교과목 : 정보보호

8. Authentication Protocol & ZKP

2022학년도 2학기
Suk-Hwan Lee

- **참고자료**
  - ✓ (교재) Mark Stamp, Information Security: Principles and Practice, 2nd edition
  - ✓ [부교재] William Stallings, Cryptography and Network Security, 7th edition

Protocols

- Human protocols – the rules followed in human interactions
  - Example: Asking a question in class
- Networking protocols – rules followed in networked communication systems
  - Examples: HTTP (HTTPS), FTP, etc.
- Security protocols – the (communication) rules followed in a security application
  - Examples: SSL, IPSec, Kerberos, etc.

Ideal Security Protocol

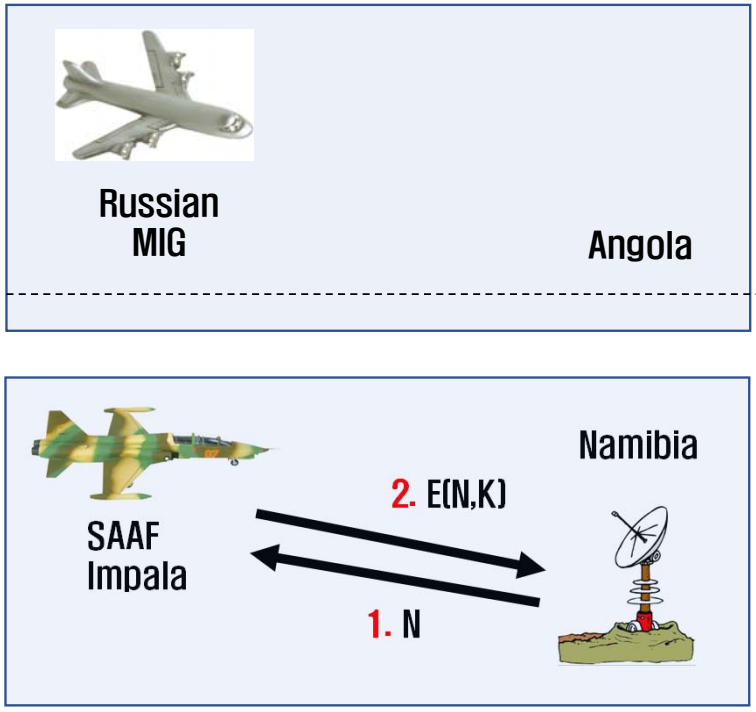
- Satisfies security requirements
  - Requirements must be precise
- Efficient
  - Minimize computational requirement – in particular, costly public key operations
  - Minimize delays/bandwidth
- Robust
  - Must work when attacker tries to break it
  - Works even if environment changes
- Easy to use and implement, flexible, etc.

❖ Very difficult to satisfy all of these!

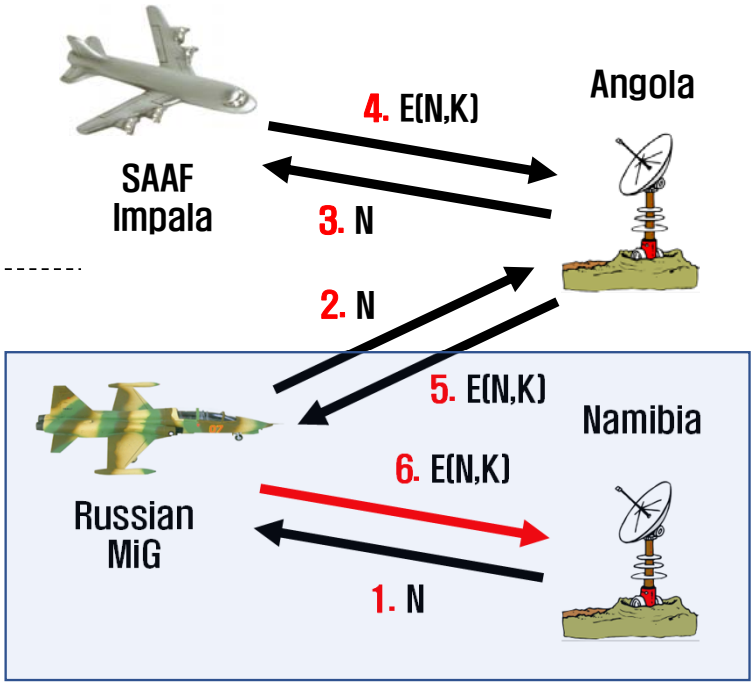
Identification Friend or Foe (IFF)
(피아식별장치)

Military needs many specialized protocols

Many cases, it could recognize friends as enemies, or ….



MIG in the Middle



Authentication Protocols

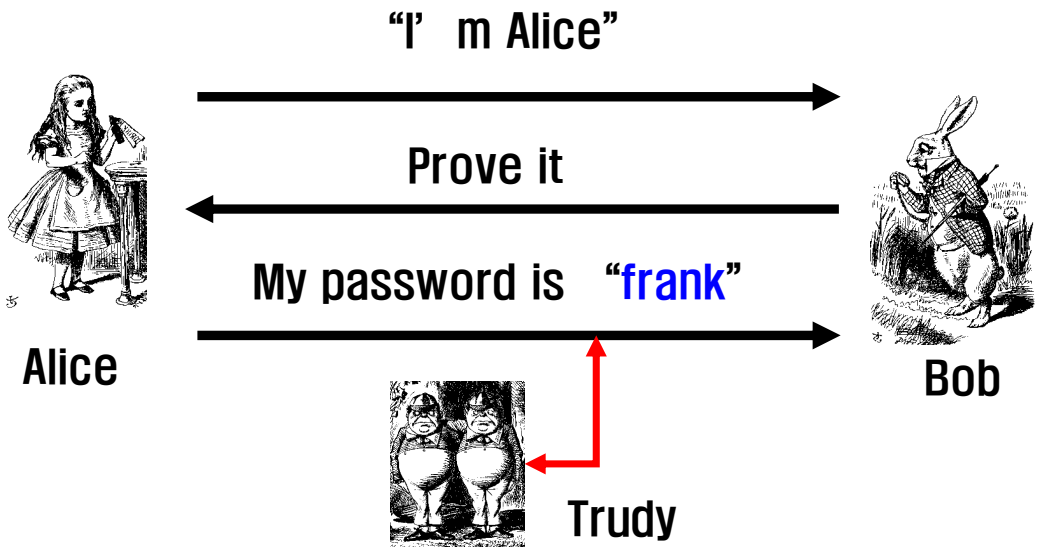
Authentication

- Alice must prove her identity to Bob
  - ✓ Alice and Bob can be humans or computers
- May also require Bob to prove he's Bob (mutual authentication)
- May also need to establish a session key
- May have other requirements, such as
  - ✓ Use only public keys
  - ✓ Use only symmetric keys
  - ✓ Use only a hash function
  - ✓ Anonymity, plausible deniability, etc., etc.

- Authentication on a stand-alone computer is relatively simple
  - ✓ "Secure path" is the primary issue
  - ✓ Main concern is an attack on authentication software (we discuss software attacks later)
- Authentication over a network is much more complex
  - ✓ Attacker can passively observe messages
  - ✓ Attacker can replay messages
  - ✓ Active attacks may be possible (insert, delete, change messages)

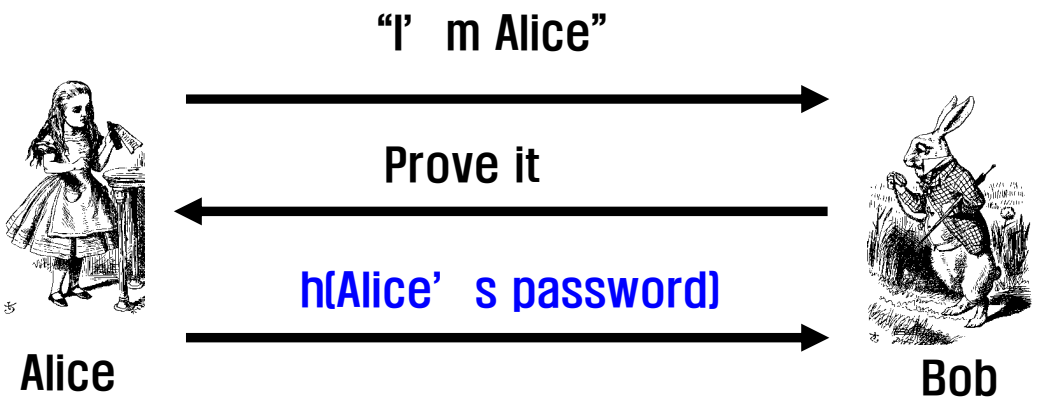
Authentication Protocols

Simple Authentication



- Simple and may be OK for standalone system
- But highly insecure for networked system
  - ✓ Subject to a replay attack
  - ✓ Bob must know Alice' s password

