Orchid: 분산 네트워크 형성 및 확률론적 소액 결제 활성화

David L. Salamon, Gustav Simonsson, Jay Freeman, Brian J. Fox 및 Brian Vohaska 와 Stephen F. Bell 및 Steven Waterhouse, Ph.D.

2018 년 5 월 28 일
버전 0.9.2

초록

브라우징을 검열하고 사적인 브라우징 정보를 탐색하는 방법이 점점 효율적이 되면서 익명화 방법에 대한 관심도 높아졌습니다. 안타깝게도, I2P 및 Tor 와 같은 제한되지 않는 비감시 인터넷 액세스에 대한 기존의 접근 방식은 널리 채택되지 않아 어려움이 있습니다. 사실, 몇천 명에 불과한 무급 자원자들이 릴레이와 출구 노드를 호스팅하기 때문에 정교한 공격자들이 모니터링하거나 다른 방식으로 침해할 수 있는 노드의 수가 다루기 쉬울 정도가 됩니다. 참가자에게 직접적으로 동기를 부여함으로써 이러한 릴레이 및 출구 노드의 부족 문제를 해결한다고 생각되는 "대역폭 마이닝"에 기반을 둔 시장에 기반하고 완전히 분산된 익명의 피어투피어 시스템을 소개합니다.

본 백서는 아직도 개발 중인 시스템에 대해 설명할 목적으로 작성되었습니다. 따라서 발생하는 구현 차이를 해결할 수 있도록 콘텐츠를 변경하고 새로운 콘텐츠를 추가할 것이라는 점은 확실합니다. 라이브러리 구성 요소와 구체적인 암호화 알고리즘 사용에 있어서 유연성이 있습니다. 하지만 시스템의 기본 사항, 목적 및 목표는 동일하게 유지될 것입니다.

기고문에는 다음이 포함됩니다.

- 패킷의 명령에 따른 거래 비용의 블록체인 기반의 스토캐스틱 결제 메커니즘
- 대역폭 판매를 위한 상품 사양
- Eclipse 공격을 무작위로 어렵게 만드는 피어투피어 시스템의 분산 유도성 증명 방법
- 공격자가 공격의 일환으로 입찰을 변경하는 상황에서 대역폭을 판매하는 데 적합한 효율적인 보안 강화 경매 메커니즘
- 완전히 분산된 익명 대역폭 시장

목차

1. 서론

Orchid Protocol 은 대역폭 판매자를 Orchid Market 라는 용어의 구조화된 피어투피어(P2P) 네트워크로 조직합니다. 고객은 Orchid Market 에 연결하고 대역폭 판매자들에게 결제를 하여 인터넷에서 구체적인 리소스로 *프록시 체인*을 구성합니다.

Orchid Market 의 프록시 체인은 글로벌 인터넷에서 데이터를 전송하고 수신하기 위한 좀 더 일반적인 방법과는 달리, 당연히 데이터 소스에 대한 정보를 대상 위치에 대한 정보에서 분리합니다. 단일 릴레이 또는 프록시가 두 가지 정보를 모두 보유하거나 그렇게 하는 사람의 신원을 파악합니다. Orchid Market 의 구조는 공모 공격에 대한 강력한 저항성, 즉 일단의 대역폭 판매자들이 이러한 지식 분리 문제를 극복할 수 있는 능력을 제공함으로써 이러한 정보의 분리를 더욱 지원합니다.

프록시 체인 참여자의 역할은 다음과 같습니다.

- *소스 노드 또는 고객* — 거래를 개시한 참여자.
- *릴레이 노드* — 네트워크 트래픽을 전달하는 중개 참여자.
- *프록시 또는 출구 노드* — 요청된 글로벌 인터넷 사이트로 연결하는 참여자.
- **emphbandwidth 판매자** — 릴레이 또는 프록시.

Orchid Market 은 소스와 대상 위치 지식을 분류하는 글로벌 인터넷으로부터 데이터를 전송하고 수신하기 위한 덜 일반적인 방법과는 달리, 트래픽 분석을 방지하기 위한 고정 속도 릴레이와 정보의 숨기기 또는 검색과 관련되지 않은 참여 유도책, 즉 토큰 결제를 제공합니다.

시스템의 세부 정보를 설명하기 전에, 이것으로 해결할 수 있는 핵심 문제와 우리 시스템의 기초를 위해 선택한 일반적인 솔루션을 간단히 검토하겠습니다.

트래픽 분석 문제

문제 진술: 수학자들이 가득한 구내식당에서 이들이 여러분의 친구에게 아무도 몰래 메시지를 전송하려 하는 상황을 상상해 보십시오. 아직 메시지 통과 프로토콜을 협상하지 않았기 때문에 모든 구현 세부 정보는 같은 공간에 있는 모든 사람들에게 공개적으로 진술되어야 합니다. 어떻게 할 수 있습니까?

이 문제에 대한 특히 섬세한 솔루션은 1981 년[56]에 Chaum 이 제안한 대로 모든 사람이 릴레이와 수신자 역할을 하는 것입니다. 이 계획에서 참여자들은 "봉투를 넣은 봉투"에 해당하는 디지털 메시지인 암호화된 메시지를 작성합니다. Alice 에게 메시지를 보내려면

$$Enc(\text{"ToBob"} \mid\mid Enc(\text{"ToAlice"} \mid\mid Enc(\text{메시지}, Alice), Bob), Carol)$$

을 계산하고 그 메시지를 Carol 에 전송하면 Carol 이 이것을 복호화하여 Bob 에게 전송하고, Bob 은 그것을 복호화하여 Alice 에게 전송합니다. 트래픽 분석을 방지하기 위해 모든 사람은 모든 주기에 고정된 수의 메시지를 전송합니다. 회신 주소를 처리하기 위해, Bob 과 Carol 에게 고유 메시지 식별자를 기억하고 체인에 따라 메시지를 다시 전송하게 할 수 있습니다.

위 방법을 사용하는 시스템에 특히 중요한 것은 공모의 가능성입니다. Bob 과 Carol 이 협조한다면 특정 메시지를 누가 전송했는지와 누구에게 전송했는지 확인할 가능성이 있습니다.

시빌 문제

위와 같은 구내식당 문제 진술은 *시빌* 공격, 즉 하나의 참여자가 무작위의 다수 사용자인 척하는 상황을 방지하기 위해 물리적인 개체를 사용했습니다. 아쉽지만 디지털 시스템에서는 이러한 접근 방식을 사용할 수 없습니다.

문제 진술: 완전히 디지털인 맥락에서 어떤 사람이 "실제" 사람인지 어떻게 알 수 있습니까?

이러한 문제에 대한 솔루션은 해시캐시[85]에서 찾을 수 있습니다. "실제" 사람이라고 주장하는 사람들에게 계산 리소스를 소비하라고 요구하면 시빌 공격자를 믿을 수 없는 수의 네트워크 참여자라는 주장을 하려면 믿을 수 없는 양의 계산 리소스를 실제로 소유할 것을 요구하는 입장에 놓이게 할 수 있습니다.

무작위 선택 문제

위 구내식당 문제 진술에서는 시스템 사용자 둘 중 하나에게 메시지를 전송하기 위한 쉬운 방법을 가정했습니다(예: 구내식당 전체에 소리 지르기). *공모 공격*에 최대로 저항하는 Chaumian 의 혼합을 구현하려면 "실제"인 릴레이로부터 무작위로 선택할 수 있어야 합니다. 순수하게 생각하면 어떤 사람이 네트워크에 들어오거나 네트워크에서 나갈 때마다 알림이 필요합니다. 안타깝지만 실제 P2P 네트워크에서는 사용자 둘 중 하나가 그러한 목록을 유지하게 하면 허용할 수 없는 양의 네트워크 트래픽(*O(n²)*개의 알림)을 일으키게 됩니다.

문제 진술: 현재 모두 "실제"인 릴레이의 분산 목록을 유지하여 네트워킹 오버헤드를 최소화하고 효율적인 무작위 피어 선택을 지원할 수 있는 방법은 무엇입니까?

이 문제에 대한 특히 섬세한 솔루션은 Chord[85] 분산 해시 표(DHT)에서 확인할 수 있습니다. 이 계획에서는 피어들이 큰 공간에서 고유 주소를 할당 받은 다음 *O(로그(n))* 시간으로 조회를 수행할 수 있는 방법으로 연결됩니다. 사용자를 추가하거나 제거해도 *O(로그(n))*명의 피어에게만 알리면 됩니다.

시스템 개요

Orchid Protocol 의 핵심은 위 솔루션이 결합된 것입니다. 우리의 접근 방식에서는 피어들이 자신의 "실제성"을 증명하기 위해서는 *메달리온*을 제시해야 하고 그 다음 *Orchid Market* 이라는 용어의 분산 P2P 네트워크로 조직됩니다. Orchid Market 참여자들의 정직성을 유지하기 위해, 모든 피어가 이웃의 행동이 올바른지 확인합니다. 그러면 고객은 Orchid Market 을 사용하여 Chaumian 의 메시지 전달을 위한 무작위 피어를 선택합니다. 참여의 동기를 부여하기 위해, Orchid Market 은 전달된 바이트당으로 릴레이와 프록시에 대해 결제하게 합니다.

이것은 단순한 생각이지만 물론 악마는 디테일에 있습니다. 시스템은 완전히 분산되고, 완전히 자율적이고, 완전히 익명이고, 결제를 취급해야 합니다. 따라서 이 설계 문서 대부분은 고객 보안, 시스템 성능 및 시스템의 경제적 건전성에 대한 공격을 방지하는 것을 중심으로 하고 있습니다. 공격 분석이 중요하고 우리의 시간을 대부분 차지하겠지만 궁극적으로는 시장이 운영되는 맥락에 대한 필요한 효과 이상의 것이 될 수 없습니다. "숲속에서 헤매고" 있는 상황이라면 위의 해설을 북극성으로 사용하시길 바랍니다. 시스템의 설계 세부 정보는 위 세 가지 문제가 결합된 상황에 대한 실제적인 솔루션을 실현하기 위한 목적을 향해 있습니다.

2. 대체 접근 방식

보호되지 않은 인터넷 액세스

보호 없이 인터넷에 액세스하는 사용자가 ISP 에 전체 브라우징 기록과 웹사이트 사용을 제공하면 ISP 는 그 데이터를 공유하거나 판매할 수 있습니다.

VPN(가상 사설 네트워킹) 서비스

VPN(가상 사설 네트워크)은 암호화를 사용하여 VPN 가입자의 트래픽을 더 광범위한 미보안 네트워크를 통해 안전하게 전송합니다. VPN 이 트래픽을 수신한 후에는 복호화하여 다른 더 광범위한 보호되지 않은 네트워크를 통해 재전송합니다. 재전송은 사용자가 웹 사이트에서 부과하는 액세스 제한을 우회하도록 돕고, 부차적으로는 웹 사이트 브라우징 습관의 추적을 줄입니다. 암호화는 사용자의 ISP 가 트래픽을 보지 못하도록 하여 모니터링 공격을 방지합니다. 이는 VPN 을 사용자를 위한 새로운 ISP 로 만드는 방식으로 수행됩니다. ISP 가 이전에 수행할 수 있었던 모든 공격을 이제 VPN 공급자가 쉽게 수행할 수 있습니다.

VPN 사용자는 VPN 제공업체를 신뢰할 수 있다고 생각해서는 안 됩니다. VPN 서비스 제공업체는 ISP 보다 경쟁이 치열하지만 궁극적으로 동일한 소스에서 인재를 영입하고 유사한 대역폭의 현금형 비즈니스 모델을 갖습니다. VPN 제공업체가 동일한 인센티브에 노출되어 않아 사용자가 ISP 를 신뢰하지 않을 가능성은 거의 없습니다. 또한 VPN 설정에서 트래픽을 중계하기 위해 IP 주소를 재사용하면 상용 웹 사이트에서 사용을 차단하는 것이 상대적으로 쉬워집니다[13].

Tor

Tor[60]은 폭넓은 대중에게 *양파 라우팅*의 개념을 소개한 것으로 유명한 무료 소프트웨어 프로젝트입니다. 이 시스템에서 사용자는 릴레이 및 종료 노드의 글로벌 목록을 다운로드하고 해당 목록에서 임의로 선택하여 그 선택으로부터 *양파 경로*를 형성합니다. 양파 경로는 순서가 지정된 릴레이 목록입니다. 양파 경로를 따라 전송된 패킷은 각 피어에 대해 차례로 암호화되어엑시트 노드가 이해할 수 있도록 패킷 경로를 각 노드가 수신해야 합니다. 그 결과 동일한 사용자가 여러 노드를 손상시키거나 실행하지 않는 한, 두 릴레이가 패킷을 보낸 사람이 누구인지, 패킷이 어디로 갔는지 모두 알 수 없습니다.

3. 공격

Orchid Protocol 의 대부분은 공격 방지를 위해 설계되었으므로 P2P 네트워크에 특히 일반적인 공격에 대한 문헌을 검토하는 것으로 시작하겠습니다.

추론 공격

Orchid Protocol 이 방어해야 하는 가장 큰 등급의 공격은 사용자에 대한 정보를 나타내는 것입니다. Orchid 는 기존 인터넷에서 오버레이로 구현되기 때문에 일부 정보는 *불가피하게* 일부 피어와 공유했습니다. 또한, Orchid 의 기본 결제 시스템은 ERC20 토큰을 사용하므로 이더리움 네트워크에서도 마찬가지로 일부 거래 정보를 사용할 수 있습니다. 아래 목록에서 이러한 정보는 "*" 로 표시됩니다. 이 문서에서 *불가피하게* 공유되는 것으로 구체적으로 명시되지 않았지만 해당 정보가 *정보 공격*으로 명명되었음을 밝히기 위한 방법이 발견되었고 Orchid 의 화이트 햇 버그 바운티의 포상금 지급 대상인 모든 정보. 공유 대상에 대한 자세한 내용은 섹션 7 의 프로토콜 사양과 섹션 B.1 의 담합에 대한 논의 및 네트워크의 참조 구현[1]을 참조하십시오.

공격자가 관심을 가질 것으로 추정되는 데이터 유형(타임리스):

- 실제 신원 정보. 사용자의 이름, 주민등록번호, 주소 등.
- 웹 사이트 계정 정보. 특정 웹 사이트의 사용자 계정. 실제 신원 정보와 다를 수 있습니다.
- *IP 정보. 사용자가 Orchid 네트워크에 액세스하는 IP 주소. 일부 사용 시나리오에서는 실제 신원 정보를 학습하는 것과 동일할 수 있습니다.
- *이더리움 정보. 사용자의 지갑과 관련된 키(*공개 또는 개인). 일부 사용 시나리오에서는 실제 신원 정보를 학습하는 것과 동일할 수 있습니다

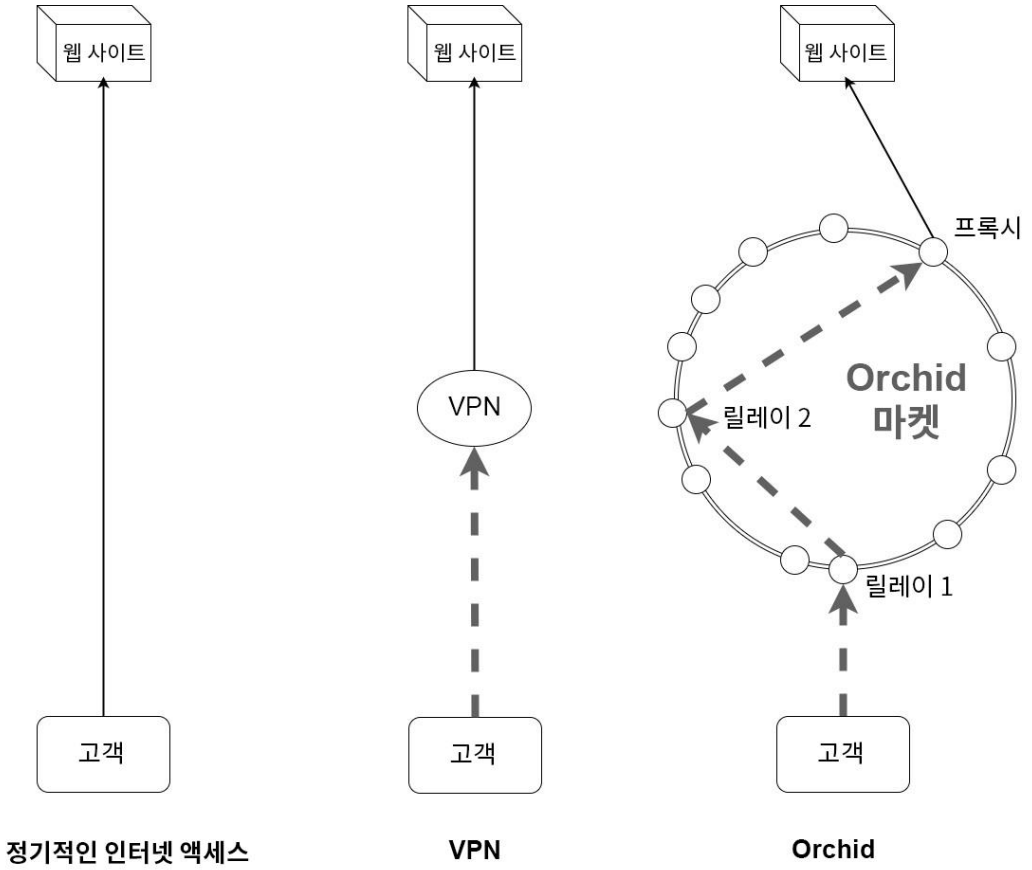

그림 1: 직접 연결, VPN 연결, Orchid 연결

- *Orchid 네트워크 정보. Orchid 네트워크에서 노드의 현재 비즈니스와 연관된 키(*공용 또는 개인).

공격자가 노릴 것으로 예상되는 행동 정보 유형(시간 및 체인 관련 데이터):

- *고객 식별. 공격자가 고객의 IP 주소를 알아냅니다.
- *릴레이 식별. 공격자가 릴레이의 IP 주소를 알아냅니다.
- *프록시 식별. 공격자가 프록시의 IP 주소를 알아냅니다.
- *링크 식별. 공격자가 두 개의 IP 주소가 체인에 사용되었음을 알아냅니다.
- *웹 사이트 액세스. 공격자가 Orchid 네트워크에서 특정 웹 사이트로 아웃바운드 연결이 이루어졌음을 알아냅니다.
- *웹 서버 액세스. 공격자가 Orchid 네트워크에서 (여러 웹 사이트를 호스팅할 수 있는) 특정

웹 서버로 아웃바운드 연결이 이루어졌음을 알아냅니다.

- *이더리움 링크. 공격자가 이더리움 공개 키를 Orchid 사용자가 보유하고 있음을 알아냅니다.
- *구매 링크. 공격자가 두 거래가 공통된 지불자를 공유한다는 사실을 알아냅니다.
- *구매 정보. 공격자가 체인을 통해 전송되는 대역폭의 양 및 타이밍을 알아냅니다.

위의 모든 동작 정보는 정상적인 작동 중에 Orchid Network 의 다른 노드와 공유되지만 아래 설명과 같이 대부분의 상황에서는 공격자가 한 번에 여러 정보를 알아낼 수 있는 경우 **동작 정보 수집**에 의해 사용자가 직접 피해를 입는 것으로만 가정합니다. 예를 들어, 사용자 X 가 웹 사이트 Y 에 액세스했다고 가정할 때 공격자는 다음을 필요로 합니다. 구매자 신원, 웹 사이트 액세스 정보 및 몇 가지 링크 식별. 이러한 이유로, 참조 사양을 따르는 피어는 고객이 구매한 서비스를 제공하는 데 필요한 경우를 제외하고 위의 정보를 저장하거나 공유하지 않습니다.

시스템 동작의 통계적 모델링에서 나온 탈익명화를 추론 공격 또는 모니터링 공격이라고 합니다. 이들은 종종 신중하게 제작되거나 시간을 맞춘 요청과 같은 프로빙 이동 또는 네트워크의 특정 피어 오프를 DoS 로 연결하고 트래픽 패턴이 어떻게 반응하는지 관찰하는 것과 같은 다른 공격과 결합됩니다.

- SSL 암호화 웹 트래픽에서 최종 사용자의 의학적 질병, 가족 소득 및 투자 선택 유추[57].
- 글로벌 트래픽 로그에서 Tor, I2P 및 Orchid 트래픽을 탈익명화[59].
- 타이밍 분석을 통해 OpenSSL 서버의 공개 키 확보[54].

경제적 공격

유사한 시스템과 달리, Orchid Protocol 은 결제 메커니즘에 대한 공격과도 관련이 있어야 합니다. 이 백서에 사용된 분류는 다음과 같습니다.

1. **경제적 악용**. 무료 샘플 대역폭을 제공하는 사용자와 같은 수익성 없는 바람직하지 않은 동작은 사용자가 무료 샘플 대역폭을 독점적으로 사용할 수 있게 합니다.
2. **경제적 서비스 거부(EDoS)**. 결제를 사용하여 구매로 Orchid Network 의 다른 노드를 압도하여 오프라인 상태로 만듭니다.

시빌 공격

다중 사용자인 것처럼 가장하는 악의적 행동을 다중 인격 장애를 겪는 환자에 비유하여 시빌 공격이라고 합니다. 이러한 유형의 공격 활동에는 다음이 포함됩니다.

- Yelp, Amazon 등에 여러 건의 리뷰 제출
- 여러 명의 침입자인 척하여 BitTorrent 에서 더 빠른 다운로드 속도를 유도[74].

Eclipse 공격

Eclipse 공격에서 공격자의 목표는 자체로부터 시스템의 일부를 숨기는 것입니다. 사용되는 방법은 일반적으로 권한 상승 공격과 동등한 네트워크입니다. 네트워크를 더 많이 제어하는 네트워크 위치를 제어한 다음 해당 제어를 사용하여 더 많은 제어를 확보합니다.

- 비트코인 채굴 P2P 네트워크를 세분하여 공격자가 컴퓨팅 파워의 51% 미만을 실질적으로 제어할 때 소위 51% 공격 을 허용합니다[65].
- 자석 링크와 관련된 주소 공간을 인계하여 BitTorrent DHT 에서 파일을 제거합니다[87].

중간자 공격

두 상호 작용 당사자 간에 자신을 끼워 넣은 후에만 수행할 수 있는 조치를 총칭하여 중간자(man-in-the-middle) 공격이라고 합니다. 메타 데이터 분석을 위해 암호화된 정보를 기록할 수 있지만(섹션 3), 암호화되지 않은 데이터는 동작을 제어하기 위해 추가로 변경될 수 있습니다. 키 교환이 보안되지 않은 경우 중간자(man-in-the-middle)는 공격자의 키가 상대방의 키라고 잘못 믿도록 두 당사자를 속일 수도 있습니다.

서비스 품질 공격

어떤 공격자는 Orchid Network 사용자의 평균적인 시스템 성능을 느려지게 하여 이용률이 감소하는 상황을 목표로 할 수 있습니다.

서비스 거부 공격

특정 리소스를 오프라인으로 전환하는 데 중점을 둔 공격을 서비스 거부 공격(DoS)이라고 합니다. 예기치 않은 상황에서 시스템 동작이 잘못 지정되고 테스트되는 경우가 종종 있습니다. DoS 공격은 P2P 네트워크에서 노드를 탈익명화하는 데 유용합니다. 주목할 만한 예:

- Tor 트래픽을 탈익명화하기 위해 시빌 공격 기반 모니터링과 함께 사용되는 표적화된 DoS 공격[52].
- I2P 의 floodfill 데이터베이스를 완벽하게 제어하기 위한 DoS 오프라인화 공격은 20 개의 시빌 노드만으로도 충분하기 때문에 네트워크의 모든 트래픽을 탈익명화할 수 있습니다[64].

해킹

집요한 공격자라면, 과거 경험에 비추어 신뢰할 수 있는 동료를 공격 경로로 변환함으로써 네트워크의 노드를 직접 손상시킬 수 있습니다. 체인을 사용하여 대역폭을 배포하면 반복적인 해킹을 통해 결국 공격자가 연결을 역추적 할 수 있습니다. 이러한 공격은 보안에 중요한 영향을 주지만 Orchid Network 의 범위를 벗어납니다. Orchid Network 의 설계 목표를 달성한 후에는 이것이 시스템 사용자에 대한 주요 공격 방식이 될 것입니다.

