ChatGPT

수탁형 지갑 설계 – 서명, Nonce, 수수료 모델에 대한 배경 지식

이 문서는 수탁형(托管) 지갑을 설계하고 운영할 때 필요한 서명·nonce·수수료 모델 관련 지식을 정리한 자료입니다. 강의용 스크립트로는 제공된 간략한 개요만으로 충분하지 않으므로, 강사가 돌발 질문에 대응할 수 있도록 각 주제의 핵심 배경을 상세히 설명합니다.

1. 트랜잭션 서명 vs 메시지 서명

1.1 한 줄 정의

- **트랜잭션 서명(온체인 서명)**: 계정의 nonce, 가스 가격/한도, 수수료, 목적지 주소 등 RLP로 직렬화된 거래 데이터를 서명합니다. 이는 **"이 체인에서, 이 nonce로, 이 상태 변화를 실행하라"**는 온체인 명령에 대한 서명입니다. 서명 값이 네트워크로 전파되면 즉시 체인에서 상태 변경(자산 이동·컨트랙트 실행 등)을 발생시킬 수 있습니다.
- **메시지 서명(오프체인 서명)**: 로그인, 권한 위임, 주문 등의 비즈니스 로직에 동의한다는 증거를 남기기 위해 문자열이나 구조화된 데이터를 서명합니다. 이 서명은 **"내가 이 내용에 동의했다/권한을 위임했다"**는 의미이며, 서명 자체는 체인에 기록되지 않습니다. 서명된 데이터와 서명값을 후에 컨트랙트나 서버에 제출하여 실행해야 온체인 상태 변화가 일어납니다.

1.2 트랜잭션 서명의 구조와 보호기제

1. **RLP 직렬화**: 트랜잭션은 `(nonce, gas price, gas limit, to, value, data…)`를 RLP(Recursive Length Prefix)로 직렬화한 뒤 서명합니다. 이 구조는 변경이 일어날 때마다 서명값이 완전히 바뀌므로 공격자가 데이터를 일부만 바꿔 재사용할 수 없습니다 [1].
2. **Chain ID 포함(EIP-155)**: 초기에는 동일한 서명값이 포크 체인에서 재사용되는 문제가 있었습니다. EIP-155는 서명 시 RLP 항목에 `chainId`와 더미값 두 개를 추가하도록 정의합니다 [2]. 이때 서명의 `v` 값은 `(0 또는 1) + 2×chainId + 35/36` 형태가 되어, 다른 체인에서 같은 서명을 재사용할 수 없게 합니다 [2].
3. **Nonce**: 트랜잭션에는 계정별 **nonce**가 포함됩니다. 이것은 계정이 지금까지 보낸 트랜잭션 수를 나타내는 순번으로, 같은 주소에서 같은 nonce를 가진 두 개의 트랜잭션은 존재할 수 없습니다. Nonce가 순차적이기 때문에 트랜잭션 순서가 보장되고, 중복 제출(replay)이 방지됩니다 [3].

1.3 메시지 서명의 구조와 EIP-191/EIP-712

1. **EIP-191 표준**: EIP-191은 모든 메시지 서명 형식을 `0x19 <version byte> <version-specific data> <data>`로 정의합니다 [4]. 첫 바이트 `0x19`는 서명 데이터가 RLP 인코딩된 거래와 혼동되지 않도록 하기 위해 선택되었습니다 [5]. 버전 `0x45`(ASCII 문자 `E`)는 개인용 `personal_sign` 메시지 형식으로, 서명 데이터 앞에 `"\x19Ethereum Signed Message:\n" + length(message)` 프리픽스를 붙여 해싱하는 것이 포함됩니다 [6]. 이 방식은 메시지가 RLP 거래처럼 해석될 가능성을 줄이고 서명을 검증하기 쉬운 형태로 만듭니다.
2. **EIP-712 표준**: 단순 문자열 대신 **구조화된 데이터**를 서명하기 위한 표준입니다. 서명하는 digest는 `\x19\x01 || domainSeparator || hashStruct(message)` 형식으로 계산됩니다 [7]. `domainSeparator`는 EIP-712Domain 구조체를 해시한 값이며, `name`, `version`, `chainId`, `verifyingContract`, `salt` 등의 필드를 가집니다 [8]. 특히 `chainId` 필드가 현재 체인과 다르면 지갑은 서명을 거부해야 하며 [9], `verifyingContract` 필드로 특정 컨트랙트에만 서명이 사용되도록 강

제하여 **도메인 분리**를 구현합니다 [10]. EIP-712는 EIP-191의 버전 `0x01`을 사용하므로 데이터 앞에 `0x1901` 프리픽스를 붙입니다 [11].

