



고급 암호기술 2부: 고급전자서명, 익명기술

NOTE 14

DATA

한국기술교육대학교 컴퓨터공학부 김상진

sangjin@koreatech.ac.kr
www.facebook.com/sangjin.kim.koreatech

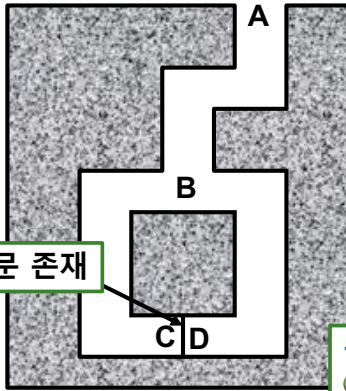
교육목표

- 영지식 증명
- 은닉 채널
- 부인불가 서명
- 지정된 확인자 서명
- 프록시 서명
- 일괄 서명, 다중 서명, 결합 서명
- 익명 인증 기술
 - 익명인증서, 익명ID, 그룹서명, 링서명, 은닉서명
- 익명 통신
 - 믹스넷, 양파 라우팅



영지식 증명

- Alice는 어떤 사실을 알고 있음. 그 사실을 상대방에게 알려주지 않고 그 사실을 알고 있다고 증명하고 싶음
- 예) Alice는 동굴의 비밀문 위치와 이 문을 여는 비밀 주문을 알고 있음. 이것을 Bob에게 증명하고 싶지만 Bob에게는 비밀 주문을 가르쳐주고 싶지 않음



프로토콜

1. Bob은 A 위치에서 기다린다.
2. Alice는 C나 D까지 이동한다. (Bob은 현재 위치에서는 Alice가 어디로 이동했는지 알 수 없다)
3. Bob은 B까지 이동한다.
4. Bob은 왼쪽 또는 오른쪽 중 한 쪽으로 나오라고 Alice에게 외친다.
5. Alice는 비밀 주문을 이용하여 Bob의 지시를 따른다.
6. 이 과정을 n 번 반복한다.

중요한 사실.

- Bob은 비밀 주문을 얻을 수 없음
- 비전이성(non-transferable): Bob은 이 과정을 비디오로 녹화하여도 다른 사람에게 Alice가 이 사실을 알고 있다고 증명할 수 없음

Where's Waldo(Wally)?



이 그림에 숨어 있는 Waldo의 위치를 영지식으로 어떻게 증명?



영지식 증명 요구사항

완전성(completeness): statement가 참이고, 증명자와 확인자가 정직하면 증명은 통과됨
 건전성(soundness): statement가 거짓이면, 부정한 증명자가 확인자를 속일 수 없음
 영지식: statement가 참이면 확인자가 얻게 되는 것은 그것이 참이라는 것 외에는 없음

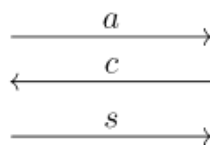
이산대수 영지식 증명 (1/3)

Prover P



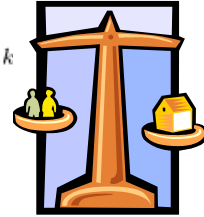
$$y = g^x$$

$$\begin{aligned} w &\in_R \mathbb{Z}_q^* \\ a &= g^w \\ s &= w - cx \end{aligned}$$



$$\begin{aligned} c &\in_R \{0, 1\}^k \\ a &\stackrel{?}{=} g^s y^c \end{aligned}$$

Verifier S



증명자가 c 를 추측할 수 있으면 다음과 같이 y 의 이산대수를 모르더라도 증명 가능함

$$\begin{aligned} s &\in_R \mathbb{Z}_q^* \\ a &= g^s y^c \end{aligned} \quad \begin{array}{c} \xrightarrow{a} \\ \xleftarrow{c} \\ \xrightarrow{s} \end{array} \quad \begin{aligned} c &\in_R \{0, 1\}^k \\ a &\stackrel{?}{=} g^s y^c \end{aligned}$$

따라서 c 가 k 비트이면 이산대수를 모르는 상태에서 증명에 성공할 확률은 $1/2^k$ 임

이산대수 영지식 증명 (2/3)

아무나 다음과 같이 트랜스크립트를 만들어 제시하면 실제 프로토콜의 트랜스크립트와 구분할 수 없음. 즉, 확인자는 다른 사람에게 증명자가 알고 있다는 사실을 증명할 수 없음

$$\begin{aligned} s &\in_R \mathbb{Z}_q^* \\ c &\in_R \{0, 1\}^k \\ a &= g^s y^c \end{aligned} \quad \begin{array}{c} \xrightarrow{a} \\ \xleftarrow{c} \\ \xrightarrow{s} \end{array} \quad a \stackrel{?}{=} g^s y^c$$

