Introduction to NIZK Proofs

- Principles, Algorithms and Applications

NAM Junghyun – HYPERITHM

Outline

- NIZK Proof?
  - ZK Proofs
  - Interactive ZK → NIZK
- NIZKP 알고리즘들
  - zk-SNARKs, zk-STARKs, Bulletproof
  - 간단한 비교
- NIZKP 알고리즘들의 응용사례
  - Zcash
  - zk Rollup
  - MINA

Disclaimer

- 수학적 & 이론적으로 깊게 파고들지 않음
  - 규민님 도와줘요!!!(토네이도 세미나에서 이어집니다)
- 엄밀하지 않음
- *실제로 증명된 것과 학계의 추측과 발표자의 추측*이 섞여있을 수 있음

Outline

- NIZK Proof?
  - ZK Proofs
  - Interactive ZK → NIZK
- NIZKP 알고리즘들
  - zk-SNARKs, zk-STARKs, Bulletproof
  - 간단한 비교
- NIZKP 알고리즘들의 응용사례
  - Zcash
  - zk Rollup
  - MINA

NIZK Proof?

- Non-Interactive Zero-knowledge Proof

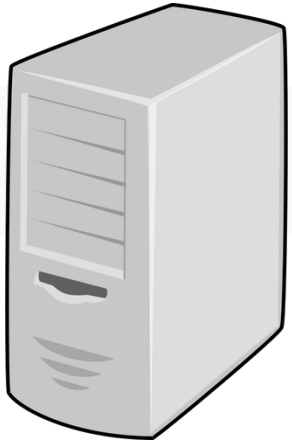
NIZK Proof?

- 단순한 로그인 서비스를 생각해봅시다

Zero-knowledge Proof?

- 단순한 로그인 서비스를 생각해봅시다

Charlie



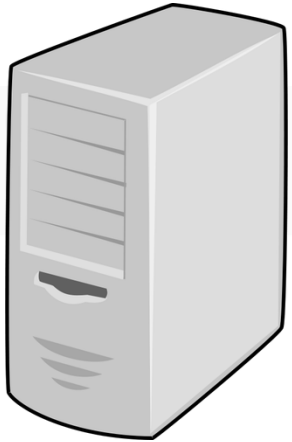
Server

ID	PW(Hashed)
Alice	…
Bob	…
Charlie	0xabcd

Zero-knowledge Proof?

- 단순한 로그인 서비스를 생각해봅시다

ID: Charlie, PW: foo



ID	PW(Hashed)
Alice	…
Bob	…
Charlie	0xabcd

Charlie

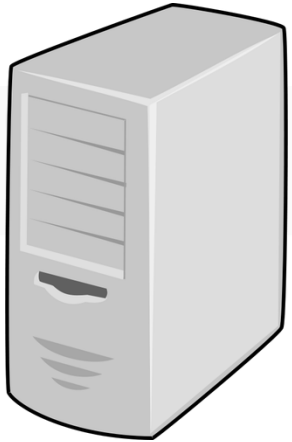
Server

Zero-knowledge Proof?

- 단순한 로그인 서비스를 생각해봅시다

ID: Charlie, PW: foo

Hi Charlie!



ID	PW(Hashed)
Alice	…
Bob	…
Charlie	0xabcd

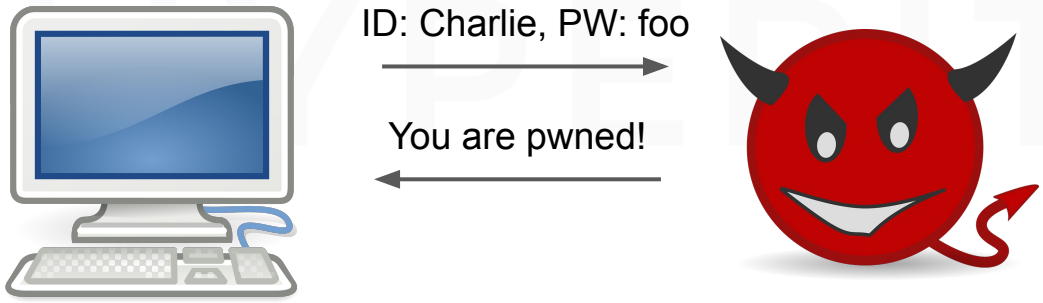
Hash(foo) = 0xabcd

Charlie

Server

Zero-knowledge Proof?

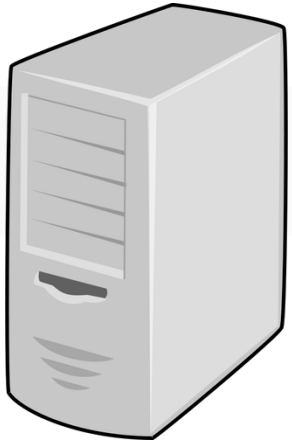
- 만약에 서버(인 줄 알았던 상대)가 공격자였다면?



Zero-knowledge Proof?

- *“비밀번호를 보여줄 수는 없지만 나는 아무튼 **Charlie**다”*

???



ID	???
Alice	…
Bob	…
Charlie	?

Charlie

Server

Zero-knowledge Proof

The Knowledge Complexity of Interactive Proof-Systems

(Extended Abstract)

Shafi Goldwasser, MIT
Silvio Micali, MIT
Charles Rackoff, University of Toronto

- 이 명제는 참임을 주장하면서 다른 정보는 *단 하나도* 노출시키지 않음(영지식)

Principles of ZKP

- 두 사람이 있어야 합니다: Prover & Verifier
- Completeness
- Soundness
- Zero-knowledgeness(ZKness)

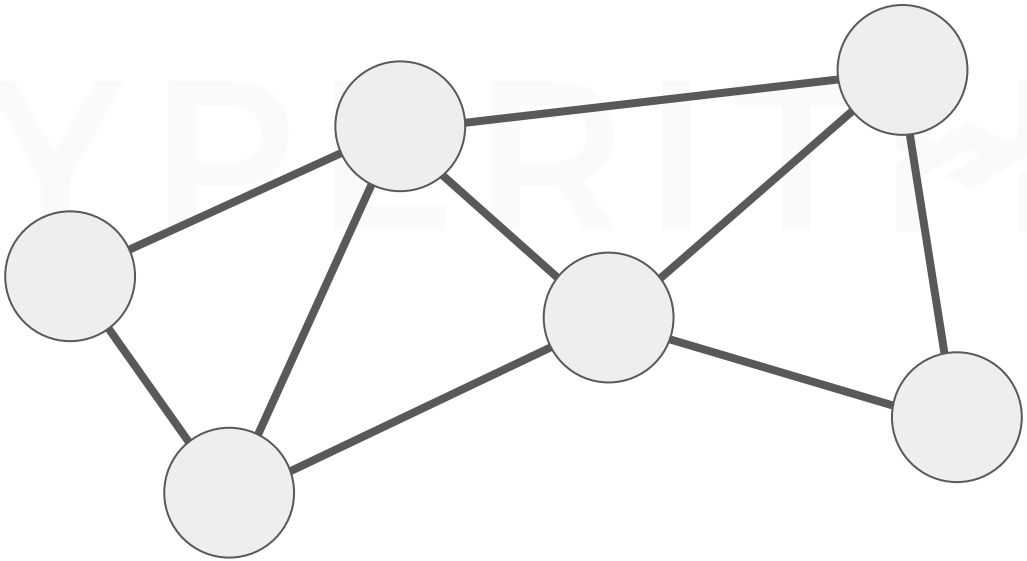
Principles of ZKP

- Completeness: 실제로 참이면, Prover가 증명할 수 있다.
- Soundness: Prover가 참임을 증명했다면, 실제로 참이다.
  - 발표자 주: 둘을 요약하면 그냥 '반례가 없다' 정도일듯?
- Zero-knowledgeness: 참/거짓 여부 외에 *어떤 정보도* 얻어갈 수 없다.