Better Authentication

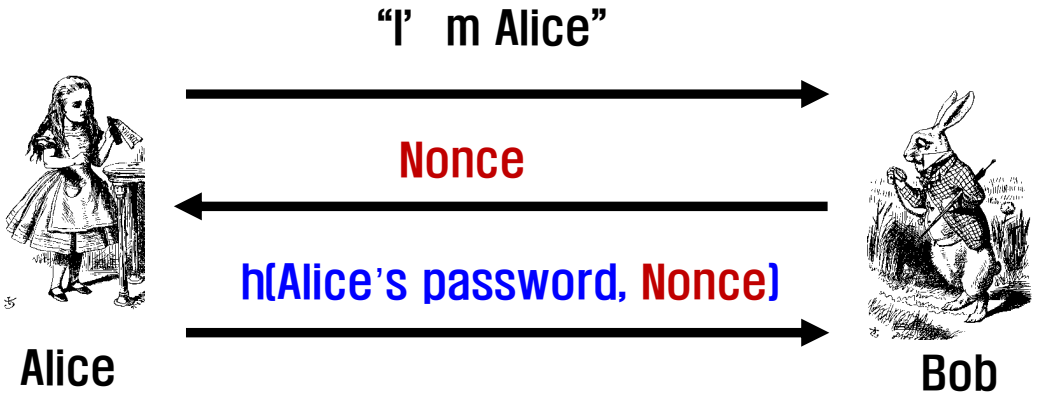


- Better since it hides Alice' s password
  - From both Bob and attackers
- But still subject to replay

Challenge-Response

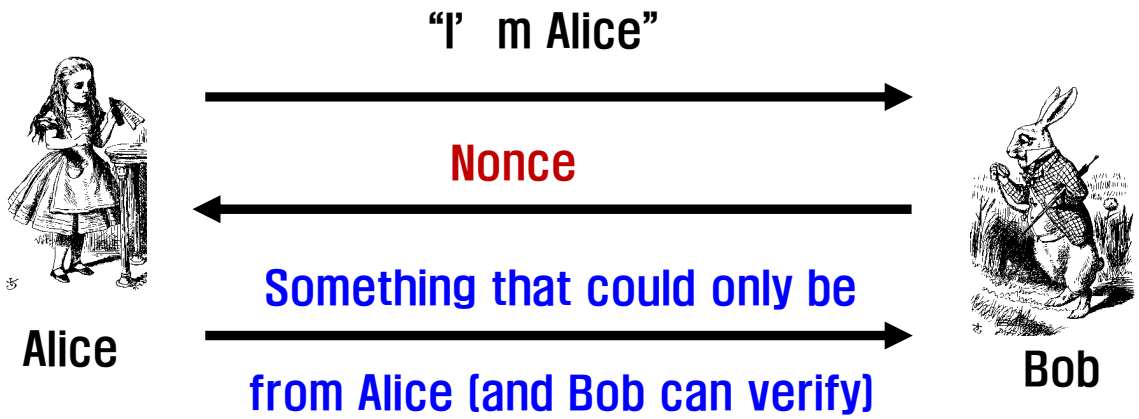
Nonce : a number or bit string used only once

- To prevent replay, challenge-response used
- Suppose Bob wants to authenticate Alice
  - ✓ Challenge sent from Bob to Alice
  - ✓ Only Alice can provide the correct response
  - ✓ Challenge chosen so that replay is not possible
- How to accomplish this?
  - Password is something only Alice should know…
  - For freshness, a "number used once" or nonce



- Nonce is the challenge
- The hash is the response
- Nonce prevents replay, insures freshness
- Password is something Alice knows
- Note that Bob must know Alice' s password

Challenge-Response



- What can we use to achieve this?
  - Hashed pwd works, crypto might be better
  - Will be discussed for Symmetric key, Public key, and so on

Symmetric Keys

- Hashed password works, but…. encryption is much better here (why?)
- Symmetric Key Notation
  - Encrypt plaintext P with key K

    $$C = E(P,K)$$
  - Decrypt ciphertext C with key K

    $$P = D(C,K)$$
  - Here, we are concerned with attacks on protocols, not directly on the crypto
  - We assume that crypto algorithm is secure

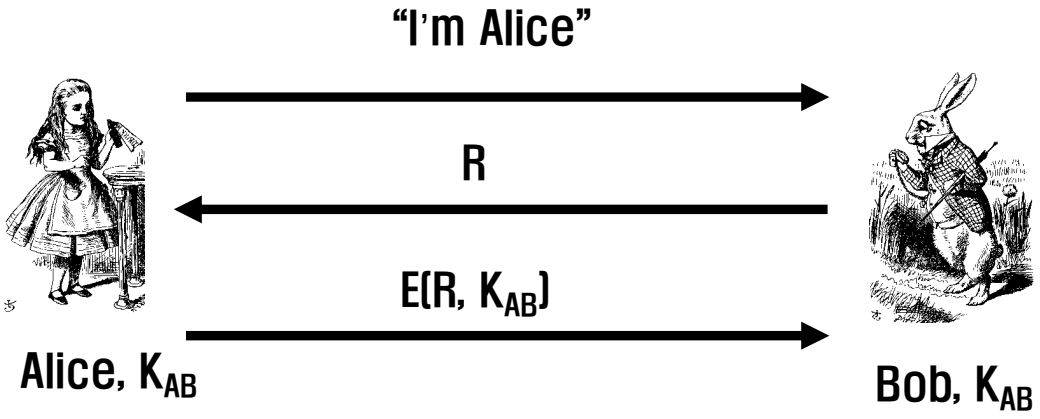
Note:

- When discussing protocols, the primarily concern is attacks on protocol, not attacks on the cryptography used in protocols.
- Consequently, we' ll assume that the underlying cryptography is secure.

Authentication : Symmetric Key

- Alice and Bob share symmetric key K_{AB}
- Key K_{AB} known only to Alice and Bob
- Authenticate by **proving** knowledge of shared symmetric key
- How to accomplish this?
  - ✓ Cannot reveal key,
  - ✓ must prevent replay (or other) attack,
  - ✓ must be verifiable…

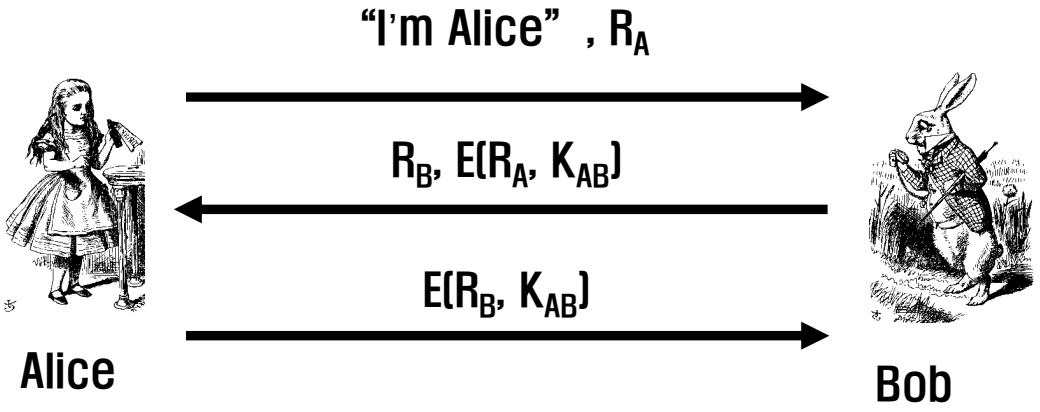
R:Nonce



- Secure method for Bob to authenticate Alice (prevents a replay attacks)
- But, Alice does not authenticate Bob (lacks mutual authentication)
- So, how can we achieve mutual authentication?