동일한 w 를 사용하면 다음 두 식을 이용하여 x 를 계산할 수 있음

$$\begin{aligned} s &= w - cx \\ s' &= w - c'x \end{aligned}$$

영지식 증명은 어떤 값이 어떤 특성을 가지고 있다는 것을 그 특성을 보여주지 않고 증명하기 위해 암호기술에서 많이 사용함

예) $A = g^a, B = h^a$ 가 있을 때 $\log_g A$ 와 $\log_h B$ 가 같음을 영지식 증명할 수 있음

이산대수 영지식 증명 (3/3)

- 대부분의 영지식 증명은 상호작용 없이 증명자가 홀로 증명하는 방식으로 바꿀 수 있음
- 이 비상호작용 증명은 c 를 계산할 때 메시지를 포함하여 전자서명으로 사용할 수 있음

$$\begin{aligned} w &\in_R \mathbb{Z}_q^* \\ a &= g^w \\ c &= H_k(g||y||a) \\ s &= w - cx \end{aligned} \xrightarrow{c, s} c \stackrel{?}{=} H_k(g||y||g^s y^c)$$

비상호작용 버전(non-interactive)

이산대수를 모르는 증명자가 s 를 임의로 선택하여도 c 를 계산할 수 없어 속이는 것이 힘들
 H_k 에 메시지를 포함하면 전자서명으로 사용 가능 (Schnorr 전자서명)
 x 를 알고 있는 사용자만 증명을 만들 수 있음

- 보통 비상호작용 증명은 비전이성을 제공하지 못함. 하지만 특정 확인자만 확인할 수 있도록 증명을 구성할 수 있음

영지식 최신 기술 동향

- zk-SNARK(zero knowledge Succinct Non-interactive ARgument of Knowledge), zk-STARK(Transparent)
 - Succinct: 증명이 간결함 → 빠르게 검증할 수 있음
 - Argument of Knowledge: 증명자의 능력 제한
 - 무한한 컴퓨팅 자원과 시간이 있으면 증명 위조 가능
 - zk-SNARK는 특정 수학적 기반의 영지식 증명이기 때문에 일반적으로 활용하기 힘들 수 있음
 - 이 때문에 front-end/back-end 개념으로 확장됨
 - front-end(컴파일러)는 각 분야에서 증명하고자 하는 것을 zk-SNARK를 구현한 back-end가 처리할 수 있도록 번역해 줌
 - Zcash에 실제 활용하고 있음
- zk-SNARK는 신뢰할 수 있는 초기 설정(안전성에 중대한 영향)이 필요하지만 zk-STARK는 이것이 필요 없음(transparent)
 - zk-STARK에서 S를 Scalability로 제시하기도 함

은닉 채널

- 예) 감옥에 수감되어 있는 두 죄수는 비밀스럽게 메시지를 주고 받고 싶음.
두 죄수는 메시지를 보낼 때 항상 교도관이 그 내용을 보게 됨. 따라서 평범한 내용의 글을 교환하지만 거기에 실제 비밀스러운 내용을 포함하고 싶음
- 전자 서명을 교환할 때 서명된 메시지 외에 서명자와 수신자만이 알 수 있는 추가 메시지를 포함할 수 있음
 - 이것을 은닉 채널(subliminal channel)이라 함
- Simmons는 1983년에 ElGamal 서명을 이용하여 은닉 채널을 만들 수 있음을 보임
- 정보보호 측면에서는 서명에 은닉채널을 포함할 수 없도록 만들어야 함

여기서 은닉 채널은 실제 통신 채널이 아니라 일반 메시지 내에 데이터를 숨기는 것을 말하며, 이렇게 숨겨 메시지를 보내면 송신자와 수신자 사이에 제3자는 알 수 없는 은닉 채널이 존재한다고 말할 수 있음

부인불가 서명 (1/2)

- 일반 전자서명은 그것의 확인키가 있으면 누구나 검증할 수 있음
- 부인불가 서명(undeniable signature)은 전자서명의 확인을 제한할 수 있음
- 전자서명을 확인하기 위해서는 원 서명자와 프로토콜을 수행해야 함
 - 서명자의 동의 없이는 서명을 확인할 수 없음
- 이 경우 원 서명자는 나중에 프로토콜 참여를 거부할 수 있음
- 이 문제를 극복하기 위해 두 종류의 프로토콜 제공
 - 서명 확인 프로토콜: 확인자에게 주어진 서명이 특정 메시지에 대한 서명자의 유효한 서명임을 확인해 줌
 - 부인 프로토콜: 확인자에게 주어진 서명이 특정 메시지에 대한 서명자의 유효한 서명임이 아님을 확인해 줌
- 부인 프로토콜에 참여하지 않으면 이 서명이 유효하다는 것의 반증이 되는 형태임

부인불가 서명 (2/2)