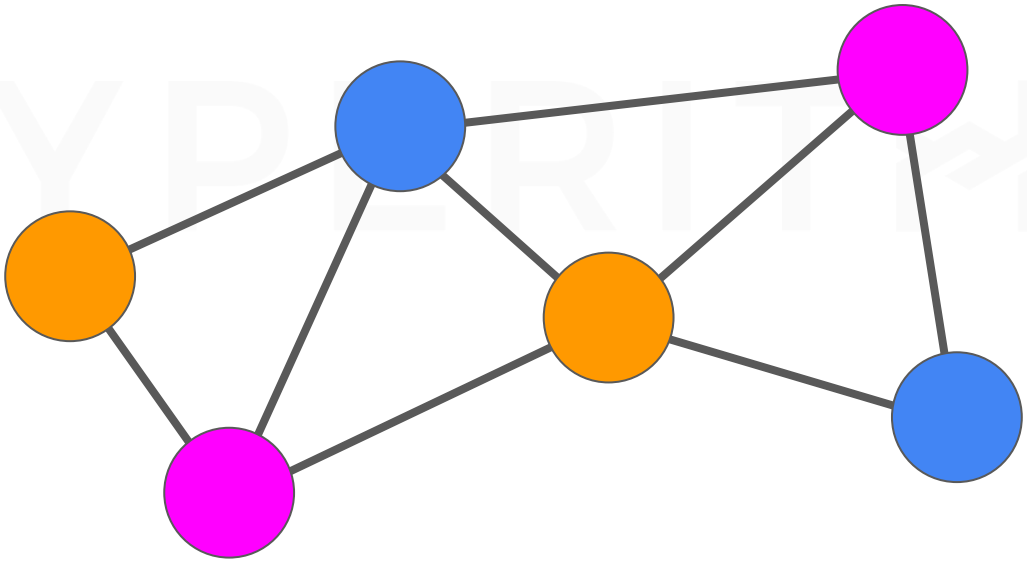
Real-world Example

- 통신 대역이 3개뿐인 중계기가 있음
- 한 중계기는 (자기가 쓰는) 대역 하나당 하나의 상대 중계기와만 통신가능
- 중계기를 1억개 설치해놨는데 주파수 배정을 까먹고 안했다

Real-world Example

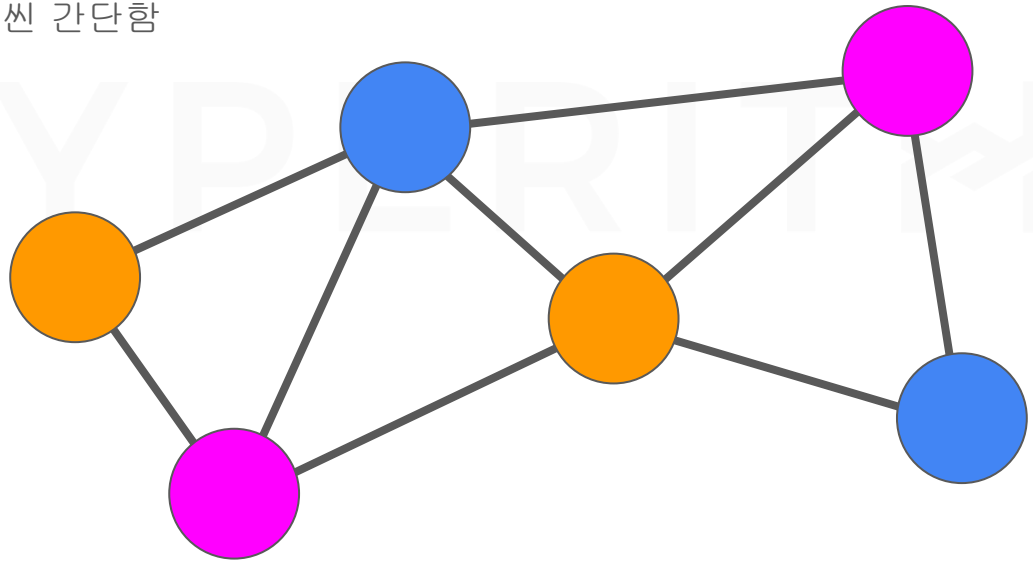


Real-world Example: 3-coloring of a Graph



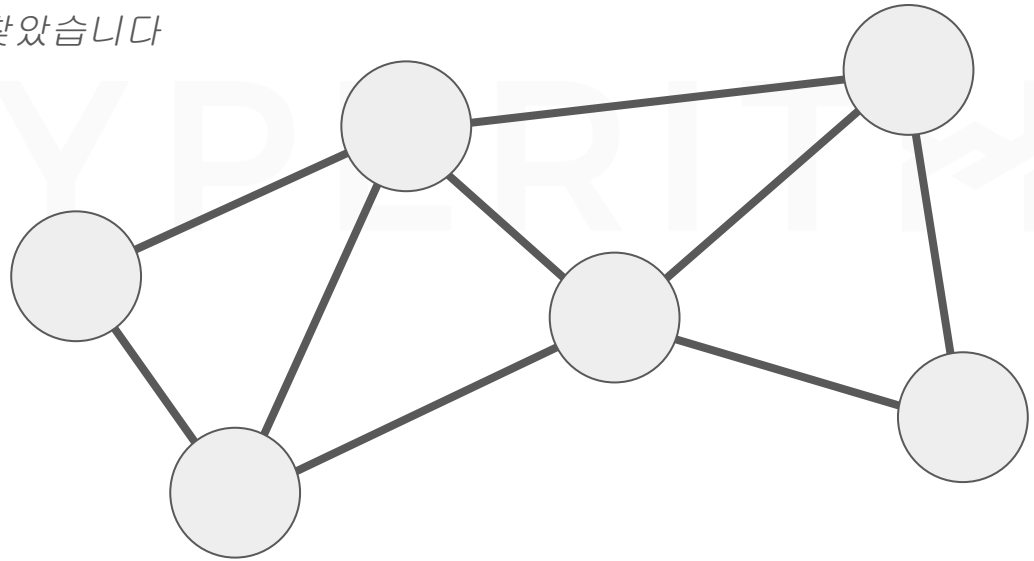
Real-world Example: 3-coloring of a Graph