4. Orchid Market

Orchid Market 은 Orchid Protocol 이 구축되는 기초입니다. 기본적으로 이는 릴레이, 프록시 및 사용자 간의 대역폭 구매 및 판매를 용이하게 하는 분산형 P2P 네트워크입니다. 우리가 메달리온이라고 부르는 작업 증명을 제시함으로써 시장에 진입하고 계속 참가할 수 있습니다. Orchid Market 의 네트워크 구조는 DHT(분산 해시 표, Distributed Hash Table)와 유사한 구조이며 수정된 Chord 의 확장으로 생각할 수 있습니다[83, 85].

4.1. 기본 시장 운영

높은 수준에서 Orchid Market 이 제공하는 운영은 다음과 같습니다.

- 페들러가 Orchid Market 에 참여하는 방법.
- 페들러에게 판매할 서비스를 요청하는 방법.
- 컴퓨팅 자원에 의해 무작위 가중치를 부여 받는 모든 피어의 하위 집합을 선택하는 방법. 여기서 *조회 속성*은 다음과 같습니다.

$$\textit{조회}(\textit{무작위 주소}) \Longrightarrow \textit{무작위}(\textit{페들러})$$

이때 페들러는 서명된 라우팅 테이블이라고 부르는 핑거 테이블 아날로그를 통해 그 가까운 이웃에 대한 정보를 추적하는 Chord 의 노드로 여겨질 수 있습니다. Orchid Market 에서 페들러는 릴레이 또는 프록시일 수도 있는 대역폭의 구매자 및 판매자 역할을 합니다. Orchid Market 내에서 사용자는 페들러가 아니어도 되지만 모든 릴레이 및 프록시는 페들러가 되어야 합니다. *조회 속성*의 중요성은 고객이 이를 통해, a 명의 공격자가 있는 n 명의 페들러 집합으로부터 무작위로 선택된 페들러인 경우 무작위 페들러는 다음의 확률을 지닌 공격자가 아니라는 사실을 알 수 있다는 사실에 있습니다.

$$P(Attacker|random(Peddler)) = 1 - \frac{a}{n}$$

Orchid 백서[90] 섹션에서는 이 속성이 Eclipse 및 기타 공격에 대한 보호 기능을 제공함을 설명합니다. 이러한 작업을 구현하기 위해 Orchid Market 은 키와 값이 없는 DHT 의 구조를 취합니다. 무작위 선택을 수행하기 위해 사용자는 임의의 주소를 생성하고 해당 지점에 가장 가까운 페들러를 찾습니다. Orchid Market 은 2^{256} 의 순서를 지닌 Chord 와 같은 링으로 표현되기 때문에, 모든 임의의 주소는 $\{1, 0\}^{256}$ 의 무작위적인 정수로서 선택되어야 합니다.

4.2. 기본적인 페들러 작업

Orchid Market 에서 페들러가 지원하는 작업은 다음과 같습니다.

- *목록 서비스*. 페들러에게 판매하는 서비스의 목록을 요청합니다.
- *라우팅 테이블 및 메달리온 받기*. 이렇게 하면 페들러의 메달리온, 서명된 라우팅 테이블 및 라우팅 테이블 구성원에게 트래픽을 릴레이하는 비용이 반환됩니다.
- *릴레이 트래픽*. 페들러에게 결제하여 라우팅 테이블의 피어 중 하나에게 트래픽을 전달하게 합니다.
- *구매 서비스*. 페들러를 서비스 제공업체로 고용합니다.

메달리온은 Orchid Market 의 작업 증명에 대한 토큰이자 시장의 *참여 라이선스*입니다. 메달리온이 없는 페들러는 현재의 이더리움 블록 다이제스트의 TTL 순서에 따라 조만간 시장에서 제거됩니다. 서명된 라우팅 테이블은 Orchid 백서에서 자세히 설명합니다[90]. 이 중 첫 번째 2 개는 고객이 관심 있는 페들러를 탐색하는 데 사용되며 두 번째 2 개는 페들러가 발견된 후 서비스 구매를 협상하는 데 사용됩니다.

체인에서 사용되는 것과 유사한 Orchid Market 을 통한 탐색. 고객은 일부 알려진 페들러(부트스트랩을 통해 발견, 4.6 참조)에 연결하고 라우팅 테이블을 검사하고 결제하여 선택한 지점에 가장 가까운 페들러로 트래픽을 전달하게 합니다. 라우팅 테이블에 관한 섹션을 통해 알 수 있듯이 이를 통해 고객은 IP 주소를 비밀로 유지하면서도 $O(log^2 n)$ 패킷의 페들러에 대한 비교적 효율적인 무작위 액세스를 제공할 수 있습니다.

대역폭이 필요한 Orchid Market 의 모든 운영에는 다른 엔터티와 동일한 대역폭 비용이 적용됩니다. 이러한 비용은 각 페들러가 최소한 대역폭을 구매할 때마다 시장 운영으로 인해 대역폭을 판매함에 따라 각 페들러에게 최소화됩니다. 페들러에 대한 이 대역폭 비용은 부록 E 에 언급된 공격을 방지합니다.

페들러는 수정된 Chord DHT 에서 사용된 것과 동일한 체계를 사용하여 Orchid Protocol 에서 연결됩니다. 우리는 보다 성숙된 문헌과 기계 확인 정확성 증명의 존재로 인해 Kademlia 보다는 Chord 를 선택했습니다[83].

페들러 주소의 집합은 2^{256} 크기의 chord 링의 정수로 표시됩니다. 여기서 피어의 주소 a와 b 사이의 거리 d는 다음 공식으로 정의됩니다.

$$a, b, d \in (0, 2^{256})$$

$$a + d \equiv b \pmod{2^{256}}$$

A 를 Orchid Market의 페들러 컬렉션으로, 를 특정 페들러라고 가정하겠습니다. Chord에서 모든 노드에 대해 예상되는 최대 피어 수는 $\log_2(n)$입니다. 그렇다면 e에 대한 강제 연결 집합은 다음과 같이 정의됩니다.

$$L = \{f : \min_{log_2(n)}\{dist(e+t, f)\}\}$$

여기서 $t \in \{1, 2, 4, ..2^{255}\}$는 어떠한 페들러도 될 수 있으며 최소값은 f까지의 거리가 가장 짧은 분포(...)로부터 가장 작은 $\log_2(n)$ 요소의 집합을 반환합니다.

배포된 시스템의 성숙도, 성공적인 추적 기록 및 정확성 증명에 따라 이 라우팅 구조를 사용하기로 선택했습니다. 더 자세한 내용에 관심이 있는 독자는 읽어보실 것을 권장합니다[85]. 우리의 목적을 위해 이 라우팅 체계에서 다음 두 가지 속성이 제공된다는 점에 유의하십시오.

1. *유한하고 결정론적인 연결*. 모든 페들러는$\leq$ 256 개의 강제 연결을 가질 것으로 기대합니다.
2. *로그 통과 거리*. 무작위 주소 t, 연결 C를 지닌 무작위 연결된 페들러 e가 주어졌을 때, $dist(e, t) \approx 2 * \min_{f \in C} dist(f, t)$입니다. 거리가 각 홉마다 반감되므로 네트워크에서 예상되는 통과 길이는 $log_2(n)$입니다(n은 네트워크 크기).

4.3. Orchid Market의 메달리온

메달리온은 작업 증명에 사용되는 토큰이며 이더리움 블록 다이제스트, 보유자의 공개 키 및 부록 D에서 설명된 기타 수량과 밀접하게 연결되어 있습니다. Orchid Market 내에서 메달리온은 두 가지 방식으로 사용됩니다.

- 공격으로 이어지게 될 사소한 시장 침입을 방지하는 방식
- 공격자가 시장에서 자신의 위치를 선택하지 못하도록 방지하는 방식

공격자가 Orchid Market의 총 계산 능력에 비례하는 것보다 많은 페들러를 실행하는 것을 방지하기 위해 모든 페들러는 메달리온 주기마다 모든 연결의 메달리온의 유효성을 확인합니다. 유효한 메달이 제공되지 않으면 네트워크에서 연결이 끊어집니다. 페들러의 위치는 부록 C에 정의된 대로 메달리온의 암호화 해시로 정의됩니다. 즉

$$\text{페들러 주소} = H(\text{메달리온}, ...)$$

이를 통해 Orchid Market의 각 구성원은 시장에서 메달리온 보유자의 위치를 간단하게 평가하고 확인할 수 있습니다. 또한, 페들러의 시장 주소를 메달리온에 연결한 결과, Eclipse 공격을 수행하기가 훨씬 어려워졌습니다.

4.4. 서명된 라우팅 및 Eclipse 공격

분산 네트워크에서 발생하는 문제 중 하나는 (공격자를 제외하고) 네트워크에 대한 전역적인 관점을 가진 사람이 없기 때문에 페들러가 악의적이고 완전히 격리된 하위 네트워크로 *Eclipse 공격*을 받았는지 판단하기 어렵습니다. 예를 들어, 위의 라우팅 체계에서 공격자가 보유 중인 연결에 대해 허위 정보를 제공했다고 가정합시다. 구매자가 이를 감지할 방법이 없다면 공격자가

모든 참가자 를 장악하는 허위 Orchid Market 으로 유도될 가능성이 있습니다. 이러한 상황에 대한 악용 위험을 완화하기 위해, 라우팅 테이블에 포함된 피어가 페들러 라우팅 테이블을 알고리즘적으로 선택함은 물론 확인합니다.

노드가 강제 연결을 설정하려는 경우 해당 노드는 그 강제 연결 목록에 있는 각 노드에게 해당 목록의 서로 다른 노드가 동일한 Orchid Market 의 구성원임을 증명해야 합니다. 그러기 위해서는 먼저 G 라우팅 테이블 $H(C_i)$의 모든 연결의 해시에 가장 가까운 주소를 보유한 페들러를 찾아내어 임의 페들러를 선택해야 합니다. 그런 후 다음과 같은 증명을 제공합니다.

1. 목록에 있는 모든 페들러가 모두 G로 라우팅될 수 있음을 증명
2. G가 각 페들러로 라우팅될 수 있음을 증명
3. 목록에 있는 각 페들러가 실제로 강제 연결임을 증명.

이러한 증명은 모두 C_i 에서 G로 유도하는 서명된 라우팅 테이블 체인의 형태, 또는 (3)의 경우 초보 페들러에서 각 C_i 로 유도한 서명된 라우팅 테이블의 형태를 취합니다. 이러한 증명이 제공된 후에는 새 라우팅 테이블의 모든 피어가 테이블에 서명하고 연결하는 페들러가 자체 테이블에 서명합니다. 그에 대한 새로운 페들러가 강제 연결인 C_i 의 그러한 요소에 대하여, 동일한 증명이 서명을 위한 각 해당 연결로 전송됩니다.

이것이 Orchid Market 에 페들러를 추가하는 유일한 방법이기 때문에 이러한 요구 사항은 Orchid Market 의 건전성에 대한 유도성 증명 방법을 구성합니다. C_i 의 노드 중 하나가 허위 라우팅 테이블의 공급을 시도하는 경우 C_i의 다른 페들러와 동일한 G로 라우팅되지 않습니다. 노드 C_i 중 하나가 Orchid Market 의 구성원이 아닌 경우 G 가 Orchid Market 으로 라우팅될 수 없습니다. 연결을 시도 중인 페들러가 강제 연결 지점에 가장 가까운 노드인 C_i 에 대한 허위 정보를 제공하는 경우 (3)에서 거짓임을 증명합니다.

이러한 속성에서 공격자에게 남아 있는 선택은 다음과 같습니다.

- 공격자가 그에 따라 모든 C_i 가 제어되는 메달리온 주소를 생성할 수 있을 경우 위의 시스템이 기능을 멈추게 됩니다. 이러한 상황이 발생활 확률식은 $\left(\frac{a}{n}\right)^{\log(n)}$ 입니다. 그러한 충돌이 발생할 경우 모든 쿼리의 $(1-\frac{n-\log(n)}{n})^{\log(n)}$ %가 손상됩니다. 이러한 수치를 고려한다면, 공격자가 네트워크의 10%를 제어할 경우 노드가 1 백만 개일 때 그러한 충돌이 발생할 확률은 $\times 10^{-8}$%이며, 모든 시스템의 약 1×10^{3}%를 제어할 경우 쿼리가 영향을 받게 됩니다. 1 억 개의 노드에서 확률은 1×10^{-12}%로 감소하여 쿼리의 1e-5%에 혼란이 야기됩니다. 이 손상은 재생 과정에서 복구됩니다(섹션 4.5 참조).
- 공격자가 네트워크에 가입했지만 유효한 라우팅 테이블을 사용해야 하는 경우 Orchid Market 의 트래픽을 라우팅하지 않고 서비스 판매와 관련된 공격만 수행할 수 있습니다. 이것이 우리의 나머지 공격 모델에서 예상되는 상황(공격자가 계산 리소스에 비례하여 다수의 Peddler 를 제어할 것임)이므로 이를 공격으로 간주하지 않습니다.

4.5. Eclipse 공격 및 재생

오래 지속되는 P2P 네트워크는 Eclipse 공격의 대상이 됩니다. 상기 서명된 라우팅 방식은 검증을 위해 점점 더 많은 피어를 포함시킴으로써 이들을 임의로 어렵게 만들 수 있지만, 또 다른 접근법은 단순히 피어의 수명을 제한하는 것이다. 이러한 이유로 Orchid Market 의 페들러는 100 이더리움 블록마다 키를 변경해야 합니다.

4.6. 엔트리 노드 찾기

엔트리 노드 분배는 어려운 주제입니다. 억압적인 정부가 이 목록에 액세스하게 될 경우 사용자의 목록 액세스가 차단됩니다. 따라서 우리는 인터넷이 차단될 경우 인터넷에 침입할 수 있는 필수 서비스를 찾았으며, 이 서비스에 포함된 데이터에 엔트리 노드 정보를 추가하는 방법을 고안했습니다.

4.7. Orchid Market 식별

새로운 머신에서 올바른 Orchid Market 을 찾을 수 있는 방법이 없다면 위의 보안 논의는 결국 의미가 없습니다. *초보 페들러*를 위해 존재하는 모든 분산 방법은 공격자가 제어하는 초보 페들러의 침입에 대한 저항력이 있는 것으로 추정할 수 없습니다. 그렇게 하려면, 우리는 단순히 주어진 Orchid Market 의 컴퓨팅 파워를 추정하고 전체적인 컴퓨팅 파워를 가장 풍부하게 보유한 시장을 선택합니다.

- 밀도 추정. 페들러의 강제 연결은 2^{256} 주소 공간의 어떤 점집합에 가장 가까운 페들러로 정의되므로 실제 상황에서 이상적인 연결과 실제 연결 사이에는 측정 가능한 간극이 있게 됩니다. 이 공간의 밀도를 추정하기 위해, 다음과 같은 방식으로 이러한 연결을 무작위적인 이항 과정의 결과로서 관찰할 수 있습니다. 이상적인 점과 실제의 점 사이의 모든 점은 실패이며 실제의 점은 성공입니다. 따라서 알 수 없는 노드 M의 주어진 수와 실현되지 않은 연결 C의 주어진 수에 대하여, 네트워크 밀도의 균일한 사전 MAP 추정치를 아래 공식을 통해 얻을 수 있습니다.

$$\frac{C}{C+M} * 2^{256}$$

- 통과 거리. Orchid Market 은 $O(log_2(n))$ 홉의 주소 순람을 제공합니다. 네트워크 밀도를 추정하기 위해 이것을 역이용할 수 있습니다.

밀도 추정으로 충분하다고 생각할 수도 있지만, 적당한 크기의 시빌 네트워크를 소유 한 영리한 공격자는 어느 노드를 잘못된 네트워크의 초보 페들러로써 이용할지 자유롭게 선택할 수 있습니다. 한편 실제 Orchid Market 의 초보 페들러는 네트워크에서 무작위로 표본 추출한 밀도를 갖게 됩니다. 설상가상으로, 통과 거리가 지표로 선택된 경우 이를 예상하는 공격자를 상상해 볼 수 있으므로 $O(log_2(n))$보다 긴 거리를 통과해야 하는 차선의 라우팅 테이블을 생성합니다. 고맙게도, 차선책으로 연결된 Orchid Market 은 밀도 지표상에서 활동이 약화될 것입니다. Orchid 시스템에서 사용되는 확인 방법은 임의의 주소로 이동하여 경로를 따라 라우팅 테이블을 저장한 다음 처음 두 홉을 제외한 모든 라우팅 테이블을 사용하여 밀도 추정을 수행하는 것입니다.

4.8. 프록시 화이트리스트

프록시 서비스를 제공하려는 일부 사용자는 공개 액세스 권한 을 제공하는 것이 불편할 수 있습니다. 예를 들어, 사용자가 facebook.com 에 액세스할 수 있도록 허용하면 릴레이 역할과 유사한 위험 프로필이 있지만 인터넷에 대한 임의의 연결을 허용하면 현지 법 집행 기관의 조사 대상이 될 수 있습니다. 따라서 Orchid Market 의 페들러는 사용자가 프록시로 사용할 때 연락할 수 있는 웹 사이트의 화이트리스트를 설정하고 *오퍼 받기*에 대한 응답에 화이트리스트를 지정할 수 있습니다.

5. 메달리온

완전 분산형 완전 익명 디지털 시스템은 단 한 명의 악의적인 사용자가 수천 명의 사용자를 가장하는 방식의 공격(시빌 공격)의 목표가 됩니다. 시빌의 생성 및 이 공격 유형의 기타 영향을 완화하기 위해 Orchid Protocol 은 작업 증명 체계를 사용합니다. 이러한 체계를 메달리온이라고 합니다. 각 메달에는 생성자가 일정 시점에 *상당한 규모의* 계산 리소스를 보유하고 있음을 암호화 방식으로 보여주는 데이터가 포함됩니다. 계산은 비용이 많이 드는 리소스이므로 메달리온을 사용하면 주어진 공격자가 여러 명의 사용자를 사칭할 수 있는 능력에 예산 제약이 따르게 됩니다.

5.1. 메달리온 작업 증명

메달리온은 핵심 보안 가정과 전체적인 네트워크 사이의 가교를 형성합니다. 우리의 기본적인 보안 목표는 상당히 집요한 공격자가 Orchid Network 를 제어하지 못하도록 제한하는 것이므로 우리가 메달리온 만들기를 선택할 경우 다음 조건을 충족해야 합니다.

1. 메달리온 만들기는 비악의적인 노드를 생성하기가 *쉬워야* 합니다.
2. 메달리온은 검증하기가 *쉬워야* 합니다.
3. 메달리온은 대량으로 만들기가 *어려워야* 합니다.

이러한 조건에 따라 우리는 *어려움*을 시간 및 돈에 있어서의 억제적 확장성을 의미하는 것으로 정의합니다. 요컨대 우리는 정상적인 노드가 네트워크로의 진입을 얻기 *쉽지만* 공격자가 네트워크로의 진입을 확장하는 것은 어려운 작업 증명 시스템을 원합니다. 섹션 5.2 에서 지분 증명 [46, 70, 72] 및 공간 증명 [63, 80]과 같은 다른 방법에 대한 작업 증명의 선택에 대해 설명하겠습니다.

현재 위의 요구 사항을 충족하는 다음의 두 가지 기본 방법이 존재합니다. 도전-응답 프로토콜 및 암호-퍼즐. 안타깝게도 공격자가 공모를 통해 공격 및 응답을 사전 계산할 수 있으므로 공격-응답 프로토콜이 Orchid 모델 내에서 충분한 보안을 제공하지 못할 수 있습니다. 이것은 현재 많이 존재하는 암호-퍼즐을 남기며 [50, 78] 각 암호-퍼즐은 자체적인 거래를 지닙니다. 다시, Orchid 의 요구 사항을 충족하기에는 이러한 암호-퍼즐의 하위 집합만이 적합합니다. 즉, 쉽게 병렬화하거나 ASIC 으로 만들거나 사소하게 확장할 수 없는 암호-퍼즐입니다. 최근 연구자들은 조정 가능한 생성이 어렵고 검증하기 쉬운 결과를 생성하는 알고리즘을 발견했습니다 [50]. 이러한 알고리즘 모음은 메모리 및 총 실리콘 면적이 확장하는 데 비용이 많이 드는 경향을 이용합니다 [45, 61]. 이러한 알고리즘 클래스를 비대칭 메모리-하드 함수라고 하며 메달리온 생성에 사용됩니다. 이러한 기능에는 여러 가지 종류가 있지만 [50, 75, 86] Equihash 를 사용하기로 선택했습니다. Equihash 는 k-XOR 생일 문제를 기반으로 하며 시공간 거래를 통해 메모리 경도를 제공합니다 11. Equihash 는 조정 가능하고 단순하며 NP 문제를 기반으로 하며 암호 화폐 커뮤니티에서 승인을 얻었으므로 작업 증명 기반으로 이러한 기능을 사용하면 수용 가능한 보안 수준과 미래 안전 보장(future-proofing) 기능을 제공할 수 있다고 생각합니다.

메달리온을 만들려면, 피어가 공개 키 K 와 이전의 이더리움 블록 해시 E 를 가져온 다음 $F(K, E, S, \ldots) \geq N$인 소금 S 의 위치를 찾아내기 위해 일련의 계산을 수행합니다. 여기서 N은 일종의 난이도 확장 계수입니다. 새로운 이더리움 블록이 체인에 추가되면, 메달리온을 최신 상태로 유지하도록 새로운 S를 계산해야 합니다. 메달리온 사양은 부록 C 에 자세히 정의되어 있습니다.

[1] 이러한 시간–공간 교환이 [67]에 의해 처음 발견된 시간–메모리 교환을 연상시키는 것은 결코 우연이 아닙니다.

5.2. 증명 유형의 선택

다른 시장 기반 분산형 네트워크에 익숙한 독자는 메달리온 사용이 그 전제에 있어서 다른 작업 증명 시스템(비트코인 등)과 유사하다는 것을 인식할 것입니다. 더 많은 독자들은 다음과 같은 질문을 할 수도 있습니다. Orchid Protocol, 특히 Orchid Market 에 대한 승인을 얻는 데 지분 증명, 유휴 증명 또는 기타 방법을 사용하는 것은 어떻습니까? 이 섹션에서는 다른 방법보다 작업 증명을 선택한 이유를 설명하겠습니다.

지분 증명

지분 증명은 공격자가 대부분의 토큰을 제어하지 않을 것이라는 가정에 근거합니다. 우리의 공격 모델에서는 매우 집요하고, 넉넉한 자원을 보유하며, 아무래도 압제적일 정부를 가정하며, 스테이크 증명 가정이 항상 진실한 것으로 간주할 수 없습니다. 비트코인의 엄청난 시가 총액조차도 중간 규모의 국가의 GDP 보다는 훨씬 적습니다. 문제를 좀 더 복잡하게 만드는 요인으로서, 우리가 가까운 시일 내에 익명 결제를 지원하도록 시스템을 확장할 경우, 이러한 적대적 인수 를 탐지하기가 훨씬 어려워질 것입니다. 따라서 시스템의 충분한 지분이 Orchid Protocol 의 익명성 및 보안을 영구적이며 비가역적으로 손상시킬 수 있기 때문에, 지분 증명 모델을 토대로 메달리온을 만들 수 없었습니다. 간단히 말하자면, 우리는 사용자의 개인 정보 보호 권리가 최고 입찰자에게 판매될 수 있는 시스템을 설계하려 하지 않았기 때문에 지분 증명을 사용하지 않았습니다.