- 확인 프로토콜과 부인 프로토콜은 모두 영지식 증명과 마찬가지로 프로토콜에 참여한 사람이 다른 사람에게 재차 확인해 줄 수 없음
- Boyar 등은 부인불가 서명을 일반 서명으로 전환할 수 있는 방법도 제안함
 - 이 서명을 **전환 가능 부인불가 서명**(convertible undeniable signature)라 함
- 나중에 서명자가 특정 값을 공개하면 부인불가 서명이 일반 서명으로 전환되어, 서명자의 도움 없이 서명을 확인할 수 있음

지정된 확인자 서명

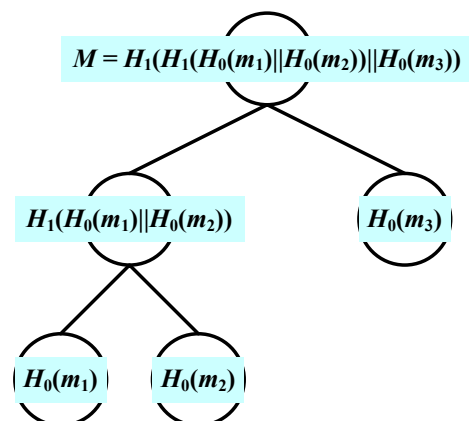
- **지정된 확인자 서명**(designated confirmer signature)은 부인불가 서명과 일반 서명의 절충안
- 부인불가 서명은 서명자의 도움 없이는 서명을 확인할 수 없는 문제점이 있음
- 반면에 일반 서명은 누구나 항상 서명을 확인할 수 있음
- 지정된 확인자 서명은 다음과 같이 진행됨
 - 서명자는 확인자와 프로토콜을 수행하여 서명을 전달함
 - 부인불가 서명과 마찬가지로 확인자는 이 프로토콜의 트랜스크립트를 이용하여 다른 사람들에게 서명의 유효성을 증명할 수 없음
 - 서명자가 지정한 사용자는 이 서명을 다른 사용자들에게 확인해 줄 수 있음

프록시 서명

- **프록시 서명(proxy signature)**은 서명자가 자신의 서명키를 지정된 프록시에게 주지 않고, 자신을 대신하여 서명할 수 있도록 해줌
- 프록시 서명의 요구사항
 - **요구사항 1. 구별가능성**: 프록시 서명은 일반 서명과 구분될 수 있어야 함
 - **요구사항 2. 위조불가능성**: 원 서명자와 지정된 프록시 서명자만 유효한 프록시 서명을 생성할 수 있음 (문제점)
 - **요구사항 3**: 프록시 서명자는 프록시 서명이 아닌 실제 서명은 할 수 없어야 함
 - **요구사항 4. 확인가능성**: 확인자는 프록시 서명으로부터 원 서명자의 서명된 메시지에 대한 동의를 확인할 수 있어야 함
 - **요구사항 5. 식별가능성**: 원래 서명자는 프록시 서명을 통해 프록시 서명자를 확인할 수 있어야 함 (프록시가 다수)
 - **요구사항 6. 부인불가능성**: 프록시 서명자는 자신이 서명한 프록시 서명을 부인할 수 없어야 함 (요구사항 2와 상충됨)

일괄 서명

- **일괄 서명(batch signature)**: Fiat가 1989년에 처음 제안
- 한 서명자가 동시에 여러 개의 다른 메시지들을 효율적으로 서명하기 위해 사용
- Pavloski와 Boyd는 1999년에 Merkle 해시 트리를 이용한 방법을 제안함 (문서의 개수와 상관없이 서명은 한 번)
- 옆에 그림과 같이 트리를 구성하여 루트 값에 서명함으로써 메시지 m_1 부터 m_n 에 대한 일괄 서명을 함
- 개별 메시지의 서명은 다음과 같이 구축함
 - 예) 메시지 m_1 에 대한 서명
 $\text{Sig. } A(M), H_0(m_3), R, H_0(m_2), R, m_1$
 - 예) 메시지 m_3 에 대한 서명
 $\text{Sig. } A(M), H_1(H_0(m_1)||H_0(m_2)), L, m_3$



다중 서명

- **다중 서명(multi-signature)**: n 명의 서로 다른 서명자가 같은 메시지에 대해 서명하지만 그 결과가 n 개의 서명이 아니라 하나의 서명을 얻게 되는 서명 기법
- 따라서 n 개의 서명 대신에 하나의 서명을 확인하여 n 명이 서명한 사실을 확인할 수 있음
- Boldyreva의 pairing을 이용한 다중 서명 기법, 2003
 - 각 사용자의 개인키/공개키: $x_i \in_R \mathbb{Z}_q^*, Y_i = x_i P$
 - 다중 서명 공개키: $Y = \sum_i Y_i = (\sum_i x_i) P$
 - 메시지 m 에 대한 각 사용자의 서명: $S_i = x_i H(m)$
 - 다중 서명: $S = \sum_i S_i = (\sum_i x_i) H(m)$
 - 다중 서명에 대한 확인: $\hat{e}(P, S) = \hat{e}(Y, H(m))$
 - $G_1 = \langle P \rangle, G_2 = \langle g \rangle, |G_1| = |G_2| = q, H: \{0, 1\}^* \rightarrow G_1, \hat{e}: G_1 \times G_1 \rightarrow G_2$