3. **메시지 서명의 위험과 피싱 공격**: 지갑 사용자들은 메시지 서명은 가스 비용이 들지 않고 "그냥 로그인"으로 보이기 때문에 방심하기 쉽습니다. 하지만 메시지 서명은 나중에 컨트랙트의 `permit` 함수처럼 자산을 이동시키거나 권한을 위임하는 명령으로 사용될 수 있습니다. 예를 들어 `eth_sign`은 임의의 바이트를 서명할 수 있어 악의적인 dApp이 보낸 난독화된 메시지를 사용자가 그대로 서명할 경우 공격자가 자산을 훔칠 수 있습니다 [12]. MetaMask는 오프체인 서명을 악용하는 **signature phishing** 공격을 경고하며, 공격자는 가짜 dApp을 만들어 사용자에게 "로그인" 메시지 서명을 요청한 뒤 이를 실제 토큰 승인(permit)으로 제출할 수 있다고 설명합니다 [13]. 따라서 서명 요청 창에 보여지는 문구를 반드시 확인해야 하며, 이상한 메시지는 서명하지 않는 것이 중요합니다.

1.4 비교 요약

항목	트랜잭션 서명	메시지 서명
목적	이더 송금·컨트랙트 실행 등 온체인 상태 변경	로그인, 토큰 승인(Permit), 주문 체결 등 오프체인 증명·위임
데이터 형식	`(nonce, gasPrice/gasLimit, to, value, data, chainId…)`를 RLP로 직렬화 후 서명 [1]	`0x19 <버전> <버전별 프리픽스> <메시지>`(EIP-191) 또는 `\x19\x01 || domainSeparator || hashStruct(message)`(EIP-712) 형식 [5] [7]
replay 보호	Nonce와 `chainId`가 포함돼 동일 서명이 다른 체인/순번에서 재사용되는 것을 막음 [2]	데이터에 nonce나 unique id가 없으면 동일 서명이 여러 번 제출되어 실행될 수 있음 – 애플리케이션 레벨에서 별도 nonce/expiry/unique id 관리 필요 [14]
위험 요소	가스비 미지정·chainId 잘못 설정 등 운영 사고 발생 가능	사용자가 이해하지 못하는 메시지 서명을 할 경우 피싱·권한 오남용 위험 [12] [13]
표준	EIP-155로 chainId 포함, EIP-1559로 fee 구조 개선	EIP-191 기본 프리픽스, EIP-712 구조화 서명, EIP-2612(permit) 등

2. 왜 사고가 나는가? – 피싱과 메시지 서명의 오남용

1. **블라인드 서명과 피싱**: 많은 지갑 인터페이스는 메시지를 해석하지 않고 단순히 "Sign" 버튼만 제공합니다. 공격자는 **eth_sign** 기능을 이용해 이해하기 어려운 해시 문자열을 보내고, 사용자가 로그인이나 단순 확인으로 착각하게 만듭니다. 바이낸스 보안 블로그는 eth_sign의 서명값이 잘못 사용될 경우 공격자가 사용자의 자산에 대한 완전한 제어권을 얻을 수 있다고 경고합니다 [15]. MetaMask도 사용자가 오프체인 서명을 악용하는 signature phishing을 주의해야 한다고 설명하며, 가짜 NFT 마켓플레이스나 cross-chain 브리지 UI가 사용자에게 메시지를 서명하게 만든 뒤 이를 permit2 승인으로 제출하는 사례를 제시합니다 [16].
2. **허위 로그인/연결(Sign)**: "지갑 연결"이라고 표시된 버튼을 누르면 실제로는 `permit()` 서명 요청일 수 있습니다. EIP-2612 Permit과 Uniswap의 **Permit2** 등은 오프체인 서명으로 토큰 승인 또는 NFT 대량 리스팅을 가능하게 하는데, 사용자가 서명 내용의 의미를 제대로 보지 않으면 무제한 승인 권한을 부여할 수 있습니다. Revoke.cash는 permit 서명이 사용자의 승인 정보를 담고 있어 서명을 악용한 스캐머가 자산을 훔칠 수 있음을 경고합니다 [17].
3. **방어 전략**: EIP-712의 구조화된 서명은 서명하려는 데이터 구조를 지갑 인터페이스에 명시적으로 표시하여 사용자가 내용을 확인할 수 있게 합니다. 또한 `domainSeparator`에 체인 ID와 verifying contract 주소를 포함해, 서명이 다른 앱이나 체인에서 재사용되지 않도록 합니다 [8]. 이외에도 지갑에서는 메시지 서명 시 도메인 정보와 실제 서명 데이터를 보여주어 사용자에게 경고해야 합니다.