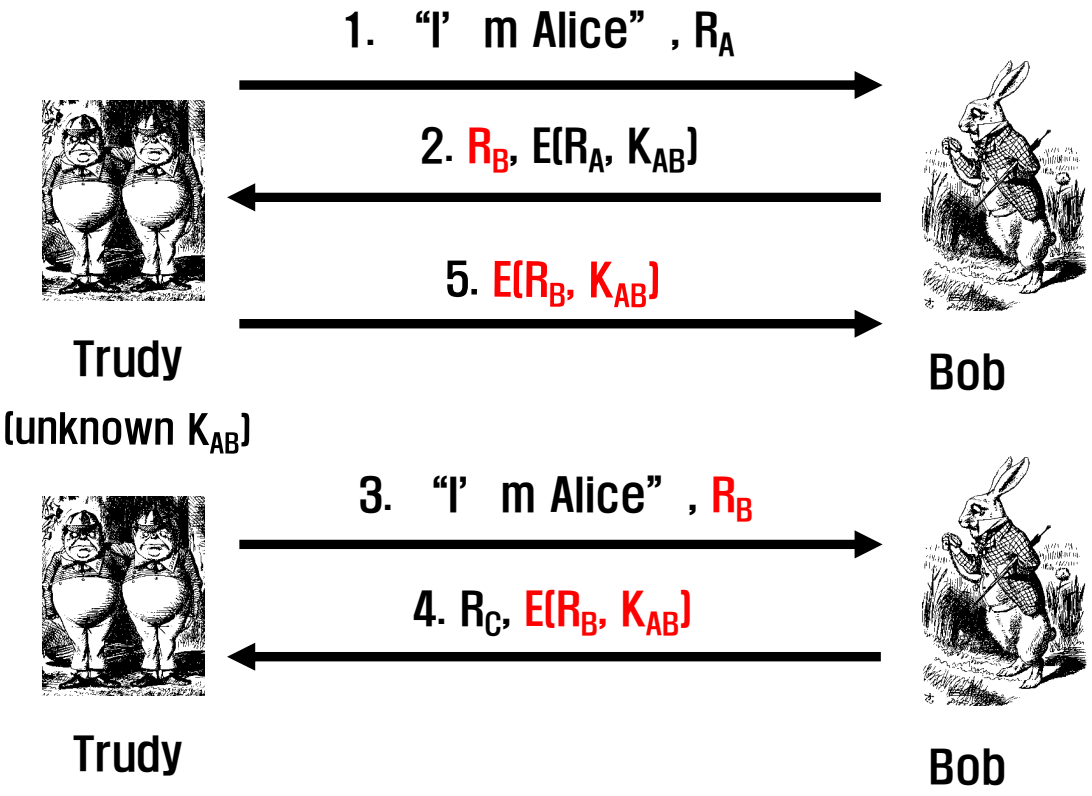
Mutual Authentication

- Since we have a secure one-way authentication protocol…
- The obvious thing to do is to use the protocol twice
  - Once for Bob to authenticate Alice
  - Once for Alice to authenticate Bob
- This has to work…



- This provides mutual authentication…. or does it"
- But, subject to **reflection attack**, which is method of attacking a challenge-response authentication system

Mutual Authentication Attack : Reflection attack



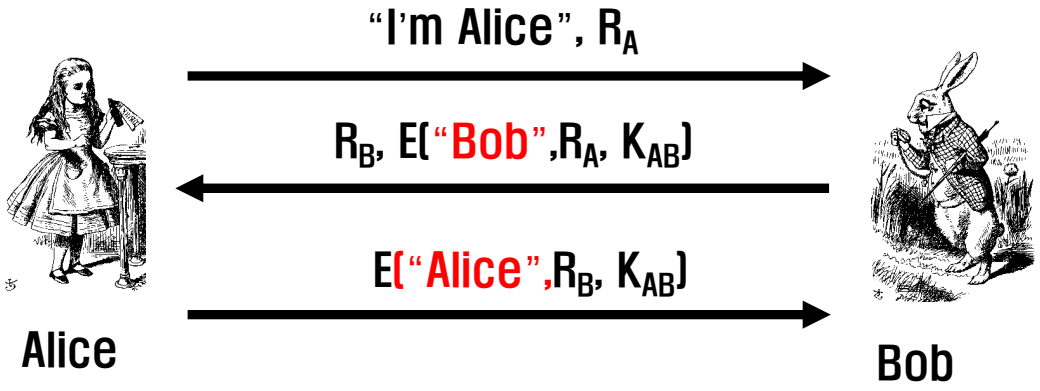
- Our one-way authentication protocol not secure for mutual authentication
- The "obvious" thing may not be secure
- Also, if assumptions or environment **changes**, protocol may not work

✓ This is a common source of security failure

✓ For example, Internet protocols

Note : non-mutual authentication protocol may not be secure for mutual authentication

Mutual Authentication (Revised)



- Encrypt the user's identity together with the nonce (R)
- Prevent Trudy's previous attack since she cannot use a response from Bob for the third message – Bob will realize that he encrypted it himself.

- Do these "insignificant" changes help?
- Yes!

Public Keys

$$C = E_{pk}(M)$$
$$M = (D_{sk}(C))$$
$$M = (D_{sk}(E_{pk}(M)))$$

Remember that in public key cryptography:

- **anybody** can do **public key** operations, while
- only Alice can use her **private key**.

Public Key Notation

- Encrypt M with Alice's public key (PK_A) : $C=\{M\}_{Alice} \rightarrow M=[C]_{Alice}$ $C=E(M,PK_A) \rightarrow M=D[C,SK_A)$
- Sign M with Alice's private key (SK_A) : $C=[M]_{Alice} \rightarrow M=\{C\}_{Alice}$
- Then
  $[\{M\}_{Alice}]_{Alice} = M$: This is called Encrypt & Sign
  $\{[M]_{Alice}\}_{Alice} = M$: This is called Sign & Encrypt
- Anybody can use Alice's public key
- Only Alice can use her private key

Session Key

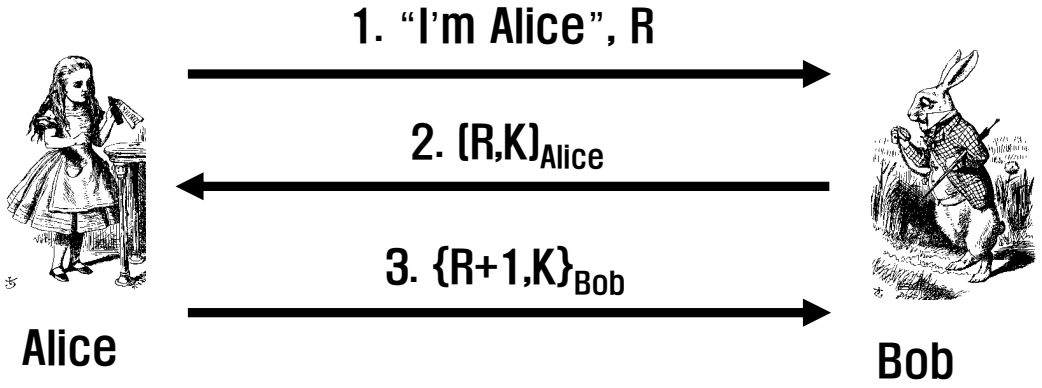
- Session key: temporary key, used for a short time period
- Usually, a session key is required in addition to authentication
  - ✓ It is a temporary symmetric key for the current session
  - ✓ Used for confidentiality and/or integrity
- Why session keys?
  - ✓ To limit the amount of data encrypted with any particular session
  - ✓ Serves to limit damage if one session key compromised
- Thus, establishing the session key as part of the authentication protocol.
  - ✓ That is, when the authentication is complete, we will also have securely established a shared symmetric key
  - ✓ Therefore, when analyzing an authentication protocol, we need to consider attacks on the authentication itself, as well as attacks on the session key.

Session Key

- How to authenticate and establish a session key (i.e. shared symmetric key)
  - ✓ When authentication completed, Alice and Bob share a session key
  - ✓ Trudy cannot break the authentication…. and
  - ✓ Trudy cannot determine the session key

- Authentication & Session Key
  - ✓ It looks to be straightforward to include a session key using the secure public key authentication protocol

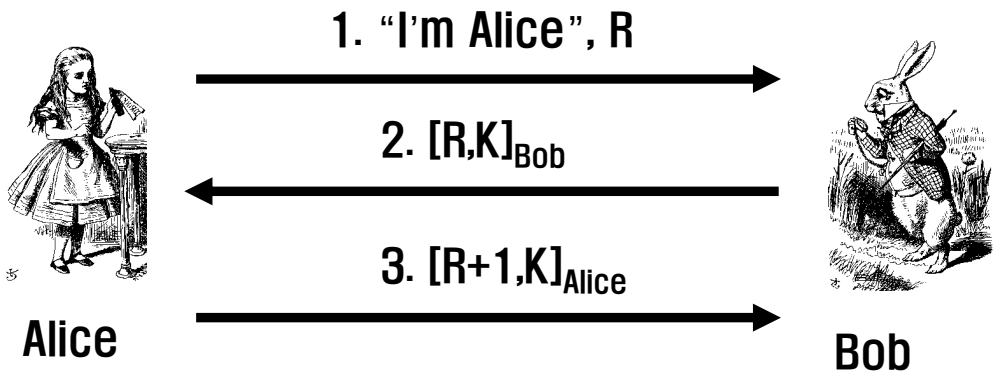
- Is this secure?
  - ✓ Alice is authenticated and session key is secure
  - ✓ Alice' s "nonce" R is useless to authenticate Bob
  - ✓ The key K is acting as Bob' s nonce to Alice



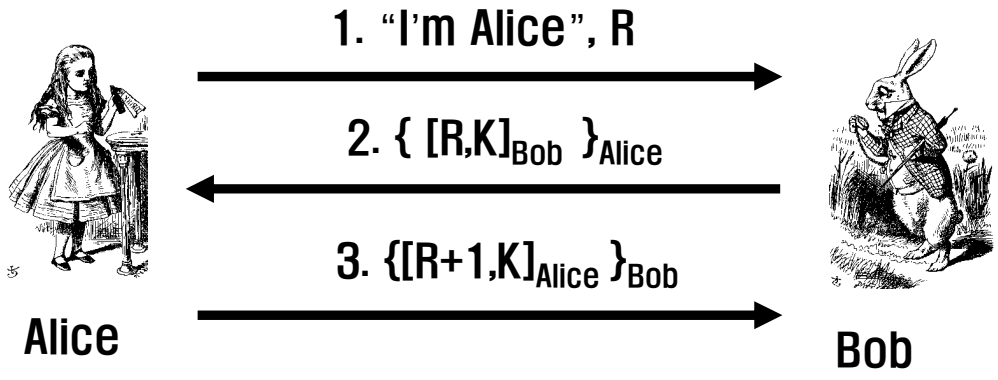
- No mutual authentication – only Alice is authenticated