공간 증명

공간 증명에서, 작업 증명 시스템에 사용되는 것과 같은 계산 자원은 저장 공간을 위해 거래됩니다. 간단히 말해, 공간 증명은 증명자가 검증자 주도형 계산을 수행하여 일정한 양의 저장 공간을 보유함을 검증하기 위해 두 참가자(한 명의 증명자와 한 명의 검증자)가 상호 작용하는 *인터렉티브* 프로토콜입니다. 이러한 계산은 증명자가 저장하고 리콜한 경우에만 실용적이라는 가정입니다[63]. 적절한 방법을 찾을 수 있을지 확실하지 않지만 우리는 곧 출시될 Orchid Protocol 버전에 공간 증명을 사용할 가능성을 모색하고 있습니다.

유휴 증명

유휴 증명은 정기적이며 동기화된 작업 증명이 사용자의 전역적 컴퓨팅 능력에 대한 점유율을 입증하기에 충분하다는 추가 가정에 근거합니다. 불행히도 네트워크가 초기 단계에 있는 동안($\leq$ 1 천만 페들러들), 이는 슈퍼 컴퓨팅 센터를 통제하는 회사가 컴퓨팅 능력의 1%만 희생하면 네트워크를 제어할 수 있는 상황을 초래합니다. 이 공격이 치명적인 활동을 중단하기에 충분한 수의 페들러가 확보될 때까지 상당한 시간이 소요될 것으로 예상되기 때문에, 이 릴리스에는 유휴 증명을 사용하지 않고 있습니다.

5.3. 메달리온 사양

높은 수준에서, 메달리온을 생성하려면 다음과 같은 두 단계가 필요합니다. (1) 공개/개인 키 쌍 K 의 생성 및 가장 최근의 이더리움 블록 다이제스트 E 검색, (2) (반복적 또는 병렬적으로) $F_N(K, E, S)$가 어떠한 당첨 조건을 확보하는 소금 S의 위치 찾기. 여기서 N은 일종의 난이도 확장 계수입니다. 메달리온의 목표는 특정 엔터티에 작업 증명을 제공하는 것입니다. 따라서 메달리온을 사용하여 여러 명의 피어를 사칭할 수 없도록 각 메달리온을 특정 공개 키에 암호화 방식으로 연결해야 합니다. 또한 우리는 모든 엔터티가 활용할 수 있는 사전 계산 이점의 양을 제한하고자 합니다. 따라서 메달리온은 이더리움 블록 다이제스트와 암호화 방식으로 연결되어

10 초 단위로 변경됩니다. 다음은 메달리온 사양에 대한 정의입니다.

pk_m 은 피어 m 에 속하는 고유한 공개 키임

sk_m 은 pk_m 과 연결된 고유한 비밀 키임

e_t 는 t 시점의 이더리움 블록 다이제스트임

$h(y)$는 입력 y 를 지닌 암호화 해시 함수의 다이제스트임

$sig(sk, r)$은 비밀 키 sk 를 사용하는 r 의 기본 서명임

$F_{n,k}(x_j)$는 시작 카운터 x_j 와 난이도(n, k)를 지닌 Equihash 출력값2임

$seed$ 는 is $h(e_t / sig(sk, e_t))$임

$h(y)$는 암호화 해시 함수일 수 있으나 Orchid Protocol 은 Keccak 을 사용합니다. 이 해시 함수 선택에 대한 설명은 부록 D.1 을 참조하시기 바랍니다. 비밀 키로 적절한 크기의 일부 데이터를 지수화하기 위해 기본 서명을 정의합니다.

이러한 정의를 사용하여 메달리온을 아래와 같이 집합으로 정의합니다.

$$M = \{t, e_t, pk_m, sig(sk, e_t), F_{n,k}(seed)\}$$

위 공식은 전 세계적으로 합의된 Equihash 난이도 매개 변수 (n, k)에 대한 것입니다. 이러한 매개 변수에 대한 자세한 내용은 [50]을 참조하십시오. $seed$ 를 F 에 대한 입력으로 사용하면 피어의 개인 키를 메달리온과 연결합니다. 메달리온은 Orchid Market 에서 피어의 Chord 주소를 결정하므로 메달리온을 소유한 모든 엔터티는 특정 피어와 연결된 pkm 을 사용하여 검증 가능합니다. 또한, 특정 Chord 주소에서 공개 키의 소유권 증명을 요청할 수 있습니다. 메달리온에 대한 엔지니어링 세부 사항은 부록 C 에 설명되어 있습니다.

6. 결제

6.1. Orchid 결제 요구 사항

대부분의 결제 시스템에서 대상 품목의 비용은 거래 비용보다 훨씬 큽니다. 특히, 대상 품목의 비용은 한 당사자에서 다른 당사자로 자금을 이체하는 데 드는 비용보다 훨씬 큽니다. 대부분의 인터넷 구매 및 네트워킹 비용은 거의 사소한 비용으로 무시될 수 있습니다. 그러나 Orchid Network 에서 대상 항목의 비용은 대역폭입니다. 즉, 유선을 통해 전송되는 각 패킷에는 관련 비용이 있습니다. 따라서 지불을 전송하기 위한 거래 비용이 단일 패킷의 비용만큼 낮을 경우 이러한 비용은 동등하게 됩니다. 이것은 물론 Orchid Protocol 의 경제적 가정을 무너뜨릴 것입니다.

임의의 정밀도로 대역폭을 판매하고 싶으며 거래 수수료를 임의로 낮출 **필요가 있기** 때문에 사용자가 최소한의 거래 비용으로 임의의 양의 릴레이 트래픽을 지불할 수 있는 새로운 형태의 결제 시스템이 필요합니다. 우리는 이제 거래 비용을 임의의 수준으로 낮출 수 있고 임의의 대역폭 분할이 가능한 결제 시스템이 필요합니다. 또한, Orchid Protocol 의 목적은 인터넷 감시 및 검열을 상당한 수준 줄이는 것입니다. 따라서 결제 메커니즘에 대한 추가 요구 사항에는 다음이 포함되어야 합니다. 검열 불가능성, 익명성 및 신뢰할 수 있는 제 3 자에 대한 비의존성. 즉, 기본 네트워크에 감시 및 검열에 대한 저항력이 있지만 결제 메커니즘은 그렇지 않더라도 시스템이 이용당할 수 있으며 사용자가 검열당하거나 추적당할 수 있습니다. 마찬가지로, 신뢰할 수 있는 제 3 자에 의존하면 Orchid Network 가 지불 제공자에게 영향을 줄 수 있는 상당히 집요하거나 강력한 엔터티의 간섭에 노출될 수 있습니다.

[2] $F(x_j)$의 출력은 출력 내 모든 j에 대해 $XOR_j = h(j)$, $\Sigma XOR_j=0$이 성립되는 카운터 j의 집합입니다.

따라서 Orchid 결제의 요구 사항은 다음과 같습니다.

1. *경제적 실행 가능성*, 결제 실행이 어떠한 경우에도 저렴해야 합니다.
2. *위조 불가능성*, 결제 토큰 소유자만이 토큰을 결제에 사용할 수 있어야 합니다.
3. *가용성*, Orchid 결제 전송 또는 결제 수령을 막을 수 있는 엔터티가 없어야 합니다.
4. *불가역성*, **모든** 엔터티가 이전의 결제를 되돌릴 수 없어야 합니다.
5. *익명성*, 참가자가 계정 주소, 결제 금액 또는 시간과 관련이 없어야 합니다.[3]

우리는 다음 섹션에서 결제를 위한 잠재적 솔루션에 대해 논의합니다. Orchid 결제(섹션 6.5)는 익명성 요구 사항을 제외한 모든 요구 사항을 충족한다는 점을 강조하고자 합니다.

6.2. 전통적인 결제

현재의 금융 결제 시스템에서, 거래는 결제 카드용 ISO/IEC 7816[3] 및 은행 결제용 EBICS[4]와 같은 프로토콜을 사용하여 은행 또는 결제 서비스 제공업체[2]와 같은 두 개 이상의 엔터티 사이의 협상을 통해 정산됩니다. 이러한 프로토콜은 SWIFT[5] 및 NYCE[6]와 같은 네트워크에서 실행되어 국내 및 국제 거래를 모두 지원합니다. 이러한 네트워크를 구성하는 엔터티는 각각 자체적인 원장을 보유하고 있으며 전자 결제 영수증 및 수동 조정을 통해 지속적으로 자체 원장을 업데이트합니다[7].

기존의 결제 네트워크에 연결하려면 일반적으로 대부분의 사법 관할권에서 특별 허가가 필요할 뿐 아니라 연결 엔터티 간에 건별 비즈니스 계약이 필요합니다. 그 결과로서 형성되는 글로벌 금융 네트워크는 비즈니스와 프로토콜 및 네트워크의 혼합물을 연결하는 허가된 임시망으로 볼 수 있습니다. 각 원장은 단일 실패 지점을 나타내며, 암호화 무결성이 결여되어 있고 제어력을 지닌 비즈니스 엔터티의 독단에 따라 자의적으로 수정할 수 있습니다.

전통적인 결제 프로토콜은 일반적으로 거래 수수료를 정의하지 않지만 프로토콜을 실행하는 엔터티는 수수료를 최우선적으로 추가합니다. 거래당 수수료는 결제 카드 거래의 경우[8] 불과 몇 센트에서 국제 송금의 경우에는[9] 최대 $75 까지 다양합니다. 그 대신 또는 그 외에도, 많은 시스템에서 거래 금액에 대한 일정 비율의 수수료를 청구합니다. 이 수수료는 은행 송금의 경우[10] 13%, 결제 카드의 경우[11] 3.5%입니다.

전통적인 결제는 신뢰할 수 있는 당사자에 의존하기 때문에 필요한 속성을 희생하지 않으면서 Orchid Network 에 사용하는 것은 사실상 불가능합니다. 특히 가역성은 설계 개념에 따라 역거래 형태로 존재합니다[77]. 거래는 일반적으로 위조하기 어렵지만 신용 카드 사기가 일반화되어 있으며 신원 도용 또는 해킹으로 인해 사용자 계정이 손상될 수 있습니다. 또한 이러한 결제 시스템은 불편한 시간대에 오작동을 일으키고 정기적으로 다운 타임이 발생하기 때문에 부분적인 가용성만 기대할 수 있습니다. 결제를 관리하는 신뢰할 수 있는 당사자는 일반적으로 발신자, 수신자, 결제 금액 및 시간에 대한 기록뿐만 아니라 발신자에 대한 신원 정보를 가지고 있기 때문에 익명성이 결여되어 있습니다. 마지막으로, 다음 섹션에서 알 수 있듯이, Orchid Network 에서는 전통적인 지불 순서에 따른 거래 수수료가 엄청나게 비쌉니다.

[3] 익명성은 악의적인 관찰자들 뿐만 아니라 악의적인 발신자 또는 수신자들에 대해서도 대응하는 것이 가장 바람직합니다.

6.3. 블록체인 결제

비트코인은 전통적인 결제 시스템의 현상태를 혁명적으로 변화시켰으며 결제 및 국제 송금을 위한 글로벌 시장을 계속 혼란시키고 있습니다. 비트코인은 지리적 경계에 구애 받지 않는 글로벌 네트워크 및 프로토콜입니다. 공개 키 암호화를 적용하면 거래는 신뢰할 수 있는 당사자가 필요 없이 사용자 자신이 생성한 주소 간에 비트코인 금액을 전송합니다. 사용자는 공개 키의 해시를 결제 주소로 사용할 수 있는 키 쌍을 생성하며, 개인 키는 주소에서 전송에 서명해야 합니다[12]. 비트코인 결제는 위조 불가능하며 비가역적입니다[79](블록 확인을 고려할 적절한 시간 내에). 비트코인 네트워크는 그 출범 이래 다운타임이 최소화되었으며 채굴자가 활발한 검열을 실행할 가능성을 배제할 경우(가능성은 거의 없음, 섹션 6.6 에서 자세히 설명함) 범용성이 있다고 볼 수 있습니다. 비트코인 결제는 의사 익명성을 지니며 익명 수준은 네트워크 사용 방법에 따라 크게 다릅니다[68].

일반적으로 분산형 암호 화폐는 사상 최초로 사람과 컴퓨터 시스템이 모두 신뢰할 수 있는 제 3 자 없이 가치를 거래할 수 있게 해 줍니다. 이는 Orchid 와 같은 인센티브화된 분산형 오버레이 네트워크의 필수 요건입니다.

비트코인의 거래 수수료는 거래 금액이 아니라 거래 데이터 구조의 크기에 발신자가 구성한 계수를 곱한 값으로 결정됩니다. 2017 년까지 평균 거래 수수료가 $1 을 훨씬 밑돌았지만, 2017 년 2 월 비트코인 네트워크가 최대 거래 용량에 도달함에 따라 수수료가 빠르게 상승했습니다. 평균 수수료가 $8 로 상승하여[13] 낮은 수수료에 의존하는 애플리케이션은 비트코인 네트워크에서 경쟁력을 상실했습니다.

공개 키 암호화를 또한 토대로 하고 비트코인과 같은 작업 증명으로 보호되는 이더리움 네트워크는 위조 불가능성, 가용성 및 (비 클래식) 비가역성의 동일한 속성을 도출합니다. 이더리움은 더 높고 동적으로 조정 가능한 거래 용량을 가지고 있으며 2015 년에 출시된 이래 네트워크 수수료가 낮았습니다. 그러나 거래 횟수가 증가하고 이더리움의 기본 토큰인 이더(Ether)의 가격이 상승함에 따라 거래 수수료(*가스*라고 함)는 최고가 $1.00 까지 평균 $0.20 로 상승했습니다 [14]. 스마트 컨트랙트 코드를 실행하는 거래는 계산 수행량에 비례하여 훨씬 더 고가입니다.

퍼블릭 블록 체인 네트워크에서 널리 이용되는 거래 수수료가 상승하면 소액 결제를 직접 처리할 수 있는 가능성이 제한되어 결제 채널과 같은 2 계층 솔루션으로 소액 결제가 집중됩니다.

6.4. 블록체인 기반 확률적 소액 결제

확률적 지불이 블록체인 프로토콜에 어떻게 적용될 수 있는지에 대한 핵심 개념을 쉽게 전달하기 위해 몇 가지 세부 사항을 살펴보겠습니다. MICROPAY1 체계에 대한 공식적인 설명은 인용된 원본 논문에 있으며, Orchid 확률적 지불 체계는 섹션 6.5 에 공식화되어 있습니다.

Pass & Shelat 는 MICROPAY1[81]을 설명하면서 디지털 서명과 확약 체계가 결합되어 정확한 확률의 무작위 결과를 포함한 릴리스 조건을 엔지니어링하는 방식을 보여 줍니다. 발신자는 먼저 비트코인을 새로 생성된 키의 에스크로 주소로 전송하여 예치금 을 만듭니다. 그런 다음 수신자(MICROPAY1 조건의 판매자)는 임의의 숫자를 선택하고 이 숫자에 대한 확약을 발신자에게 보냅니다. 확약과 함께 수신자는 새로운 비트코인 주소도 제공합니다. 발신자는 또한 임의의 숫자를 선택하고 이 숫자의 연결(일반 텍스트), 수신자의 확약 및 수신자가 제공한 지불 대상 주소와 같은 기타 지불 데이터에 서명합니다.

그에 따라 발행된 티켓을 확인하려면 수신자의 확약이 공개한 숫자와 일치하는지 확인하고 발신자의 서명이 비트코인 예치금의 주소와 일치하는지 확인해야 합니다. 발신자 및 수신자로부터 난수의 XOR 의 마지막 두 자리가 00 이면 티켓이 이기고 수신자가 이용할 수 있습니다.

직관적으로, 발신자가 약정의 바인딩 속성을 위반할 수 없거나 (서명을 위조할 수 없거나) 사용자가 약정의 숨기기 속성을 깰 수 있는 경우에는 이 체계의 코인 던지기 를 편기가 없는 것으로 생각할 수 있습니다.

발신자는 수신자의 티켓 청구를 확인했을 때 지출을 브로드캐스팅하여 여러 명의 수신자에게 병렬적으로 또는 해당 수신자에게 선매매 거래로 티켓을 발행하여 예치금을 이중 지출 할 수 있습니다. MICROPAY1 의 저자는 페널티 에스크로 (미래의 일정 시점에 동일한 지불 에스크로에 대해 두 개의 유효한 티켓을 제출할 수 있는 사람이 삭제 하거나 구울 때까지 발신자에게 상환될 수 있는, 발신자가 예치한 2 차 금액)에 의한 해결 방법을 설명합니다. 이렇게 하면 발신자가 수신자와 공모하거나 본인이 수신자인 것처럼 가장하는 행위를 방지할 수 있습니다.

MICROPAY1 의 작성자는 MICROPAY2 및 MICROPAY3 에서 반복 개선을 구성합니다. MICROPAY3 에서는 티켓에 대해 일부 계산 유효성 검사 단계를 수행하고 계산이 올바른 경우 서명을 해제하기 위해 신뢰할 수 있는 당사자가 도입됩니다.

6.5. Orchid 지불 체계

이제 지불에 적합한 추상화를 찾았으니 다음과 같이 질문할 차례입니다. 어떻게 구현해야 하는가?

섹션 6.1 에서 논의된 요구 사항과 함께 다음 사항도 충족하고자 합니다.

- *재사용성*, 각각의 새로운 티켓을 구성하는 방법은 각각의 티켓에 대해 새로운 거래 수수료 또는 새로운 온체인 거래를 요구하지 않아야 합니다. 그렇지 않으면 거래 수수료가 다시 한 번 문제됩니다.
- 이중 지출을 막거나 수익성 악화를 막아야 합니다.
- 시스템은 패킷 비용을 압도하지 않도록 계산 비용 측면에서 충분히 성능이 우수해야 합니다.

이러한 요구 사항 중 마지막 요소는 아마도 가장 번거로운 것일 수 있습니다. 우리가 아는 한, 이더리움 토큰을 기반으로 한 복권 티켓을 구성하는 방법은 존재하지 않으므로 ECDSA 서명 확인 순서에 따라 계산할 필요가 없습니다. 이 섹션에서 자세히 설명된 것처럼, 이는 발신자가 티켓 금액과 당첨 확률뿐 아니라 발신자의 이더리움 계정에 티켓 전송을 위한 충분한 양의 Orchid 토큰이 고정되어 있음을 수신자에게 암호화된 방식으로 입증해야 할 필요에 따른 것입니다.

이러한 이유로, 단독으로 사용하기에는 충분하지 않았지만, 위에서 언급한 것과 유사한 거래 수지(balance-of-trade) 접근법을 사용해야 합니다. 이것은 새로운 요구 사항, 즉 거래 수지(balance of trade)를 충분히 작게 유지하여 거래 중에 인센티브가 단절되지 않도록 해야 한다 는 새로운 요구로 이어집니다. 이것은 구현 현실에 의해 야기된 메커니즘 설계 문제이므로, 솔루션이 존재한다고 가정하여 지금은 구현에 초점을 맞추고 추가적인 사항은 이후에 섹션 6.9 에서 설명 드리겠습니다.

Orchid 지불 체계는 MICRO-PAY1 및 관련 구성에서 영감을 얻은 의사 익명의 확률적 소액 지불 체계입니다. 이더리움 스마트 컨트랙트와 감액 가능 페널티 예치금을 활용하여 신뢰할 수 있는당사자 없이 전면 실행 및 병렬(이중 포함) 지출 공격을 완화합니다. Orchid 결제의 의사 익명성은 정규 이더리움 거래에서 달성할 수 있는 것과 동등합니다(Orchid 클라이언트는 일회성 주소 및 노드 ID 와 결제 주소 사이의 키 분리와 같은 추가적인 개인 정보 보호 기술을 사용하여 제한된 익명성을 달성합니다).

MICROPAY2 및 MICROPAY3 에 도입된 신뢰할 수 있는 당사자는 이더리움 스마트 컨트랙트 코드로 효과적으로 대체될 수 있습니다. EVM 을 통해 소액 결제 티켓을 검증하기 위한 임의의 논리(계산에 대한 경제 범위 내)를 구현할 수 있으며 ECDSA[71] 복구 작업과 암호화 해시 기능을 위한 기본 요소[89]를 제공합니다. 결제 체계에 대한 자세한 설명은 부록 D 에 설명되어 있습니다.

6.6. Orchid 토큰

Orchid Network 는 위조 불가능성, 가용성 및 비가역성의 결제 요구 사항을 충족하기 위해 이더리움 기반 ERC20 토큰을 사용하고 있습니다. 다음 섹션에서는 ERC20 전송에 대한 거래 수수료를 낮추어 임의의 적은 양의 토큰을 보낼 수 있는 방법에 대해 설명합니다. 결제 익명성은 섹션 6.10 에서 설명합니다.

Orchid 토큰(OCT)은 Orchid Network 내 결제에 사용됩니다. Orchid 토큰은 새로운 이더리움 기반의 ERC20 과 호환되는 고정 공급 토큰입니다. 공급이 1×10^9 (10 억)토큰으로 고정되며, 여기서 각 토큰은 1×10^{18} 의 나눌 수 없는 하위 단위(이더(Ether)와 동일한 가분성)를 갖습니다.

언뜻 보기에 다음 섹션에서 자세히 설명하는 Orchid 결제 시스템은 이더(Ether) 또는 ERC20 토큰을 사용하도록 구성할 수 있습니다. 실제로 이더(Ether)를 사용하면, 사용자가 (거래 수수료를 위해) Orchid 토큰과 이더를 모두 확보할 필요 없이 이더만 필요하므로, 티켓 계약을 단순화하고 가스 비용을 약간 줄이고 사용성을 향상시킬 수 있습니다.

그러나 이더리움은 ERC20 토큰을 포함한 임의의 메커니즘으로 거래 수수료를 지불할 수 있도록 향후 프로토콜 업그레이드를 계획하고 있습니다[15] [16]. 이것은 새로운 토큰 사용의 단점을 대부분 제거합니다. 가스 비용에는 차이가 없으며 사용자가 단일 토큰만 구입하면 됩니다. 가스 가격을 0 으로 설정하고 계약 실행 시 ERC20 토큰 지불을 채굴자(EVM COINBASE [89] OP 코드 사용)에 추가할 수도 있습니다[17]. 가스 가격이 제로가 되고 거래 실행에 코인베이스 주소로의 ERC20 토큰 전송이 포함되어 있는지 검증하기 위해 채굴 전략을 구성해야 하므로 채굴자의 명시적인 지원이 필요합니다.

그러나 단순히 이더를 사용하는 대신 새로운 토큰을 도입하기로 한 결정은 기술적인 이유 때문이 아니라 사회경제적 이유 때문입니다. Orchid Network 에서 새 토큰을 만들어 유일하게 유효한 결제 옵션으로 삼으면 복잡성 증가를 보장할 만큼 중요한 것으로 여겨지는 사회경제적 효과를 창출하게 됩니다.

6.7. Orchid 가스 비용

위 체계의 견고성 프로토타입 구현을 통해 약 87,000 건의 가스 비용을 측정했습니다. 이 비용은 당첨된 티켓을 입력으로 호출할 때 티켓 소유권 주장을 위한 API 의 전체 실행을 위한 비용입니다. 티켓 청구 실행에는 Orchid ERC20 원장 *전송* API 에 대한 하위 호출이 포함됩니다. 모든 Orchid 스마트 계약의 견고성 구현은 암호화 검토 및 최소한의 외부 보안 감사 후에 소스를 공개합니다.

6.8. 검열 저항

대부분의 퍼블릭 블록체인 네트워크에서와 마찬가지로 이더리움 거래는 검증자(이더리움 네트워크의 채굴자)가 생성된 블록에 거래를 포함시키지 않기로 결정하지 않는 한 검열할 수 없습니다. 블록이 해시 파워에 비례하여 모든 채굴자 간에 무작위로 채굴되기 때문에, 대다수의 채굴자가 Orchid 결제를 적극적으로 검열해야 함에 따라 Orchid Network 에 상당한 정체 및 혼란을 야기합니다. 예를 들어, 해시 파워의 90%가 Orchid 관련 거래를 포함하지 않기로 선택한 경우에도 거래를 확인하기까지 평균적으로 10 배 더 오랜 시간이 소요된다는 경고와 함께 Orchid Network 가 여전히 작동합니다. 다수의 채굴자 그룹(예: 51%)이 Orchid 관련 거래를 포함하는 블록을 거부함으로써 Orchid 관련 거래를 검열하기로 선택한 경우 더 엄격한 형태의 검열이 가능할 것입니다[73]. 이는 이더리움 프로토콜의 규칙에 따라 유효하며, 소프트 포크를 효과적으로 생성합니다. 하지만 그와 같은 소프트 포크를 생성하기 위해 대규모 채굴자 담합을 조직하는 경우 엄청난 수익 손실 위험이 수반됩니다. 소프트 포크가 충분한 해시 파워를 구현하지 못하면, 담합 채굴자들이 블록 보상을 놓치게 됩니다. 수익 손실 위험과는 별개로, 이더리움 채굴자의 분산화된 특성과 블록체인 채굴 전략에 대한 법적 및 규제적 제한의 부재 때문에 그럴 가능성은 극히 낮다고 판단됩니다.