- NP-Complete 문제로 알려짐
  - 푸는 건 힘듬(노드 수가 많아질수록)
  - 검증은 훨씬 간단함



Real-world Example: 3-coloring of a Graph

- 6개는 쉽지만 1억개라면? 💥
- 구글 클라우드 플랫폼에 후불로 의뢰했다
  - 고객님, *찾았습니다*



Deadlock

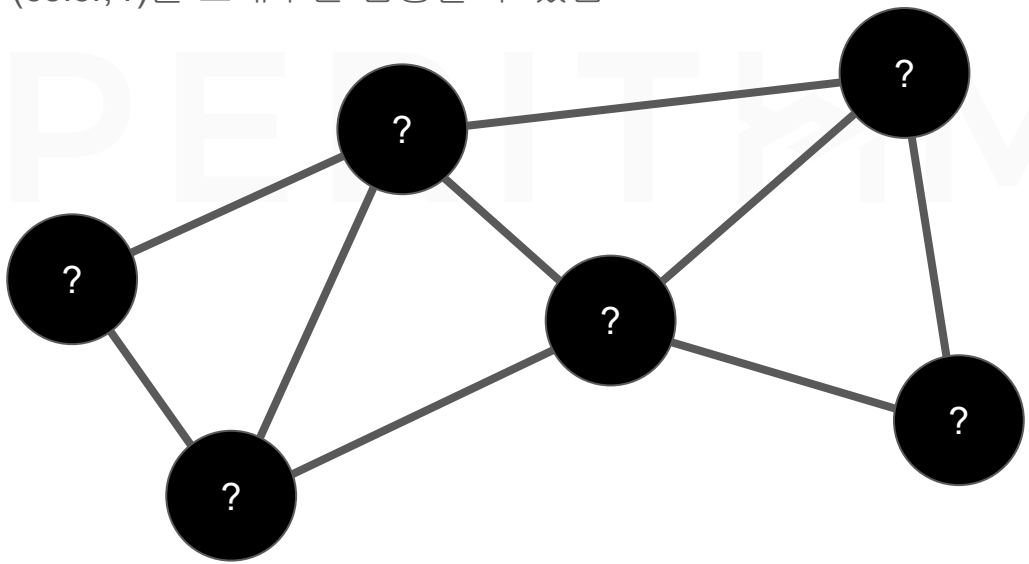
- 구글: 고객님, 3-coloring을 *찾았습니다*
- 나: 네 알려주세요
- 구글: 돈부터 입금해주세요
- 나: 뭘 믿고? 그냥 먼저 알려주세요
- 구글: 우리는 뭘 믿고???

ZKP to the Rescue

- 구글: 고객님, 3-coloring을 *찾았습니다*
- 나: 네 알려주세요
- 구글: 돈부터 입금해주세요
- 나: 뭘 믿고? 그냥 먼저 알려주세요
- 구글: 우리는 뭘 믿고???
- 나: 그럼 정답은 안 보여줘도 되니까 찾았다는 증거만 보여주세요

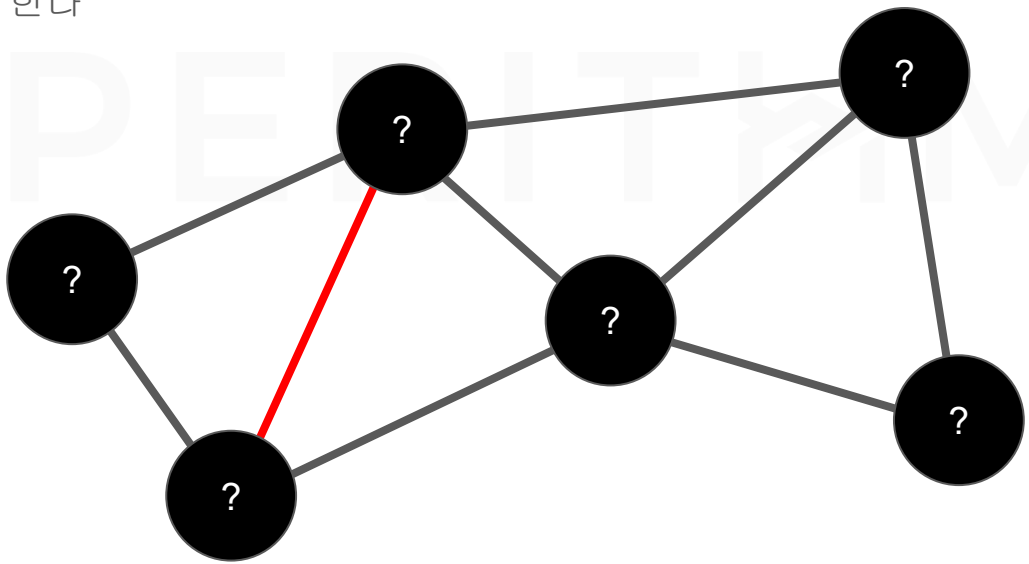
Real-world Example: 3-coloring of a Graph

- 구글은 일단 (보여주면 안되니까) 모든 노드를 *가린 채로* 정답을 보내준다
  - Programmatic Implementation: H(color, r) where r is a random nonce
  - 나중에 공개할 때는 (color, r)을 보내주면 검증할 수 있음



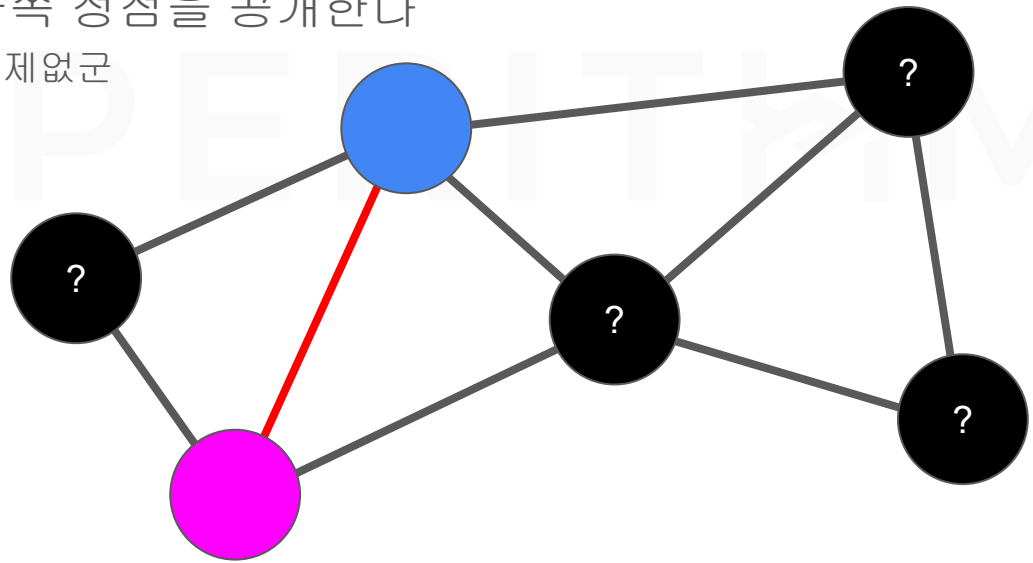
Real-world Example: 3-coloring of a Graph

- 구글은 일단 (보여주면 안되니까) 모든 노드를 *가린 채로* 정답을 보내준다
- 나는 간선을 하나 무작위로 선택한다
  - 이걸 challenge라고 한다