Public Key Authentication and Session Key

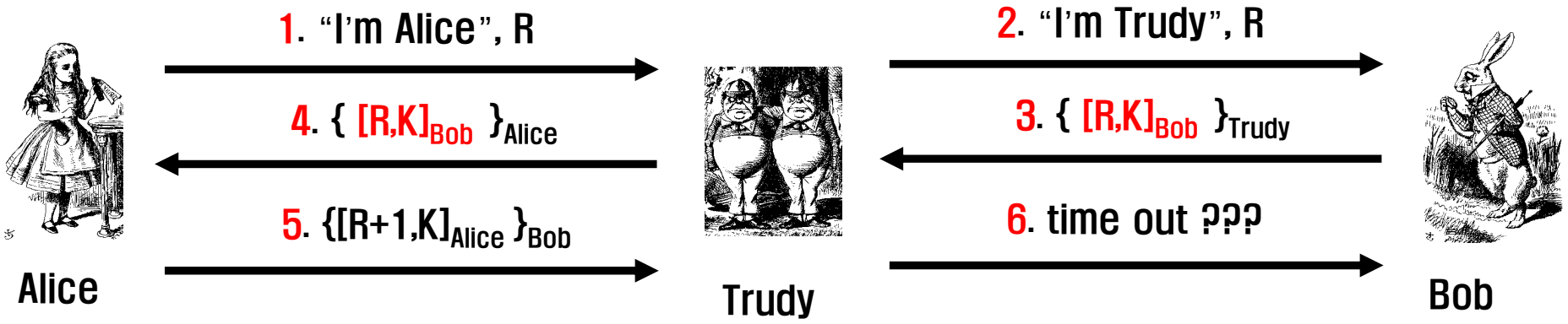
- It uses digital signatures instead of public key encryption, Is this secure?
  - ✓ It does provide Mutual Authentication (very good),
  - ✓ but… fatal flaw… session key is not protected (very bad)
  - ✓ Can we combine these two to achieve both mutual authentication and a secure session key



- It provides mutual authentication and a session key using sign and encrypt
- Is this secure?
  - ✓ No! It' s subject to subtle/elusive MiM attack



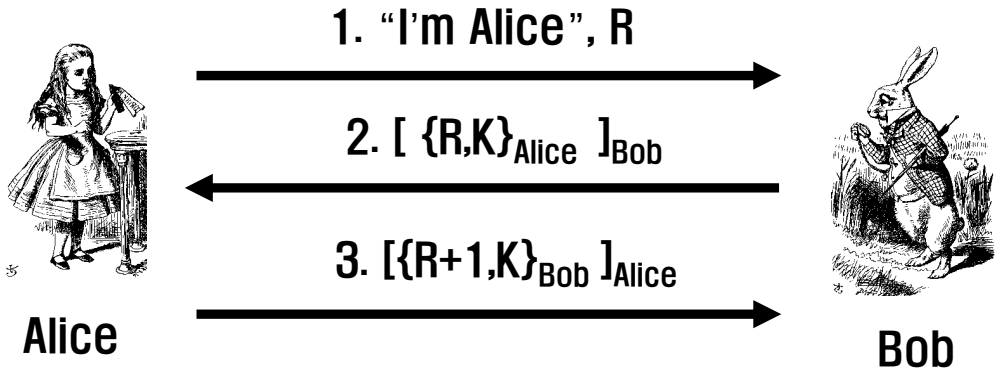
Public Key Authentication and Session Key



- Trudy can get $[R,K]_{Bob}$ and K from 3. Then Trudy can apply Bob's public key and get R and K.
- Alice uses this same key K
- And Alice thinks she's talking to Bob, and K now is with Trudy

Public Key Authentication and Session Key

- Encrypt and sign approach?

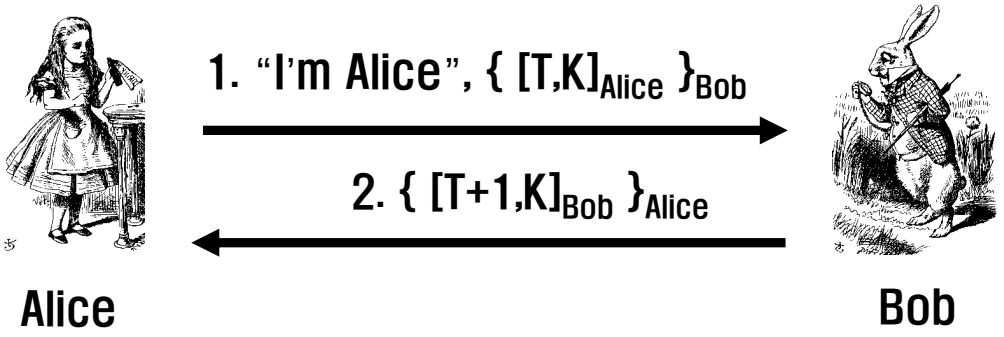


- Is this secure?
- Seems to be OK, but Anyone can see $\{R,K\}_{Alice}$ and $\{R+1,K\}_{Bob}$,
  - ✓ Available to anyone who has access to Alice's or Bob's public keys
  - ✓ Which, by assumption, is anybody who wants them
  - ✓ But, they can be recorded but not decrypted

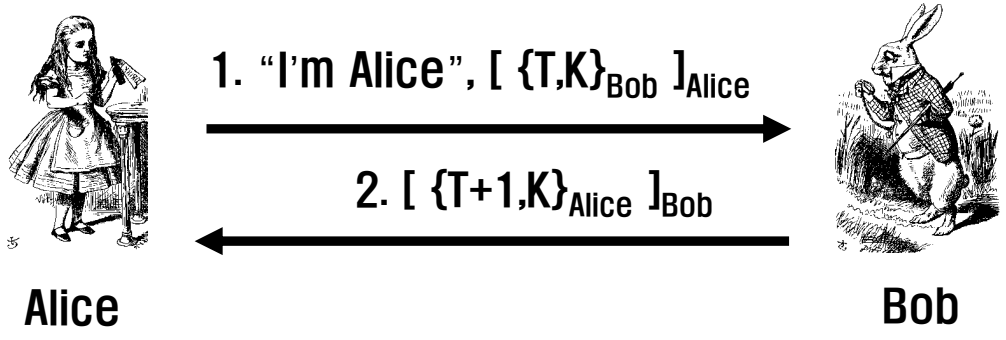
Timestamps instead of nonces (R)

- **Timestamps can be used instead of nonces**
  - ✓ Assuming that the current time is known to both Alice and Bob
    - Alice sends the time she performed her calculation and Bob accepts if it is within the **clock skew**
  - ✓ A timestamp T is derived from **current time** (*value in milliseconds*)
    - current timestamp ensures freshness
  - ✓ **Timestamps can be used to prevent replay (good) (i.e. Used in Kerberos protocol)**
  - ✓ **Timestamps reduce number of messages (good)**
    - A challenge that both sides know in advance (potential for increased efficiency)
  - ✓ "Time" is a **security-critical parameter** (bad)
    - Attack Alice's system clock and then you cause Alice's authentication to fail
- **Clocks not same and/or network delays are present,**
  - ✓ Thus, must allow for clock skew (시간오차) creates risk of **replay**
  - ✓ **How much clock skew is enough?**
    - **Too much** : Trudy can do a replay

Public Key Authentication with TImestamp

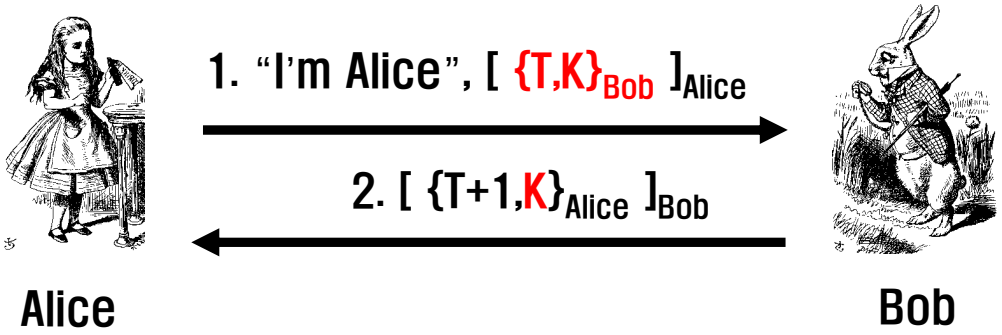


- Provides the **timestamp version** of the ***sign and encrypt*** protocol
- Secure mutual authentication? Session key secure? Seems to be OK