6.9. 거래 수지(balance of trade)

2 명의 Orchid 참가자인 Alice 와 Bob 이 완전히 익명으로 거래하기를 원한다고 상상해 봅시다. Bob 은 자신이 x 를 청구하는 몇 가지 작업을 수행해야 하며, Alice 는 y 회의 작업마다 Bob 에게 지불해야 합니다. 불행히도, 익명의 본질이란 사전 거래가 없을 경우 Alice 와 Bob 이 서로를 신뢰하는 메커니즘이 없다는 것입니다. Alice 와 Bob 이 협력할 수 있을까요?

Alice 와 Bob 의 관계에 약간의 설치 비용이 있는 경우(S_{Alice}, S_{Bob} s.t. $S_{Alice} > xy$, $S_{Bob} > xy$), 답은 '예'입니다. (1) Alice 가 도모하는 작업의 총량이 $\leq xy$ 이거나 (2) Bob 이 수행할 수 있는 작업의 총량이 $\leq xy$ 인 경우가 아니면, 돈을 착복하고 도주하거나 작업의 경제적 합리성을 상실하게 됩니다. Orchid Market 에 관한 설명(섹션 4)에서 알 수 있듯이 Orchid Network 에 1×10^3 단위의 패킷이 남는 거래 불균형을 유지하는 설치 비용이 존재합니다. Orchid Market 의 판매자는 일반적으로 구매자보다 높은 설치 비용을 지불하고 고객은 필요한 작업량에 대한 정보를 비대칭적으로 소유하기 때문에, Orchid Network 는 고객에게 선불 결제를 요구합니다.

6.10. 익명성

이전 섹션에서 설명한 Orchid 결제는 일반적인 이더리움 거래와 마찬가지로 의사 익명적입니다. 금액과 발신자 및 수신자 계정을 포함한 모든 거래는 공개됩니다. Orchid Client 는 단일 루트 키를 사용하더라도 지불 주소의 연결 불가능성을 입증할 수 있도록 일회용 주소[18] 및 HD 지갑 사용[19]과 같은 최신 지갑 기술을 사용하여 공개 블록체인 거래의 기본적인 의사 유사성을 향상시키는 것을 목표로 합니다.

이더리움 비잔티움 릴리스에서는 타원 곡선 작업을 위한 새로운 EVM 기본 요소를 활용하여 저렴한 가스 비용으로 연결 가능한 링 서명을 구현할 수 있습니다[20]. 이더리움 스마트 계약과 HD 지갑 및 연결 가능한 링 서명과 같은 스텔스 주소를 결합하면 M¨obius[76] 혼합 서비스와 같은 혼합 기술 등급을 사용할 수 있습니다. M¨obius 는 혼합 서비스를 위한 게임 기반 보안 모델을 사용하여 암호화 방식으로 입증된 강력한 익명성 보장을 제공합니다. 그러나 이전 혼합 기술과 달리 악의적인 관찰자 및 발신자에 대해서는 익명성을 제공하지만 악의적인 수신자에게는 익명성을 제공하지 않습니다. M¨obius 와 같은 서비스를 Orchid 확률적 소액 결제와 결합하면 결제에 대한 최종 요구 사항인 익명성에 더 다가갈 수 있습니다.

악의적인 행위자(결제의 관찰자, 송금자 또는 수취인인지 여부는 상관없음)를 상대로 완전한 익명성을 보장하려면 제로 지식(zero-knowledge) 기술을 지향해야 합니다.

7. 대역폭 채굴

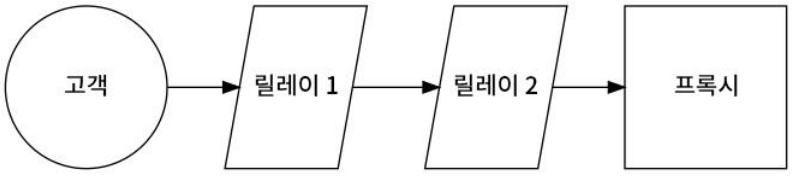

그림 2: 고객을 위한 3 개의 페들러가 있는 체인 라우팅 트래픽

이 섹션에서는 릴레이 및 프록시 동작에 대한 사양을 설명하고 검열할 수 없는 익명의 웹 브라우징을 지원하기 위한 이러한 노드의 체인 연결 에 대해 설명합니다.

대역폭 판매를 위한 사양

릴레이 노드는 다음과 같이 비교적 간단한 동작 패턴을 구현합니다.

- 각각 자체 암호화 키를 사용하여 하나 이상의 연결을 유지합니다.
- 받은 티켓과 현금 수상자를 확인합니다.
- 거래 수지를 모니터링하고 사전 신고 금액을 초과하면 연결을 끊습니다.
- 열린 연결에서 데이터를 수신하고 메시지 경계에서 암호 해독을 수행합니다.
- 다음과 같이 해독된 메시지를 처리합니다.
  - 비제어 세그먼트를 메시지에 지정된 연결로 전달합니다.
  - 다음과 같이 제어 세그먼트를 처리합니다.
    - *더미 데이터.* 이 세그먼트를 삭제하도록 릴레이에 지시합니다.
    - *해당 속도로 굽기.* 고정 속도로 연결을 통해 데이터를 전송하고, 패킷을 대기열에 넣고 속도를 유지하는 데 필요한 데이터를 생성하도록 릴레이에 지시합니다.
    - *래칫 티켓.* 나에게 이 패킷을 보낸 피어에게 티켓을 전달하도록 릴레이에 지시합니다.
    - *연결 시작.* 새 연결을 설정하도록 릴레이에 지시합니다. 설정 과정 및 연결 해제를 처리하는 데 사용됩니다.
    - *초기 웹 연결.* (프록시만 해당.) 지정된 호스트에 대한 SSL 연결을 열도록 프록시에 지시합니다. 화이트리스트를 지원하려면, 이것은 원시 IP 주소가 될 수 없습니다.

위의 동작에서 중요한 고려 사항은 지속적으로 릴레이에 요구되는 작업 증명은 없다는 것입니다. WebRTC 연결인 모든 연결과 결합할 경우 순수한 자바 스크립트 릴레이 코드를 실행하여 방문자에게 잠재적인 수익을 창출해 줄 수 있는 웹 사이트의 문을 열어둘 수 있습니다.

애플리케이션별 제어 세그먼트를 통해 가능한 확장에 대한 설명은 섹션 10 을 참조하십시오.

가드 노드 및 대역폭 굽기

고객이 연결된 릴레이에는 고객의 IP 주소와 같은 매우 중요한 정보가 있습니다. 우리는 고객이 가능한 한 이것을 비공개로 유지하기를 원하므로 기본 클라이언트는 장기적인 피어에 대한

선호를 첫 번째 홉으로 표시합니다.

담합에서 비롯된 정보 공격에 대한 논의에서 깊이 있게 설명한 첫 번째 홉의 노드에 대한 또 다른 관심사(섹션 B.1)는 타이밍 공격을 수행하기에 이상적인 위치에 있다는 점입니다.
이러한 공격을 막기 위해, 개인 정보 보호를 중시하는 사용자라면 *대역폭 굽기*(고정된 양의 대역폭을 고객에게 보내기 위해 두 번째 홉을 지불)라고 부르는 방법을 사용할 것을 권장합니다. 이 방법이 네트워크 사용과 전혀 관련이 없는 데이터 사용량을 초래하므로, 이 방법은 릴레이 3 의 인바운드 트래픽을 볼 수 없는 공격자가 수행하는 타이밍 공격을 방지합니다.

회피를 원하는 사용자들에게 도움을 제공하기 위해(섹션 B.3), 대역폭 굽기는 인기 있는 비 Orchid WebRTC 프로토콜의 통계적 속성에 의해 결정되는 변동 요율도 지원할 것입니다.

체인 형성

익명 인터넷 액세스를 위한 릴레이 사용에 관심이 있는 고객은 위의 사양을 사용하여 릴레이의 체인 을 형성합니다.

8. 성능 확장

이 섹션에서는 사용자 수가 증가함에 따라 시스템이 작동하는 방식을 살펴보겠습니다.

알고리즘 성능

Orchid Protocol 에는 크게 이더리움 기반 결제, 매니폴드, Orchid Market 의 세 부분이 있습니다.

이더리움 기반 결제는 일반 거래로서의 이더리움과 함께 확장합니다. 이더리움 시스템 설계를 검토한 결과, Orchid Network 가 매우 성공적이고 이더리움 총 거래량의 상당 비중을 차지하게 되더라도 이 구성 요소는 설계 공차 내에서 작동할 것으로 확신합니다.

매니폴드는 대역폭 판매자(릴레이 및 프록시)의 체인으로, 모두 Orchid Network 참가자의 총 수에 관계없이 성능 특성을 갖습니다.

Orchid Market 의 핵심 운영은 잘 연구된 Chord DHT 를 기반으로 합니다. 페들러가 유지해야 하는 연결 수는 $O(log(n))$의 속도로 최대 256 개의 연결로 증가합니다. 네트워크상의 쿼리는 $O(log(n))$개의 홉을 요구합니다. 네트워크의 크기가 커짐에 따라 이러한 작업이 더 많은 부담이 되지만 성능에 큰 영향을 줄 것이라고는 생각하지 않습니다.

희소 자원의 할당

Orchid Protocol 은 토큰을 중심으로 구축되었습니다. 이러한 토큰을 사용하면 가격 발견을 통해 구매자와 판매자 간의 균형 변화를 정상적으로 처리할 수 있습니다.

예를 들어, 모든 고객에게 느린 경험을 제공하지 않으며 릴레이가 부족한 경우, 부족한 부분이 해결될 때까지 시스템을 사용할 수 있는 사람을 결정하기 위해 고객은 입찰 전쟁에 참여하게 됩니다. 반대로, 릴레이의 공급이 넉넉한 경우 일부 릴레이는 가격이 상승하여 수급이 균형을 이룰 때까지 시스템을 떠나 있을 수 있습니다.

실제 성능

소프트웨어가 아직 완성되지 않았으므로 여기에 제공할 구체적인 수치가 없습니다. 출시하는 시점에서 이 문서를 다음 그래프로 업데이트합니다.

1. Orchid Market 크기의 함수로서의 체인 설정 시간.
2. Orchid Market 크기의 함수로서의 Orchid Market 참여 시간.
3. 가격이 희소성, 풍부함에 적응하는 속도.
4. 흥미로운 아이디어를 여기에 추가하십시오!

9. 외부 라이브러리

Orchid 의 기능은 몇 가지 중요한 기본 요소를 기반으로 합니다. 일부 독자는 이러한 기본 요소에 익숙하지 않거나 Orchid Network 에 사용된 특정 속성에 익숙하지 않을 수 있으므로 여기에 간단히 요약합니다.

WebRTC

WebRTC[47]는 원래 웹 브라우저 간의 실시간 통신을 원활하게 하도록 설계된 시스템입니다. STUN, ICE, TURN 및 RTP-over-TCP 를 포함한 NAT 및 방화벽 통과 방법의 탁월한 구현을 제공합니다. 맞춤 코딩된 TCP 및 UDP 네트워킹 코드가 아닌 네트워킹 프로토콜의 기초로 WebRTC 를 선택함으로써, 우리는 이러한 기술을 세계적 수준으로 구현하고 사용자의 트래픽을 일반적인 웹 트래픽으로 (어느 정도) 가려냅니다.

NACL

NaCL[48](소금 으로 발음)은 Daniel J. Bernstein 등이 개발한 암호화 라이브러리로, 고급 암호화 도구를 작성하는 데 필요한 핵심 작업을 작성하는 데 중점을 둡니다. NaCL 자체와 저자의 강력한 평판으로 인해 이 프로젝트에서 암호화 기본 요소를 위한 소스로 선택되었습니다. 아래 설명된 모든 암호화 작업은 이더리움 스마트 계약 암호화 코드 외에 NaCL 을 사용하여 구현됩니다.

이더리움

이더리움[55]은 기본 통화(ETH) 및 튜링 완전(Turing-complete) 스마트 계약을 포함하는 분산형 블록체인 및 플랫폼입니다. 스마트 계약은 Orchid 설계에 매우 유용한 것으로 입증되었으며, 이를 통해 지불 잔고 추적과 Orchid 지불 티켓의 검증 및 공정성과 관련된 수많은 설계 문제를 해결할 수 있습니다.

10. 미래의 작업

이 섹션의 항목은 향후 유망할 것으로 기대되는 기능과 대중을 향한 출시를 두고 내부적으로 이견이 갈리고 있는 기능, 두 가지 범주로 구분됩니다. 우리는 이러한 갈등이 보편적인 현상이라고 믿습니다. 거의 모든 사람들이 억압을 위해 권력이 사용된 사례를 지지하고 있지만, 선을 위해 권력이 사용되는 사례 또한 무수히 있습니다. Orchid 와 같은 프로토콜은 자체적인 판단력이 없으므로, 지금 라우팅하고 있는 트래픽이 자유를 위해 싸우는 투사 또는 영웅을 위한 것인지 테러리스트나 악당을 위한 것인지 인식할 수 없습니다.

공간 증명

섹션 5 에서 언급했듯이 우리는 대안적인 증명 방식을 모색하는 데 큰 관심이 있습니다. 이는 작업 증명 시스템이 환경에 미치는 영향과 현재의 작업 증명 알고리즘이 초고성능 컴퓨터를 네트워크 라우터로 작동시킬 필요 때문에 중요한 문제입니다.
우리는 디스크 공간을 사용하는 것이 보안의 핵심 영역에서 희소한 자원이 될 수 있는 가능성을 발견한 것을 기뻐하고 있습니다. 그에 따라 구형 전화기나 유사한 하드웨어가 Orchid 네트워크에 수익성에 기여하는 방식으로 참여할 수 있습니다.

콘텐츠 호스트 보호

이전의 많은 접근 방식(섹션 2)을 통해 콘텐츠 호스트가 웹 사용자와 유사한 보호 방식을 모색하고 있음을 밝혀냈습니다. 우리는 자유롭게 배포하지 않은 것이 공공의 이익에 부합하는 콘텐츠(예: 핵무기 제조 관련 정보)도 있다고 생각하기 때문에, 이 지점에서 내부적으로 이견이 갈리고 있습니다. 그러나 예상치 못한 상황에 따라 필요할 경우 다음 다이어그램과 같이 제한되지 않고 감시되지 않는 호스트 를 지원하도록 Orchid 를 확장할 수 있습니다.

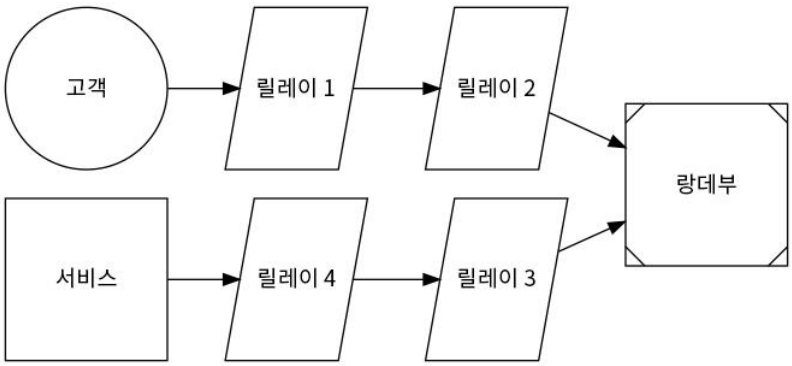

그림 3: 서비스와 고객 간의 릴레이 역할을 하는 랑데부 노드

이더리움 트래픽 보호

방화벽 회피에 관한 섹션(섹션 B.3)에서 설명한 바와 같이, 클라이언트의 이더리움 네트워크 트래픽은 약한 고리일 가능성이 있습니다. 모든 노드가 이 정보를 유지해야 하므로 Orchid Protocol 을 사용하여 이더리움 정보를 배포하는 것은 자연스러운 것처럼 보입니다.

불행히도 지불에 대한 정보를 지불하는 사람들에게 의존하는 것은 까다로운 문제로 이어지게 됩니다. 가까운 시일 내에 이를 추가할 예정이지만 초기 릴리스에는 포함시키지 않을 것입니다.

플랫폼으로서의 Orchid

머지 않은 미래에 핵심 시스템의 설계에 미래에 많은 시간이 소요될 것으로 예상되지만, 다음 사용 사례를 지원하는 기능을 추가함으로써 Orchid Network 를 통해 라우팅되는 대역폭의 양이 크게 증가할 가능성에 매우 큰 관심을 두고 있습니다.

1. 웹 사이트가 네트워크에 직접 인터페이스하고 서비스에 토큰을 통합하기 위한 API.
2. 네트워크상 파일 저장 및 정적 웹 사이트 호스팅.
3. 파일 공유.
4. 이메일/메시징 서비스.
5. 중재/조정 서비스.

참고 문헌

[1] URL: http://www.meshlabs.org.
[2] URL: https://en.wikipedia.org/wiki/Payment_service_provider.
[3] URL: https://en.wikipedia.org/wiki/ISO/IEC_7816.
[4] URL: http://www.ebics.org/home-page.
[5] URL: https://www.swift.com.
[6] URL: http://www.nyce.net/about.
[7] URL: http://www.investopedia.com/terms/r/reconciliation.asp.
[8] URL: https://www.quora.com/What-are-common-credit-card-processing-fees.
[9] URL: https://www.nerdwallet.com/blog/banking/wire-transfers-what-banks-charge.
[10] URL: https://www.economist.com/blogs/dailychart/2010/12/remittances.
[11] URL: https://www.valuepenguin.com/what-credit-card-processing-fees-costs.
[12] URL: https://bitcoin.org/en/developer-guide#transactions.
[13] URL: https://bitinfocharts.com/comparison/bitcoin-transactionfees.html.
[14] URL: https://bitinfocharts.com/comparison/ethereum-transactionfees.html.
[15] URL: https://blog.ethereum.org/2015/07/05/on-abstraction.
[16] URL: https://blog.ethereum.org/2015/12/24/understanding-serenity-part-i-abstraction.
[17] URL: https://github.com/ethereum/EIPs/issues/662#issuecomment-312709604.
[18] URL: https://en.bitcoin.it/wiki/Address_reuse.
[19] URL: https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki.
[20] URL: https://ropsten.etherscan.io/address/0x5e10d764314040b04ac7d96610b9851c8bc02815# code.
[21] URL: https://p16.praetorian.com/blog/man- in- the- middle- tls- ssl- protocol- downgrade- attack.
[22] URL: http://ethgasstation.info.
[23] URL: https://etherscan.io/token/OmiseGo.
[24] URL: https://solidity.readthedocs.io/en/develop/assembly.html.
[25] URL: https://hackernoon.com/zksnarks-and-blockchain-scalability-af85e350a93a.
[26] URL: https://hackernoon.com/scaling-tezo-8de241dd91bd.
[27] URL: http://mojonation.net.
[28] URL: https://en.bitcoin.it/wiki/Payment_channels.
[29] URL: https://en.bitcoin.it/wiki/Contract.
[30] URL: https://lightning.network/lightning-network-paper.pdf.

[31] URL: https://en.wikipedia.org/wiki/Mining_pool#Mining_pool_methods.
[32] URL: https://bitcoin.stackexchange.com/questions/1505/what-is-a-share-can-i-find-it- while-mining-solo-or-only-when-pool-mining.
[33] URL: https://blog.coinbase.com/app-coins-and-the-dawn-of-the-decentralized-business- model-8b8c951e734f.
[34] URL: http://onchainfx.com.
[35] URL: https://0xproject.com/.
[36] URL: https://etherdelta.com/#REQ-ETH.
[37] URL: https://augur.net.
[38] URL: https://gnosis.pm.
[39] URL: https://medium.com/@vishakh/a- deeper- look- into- a- financial-derivative- on- the- ethereum-blockchain-47497bd64744.
[40] URL: https://tools.ietf.org/html/draft-goldbe-vrf-00.
[41] URL: https://blog.ethereum.org/2017/10/12/byzantium-hf-announcement.
[42] URL: https://github.com/ethereum/EIPs/pull/213.
[43] URL: https://github.com/ethereum/EIPs/pull/212.
[44] URL: https://theethereum.wiki/w/index.php/ERC20_Token_Standard.
[45] Martin Abadi 외. Moderately hard, memory-bound functions . 출전: *ACM Transactions on Internet Technology (TOIT)* 5.2 (2005 년), p.299–327.
[46] Iddo Bentov, Rafael Pass 및 Elaine Shi. Snow White: Provably Secure Proofs of Stake. *출전: IACR Cryptology ePrint Archive* 2016 (2016 년), p.919.
[47] Adam Bergkvist 외. Webrtc 1.0: Real-time communication between browsers . 출전: *Working draft, W3C* 91 (2012 년).
[48] Daniel Bernstein, Tanja Lange 및 Peter Schwabe. The security impact of a new cryptographic library . 출전: *Progress in Cryptology–LATINCRYPT 2012* (2012 년), p.159–176.
[49] Jean-Luc Beuchat 외. High-Speed Software Implementation of the Optimal Ate Pairing over Barreto–Naehrig Curves . 출전: *4th International Conference on Pairing-Based Cryptography*. Springer. 2010 년. URL: https://eprint.iacr.org/2010/354.pdf.
[50] Alex Biryukov 와 Dmitry Khovratovich. Equihash: Asymmetric proof-of-work based on the generalized birthday problem . 출전: *Ledger* 2 (2017 년).
[51] N. Bitansky 외. Recursive composition and bootstrapping for SNARKs and proof-carrying data . 출전: *In Proceedings of the forty-fifth annual ACM symposium on Theory of computing*. ACM. 2013 년, p.111–120. URL: https://eprint.iacr.org/2012/095.pdf.
[52] Nikita Borisov 외. Denial of service or denial of security? 출전: *Proceedings of the 14th ACM conference on Computer and communications security*. ACM. 2007 년, p.92–102.
[53] Michael Brown 외. Software implementation of the NIST elliptic curves over prime fields . *출전: Topics in Cryptology—CT-RSA 2001*. Springer. 2001 년, p.250–265. URL: https://pdfs.semanticscholar. org/ac3c/28ebf9a40319202b3c4f64cc81cdaf193da5.pdf.
[54] David Brumley 와 Dan Boneh. Remote timing attacks are practical . 출전: *Computer Networks* 48.5 (2005 년), p.701–716.
[55] Vitalik Buterin. *Ethereum: A next-generation smart contract and decentralized application platform.* https://github.com/ethereum/wiki/wiki/White-Paper. 액세스 날짜: 2016 년 8 월 22 일. 2014 년. URL: https: //github.com/ethereum/wiki/wiki/White-Paper.
[56] David L Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms . 출전: *Communications of the ACM* 24.2 (1981 년), p.84–90.
[57] Shuo Chen 외. Side-channel leaks in web applications: A reality today, a challenge tomorrow . *출전: Security and Privacy (SP), 2010 IEEE Symposium on*. IEEE. 2010 년, p.191–206.