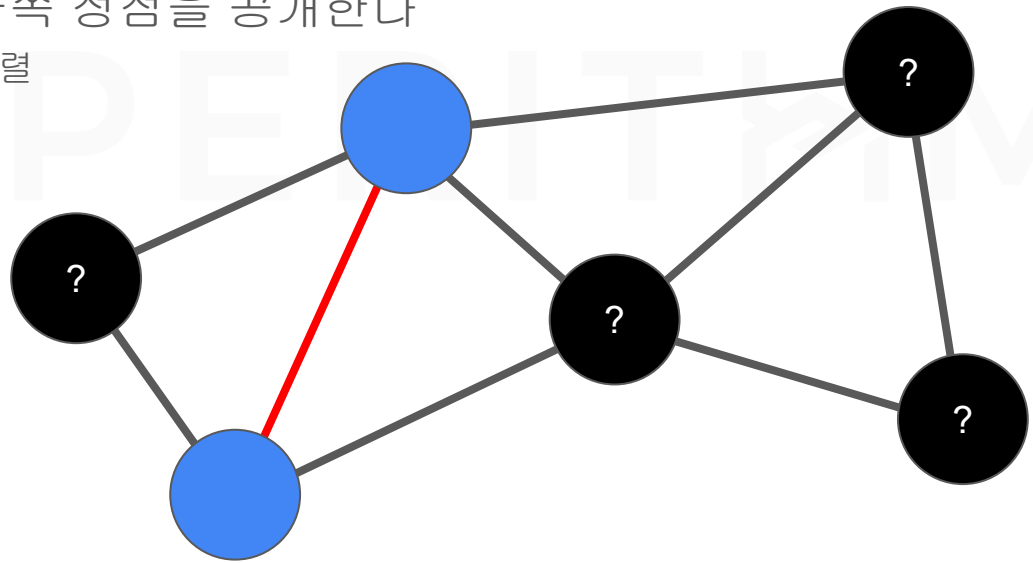
Real-world Example: 3-coloring of a Graph

- 구글은 일단 (보여주면 안되니까) 모든 노드를 *가린 채로* 정답을 보내준다
- 나는 간선을 하나 무작위로 선택한다
- 구글은 그 간선의 양쪽 정점을 공개한다
  - 음, (아직까지는) 문제없군



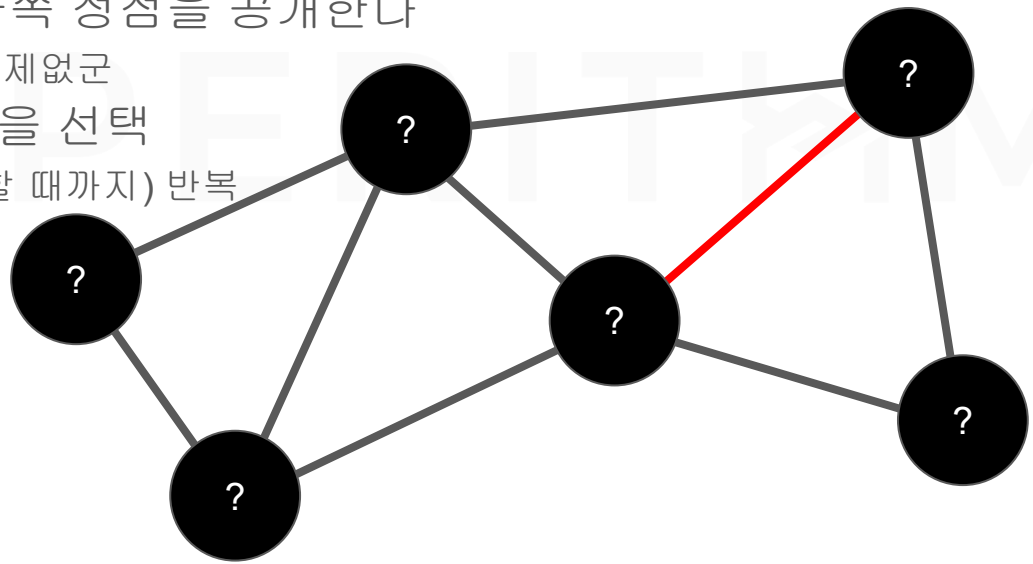
Real-world Example: 3-coloring of a Graph

- 구글은 일단 (보여주면 안되니까) 모든 노드를 *가린 채로* 정답을 보내준다
- 나는 간선을 하나 무작위로 선택한다
- 구글은 그 간선의 양쪽 정점을 공개한다
  - 어 들켰네? 협상 결렬



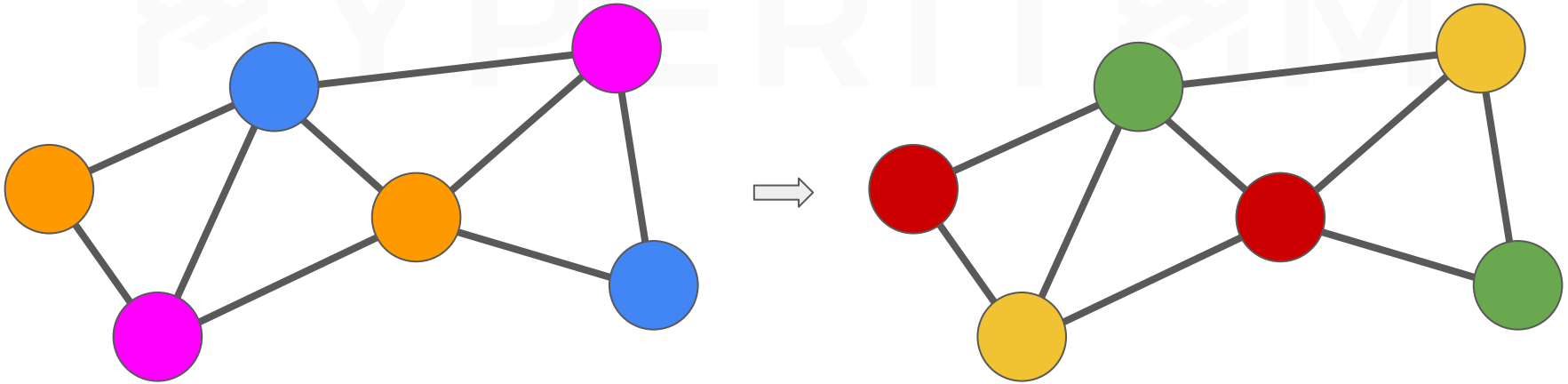
Real-world Example: 3-coloring of a Graph

- 구글은 일단 (보여주면 안되니까) 모든 노드를 *가린 채로* 정답을 보내준다
- 나는 간선을 하나 무작위로 선택한다
- 구글은 그 간선의 양쪽 정점을 공개한다
  - 음, (아직까지는) 문제없군
- 다시 무작위로 간선을 선택
  - (내가 충분히 납득할 때까지) 반복



잠깐만요

- 이러면 언젠가는 구글이 모든 노드를 공개하지 않나요?
- 그래서 매번 공개할 때마다 색을 *섞어서* (여전히 가린 상태로) 다시 보냅니다



Analogy

- 구글: Prover
- 나: Verifier
- Secret: 주어진 그래프의 3-coloring
  - 구글 외의 사람에게 **절대** 노출되면 안 되는 정보
- 증명하고자 하는 명제: “구글”은 이 그래프의 3-coloring을 알고 있다

Let’s Show Three Principles of ZKP

- Completeness: 구글이 실제로 coloring을 찾았다면, 증명할 수 있다
- Soundness: 구글이 증명을 잘 해내면, 나는 구글을 믿을 수 있다
- ZKness: 나는 구글의 증명으로부터 3-coloring에 대해 *어떠한* 단서도 얻을 수 없다

