된 참여
이익 집단 형태의 변질 가능
민간 참여에 대한 신뢰 한계
특정 노드(검증자) 공격 무방비

VS

LFT

대표 코인

아이콘

장점

검증된 Tangaroa 알고리즘 개선
2단계의 비잔틴 노드를 활용

단점

초기 모델
노드 검증 단계의 확장은 속도
한계
알고리즘에 의존적

POS VS POPS

POS

대표 코인

퀀텀, 네오, 스트라티스

장점

해시 파워 불필요.
블록 생산자의 탈중앙화
지분을 담보

단점

에어드랍(코인이 묶임)
보안성에 대한 의심
지분 쏠림

POPS

(Power Specification & Stake)

대표 코인

GMB

장점

연산형이 아닌 관계형 해시
블록체인 네트워크 안정성을 높인 모델
**해시와 지분 모형 결합을 통한 실용성 높
인 최초의 하이브리드 모델**

단점

초기 모델

VS

DPOS VS DPOR

DPOS

대표 코인

스팀, 이오스, 아크, 라이즈

장점

소규모 참여에 대한 이익
공유
속도가 빠름

단점

탈중앙화는 아님
보안 취약

VS

대표 코인

파이널 체인(FINAL CHAIN)

장점

보안성이 높고 속도가 빠름
노드의 물리적인 신뢰성을 기반
으로 **효율적이고 실용성을 높인**

DPOS 모델

단점

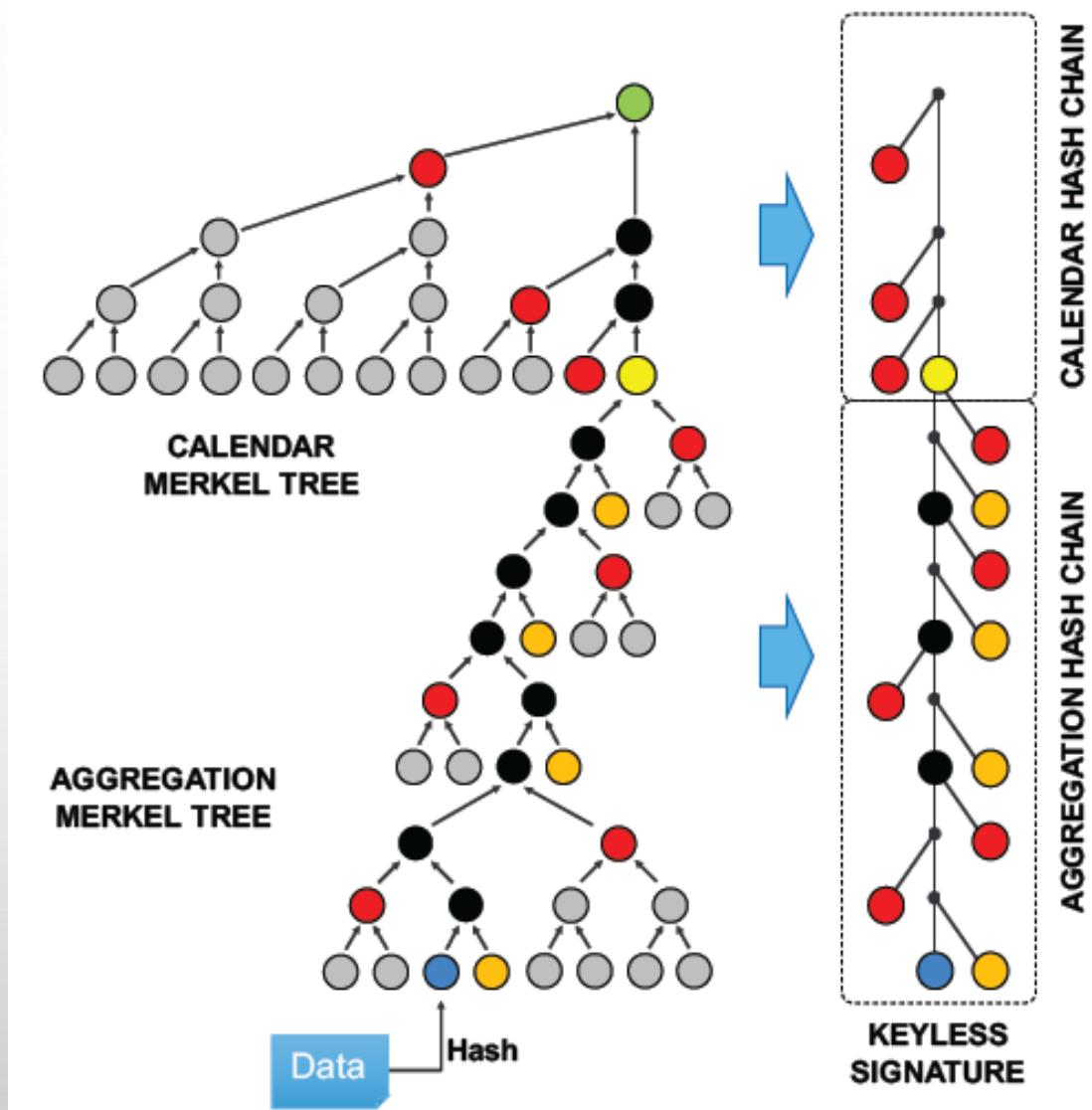
초기 모델

잘 알려지지 않은 합의 모델

Guardtime Keyless Signatures

Proof of Participation Model

에스토니아 e-Gov Blockchain



아직도 걸음마 단계...

P2P

Public vs Private

Did you find it?

Responsibility

POW, POS, DPOS

Is it for the trust
of the block?

Share vs Incentive

Storage

Node's Full Block Size
GB → TB → PB

Is it efficient?

오리건주 프린빌?
300,000 square feet

Segwit / Fork

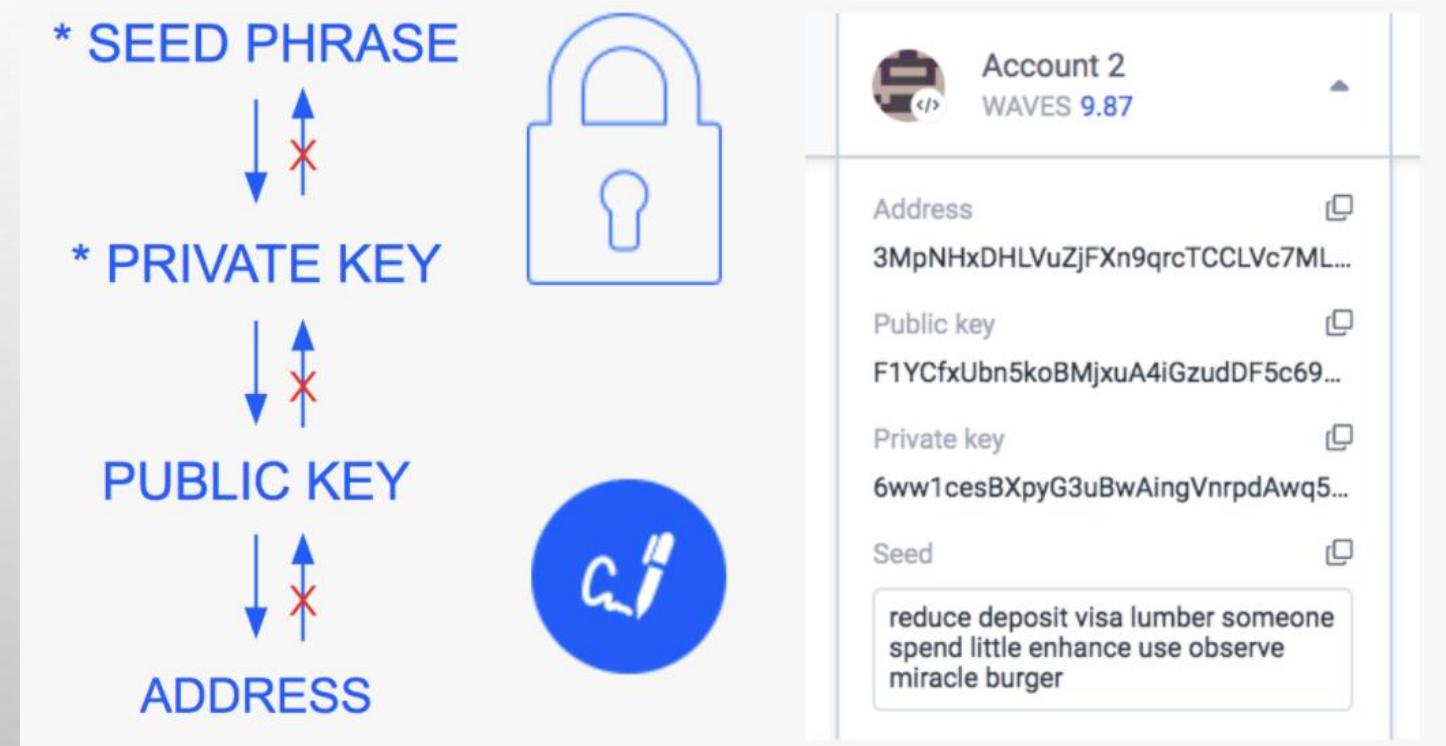
Delay
Transaction

Smart
Contract

Unreasonable
Fees

공개키/개인키

- 공개키: 은행의 '계좌번호'와 유사한 블록체인 지갑을 식별하는 식별번호. 개인키를 통해 생성되지만 공개키를 통해 개인키를 알아내는 것은 불가능
- 개인키: 블록체인 지갑을 만들 때 생성되며 생성된 개인키를 특정 함수에 적용하여 공개키를 얻어냄. 모든 거래는 개인 키로 거래를 승인해야 이루어짐
- 시드구문: 다수의 지갑에 접근할 수 있으며 하나의 시드구문에서 무한대의 개인키를 생성할 수 있음
- 주소: 공개키의 해시된 형식