[58] Alessandro Chiesa 외. Decentralized Anonymous Micropayments . 출전: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2017 년, p.609–642.
[59] George Danezis. The Traffic Analysis of Continuous-Time Mixes . 출전: *Proceedings of Privacy Enhanc- ing Technologies workshop (PET 2004)*. Vol. 3424. LNCS. 2004 년 5 월, p.35–50.
[60] Roger Dingledine, Nick Mathewson 및 Paul Syverson. Tor: *The second-generation onion router*. 기술 대표 Naval Research Lab Washington DC, 2004 년.
[61] Cynthia Dwork, Moni Naor 및 Hoeteck Wee. Pebbling and proofs of work . 출전: *CRYPTO*. Vol. 5. Springer. 2005 년, p.37–54.
[62] Kevin P Dyer 외. Peek-a-boo, i still see you: Why efficient traffic analysis countermeasures fail . 출전: *Security and Privacy (SP), 2012 IEEE Symposium on*. IEEE. 2012 년, p.332–346.
[63] Stefan Dziembowski 외. Proofs of space . 출전: *Annual Cryptology Conference*. Springer. 2015 년, p.585–605.
[64] Christoph Egger 외. Practical attacks against the I2P network . 출전: *International Workshop on Recent Advances in Intrusion Detection*. Springer. 2013 년, p.432–451.
[65] Ittay Eyal 과 Emin Gu¨n Sirer. Majority is not enough: Bitcoin mining is vulnerable . 출전: *International conference on financial cryptography and data security*. Springer. 2014 년, p.436–454.
[66] Mike Hearn. [*Bitcoin-development] Anti DoS for tx replacement*. Bitcoin development mailing list. 2013 년. URL: https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2013-April/002417.html.
[67] Martin Hellman. A cryptanalytic time-memory trade-off . 출전: *IEEE transactions on Information Theory* 26.4 (1980 년), p.401–406.
[68] J Herrera-Joancomart´ı. Research and challenges on bitcoin anonymity . 출전: *In Data Privacy Management, Autonomous Spontaneous Security, and Security Assurance*. Springer. 2015 년, p.3–16. URL: https://www.researchgate.net/profile/Jordi_Herrera-Joancomarti/publication/281773799_Research_and_Challenges_on_Bitcoin_Anonymity/links/55f7c7d408ae07629dcbc471.pdf.
[69] Douglas R. Hofstadter. *Metamagical Themas: Questing for the Essence of Mind and Pattern*. New York, NY, USA: Basic Books, Inc., 1985 년. ISBN: 0465045405.
[70] Nicolas Houy. It Will Cost You Nothing to'Kill'a Proof-of-Stake Crypto-Currency . 출전: *Browser Download This Paper* (2014 년).
[71] Don Johnson, Alfred Menezes 및 Scott Vanstone. The elliptic curve digital signature algorithm (ECDSA) . 출전: *International Journal of Information Security 1, no. 1 (2001 년)*. Springer, p.3–63. URL: http://residentrf.ucoz.ru/_ld/0/34_Digital_Signatu.pdf.
[72] Aggelos Kiayias 외. Ouroboros: A provably secure proof-of-stake blockchain protocol . 출전: *Annual International Cryptology Conference*. Springer. 2017 년, p.357–388.
[73] Joshua A. Kroll, Ian C. Davey 및 Edward W. Felten. The economics of Bitcoin mining, or Bitcoin in the presence of adversaries . 출전: *Proceedings of WEIS (Vol. 2013)*. WEIS. 2013 년, p.11. URL: http: //www.thebitcoin.fr/wp-content/uploads/2014/01/The-Economics-of-Bitcoin-Mining-or-Bitcoin-in-the-Presence-of-Adversaries.pdf.
[74] Thomas Locher 외. Free riding in BitTorrent is cheap . 출전: *Proc. Workshop on Hot Topics in Networks (HotNets)*. 2006 년, p.85–90.

[75] Daniel Lorimer. *Momentum–a memory-hard proof-of-work via finding birthday collisions, 2014 년*. url: http://www.hashcash.org/papers/momentum.pdf.
[76] Sarah Meiklejohn 과 Rebekah Mercer. Möbius: Trustless Tumbling for Transaction Privacy . 출전: 2017 년. URL: https://allquantor.at/blockchainbib/pdf/meiklejohn 2017moebius.pdf.
[77] Adnan Noor Mian 외. Enhancing communication adaptability between payment card processing networks . 출전: *IEEE Communications Magazine, 53(3)*. IEEE. 2015 년, p.58–64.
[78] Satoshi Nakamoto. *Bitcoin: A peer-to-peer electronic cash system*. 2008.
[79] Arvind Narayanan 외. *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction.* Princeton University Press, 2016 년.
[80] Sunoo Park 외. *Spacecoin: A cryptocurrency based on proofs of space*. 기술 대표 IACR Cryptology ePrint Archive 2015, 2015 년.
[81] Rafael Pass 와 Abhi Shelat. Micropayments for Decentralized Currencies . 출전: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM. 2015 년. URL: https: //pdfs.semanticscholar.org/bca9/92b35d844160b30edbbf1809e17551d867ea.pdf.
[82] Ronald L Rivest. Electronic Lottery Tickets as Micropayments . 출전: *International Conference on Financial Cryptography.* Springer. 1997 년. URL: http://citeseerx.ist.psu.edu/viewdoc/download? doi=10.1.1.135.2668&rep=rep1&type=pdf.
[83] David L. Salamon 외. How to make Chord correct . 출전: (2017 년). URL: https://Orchidprotocol. com/whitepaper.pdf.
[84] Rabin Silvio 와 Vadhan. Verifiable Random Functions . 출전: *Foundations of Computer Science, 1999. 40th Annual Symposium on*. IEEE. 1999 년. URL: https://dash.harvard.edu/bitstream/handle/1/ 5028196/Vadhan_VerifRandomFunction.pdf.
[85] Ion Stoica 외. Chord: A scalable peer-to-peer lookup service for internet applications . 출전: *ACM SIGCOMM Computer Communication Review* 31.4 (2001 년), p.149–160.
[86] John Tromp. Cuckoo Cycle: a memory-hard proof-of-work system. 출전: *IACR Cryptology ePrint Archive* 2014 (2014 년), p.59.
[87] Liang Wang 과 Jussi Kangasharju. Real-world sybil attacks in BitTorrent mainline DHT . 출전: *Global Communications Conference (GLOBECOM), 2012* IEEE. IEEE. 2010 년, p.826–832.
[88] David Wheeler. Transactions using bets . 출전: *International Workshop on Security Protocols. Springer.* 1996 년, p.89–92.
[89] Gavin Wood. ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER. URL: https://ethereum.github.io/yellowpaper/paper.pdf.
[90] Pamela Zave. Orchid: A Fully Distributed, Anonymous Proxy Network Incentivized Through Band- width Mining . 출전: *arXiv preprint arXiv:1502.06461* (2015 년).

A. 경매

대역폭을 구매할 때 가격에 민감한 고객은 매우 저렴한 가격을 제공하는 공격자에게 이용당할 우려가 있습니다. 예를 들어, 가장 낮은 입찰 기준으로 길이가 4 인 체인을 구매하려는 고객을 상상해 봅시다. 이러한 사정을 아는 공격자는 가격을 가능한 최소 금액으로 설정하여 체인의 각 노드에 대해 선택될 확률을 $\frac{a}{n}$ 이상으로 높일 수 있습니다.

이러한 문제를 해결하기 위해 Orchid Market 을 사용하는 고객은 대역폭 판매 집단의 무작위의 하위 집합을 선택한 다음 해당 임의의 하위 집합에서 만들 수 있는 필요한 길이의 저렴한 체인의 집합으로부터 제공자를 무작위로 선택합니다.

그 결과, 공격자가 할당된 위치와 무작위로 선택한 다른 판매자가 설정한 가격과 비교해 선택한 가격 면에서 모두 운이 좋아야만 하는 상황이 됩니다. 이러한 제약에도 불구하고, 판매자가 가격 변동을 통해 수요에 영향을 미치는 능력과 같은 익숙한 시장 속성이 유지됩니다.

A.1. 부록 개요

이 부록에서는 프라이버시를 중시하는 대역폭 구매자가 사용할 수 있는 전략과 다양한 전략의 성능에 대해 살펴보겠습니다.

이 문제의 분석에 적합한 Orchid Market 의 단순화된 모델이 도입되었으며, 이론과 구체적인 예제 모두에서 몇 가지 예제 접근법이 연구되었습니다. 이 섹션의 독자가 대역폭 구매 알고리즘을 선택할 때 만든 거래에 대해 이해하고 다른 방법을 사용하기 위해 클라이언트를 하드 코딩하지 않는 것이 좋습니다.

A.2. 분석을 위한 단순화된 모델

경매 전략을 평가하기 위해 Orchid Market 의 전체적인 복잡성을 고려할 필요는 없습니다. 간단한 분석을 위해 다음의 참가자 목표에 대한 일련의 가정을 소개합니다.

1. **판매자**. 판매자는 액세스를 판매하려는 r 대역폭 슬롯 을 소유합니다. 판매자의 유일한 목표는 이 소스에서 수익을 극대화하는 것입니다.
2. **공격자.** 공격자는 $a \geq 2$ 판매자를 보유합니다. 공격자의 유일한 목표는 단일 구매자가 판매자로부터 하나 이상의 슬롯을 구매하도록 하는 것이며, 이를 '성공적인 공격'이라고 합니다.
3. **구매자.** 구매자는 가능한 최저 가격으로 3 명의 다른 판매자로부터 3 개의 대역폭 슬롯을 구매하려 합니다(따라서 공격자를 방해).

우리는 Orchid Market 의 세부 사항에 관심을 두기보다는 모든 구매자가 모든 현재 메달리온 보유자 및 현재 대역폭 가격의 최신 목록을 보유하고 있다고 가정합니다. 또한 릴레이와 프록시의 구별 또는 화이트리스트 및 기타 기능 필터링으로 인한 복잡성에 대해서는 신경 쓰지 않을 것입니다.

이제 기본 구조를 확보했고 참가자 목표를 이해했으므로 게임 구조를 구체화해 보겠습니다.

1. **설치.** 구매자가 사용할 전략이 모든 판매자와 모든 공격자에게 알려집니다. 구매자 전략에 따라 변경될 수 있는, 판매자가 사용할 전략이 공격자에게 알려집니다.
2. **1 단계.** 모든 판매자가 가격을 선택합니다. 그런 다음 판매자 가격이 공격자에게 알려지고 공격자가 자체적인 가격을 선택합니다.

3. **2 단계.** 모든 가격이 구매자에게 공개됩니다. 구매자가 목록에서 사용 가능한 오퍼를 최대 3 개까지 선택하도록 무작위 순서로 요청 받습니다.
4. **3 단계.** 이익이 분배됩니다.
   (a) 판매자와 공격자는 자신을 선택한 구매자로부터 제안 가격과 동일한 금액을 받습니다.
   (b) 구매자가 공격을 받지 않고 3 개의 슬롯을 구매한 경우, 구매자는 각 구매자별로 특정된 이익 금액에서 해당 슬롯에 대해 지불된 가격을 공제한 금액을 받습니다.
   (c) 구매자를 성공적으로 공격한 공격자는 구매자별 현상금 U_b를 받습니다.

시장은 우리가 다음의 세 가지 전략을 추구하는, n 명의 참가자로 이루어진 게임입니다. 구매자가 해야 할 일, 판매자가 해야 할 일, 공격자가 해야 할 일. 구매자가 자신의 요구에 비추어 경제적으로 전혀 정상적이지 않은 방식을 취하기 때문에, 우리는 위의 설계에서 의도적으로 구매자에게 우선 순위 를 부여했습니다.

일부 독자는 위의 게임에서 구매자의 전략이 공격자에게 공유된다고 생각하여 문제를 제기할 수 있습니다. 우리가 명시적으로 이러한 방식을 취하는 이유는, 동기를 지닌 공격자가 최초 공격에 성공하지 못하더라도 특정 구매자가 지불한 대략적인 가격에 관한 정보를 확보할 수 있기 때문입니다(예: 모든 판매자의 IP 주소 및 가격을 결정한 다음 해당 구매자의 인터넷 연결을 모니터링하여 첫 번째 홉에서 판매자가 어느 대역폭을 사용했는지 판단하는 경우). 시간이 지남에 따라 이러한 정보 유출을 이용하여, 사용되는 전략을 유추할 수 있으므로, 이 분석의 목적에 비추어, 공격자가 해당 정보를 보유하고 있다고 가정하는 것이 가장 좋습니다.

성공 기준

성공적인 솔루션은 다음 기준에 따라 성능이 결정됩니다.

1. **보안.** 고정적인 예산과 체인 길이로 고객은 공격으로부터 최대한 보호됩니다.
2. **안정성.** 세 그룹 중 어느 구성원도 전략을 변경하여 개인 유틸리티를 향상시킬 수 없습니다.
3. **경제적 호환성.** 판매자는 구매 주문 수를 조절하기 위해 가격을 올리거나 내릴 수 있으므로 판매자 이익을 극대화하기 위해 익숙한 방법을 사용할 수 있습니다.

그룹의 구성원들이 자신들의 이해를 전체적으로 보호하기 위해 함께 결속하는 그룹 수준의 최적 전략 을 제안하려는 상당한 유혹이 있기 때문에 최적화에 익숙하지만 게임 이론에 익숙하지 않은 독자들에게는 안정성에 특별한 주의를 기울여야 합니다. 그러한 행동은 표면적으로는 합리적으로 보일지 모르지만, 완전 익명의 분산된 시장에 존재하는 인센티브 구조로 인해 안정화되기 어렵습니다. 예를 들어, 제공되는 총 슬롯이 수요를 초과하는 상황에서 판매자가 서로 담합하여 최소 가격을 설정하는 상황이 발생할 수 있습니다(경제 용어로 *트러스트*, 게임 이론 용어로 초합리성(Superrationality)[69], 마르크스주의 용어로는 계급 혁명). 불행히도, 가격을 신탁 가격 이하로 낮추는 판매자는 추가 이익을 취할 수 있기 때문에 그러한 합의는 안정적이지 않습니다. 따라서 우리는 이 분석에서 그러한 사항을 고려하지 않을 것입니다. 이것은 이 도메인에서 최종 적용을 배제하는 것이 아닙니다. 아마도 블록체인 기술의 향후 발전은 그러한 계약 준수에 대한 분산화된 검증을 가능하게 할 것입니다.

또 다른 잠재적 놀라움은 구매자를 최대한 활용하려면 공격자가 어떻게 행동해야 할지 명확히 판단하고 성공 기준과 같은 전략의 안정성을 포함시킨다는 것입니다. 주어진 접근 방식의 보안에 대한 경계를 제공하는 다른 방법이 없기 때문에 우리는 이 방식을 따릅니다.

A.3. 선택 공격

구매자 전략에 대해 너무 깊이 살펴보기 전에 먼저 공격자의 목표 및 공격을 구성하는 요소를 살펴보겠습니다. 우리가 무한한 예산을 가지고 판매자 목록에 포함된 것 이외의 정보가 없는 완벽한 편집증 구매자를 상상한다면, 그런 구매자에게는 명백히 무작위 선택보다 더 좋은 것이 없을 것입니다. 이를 통해 공격자에게 $\left(\frac{a}{n}\right)^2$ *의 성공적인 공격 속도를 제공하게 됩니다.* 이것이 최선의 결과이기 때문에 우리는 그러한 공격 가능성에 대처할 수 있는 전략을 개발합니다.

이 영역에 대한 공격은 공격자가 공격 성공률을 $\left(\frac{a}{n}\right)^2$의 확률 이상으로 올릴 수 있는 방법입니다.

A.4. 후보자 전략

이제 지금까지의 조건에서 벗어나 구매자 전략을 평가하겠습니다.

최저 가격

구매자가 가장 낮은 비용의 공급자를 선택한다면 구성 요소 공격자는 가격을 가장 낮은 값(0 토큰)으로 설정하게 됩니다. 그럼으로써 a 윗 성공적인 공격을 야기하게 됩니다(여기서 z 는 0 를 청구하는 노드에 해당하는 수임).

예를 들어, 가격대별로 단일 슬롯을 제공하는 3 명의 진실된 판매자가 있는 시장을 생각해 보십시오. $\{2, 4, 6\}$, 최대한 지불할 수 있는 총 가격이 12 인 구매자. 유능한 공격자는 $\{0, 0\}$의 가격에 시장에 진입할 것이며, 따라서 거래가 반드시 성사될 것이므로 공격 성공 확률은 1 이 됩니다.

가격 가중치 무작위성

구매자가 대역폭 비용 차이의 단조 함수를 구입할 기회를 선택한다면 다음과 같은 성능이 약간 향상될 것입니다.

$$\frac{af(0)}{\sum_{e \in S} f(e)}$$

위의 예로 돌아가서 비용의 역제곱 증가를 함수로 사용하면 공격 성공 확률은 $\frac{288}{337} \approx 85\%$가 될 것입니다. 85%가 100%보다 훨씬 낫긴 하지만 여전히 불만족스럽습니다.

저렴한 릴레이에서 무작위 선택

구매자가 최대 가격의 $\frac{1}{3}$보다 적은 금액을 청구하는 판매자 중에서 무작위로 선택하기로 할 경우, 구성 요소 공격자는 가격을 최대 값 이하로 설정할 것입니다. 확신이 없을 경우 공격자는 다시 가격을 0 으로 선택할 수 있습니다.

위의 예로 돌아가서, 구성 요소 공격자는 0 과 1 또는 그와 동등한 가격으로 시장에 진입하여 다음과 같은 2 가지의 비공격 조합을 야기합니다. $\{(0, 2, 4), (1, 2, 4)\}$ 및 두 가지 공격 조합: $\{(0, 1, 2)$ 및 $(0, 1, 4)\}$. 이로써 공격 성공 확률은 $\frac{1}{2}$이 됩니다.

비용에 따른 무작위 선택

구매자가 3 단 조합 (S_i, S_j, S_k)인 판매자로부터 무작위로 선택하여 총 비용이 최대 비용보다 낮거나 동등해진다면, 공격자는 (1) 구매자가 감당할 수 있는 3 단 조합 (A_i, A_j, S_k)의 수를 최소화하고 (2) 구매자가 감당할 수 있는 3 단 조합 (A_i, S_j, S_k)의 수를 최소화하는 방향으로 가격을 선택할 수 있습니다.

위의 예로 돌아가서, 유능한 공격자는 1 과 4.1 또는 그와 동등한 가격으로 시장에 진입하여 다음과 같은 5 가지의 비공격 조합을 야기합니다. $\{(1, 2, 4), (1, 2, 6), (1, 4, 6), (2, 4, 4.1), (2, 4, 6)\}$ 및 3 가지 공격 조합: $\{(1, 2, 4.1), (1, 4, 4.1), (1, 4.1, 6)\}$ 결과적으로 공격 성공 확률은 $\frac{3}{8}$입니다.

공격자가 4.1 가격대를 선택함으로써 달성하는 효과적인 밀어내기 효과 에 주목하십시오. 4.1 을 포함한 4 가지 조합 중 하나만이 성공적인 공격이 아닙니다. 공격자 측의 그러한 행동을 규명한 후에도 비용에 따른 무작위 선택은 여전히 6 을 청구하는 판매자를 배제하는 것보다 낫습니다.

피어에 의해 정규화된 무작위 비용 선택

자연스럽게 다음과 같은 질문을 해 볼 수 있습니다. 피어가 과소 표시되지 않도록 무작위 표본에 편기를 적용하면 어떻게 됩니까? 안타깝게도 공격자는 이를 본인에게 유리하게 활용할 수 있습니다.

위의 예를 계속 살펴볼 때, 구성 요소 공격자는 1 또는 6.1 또는 그와 동등한 가격으로 선택할 것입니다. 구매자가 1 과 쌍을 이루는 경우 6.1 만 감당할 수 있으므로 확률을 조정하면 공격 성공 확률은 $\frac{3}{7}$이 됩니다.

쌍으로 정규화된 무작위 비용 선택

위와 비슷한 다음과 같은 또 다른 질문이 가능합니다. 선택되는 판매자의 *쌍*에 대한 확률을 정규화하기 위해 무작위 표본에 편기를 적용하면 어떻게 됩니까? 공격자가 희귀한 쌍을 제어함으로써 또 다시 공격 성공 가능성을 높입니다.

위의 예를 계속 살펴볼 때, 구성 요소 공격자는 다시 1 과 6.1 또는 그와 동등한 가격을 선택하여 이번에는 $\frac{24}{49}$의 성공률을 실현할 것입니다. 왜냐하면 6.1 을 포함한 쌍이 상당히 과소 표시되었기 때문입니다.

A.5. 안정성 분석

이제 몇 가지 후보적인 접근 방식을 설명했으므로 다음과 같은 문제가 발생합니다. 구매자는 주어진 전략에서 벗어나도록 장려되고, 그렇다면 공격자가 다양한 구매자 집단을 악용하려고 할 때 각 전략의 보안 속성에는 어떤 영향이 있습니까?

분석을 위한 분석을 피하기 위해 가격 관점과 보안 관점 모두에서 차선책인/불안정한 전략에 대한 섹션을 생략했습니다.

최저 가격

선택한 가격이 이미 가능한 최저 가격이므로 최저 가격은 경제적 인센티브에 비해 안정적입니다. 그러나 보안 관점에서 보면 차선책이며 불안정합니다. 보안을 염두에 둔 구매자는 다른 전략, 아마도 비용에 따른 임의 선택을 사용할 것입니다.

이로 인해 다음 섹션에서 설명된 것처럼 공격자가 두 유형의 구매자 간에 리소스를 할당하는 방법을 결정하여 양쪽 그룹의 보안 이점을 누리는 흥미로운 상황이 발생합니다.

비용에 따른 무작위 선택

이전 논의와 반대로 이 전략은 보안 관점에서는 안정적이지만 경제적 인센티브 면에서는 안정적이지 않습니다. 보안에 관심이 없는 구매자는 최저 비용 선택을 사용할 것입니다.

일부 독자들은 이 전략이 보안 관점에서 안정적이라는 주장에 놀랄 수 있습니다. 아마도 Orchid Market 의 일부 구매자는 공격자가 비용에 따라 무작위 선택을 악용하는 방법에 대한 지식에 근거하여 자신의 선택 방법을 개선하지 않겠습니까? 전술한 바와 같이, Orchid Market 은 구매 전략의 추론에 있어서 안전하지 않으며, 더 큰 문제는 공격자가 대안적인 방법을 유추해낼 수 있는 정도를 알 수 있는 방법이 없다는 것입니다. 따라서 현저히 편집증적인 구매자에게 이 방법은, 가정된 최악의 상황에서도 최선의 성과를 내므로 안정적입니다.

그러나 Orchid Market 구매자 중 일부가 이 조언을 무시하거나 단순히 최저가 선택 방법을 사용하는 만큼 보안에 민감한 구매자에게는 좋은 소식입니다. 2 차 공격 최적화를 통해 공격자는 비용에 따른 무작위 선택을 최적으로 악용하지 못하게 될 뿐입니다.

A.6. 경제적 호환성 분석

이제 판매자 전략이라는 문제에 관심을 돌릴 순서입니다. 여기서 우리의 목표는 판매자가 일반적인 종류의 경제적 알고리즘을 사용할 수 있는 정도를 설명하는 것입니다.

가격 관점과 보안 관점 모두에서 차선책인/불안정한 전략에 대한 섹션을 생략했습니다.

최저 가격

이 접근법은 경제학에서 예상되는 경우이므로 완전한 경제적 호환성이 있습니다.

비용에 따른 무작위 선택

초기에는 이 전략이 경제적 호환성을 지닌 것으로 보이지 않지만 최대 가격을 공유하지 않는 구매자 집단을 고려할 때 판매자 상품에 대한 관심 빈도는 친숙한 가격 민감도의 형태를 따릅니다.

판매자는 일반적인 방식으로 가격을 올리거나 내림으로써 구매 주문 수를 조절할 수 있으므로 비용에 따른 무작위 선택에는 경제적 호환성이 있습니다.

A.7. 결론

지금까지 Orchid Market 의 구매자에게 적합한 경매 방법에 대한 분석을 살펴보았으며 비용에 따른 무작위 선택이 일반적인 용도로 선택되는 방법을 설명했습니다.

순수한 무작위 접근 방식을 선택한 이유는 공격자가 구매자의 전략을 완전히 파악하고 있다는 가정과 합법적인 판매자가 자신의 가격을 선택한 후에 공격자가 가격을 선택할 것이라는 가정에서 비롯됩니다. 이러한 맥락에서 편기된 표본이 기여할 수 있는 부분은 전혀 없는 반면, 최악의 경우 이를 통해 공격자가 선택 기회를 늘릴 수 있습니다. 표본에 편기를 적용하지 않고 사용 가능한 옵션의 수를 극대화한 후 해당 공간에서 균일하게 선택했습니다.