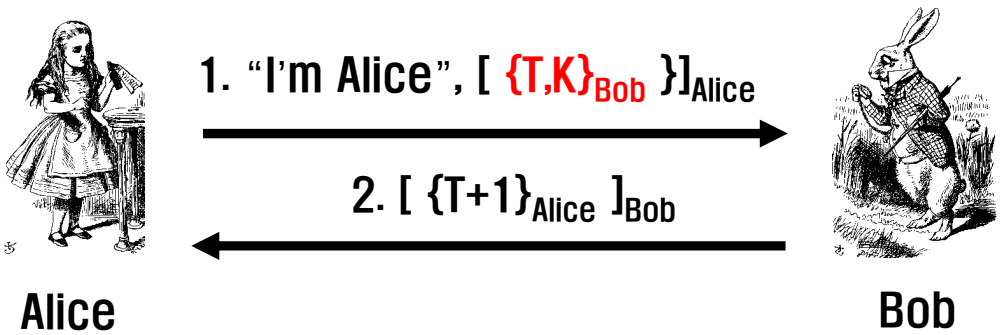


- Is the timestamp version of the following ***encrypt and sign*** also secure?
- Secure authentication and session key?
  - ✓ No, The obvious is not always correct!
- Trudy can use Alice' s public key to find $\{T,K\}_{Bob}$ and then… open a connection and send it to Bob… then Bob will send the key K to Trudy

Public Key Authentication with TImestamp



- Trudy obtains Alice-Bob shared session key K.
- Note : Trudy must act within clock skew
- Can we improve/secure it?



- Is this "encrypt and sign" secure?
  - ✓ Yes, seems to be OK
- Does "sign and encrypt" also work here?

Public Key Authentication

- Sign and encrypt with nonce…
  - Insecure (MiM)
- Encrypt and sign with nonce…
  - Secure
- Sign and encrypt with timestamp…
  - Secure
- Encrypt and sign with timestamp…
  - Insecure
- Protocols can be subtle!

Zero Knowledge Proofs

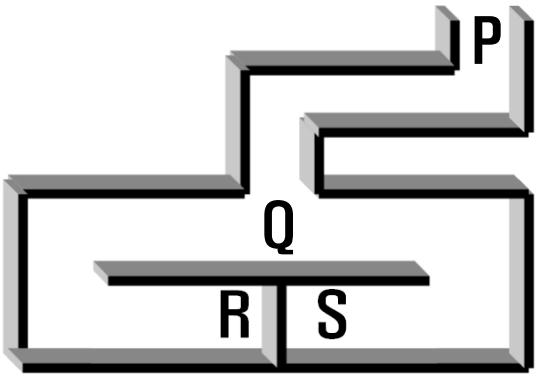
Zero Knowledge Proofs (ZKP)

- 영지식 증명 : 거래 상대방에게 어떠한 정보도 제공하지 않은 채, 자신이 해당 정보를 가지고 있다는 사실을 증명하는 것
- Alice wants to prove that she knows a secret without revealing any info about it
- Bob must verify that Alice knows secret
  - ✓ But, Bob gains no information about the secret
- Process is probabilistic
  - ✓ Bob can verify that Alice knows the secret to an arbitrarily high probability, how?
- An "interactive proof system"

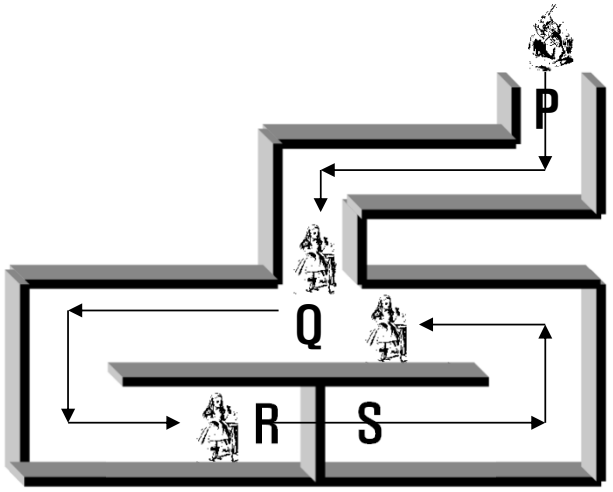
- 영지식 증명은 3가지 조건을 모두 만족해야 함
  - Completeness (완전성) : 어떤 조건이 참이라면 신뢰할 수 있는 검증자(honest verifier)는 신뢰할 수 있는 증명자(honest prover)에 의해 이 사실을 납득할 수 있어야 한다.
  - Soundness (건전성) : 어떤 조건이 거짓이면 신뢰할 수 없는 증명자(dishonest prover)는 거짓말을 통해 검증자에게 조건이 참임을 절대 납득시킬 수 없다.
  - Zero-knowledge (영지식성) : 어떤 조건이 참일 때, 검증자는 이 조건이 참이라는 사실 이외의 아무 정보를 알 수 없다.

Bob’ s Cave

- Alice claims to know secret phrase to open path between R and S (“open sasparilla”)
- Can she convince Bob that she knows the secret without revealing phrase?



- Bob : “Alice, come out on S side”
- Alice (quietly) : “Open sarsaparilla”
- Suppose that Alice does not know the secret..
  - ✓ Without knowing secret, Alice could come out from the correct side with probability ½
  - ✓ If Bob repeats this n times and Alice does not know secret, then Alice can only fool Bob with probability $(1/2)^n$



Best Authentication Protocol?

- **What is best depends on many factors…**
  - The sensitivity of the application
  - The delay that is tolerable
  - The cost that is tolerable
  - What crypto is supported
    - ✓ Public key, symmetric key, hash functions
  - Is mutual authentication required?
  - Is a session key required?
  - Is PFS a concern?
  - Is anonymity a concern?, etc.

부 록

Zero-Knowledge proof ::
chapter 1. Introduction to Zero-Knowledge Proof & zk-SNARKs

참조자료 :
1) 원본 : Ameer Rosic, What are zkSNARKs? The Comprehensive Spooky Moon Math Guide, https://blockgeeks.com/guides/what-is-zksnarks/
2) 번역 및 수정 : 서울대, Blockchain Academy Decipher, https://medium.com/decipher-media/zero-knowledge-proof-chapter-1-introduction-to-zero-knowledge-proof-zk-snarks-6475f5e9b17b

- 블록체인 문제
  - ✓ Privacy (프라이버시) : Public Blockchain에서 개인 간 Transaction이 발생하면, Transaction의 상세 내역이 모두에게 공개되어, 이로 인한 피해가 발생
  - ✓ Capacity (용량) : 블록체인의 용량
- zk-SNARKs 개념 : Privacy & Capacity 해결할 수 있는 방식으로 대두
  - ✓ 거래의 익명성 보장하고, 블록체인 데이터를 짧은 해시값으로 압축시켜 저장 용량을 줄여줌
  - ✓ Zero-knowledge proof가 기반 기술

Zero-knowledge Proof (ZKP)

- ZKP : Prover가 자신이 알고 있는 지식을 공개하지 않으면서, 그 지식을 알고 있다는 사실을 verifier에게 증명하는 proof system → 1980년대 MIT의 S. Goldwasser, S. Micali, and C. Rackoff가 제시
  - ✓ Prover : knowledge을 알고 있음을 증명하는 주체
  - ✓ Verifier : prover가 해당 knowledge을 알고 있다는 사실을 확인 및 검증해주는 주체

- ZKP의 이론적인 기반은 interactive proof system이다.
  - ✓ Interactive proof system이란, prover와 verifier 상호 간 메시지를 교환하는 computation을 모델링한 abstract machine(이론적인 컴퓨팅 모델)을 뜻한다.
  - ✓ 이전 Interactive proof system에서 prover는 전능하고 무한정의 계산 자원을 가지고 있지만 신뢰할 수 없는 존재인 반면, verifier는 한정된 계산 자원을 가지고 있지만 신뢰할 수 있는 존재임을 전제
  - ✓ ZKP 제시한 3명 연구자들 : interactive proof system에서 추가적으로 verifier가 악의적인 목적을 품는 시나리오 고려
    - 예; prover가 자신의 주민등록번호를 알고 있고, 자신이 알고 있다는 사실을 증명하기 위해 관련 정보를 verifier에게 보낸 상황을 가정하자. 이 때, verifier는 전달받은 prover의 정보를 타인에게 판매하여 부당한 이익을 챙길 수 있음

Zero-knowledge Proof (ZKP)

- ZKP 제시한 3명 연구자들 : 기존 interactive proof system에 2가지 질문
  1. 누구나 verifier가 knowledge를 누설하지 않았다는 걸 확인할 수 있는가
  2. verifier가 검증 과정 동안 알고 있어야 하는 knowledge의 비중은 어느 정도인가

➢ prover가 제공한 proof를 통해 악의적인 verifier가 검증을 수행할 수는 있지만, prover의 knowledge 자체에 대해서는 유추할 수 없는 proof system이 필요했다. 제시한 해결책이 ZKP 이다.