3. 리플레이(Replay) 공격: 트랜잭션 vs 메시지

3.1 트랜잭션 리플레이 – EIP-155

과거에는 같은 서명값이 여러 체인에서 재사용되는 문제가 있었습니다. EIP-155는 트랜잭션 서명 시 `chainId`를 추가해 서명 해시를 `(nonce, gasprice, startgas, to, value, data, chainId, 0, 0)`으로 계산하도록 하고, 서명의 `v` 값을 `chainId * 2 + 35/36`으로 설정했습니다 [2]. 이로써 **"tx 서명은 chainId로 체인 경계를 박는다"**는 원칙이 확립되었으며, 사용자가 다른 체인으로 잘못 서명하면 곧바로 자산 손실로 이어질 수 있습니다. 실제 운영에서는 메인넷과 테스트넷을 혼동하지 않도록 각 체인에 맞는 chainId를 하드코딩하거나 지갑에서 명확히 표시해야 합니다.

3.2 메시지 리플레이 – EIP-712 및 idempotency

메시지 서명은 자동으로 nonce가 붙지 않기 때문에 서명을 여러 번 제출하면 같은 명령이 반복 실행될 수 있습니다. EIP-712의 **Security Considerations**에서는 서명된 메시지를 재사용(replay)할 수 없도록 애플리케이션이 서명 digest를 저장하거나 nonce/만료 시간을 포함해야 한다고 강조합니다 [14]. 일반적으로 off-chain 서명을 사용하는 컨트랙트(예: Permit, meta-transaction relay)는 다음과 같은 방식을 취합니다.

- 메시지에 `nonce`, `deadline`(expiry) 또는 `uid`를 포함해 **한 번만** 사용될 수 있도록 합니다. EIP-2612 Permit 구조체에는 `nonce`와 `deadline` 필드가 존재합니다.
- 컨트랙트 또는 서버에서 `usedNonces`(mapping) 혹은 `usedDigests`(해시 테이블)를 유지해 이미 사용된 서명을 거절합니다.
- idempotent한 처리: 동일한 서명이 두 번 제출되어도 상태 변화가 중복 발생하지 않게 설계합니다.

4. Nonce, 트랜잭션 교체(Replacement), Dropped/Replaced 상태

4.1 Nonce의 역할과 순서 보장

- Ethereum에서 **nonce**는 계정이 보낸 트랜잭션의 총 수를 나타내는 **순번 카운터**입니다. 각 주소의 nonce는 0부터 시작하며, 트랜잭션을 전송할 때마다 1씩 증가합니다 [3]. 같은 주소에서 같은 nonce 값을 가진 두 트랜잭션은 존재할 수 없고, nonce 값이 순차적으로 증가해야 합니다 [18].
- Nonce 덕분에 네트워크의 모든 노드는 트랜잭션을 동일한 순서로 처리해 **이중 지불** 및 순서 변경을 방지합니다 [19] [20]. 높은 nonce를 가진 트랜잭션은 낮은 nonce 트랜잭션이 처리될 때까지 pending 상태로 남게 됩니다 [21].

4.2 Replacement 트랜잭션과 Geth/Besu의 정책

- **같은 nonce로 여러 트랜잭션을 보내면** 블록체인 노드들은 기존 트랜잭션을 새 트랜잭션으로 "교체"(replacement)할 수 있습니다. 그러나 노드들은 스팸을 막기 위해 **fee bump 조건**을 둡니다. Geth의 `txpool.pricebump` 기본값은 10%로, 새 트랜잭션의 가스 가격이 기존 트랜잭션보다 10% 이상 높아야 교체됩니다 [22].
- EIP-1559 이후 fee 구조가 복잡해져, Besu의 문서에는 **두 가지 조건 중 하나**를 만족해야 교체가 일어난다고 명시합니다 [23]:
- 새 트랜잭션의 **효과적 가스 가격(effective gas price)** > `(1 + priceBump)` × 기존 가스 가격 **AND** 새 **효과적 priority fee** ≥ 기존 priority fee.
- 새 트랜잭션의 **효과적 가스 가격이 기존과 같고**, 새 **효과적 priority fee**가 `(1 + priceBump)` 이상 증가.