Let’s Show Three Principles of ZKP

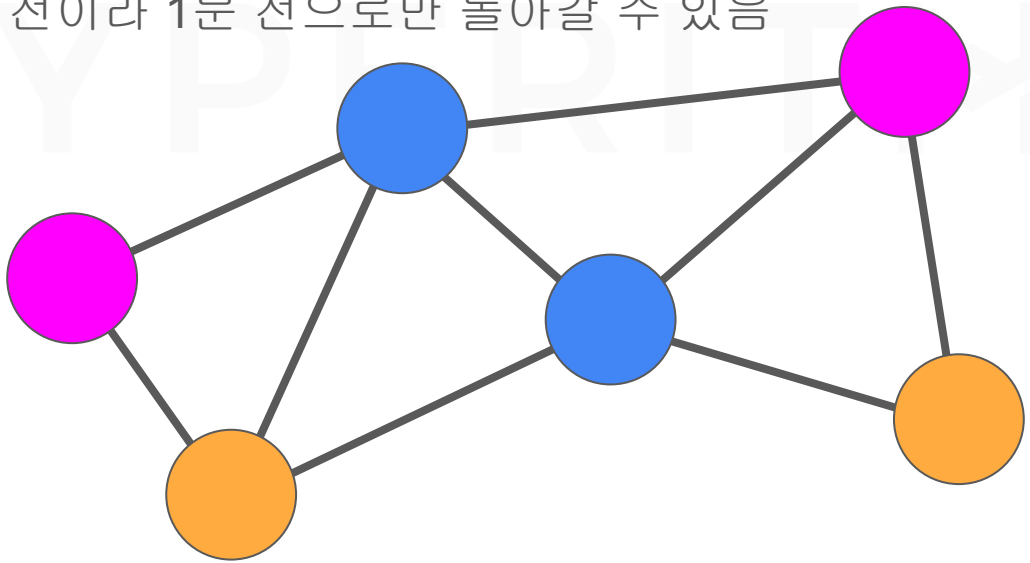
- Completeness: 아까 설명한 대로 따라가면 됨. 자명.

Let’s Show Three Principles of ZKP

- Completeness: 아까 설명한 대로 따라가면 됨. 자명.
- Soundness: Edge의 수를 E라 하자.
  - E개의 명제가 거짓이라면 최소 하나의 간선은 *들통나는 간선*
  - (간선 무작위 선택을) 한번 할때마다 마다 구글은 $(E-1)/E$ 확률로 들통나게 됨
  - 납득할 수준으로 반복하면 (E^2) soundness가 깨질 확률은 *무시할 수 있을 만큼* 낮아짐
- ZKness?

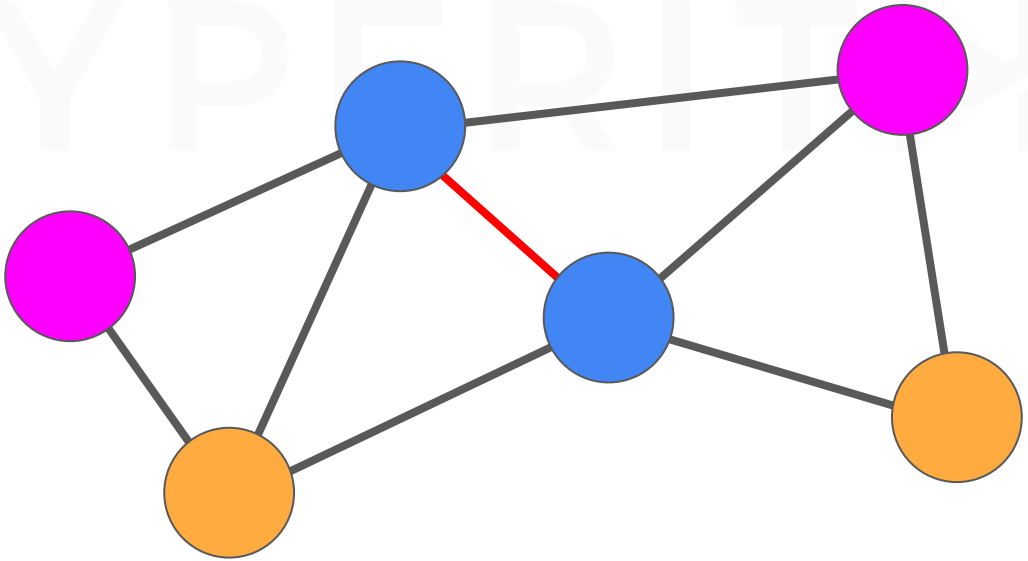
Simulator Argument

- 구글은 실제로 coloring을 모르지만 알고 있다고 주장했다(고 가정하자)
  - 대충 랜덤하게 색 배정해서 채워놓은 상태
- 이렇게 거짓말한 이유는 구글이 타임머신을 갖고 있기 때문
- 근데 베타 버전이라 1분 전으로만 돌아갈 수 있음



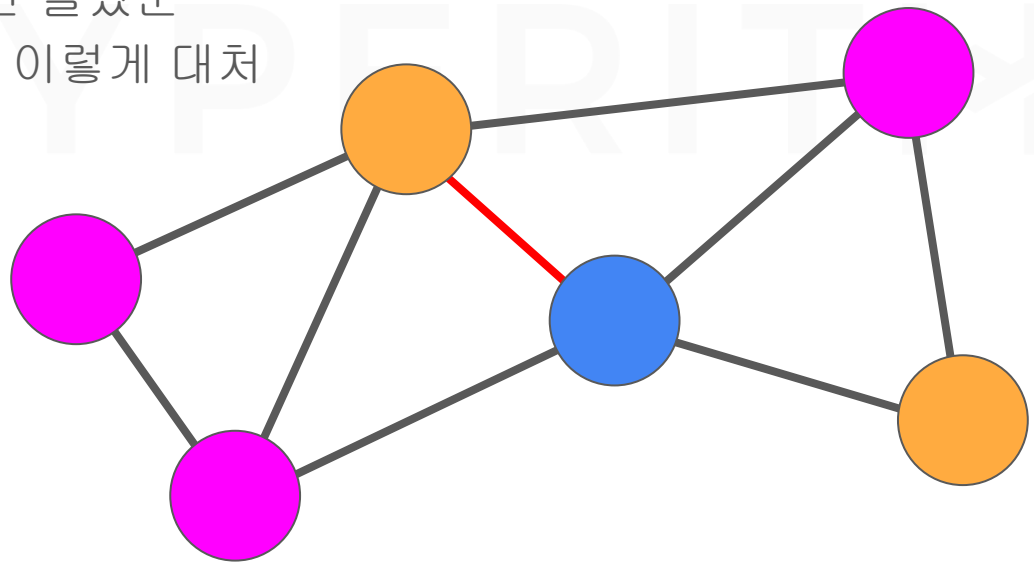
Simulator Argument

- 제발 이 간선만 걸리지 말아라...