ZKP Property

- ZKP는 항상 다음과 같은 조건을 모두 만족하여야 함
  - Completeness: 어떤 조건이 참이라면, honest verifier는 honest prover에 의해 이 사실을 납득할 수 있다.
  - Soundness: 어떤 조건이 거짓이라면, dishonest prover는 거짓말을 통해 verifier에게 조건이 참임을 절대 납득시킬 수 없다.
  - Zero-knowledge: 어떤 조건이 참일 때, verifier는 이 조건이 참이라는 사실 이외의 정보를 아무것도 알 수 없다

일반적 interactive proof system의 properties

ZKP의 주요 property로 추가

Examples of ZKP

❖ Case 1 : Alibaba’ s Cave (ZKP의 대표적 문제)

- 고리 형태의 동굴 가운데에 문이 있고, 그 문에는 도어락이 설치되어 있다.
- verifier는 prover에게 동굴에 설치된 도어락의 비밀번호가 무엇인지 직접 물어보지 않고 prover가 비밀번호를 알고 있다는 명제가 참인지를 확인하고 싶다.
- 이 조건문이 참인지를 확인하기 위해, 다음과 같은 과정을 따른다.

1) prover가 먼저 동굴에 들어간 다음, 도어락 근처로 이동한 후 verifier를 동굴 안으로 부른다.
2) verifier는 A와 B의 갈림길에 서서, prover에게 특정 길(A 또는 B)로 나오라고 지시한다.
3) prover는 verifier가 지시한 길로 나온다.
4) 1.~3. 과정을 반복한다.

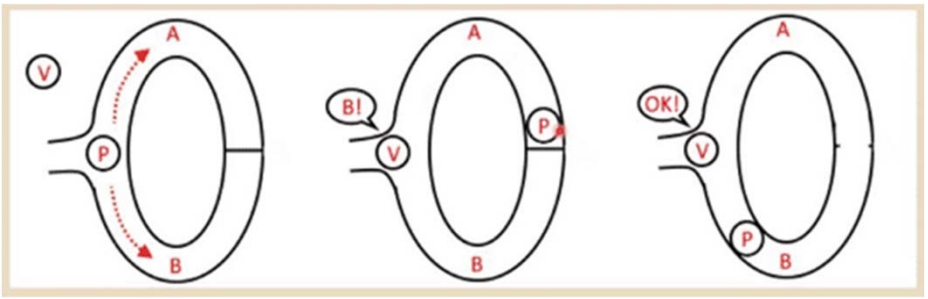


그림 1. Alibaba’ s Cave [출처] What is zkSNARKs: Spooky Moon Math

1.~3. 과정만을 보았을 때는, prover가 정말 도어락의 비밀번호를 알고 있어서 verifier가 지시한 길로 나왔는지 확신할 수 없다. 우연에 의해 처음부터 prover가 A를 선택했고, 문을 열지 못했으나 verifier가 A로 나오라고 지시하는 경우가 충분히 있을 수 있기 때문이다. 따라서 우연의 가능성을 낮추기 위해, 위 과정을 일정 횟수 이상 반복한다. 여러 case를 수행하였을 때에도 매번 prover가 verifier의 지시대로 행동한다면, prover가 도어락의 비밀번호를 알고 있다는 사실을 verifier에게 납득시킬 수 있다.

Examples of ZKP

❖ Case 1 : Alibaba’ s Cave (ZKP의 대표적 문제)

- Completeness: 어떤 조건이 참이라면, honest verifier는 honest prover에 의해 이 사실을 납득할 수 있다.
  - 여러 번의 경우에도 prover가 verifier의 지시를 지속적으로 따른다면, verifier는 prover가 도어락의 키를 안다고 납득할 수 있다.
- Soundness: prover가 정직하지 못하다면, prover는 거짓말을 통해 verifier에게 조건의 타당함을 납득시킬 수 없다
  - prover가 사실은 도어락의 키를 모르지만 안다고 거짓말을 하였을 경우, verifier에게 언젠가 한 번은 지시대로 수행하지 못하는 경우가 생길 것이기 때문에 자신이 도어락의 키를 안다는 것을 증명할 수 없다.
- Zero-knowledge: 어떤 조건이 참일 때, verifier는 이 조건이 정확히 무엇인지 알지 못하여야 한다.
  - 여러 번의 수행을 통해서 verifier는 prover가 도어락의 키를 알고 있다는 사실을 납득하였지만, 도어락의 키가 무엇인지에 대해서는 알지 못한다.

Examples of ZKP

❖ Case 2 : Finding Waldo prover가 Waldo를 찾았다는 것을 verifier에게 증명

- 색상이 혼합된 그림에서 Waldo 인물 찾기

그림 2. 이 그림에서 Waldo를 찾아라!
[출처] What is zkSNARKs: Spooky Moon Math

- First Solution

① 먼저, prover와 verifier가 같은 사진을 복사하여 한 장씩 나눠가진 다음, prover는 사진 속에서 Waldo가 있는 부분을 찾아서 자른 다음 나머지 부분을 버린다.

② 나머지 부분을 버렸기 때문에 prover가 자른 Waldo 조각이 사진에서 어디에 있는지는 verifier에게 밝혀지지 않는다.

③ prover는 Waldo가 포함된 조각을 verifier에게 보여준다. prover와 verifier는 같은 사진을 나누어 가졌기 때문에, verifier는 prover가 증거로써 제시한 Waldo 조각이 같은 사진로부터 나왔음을 납득할 수 있고, prover는 Waldo가 어디에 위치해 있는지 밝히지 않으면서(zero-knowledge) Waldo가 위치한 곳을 찾았음을 verifier에게 증명할 수 있게 된다.

✓ Soundness 미충족 : prover가 verifier를 속이는 다양한 방법이 존재하기 때문 (prover와 verifier가 따로 떨어져 있을 때 Waldo의 모습을 기존의 사진과 같은 재질과 크기의 용지에 프린트한 후 내가 Waldo가 위치한 곳을 아는 것처럼 눈속임할 수 있게 된다.)

Examples of ZKP

❖ Case 3 : Mini Sudoku [참조] https://chriseth.github.io/notes/talks/intro_to_zksnarks/#/1

- Sudoku : 주어진 9*9 정사각형 판에 있는 각각의 가로, 세로줄에 1부터 9까지의 숫자를 한 번씩만 사용하여 판을 모두 채우는 게임이다. 이 9*9 정사각형 판은 다시 9개의 작은 3*3 정사각형 판으로 쪼개져 있는데, 해당 정사각형 판에도 1에서 9까지의 숫자가 한 번씩만 들어가야 한다.

- Mini Sudoku : 4*4 축소 버전 (9*9는 경우의 수가 많으므로)

- Mini Sudoku 주어졌을 때, prover와 verifier가 해결하고자 하는 일은 다음과 같음
  - ➢ prover: Mini-Sudoku의 해답을 공개하지 않고, 해당 문제가 풀릴 수 있다는 것을 증명하려 함
  - ➢ verifier: prover가 Mini-Sudoku의 해답을 알고 있다는 것을 납득하려 함

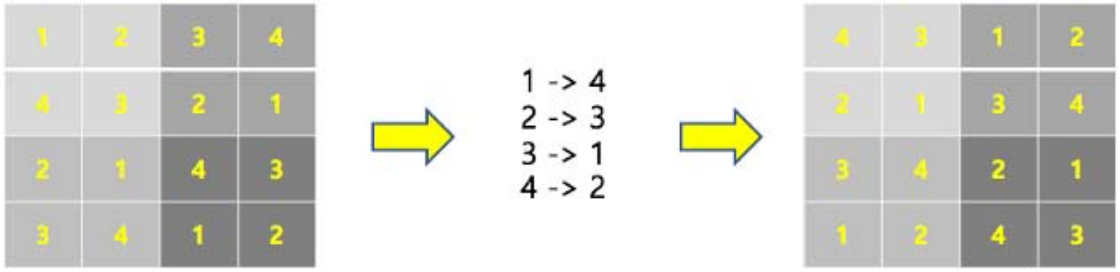
- 증명 과정
  1) Shuffling solution
  2) shuffled solution과 mapping table을 가리는 과정
  3) verifier가 선택한 sub solution을 공개하는 과정
  4) shuffling solution

Examples of ZKP

❖ Case 3 : Mini Sudoku

1) Shuffling solution

- prover는 solution을 verifier에게 바로 공개할 수는 없으니, random one-to-one mapping을 통해, 가지고 있는 solution을 shuffling함