여기서 효과적 priority fee는 1559 트랜잭션에서 `min(maxFee - baseFee, maxPriorityFee)`로 계산됩니다 [24]. Geth는 `maxFeePerGas`와 `maxPriorityFeePerGas`를 모두 10% 이상 올리지 않으면 **"replacement transaction underpriced"** 오류를 반환합니다.

- **Dropped / Replaced**: Etherscan은 dropped & replaced 상황을 이렇게 설명합니다. dropped는 노드들이 트랜잭션을 더 이상 브로드캐스트하지 않아 블록체인에 기록되지 않는 상태이고, 동일 nonce의 다른 트랜잭션이 해당 nonce를 차지하면서 replaced가 발생합니다 [25]. dropped된 트랜잭션은 실행되지 않았으므로 자산과 가스비가 반환되고 nonce가 다시 사용됩니다 [26]. 여러 트랜잭션이 같은 nonce를 사용하면 가스비가 높은 트랜잭션만 채굴되고 나머지는 dropped됩니다 [27].

- **운영 모범 사례**:

- 출금 1건에 대해 **TxAttempt** 엔티티를 여러 개 만들어 (from, nonce)를 기준으로 추적합니다. 트랜잭션 해시가 아니라 `(from, nonce)`로 상태를 관리해야 replacement나 dropped 상황을 올바르게 처리할 수 있습니다.
- 브로드캐스터는 재전송 시 **수수료 증가 정책**을 코드로 구현합니다. 예: priority fee와 max fee를 각각 10~15% 이상 올려야 합니다.
- 긴 대기 끝에 mempool에서 사라지는 상황을 고려해 `txpool.lifetime` 기본 3시간 (Geth) [28] 정도 지나면 새로운 nonce로 다시 보내는 로직을 마련합니다.

4.3 Nonce 관리 실무 팁

- 높은 빈도로 트랜잭션을 보내는 서비스는 노드의 자동 nonce 할당에 의존하지 말고 **서버 측에서 nonce를 직접 관리**해야 합니다. Tatum 문서는 대량 전송 시 자동 계산이 충분히 빠르지 않아 replacement 오류가 발생할 수 있으며, 각 주소별 nonce를 수동으로 추적해야 한다고 권고합니다 [29].
- 트랜잭션이 계속 pending 상태로 남는다면 **"speed up / cancel"** 기능으로 같은 nonce에 더 높은 수수료를 지정하여 교체합니다. 반대로 수수료를 충분히 높이지 않으면 `replacement transaction underpriced` 오류가 발생합니다.

5. EIP-1559 수수료 모델

5.1 기본 구조

EIP-1559는 Ethereum의 fee 시장을 전면적으로 개편했습니다. 주요 내용은 다음과 같습니다:

1. **Base Fee per gas (baseFee)**: 프로토콜에 의해 블록마다 조정되는 최소 가스비입니다. 전 블록이 목표 가스 사용량보다 높으면 baseFee가 증가하고, 낮으면 감소합니다 [30]. baseFee는 **소각(burn)**되므로 채굴자/검증인은 받지 못합니다 [31].
2. **Priority Fee (tip)**: 사용자가 블록 생산자(검증자)에게 지급하는 팁으로, 채굴자가 수수료를 받는 부분입니다 [32]. 네트워크 혼잡 시 priority fee를 높이면 거래가 더 빨리 처리됩니다.
3. **Max Fee**: 사용자가 지불할 최대 수수료로, `maxFeePerGas` 필드에 지정합니다. 실제 지불액은 `baseFee + priorityFee`이며, 이 합이 `maxFee`보다 높으면 트랜잭션은 거부됩니다 [32]. `maxFee`는 최악의 경우 지불 의향을 나타내는 상한선입니다.
4. **Dynamic adjustment**: baseFee는 블록 가스 사용량과 목표값에 따라 예측 가능한 범위 내에서 증가/감소하며, 지갑이 자동으로 수수료를 계산할 수 있게 합니다 [33].

5.2 Replacement와의 결합

- **기존 수수료 교체 규칙**: 1559 트랜잭션을 교체하려면 `maxFee`와 `maxPriorityFee`를 모두 10% 이상 올려야 하는 것이 일반적입니다. Geth에서는 fee bump가 부족하면 `replacement transaction underpriced` 오류가 발생하며, Besu는 효과적 가스 가격과 priority fee 조건을 모두 확인합니다 [23].
- **팁만 올리기 vs Max Fee도 올리기**: 혼잡 시 baseFee가 급격히 상승할 수 있으므로 priority fee만 올리면 `baseFee + tip`이 기존 `maxFee`를 초과하여 트랜잭션이 무효화될 수 있습니다. 따라서 교체 시에는 **priority fee와 max fee를 함께** 올려야 합니다.