종류

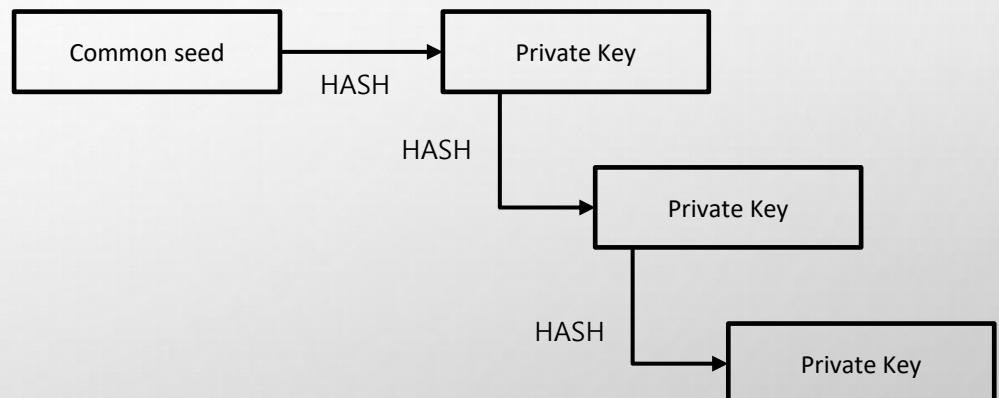
- **Nondeterministic(Random)**

- 무작위의 Private Key 사용
- 보안 문제 해결(여러 개의 지갑 사용)
- 거래시마다 새로운 지갑



- **Deterministic(Seeded)**

- 단방향 Hash 함수를 통해 개인키를 연속적 생성
- Private Key의 Depth에 따라 1:1 Maping 제한



- **Hierarchical Deterministic**

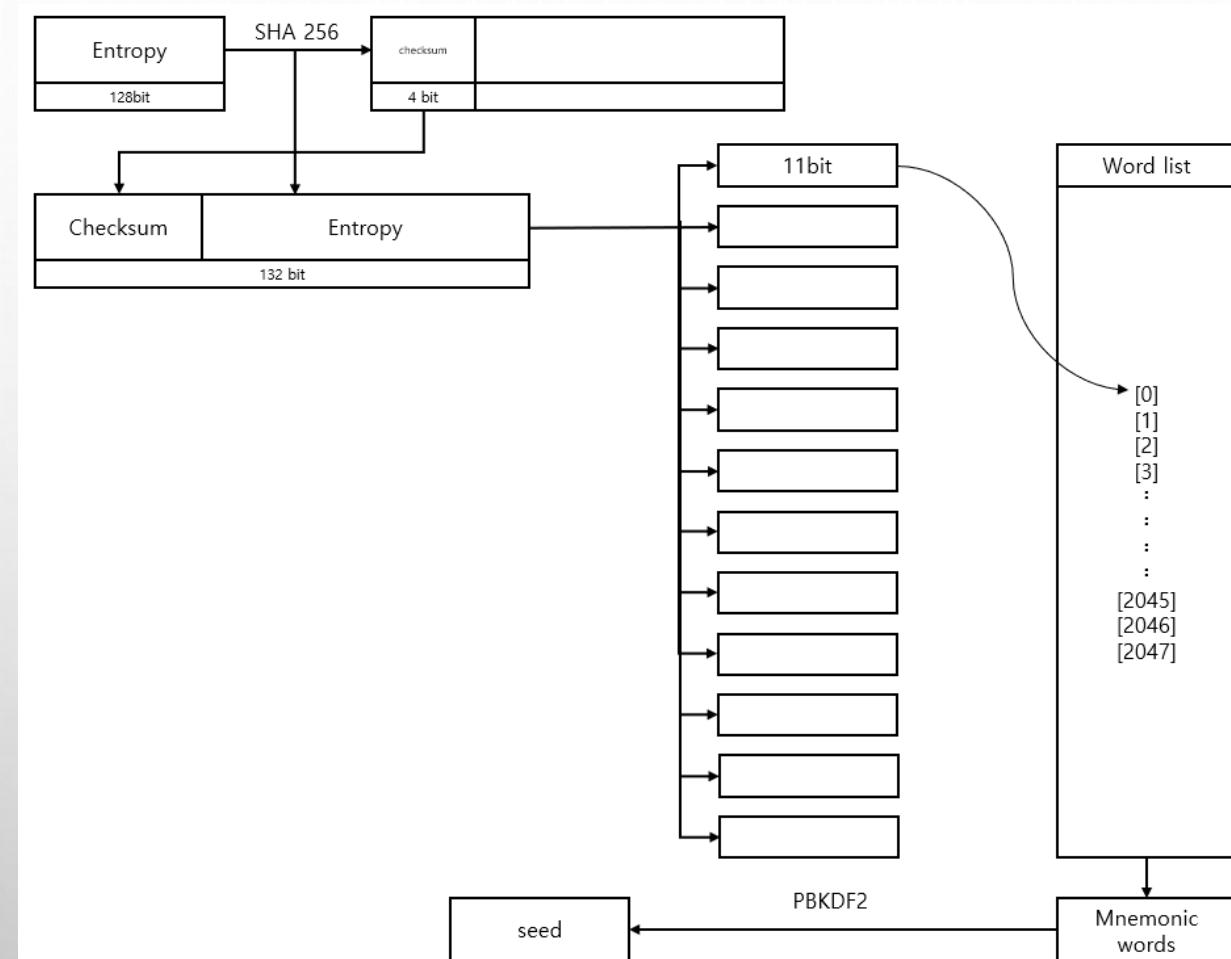
- Private Key의 Depth에 따라 1:N Maping 가능

SEED CREATE

- seed : 128~256 bits 무작위성 RandomBytes
- 복원 : Mnemonic word 로 치환
- DIGEST = PBKDF2(PRF, Password, Salt, c, DLen)
 - PRF : 난수(HMAC)
 - Password : 패스워드
 - Salt : 암호화 솔트
 - c : 원하는 iteration 반복 수(default = 2048)
 - DLen : 원하는 다이제스트 길이(default = 64 Byte)
 - DIGEST = PBKDF2(PRF, Password, Salt, c, DLen)
- CHECKSUM = ENTROPY / 32
- MNEMONIC WORD = (CHECKSUM + ENTROPY) / 11

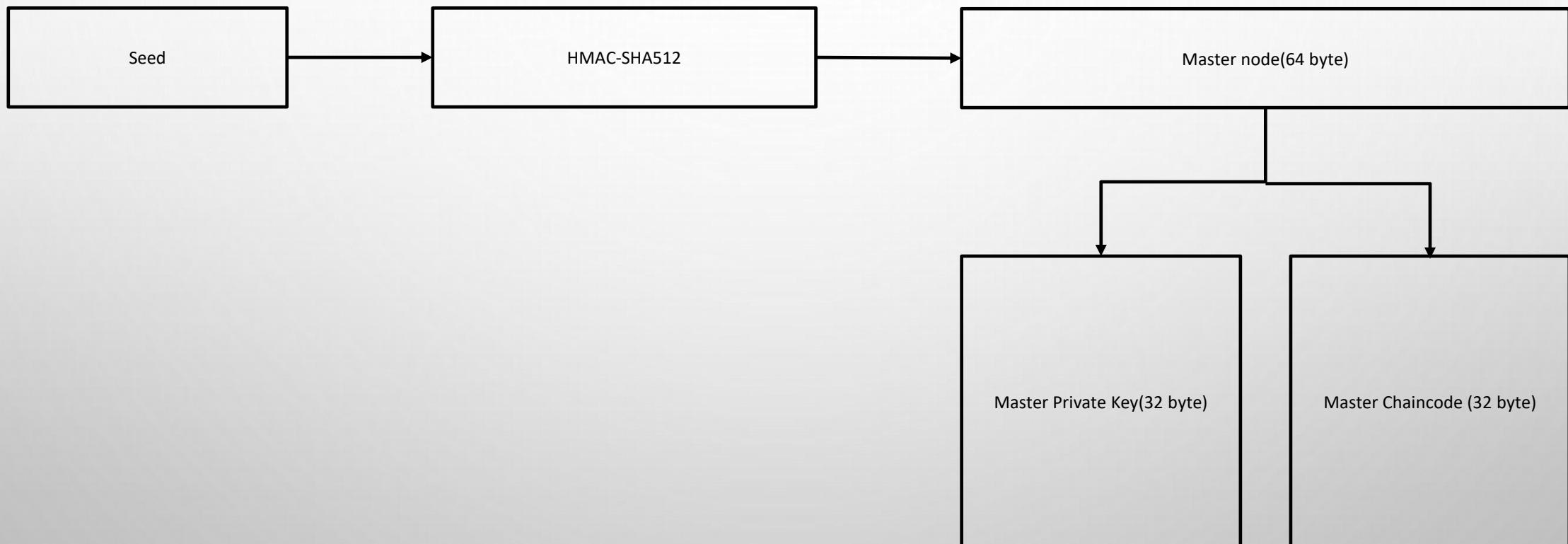
ENTROPY	CHECKSUM	ENTROPY + CHECK SUM	MNEMONIC WORD
128	4	132	12
160	5	165	15
192	6	198	18
224	7	231	21
256	8	264	24

ENTROPY 증가를 위한 MNEMONIC MAPPING



MASTER NODE CREATE

- seed 를 통한 Master Private Key와 Master Chain Code 생성



EXTENDED KEY CREATE

- publicKey : 선택한 알고리즘에 의해 생성되는 public key
- Extended Public Key
- Extended Private Key
- Identifier : 해시값(MD160.(Sha256(Public Key)))
- FingerPrint : Identifier의 Check Sum
- Index : Normal Key와 Hardened Key를 구분 지으며, Identity로 계층을 구분 짓는 역할
- Depth : BIP44의 Tree구조의 현재 depth를 나타냄

Extended (Public / Private) Key							
Version		Depth	Parent Fingerprint	Index		Chain Code	Public Key or '0'+Private Key
32bit		8bit	32bit	32bit		32byte(256bit)	33byte(264bit)
Network	BITCOIN	m : 0x00 m/0 : 0x01 m/0/0 : 0x02 . .	0x00000000 if master key	Normal (public)	i	I = HMAC-sha512()	Public Key
Public	0x0488b21e			Hardened (private)	i + 2^31		'0x00' + Private Key
Private	0x0488ade4		.			IR	
78 Byte (위의 모든 데이터를 이어 붙여서 Base58로 encoding한다.)							

CKD(CHILD KEY DERIVATION)

- **Hardened**

- 부모의 Private Key를 인자로 이용하여 자식의 Private Key를 생성, 입출금 가능
- Hardened Key (Private Key -> child Private Key)
- $I = \text{HMAC-sha512}(\text{Parent.chainCode}, '0x00' + \text{Parent.privateKey})$
- $IL = I.\text{slice}(0,32) \Rightarrow \text{child.PrivateKey}$
- $IR = I.\text{slice}(32) \Rightarrow \text{child.chainCode}$