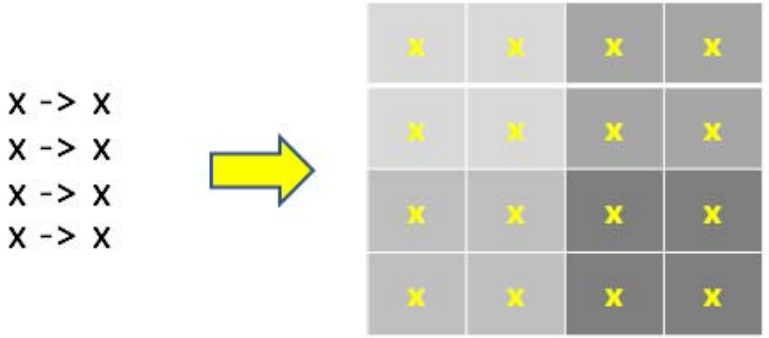


prover가 실제 가지고 있는 solution

mapping table에 의해 한 번 shuffle된 shuffled solution

2) shuffled solution과 mapping table을 가리는 과정

- prover는 shuffled solution과 mapping table을 verifier에게 공개하지 않고, 각각의 값을 숨긴 뒤에 공개를 하게 된다.



masking shuffled solution & mapping table

Examples of ZKP

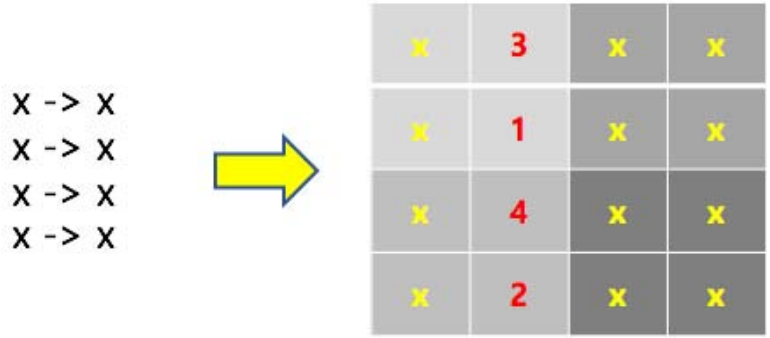
❖ Case 3 : Mini Sudoku

3) verifier가 선택한 sub solution을 공개하는 과정

- Sudoku에는 여러 sub solution들이 존재한다.
- 만일 prover가 solution을 알고 있다면, verifier가 다음과 같은 정보의 공개를 요구했을 때 이를 모두 만족해야 비로소 sudoku 문제의 해법을 안다고 할 수 있다.

- ➢ 특정 row를 공개했을 때, 해당 row에 포함된 숫자가 모두 달라야 함
- ➢ 특정 column을 공개했을 때, 해당 column에 포함된 숫자가 모두 달라야 함
- ➢ 특정 sub-square를 공개했을 때, 해당 sub-square에 포함된 숫자가 모두 달라야 함
- ➢ 임의의 cell들을 공개했을 때, 공개된 cell들이 기존 sudoku의 constraint를 해치지 않아야 함

- 위 조건들 중에서 verifier가 한 가지를 선택하여 정보공개를 요구할 수 있다.
- 예를 들어서, verifier가 "column 2를 공개하라" 는 요청을 했다고 가정하자. prover는 요구 조건에 맞추어 masking한 column 2을 공개하게 된다.



revealing sub solution

Examples of ZKP

❖ Case 3 : Mini Sudoku

- 공개된 solution은 sudoku의 constraint(특정 column에는 1부터 4까지의 숫자가 한번씩만 들어간다)를 해치지 않기 때문에, verifier는 prover가 solution을 알고 있다고 납득할 수 있는 근거가 됨
- But, prover가 solution을 알고 있다고 하기에는 무리가 있음. 왜냐하면 sub-solution을 만족하더라도 공개되지 않은 판에 적힌 숫자들은 sudoku의 solution이 아닐 수 있기 때문이다.
- 따라서 1.부터 3.까지의 과정을 여러 번 반복함으로써 판단의 정확성을 높이게 된다.

- 3개의 예제들은 모두 prover와 verifier가 항상 on-line 상태여야 한다는 점이다.
- 이러한 constraint는 전체적으로 보았을 때 매우 비효율적인 구조로, 어느 정도의 개선이 필요했다.
- 1986년, Fiat과 Shamir가 결국 non-interactive ZKP를 제시하였고 이는 prover와 verifier의 on-line 여부에 관계없이 증명을 할 수 있도록 고안되었다.

From interactive to non-interactive: Schnorr Identification Protocol

- Non-interactive 핵심 : prover와 verifier의 메시지 교환이 최소화 되어야 한다는 것을 의미
  - ✓ prover가 특정 메시지를 verifier에게 보낸 다음, verifier로부터 추가로 전달받는 메시지가 필요하다면 이는 non-interactive한 방식이 아님
  - ✓ prover가 증명에 필요한 메시지를 보낸 후, 연결이 끊어지더라도 그 메시지가 verifying되어야 함

- Schnorr Protocol : prover가 자신의 private key를 공개하지 않고 이를 가지고 있다는 것을 증명할 수 있는 방법
  - ✓ 키 교환과 관련된 프로토콜이기 때문에 실제 블록체인에 적용될 수 있는 non-interactive ZKP와 가장 가까운 예제임

prover만 알고 있는 변수(Prover's side)와, prover 및 verifier가 같이 공유하는 변수(Global variables)

Prover's side	Global variables
s: $private\ key\ (0 < s < q)$ r: $random\ num$ $X = a^r \bmod p$	p: 소수 q: $(p-1)$의 인수 a: $a^q \equiv 1\ (mod\ p)$ v: $public\ key\ (v = a^{-s} \bmod p)$

From interactive to non-interactive: Schnorr Identification Protocol

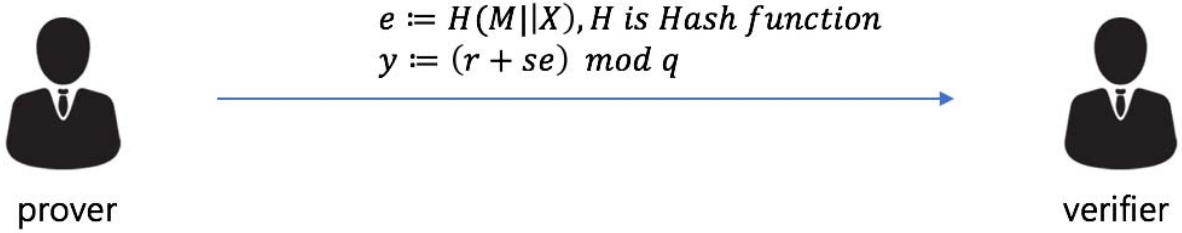
- prover는 s의 값을 verifier에게 공개하지 않고, 자신이 s를 알고 있다는 것을 납득시키려 한다.

prover Side

- ✓ random value인 r (0<r<q) 을 하나 선택하여 그에 따른 X 값을 계산한다.
- ✓ 메시지 M과 X 값을 concatenate 한 후, 이를 hash function에 통과시켜 새로운 서명 e를 만든다.
- ✓ e를 만든 이후에, prover는 이로부터 또 다른 서명 y 값을 도출해낼 수 있다.
- ✓ prover는 M과 도출해낸 서명 e, y 를 verifier에게 보낸다.
- ❖ *참고로, 이 방식은 non-interactive하기 때문에 prover는 해당 메시지를 보내고 난 후 off-line 상태가 되어도 증명에는 지장이 없다.*

Prover's side
s: *private key* $(0 < s < q)$ r: *random num* $X = a^r \bmod p$

Global variables
p: 소수 q: $(p-1)$의 인수 a: $a^q \equiv 1 \pmod p$ v: *public key* $(v = a^{-s} \bmod p)$



From interactive to non-interactive: Schnorr Identification Protocol

- prover는 s의 값을 verifier에게 공개하지 않고, 자신이 s를 알고 있다는 것을 납득시키려 한다.

verifier Side

✓ verifier는 전달받은 M,e,y를 통해 reproduce한 X' 와 prover가 가지고 있는 X가 일치하는지를 확인하는 것이다.

$$X' = a^y * v^e \bmod p$$
$$<=> X' = a^y * a^{-se} \bmod p = a^{y-se} \bmod p$$
$$y = r + se \Rightarrow r = y - se$$
$$\Rightarrow X' = a^r \bmod p$$

✓ 여기서 verifier는 X값을 prover로부터 전달받지 못하여, X=X' 확인할 수 없다. 그래서 verifier는 도출한 X' 를 이용하여 e' 값을 계산한다.

$$e' = H(M||X')$$

Prover's side
s: private key $(0 < s < q)$ r: random num $X = a^r \bmod p$

Global variables
p: 소수 q: $(p-1)$의 인수 a: $a^q \equiv 1 \pmod p$ v: public key $(v = a^{-s} \bmod p)$

✓ 만일 X=X' 라면, e = e' 가 되어 verifier는 X=X' 임을 납득할 수 있음

✓ 결국 verifier는 private key s 값이 무엇인지는 모르지만, prover가 s를 알고 있다는 사실을 수학적으로 납득할 수 있게 된다.

From interactive to non-interactive: Schnorr Identification Protocol

- 위 예제가 ZKP의 property를 만족하는지 확인해 보겠다.