결합 서명

- **결합 서명(aggregate signature)**: n 명의 서로 다른 서명자가 n 개의 서로 다른 메시지에 대해 서명하지만 그 결과가 n 개의 서명이 아니라 하나의 서명을 얻게 되는 서명 기법
- 따라서 n 개의 서명 대신에 하나의 서명을 확인하여 n 개의 서로 다른 메시지에 대한 n 명의 서명을 확인할 수 있음
- Boneh 등의 pairing을 이용한 결합 서명 기법, 2003
 - 각 사용자의 개인키/공개키: $x_i \in_R \mathbb{Z}_q^*, Y_i = x_i P$
 - 결합 서명 공개키: $Y = \sum_i Y_i = (\sum_i x_i) P$
 - 메시지 m_i 에 대한 각 사용자의 서명: $S_i = x_i H(m_i)$
 - 결합 서명: $S = \sum_i S_i = \sum_i x_i H(m_i)$
 - 결합 서명에 대한 확인: $\hat{e}(P, S) = \prod_{i=1}^n \hat{e}(Y_i, H(m_i))$
 - $G_1 = \langle P \rangle, G_2 = \langle g \rangle, |G_1| = |G_2| = q, H: \{0, 1\}^* \rightarrow G_1, \hat{e}: G_1 \times G_1 \rightarrow G_2$

- Boneh 등의 결합 서명은 결합하는 서명의 개수와 무관하게 결합 서명이나 개별 서명이나 그것의 크기는 같음
- 하지만 결합 서명을 확인할 때 n 개의 메시지와 n 개 공개키가 필요함

일괄 확인

- 일괄 서명은 서명자의 서명 비용을 줄이기 위한 수단임
- 일괄 확인은 확인자의 서명 확인 비용을 줄이기 위한 수단임
 - 앞 슬라이드의 결합서명은 일괄 확인 기능을 제공하는 서명임
- 일괄 확인
 - 각 서명자는 개별 서명을 생성함. 이들 서명을 개별적으로 확인 가능함
 - 확인자는 개별 서명들을 결합하여 하나의 결합서명을 만든 후에 일괄 확인함
 - **요구사항**. 유효하지 않은 서명이 일괄확인 집합에 포함되어 있을 때, 일괄 확인이 성공할 확률은 계산적으로 어려워야 함
- 문제점
 - 응용에 따라 다르지만 일괄 확인하였을 때 결과가 실패이면 어떤 서명(들) 때문에 실패하였는지 파악할 필요가 있을 수 있음
 - 무식한 방법: 개별 확인, 더 효율적인 방법: 분할 정복
- 위 문제점 때문에 보통 한 사용자가 모두 개별 확인한 서명을 결합하여 공개하면 나머지 사용자들은 매우 효율적으로 서명을 확인할 수 있도록 해주는 용도로 많이 사용함

전자서명과 프라이버시

- 전자서명은 인증을 목적으로 하기 때문에 본질적으로 서명한 주체가 명확하게 나타남
- 적법한 사용자가 서명한 것이지만 어떤 특정 사용자가 서명한 것인지 모르게 하고 싶을 수 있음
- 이것을 익명 인증 기술이라 하며, 다음과 같은 기법들이 제안되고 있음
 - **익명 ID/익명 인증서**
 - **그룹서명(group signature)**: 그룹을 대신하여 그룹 멤버가 익명으로 서명할 수 있도록 해줌
 - **링서명(ring signature)**: 자신을 숨길 수 있는 링을 구성하여 익명으로 서명할 수 있도록 해줌
 - **은닉서명(blind signature)**

익명인증서

- 전자서명은 공개키 기반 기술이며, 크게 인증서 기반과 신원 기반 방식이 있음
- 중앙집중 PKI의 경우 인증서의 주체 정보에 보통 사용자의 실명을 사용함
 - 실제로는 사용자의 실명 대신에 다양한 용도로 활용 가능함
 - 예) 실명 대신 역할 인증서: 한기대 교수 인증서
- 프라이버시를 위해 인증기관이 실명 대신에 익명으로 인증서를 발급할 수 있음
 - 문제점. 항상 같은 익명으로 서명할 경우에는 불연결성을 제공하지 못하므로 강한 프라이버시의 제공이 가능하지 않음
 - 해결책. 다수의 익명 인증서를 발급하여 사용함
 - 관리 및 저장공간의 문제, 주기적 재발급 문제 등이 있음
 - 조건부 익명성. 인증기관은 실명과 익명을 알고 있음