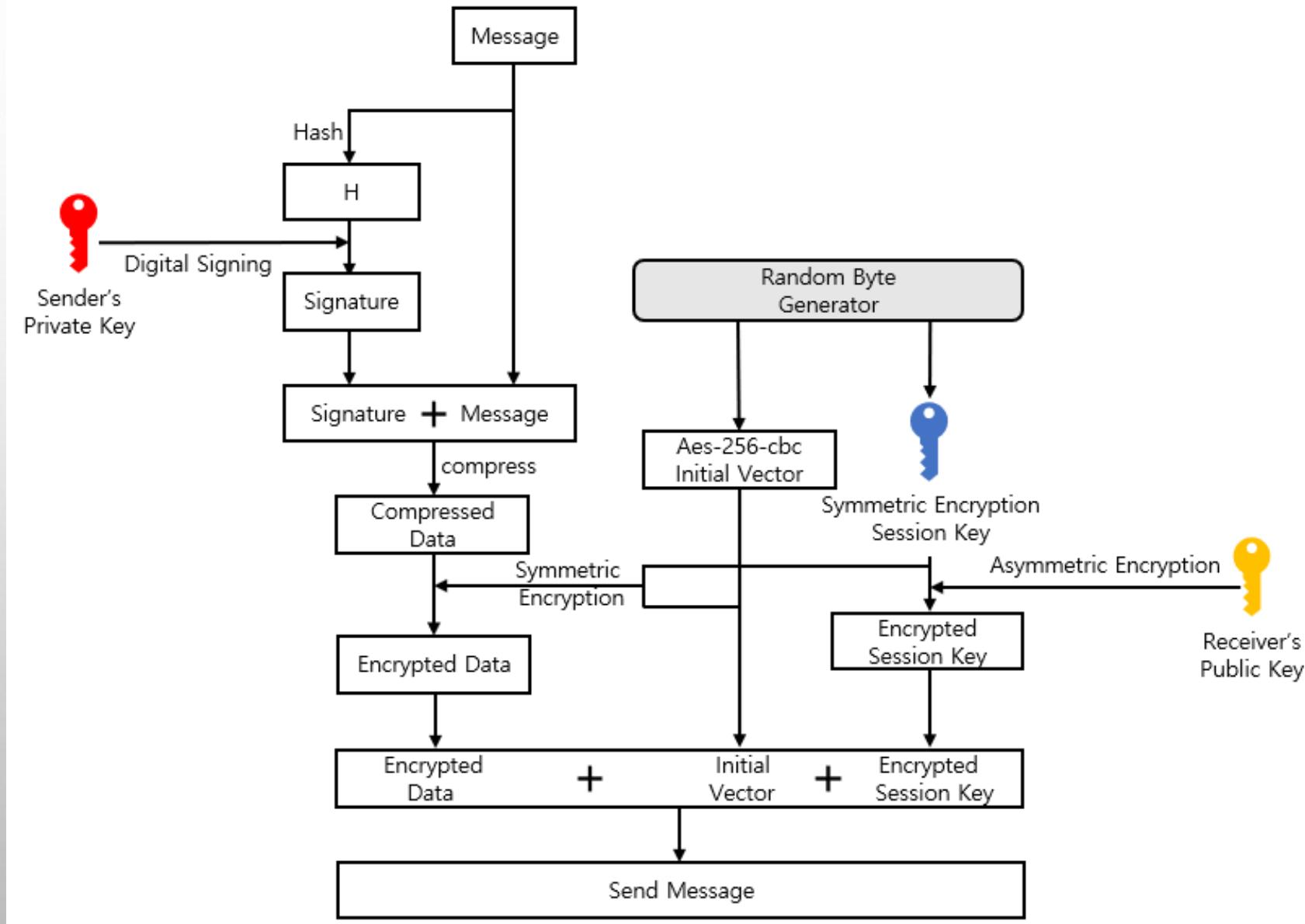
- **Normal**

- 부모의 Private Key를 가져와 Public Key를 계산하고 계산한 Public Key를 인자로
- 자식의 Private Key를 생성한다. Hardened와 마찬가지로 Private Key로부터 파생되었기 때문에 입출금이 가능하다.
- Normal key (Public Key -> child Private Key)
- $I = \text{HMAC-sha512}(\text{Parent.chainCode}, \text{Parent.publicKey})$
- $IL = I.\text{slice}(0,32) \Rightarrow \text{child.PrivateKey}$
- $IR = I.\text{slice}(32) \Rightarrow \text{child.chainCode}$

- **Neutered**

- Private Key가 없는 방식으로, 부모의 Public Key를 통해 자식의 Public Key만 생성한다. Private Key가 없기 때문에 입금만 가능하고 출금은 불가능하다.
- Neutered Key (Public Key -> child Private key)
- $I = \text{HMAC-sha512}(\text{Parent.chainCode}, \text{Parent.publicKey})$
- $IL = I.\text{slice}(0,32) \Rightarrow \text{child.publicKey}$
- $IR = I.\text{slice}(32) \Rightarrow \text{child.chainCode}$

송신 서명 처리



송신 서명 처리(코드 #1)

서명 데이터 생성

```
int Transaction::Run(Form* C_Transaction_Form, LoginInformation* C_LOGIN)
{
    Key::Sig* C_SIG = new Key::Sig();
    Kafka::KafkaInfomation* S_KAFKA_INFO = new Kafka::KafkaInfomation();
    Kafka* C_KAFKA = new Kafka();
    Boost_TCP::Client* C_BOOST_TCP_CLIENT = new Boost_TCP::client();
    try
    {
        std::string sigData;
        Json::Value Root;
        Json::Value NoteRoot;
        Json::StyledWriter writer;

        Root["Revision"] = C_Transaction_Form->Revision;
        SigData += std::to_string(C_Transaction_Form->Revision);

        Root["PreviousKeyID"] = C_Transaction_Form->PreviousKeyID;
        SigData += C_Transaction_Form->PreviousKeyID;

        Root["ContractCreateTime"] = C_Transaction_Form->ContractCreateTime;
        SigData += C_Transaction_Form->ContractCreateTime;

        Root["Fintech"] = C_Transaction_Form->Fintech;
        SigData += std::to_string(C_Transaction_Form->Fintech);

        Root["From"] = C_Transaction_Form->From;
        SigData += C_Transaction_Form->From;

        Root["Balance"] = C_Transaction_Form->Balance;
        SigData += C_Transaction_Form->Balance;

        Root["NotePrivacy"] = C_Transaction_Form->NotePrivacy;
        SigData += std::to_string(C_Transaction_Form->NotePrivacy);

        int XorSize = 0;
        char* XorData = NULL;
        for (int i = 0; i < C_Transaction_Form->Notesize; i++)
        {
            /* Note */
            NoteRoot[i]["To"] = C_Transaction_Form->S_NOTE[i].To;
            NoteRoot[i]["Kind"] = C_Transaction_Form->S_NOTE[i].Kind;
            NoteRoot[i]["Fee"] = C_Transaction_Form->S_NOTE[i].Fee;
        }
    }
```

송신 서명 처리(코드 #2)

```
C_Transaction_Form->SigSize = SigData.length() + XorSize;  
C_Transaction_Form->SigData = (char*)malloc(sizeof(char) * C_Transaction_Form->SigSize));  
memset(C_Transaction_Form->SigData, 0x00, C_Transaction_Form->SigSize);  
  
strcpy(C_Transaction_Form->SigData, SigData.c_str());  
memcpy(C_Transaction_Form->SigData + SigData.length(), XorData, XorSize);  
  
//SigData += Writer.write(NoteRoot[0]);  
//SigData.erase(remove_if(SigData.begin(), SigData.end(), isspace), SigData.end());  
  
if (C_LOGIN->Algorithm == NID_ED25519)  
{  
    char* SigKey;  
    C_SIG->Ed25519((char*)C_LOGIN->PrivateKey.c_str(), C_Transaction_Form->SigData, C_Transaction_Form->SigSize, &SigKey);  
    Root["Signature"] = SigKey;  
    SAFE_FREE(SigKey);  
}  
else if (C_LOGIN->Algorithm == NID_secp256k1 || C_LOGIN->Algorithm == NID_X9_62_prime256v1)  
{  
    char* SigR;  
    char* SigS;  
    C_SIG->ECC((char*)C_LOGIN->PrivateKey.c_str(), C_Transaction_Form->SigData, C_Transaction_Form->SigSize, &SigR, &SigS, C_LOGIN->Algorithm);  
    Root["Signature"] = std::string(SigR) + std::string(SigS);  
    //Root["KeyR"] = SigR;  
    //Root["KeyS"] = SigS;  
    SAFE_FREE(SigR);  
    SAFE_FREE(SigS);  
}
```

송신 서명 처리(코드 #3)

SHA256 암호화키

Private Key를 EVP_PKEY 구조화

서명 초기화

서명 단계

```
int Key::sig::Ed25519(char* PrivateKey, char* Data, int dataSize, char** sigKey)
{
    OPENSSL_init();

    uint8_t hash[32];
    SHA256_CTX sha256;
    SHA256_Init(&sha256);
    SHA256_Update(&sha256, Data, dataSize);
    SHA256_Final(hash, &sha256);
    uint8_t* ed_pkey = OPENSSL_hexstr2buf(PrivateKey, NULL);

    EVP_PKEY* p_pkey = EVP_PKEY_new_raw_private_key(NID_ED25519, NULL, ed_pkey, 32);

    EVP_MD_CTX* p_mdctx = NULL;
    p_mdctx = EVP_MD_CTX_new();

    if (p_mdctx == NULL)
    {
        throw std::runtime_error("Failed to create md ctx : openssl");
    }

    if (EVP_DigestSignInit(p_mdctx, NULL, NULL, NULL, p_pkey) != 1)
    {
        throw std::runtime_error("Failed to DigestSignInit : openssl");
    }

    size_t sig_len = 64;

    uint8_t* signature = (uint8_t*)OPENSSL_malloc(sizeof(uint8_t) * 64);
    if (!signature)
    {
        throw std::runtime_error("Not signature");
    }

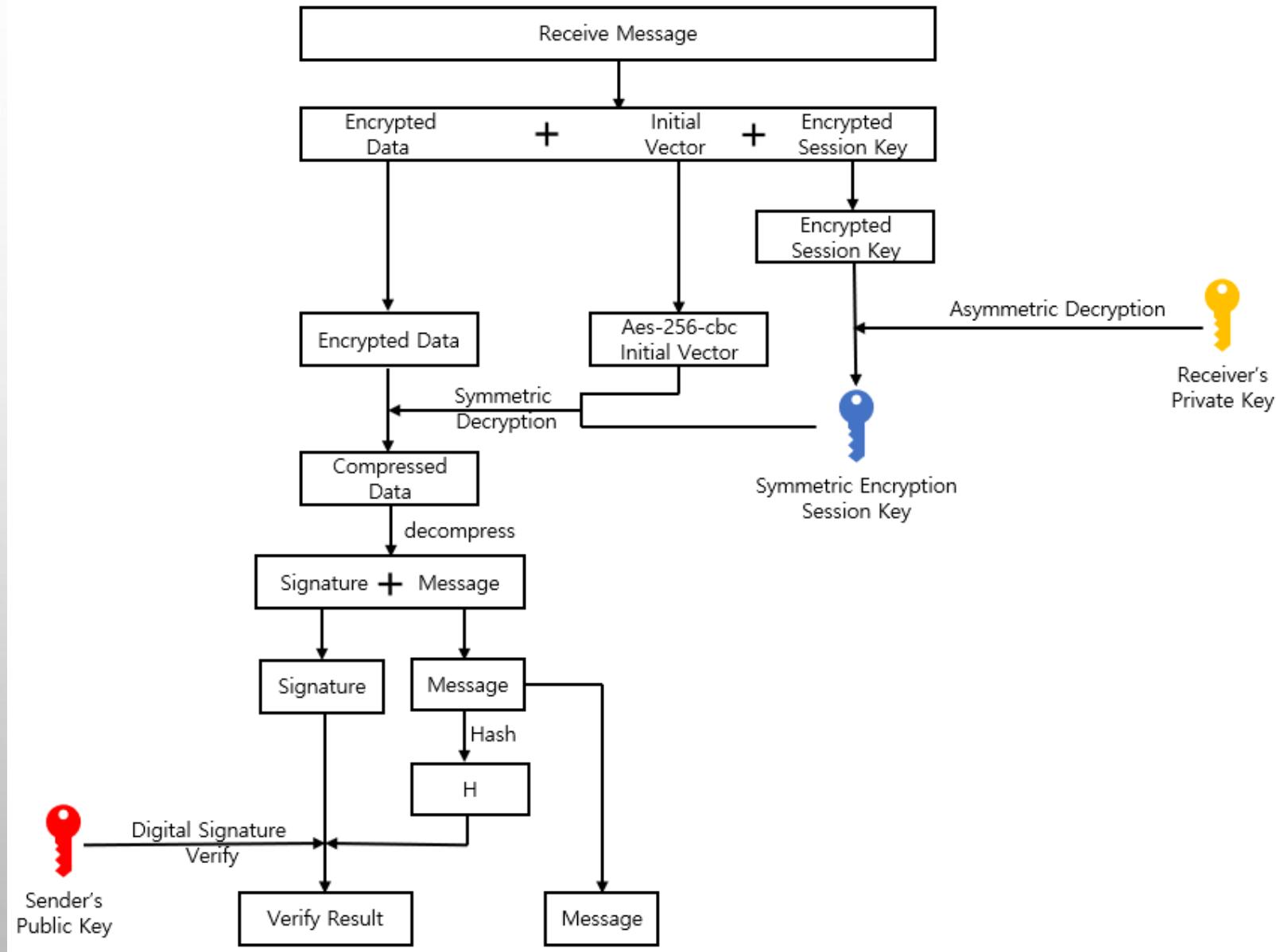
    if (EVP_DigestSign(p_mdctx, signature, &sig_len, hash, 32) != 1)
    {
        throw std::runtime_error("Failed to sign a message : openssl");
    }

    HexToCharArray(signature, 64, sigKey);

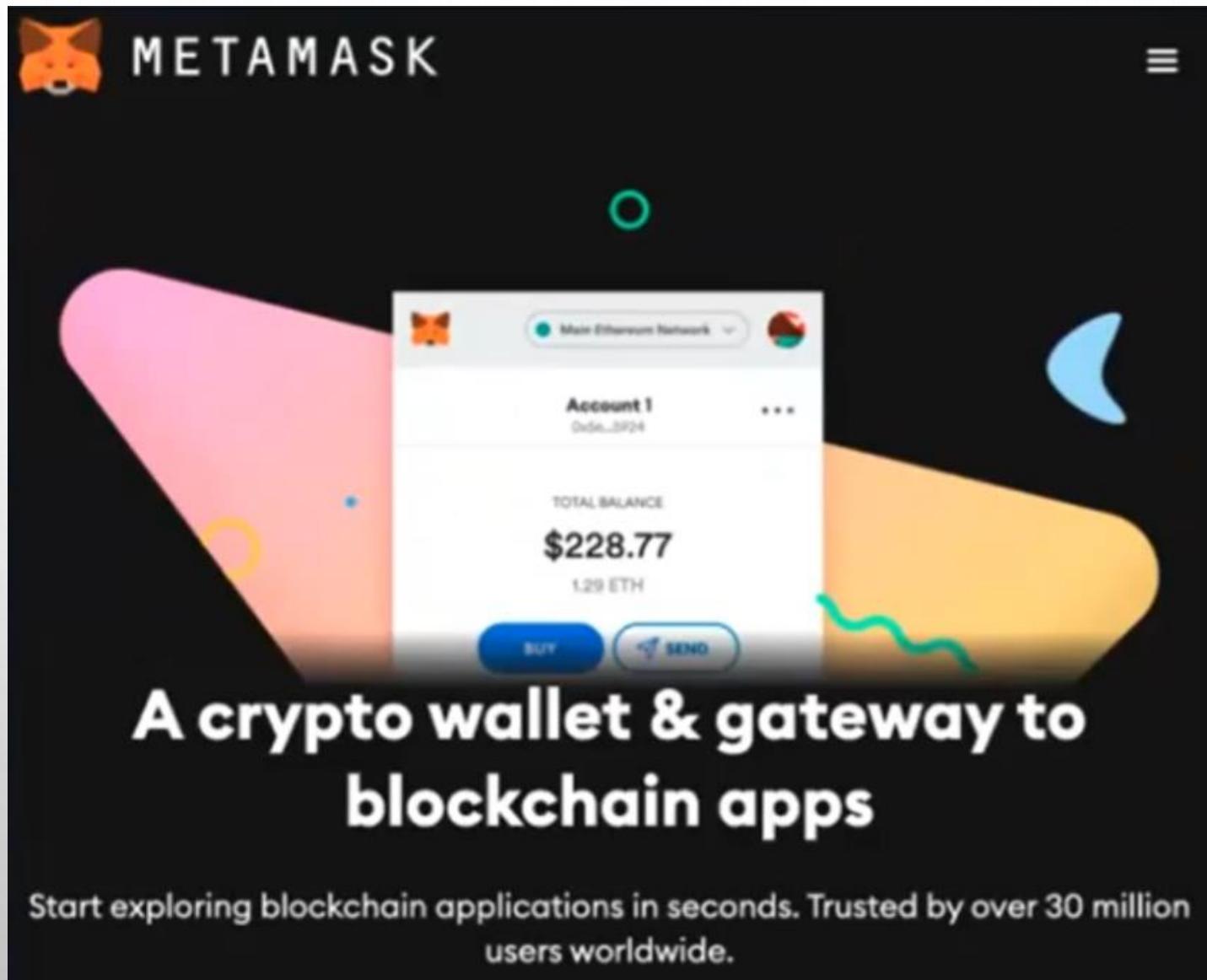
    //free(ed_pkey);
    //free(p_pkey);
    SAFE_FREE(signature);
    EVP_MD_CTX_free(p_mdctx);

    return SUCCESS;
}
```