✓ Completeness: 어떤 조건이 참이라면, honest verifier는 honest prover에 의해 이 사실을 납득할 수 있다. => X = X' 가 참이라는 사실을 honest prover에 의해 납득할 수 있다.

✓ Soundness: prover가 정직하지 못하다면, prover는 거짓말을 통해 verifier에게 조건의 타당함을 납득시킬 수 없다. => prover가 private key를 몰랐을 경우, prover는 X = X' 임을 납득시킬 수 없었을 것이다..

✓ Zero-Knowledge: 어떤 조건이 참일 때, verifier는 이 조건이 정확히 무엇인지 알지 못하여야 한다. => verifier는 결국 prover의 private key를 알지 못한다.

Add one more property — Succinctness: zk-SNARKs

- zk-SNARKs는 zero-knowledge Succinct Non-interactive Argument of Knowledges의 줄임말로, 기존의 non-interactive ZKP에서 succinctness(간결함)가 추가된 개념이다.

- 기존 non-interactive ZKP
  - ✓ prover가 항상 on-line 상태일 필요가 없지만, 증명을 완료하는데 상당한 시간이 걸림
  - ✓ Schnorr Identification Protocol에서, verifier가 X' 를 계산하기 위해서는 a^r mod p 라는 연산을 수행해야 한다. 이 연산은 a, r, p의 값이 커짐에 따라 상당한 시간이 소요됨

- zk-SNARKs의 중요한 property로 succinctness
  - ✓ interactive proof system은 verifier가 한정된 계산 자원을 가지고 있음을 전제로 함.
  - ✓ prover가 특정 knowledge를 알고 있다는 증거로 제출한 proof가 엄청난 size의 데이터라면, 이의 증명은 굉장히 비효율적일 것이다.
  - ✓ zk-SNARKs에서는 non-interactive ZKP의 proof size를 줄이고 빠른 시간 내에 verify를 수행할 수 있도록 하여 non-interactive ZKP의 실용성을 극대화하였다.

Add one more property — Succinctness: zk-SNARKs

- zk-SNARKs는 과정
  1. keygen : key generator G를 이용해 key pair (pk, vk)를 생성하는 과정
  2. Prove: prover가 proof 를 생성하는 과정
  3. Verify: verifier가 proof를 verifying하는 과정

  ➢ 3 단계 과정을 통해서 verifier는 prover의 knowledge를 직접 확인하지 않고 이를 빠른 시간에 verifying 할 수 있음

1. Keygen: key generator G를 이용해 key pair (pk, vk)를 생성하는 과정

✓ prover가 알고 있다고 주장하는 값인 witness(w)가 있을 때, 이를 parameter로 받는 특정 program C가 있다고 하자 (아래의 예는 가장 직관적이고 간결한 예시이며, 실제로 C의 세부적인 구현은 당연히 훨씬 더 복잡할 수 있다)

```
function C(x, w) {
        return ( hash(w) == x) ;
}
```

Add one more property — Succinctness: zk-SNARKs

- ✓ generator G에 program C(w를 알고 있음을 증명할 수 있는)와 random sampling seed인 lambda를 parameter로 하여, key pair를 생성한다.
  - 여기서 lambda는 prover 및 외부에 노출되어서는 안된다. prover가 이 값을 알고 있을 경우 fake proof를 생성할 수 있기 때문이다. 그렇기 때문에 key generating은 verifier가 수행하게 된다.

```
G(C, lambda) = (pk, vk)
pk = proving key
vk = verifying key
```

- ✓ C의 program size bound를 l, input(x) size의 bound를 n, execution time bound를 T라고 할 때, key generating의 시간 복잡도는

$$O\,(l + n + T)\ \cdot\ \log(l + n + T)$$

  - program이 길고, input size가 크고, 실행 시간이 길수록, key generating에 소요되는 시간이 길다.

Add one more property — Succinctness: zk-SNARKs

2. Prove: prover가 proof(prf)를 생성하는 과정

- ✓ 증명 알고리즘을 P, 증명하고자 하는 witness인 w, w의 hash를 x 라고 할 때, prf를 구하는 방법은 다음과 같다.

```
prf = P(pk, w, x)
```

- ✓ prover는 prf를 계산한 후, prf만을 verifier에게 전달한다.
- ✓ prf을 계산하는 데 걸리는 시간은 입력값 x와 w의 크기에 비례한다.
- ✓ 계산된 prf로부터 w 값을 유추할 수 없으며, prf의 길이는 매우 짧다. 실제로 Eli의 논문에서 구현된 zk-SNARKs 구현체의 proof는 230바이트, 288바이트 정도로 매우 짧았다.

Add one more property — Succinctness: zk-SNARKs

3. Verify: verifier가 prf를 verifying하는 과정

- ✓ verifier는 prf를 prover로부터 전달 받은 후, verifying algorithm V를 수행하여 prf의 진위 여부를 판단
- ✓ verifying에 걸리는 시간은 매우 짧음 (zk-SNARKs의 간결함 특성)

```
boolean: V(vk, x, prf)
```

- ✓ 위 값이 TRUE이면 prover는 w 값을 정말로 알고 있다고 할 수 있고, FALSE이면 prover는 w 값을 속였다고 판단하게 된다.

Applications of zk-SNARKs

➢ Confidential Tx: Ethereum에서 transaction의 상세 내용을 숨기는 방식

- ✓ zk-SNARKs를 이용하면 Ethereum에서 ETH를 송금하는 트랜잭션을 발생시킬 때, sender가 receiver에게 트랜잭션을 보낸 사실은 공개하고, sender와 receiver의 balance와 transfer amount에 대한 상세 내용을 숨기는 전송 방식이 가능하다.

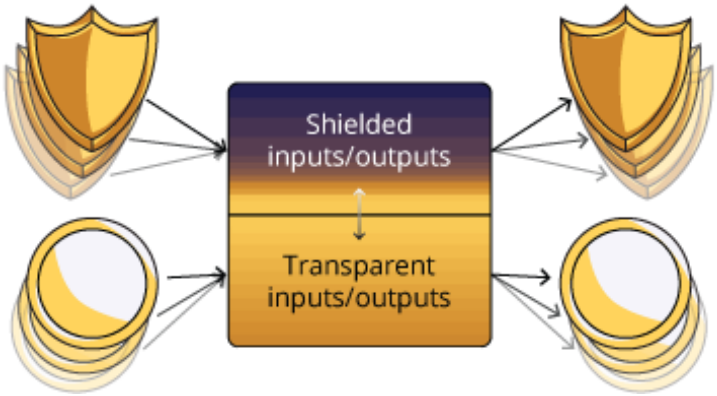
➢ For Blockchain Scalability

- ✓ zk-SNARKs를 이용하여 blockchain의 scalability를 해결할 수 있다는 아이디어가 많이 제시
- ✓ blockchain에 참여하는 client는 실제로 블록의 내용을 모르더라도 full node의 proof를 보고 해당 블록의 내용이 변조되지 않았음을 빠르게 verifying할 수 있다. 결국, 이는 client의 block sync 속도를 급격하게 높여주게 되어 새로운 client들이 빠르게 네트워크에 참여할 수 있게 된다.
- ✓ 또한, blockchain의 append-only 한 속성으로 blockchain의 용량은 시간이 지날수록 계속 커진다. 특히 Ethereum에서는 trie 형태의 여러 global data structure(state trie, transaction trie, receipt trie 등)가 쌓여가고 있다. 이 trie 구조에는 각종 smart contract의 code와 storage 등 여러 데이터들이 자리를 차지하고 있는데, zk-SNARKs를 통해 실제 데이터를 pruning하고 데이터에 대한 proof만 남기는 방식으로 압축시킴으로써 contract가 차지하는 공간을 획기적으로 줄일 수 있다. smart contract의 code와 storage를 체인 위에 공개하지 않으면서 state change를 증명하는 방법이 가능하기 때문이다.

Applications of zk-SNARKs

➢ Zcash

✓ Zcash는 영지식 증명을 기반으로 완전히 익명화된 트랜잭션의 전송을 가능하게 한다. 기본적인 동작방식은 Bitcoin과 상당히 유사하나, 가장 두드러지는 차이점은 address의 종류가 2가지라는 것이다.



✓ Zcash는 영지식 증명을 기반으로 완전히 익명화된 트랜잭션의 전송을 가능하게 한다. 기본적인 동작방식은 Bitcoin과 상당히 유사하나, 가장 두드러지는 차이점은 address의 종류가 2가지라는 것이다.

✓ Shielded address와 Transparent address를 이용하여, 사용자들은 이 두 address를 사용하여 ZCash의 화폐인 ZEC를 공개적으로 전송할지, 사적으로 전송할지 선택할 수 있다.

✓ Shielded address에서 Transparent address로 송금을 하게 되면 전달받은 ZEC가 공개가 되며, 반대의 경우는 숨겨진다.

✓ ZCash는 두 주소 체계의 공존을 통해 자금 흐름의 추적을 거의 불가능하게 만들었고, 이는 ZCash가 익명 코인이라고 불리는 대표적인 이유 중 하나이다.