보다 전통적인 경매 모델에 비해 구매자에 대한 이러한 비용 증가에 대해 우려하는 독자는 프리미엄을 보안 가격 으로 간주하는 것이 좋습니다. 앞에서 살펴보았듯이 자신을 공격에 노출시켜 최저 가격을 확보하는 것은 그 이점이 미미합니다.

B. 공격 및 보안

B.1. 체인에 대한 공모 공격

이 섹션에서는 공격자가 여러 개의 릴레이 및/또는 ISP(인터넷 서비스 사업자)를 제어하거나 모니터링하여 유추하거나 추론할 수 있는 정보의 종류를 살펴보겠습니다. 릴레이 및 프록시가 무작위로 선택되었다고 가정(따라서 ISP 도 마찬가지임)할 때, 특정 공격이 다른 체인 길이에서 수행될 수 있는 가능성에 대한 확률 모델을 작성해 보겠습니다.

개별 릴레이 및 프록시에 사용 가능한 정보

IP 기반 네트워킹의 내재적 구조와 Orchid Protocol 의 이더리움 기반 결제 사용으로 인해 릴레이 및 프록시 노드와 *이들의 IPS 가* 다음 정보에 대한 액세스 권한을 획득합니다.

- 연결된 모든 컴퓨터의 IP 주소.
- 전달하는 패킷의 크기, 타이밍 및 수.
- 패킷 비용을 지불하는 토큰을 제어하는 공개 키.
- 패킷으로 향하는 제어 세그먼트의 내용.

또한 프록시 노드와 *이들의 ISP 가* 다음 정보에 대한 액세스 권한을 획득합니다.

- 웹 서버의 호스트 이름 및 SSL/TLS 세션 협상의 plantext 부분.

공모의 잠재적 당사자

다음 역할은 고객 정보에 액세스할 수 있으므로 공격의 일부로 효과적으로 공모하거나 모니터링될 수 있습니다.

- 고객, 릴레이, 프록시 또는 웹 서버의 인터넷 서비스 사업자(ISP). s 의 확률, 신뢰할 수 없음
- 웹 사이트. 프록시가 연결된 웹 서버. w 의 확률, 신뢰할 수 없음
- 릴레이 $_n$. 체인의 n 번째 릴레이. $\frac{r}{n}$의 확률, 신뢰할 수 없음.
- 프록시. 웹 서버에 대한 프록시 릴레이 대역폭. $\frac{x}{n}$의 확률, 신뢰할 수 없음

공격자가 컴퓨팅 능력의 총량을 제어할 수 없더라도 작업 증명 계산을 활용할 수 있으므로 릴레이와 프록시 노드 간에 컴퓨팅이 할당되는 방식을 제어할 수 있기 때문에, 위의 r 와 x 를 분리했습니다.

공격의 유형

공모 공격의 주요 목표는 특정 Orchid 고객과 특정 SSL 연결을 연결하는 것입니다. 이 작업을 수행할 수 있는 두 가지 방법이 있습니다.

- 관계. 이것이 가능한 경우 공격자는 고객이 경로를 따라 충분한 지점을 관찰할 수 있기 때문에 고객이 특정 웹 사이트와 통신하고 있을 것으로 추론할 수 있습니다.

- 타이밍. 이것이 가능할 때 공격자는 패킷 타이밍을 제어한 다음 관찰하여 고객이 특정 웹 사이트와 통신하고 있을 것으로 추론할 수 있습니다.
- 언버닝(Unburning). 이것이 가능할 경우 공격자는 고객이 사용하는 대역폭 굽기에도 불구하고 타이밍 공격을 수행할 수 있습니다.

정기적인 인터넷 액세스: 제로 릴레이, 제로 프록시

고객이 웹 사이트에 직접 연결할 때 Orchid 시스템은 물론 사용되지 않지만, 나머지 분석을 좌절시키기 위해 이 설정에 어떤 정보 위험이 있는지 검토하는 것이 중요하다고 생각합니다.

ISP	웹 사이트	P(관계)	P(타이밍)	P(언번)
x		s		
	x	w		

위 표에서 X 는 공모에 참여함을 나타내며 P(관계) 및 P(타이밍)의 값은 이러한 상황이 발생할 가능성을 나타냅니다. 외부적인 X 가 있는 라인과 같이 공격이 불가능한 라인은 생략하고 보다 간단한 공격이 가능한 보다 정교한 공격에 대한 언급.

VPN: 제로 릴레이, 제로 프록시

분석을 좌절시키기 위해, VPN 액세스 고유의 공모 위험도 제시합니다.

ISP	VPN	웹 사이트	P(관계)	P(타이밍)	P(언번)
	x		g		
x		x	sw		

g가 VPN 사업자가 모니터링 당하거나 공격자와 공모할 확률인 경우. g가 예를 들어 VPN 사용의 결과로, 시간이 지남에 따라 모델링하기 어려운 방식으로 변화할 수 있습니다.

0 개의 릴레이, 1 개의 프록시

ISP	프록시	웹 사이트	P(관계)	P(타이밍)	P(언번)
	x		$\left(\frac{x}{n}\right)$		
x		x	sw		

이 경우의 위험이 VPN 사용에 따른 위험과 매우 유사하다는 것은 놀라운 일이 아닙니다. 릴레이를 사용하지 않는 체인은 각 브라우징 세션 전에 새로운 VPN 사업자가 무작위로 선택되고 VPN 사업자가 개인 정보를 저장하지 않는 VPN 과 같습니다.

1 개의 릴레이, 1 개의 프록시

ISP	릴레이 $_1$	프록시	웹 사이트	P(관계)	P(타이밍)	P(언번)
	x	x		$\left(\frac{rx}{n^2}\right)$		
	x		x	$w\left(\frac{r}{n}\right)$		
x		x		$s\left(\frac{x}{n}\right)$		
x			x		sw	

이 구성에서 대역폭 굽기가 사용되면 모든 타이밍 공격이 완화됩니다. 타이밍 사례에 릴레이 1 또는 프록시를 추가하면 관계가 허용됩니다.

2 개의 릴레이, 1 개의 프록시

ISP	릴레이 1	릴레이 2	프록시	웹 사이트	P(관계)	P(타이밍)	P(언번)
	x		x		$\left(\frac{rx}{n^2}\right)$		
x		x	x		$s\left(\frac{rx}{n^2}\right)$		
	x	x		x	$s\left(\frac{r}{n}\right)^2$		
x		x		x	$sw\left(\frac{r}{n}\right)$		
	x			x		$s\left(\frac{r}{n}\right)$	
x				x		sw	

이 구성에서 대역폭 굽기가 사용되면 모든 타이밍 공격이 완화됩니다 릴레이 1 과 웹 사이트에 의한 타이밍 공격의 경우 릴레이 2 또는 프록시를 공모에 추가하면 관계가 형성됩니다. 고객의 ISP 가 웹 사이트와 공모하는 경우 릴레이 2 를 추가하면 관계가 형성됩니다.

3 개의 릴레이, 1 개의 프록시

ISP	릴레이 1	릴레이 2	릴레이 3	프록시	웹 사이트	P(관계)	P(타이밍)	P(언번)
	x	x		x		$\left(\frac{r^2x}{n^3}\right)$		
	x		x	x		$\left(\frac{r^2x}{n^3}\right)$		
x		x		x		$s\left(\frac{rx}{n^2}\right)$		
	x		x		x	$w\left(\frac{r}{n}\right)^2$		
x		x	x		x	$sw\left(\frac{r}{n}\right)^2$		
	x				x		$s\left(\frac{r}{n}\right)$	
x					x		sw	
	x	x			x			$s\left(\frac{r}{n}\right)^2$
x		x			x			$sw\left(\frac{r}{n}\right)$

B.2. SL 및 TLS 취약성

SSL 및 TLS 는 복잡한 프로토콜로서, 구현 결함이 발견될 때 일정한 보안 업데이트 스트림을 받습니다. 불행하게도 사용자는 때때로 소프트웨어 업그레이드를 지연시키고 신뢰할 수 없거나 잘못 작성된 소프트웨어를 사용하며 소프트웨어를 잘못 구성합니다. Orchid Protocol 은 사용자를 최대한 보호하기 위해 위생 검사 기능을 제공합니다.

SSL 다운그레이드 공격

소위 *SSL 다운그레이드 공격* 시, 공격자는 보안 연결이 낮은 품질의 암호화를 사용하도록 조종합니다([21]). 이러한 공격을 수행하기 위해 공격자는 클라이언트가 지원하는 보다 안전한 암호화 방법에 대한 언급을 초기 키 협상 패킷에서 간단히 제거합니다. 이러한 공격을 방지하기 위해 Orchid Client 는 가능한 한 자동으로 역작업을 수행합니다. 키 협상 패킷(SSL 업그레이드 공격)에서 안전하지 않은 옵션에 대한 언급을 제거합니다.

이전 브라우저 및 전화 앱

SSL 및 TLS 보안 취약성은 웹 브라우저에서 주기적으로 발견되고 패치됩니다. 그러나 모든 사용자가 최신 브라우저를 사용한다고 가정할 수는 없습니다. 개발자가 때로는 SSL 인증서 유효성 검사와 같은 것을 생략하는 휴대 전화 앱에서도 비슷한 상황이 발생합니다.

이러한 문제를 해결하기 위해 Orchid Client 는 구글 크롬에서 사용되는 오픈 소스 SSL 라이브러리인 Boring SSL 의 최신 사본을 사용하여 인증서 체인을 자동으로 확인합니다.

B.3. 방화벽 우회 기능

이미 인터넷에 대한 무료 및 공개 액세스 권한을 보유한 사용자만 사용할 수 있다면 위의 시스템은 거의 쓸모가 없습니다. 이 섹션에서는 공격자가 제공하는 인터넷 액세스 권한을 보유하는 사용자의 액세스를 용이하게 하는 기능에 대해 설명합니다.

공격자가 모든 인터넷 액세스를 완전히 차단하려는 경우 이 영역을 방어할 수 없습니다. 따라서 이 섹션의 모든 방어 분석에서는 공격자가 담요 차단(blanket blocking) 비용을 부담하고 있다고 가정하며, 비용이 상당히 소요되는 공격은 회피할 것으로 기대하여 그러한 비용을 최대화하는 방안을 찾습니다.

부트스트래핑

방화벽 제공업체가 Orchid Network 에 대해 시도할 것으로 예상되는 첫 번째 공격 중 하나는 초보 페들러 목록을 작성하고 그에 대한 모든 액세스를 차단하는 것입니다. 고객이 초보 페들러에 액세스 할 수 없으면 네트워크를 사용할 수 없기 때문입니다. 설상가상으로 유능한 공격자는 고객이 사용할 수 있는 IP 주소 목록이 있다고 가정해야 합니다.

우리는 초기에 이 문제를 해결하기 위해 사용자가 작업 증명 대신 새로운 릴레이 IP 주소를 알아낼 수 있는 서비스를 제공할 것입니다. 부트스트래핑 자체를 차단하기 위해 웹, 이메일 및 인기 있는 인스턴트 메시징 플랫폼을 통해 이 부스트래핑 서비스에 액세스할 수 있도록 할 것입니다. 사용자는 클라이언트의 옵션 화면에서 문제를 가장 편리한 통신 메커니즘으로 복사/붙여넣기한 다음 회신을 클라이언트에 다시 복사/붙여넣기하게 됩니다.

DPI, 추론 및 능동적 프로빙

보다 정교한 방화벽은 심층 패킷 검사(DPI: 헤더가 아닌 패킷 내용 분석), 타이밍 추론(패킷 크기, 수량 및 타이밍에 대한 총계 통계 측정 사용) 및 능동적 프로빙(제공되는 서비스를 식별하기 위해 연결 중인 사용자 또는 서버와 연결을 시도)과 같은 방법들을 활용합니다.

심층 패킷 검사 또는 능동적 프로빙을 사용하더라도 중요한 정보를 제공할 것으로 예상되지는 않습니다. WebRTC 를 사용함으로써 모든 통신은 암호화되며, 활성 WebRTC 오퍼가 발행되지 않는다면 열린 포트는 없습니다. 이는 그 외 모든 WebRTC 사용의 동작과 일치하므로 이러한 동작으로는 Orchid 사용자들을 명확히 구별할 수 없습니다.

암호화된 스트림을 통한 웹 요청의 타이밍 및 크기가 다른 종류의 WebRTC 트래픽처럼 보일 가능성은 거의 없기 때문에 타이밍 추론은 잠재적으로 Orchid 사용자를 감지하는 효과적인 방법인 셈이 됩니다([62]). 이 문제를 해결하기 위해 추론 공격이 발생할 만한 상황에서 Orchid 네트워크에 액세스하는 사용자들은 대역폭 굽기 를 사용하는 것이 좋습니다(섹션 7).

공개: 이더리움 트래픽

현재 클라이언트는 지불 상태를 추적하기 위해 이더리움 클라이언트를 활용하며 이더리움에는 강화되지 않은 자체 네트워킹 서명이 있기 때문에 이 이더리움 트래픽과 관련된 탐지는 약점일 가능성이 높습니다. 방화벽 운영자는 단순히 이더리움을 실행하는 컴퓨터입니까? *그리고* 이 컴퓨터는 많은 양의 WebRTC 트래픽을 소모합니까? 라고 질문할 수 있습니다.

프로젝트의 초점을 유지할 목적으로 Orchid 네트워크를 통한 이더리움 강화 및/또는 이더리움 트래픽 제공은 초기 릴리스의 기능에 속하지 않습니다. 향후 버전에서는 이 문제를 해결할 계획입니다(섹션 10 참조).

B.4. 공격 분석 및 공격자의 사용자 사례

억압적인 웹 애플리케이션

공격자 목표: 모든 Orchid 릴레이 및 프록시 IP 주소를 식별합니다.

Orchid Market 에는 모든 릴레이와 프록시가 포함되어 있기 때문에 이것은 섹션 B.3 에서 설명한 것과 반대되는 역공인 셈이 됩니다.

Orchid Market 에서 강제 연결의 수는 $O(log(n))$으로 증가합니다(여기서 n 은 네트워크 크기를 나타냄). 어떤 공격자가 전체 계산의 m%를 보유하면 작업 증명 계산을 완료할 때마다 $log(n)$ IP 주소들을 학습하게 됩니다. 따라서 c 에포크에서 릴레이 및 프록시 IP 주소의 $1 - (1 - \frac{\log(n)}{n})c$%를 학습하게 됩니다.

Tor 트래픽 차단이 어떻게 진행되는지를 잘 알고 있는 독자라면 상기의 설명이 매우 심각한 시스템상의 문제를 다루고 있다는 우려를 표시할 것 같습니다. 다행히도 실상은 그렇지 않습니다. Tor 에는 약 1,000 개의 출구 노드가 있어 쉽게 필터링할 수 있습니다. 우리의 경우에는 주로 화이트리스트에 대한 지원으로 인해 수십만 개의 출구 노드가 있을 것으로 예상됩니다. 이렇게 하면 상기의 방법을 사용하여 마스크를 해제하는 데 있어 훨씬 더 큰 계산 문제가 발생하게 되며, 이러한 IP 주소들을 차단하면 억압적인 웹 애플리케이션이 자체 사용자들을 차단하는 결과를 초래하게 됩니다.

기업 네트워크와 거대한 방화벽

공격자 목표: 인터넷 액세스를 제공하는 대상에 해당되는 사용자가 Orchid 네트워크를 사용하는 것을 방지합니다.

이와 관련된 제반 기능은 섹션 B.3 에서 자세히 설명하고 있습니다. 이 공격자에 대한 전망은 어둡습니다. Orchid 네트워크 사용량은 일정한 양의 데이터를 중계하는 WebRTC 연결로 나타납니다. 탐색할 수 있는 열린 포트가 없으며, 포트를 아웃(out) 하는 데 사용할 수 있는 IP 주소가 없습니다.

수동 모니터링 및 추론(Sybil Attack 을 이용할 경우)

공격자 목표: 고객 IP 식별 및 웹 사이트 식별을 학습합니다.

이 등급의 공격과 관련된 분석은 섹션 B.1 에서 자세히 논의하고 있습니다. The outlook for this attacker is bleak: 어떤 체인의 여러 위치에 자신을 배치하기가 어렵게 하려면 글로벌 컴퓨팅 성능의 상당 부분을 소유(하고 이러한 공격에 전념)해야 합니다.

3 류 트롤링 및 QoS 공격

공격자 목표: 공격자는 가능한 한 많은 네트워크에서 대혼란을 일으키려고 합니다.

보안을 염두에 두면서 좋은 때를 기다리고 있는 독자들이라면 이 영역에서 분석을 수행하는 것이 좋습니다. 여기서 해야 할 일은 한정된 예산(대략 미화 10,000 달러로 추정됨)을 감안해 네트워크를 최대한 많이 중단시키는 것입니다.

공격 체인 - 공격자는 여기서 다양한 공격을 시도할 수 있습니다. 임의로 패킷을 삭제하고 매우 느린 서비스만 제공하거나 간헐적으로 느린 서비스를 제공한다든지 아니면 간단히 연결을 끊기도 합니다. 어떤 경우든 간에 고객은 단순히 문제의 노드를 교체하기 때문에 결과적으로 모든 고객들은 약간의 불편함을 감수하는 셈이 됩니다. A 가 패킷을 전달하지 않았는지 아니면 B 가 패킷을 수신하지 못했다고 거짓말을 하는지 여부를 판단할 만한 방법이 없을 수도 있으므로 패킷을 삭제하는 경우, 다른 릴레이에 추가적인 불편이 발생할 수 있습니다. 이 경우, 고객은 두 릴레이를 모두 교체합니다.

Orchid Market 을 공격 - 마찬가지로 공격자들은 이러한 상황에 대응할 다양한 방안을 선택할 수 있습니다.

- 가입 프로토콜을 잘못 구현합니다. 우리는 이것을 공격으로 간주하지 않습니다. 이 경우, 우리의 공격자 는 단순히 다른 페들러들에게 패킷 전달 비용을 지불하고 있기 때문입니다.
- Orchid Market 에 가입하여 라우팅 테이블 정보를 사용자에게 제공하는 것을 거부하거나 패킷 전달을 거부합니다. 이렇게 하면 문제의 페들러가 네트워크에서 분리될 때까지 Orchid Market 쿼리의 $\frac{log(n)}{n}$에서 약간의 라우팅 부담이 추가로 발생하게 됩니다.
- 서비스를 제공하지 않고 Orchid Market 에 자리를 잡고 있습니다. 이 경우, 고객이 수행한 경매는 효율성이 떨어집니다(당시 한 참여자 $\frac{log(n)}{n}$의 손실로 발생).

C. 메달리온 엔지니어링 사양

이 추가 서버는 섹션 5.3 에서 높은 수준의 메달리온 사양을 기반으로 하여 메달리온 및 그 생성에 대한 보다 정확한 정의를 제시합니다. 이 부록은 [50]에 사용된 표기법을 기반으로 합니다. 아래의 목록은 유형별로 정렬한 것입니다.

t **(uint)** – 정밀도가 p 인 유닉스 시간을 가리키며, 정밀도는 적어도 약 100ms 이상이어야 함

skm **(uint)** – 임의로 선택한 비밀 키 x

e_t **(uint)** – 시간 t 의 이더리움 블록 다이제스트(이른바 '블록 해시'라고 함)

$h(y)$ **(uint)** – 입력 y 가 포함된 Keccak 의 다이제스트

$seed$ **(uint)** – $h(e_t \,/\, sig(sk,\, e_t))$

pk_m **(tuple)** – 타원 곡선 C 에 있는 공개 키 $x * G$(기준점 G 및 주문 N)

$sig(sk,\, r)$ **(tuple)** – 비밀 키 sk 를 사용하는 r 의 ECDSA 서명

$F_{n,k}(x)$ **(struct)** – $\{n,\, k,\, x,\, i_0,\, \ldots,\, i_2{}^k\}$: $n,\, k,\, x,\, i_j \in \mathrm{Z}\,/h/$

M **(struct)** – $\{t,\, e_t,\, pk_m,\, sig(sk,\, e_t),\, F_{n,k}(seed)\}$

메달리온 알고리즘

메달리온을 생성하기 위해 제안된 방법으로는 2 가지가 있습니다. 첫 번째 방법은 대화형이 아니며, 메달리온 생성 시 현재 이더리움 블록 다이제스트와 공개 키 쌍이 있어야 합니다. 아래에 제시된 Equihash 작업 증명은 단순화되었습니다.

알고리즘 1: 비대화형 메달리온 생성

준비 단계:

$sk \leftarrow$ 임의$\in \mathrm{Z}_{/\mathrm{h}/}$
$pk \leftarrow sk * G$
$sig(sk,\, e_t) \leftarrow \mathrm{ECDSA}(sk,\, e_t)$
$seed \leftarrow h(e_t,\, sig(sk,\, e_t))$

Equihash 작업 증명:

set 난이도 $(n,\, k)$
set 카운터 $i_1 = seed$
2^k 항목의 *set* $\{i_j\}$
이때 $\{h(i1) \oplus h(i_2) \oplus \ldots \oplus h(i_2{}^k) \neq 0\}${
build $\{\mathrm{i}_j\}$의 더 큰 목록
find 충돌하는 $\{\mathrm{i}_j\}$의 하위 집합
sort $\{\mathrm{h}(\mathrm{i}_j)\}$
}

반환: $t, e_t, pk, sig(sk, e_t), \{i_j\}$

두 번째 방법은 첫 번째 방법을 기반으로 하며, Orchid Market 내 페들러들은 메달리온 구축에 참여해야 한다는 요건을 추가합니다. Orchid Market 에서 참여를 요구함으로써 도전-응답 유형 프로토콜이 생성되는데 이 프로토콜은 원시적인 시간 증명으로 비유할 수 있습니다.

이 계획에는 m 요소, 메달리온 생성기, *Alice* 그리고 Orchid Market 의 페들러 커뮤니티가 있는데, 이는 $pi \in \{Bob, Chris, Dana, ...\}$로 표기됩니다. *Alice* 는 $sig(skBob, e)$를 계산하고 서명을 *Alice* 에게 반환할 초보 페들러 Bob 과 상호 작용할 것입니다. Orchid Market 에서 추가적인 참여를 요구할 경우, *Bob* 은 *Bob* 을 통해 *Alice* 에게 $sig(sk_{pi}, e)$를 반환할 m 명의 임의 페들러들에게 연락할 것입니다. *Alice* 는 $\{sig(sk_{pi}, e)\}$ 를 추가 입력으로 사용해 *seed* 를 계산한 후 M을 *Bob* 에 반환합니다.

알고리즘 2: 응답 지향 메달리온 생성

준비 단계:

$sk \leftarrow$ 임의$\in Z_{|h|}$

$pk \leftarrow sk * G$

$sig(sk, e_t) \leftarrow \text{ECDSA}(sk, e_t)$

대화형 단계:

시간 증명을 수행

$seed \leftarrow h(et, \{sig(sk_{pi}, e_t))\}$

Equihash 작업 증명:

set 난이도 (n, k)

set 카운터 $i1 = seed$

2_k 항목의 *set* $\{i_j\}$

이때 $\{h(i_1) \oplus h(i_2) \oplus ... \oplus h(i_2k) \neq 0\}${

build $\{i_j\}$의 더 큰 목록

find 충돌하는 $\{i_j\}$의 하위 집합

sort $\{h(i_j)\}$

}

반환: $t, e_t, pk, sig(sk, e_t), \{i_j\}$

D. 결제 프로토콜 및 정의

D.1. 결제 티켓 암호화 선택

Orchid 소액 결제의 비용을 줄이기 위해 우리는 다른 임의의 함수에 비해 이더리움 가스 비용이 감소한 이유로 특정 암호화 함수들을 다른 함수들보다 우선적으로 선택했습니다.

h **(Keccak-256)** – 모든 보안 암호화 해시 함수를 Orchid 프로토콜에서 사용할 수 있습니다. 그러나 Keccack 은 EVM 에서 사용할 수 있는 모든 해시 함수 중 가스 비용이 가장 낮기 때문에 시스템의 다른 해시보다 우선하여 특별히 선택되었습니다[4]. 이것은 Orchid 거래 비용을 추가로 최소화하기 위해 선택되었습니다.

sig(*sk, r*) **(ECDSA)** – Kecak-256 이 포함된 secp256k1 곡선을 내부 암호화 해시 함수로 사용합니다. 여기서도 이러한 선택을 한 이유는 EVM 이 기본적으로 ECDSA 를 지원하여 Orchid 가스 비용을 절감했기 때문입니다. 또한 곡선 선택은 기존 블록체인 소프트웨어 라이브러리 및 도구들과도 호환됩니다.