수신 서명 처리



메타마스크 설치 #1



메타마스크 설치 #2



METAMASK

MetaMask가 처음이세요?



아니요. 이미 비밀 복구 구문이 있습니다.

비밀 복구 구문을 사용하여 기존 지갑 가져오기

지갑 가져오기



설정을 시작하죠!

이렇게 하면 새 지갑과 비밀 복구 구문이 만들어집니다

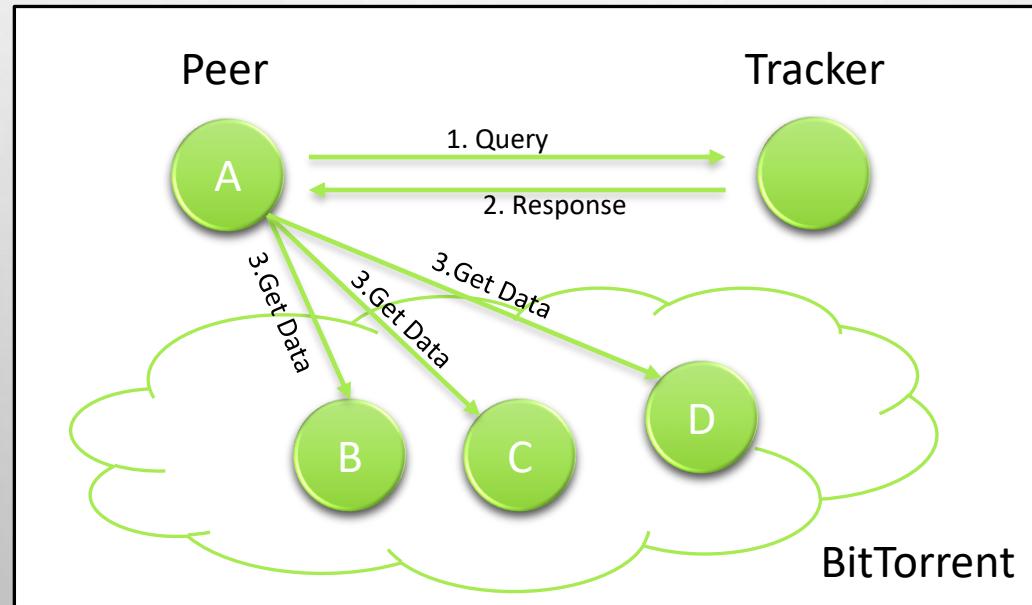
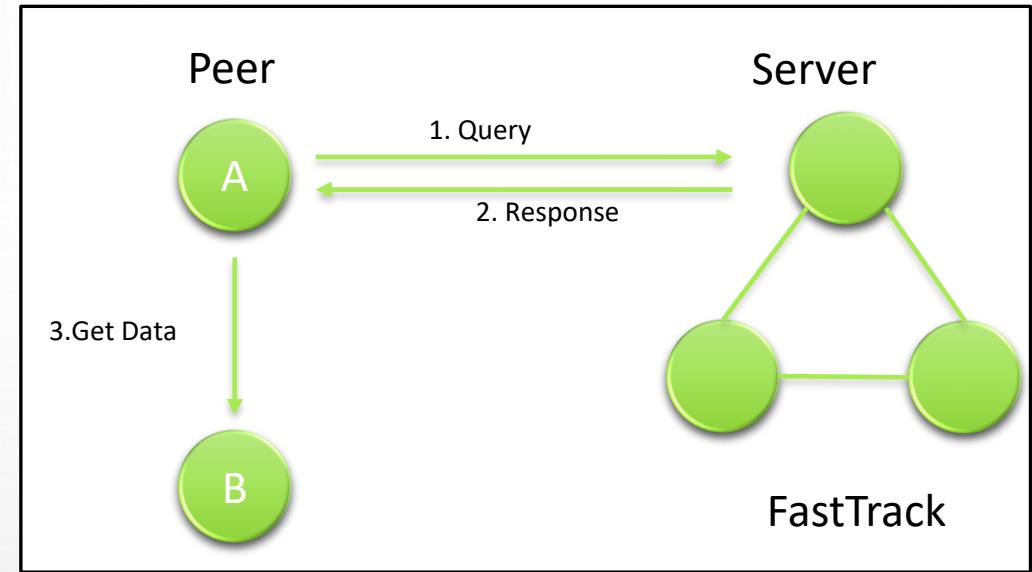
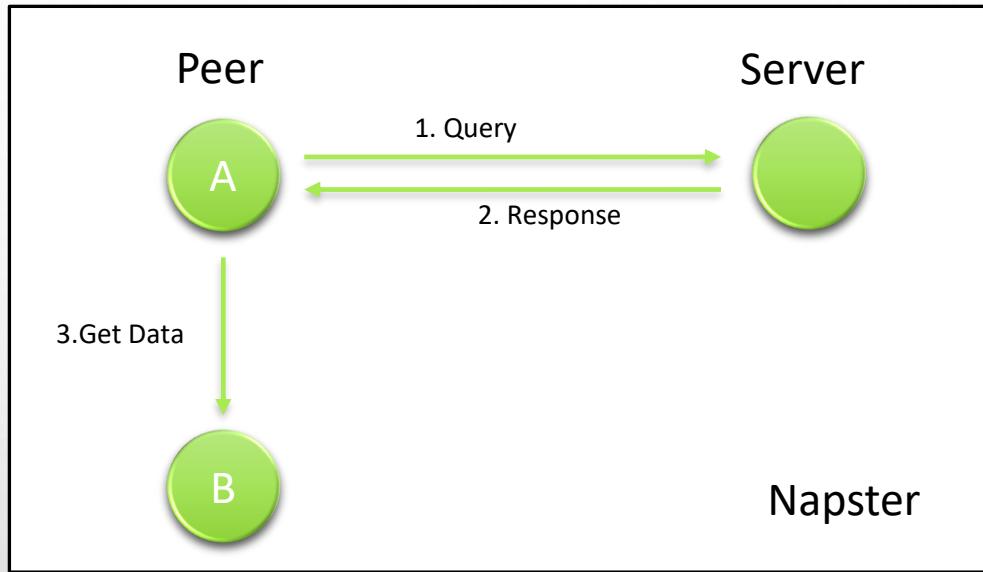
지갑 생성

P2P

Overlay Type

- **Structured system : impose particular structures on the overlay networks**
 - Distributed Hash Table
 - The topology of the peer network is tightly controlled
 - Any file can be located in a small number of overlay hops
- **Unstructured system – do not impose any structure on the overlay networks or loosely structured**
 - Napster, Gnutella, Freenet, BitTorrent
 - Usually resilient to peer dynamics
 - Support complex search based on file metadata

Unstructured P2P system



Structured P2P system

- Structure : to accommodate participating **nodes and data** in the overlay
- Routing algorithm : to locate **nodes** in the overlay and **insert / retrieve data to/from** them
- Join/Leave mechanisms : to enable **self-organization** and **fault tolerance**
- Structures (geometries)
 - rings, trees, hypercubes, tori, XOR, ...
 - deterministic lookup (특정 시간 보장)
 - Support complex search based on file metadata
 - performance : node array, overlay structure (arrive, leave)

DHT

- Distributed Hash Table
 - Put(Key, Value)
 - Get(Key)->Value
 - $\log(n)$ lookup
- $\log(n)$ lookup
- traceroute (ttl)

```
// Function: SetTtl
// Description: Sets the TTL on the socket.
int SetTtl(SOCKET s, int ttl)
{
    int      optlevel, option, rc;

    rc = NO_ERROR;
    if (gAddressFamily == AF_INET)
    {
        optlevel = IPPROTO_IP;
        option   = IP_TTL;
    }
    else if (gAddressFamily == AF_INET6)
    {
        optlevel = IPPROTO_IPV6;
        option   = IPV6_UNICAST_HOPS;
    }
    else
    {
        rc = SOCKET_ERROR;
    }

    if (rc == NO_ERROR)
    {
        rc = setsockopt(s, optlevel, option, (char *)&ttl, sizeof(ttl));
    }

    if (rc == SOCKET_ERROR)
    {
        fprintf(stderr, "SetTtl(): setsockopt() failed with error code %d\n", WSAGetLastError());
    }
    else
        printf("SetTtl(): setsockopt() should be fine!\n");
}

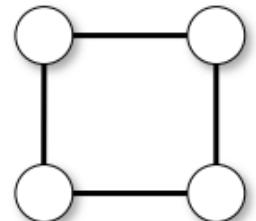
return rc;
}
```