6. EIP-712(typed signing): 피싱 방어의 실전 도구

6.1 동기와 구조

- EIP-712는 사용자가 **구조화된 데이터**를 서명하도록 하여 무의식적 서명(블라인드 서명)을 방지하는 표준입니다. 서명 데이터는 `\x19\x01 || domainSeparator || hashStruct(message)` 형식으로 계산합니다 [7]. 여기서 `domainSeparator`는 애플리케이션 이름, 버전, chainId, verifyingContract, salt로 구성됩니다 [8]. 이는 **도메인 분리** 기능을 제공하여 서명이 다른 체인이나 다른 컨트랙트로 재사용되는 것을 크게 줄입니다.
- user-agent(지갑)는 도메인 정보를 표시하고, 사용자가 실제로 어떤 계약에 서명하는지 확인할 수 있게 해야 합니다. chainId가 현재 체인과 다르면 지갑은 서명을 거부해야 합니다 [9].
- **Security considerations**: EIP-712 표준 자체는 서명 및 검증 방법만 정의하며, 서명의 재사용을 막는 nonce/expiry 처리 등은 애플리케이션 구현자에게 맡깁니다. 같은 서명이 두 번 제출될 때 애플리케이션이 거부하거나 idempotent하게 처리해야 한다는 점을 명시합니다 [14].

6.2 사용 사례와 주의 사항

- **Permit(EIP-2612)**: ERC-20 토큰을 `approve()` 대신 off-chain 서명으로 승인하는 기능입니다. permit 메시지는 EIP-712을 기반으로 하며 spender, value, nonce, deadline을 포함합니다. Revoke.cash는 permit 서명이 가스 비용을 줄이지만 메시지 서명에 익숙하지 않은 사용자가 무제한 승인을 서명하는 피싱에 취약할 수 있다고 경고합니다 [17].
- **Permit2**: Uniswap의 Permit2는 여러 토큰에 대해 하나의 컨트랙트에서 승인 및 호출을 수행하도록 설계하여 사용자 경험을 개선합니다. 그러나 공격자는 permit2 서명만 받아도 사용자의 NFT나 토큰을 마음껏 전송할 수 있으므로, 지갑은 도메인 및 메시지 내용을 명확히 표시해야 합니다 [16].

7. 질문에 대비한 추가 지식과 모범 사례

7.1 서명 유형 구분을 위한 사용자 인터페이스 설계

- 트랜잭션 서명 팝업은 보통 가스비·nonce·chainId 등을 표시하고 최종 금액을 명확히 보여줘야 합니다. 반면 메시지 서명은 `eth_sign`처럼 날것의 해시가 아닌 EIP-712 구조화 데이터로 보여주어 사용자가 내용을 이해하고 확인할 수 있게 해야 합니다.
- 지갑 개발자는 서명창에서 **도메인 이름과 verifying contract 주소**를 강조하고, 사용자가 연결하려는 dApp의 도메인과 일치하는지 확인할 수 있도록 해야 합니다.

7.2 체인 ID와 네트워크 분리

- Ethereum 메인넷과 여러 테스트넷/레이어2의 chainId가 다르므로, 서명 시 항상 올바른 chainId를 사용해야 합니다. EIP-155로 chainId가 서명에 포함되어 있어 다른 체인에서 서명이 재사용되지 않지만, 잘못된 chainId를 지정하면 자산이 잘못된 체인으로 전송될 수 있습니다.
- 지갑과 백엔드는 체인 전환 시 사용자의 서명 요청을 취소하고 새로운 chainId를 설정한 뒤 다시 요청해야 합니다.

7.3 메타트랜잭션과 가스리스 거래

- 메타트랜잭션은 사용자가 서명만 하고, relayer가 수수료를 지불하여 트랜잭션을 보내는 방식입니다. 이때 사용자는 off-chain 메시지를 서명하므로 EIP-712 구조체에 nonce와 relayer의 주소, 수수료 등을 포함하여 replay를 막아야 합니다.