한기대 교수 인증서

방법 1. 모든 교수에게 다른 인증서를 발급 (일련번호와 특정 인증서가 매핑됨, 약한 프라이버시)

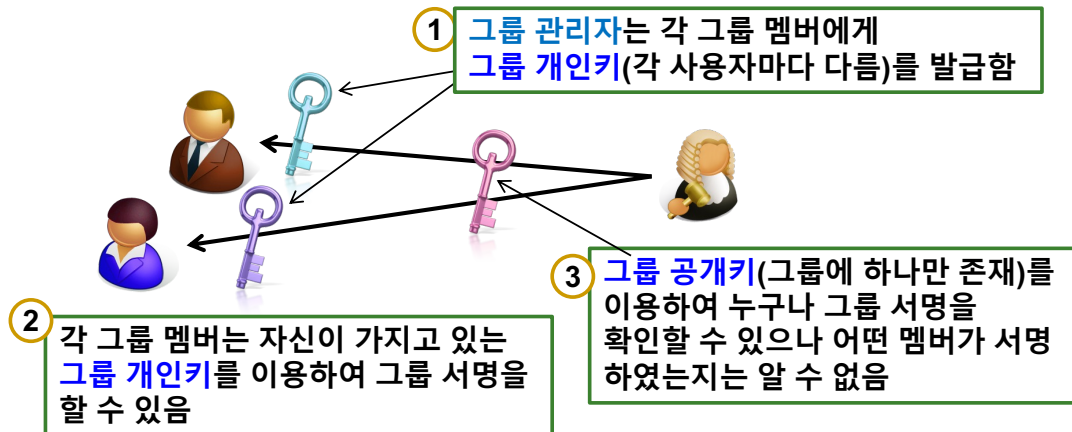
방법 2. 모든 교수에게 같은 인증서를 발급 (모두 동일 개인키를 사용, 강한 프라이버시)
다른 경로로 정보가 노출될 수 있음 (예: IP 주소 등)

익명ID

- 신원기반의 경우에는 잘 알려진 신원 대신에 익명ID로 개인키를 발급 받을 수 있음
- 이 경우 원래 신원기반 시스템의 추구한 목표와는 다른 형태가 됨
 - 원래 신원기반에서는 다른 사용자의 신원만 알고 있으면 그 사용자의 공개키를 생성하여 사용할 수 있음
 - 하지만 익명ID 방식의 경우에는 서명값과 함께 익명ID를 제시하여야 함
- 익명ID의 경우에도 익명인증서와 마찬가지로 불연결성 문제 때문에 다수의 익명ID의 사용이 필요할 수 있어 같은 문제점을 가지고 있음

그룹 서명 (1/3)

- 그룹 서명에는 그룹 관리자와 그룹 멤버들이 참여하며 다음과 같이 진행됨



그룹 서명 (2/3)

- **그룹 서명(group signature)의 요구사항**
 - **요구사항 1.** 그룹의 멤버만 서명을 할 수 있음
 - **요구사항 2.** 확인자는 그룹의 멤버 중 한 명이 서명하였다는 것을 확인할 수 있지만 누가 실제로 서명하였는지는 알 수 없음
 - **요구사항 3.** 두 개의 그룹 서명이 주어졌을 때 동일한 멤버가 서명한 것인지 알 수 없음 (불연결성)
 - **요구사항 4.** 그룹 멤버들이 공모하여도 다른 특정 그룹 멤버가 서명한 것으로 옳아말 수 없어야 함
 - **요구사항 5.** 분쟁이 발생한 경우에는 실제 서명자를 밝힐 수 있어야 함 (**조건부 익명성**)
- 요구사항 2, 3, 5 때문에 조건부 익명성을 제공하는 기법으로 많이 활용되고 있음

그룹 서명 (3/3)

- 그룹 개념이므로 그룹 멤버의 변경을 적절히 처리할 수 있어야 함
 - 특히, 그룹 멤버가 탈퇴하면 그 멤버는 더 이상 그룹 서명을 할 수 없어야 함
 - 보통 철회된 멤버를 제외한 다른 모든 멤버들에게 값을 전달해 주며, 이들은 이 값을 이용하여 자신의 그룹 개인키를 갱신함
 - 확장성이 부족하여 응용에 따라 효과적인 철회 방법이 될 수 없음
 - CRL을 사용하는 경우도 있지만 이 경우에도 CRL에 있는 철회된 사용자 수에 비례한 노력(공개키 연산)이 필요함
- 그룹 서명은 기존 일반 서명에 비해 상대적으로 계산 비용이 높거나 서명 값의 크기가 크다는 문제도 있음