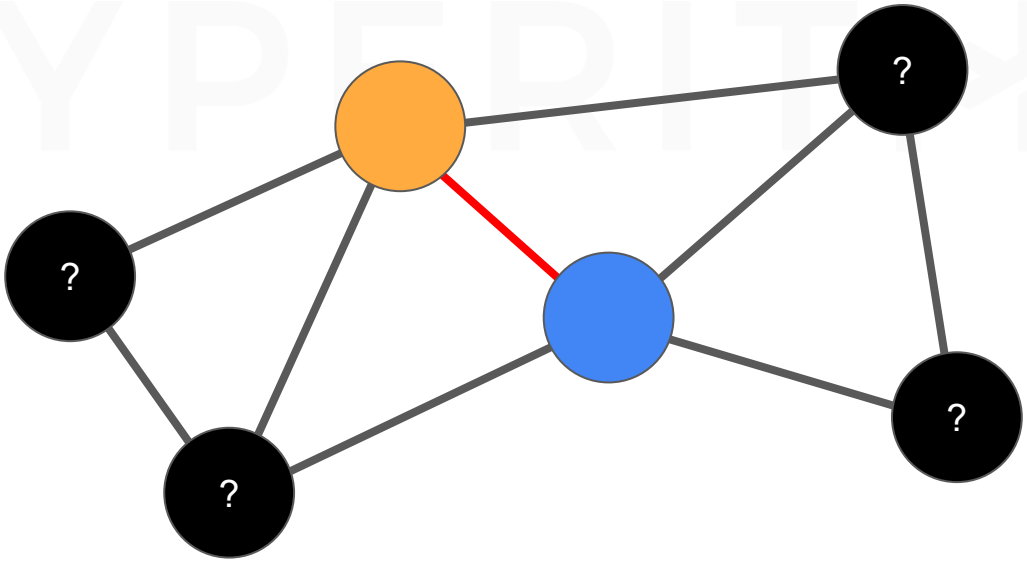
Simulator Argument

- 어 진짜 선택해버렸네?
  - 타임머신 작동(1분 전으로 돌아감)
  - 랜덤하게 다시 색을 배치
- 흠, 이번엔 안 걸렸군
- 걸릴 때마다 이렇게 대처



Simulator Argument

- 내 입장에서는?
- 간선을 선택하는 족족 색이 다름
- 설득 당해버렸다(구글이 coloring을 진짜 찾았구나)



ZKness Proved

- 내 입장에서는?
- 간선을 선택하는 족족 색이 다름
- 설득 당해버렸다(구글이 coloring을 진짜 찾았구나)
- 이 경우에 구글은 coloring에 관한 *어떠한 정보도 갖고 있지 않음*
- 하지만 나는 구글이 성실한지, 타임머신으로 cheating했는지 구분 불가
  - 두 경우에 얻을 수 있는 정보량이 ***동일하다!!!***
  - 타임머신으로 cheating하면 정보량이 0이다
  - 그러므로 구글이 성실하게 증명해도 ***정보량이 0이다!!!*** = ZKness

ZKness Proved

- 내 입장에서는?
- 간선을 선택하는 족족 색이 다름
- 설득 당해버렸다(구글이 coloring을 진짜 찾았구나)
- 이 경우에 구글은 coloring에 관한 *어떠한 정보도 갖고 있지 않음*
- 하지만 나는 구글이 성실한지, 타임머신으로 cheating했는지 구분 불가
  - 두 경우에 얻을 수 있는 정보량이 ***동일하다!!!***
  - 타임머신으로 cheating하면 정보량이 0이다
  - 그러므로 구글이 성실하게 증명해도 ***정보량이 0이다!!!*** = ZKness
- 더 나아가서, 제 3자(Apple?)가 증명 과정을 누가 녹화해놓는 것은 의미가 없음
  - *내가* verify할 때만 증명이 의미를 가짐

Is It Contradictory?

- Soundness의 전제: *타임머신은 존재하지 않는다*
  - 타임머신이 존재하면 확률이 의미가 없어짐 → 납득할 수 없음
  - 실제로도 타임머신이 있을 리가 없잖아요
- ZKness에서의 트릭: *타임머신을 쓰자*
  - ???

Time Machine Exists (in terms of…)

- 우리는 이걸 가상머신이라 부르기로 했어요

vmware®

Time Machine Exists (in terms of…)

- Prover가 *전지전능하다면* 가능하다
  - 이 Prover를 Simulator라고 하자
- *이론상* 시간을 되돌릴 수는 있으므로 ZKness OK
- 현실에서는 불가능하므로 Soundness OK

vmware®

Interactive ZK to Noninteractive ZK

- 블록체인에서 이렇게 수십수백번의 interaction을 하면 코스트가 너무 큼
  - 트랜잭션 한번 보내는데 5분동안 연결 유지해야하는 체인이 있다?
  - 이걸 전세계 수십억명이 동시에?
- Noninteractive하게 바꿉시다
  - *단 한번만* 증명을 보내면 끝

Fiat-Shamir Heuristic

- 대화가 없으므로 대신 챌린지를 만들어 줄 Oracle이 필요함
- 결과를 예측할 수는 없음(Random)
- 하지만 일단 챌린지가 나왔으면 재현 가능해야 함(Verifiable)

Fiat-Shamir Heuristic

- 대화가 없으므로 대신 챌린지를 만들어 줄 Oracle이 필요함
- 결과를 예측할 수는 없음(Random)
- 하지만 일단 챌린지가 나왔으면 재현 가능해야 함(Verifiable)
- 어디서 많이 봤는데?
  - Cryptographic Hash(e.g. SHA-256)
  - 추상적으로는 Random Oracle이라 합니다

NIZK of 3-coloring of a Graph

- 주의: 발표자가 임의로 구성한 내용임
  - 원문은 Schnorr identification protocol을 NIZK로 바꿈
  - 순환군, 차수, generator가 뭔데??? 😇
- 그래서 그냥 직접 3-coloring 문제를 NIZK로 바꿔봤습니다
  - 엄밀하지 않음. 틀려도 발표자는 모릅니다
  - 맛만 보는 느낌으로…

- 사전 셋업
  - 어떤 소수 p를 고름. g는 소수 차수 q를 가지는 순환군의 generator임.
  - Alice는 임의의 $a \in \{1, 2, \ldots, q\}$를 골라 secret key로 삼음
  - Alice의 public key: g^a
- Alice는 임의의 k를 골라 Bob에게 $h = g^k \mod p$를 보냄
- Bob은 임의의 $c \in \{1, 2, \ldots, q\}$를 골라 Alice에게 보냄(챌린지)
- Alice는 $s = ac + k \mod q$를 Bob에게 보냄

NIZK of 3-coloring of a Graph

- Cryptographic hash function H에 대해 챌린지를 다음처럼 계산
- $c = H(MaskedNode1, MaskedNode2, \ldots) \bmod CountOfEdges$
- Verifier에게 보낼 정보: $(MaskedNode1, MaskedNode2, \ldots)$, c, $NodeX$, $NodeY$, $NonceX$, $NonceY$
  - $NodeX$, $NodeY$는 c에 연결된 두 노드
  - 일단 보내지 말고 저장
- 이제 색을 섞고 노드를 다시 가린 후(해시를 다시 계산한 후) 챌린지를 다시 생성
  - 아까처럼 계속 반복
- 보낼 정보를 (만족할만큼=Verifier가 100%에 가깝게 확신할 만큼) 충분히 쌓았으면 Verifier에게 보냄

NIZK of 3-coloring of a Graph

- Cryptographic hash function H에 대해 챌린지를 다음처럼 계산
- $c = H(MaskedNode1, MaskedNode2, \ldots) \bmod CountOfEdges$
- Verifier에게 보낼 정보: $(MaskedNode1, MaskedNode2, \ldots)$, c, $NodeX$, $NodeY$, $NonceX$, $NonceY$
  - $NodeX$, $NodeY$는 c에 연결된 두 노드
  - 일단 보내지 말고 저장
- 이제 색을 섞고 노드를 다시 가린 후(해시를 다시 계산한 후) 챌린지를 다시 생성
  - 아까처럼 계속 반복
- 보낼 정보를 (만족할만큼=Verifier가 100%에 가깝게 확신할 만큼) 충분히 쌓았으면 Verifier에게 보냄