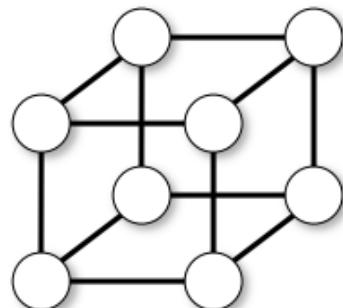
$$d = 0 \\ N = 1$$



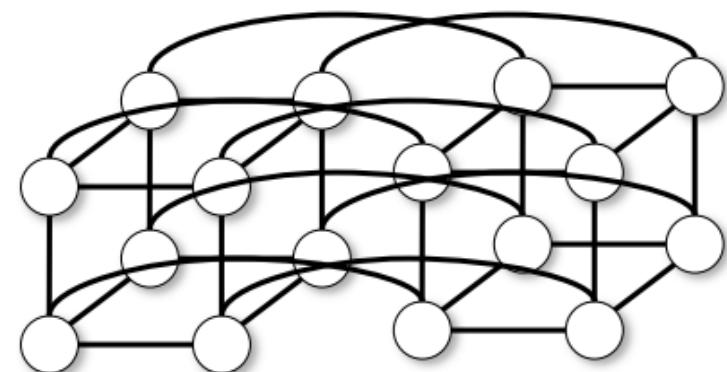
$$d = 1 \\ N = 2$$



$$d = 2 \\ N = 4$$



$$d = 3 \\ N = 8$$



$$d = 4 \\ N = 16$$

DHT

```
ttl = 1;

// Start sending the ICMP requests
do
{
    notdone = 1;

    SetTtl(s, ttl);

    // Set the sequence number and compute the checksum
    SetIcmpSequence(icmpbuf);
    ComputeIcmpChecksum(s, icmpbuf, packetlen, dest);
    // Send the ICMP echo request
    time = GetTickCount();
    rc = sendto(s, icmpbuf, packetlen, 0, dest->ai_addr, dest->ai_addrlen);
    if (rc == SOCKET_ERROR)
    {
        fprintf(stderr, "sendto() failed with error code %d\n", WSAGetLastError());
        return -1;
    }
    else
        printf("sendto() is OK\n");

    // Wait for a response
    rc = WaitForSingleObject((HANDLE)recvvol.hEvent, gTimeout);
    if (rc == WAIT_FAILED)
    {
        fprintf(stderr, "WaitForSingleObject() failed with error code %d\n",
GetLastError());
        return -1;
    }
    else if (rc == WAIT_TIMEOUT)
    {
        printf("Request timed out.\n");
    }
    else
```

```
{

    printf("WaitForSingleObject() should be fine lol!\n");
    // Check for an error
    rc = WSAGetOverlappedResult(s, &recvvol, &bytes, FALSE, &flags);
    if (rc == FALSE)
    {
        fprintf(stderr, "WSAGetOverlappedResult() failed with error code %d\n",
WSAGetLastError());
    }
    else
        printf("WSAGetOverlappedResult() looks OK!\n");

    time = time - GetTickCount();
    WSAResetEvent(recvvol.hEvent);
    // See if we got an ICMP ttl expired or echo reply, if not ignore and receive again.
    if (AnalyzePacket(recvbuf, bytes) == NO_ERROR)
    {
        if (bResolve)
        {
            if(ReverseLookup((SOCKADDR *)&from, fromlen, hopname, hopbuflen) != NO_ERROR)
            {
                printf("ReverseLookup() failed lor!\n");
            }
            else
            {
                printf("TTL:%d Time:%d ms Hop name: %s [", ttl, time, hopname);
                PrintAddress((SOCKADDR *)&from, fromlen);
                printf("]\n");
            }
        }
    }
}
```

```
}

else
{
    printf("TTL:%d Time:%d ms ", ttl, time);
    PrintAddress((SOCKADDR *)&from, fromlen);
    printf("\n");
}

// See if the response is from the desired destination
notdone = IsSockaddrEqual(dest->ai_addr, (SOCKADDR *)&from);
// Increment the TTL
ttl++;
}

// Post another receive
if (notdone)
{
    fromlen = sizeof(from);
    PostRecvfrom(s, recvbuf, recvbuflen, (SOCKADDR *)&from, &fromlen, &recvvol);
}
}

Sleep(1000);
} while ((notdone) && (ttl < gTtl));
```

chord

- Each node has a **successor** and a **predecessor**
- Since nodes may disappear from the network, each node records several nodes preceding it and following it
- Each node also maintains information about (at most) m other neighbors, called **fingers**, in a **finger table**
- The i -th entry, $i = 1; 2; \dots; m$, in the finger table of node N points to the node whose ID is the smallest value bigger than or equal to $N + 2^{i-1}(\text{mod } 2^m)$ in the clock wise direction

```
void NodeInformation::fixFingers(){

    HelperFunctions help;

    int next = 1;
    long long int mod = pow(2,M);

    while(next <= M){
        if(help.isNodeAlive(successor.first.first,successor.first.second) == false)
            return;

        long long int newId = id + pow(2,next-1);
        newId = newId % mod;
        pair< pair<string,int> , long long int > node = findSuccessor(newId);
        if(node.first.first == "" || node.second == -1 || node.first.second == -1 )
            break;
        fingerTable[next] = node;
        next++;
    }
}
```

분산 해시 테이블

Uni-dimensional	$\log N$	$\log N$
Multi-dimensional	$dN^{1/d}$	$2d$
Global Mesh	$\log_b N$	$b \log_b N$
Neighbor map	$\log_b N$	$b \log_b N + b$

BUG > TRUST

실현 가능성과 제약 조건

- 네트워크의 신뢰 보장 방안 : 알고리즘, 코드화
- 제약 조건 성립 : 알고리즘
- 해커의 비용 증대 방안 : 구조화, 알고리즘
- TCO 절감과 ROI 극대화 : 구조화
- 통신 스택 : 구조화, 코드화
- 물리적 요소 : 구조화
- 지속 가능한 잠재적 위협 : 구조화, 알고리즘, 코드화
- PSEUDO-CODE 의 부재

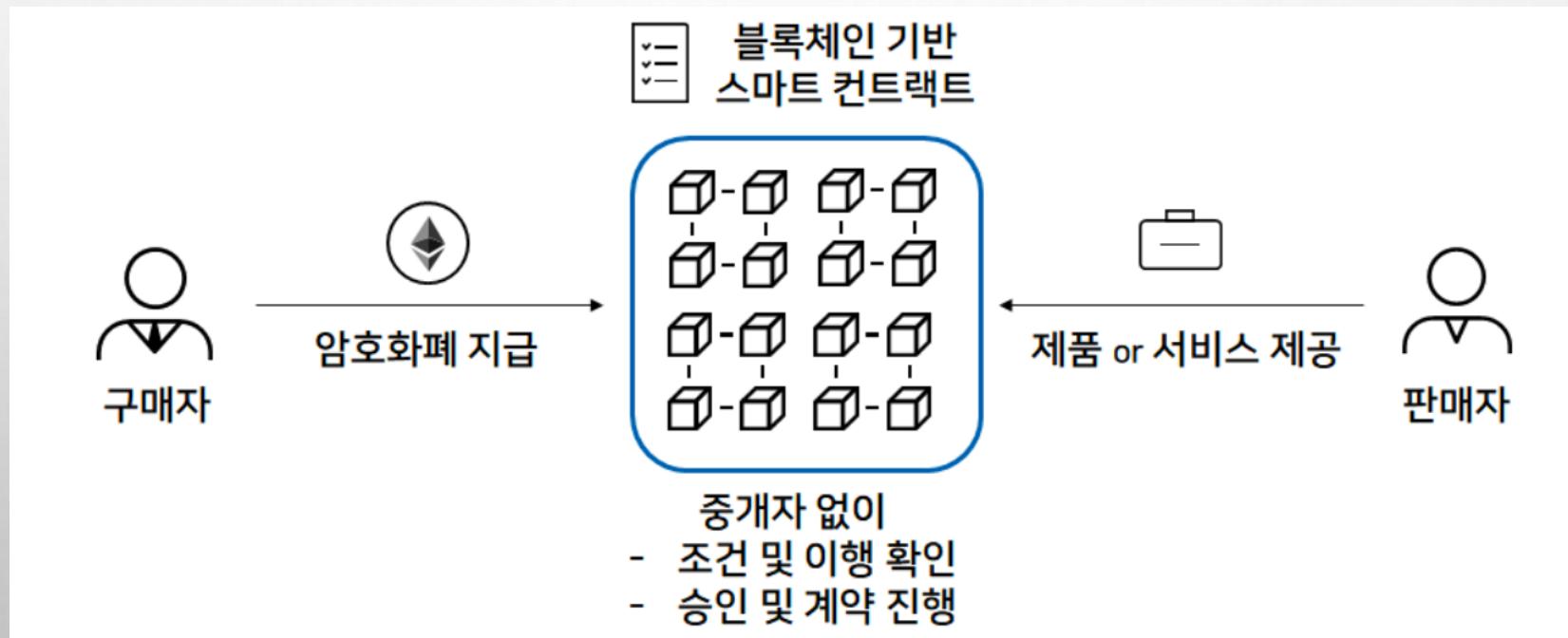
스마트 컨트랙트와 이더리움

이더리움

- 비탈릭부테린 고안
 - 스마트 컨트랙트 : 프로그래밍이 가능한 계약
 - DAPP : 분산 어플리케이션
 - 토큰 이코노미 : DAPP 의 비즈니스 모델 – 주로 인센티브 모델로 활용
- 주요 가치
 - 개방성
 - 무허가성
 - 상호운용성
 - 결합성

스마트 컨트랙트

- NICK SZABO : 계약 내용을 코드화하고 조건이 만족되면 자동으로 동작하는 개념을 최초로 체계적인 방법화로 스마트 컨트랙트라는 용어를 제안
- 블록체인을 통해 위변조 차단 및 탈중앙화로 거래가 가능해지면서 구현이 가능하게 됨



스마트 컨트랙트

- 계약을 코드로 구현 => 특정 조건 충족시 계약이 수행하는 스크립트(인터프리터)
- 투명한 거래내역 공개, 중개인 비용 및 거래 수수료 비용을 절감할때 유용
 - 누구나 컨트랙트 배포 가능
 - 컨트랙트의 소유자가 아니더라도 누구든지 검증 가능
 - 코드의 실행을 자동화
 - 위변조가 어려움
- 1994년 닉 자보에 의해 고안, 2013년 비탈릭부테린이 도입
- 제3의 중개자나 상호 신뢰 없는 거래가 가능한 구조
- NFT로 스마트 컨트랙트의 경쟁력 확인
 - ERC721 (크립토 키티)
 - ERC1155
 - 효과적인 이전 : 대규모의 토큰 전송 가능
 - 단일 계약의 여러 토큰 : 같은 계약 조건에서는 대체 가능
 - 보안 토큰 전송 : 트랜잭션 유효성이 없다면, 토큰 발급자에게 반환 (코드적 취소 기능)

토큰 이코노미

- 행동 심리학에 기반한 치료법에서 유래
- 행동 장애가 있는 사람에게 토큰 보상을 통해 행동 변화를 일으키는 방법 연구
- 메인넷, DAPP 의 보상 매커니즘이 필요
 - 자발적 참여에 대한 보상 체계로 설계에 반영

Documentation
(지속성, no latency ..)