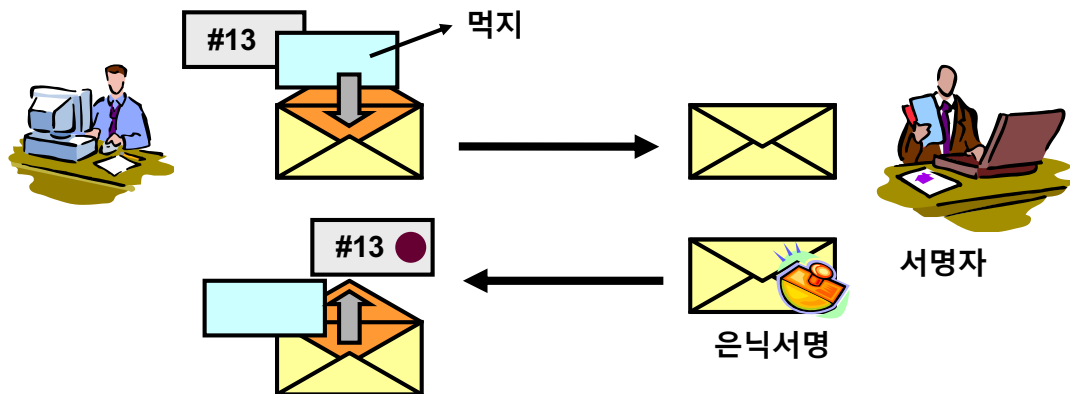
링 서명

- 사용자가 여러 개의 공개키로 링을 형성할 수 있으며, 자신의 개인키를 이용하여 서명하게 되지만 확인자는 형성된 링에 포함된 멤버 중 누가 서명하였는지를 알 수 없음
 - 그룹 서명과 유사하지만, 링은 보통 사용자가 **직접 만들기 때문에** 사용자마다 사용하는 링이 다르게 되지만 그룹 서명에서 그룹은 사용자마다 보통 동일함
 - 그룹 관리자가 필요 없다는 것이 장점
 - 링의 크기는 그룹의 크기보다 보통 작은 단위임
 - 링은 **항상 새롭게 형성**될 수 있으므로 링 서명을 전달할 때 링 정보를 함께 전달함 (dynamics of group)
 - 서명의 크기가 링의 크기에 비례한다고 할 수 있으며, 이것이 큰 단점임
- 링 서명도 익명성 보장이 필요할 때 사용할 수 있는 기법임
- 철회가능 링 서명도 있음 (조건부 익명성)
- Monero 암호화폐에서 활용하고 있음




은닉 서명

- 본질적으로 서명자는 자신이 서명하는 내용을 알고 있음. 하지만 은닉 서명 (blind signature)에서 서명자는 자신이 서명하는 내용을 전혀 알 수 없음
- 은닉서명은 주로 전자화폐, 전자선거 등에서 사용자의 익명성을 제공하기 위해 많이 사용함
 - 서명자의 익명성보다는 서명을 사용하는 사용자의 익명성에 초점을 두고 있음
- 은닉서명 프로토콜의 원리



은닉서명 - 계속

- RSA 기반 은닉서명 프로토콜



$$r \in_R \mathbb{Z}_n^*$$


$$\tilde{m} = H(m)r^e \bmod n$$

$$s = \tilde{s}/r \bmod n$$

$$H(m) \equiv s^e \pmod{n}$$

$$\begin{matrix} \tilde{m} & \rightarrow \\ \tilde{s} & \leftarrow \end{matrix}$$

$$\tilde{s} = \tilde{m}^d \bmod n$$

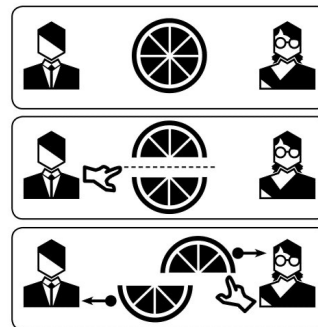


서명자

중요한 사실.
서명자 Alice는 은닉요소(blinding factor) r 를 알 수 없으면 나중에 s 를 보더라도 이 s 와 서명 순간을 연결할 수 없음

Cut-and-choose (1/2)

- 은닉서명은 서명자가 서명하는 내용을 볼 수 없으므로 위험할 수 있음
- 이런 위험을 줄이기 위해 사용할 수 있는 대표적 기법이 cut-and-choose임
 - Cut-and-choose
 - Alice는 케이크를 두 조각으로 나눈 후, 각 조각을 동일한 크기의 박스에 포장함
 - Bob은 두 박스 중 하나를 선택함
 - 선택되지 않은 박스는 Alice가 가지게 됨
 - Alice는 나중에 Bob이 어떤 박스를 선택할지 모르기 때문에 공정하게 나누지 않으면 본인이 손해를 보게 됨