7.4 기타 EIP:

- **EIP-2930 (Access List)**: EIP-1559와 함께 도입된 트랜잭션 타입 0x01로, 트랜잭션에서 미리 접근할 storage 키 목록(access list)를 제공해 gas 비용을 예측 가능하게 합니다. 이는 수수료 모델과 nonce 교체 전략에도 영향을 줄 수 있습니다.
- **EIP-4337(Account Abstraction)** 및 **ERC-4337**: 지갑을 스마트 컨트랙트로 구현하고, 거래 수수료를 다른 토큰으로 지불하거나 번들러를 통해 처리하는 구조입니다. 향후 수탁형 지갑에서 사용자 경험을 개선하기 위해 참고할 수 있습니다.

8. 결론 및 핵심 메시지

1. **지갑은 키를 보관하는 것이 아니라 서명 권한을 안전하게 관리하는 시스템**입니다. 서명에는 온체인 상태 변화를 위한 트랜잭션 서명과 오프체인 의사표시를 위한 메시지 서명이 있다는 점을 명확히 구분해야 합니다.
2. **트랜잭션 서명**은 RLP 구조와 nonce·chainId가 결합되어 있으며, EIP-155로 체인 경계를 고정합니다 [2]. **EIP-1559**로 baseFee/tip/maxFee 구조가 도입되면서 교체 시 수수료 정책을 세심하게 관리해야 합니다 [31]. Nonce는 트랜잭션 순서를 보장하며, 같은 nonce를 사용할 경우 가스비를 충분히 올려야 교체가 가능합니다 [23].
3. **메시지 서명**은 자체적으로 블록체인 상태를 변경하지 않지만, 권한 위임이나 주문 체결에 사용되므로 피싱의 핵심 표적입니다. EIP-191의 프리픽스와 EIP-712의 도메인 분리는 메시지가 RLP 거래와 혼동되거나 다른 도메인에서 재사용되는 것을 방지합니다 [5] [8]. 구현자는 메시지에 nonce/만료를 포함하고 재사용을 방지해야 합니다 [14].
4. **수탁형 지갑 운영자**는 (from, nonce) 단위의 상태 머신을 구축하여 각 트랜잭션 시도를 추적하고, 수수료 증가 로직을 자동화하며, dropped/replaced 상황에 대응해야 합니다. 메시지 서명 기능에서는 EIP-712를 적극적으로 활용하고 서명 데이터의 의미를 사용자에게 명확히 보여줘야 합니다.
5. 마지막으로, 사용자 교육과 UI 설계가 중요합니다. 피싱 공격은 기술적인 취약점보다는 사람의 방심을 노리므로, **서명 내용을 읽고 이해하는 습관**을 키우게 하고, 지갑·dApp은 서명 창에 충분한 정보를 제공해야 합니다.

---

[1] Transaction Signatures vs Message Signatures: Understanding the Difference
https://www.kayssel.com/post/web3-20/

[2] EIP-155: Simple replay attack protection
https://eips.ethereum.org/EIPS/eip-155

[3] [19] [20] [21] What is a nonce in Ethereum (blockchain)?
https://thirdweb.com/learn/guides/what-is-an-ethereum-nonce-in-blockchain

[4] [5] [6] ERC-191: Signed Data Standard
https://eips.ethereum.org/EIPS/eip-191

[7] [8] [9] [10] [14] EIP-712: Typed structured data hashing and signing
https://eips.ethereum.org/EIPS/eip-712

[11] The Magic of Digital Signatures on Ethereum | MyCrypto Blog
https://blog.mycrypto.com/the-magic-of-digital-signatures-on-ethereum

[12] [15] Web3 wallet security: blockchain message signature risks | Binance Blog on Binance Square
https://www.binance.com/en/square/post/14545618347818

13 16 Signature phishing | MetaMask Help Center
https://support.metamask.io/stay-safe/protect-yourself/wallet-and-hardware/signature-phishing

17 What Are EIP2612 Permit Signatures? | Revoke.cash
https://revoke.cash/learn/approvals/what-are-eip2612-permit-signatures

18 25 26 27 Etherscan Information Center | What happens when a Dropped Transaction is replaced by another Transaction?
https://info.etherscan.com/dropped-and-replaced-meaning/

22 28 Command-line Options | go-ethereum
https://geth.ethereum.org/docs/fundamentals/command-line-options

23 24 Transaction pool | Besu documentation
https://besu.hyperledger.org/public-networks/concepts/transactions/pool

29 Replacement Transaction Underpriced
https://docs.tatum.io/docs/replacement-transaction-underpriced

30 31 32 33 EIP-1559: Fee market change for ETH 1.0 chain
https://eips.ethereum.org/EIPS/eip-1559