D.2. 결제 티켓 정의

Orchid 결제 티켓의 필드는 다음과 같습니다.

h **(함수)** – 암호화 해시 함수

timestamp **(uint32)** – 가치가 감소하기 시작하는 시점을 나타내는 시간

recipient **(uint160)** – 티켓 수신자의 160 비트 이더리움 계정 주소

rand **(uint256)** – 수신자가 선택한 임의의 정수

nonce **(uint256)** – 티켓 발신자가 선택한 임의의 정수

faceValue **(uint256)** – 당첨 티켓의 가치

marketValue **(uint256)** – 대역폭 시장을 기반으로 한 티켓의 예상 가치

acceptedValue **(uint256)** – 수신자가 수락한 것을 근거로 할 때 예상되는 티켓 가치

winProb **(uint256)** – 특정 티켓이 발신자로부터 *faceValue* 를 획득할 확률

randHash **(uint256)** – h(*rand*)의 다이제스트

ticketHash **(uint256)** – h(*randHash, recipient, faceValue, winProb, nonce*)의 다이제스트

(*v*1, *r*1, *s*1) **(tuple)** – 티켓 발신자의 ECDSA 서명 요소

(*v*2, *r*2, *s*2) **(tuple)** – 수신자의 ECDSA 서명 요소

[4] 32 바이트 해싱을 위한 이더리움 비용은 36 가스[89]임

D.3. 결제 티켓 생성

*Alice*가 수신자에 해당되며 *Bob*이 발신자에 해당된다고 할 때,

1. *Alice*는 임의의 256 비트 숫자, *rand*를 선택하고, randHash를 계산한 다음, 다이제스트를 Bob에게 전달합니다.
2. *Bob*은 (nonce, faceValue, winProb, recipient)에 대한 값을 선택합니다.[5]
3. *Bob*은 ticketHash를 계산합니다.
4. *Bob*은 Sig(PrivKey, ticketHash)를 계산합니다.
5. 이로 인해 발생하는 티켓은 다음 항목들로 구성됩니다.

   (a) *randHash*
   (b) *recipient*
   (c) *faceValue*
   (d) *winProb*
   (e) *nonce*
   (f) *ticketHash*
   (g) *creator*(*ticketHash*를 서명한 발신자 키의 주소)
   (h) *creatorSig*(*ticketHash*에 대한 발신자의 서명)

이 티켓은 수신자가 모두 확인할 수 있다는 점에서 유효하지만 Orchid 결제 이더리움 스마트 계약에서 티켓을 청구하려면 수신자가 티켓에 서명해야 합니다(아래 참조).

D.4. 결제 티켓 확인

Alice(대역폭 판매자)는 다음과 같은 연산을 수행합니다.

검증:

(a) $randHash = H(rand)$
(b) $faceValue \geq marketValue$
(c) $winProb \geq acceptedValue$
(d) *recipient* = {수신자가 발행한 이더리움 계정 주소}
(e) *creator* = {발신자가 발행한 이더리움 계정 주소}

확인:

(a) 확인: *creatorSig*는 개인 키에 따른 서명에 속하며 공개 키는 생성자 주소에 해당됩니다.

검사:

(a) 확인: *creator*는 Orchid Payment 스마트 계약에 충분한 Orchid 토큰이 고정되어 있습니다.

주장:

(a) 티켓은 이제 유효한 것으로 증명되었으며 당첨 티켓일 수도 있습니다.

[5] 다음을 포함해 이러한 사양의 정보를 사용: 일반 대역폭 시장 데이터 및 *수신자*가 서명한 공개 기능

D.5. 티켓 청구 결제

수신자는 하나의 티켓이 유효한지 여부와 이것이 당첨 티켓인지 여부를 로컬에서 충분히 확인할 수 있는 반면, 당첨 티켓에서 토큰의 실제 지불은 Orchid Payment 스마트 계약에 따라 이루어집니다. Orchid 스마트 계약은 다음 항목들을 입력으로 처리하는 Solidity API 를 공개합니다.

1. *rand*
2. *nonce*
3. *faceValue*
4. *winProb*
5. *receipient*
6. *recipientSig* (ticketHash 에 대한 *recipient*)
7. *creatorSig*(*ticketHash* 에 대한 발신자의 서명)

D.5.1. 스마트 계약 실행

Alice 가 대역폭을 구입하려는 사용자라고 가정합니다. *Alice* 는 이더리움 계정 주소, *addressAlice* 및 Orchid 토큰을 갖고 있어야 합니다. 이 주소에는 *PubKeyAlice* 라는 관련 공개 키가 포함됩니다. 또한 Alice 는 이전 섹션에서 정의한 바와 같이 이더리움 스마트 계약에 *고정되어* 있으며 *PubKeyAlice* 와 함께 잠겨 있는 Orchid 토큰이 있어야 합니다. 이전 섹션에서 *Alice 의* 주소는 *creatorSig* 에서 *ticketHash* 를 통해 복구된 공개 키와 일치하는 이더리움 계정 주소입니다.

SLASH 를 FALSE 로 설정된 임시 부울 값이라 가정하고, *PubKey* 를 *recipientSig* 에서 *ticketHash* 를 통해 복구된 공개 키라 가정합니다.

계산:

(a) *ticketHash*

검증:

(a) *randHash*. 그렇지 않을 경우, 실행을 중단합니다[6].

(b) *PubKey* = *recipient* 주소. 그렇지 않을 경우, 실행을 중단합니다.

(c) *addressAlice* 는 페널티 에스크로 계정에 고정된 Orchid 토큰이 있습니다. 그렇지 않을 경우, 실행을 중단합니다.

(d) *addressAlice* 의 티켓 계정에는 티켓 비용을 결제할 수 있을 정도로 충분한 수량의 Orchid 토큰이 고정되어 있습니다. 그렇지 않으면 SLASH 를 TRUE 로 설정하고 실행을 계속하십시오.

(e) $H(ticketHash, rand)^{7} \leq winProb$. If not, abort execution.

결정:

[6] 거래가 중단되었으므로 이더리움 상태 전환이 발생하지 않으며 가스가 소비되지 않음

[7] uint256 으로 해석됨

(a) SLASH = FALSE 인 경우, 티켓 대금이 결제됩니다. *faceValue* 는 생성자의 티켓 펀드에서 *recipient(수신자)*로 전달됩니다.

(b) SLASH = TRUE 인 경우, 생성자는 삭제됩니다.

정산:

(a) 생성자의 티켓 펀드(있다면)를 *recipient(수신자)*에게 보냅니다(이는 *faceValue* 보다 작은 값으로 보증된 이전 유효성 검사 결과를 출처로 합니다).

(b) 생성자의 페널티 에스크로 계정을 0 으로 설정(그러한 토큰들을 굽기/삭제하기)합니다.

페널티 에스크로를 삭제하면 티켓 발신자가 잠재적 이중 지출에서 얻을 수 있는 이득보다 더 많은 손실을 보게 될 때 티켓 발신자의 지출 의욕을 억제함으로써 이중 지출을 방지하지만 티켓 발신자는 여전히 대규모로 초과 지출을 할 위험이 있습니다. 이 문제를 해결하려면 복권 당첨의 가치가 *타임 스탬프*에서 기하급수적으로 감소하기 시작함으로써 결과적으로 당첨자들이 즉시 현금으로 교환하도록 강력한 동기를 부여해야 합니다. 수신자는 이러한 즉각성을 이용하여 발신자가 가진 Orchid 토큰 잔액의 폐기율 을 계산할 수 있습니다.

E. 추가적인 결제 세부 정보

E.1. 패킷 비용은 얼마입니까?

논의를 위해 평균 패킷의 길이는 1KB 라고 가정합니다. 이제 패킷의 평균 총 비용에 대한 합리적인 상한 값을 계산해보겠습니다. 클라우드 서비스에서 상대적으로 더 비싼 대역폭 제공업체 중 하나인 Amazon Web Services 의 Singapore CloudFront 는 1×10^9 바이트당 0.14 달러의 비용을 청구하고 있습니다. 따라서 패킷당 비용은 1.4×10^{-5} 센트(0.00000014 달러)입니다.

대역폭은 대부분의 사용자들에겐 소모재(판매되지 않은 대역폭은 영구적으로 손실됨)이기 때문에 Orchid Market 의 실제 대역폭 가격은 이러한 상한 값보다 훨씬 낮을 것입니다.

E.2. 이더리움 거래 비용

이더리움 스마트 계약은 Turing 의 전체 실행 환경을 (경제적 한도 내에서) 제공하는 EVM(Ethereum Virtual Machine, 이더리움 가상 머신)[89]의 성능과 유연성을 활용하면서 정교한 결제 메커니즘을 만들 수 있습니다. 이더리움 스마트 계약에 의해 실행되는 각 명령은 원래 거래의 거래 수수료에 추가됩니다.

각 EVM 명령은 일정량의 가스를 소비하며, 이더리움 거래 수수료는 거래에 사용된 가스의 총량에 발신자가 설정한 가스 가격을 곱한 값으로 정의됩니다. 채굴자들은 그들이 채굴한 블록에 포함시킬 유효한 거래를 모두 선택하고 모든 가스 가격(0 을 포함)을 조건으로 한 거래를 포함시킬 수 있습니다. 가스 가격이 더 높은 거래를 선택하면 블록마다 포함시킬 수 있는 거래의 한도가 있기 때문에 더 많은 수익이 발생할 수 있습니다. 마찬가지로, 비교적 낮은 가스 가격을 수락하면 네트워크가 최대 용량으로 작동하지 않을 때 채굴자가 블록을 채울 수 있기 때문에 더 많은 수익을 창출할 수도 있습니다. 이 메커니즘은 변화무쌍하면서도 안정적인 게임 이론 평형을 형성하며, 이러한 평형은 Ethereum Gas Station[22]과 같은 사이트에 의해 추적됩니다.

2017 년 10 월 현재, 몇 블록 내에서 어떤 거래가 높은 확률로 포함되는 데 드는 비용은 0.026 달러입니다. 0.006 달러 정도면 15 분 이내에 충분히 확인할 수 있습니다. 이는 기본 거래 비용에 대한 추정치입니다. 즉, 스마트 컨트랙트 코드를 실행하지 않고 일반 이더 전송(plain ether transfer) 시 21,000 가스가 소요될 것으로 추정됩니다. 해당 거래에서 스마트 컨트랙트 코드가 실행되면 각 EVM 명령은 부가적인 가스 비용을 추가합니다. 예를 들어, 새로운 256 비트 값을 스마트 계약 스토리지에 영구적으로 저장하려면 20,000 가스가 필요하며 기존 값을 업데이트하려면 5,000 가스가 필요합니다.

이더리움 ERC20 원장은 단순히 계정 주소를 잔액에 매핑하는 것이므로 ERC20 토큰 전송을 실행하려면 새 계정에서 약 21,000 + 20,000 가스의 비용이 소요되며, 이후의 전송에서는 21,000 + 5,000 가스가 필요합니다(수신자 계정이 이미 토큰 원장에 입력되어 있기 때문). 실시간[23] ERC20 거래를 살펴보면, 신규 및 기존 계정으로 전송 시 가스 비용이 각각 약 52,000 가스 및 37,000 가스로 약간 더 높은 것을 알 수 있습니다. 이러한 차이는 발신자가 충분한 잔액과 그 외 세부적인 구현 정보(예: 결제 영수증 로깅)가 있는 경우와 같이 불변량의 유효성 검사를 실행하는 스마트 컨트랙트 코드를 설명합니다. 50,000 가스의 경우, 원하는 거래 확인 속도에 따라 0.014 달러~0.062 달러의 거래 수수료가 소요될 것으로 추정됩니다.

E.3. 성능

Orchid Payments 스마트 계약은 변경할 수 없으며 다만 신규 계약을 배포하고 Orchid 클라이언트 소프트웨어가 신규 계약을 가리키도록 이 소프트웨어를 업그레이드(필요하다면 이전 계약과 역호환성을 유지)하면 Orchid Payments 스마트 계약을 효과적으로 업그레이드할 수 있습니다. 따라서 이더리움 스마트 계약은 가스 비용을 줄이기 위해 다양한 최적화를 지원하며, Orchid Payments 스마트 계약의 향후 버전들은 일반 소프트웨어 시스템이 고가의 서브루틴을 종종 인라인 어셈블리로 대체하는 방법과 유사한 방법으로 가스 비용을 최적화하기 위해 이를테면 인라인 솔리드 어셈블리[24]를 사용할 것으로 예상됩니다.

그러나 Orchid 결제 티켓의 확인 과정에서 ECDSA 복구와 같은 암호화 작업과 Orchid 토큰 원장 내 발신자 및 수신자 항목의 상태 업데이트 시 병목 현상이 발생합니다. 여기서 한 가지 개선된 점은 티켓 데이터 구조를 포함하는 서명으로서 스마트 계약 API 호출 페이로드를 전달하는 이더리움 트랜잭션의 수신자 서명을 이중으로 사용하는 것일 수도 있습니다. 현재 Orchid 방식은 이더리움 세부 사항에 의존하지 않고도 결제 방식을 보다 쉽게 지정하고 추론하기 위해 Orchid 클라이언트의 유연성을 이유로 2 가지 서명을 정의하고 있습니다. 보다 단순한 최적화를 위해 티켓 필드를 빈틈없이 패킹하는 한편, EVM 스택 워드와 영구 계약 스토리지 슬롯(둘 다 크기가 256 비트)에 맞게 다수의 내부 변수들을 단일 256 비트의 단어로 인코딩하고 있습니다.

한편, 익명성을 더욱 강화하기 위해 혼합 기법들을 선택적으로 또는 의무적으로 사용하면 Orchid Payments 의 가스 비용이 상당히 증가할 수 있습니다. 연결 가능한 링 서명에 기반한 혼합 서비스를 사용하면 훨씬 더 많은 거래 수수료가 쉽게 발생할 수 있습니다[20]. 그러나 익명성이 강력하게 보장된다면 사용자들의 입장에서 이러한 서비스 이용은 그만한 가치가 있다고 볼 수 있습니다. Orchid 결제의 확률 변수(매표 빈도, 당첨 확률 및 당첨 금액)는 쉽게 조정할 수 있기 때문에 우리는 거래 수수료를 줄이기 위해 티켓 청구 간 평균 시간을 조정할 수 있습니다(특히, 평균적으로 며칠에 한 번만 결제해도 되는 장기간 실행 중인 노드의 경우).

마지막으로 zk-SNARK 와 같은 제로 지식(zero-knowledge) 기술의 매우 흥미로운 특성은 이더리움 스마트 계약 실행과 같은 임의 계산의 연산 오버헤드를 크게 줄이는 데 있습니다[25]. zk-SNARK 증명을 생성하려면 많은 비용이 들지만 검증에 소요되는 비용은 원래 코드와 비교해볼 때 오히려 더 저렴합니다. 검증만 온체인으로 실행하면 되기 때문에 Orchid 티켓 청구의 제로 지식 증명은 원래의 검증 코드보다 검증 비용이 적게 들 수 있습니다.

더 나아가 재귀적인 SNARK[51]는 일련의 SNARK 증명을 단일한 증명으로 집계할 가능성이 있습니다. 이러한 SNARK 는 블록체인 합의 프로토콜에 오히려 더 적합할 수 있겠지만[26] Orchid 가 선형적인 가스 비용 복잡성을 피하면서 다수의 티켓 청구를 단일한 스마트 계약 거래로 일괄 처리하는 데 유용할 수도 있습니다.

E.4. 거액 결제에서 소액 결제를 구성

지금까지 논의한 거래 비용 및 결제 토큰의 선택에 이어, 실행 가능한 결제 방법을 살펴보기로 하겠습니다. 블록체인 기반 소액 결제의 기본적인 과제 중 하나는 거래 수수료를 방지할 방법입니다. 1 센트를 수많은 횟수에 걸쳐 송금하는 상황을 가정합니다. 만약 1 센트를 일반 이더리움 ERC20 거래로 송금한다면 1.4 센트를 부담하는 셈이 됩니다(즉, 결제 건마다 140%의 거래 수수료가 발생)! 효과적인 소액 결제를 위해서는 거래 수수료를 몇 분의 1 수준으로 낮춰야 합니다.

MojoNation[27]에서 채택한 잠재적으로 흥미로운 한 가지 접근법은 각 노드 쌍 간에 거래 수지(balance of trade) 를 유지하는 것입니다. 대역폭은 한 쌍의 노드 간에 흐르기 때문에 거래 수지가 0 에서 너무 멀어지면 결제는 주기적으로 정산됩니다. 그러나 앞서 살펴본 바와 같이, 일반 이더리움 거래를 이용하여 결제를 정산하면 최소한 0.014 달러의 거래 수수료가 발생하는 셈이 됩니다. 이 가격은 앞서 설명한 상한 값을 기준으로 할 때 약 140MB 의 대역폭과 같다고 볼 수 있습니다. 이러한 접근 방식의 두 번째 문제는 조정 임계값에 근접하는 피어(peer)들이 그러한 사실을 알게 되어 수수료를 부담하는 대신, 연결을 끊고 새 ID 를 생성하고 싶어한다는 점입니다.

E.5. 결제 채널

비트코인 네트워크에서 맨 처음 볼 수 있는 블록체인 애플리케이션의 범용 기술은 결제 채널입니다[28]. Satoshi Nakamoto[66]가 부분적으로 설명하고 이후에 Hearn & Spilman[29]이 정의 및 구현한 결제 채널은 그 후 Poon & Dryja[30]가 비트코인 번개 네트워크를 대상으로 하여 연구한 바 있습니다. 결제 채널을 사용하면 발신자와 수신자가 서로 임의의 거래 금액을 송금하고 2 건의 거래에 대해서만 거래 수수료(하나는 결제 채널 설정을 위한 수수료이며 다른 하나는 결제 채널 종료를 위한 수수료임)를 부담할 수 있습니다. 이러한 절차는 수신자에게만 발송이 가능하거나 또는 발신자에게 다시 보낼 수 있는 일정량의 토큰을 잠그는 거래 건을 발신자로 하여금 게시하도록 허용함으로써 처리됩니다. 일반적으로 토큰은 이후 어떤 시점 T에서 발신자에게만 다시 보낼 수 있습니다. 한편, 토큰은 수신자에게 (단계적으로 또는 전부) 전송할 수 있습니다. 발신자는 수신자를 대상으로 더욱 더 많은 양의 토큰을 사용하는 거래에 계속 서명하며, 거래 내역을 블록체인에는 게시하지 않고 수신자에게 직접 전달합니다. 수신자는 집계된 송금액을 청구하기 위해 T 시점까지 마지막으로 수신한 거래 내역을 언제든지 게시할 수 있습니다.

결제 채널은 발신자가 지속적인 결제의 암호화된 증거를 수신자에게 효율적으로 제시할 수 있는 방법을 제공합니다. 중간 결제는 거래 수수료가 발생하지 않기 때문에 임의로 소액 결제를 할 수 있으며 임의의 빈도로 송금할 수도 있습니다. 실제로 병목 현상은 거래 내역을 확인하는 데 따른 컴퓨팅 오버헤드와 거래 내역 전송 시 발생하는 대역폭 요구로 나타나게 됩니다.

결제 채널에서는 임의의 중간 결제 금액에 대해 항상 복잡한 거래 수수료 내역을 효과적으로 제공하며 다만 사용 사례에 따라서는 그다지 효율적이지는 않습니다. 특히 상호작용 과정에서 수많은 발신자와 수신자들이 자주 변경되는 시스템의 경우, 새로운 결제 채널을 지속적으로 생성하는 데 따른 비용이 너무 많이 들 수 있습니다. 마찬가지로, 단일 HTTP 요청 또는 10 초 분량의 비디오 스트리밍 등 매우 작거나 단기적인 서비스가 제공될 경우, 필요한 온체인 거래를 진행할 때 너무 많은 거래 수수료 부담이 발생할 수 있습니다.

E.6. 확률적 결제

블록체인에서 거래 수수료를 부담하는 조건으로 결제 정산이 진행된다는 가정에 대해 이의를 제기할 수 없다면 이론상의 최소 비용은 단일 거래 비용에 해당되는 데, 그 이유는 블록체인은 상태 전환을 실행하기 위해 적어도 한 건 이상의 거래가 필요하기 때문입니다. 따라서 일정 금액의 (소액) 결제를 정산하려면 적어도 한 건 이상의 거래가 필요합니다.

결제 채널에서 요구하는 설정 거래를 없애고 결제가 진행 중임을 수신자에게 계속 증명할 수 있다면 어떨까요?

다행히 블록체인 업계에서는 이와 비슷한 문제가 해결되었는데 바로 채굴 풀 공유[31]가 그것입니다. 비트코인과 같은 네트워크에서 작업 증명 난이도가 높아짐에 따라 채굴자들은 한 명의 채굴자가 블록 솔루션을 찾는 데 수년의 시간이 걸릴 수 있는 높은 분산을 피하기 위해 컴퓨팅 능력을 끌어 모으기 시작했습니다. 채굴 풀은 해시 파워에 비례한 보상을 제공하며, 개인 채굴자들은 동일한 기반 블록 해시를 위한 솔루션을 비교적 낮은 난이도로 계속 전달함으로써 그들이 보유한 해시 파워를 증명합니다[32]. 이 기법을 사용하면 풀 구성원이 실제 작업 증명 대상을 충족하는 솔루션을 찾는지 여부에 관계없이 채굴 풀에서 각 풀 구성원의 해싱 파워를 암호화된 방식으로 확인할 수 있습니다.

이와 동일한 접근 방식을 결제 채널에 적용하면 실제 결제가 진행되는지 여부에 관계없이 *평균적으로* 결제가 진행 중임을 발신자가 수신자에게 지속적으로 증명하는 확률적 결제 방식을 구성할 수 있습니다. 이러한 방식을 활용하면 거래를 설정할 필요가 없는 확률적 소액 결제를 생성할 수 있으며, 수신자는 현금 교환 을 할 때만 거래 수수료를 부담하면 됩니다.

이더리움 스마트 계약을 사용하여 이러한 확률적 소액 결제를 구성하는 방법을 살펴보기 전에 한 걸음 물러서서 확률적 결제에 대한 최초의 개념이 블록체인 기술보다 앞서 등장하고 있으며 1996 년 David Wheeler[88]에 의해 처음으로 발표되었다는 사실에 주목해 볼 필요가 있습니다. Wheeler 는 확률적 결제의 핵심 개념을 설명하고 있으며, 한편으로는 발신자나 수신자(Wheeler 의 논문에서는 '구매자'와 '판매자'라는 용어로 인용됨)가 당첨 확률과 당첨 금액이 얼마인지를 서로에게 계속 증명하면서도 확률적 사건의 결과를 조작할 수 없는 방식으로 난수 약정을 사용한 전자 프로토콜에 그러한 핵심 개념을 적용하는 방법에 대해서도 설명하고 있습니다.

몇몇 연구 논문들은 Wheeler 의 이러한 개념에 대한 후속 논의를 진행했으며 1997 년 Ronald Rivest[82]는 전자 소액 결제에서 확률적 결제를 적용하는 방법을 설명하는 1 건의 논문을 발표하기도 했습니다. 2015 년 Pass & Shelat[81]는 비트코인과 같은 분산형 통화에 확률적 소액 결제를 적용하는 방법을 설명했으며, 이전의 결제 방식들은 모두 신뢰할 수 있는 제 3 자에 의존했다는 점에 주목했습니다. 이듬해인 2016 년 Chiesa, Green, Liu, Miao, Miers 및 Mishra[58]는 이 연구를 제로 지식 증명 없이 진행할 수 있도록 확장했으며, 암호 화폐 프로토콜에 적용할 수 있는 분산형 및 익명의 소액 결제 방식을 제시했습니다.