Wrap-up

- NIZK Proof?
  - ZK Proofs
  - Interactive ZK → NIZK
- NIZKP 알고리즘들
  - zk-SNARKs, zk-STARKs, Bulletproof
  - 간단한 비교
- NIZKP 알고리즘들의 응용사례
  - Zcash
  - zk Rollup
  - MINA

Outline

- NIZK Proof?
  - ZK Proofs
  - Interactive ZK → NIZK
- NIZKP 알고리즘들
  - zk-SNARKs, zk-STARKs, Bulletproof
  - 간단한 비교
- NIZKP 알고리즘들의 응용사례
  - Zcash
  - zk Rollup
  - MINA

Bulletproof

- 모네로가 씀
- DLP(Discrete Logarithm Problem; 이산로그문제) 기반
  - 타원곡선 집합에서 g, g^a만 가지고 a를 계산하는 게 아주 어렵다고 알려짐
  - 아까 슈노르 서명이 DLP 기반 프로토콜
- 모네로 말고는 안씀
  - 이유: 구림

- 사전 셋업
  - 어떤 소수 p를 고름. g는 소수 차수 q를 가지는 순환군의 generator임.
  - Alice는 임의의 $a \in \{1, 2, \ldots, q\}$를 골라 secret key로 삼음
  - Alice의 public key: g^a
- Alice는 임의의 k를 골라 Bob에게 $h = g^k \mod p$를 보냄
- Bob은 임의의 $c \in \{1, 2, \ldots, q\}$를 골라 Alice에게 보냄(챌린지)
- Alice는 $s = ac + k \mod q$를 Bob에게 보냄

zk-SNARKs

- NIZK 프로토콜 중에 가장 유명함
  - zk: Zero-knowledge
  - S: Succinct
  - N: Noninteractive
  - ARKs: Argument of Knowledge
- Zcash, zkSync, MINA, etc.
- Prover-Verifier 간에 *Common Reference String*을 미리 셋업해야 함
  - 일종의 타원곡선 키 페어(Pubkey-SecKey)
  - Secret Key를 *무조건 버려야 함:* Secret Key를 쓰면 블록체인 전체의 보안이 무력화됨
  - 일종의 중앙화 리스크(파라미터 생성자를 믿어야 함)

Parameter Generation of Zcash

https://z.cash/technology/paramgen/

- 키페어를 MPC로 구성
  - 이렇게 하면 **모든** 참가자가 시크릿키(의 일부)를 제거하지 않았어야 가정이 깨짐
- 6명의 위원들이 각각…
  - 동네 컴퓨터 가게에서 새 컴퓨터를 **두 대** 사서

Parameter Generation of Zcash

https://z.cash/technology/paramgen/

- 키페어를 MPC로 구성
  - 이렇게 하면 **모든** 참가자가 시크릿키(의 일부)를 제거하지 않았어야 가정이 깨짐
- 6명의 위원들이 각각…
  - 동네 컴퓨터 가게에서 새 컴퓨터를 **두 대** 사서
  - 모든 네트워크 장비(Wi-Fi, Bluetooth, Ethernet)를 제거한 다음
  - 키 생성 전용 자체 OS를 부팅해서
  - 각각 **네트워크 노드**와 **계산 노드**로 사용. 두 컴퓨터는 물리적으로 완전히 분리(air-gapped)
    - 통신은 append-only DVD만을 써서 함
- Note: 이건 2016년 sprout 네트워크 기준이고 2018년에 sapling 셋업할때는 Powers of Tau를 쓴듯

Parameter Generation of Zcash

https://z.cash/technology/paramgen/

- 키페어를 MPC로 구성
  - 이렇게 하면 **모든** 참가자가 시크릿키(의 일부)를 제거하지 않았어야 가정이 깨짐
- 6명의 위원들이 각각…
  - 동네 컴퓨터 가게에서 새 컴퓨터를 **두 대** 사서
  - 모든 네트워크 장비(Wi-Fi, Bluetooth, Ethernet)를 제거한 다음
  - 키 생성 전용 자체 OS를 부팅해서
  - 각각 **네트워크 노드**와 **계산 노드**로 사용. 두 컴퓨터는 물리적으로 완전히 분리(air-gapped)
    - 통신은 append-only DVD만을 써서 함
  - 키 계산을 순서대로 한바퀴 돌리고
  - 서버에 결과를 보내서 FFT 계산을 1시간 동안 하고
  - 다시 한바퀴 돌려서 키 계산하고 한번 더 계산함. 컴퓨터를 끔으로서 RAM에 있던 시크릿키가 날아감
    - 이 시크릿키가 *toxic waste*

zk-STARKs

- 비교적 신생
  - zk: Zero-knowledge
  - S: Succinct
  - T: Transparent
  - ARKs: Argument of Knowledge
- Common Reference String 불필요
- zk-SNARK의 타원곡선 DSA 대신 해시함수를 씀
  - 양자컴퓨터에 비교적 강하다고 함

Comparison of NIZK Algorithms

Comparison of the most popular zkp systems			
	SNARKs	STARKs	Bulletproofs
Algorithmic complexity: prover	O(N * log(N))	O(N * poly-log(N))	O(N * log(N))
Algorithmic complexity: verifier	~O(1)	O(poly-log(N))	O(N)
Communication complexity (proof size)	~O(1)	O(poly-log(N))	O(log(N))
- size estimate for 1 TX	Tx: 200 bytes, Key: 50 MB	45 kB	1.5 kb
- size estimate for 10.000 TX	Tx: 200 bytes, Key: 500 GB	135 kb	2.5 kb
Ethereum/EVM verification gas cost	~600k (Groth16)	~2.5M (estimate, no impl.)	N/A
Trusted setup required?	YES 😒	NO 😄	NO 😄
Post-quantum secure	NO 😒	YES 😄	NO 😒
Crypto assumptions	DLP + secure bilinear pairing 😒	Collision resistant hashes 😄	Discrete log 😉

Comparison of NIZK Algorithms

	Proof Size	Prover Time	Verification Time
SNARKs (has trusted setup)	288 bytes	2.3s	10ms
STARKs	45KB-200KB	1.6s	16ms
Bulletproofs	~1.3KB	30s	1100ms

Wrap-up

- NIZK Proof?
  - ZK Proofs
  - Interactive ZK → NIZK
- NIZKP 알고리즘들
  - zk-SNARKs, zk-STARKs, Bulletproof
  - 간단한 비교
- NIZKP 알고리즘들의 응용사례
  - Zcash
  - zk Rollup
  - MINA

Outline

- NIZK Proof?
  - ZK Proofs
  - Interactive ZK → NIZK
- NIZKP 알고리즘들
  - zk-SNARKs, zk-STARKs, Bulletproof
  - 간단한 비교
- NIZKP 알고리즘들의 응용사례
  - Zcash
  - zk Rollup
  - MINA

Zcash

- Bitcoin fork
- t-Address(public), z-Address(private)
  - t-to-t: Public transaction #
  - z-to-z: Private transaction(sender, receiver, amount 모두 비공개) #
  - t-to-z: Shielding #
  - z-to-t: Deshielding #
- Computation → Arithmetic Circuit → R1CS → QAP → zk-SNARK

Zcash (shielded tx)

- UTXO 대신에 *Commitment*라고 용어를 바꿈
- UTXO 소비 → *Nullifier*을 세상에 공개
- $\text{Commitment} = H\left(\text{RecipientAddr}, \text{Amount}, \rho, r\right)$ where r is random nonce
  - Rho: UTXO마다 가지는 고유한 값
- $\text{Nullifier} = H\left(\text{SpendingKey}, \rho\right)$