Tokenization

Governance(DAO)

Trading

ERC20

- 비탈릭부테린이 2015년 제안
- 이더리움 기반에서 토큰을 생성하는 규약
 - 개발하지 않고도 생성이 쉽게 가능
- DAPP 토큰 규격으로 주로 많이 사용
 - 전체 발행량, 훌더, 프로젝트명(이름), 심볼, 락, 회수(소각)
- Fungible Token \Leftrightarrow NFT

토큰의 종류

- SECURITY TOKEN : 권리(증권) 투자 목적
- UTILITY TOKEN : 생태계의 서비스 및 재화 구매/사용이 목적으로 결제, 투표, 거래 수수료 등 다양한 활용

Token Rights



Digital tokens being sold in ICOs confer a combination of rights to holders

Payment

Token is the only way to make payments on the network



GNT are the only way to pay for services on the network.

Access

Token provide the ability to use the platform itself



LSK is needed to pay transaction fees on the network.

Profit or Fee

Holders get a portion of revenues or profits



Holders of TIME earn the fees from Labour-hour tokens.

Contribution

Tokens needed to play certain roles on the platform or app



1ST allow holders to determine who won gaming matches

Block Creation

Tokens determine who secures the blockchain



KMD holders select the notary nodes who secure the blockchain

Governance

Holders influence features, project direction, protocol details, or more



DGD holders determine how DigixDAO funds are spent

M2E 프로젝트, STEPn과 GST/GMT 토큰

- 앱 다운로드후 솔라나 기반의 운동화 nft 를 구매하고 뛰면 gst 토큰 채굴
- GST 사용처
 - 신발 수리
 - 신발의 능력치 향상시키는 Gem 부착에 필요한 소켓 장착
 - Gem 업그레이드
 - 신발 레벨업
- GMT 사용처
 - 10/20/30 특정 레벨 달성 필요
 - 신발 능력치 재설정
 - 희귀 등급, 상위 레벨 신발에 민팅 필요
 - Gem 을 4레벨 이상 업그레이드시

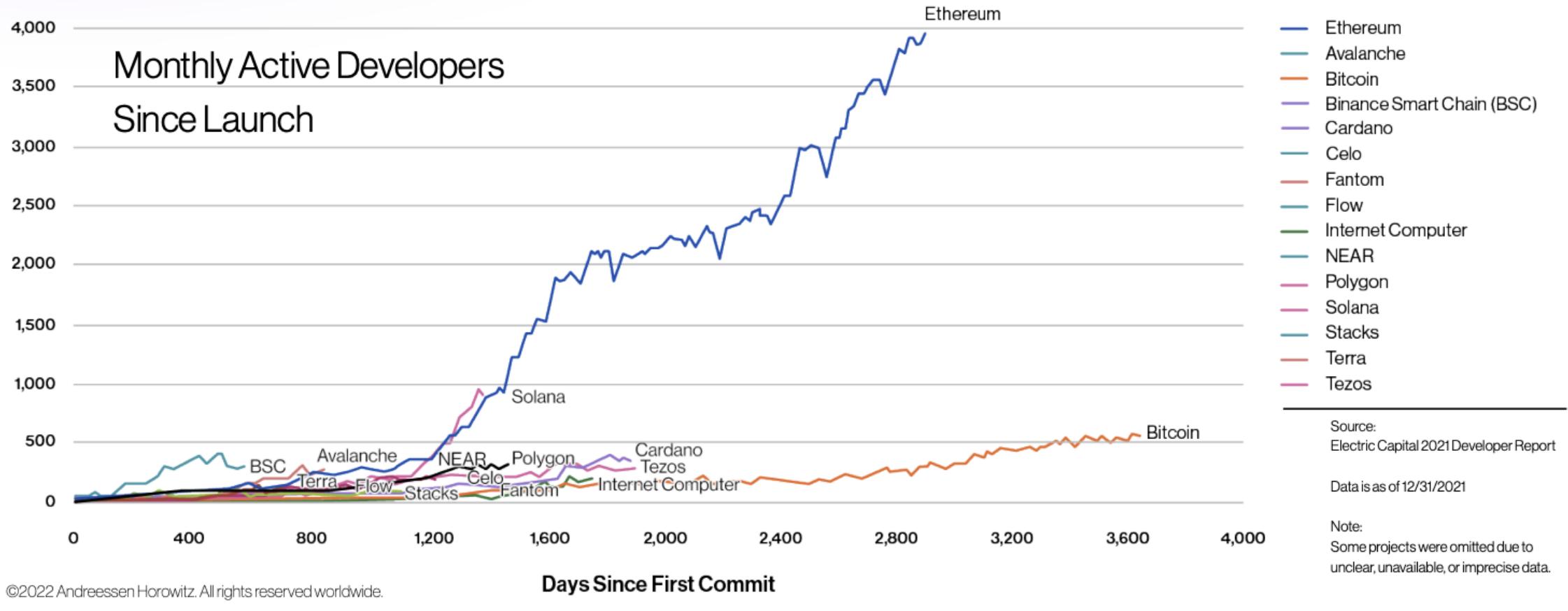
3、SNEAKER DETAIL PAGE



상호운용성

- 서로 다른 기종이나 시스템간의 서비스 공유나 정보 교환이 자유롭게 가능한 것
- 블록체인 관점
 - 서로 다른 메인넷간의 연결성
 - 서로 다른 코인간의 교환성
 - 암호화폐와 은행권과의 호환성등
- 상호운용성 구현을 위해 등장한 것이 인터체인 프로젝트
 - 오픈 프로토콜 : 서로 다른 블록체인간 통신과 데이터 교환을 위한 프로토콜 제안
 - EX)아토믹스왑
 - 멀티체인 프레임워크 : 보다 넓은 상위 네트워크를 구축하고 다양한 블록체인 연결하는 개방형 환경
 - EX) 폴카닷
- 블록체인 기술과 DAPP 이 유용성과 시장성을 인정받는데 중요한 요소

개방성 및 결합성



**Ethereum**

5.5M

Active Addresses

**Solana**

15.4M

**Polygon**

2.6M

**BNB Chain**

9.4M

**Avalanche**

609K

**Fantom**

308K

1.1M

Daily Transactions

15.3M

3.4M

5.0M

832K

743K

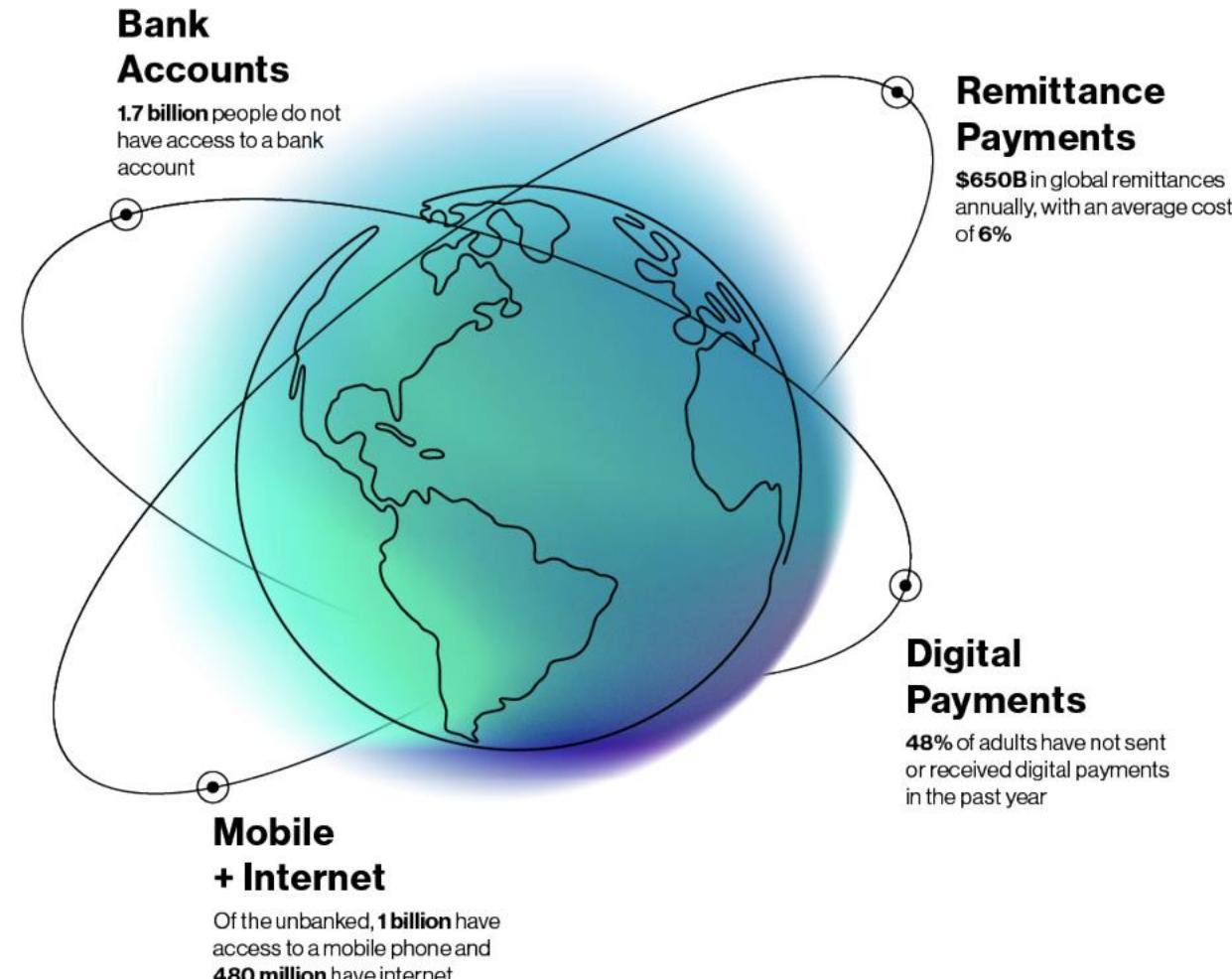
©2022 Andreessen Horowitz.
All rights reserved worldwide.

Source: Nansen; Active addresses is measured over a 30-day period as of 5/12/2022; Daily transactions
is represented by a 30-day average as of 5/12/2022.