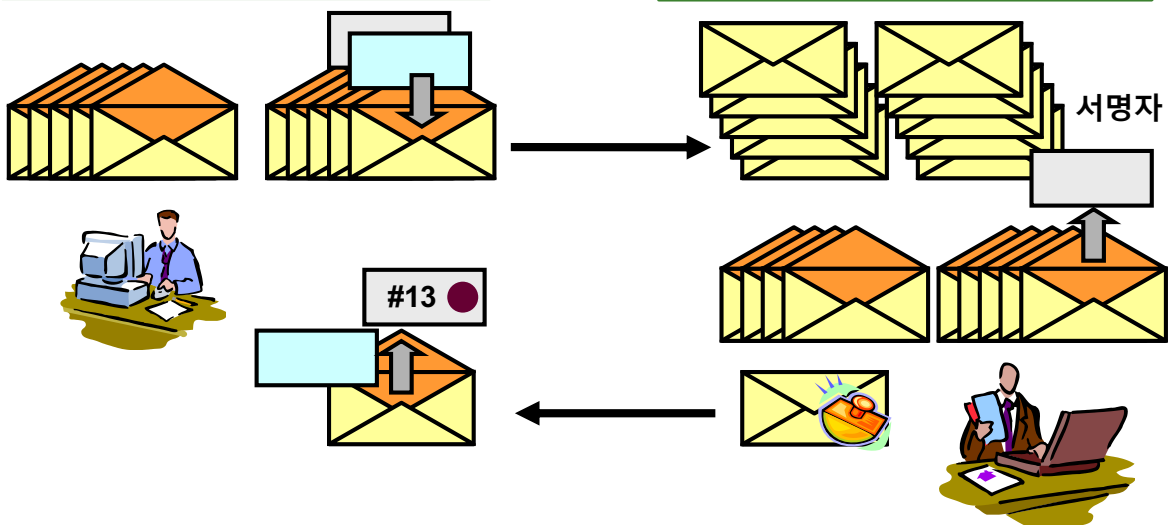


Cut-and-choose (2/2)

- 은닉서명과 cut-and-choose

각 봉투에 모두 다른 랜덤값을 포함

한 개를 제외한 나머지 봉투는 모두 개봉하여 올바르게 구성되어 있는지 내용을 확인

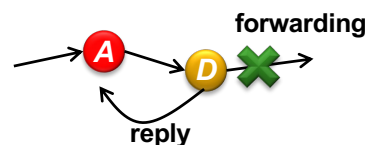


익명 통신 (1/2)

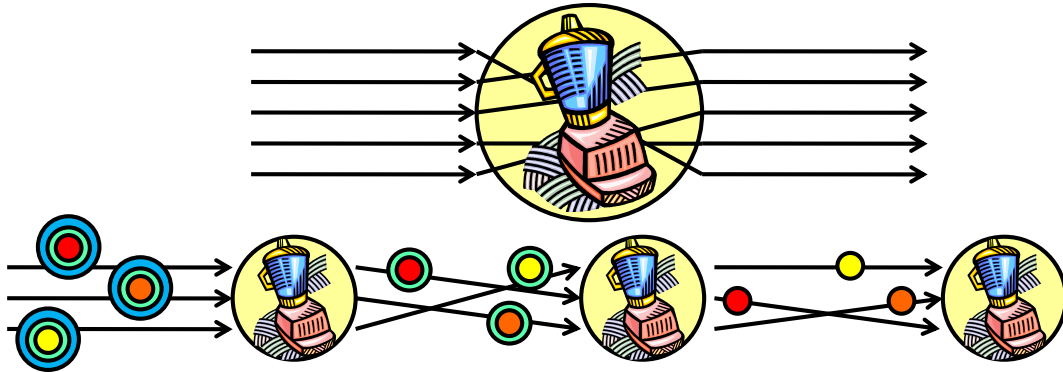
- 익명 통신: 트래픽 분석을 통해 교환되는 메시지의 소스 노드와 목적 노드를 식별할 수 없도록 메시지를 교환하는 통신 방식
- 보통 통신은 노드를 특정할 수 있는 식별자(IP주소, MAC주소)를 사용함
 - 익명 통신에서는 이들을 사용하더라도 이를 통해 소스 노드나 목적 노드를 식별할 수 없어야 함
- 요구사항
 - 노드의 관찰(수신/전송하는 메시지의 관찰)을 통해 소스 노드나 목적 노드를 식별할 수 없어야 함
 - 메시지 내용, 메시지 크기, 메시지 송수신 시점에 의해 식별할 수 없어야 함
 - 사용하는 네트워크 환경에 영향을 받음

익명 통신 (2/2)