Zcash (shielded tx)

- UTXO 대신에 *Commitment*라고 용어를 바꿈
- UTXO 소비 → *Nullifier*을 세상에 공개
- $\text{Commitment} = H\left(\text{RecipientAddr}, \text{Amount}, \rho, r\right)$ where r is random nonce
  - Rho: UTXO마다 가지는 고유한 값
- $\text{Nullifier} = H\left(\text{SpendingKey}, \rho\right)$
- zk-SNARKs로 다음을 증명(하되 내용은 안 보여준 채로 블록체인에 추가)
  - Input note들의 가치 합과 output note들의 가치 합이 동일함.
  - Sender는 input note들의 spending key를 가지고 있음(=sender가 input note들을 소유하고 있음)
  - Input note들이 유효함(=note마다 valid commitment가 사전에 공개되었음)
  - Nullifier와 Commitment들이 정확히 계산됨.
  - Nullifier들이 기존 Nullifier들과 충돌할 확률은 무시할 수 있을 정도로 작음
- 아무튼 증명이 된답니다. 자세한 원리는 패스

zk-Rollup

- 이더리움은 너무 느리고 비싸고 참여하기 어렵다
- 모든 트랜잭션을 자체 체인에서 처리하고 벌크로 묶어서 L1에 올리자
- 유효성은 어떻게 검증? zk-SNARKs
  - 트랜잭션이 아주 많아도 zk-SNARKs는 *Succinct*하므로 블럭 하나에 올릴 수 있음
  - Verification은 SNARKs에 따라 스마트 컨트랙트가 해줌(실패하면 당연히 롤백)

zk-Rollup

- 이더리움은 너무 느리고 비싸고 참여하기 어렵다
- 모든 트랜잭션을 자체 체인에서 처리하고 벌크로 묶어서 L1에 올리자
- 유효성은 어떻게 검증? zk-SNARKs
  - 트랜잭션이 아주 많아도 zk-SNARKs는 *Succinct*하므로 블럭 하나에 올릴 수 있음
  - Verification은 SNARKs에 따라 스마트 컨트랙트가 해줌(실패하면 당연히 롤백)
  - 스컨으로 검증한 뒤에는 *사실상* 즉시 finalize 가능
- Ex: zkSync(zk-SNARKs), Loopring(zk-SNARKs), Starkware(zk-STARKs)

참고: Optimistic Rollup

- 롤업을 할 때 **아무것도 하지 않음**. 롤업 데이터를 체인 위에 올리기만 함

```
publish(bytes _transactions) public { }
```

- 이후에 일정 기간동안 감시자들이 트랜잭션을 검증하고 유효하지 않으면 *사기 증명(Fraud Proof)*을 제출함
  - 기간은 약 1주
  - 기간이 지나면 실제로 사기였건 말건 그냥 정상으로 간주하고 확정
- 롤업해주는 사람과 감시자간의 눈치게임 문제, finality 문제 등이 있음
- 가장 많이 퍼진 롤업방식임: ***Arbitrum***, Optimism
  - zk-SNARKs보다 기능 구현이 쉬움(EVM 호환 등) → 이더리움 스케일업 시장 선점 성공
  - *근데 **zkSync 2.0**이 **EVM** 호환 구현에 성공했다 함(작년 8월 메인넷) – 팔로업 못했음. 아시는분?*

Mina Protocol

- Succinctness를 극한으로 활용
- 블록체인 전체를 머클트리로 만들고 가지마다 zk-SNARKs로 증명
  - ex: Bitcoin에서 블록 1-2-3-4를 순서대로 붙였다면
  - Given 1, 2 is valid(1-2), 2-3, 3-4를 zk-SNARKs로 증명
  - (1-2)-(2-3), (2-3)-(3-4)를 zk-SNARKs로 증명
  - ...반복 → 마지막에 하나의 zk-SNARKs 증명으로 합쳐짐
  - 발표자 주: 근데 이러면 새 블록마다 기존 체인의 길이만큼 증명을 반복해야 해서 비효율적일거같은데 몇가지 트릭이 있을듯. DAG가 아니라 트리로 만든다거나
- Bitcoin 체인 크기 300GB → 1~2KB 수준에서 계속 유지가능
  - 단, 머클트리에서 현재 상태를 찾아가기 위한 머클증명이 20KB 정도
  - 실제로 *22KB Blockchain*으로 홍보

Mina Protocol

- SNARK-based Dapp(Snapp)
  - Zcash와 유사하게 내부 정보를 숨기고 컨트랙트 결과만 공개할 수 있음
- 적용사례: Defi의 신용 평가
  - 특정 Account의 신용점수를 계산하지만 누구에게도 공개하지 않고 기준값 이상인지만 보여줌
  - Teller라는 무담보 대출 Defi와 파트너십
  - 근데 링크가 깨졌고 현재의 Teller는 이더리움에 디플로이됨. 확인 필요
    - 아마 신용평가만 Mina를 쓸 가능성이 있는데 중간에 엎어진 건지 설명이 부족함
- 다른 적용사례: Mina SSO(스마트컨트랙트로 로그인하기)
  - 역시 스컨이 로그인 결과는 알려주지만 원하는 정보만 공개하도록
  - 이건 아직 컨셉트고 개발중인듯

Mina Protocol – etc.

- 컨센서스: Ouroboros Samasika
- Block Producer와 별개로 SNARKs Producer 노드가 필요
  - SNARK 특성상 증명 과정이 computation-heavy하기 때문
  - 블럭생성 + 증명 생성 시 추가 보상
- 아쉽게도 Finality가 약함
  - 약 1시간(15 blocks) 기다리면 99.9% 이상의 확률로 신뢰가능

Wrap-up

- NIZK Proof?
  - ZK Proofs
  - Interactive ZK → NIZK
- NIZKP 알고리즘들
  - zk-SNARKs, zk-STARKs, Bulletproof
  - 간단한 비교
- NIZKP 알고리즘들의 응용사례
  - Zcash
  - zk Rollup
  - MINA

Wrap-up

- 유망한 기술임에는 이견의 여지가 없음
- 엔드유저에게 설득하는 게 쉽지 않음
  - ECDSA(와 군론과 체론 등등 추상대수…), DLP, …
  - NIZK 기반 솔루션들 글들 보면 죄다 zk-SNARKs 쉽게 설명한다고 진땀빼고 있음
  - 어차피 전공자 아니면 *검증 가능할 정도로 깊게* 이해하기 어렵지 않나?
- SSL을 생각해보면 결국 오래 쓰면서 자연스럽게 검증되고 받아들여짐
  - 엔드유저들이 납득하는 이유: '전문가들이 그렇게 조언하니까'
  - 은행 로그인할때는 주소창에 초록 자물쇠 어쩌고저쩌고
- NIZK 기술이 *언젠가* 메인스트림 기술이 될지도?
  - Arbitrum 대신 zkSync 쓰고 솔라나 대신에 Mina 쓰고(죄송합니다)

끝

- Mina가 신기해서 찾아봤는데 2주 삭제됨
- Mina 재단에 손해배상을 청구하기 위해 소량을 구매했습니다
  - 정반꿀 노리시면 short on FTX NOW
- THIS IS NOT AN INVESTMENT ADVICE

- 질문?