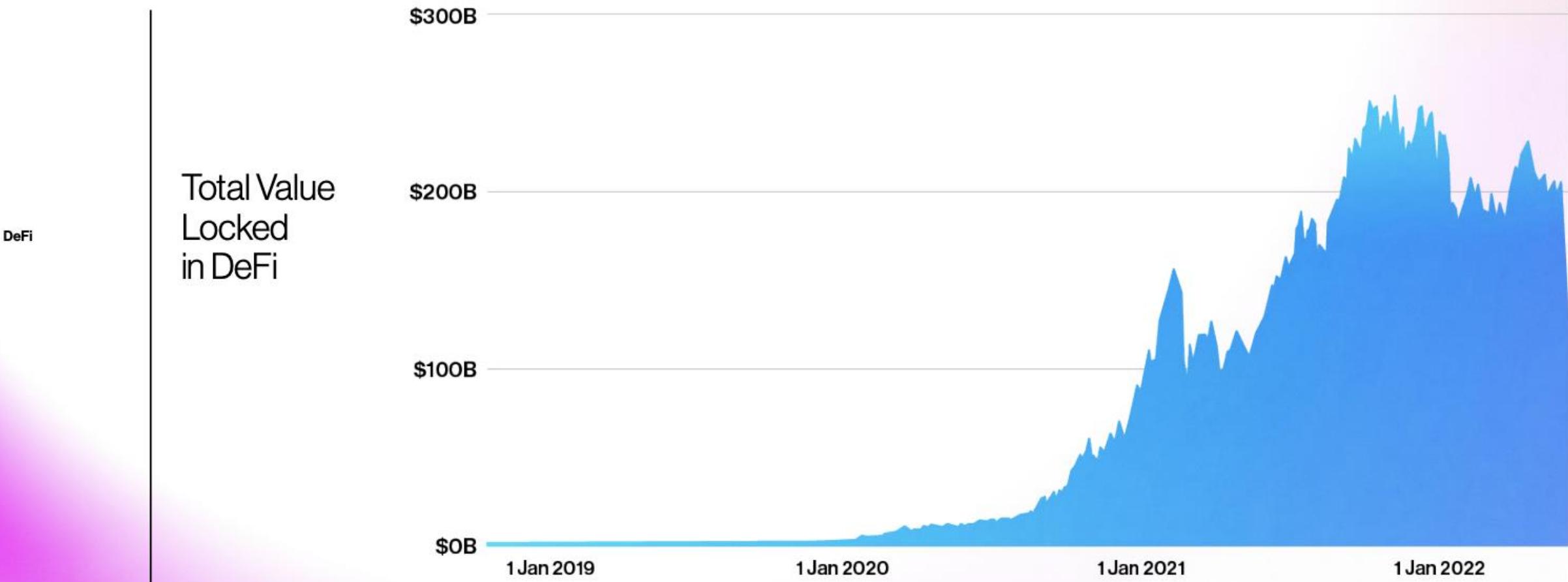
A huge part
of the world
is *underserved*
by existing
financial
institutions

Sources:
The Global Findex Database, 2017 Report;
The World Bank (2020 and 2021 data)

©2022 Andreessen Horowitz.
All rights reserved worldwide.

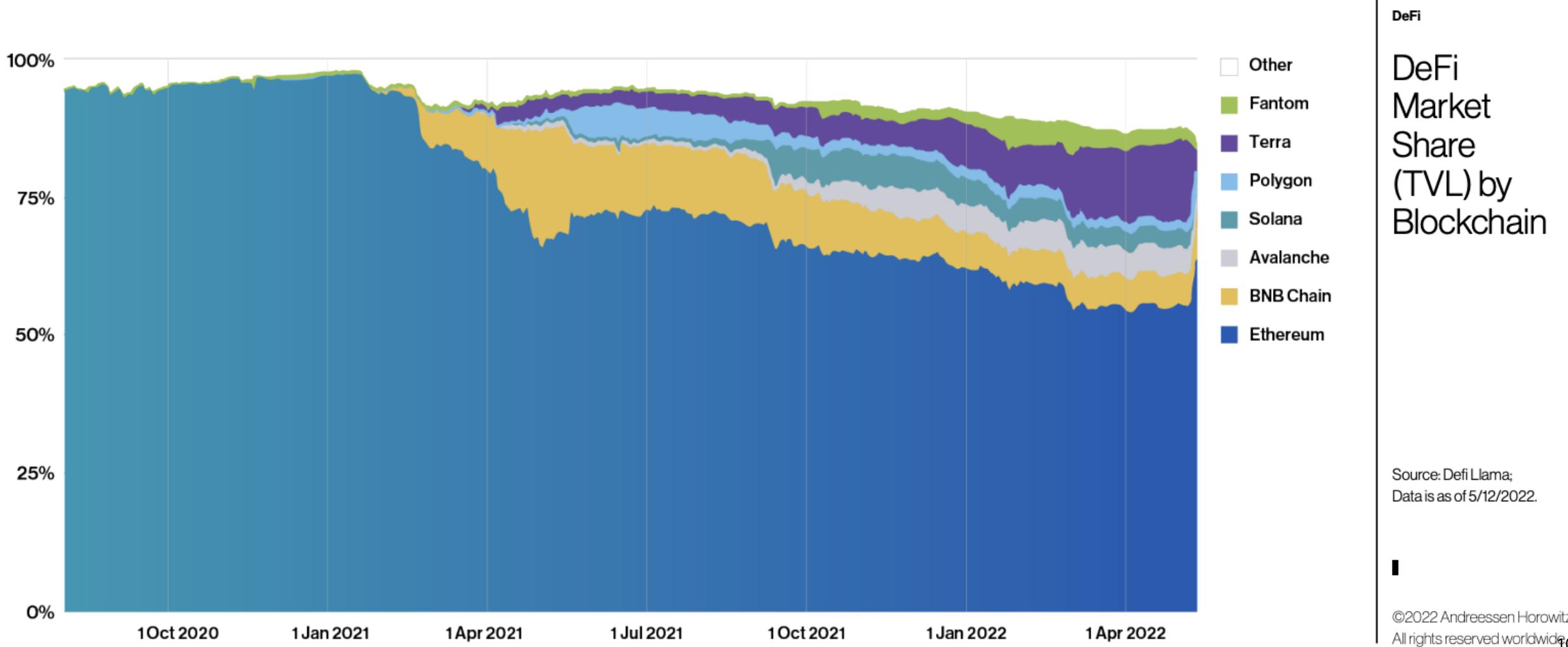


DeFi has grown from nearly zero to over \$100 billion in less than two years



Newer blockchains are trying to replicate the success of *DeFi* on Ethereum

27



Token exchange and lending protocols are the most popular DeFi use cases today

DeFi

28

Source: Token Terminal
Data is through April 2022.

Trading Volume on Decentralized Exchanges

\$400B

\$300B

\$200B

\$100B

\$0B

Ox
Curve
PancakeSwap
dYdX
Uniswap
Trader Joe
SpookySwap
SushiSwap

Jan 20
Feb 20
Mar 20
Apr 20
May 20
Jun 20
Jul 20
Aug 20
Sep 20
Oct 20
Nov 20
Dec 20
Jan 21
Feb 21
Mar 21
Apr 21
May 21
Jun 21
Jul 21
Aug 21
Sep 21
Oct 21
Nov 21
Dec 21
Jan 22
Feb 22
Mar 22
Apr 22

Borrowing Volume on Decentralized Lending Protocols

\$30B

\$20B

\$10B

\$0B

Reflexer
BENQI
Alpha Finance
Compound
Liquity
Aave
Maple Finance
MakerDAO

Jan 20
Feb 20
Mar 20
Apr 20
May 20
Jun 20
Jul 20
Aug 20
Sep 20
Oct 20
Nov 20
Dec 20
Jan 21
Feb 21
Mar 21
Apr 21
May 21
Jun 21
Jul 21
Aug 21
Sep 21
Oct 21
Nov 21
Dec 21
Jan 22
Feb 22
Mar 22
Apr 22

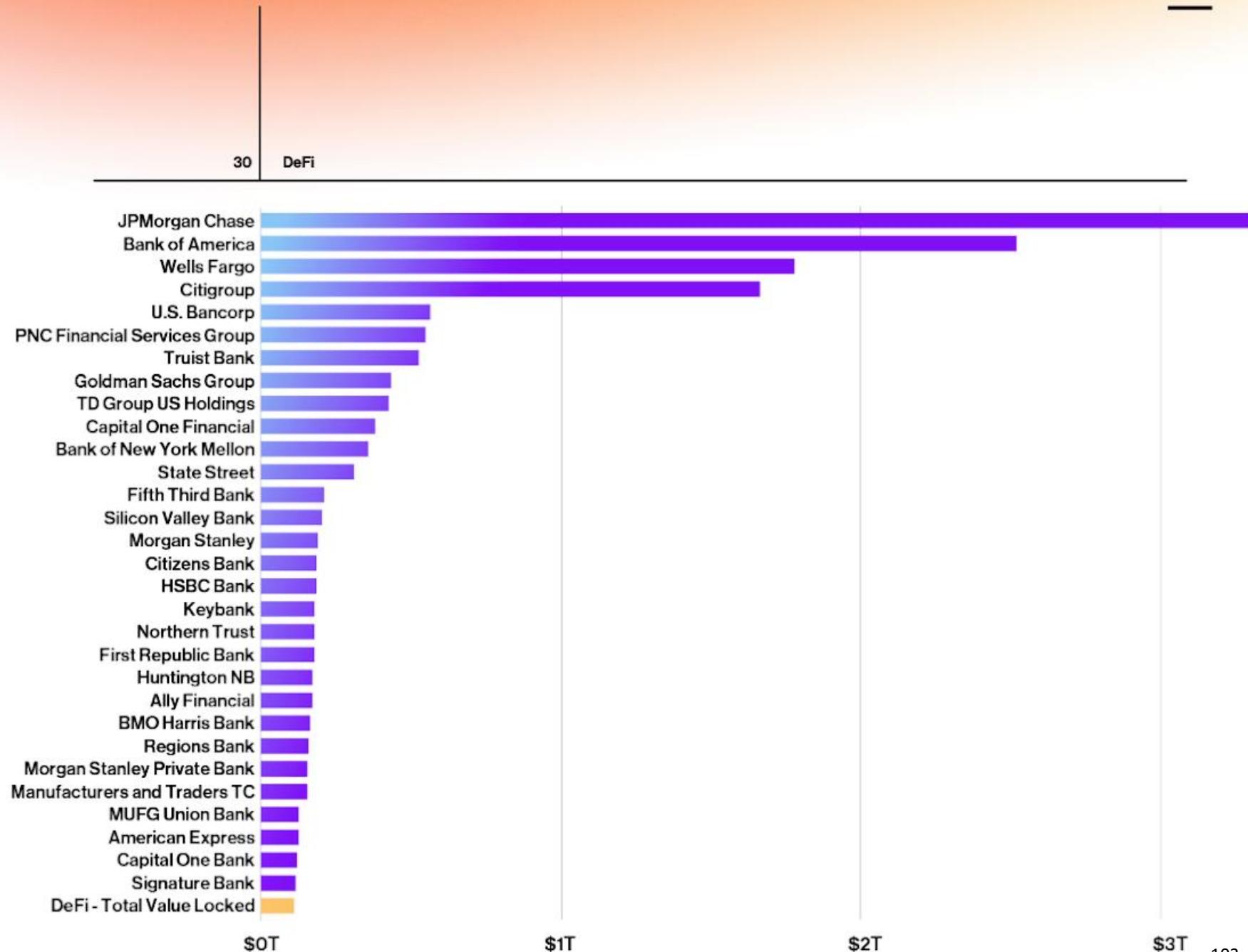
DeFi would represent the 31st largest US bank by total assets under management

Source:

Defi Llama, <https://www.federalreserve.gov/releases/lbr/current/>;