최근 이더리움 기반 시스템에서 결제 채널에 대한 관심과 이러한 채널의 유행을 감안한다면 확률적 결제를 결제 채널의 관점에서 보는 것이 유용할 수 있습니다. 첫 번째 설정 거래를 생략하는 대신, 정확한 금액의 송금을 보증할 수 있는 능력은 잃게 되며 확률적 보증만 얻게 됩니다. 다만 확률, 당첨 금액 및 결제 빈도를 조정하면 확률적 소액 결제를 세분화할 수 있으며 이를 통해 여러 가지 등급의 블록 체인 기반 애플리케이션에 대한 결제 채널들을 큰 결점 없이 대체할 수 있음을 보여드리겠습니다.

기본적으로 초기 설정 거래는 제거할 수 있으므로 동일한 발신자 계정에서 임의의 소규모 서비스 세션에 대한 비용을 임의의 수신자들에게 지불하면서도 각 수신자에게 해당 금액을 지불할 수 있는 정확한 확률을 계속 입증할 수 있게 됩니다. 서비스 제공업체(Orchid Network 의 릴레이 노드 또는 프록시 노드)가 충분한 양의 서비스를 제공한다고 가정할 때, 확률적 지불의 분산은 빠른 시간 내에 균일하게 됩니다.

E.7. 추가적인 Orchid 토큰 세부 사항

인센티브 제공

인센티브 제공은 사람들에게 네트워크의 부분적 소유권을 부여함으로써 새로운 프로토콜과 네트워크를 부트스트랩하는 하나의 방법에 속합니다[33]. Orchid 와 같은 새로운 분산형 네트워크는 '닭이 먼저냐 달걀이 먼저냐(chicken and egg)'의 문제에 부딪히고 있습니다. 프록시 및 릴레이 노드가 많을수록 네트워크는 사용자에게 더 많은 유틸리티를 제공합니다. 또한 사용자가 많을수록 프록시 노드 또는 릴레이 노드를 실행하는 것이 더욱 중요하게 됩니다. 새로운 네트워크 토큰을 배포하면 모든 잠재적 사용자가 네트워크를 조기에 사용하려는 동기가 생기기 때문에 네트워크 효과를 앞당겨 실현할 수 있습니다.

분리

그 밖의 분산형 시스템들을 기반으로 구축된 분산형 시스템에서 새로운 토큰은 새 시스템의 시장 가치를 기본 시스템에서 분리합니다. 예를 들면 2017 년 10 월 현재, 이더(Ether)의 시가 총액은 약 3 백억 달러에 달하며 전 세계 일일 거래량은 약 5 억 달러에 이르고 있습니다[34]. 이더의 가격은 암호 화폐에 대한 전반적인 추측, 이더리움 채굴자들의 해시 파워, 이더리움을 기반으로 한 수백 건의 프로젝트의 성패 등 다양한 요인의 영향을 받습니다. 물론 단일 프로젝트의 성패는 이더 가격에 큰 영향을 미치지 않을 수도 있겠지만 문제의 프로젝트와 관련된 특정 토큰에 지대한 영향을 미치게 됩니다. 새로운 토큰을 사용하여 시장 가치를 분리하면 문제의 프로젝트 및 시스템의 규모와 상태를 보다 잘 나타내는 지표를 얻을 수 있으며, 해당 시스템의 미래에 대한 예측 시장이 효과적으로 창출됩니다.

유동적 시장

시스템별 토큰을 거래하는 유동적 시장에서 시스템에 크게 의존하는 사용자들은 짧은 포지션을 취함으로써 시스템의 잠재적 장애에 대비할 수 있습니다. 만약 이것이 너무 무리한 것처럼 보인다면 금융 파생상품이 애초에 기업들로 하여금 미래에 닥칠 불운한 사건에 미리 대비할 수 있도록 하는 데 그 의의를 두고 있었다는 점에 주목해야 합니다. 0x[35] 및 etherdelta[36]와 같은 분산형 거래소와 Augur[37] 및 Gnosis[38]와 같은 예측 시장이 등장함에 따라 그리 멀지 않은 미래에 이더리움을 기반으로 한 토큰 및 시스템의 파생상품들이 출시될 것으로 전망됩니다. 사실 그러한 파생상품은 전통적인 금융 파생상품보다 훨씬 더 효과적일 수 있는데[39], 그 이유는 신뢰할 수 있는 당사자가 없으며 무허가 방식으로 거래될 뿐만 아니라 심지어 익명으로 거래될 수도 있기 때문입니다.

새 토큰

또한 새 토큰을 사용하면 이해 관계자들을 위한 특정 인센티브를 보다 쉽게 설계할 수 있습니다. 토큰은 새로운 시스템에서 그 가치를 배타적으로 파생하기 때문에 시스템의 성공을 위해 노력하는 모든 이들에게 강력한 인센티브로 작용합니다. 이더리움 스마트 계약은 토큰 소지자가 오로지 지정된 일정에 따라 자신의 토큰에 액세스할 수 있도록 토큰의 자동 잠금 기능을 구현할 수 있습니다. 이러한 계약에서는 시간이 지남에 따라 인센티브가 조정되며 토큰 소지자들은 특정 팀이나 관련 회사와 같은 사회적 구조보다는 *시스템*의 장기적인 성공에 역점을 두게 됩니다. 만약 Orchid 네트워크가 단순히 이더(Ether)를 사용했고 이해 관계자들이 이더의 동결을 수용했다면 그들은 이더리움을 사용하는 특정 시스템보다는 오히려 이더리움의 전반적인 성공을 위해 노력하려는 동기(인센티브)가 실제로 더 컸을 것입니다. 그러한 결과는 Orchid 네트워크 및 프로젝트에 대한 최적의 인센티브 조정이 아니라고 주장할 수 있습니다.

E.8. 검증 가능한 임의 함수

이전 섹션에 설명된 결제 티켓은 수신자의 난수 약정을 검증 가능한 임의 함수(VRF)로 대체하여 대화성이 떨어질 수 있습니다. 1999 년 Micali, Rabin 및 Vadhan[84]에 의해 처음으로 발표된 VRF 에 대한 IETF 초안은 최근 Goldberg 와 Papadopoulos[40]가 제안한 바 있습니다. 이 초안은 2 가지 VRF 구성을 지정하는데, 하나는 RSA 를 사용하는 구성이며 다른 하나는 타원 곡선(EC-VRF)을 사용하는 구성입니다.

VRF 를 사용하면 Orchid Payment 티켓의 발신자는 수신자의 티켓당(또는 당첨 티켓이 나올 때까지 티켓당) 약정 없이도 티켓을 만들 수 있습니다. 그 대신, 발신자는 수신자의 공개 키만 알면 됩니다. 발신자는 앞서 설명한 티켓 체계에서 난수 해시를 이러한 공개 키로 대체합니다. 티켓에 이미 존재하는 자금을 수령하기 위한 수신자 공개 키를 사용하면 효율적이겠지만 키 분리의 암호화 원칙을 준수하려면 두 번째 키가 필요할 수 있습니다.

그러나 Orchid Payments 스마트 계약에서 EC-VRF 를 검증하려면 타원 곡선 작업의 명시적인 EVM 가속이 필요한데, 그 이유는 이러한 작업을 직접 견고하게 구현하거나 EVM 어셈블리에 구현하면 가스 비용 면에서 엄청나게 많은 비용이 들기 때문입니다.

다행히도 이더리움 비잔티움(Ethereum Byzantium)[41] 릴리스에서 이더리움 네트워크는 타원 곡선 스칼라 덧셈 및 곱셈[42]은 물론, alt bn128 곡선[49]에 대한 페어링 점검[43]을 추가했습니다. EC-VRF 구성은 모든 타원 곡선에 대해 정의되며 IETF 초안은 특별히 EC-VRF-P256-SHA256 을 EC-VRF 암호 스위트로써 정의합니다(여기서 P256 은 NIST-P256 곡선에 해당됨[53]). 그러나 충분한 보안 수준을 계속 실현하면서 alt bn128 곡선을 대신 사용할 수 없는 이유는 없는 것 같습니다. 또한 SHA256 은 Keccak-256 으로 대체할 수 있습니다. 이렇게 하면 이더리움 스마트 계약에서 VRF 검증을 할 수 있기 때문에 Orchid Payments 스마트 계약과 통합이 가능합니다.

alt bn128 곡선은 zcash 에서 사용되지만 P256 에 비해 훨씬 최근에 밝혀진 곡선이며 충분히 연구되지 않은 실정입니다. 어쩌면 더 중요한 점은, EC-VRF 구성이 검토를 앞둔 초기 초안이며 본 백서를 집필할 당시에 EVM 비잔티움 업그레이드가 진행되고 있었지만 중요한 가치를 다루는 실제 시스템에서는 아직 입증되지 않았다는 사실입니다. 따라서 Orchid 확률적 소액 결제에서 EC-VRF 를 사용하는 것은 지금 당장 실현 가능성이 없으며 Orchid 프로젝트는 EVM 에서 검증할 수 있는 EC-VRF-ALTBN128-KECCAK256 구성을 실제로 사용할 가능성에 대한 추가 연구를 진행하는 데 그 목적이 있습니다.

E.9. 비대화형 결제 방식

섹션 E.8 에서는 Orchid Payment Scheme 의 난수 약정을 VRF 로 대체하면 난수 약정과 관련된 통신 단계들을 제거함으로써 이 결제 방식의 비대화성이 더욱 강화된다는 사실을 보여 줍니다. 발신자가 티켓을 구성하기 전에 수신자가 발신자에게 약정을 전달할 필요 없이 발신자는 공개된 수신자 정보만 사용하여 티켓을 즉시 구성할 수 있습니다.

각 수신자는 특별히 VRF 를 위한 새로운 키 쌍을 생성하고 섹션 4.1 에 자세히 설명된 그 밖의 공개 수신자 정보와 함께 공개 키를 게시합니다. 발신자는 티켓에서 이 공개 키를 구성하면 되며 수신자는 그에 상응하는 개인 키를 사용해 수신된 티켓에 서명하면 됩니다. 섹션 D.4 에 정의된 티켓 검증 로직은 수신자 VRF 서명을 당첨 확률 임계값과 비교되는 값으로 해석합니다.

섹션 E.8 에서 논의한 바와 같이 이것은 결제 방식을 비교적 간단하게 수정하는 셈이 되겠지만 EVM 에서 VRF 검증의 타당성은 추가적인 연구가 필요합니다.

F. 관련 작업

Orchid 프로젝트는 P2P(Peer-to-Peer Network), 블록체인, 암호화 및 오버레이 네트워크의 영역에서 진행되는 대규모의 작업에 의존합니다. Orchid 는 그러한 초기 작업에서 얻은 통찰력을 블록체인 기술, 특히 이더리움[11]과 Zcash[12]에 대한 최신 P2P 연구와 결합합니다.

다음 섹션에서는 Orchid 프로젝트에서 이전 작업이 담당하는 역할에 대해 설명합니다.

F.1. VPN(가상 사설 네트워크)

VPN(가상 사설 네트워크)은 암호화를 사용하여 VPN 가입자의 트래픽을 더 광범위한 미보안 네트워크를 통해 안전하게 전송합니다. 이 암호화를 활용하면 브라우징 습관과 고유한 온라인 식별자(예: 사용자의 IP 주소)의 추적을 방지하여 액세스 제한을 우회할 수 있습니다.

VPN 사용자는 VPN 연결이 실제 보안상 안전하거나 익명성이 보장된다고 단정해서는 안 됩니다. 일부 VPN 서비스 제공업체들은 고객의 네트워크 활동을 추적한 다음, VPN 가입자의 승인이나 지식 없이 데이터를 제 3 의 상거래 기관에 판매합니다. VPN 제공업체 측 네트워크 노드의 IP 주소도 식별할 수 있습니다. 이를 통해 Netflix 와 같은 정부 또는 상거래 기관이 VPN 제공업체 측 서버와 주고 받는 트래픽을 차단할 수 있습니다[13].

VPN 의 그러한 약점들 때문에 분산형 오버레이 네트워크가 개발되었습니다. 분산형 오버레이 네트워크는 지속적으로 변화하는 일련의 출구 노드를 VPN 서비스에 제공합니다. 어떤 사이트가 VPN 출구 노드에서 발신되는 트래픽을 차단하면 하나 이상의 대체 출구 노드가 동적으로 사용됩니다.

F.2. P2P 프로토콜

P2P 프로토콜의 기원은 Napster 파일 공유 네트워크로 거슬러 올라갑니다[42]. Napster 는 구독자가 다른 피어에서 파일을 다운로드 할 수 있는 대가로 음악 파일을 호스팅하도록 권장하기 위해 인센티브를 사용했습니다.

Napster 네트워크

Napster 는 파일과 피어의 위치를 인덱싱하는 중앙 집중식 디렉토리 서비스를 사용했습니다. Napster 의 이러한 접근 방식으로 얻은 지식의 중앙 집중화로 인해 Napster 는 MPAA(Motion Picture Association of America)로부터 소송을 당했습니다. 이에 Napster 는 결국 사업을 접어야 했습니다.

Napster 의 중앙 집중식 디렉토리가 가진 취약성 때문에 Napster 의 승계자인 Gnutella 는 네트워크의 각 피어에 파일 및 노드 주소의 인덱스를 배포하게 되었습니다[43].

Gnutella 네트워크의 분산 인덱스 응답

Gnutella 네트워크 설계자는 분산 인덱스 접근 방식을 구현하여 Napster 의 중앙 집중형 디렉토리가 안고 있던 단점을 해결했습니다. 이 접근 방식은 Napster 보다 향상된 탄력성 및 확장성을 제공했습니다. 이러한 개선은 P2P 네트워크에 인덱스를 배포하기 위한 추가 프레임워크의 개발에도 영향을 미쳤습니다. 주목할만한 한 가지 예를 들자면, P2P 네트워크에서 노드 및 리소스를 효율적으로 검색할 수 있도록 DHT(Distributed Hash Tables)를 채택한 것입니다.

Tor 네트워크

Tor 는 1990 년대 중반 미 해군에서 개발했습니다. 그 이후로 오픈 소스 커뮤니티에서 Tor 개발을 주도했지만 Tor 의 사용은 여전히 제자리걸음인 실정입니다. 현재 전 세계적으로 약 7,000 개의 노드, 3,000 개의 출구 노드 및 약 2 백만 명의 사용자가 있습니다.

Tor 가 중앙 집중식 노드 선택을 사용하고 노드 서비스(출구 노드 서비스를 포함)를 제공하기 위해 자원자들에게 의존하는 것은 Tor 의 처리량에 부정적인 영향을 미칩니다. 그 이유는 Tor 가 BitTorrent 또는 그 외 P2P 파일 공유 시스템을 사용할 수 없고 출구 노드가 출구 트래픽의 내용을 검사할 수 있기 때문입니다. 또한 Tor 는 출구 노드가 다른 사용자들을 대신하여 불법 정보 또는 위험한 정보에 강제로 액세스하는 것을 방지하는 메커니즘이 없습니다.

그러한 문제들이 있음에도 불구하고 비교적 규모가 작은 Tor 의 개발 커뮤니티는 Tor 네트워크의 처리 속도, 신뢰성 및 보안을 어떻게 높일 수 있는지를 계속 조사하고 있습니다[25]. 이 논의의 핵심을 이루는 부분은 지연 시간이 더 짧고 대역폭은 더 높은 노드를 네트워크 내에 전달하는 방법입니다[20, 21, 22, 23, 24].

이러한 목표를 달성하기 위해 Tor 네트워크는 사용자의 인센티브를 제공할 방법을 찾아야 합니다. 사용자로부터 재정적인 기부를 받는 수단을 가지고 Tor 를 개조하면 여러 가지 장애물들이 발생합니다. Tor 는 결제를 라우팅과 긴밀하게 결합할 수 없기 때문에 익명의 디지털 결제를 효과적으로 관리하기가 어렵습니다. 일부 노드는 무료로 계속 라우팅하면서 다른 노드에서는 골드 스타(gold star) 멤버가 되기 위한 비용을 부담해야 한다는 Tor 커뮤니티의 주장은 문제를 오히려 한층 더 복잡하게 만듭니다.

Tor 가 성장 한계를 보이는 또 하나의 비기술적인 이유는 이것이 주로 기술적으로 정교한 사용자(기술 전문가)로 하여금 불법적인 서비스나 숨겨진 웹 사이트에 액세스할 수 있도록 설계된 도구로 종종 인식되기 때문입니다. 다양한 불법 상품 및 서비스를 제공하는 *Silk Road* 웹 사이트는 바로 이러한 유형의 숨겨진 서비스를 보여주는 한 가지 예에 속합니다[18, 19].

이와는 달리, Orchid Network 는 숨겨진 서비스를 사용하도록 설정하지 않고 공개적이면서 보안상 안전한 익명의 인터넷 액세스에만 중점을 둡니다.

양파 라우팅

여기에 설명된 양파 라우팅(및 섹션 F.2 에 설명된 마늘 라우팅)의 제반 기법은 암호화와 결합되어 P2P 네트워크를 통해 더 높은 수준의 익명 라우팅을 제공합니다.

양파 라우팅은 P2P 네트워크를 통해 경로를 생성하는 데이터 암호화에 대한 계층화된 접근 방식입니다. 메시지는 발신 노드에 의해 반복적으로 암호화된 후, 메시지가 통과하는 각 노드에 의해 연속적으로 해독됩니다. 중간 노드는 메시지를 라우팅하는 데 필요한 라우팅 명령만 수신합니다. 라우팅 명령과 메시지를 모두 수신하는 노드는 최종(출구) 노드 밖에 없습니다.

Tor 네트워크는 양파 라우팅에서 자주 인용되는 한 가지 예에 속합니다(섹션 F.2).

마늘 라우팅

I2P(Invisible Internet Project, 보이지 않는 인터넷 프로젝트)는 Tor(F.2)와 유사한 원리를 기반으로 하는 분산화된 익명 네트워크에 속하며, 다만 처음부터 독립적인 다크넷(darknet)으로 설계된 것이 특징입니다. I2P 의 주요 디자인 특징은 마늘 라우팅을 사용한다는 것입니다[**26**].

마늘 라우팅은 여러 메시지를 *구근(bulb)*이라고 하는 단일 패킷으로 함께 묶습니다. 구근 내에 있는 각 메시지는 차례대로 양파 라우팅의 계층화된 암호화 스타일로 암호화됩니다. 여러 메시지를 한 데 묶는다는 것(번들링)은 숨겨진 서비스에서 I2P 에 액세스하는 것이 Tor 보다 훨씬 더 빠르다는 것을 의미합니다. I2P 는 더 넓은 인터넷으로 라우팅하는 과정만 일부분 지원하기 때문에 성능 개선의 효과를 충분히 비교 평가하기는 어렵습니다. 또한 이러한 번들링은 트래픽 분석의 판정을 더욱 어렵게 하는 원인이 되기도 합니다.

I2P 사용자는 Tor 에서 사용하는 중앙 집중식 디렉토리 없이 P2P(Peer-to-Peer) 암호화 터널을 사용하여 서로 연결됩니다. I2P 는 수신 트래픽과 발신 트래픽을 완전히 분리합니다. 그런 다음, 회선 교환 대신에 패킷 교환을 사용하여 여러 피어에 걸쳐 메시지의 투명한 부하 분산을 제공합니다. 이러한 디자인 특징들이 결합되어 보안과 익명성이 모두 향상되었습니다.

상당한 개선이 필요한 I2P 의 한 가지 측면은 노드의 분산 데이터베이스를 관리하는 것입니다. I2P 는 원래 2002 년 당시에 설계된 대로 Kademlia 를 사용했습니다[**27**]. Kademlia 의 초기 버전은 너무 많은 CPU 와 네트워크 대역폭을 계속 소비했기 때문에 확장을 할 수 없었습니다. 그런 다음, I2P 는 *floodfill* 이라고 하는 알고리즘으로 전환했습니다. 그러나 floodfill 메커니즘의 경우, I2P 분산 데이터베이스의 정보를 손상시키고 조작하기 위해 악용될 수 있는 설계 결함에 따른 문제도 발생합니다[**28**].

F.3. 블록체인 플랫폼

블록체인 프로토콜을 사용하면 글로벌 상태 및 암호화 사용에 대한 무허가 방식의 분산된 합의를 통해 노드 실행에 대한 인센티브를 제공할 수 있습니다.

비트코인, 이더리움, Zcash 등과 같은 블록체인 디자인은 상태 전환 함수들을 사용하여 글로벌 상태에서 항목을 추가하거나 변경하는 블록체인 프로토콜의 예에 속합니다. 또한 이들 프로토콜은 작업 증명과 같은 기법들을 사용하여 거래를 확인하고 주문 내역에 대한 합의를 형성한 노드에 보상을 제공합니다[**44**].

이더리움

이더리움[55]은 비트코인[78]과 함께 새로운 형태의 애플리케이션별 암호화 토큰을 개척했습니다[33]. 블록체인 시스템을 사용하면 임의로 선택한 계산 방법에 따라 스마트 계약을 지원함으로써 투표, 관리 기능, 수수료 결제 등 애플리케이션별 기능들을 제공하는 맞춤형 원장을 생성할 수 있습니다.

이더리움은 *스마트 계약*을 실행하고 배포할 수 있는 기능을 갖춘 분산형 블록체인 플랫폼입니다. 이더리움의 스마트 컨트랙트 코드는 불변하며 실행 시 완전히 결정론적인 특성을 보입니다(비결정론적 행동을 명시적으로 추가하지 않는 한). 이를 통해 모든 노드가 스마트 계약의 실행을 확인하고 애플리케이션 상태의 변경 사항을 감사할 수 있습니다. 그 결과, 이더리움의 스마트 계약은 프로그래밍된 대로 정확히 실행할 수 있습니다.

이더리움 애플리케이션은 강력한 공유 글로벌 인프라를 기반으로 하여 실행됩니다. 애플리케이션은 가치를 빠르게 이전하고 자산의 소유권을 나타낼 수 있기 때문에 소프트웨어 개발자는 애플리케이션을 활용해 시장을 창출하고 부채 또는 약정의 레지스트리를 저장하며 과거에 마련된 규칙에 따라 자금을 이동하는 등 다양한 작업을 수행할 수 있습니다. 모든 작업은 타사 제공업체 또는 거래 상대방의 위험 없이 진행됩니다.

대체로 서버 다운타임, 손상 및 사기 행위와 같은 문제들을 처리해야 하는 신흥 시장이야말로 그 어떤 곳보다 이더리움의 유용한 기능을 십분 활용할 수 있습니다.

이더리움과 Orchid Market ERC20 토큰

이더리움 네트워크에 배포된 토큰들은 대부분 ERC20 표준을 준수합니다[44]. 이 표준은 토큰을 하드웨어 또는 소프트웨어 사용자 지갑에 쉽고 빠르게 통합할 수 있는 토큰 및 메타데이터를 전송하기 위해 작고 단순한 API 를 지정합니다.

예를 들면, Augur[37] 및 Gnosis[38] 플랫폼은 ERC20 토큰을 사용하여 토큰 데이터에서 예측 시장을 만듭니다. ERC20 을 사용하면 임의의 스마트 계약이 Orchid 프로토콜과 쉽게 인터페이스로 연결될 수 있습니다. 이 토큰은 인터넷 엔드포인트에 안전하게 액세스하려는 IoT 기기에서 유용하게 쓰일 수 있습니다.

또한 ERC20 사양을 준수함으로써 토큰 교환 및 애플리케이션이 새로운 유형의 ERC20 토큰에서 제공하는 확장된 기능들을 보다 쉽게 활용할 수 있게 됩니다. 그 결과, 다양한 애플리케이션에서 ERC20 사양을 준수하는 토큰을 사용하여 정보와 상태를 교환할 수 있게 됩니다.