- 메시지 내용에 의한 식별. 보통 메시지 내용뿐만 아니라 헤더도 암호화함
- 메시지 크기에 의한 식별. 항상 채우기를 통해 일정한 크기의 메시지를 교환함
- 메시지 송수신 시점에 의한 식별
 - 소스노드 익명성: 수신된 메시지가 이전 노드에서 시작된 것인지 중계된 것인지 구분할 수 없어야 함
 - 목적노드 익명성: 중계의 중단이나 응답 메시지를 보내는 행위 때문에 노출될 수 있음
 - 불필요한 메시지의 송신이나 지연 중계가 필요함



- 믹스넷: 여러 겹으로 암호화된 데이터를 섞어 트래픽 분석을 할 수 없도록 함

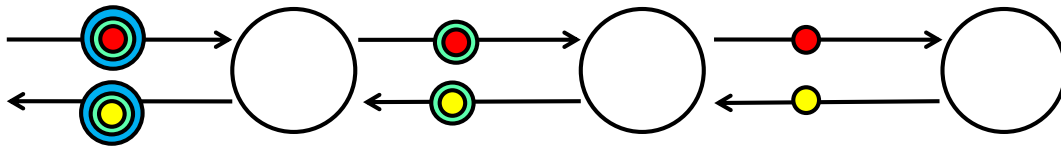


- 시간을 이용하여 수신된 메시지와 중계된 메시지를 연결할 수 있기 때문에 메시지를 즉시 전송하지 않고 버퍼에 유지한 후 충분한 수의 메시지가 확보되면 전송함
 - 불가피한 지연 발생
- 메시지 크기를 이용한 연결은 채우기를 이용하여 숨김

양파 라우팅 (1/3)

- 양파 라우팅
 - 소스 노드와 목적 노드 사이에 익명 통신, 비밀 통신 제공
 - 꼬리 노드 기반: 실제 통신은 소스 노드가 지정한 꼬리 노드에 의해 이루어짐
- 두 단계 통신: 양파 경로 설정, 데이터 전달
- 양파 경로 설정
 - 소스 노드가 데이터를 전달할 때 사용할 중계 노드(양파 라우터)의 수와 중계 순서를 결정
 - 여러 겹으로 암호화하여 전달하며, 각 중계 노드는 바깥층을 제거하여 중계함
 - 데이터 전달할 때 사용할 키를 확립함
 - 소스 노드 대신 프록시 노드가 양파 경로를 설정할 수 있음
- 헤더 정보, 메시지 크기, 메시지 전송 시점에 의한 노출을 막기 위해
 - 항상 정해진 크기의 데이터 교환
 - 중계 노드 간의 비밀 통신을 하며, 중계 노드는 랜덤 채우기, 믹스넷 역할 수행

양파 라우팅 (2/3)



● 양파 경로 설정

- 중계 노드는 공개키 쌍을 가지고 있음 (TLS 인증서)
- 소스 노드 S는 X, Y, Z(꼬리노드)를 중간 노드로 사용하기로 결정함
 - $O_Z = \{ \lfloor K_Z \rfloor \}. + K_Z$
 - $O_Y = \{ \lfloor K_Y \rfloor O_Z \}. + K_Y$
 - $O_X = \{ \lfloor K_X \rfloor O_Y \}. + K_X$
- 확립된 키로부터 실제 사용할 키를 계산함
 - 예) $K_{FZ} || K'_{FZ} || K_{BZ} || K'_{BZ} = \text{KDF}(K_Z)$

양파 라우팅 (2/2)

● 데이터 교환

- 소스에서 목적: O_X 를 만들어 시작 중계 노드에 전달함
 - $O_Z = \lfloor C_Z = \{M\}. K_{FZ}, MAC. K'_{FZ}(C_Z) \rfloor$
 - $O_Y = \lfloor Z, C_Y = \{O_Z\}. K_{FY}, MAC. K'_{FY}(C_Y) \rfloor$
 - $O_X = \lfloor Y, C_X = \{O_Y\}. K_{FX}, MAC. K'_{FX}(C_X) \rfloor$
- 목적에서 소스
 - $Z \rightarrow Y: O_Z = \lfloor C_Z = \{M\}. K_{BZ}, MAC. K'_{BZ}(C_Z) \rfloor$
 - $Y \rightarrow X: O_Y = \lfloor Z, C_Y = \{O_Z\}. K_{BY}, MAC. K'_{BY}(C_Y) \rfloor$
 - $X \rightarrow S: O_X = \lfloor Y, C_X = \{O_Y\}. K_{BX}, MAC. K'_{BX}(C_X) \rfloor$

Z와 목적 노드 D 구간도 암호화하고 싶으면 M으로 시작하는 것이 아니라 $C = \{M\}. K_{SD}, MAC. K'_{SD}(C)$ 로 시작해야 함