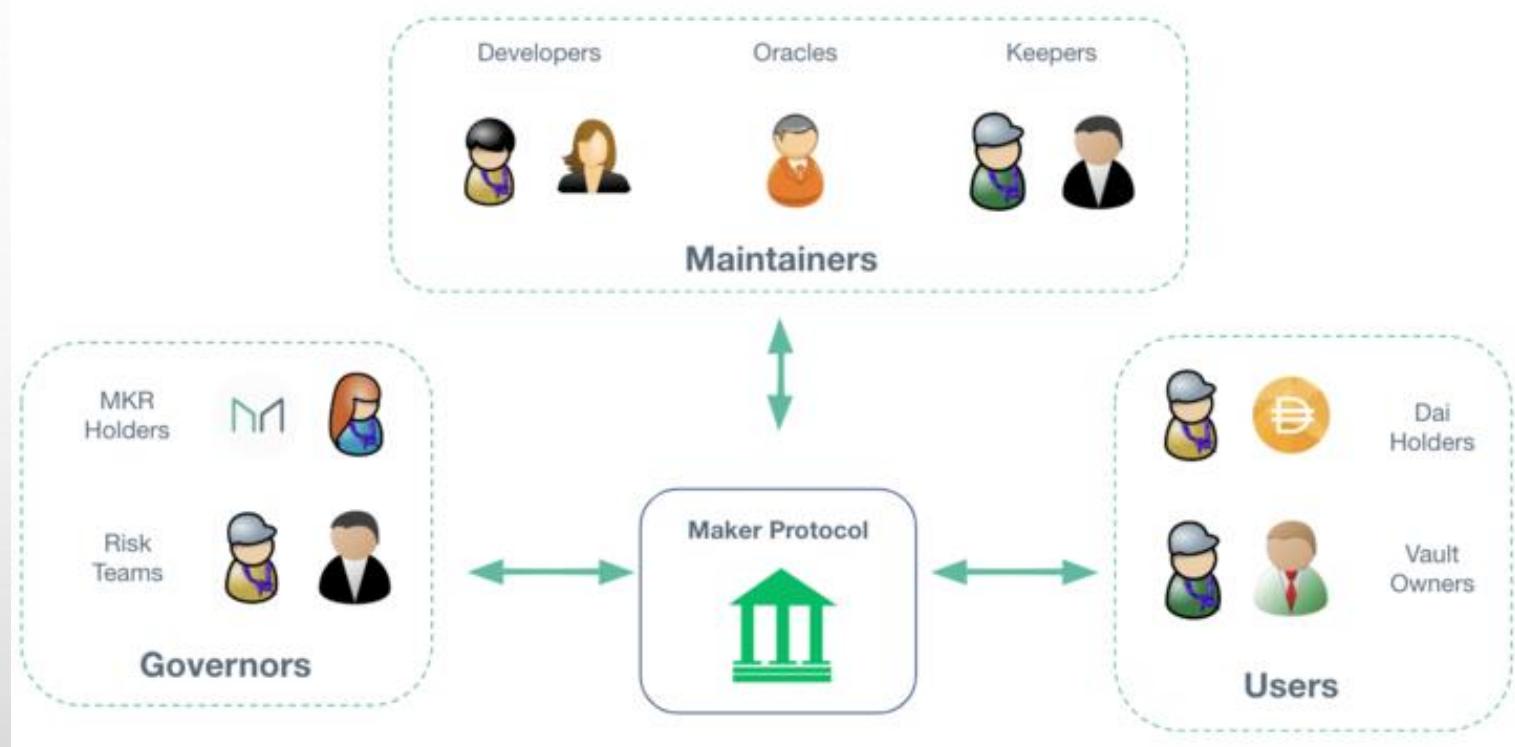
Bank AUM data is as of 12/31/2022

DeFi TVL data is as of 5/12/2022.



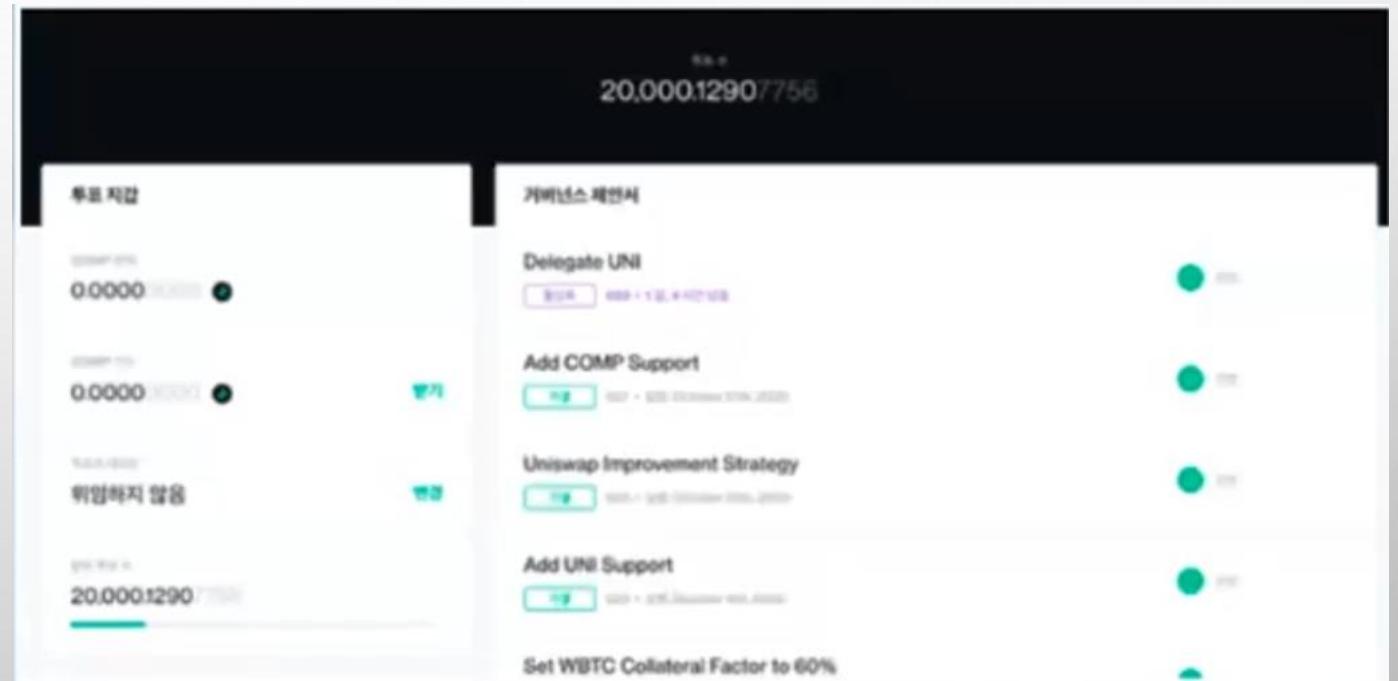
DeFi 탄생 및 현황

- 화폐, 거래소를 모두 포함하여 탈중앙화 확장
- 테더(USDT) : 스테이블 코인의 시작
 - 미국의 USD 를 1:1로 Pair
 - USD를 담보
 - 주요한 디지털 기축 통화 역할
 - USDC (Circle 사와 코인베이스간 협력)
- 다이 : 메이커다오의 스테이블 코인
 - 탈중앙화 금융의 시작
 - 이더리움을 담보
- 테라 : 알고리즘형 스테이블 코인
 - 테라 생태계 내에서 준비금인 '루나'를 발행으로 조절



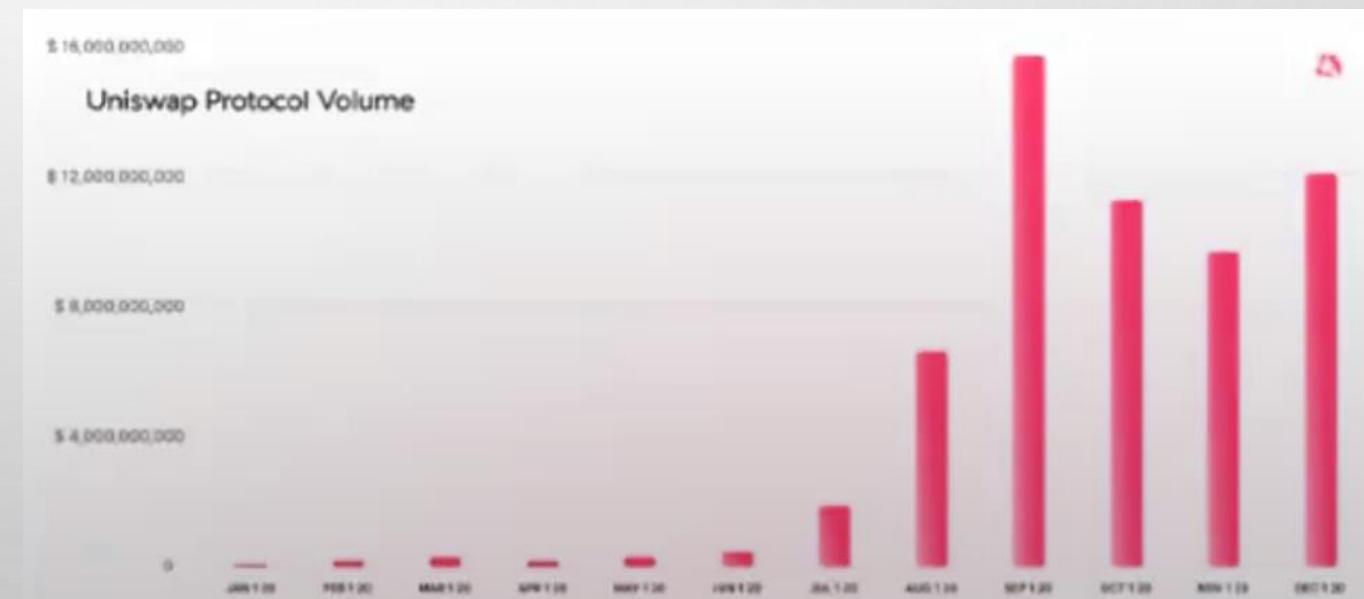
Compound

- 메이커다오 x 확장성 => 다양한 암호화폐 대출, 예치 이자로 확장
- 완전한 탈중앙화
 - 암호화폐 재고에 따라 이자율이 자동 결정
 - 회사 부도 우려 x, 횡령 x
- 거버넌스 토큰 COMP
 - 투표권한
 - 배당



Uniswap

- 다양한 대출 플랫폼의 등장
- Yearn Finance : 이자농사의 자동화
- Wrapped BTC : 비트코인 DeFi
- 2020년 5월 개발자 4명이 DEX 의 시작과 함께 개발
 - AMM 호가 개념이 없이 Pair 거래
 - Swap
 - 수수료 배당 모델로 유동성 풀 확보





Coinbase



Uniswap

FOUNDED

2012

2018

EMPLOYEES

3,730

<100

TRADING VOLUME

1.99B*

1.58B*

Source: The Block Crypto (7DMA as of 4/18/22)

DEX

- Swap 거래
- Yield Aggregator
- 이더리움 외에도 다양한 블록체인 디파이 탄생
- 브릿지 : Orbit chain, TerraBridge
- 디파이 x 토큰이코노미 => 게임, 결제, NFT
 - 단기 수익창출을 위해 거버넌스 덤프하는 행위를 방지하기 위해서 사용처 확보

DeFi, CeFi

- **DeFi(Decentralized Finance, 탈중앙화 금융)**
 - 공개형 블록체인 네트워크 상에서 Smart Contract을 기반으로 중앙 집중적인 주체나 중개자 없이 P2P 방식으로 가상자산을 거래할 수 있는 탈중앙화 금융시스템
 - 블록체인 기술을 기반으로 그중 Smart Contract 기능을 가진 이더리움이 디파이의 핵심이다.
 - 소프트웨어나 코드가 신뢰의 주체 역할을 한다. 중앙 주체의 데이터베이스가 아닌 글로벌하게 분포된 노드에 의해 시스템이 운영되기 때문에 해킹이나 시스템 셧다운의 우려가 낮으며, 누구나 네트워크 데이터를 열람 가능한 투명성이 보장된다.
 - Smart Contract를 통해 이미 짜여진 코드대로 동작하기 때문에 시간과 비용을 절감할 수 있다.
 - 대표 코인 : 테라(루나), 아발란체, 체인링크, 랩트비트코인, 유니스왑, 다이, 팬텀, 테조스, 에이브, 팬케이크스왑 등
- **C-Fi(Crypto Finance, 암호화폐를 활용한 모든 금융)**
 - 컴파운드(DeFi 기반의 담보대출 시스템)나 카이버 등은 DeFi이면서 C-Fi이다.
 - 바이낸스나 크립토닷컴 같은 거래소들이 운영하고 있는 암호화폐 스테이킹 기반 이자 지급 시스템이나 대출 서비스 등은 C-Fi이지만 중앙화 된 거래소가 컨트롤하기에 DeFi는 아니다.