

PrivGas: A Privacy-Preserving Gas Sponsorship Protocol for Blockchain Stealth Addresses

M M Imran
Department of Computer Science
Wayne State University
Detroit, MI, USA
mmimran@wayne.edu

Anik Tahabilder
Department of Computer Science
Wayne State University
Detroit, MI, USA
tahabilderanik@wayne.edu

Changxin Bai
Department of Computer Science
Kettering University
Flint, MI, USA
cbai1@kettering.edu

Qiang Zhu
Department of Computer and Information Science
University of Michigan-Dearborn
Dearborn, MI, USA
qzhu@umich.edu

Shiyong Lu
Department of Computer Science
Wayne State University
Detroit, MI, USA
shiyong@wayne.edu

Abstract—ERC-5564 stealth addresses enable privacy-preserving transfers by deriving fresh destination accounts. However, these accounts lack the native ETH required to pay outbound transaction gas. Funding them from known wallets reintroduces linkability and has been shown to deanonymize a large fraction of Umbra users. Under the ERC-4337 account-abstraction standard, where *UserOperation.sender* is structurally public, stealth-account gas sponsorship becomes a Sybil-resistant admission-control problem, that is, the protocol must avoid recipient-side gas top-up linkage while limiting what the paymaster learns during credit redemption. We propose PrivGas, a three-stage privacy-preserving gas sponsorship protocol (Fund, Bootstrap, and Spend) that sequentially binds a non-refundable fee to a public announcement, sponsors a credit deposit into a zero-knowledge Shielded Credit Pool, and redeems the credit using a Merkle-membership proof and domain-separated nullifier. The PrivGas architecture separates the non-refundable protocol fee from recoverable forwarded value to eliminate credit-farming incentives. This avoids a recipient-controlled gas top-up transaction and hides the redeemed commitment index from the paymaster. We formalize the threat model, derive a Sybil-resistance fee bound over total sponsored gas exposure, and evaluate an ERC-4337-compatible implementation designed to follow the ERC-7562 account-abstraction validation-scope rules. In local Foundry simulation, our PrivGas prototype validates a Semaphore Groth16 proof at a peak cost of 284,527 gas, below the 500,000-gas verification limit specified by ERC-4337. These gas figures are not testnet or mainnet transaction measurements. Local ERC-4337 integration tests confirm that PrivGas rejects invalid redemption and sponsorship conditions, while preliminary validation-scope checks support its ERC-7562-oriented storage design.

Index Terms—Blockchain Privacy, Stealth Addresses, Account Abstraction, Gas Sponsorship, Paymasters, Zero-Knowledge Proofs, Sybil Resistance

I. INTRODUCTION

The transparent ledger of public blockchains fundamentally conflicts with the financial privacy required for everyday transactions. To mitigate the traceability of on-chain asset transfers, stealth address protocols allow a sender to derive a one-time,

cryptographically secure recipient address from a published meta-address [1], [2]. This ensures that no two payments to the same recipient share an identifiable on-chain destination. The ERC-5564 standard [3] formally introduces this capability to the Ethereum ecosystem, allowing for decentralized, non-interactive generation of stealth accounts.

However, a structural weakness remains at the core of this architecture: a freshly derived stealth address inherently holds no native currency (ETH) [4]. Consequently, the recipient is completely unable to pay the base execution gas required to move or manage their received assets. To circumvent this, users typically fund the stealth address from a known wallet, inadvertently reintroducing the exact on-chain linkability the protocol was designed to eliminate. A recent measurement study of Umbra, currently the most widely deployed ERC-5564 system, revealed that up to two-thirds of real-world recipients were completely deanonymizable strictly through these funding and withdrawal heuristics [5].

Third-party relayer networks can reduce funding friction, but they do not solve the privacy leakage, as the relayer must still be compensated. The advent of ERC-4337 Account Abstraction [6] introduced the *Paymaster*, a smart contract capable of flexibly sponsoring transaction gas for users. However, hiding the identity of the sponsorship requester is insufficient under ERC-4337 because the *UserOperation.sender* field is structurally visible to both off-chain bundlers and the on-chain paymaster. We argue that gas sponsorship for stealth smart accounts must be treated not as a relaying convenience, but as a Sybil-resistant admission-control problem, where the critical property is preventing recipient-side gas top-up linkage while preserving cryptographic unlinkability between shielded credit commitments and later redemptions.

Recent account-abstraction research has examined gas sponsorship [7], yet no existing work offers a complete solution for Ethereum Virtual Machine (EVM)-compatible stealth smart accounts. BaseSAP [2], the foundational work expanding

upon ERC-5564, acknowledges the gas-funding paradox and sketches a theoretical market of transaction aggregators that sell prepaid gas “tickets” via Chaumian blind signatures. However, this approach relies on a semi-trusted off-chain aggregator, and, to the best of our knowledge, no mainnet deployment of this gas-ticket mechanism has been reported.

SuperPaymaster [8] attempts to decentralize gas sponsorship by binding eligibility to an Asset-Oriented Abstraction, specifically, a non-transferable Soulbound Token (SBT) known as a Gas Card. While highly Sybil-resistant, this persistent token introduces a visible on-chain eligibility marker and therefore does not target stealth-recipient unlinkability. Conversely, systems like StealthHub [9] achieve strong privacy by unifying stealth-address discovery with an unspent transaction output (UTXO)-based zero-knowledge tumbler. This changes the custody and composability model: funds reside as notes inside the protocol rather than directly at an independently executable EVM account. Table I compares these systems against three practical criteria a complete solution should satisfy. No existing system satisfies all three criteria jointly for ERC-4337 stealth smart accounts while avoiding recipient-side gas top-up linkage.

This work makes the following contributions:

- We formalize stealth-address gas funding as a Sybil-resistant admission-control problem for ERC-4337 paymasters, identifying recipient-side gas top-up unlinkability as the key privacy requirement.
- We design **PrivGas**, a privacy-preserving gas sponsorship protocol that separates announcement, shielded credit creation, and proof-based redemption while preventing credit farming and hiding the redeemed commitment index.
- We implement PrivGas as an ERC-4337-compatible prototype that validates a Semaphore Groth16 proof at 284,527 gas, 43.1% below the 500,000-gas ERC-4337 verification limit, and rejects five invalid redemption and sponsorship conditions in local integration tests.

The PrivGas implementation and evaluation artifacts are publicly available online.¹

II. TECHNICAL BACKGROUND

This section summarizes the core technical foundations required to understand the design and operation of PrivGas.

A. ERC-5564 Stealth Addresses

Stealth address protocols allow a sender to transfer assets to a recipient without revealing the recipient’s public identity on-chain [1]. In ERC-5564 [3], the sender uses the recipient’s meta-address to derive a one-time destination address *addr* and broadcasts the associated ephemeral public key through an *Announcer* contract.

¹<https://github.com/imranpollob/stealth-protocol>

TABLE I
COMPARISON WITH CLOSEST PRIOR SYSTEMS

System	Sybil-res.	Recipient top-up unlinkability	ERC-4337 stealth account support
Umbra (deployed) [5]	✗	✗	✗
BaseSAP ticket aggregator [2]	Partial	Partial ^a	✓
SuperPaymaster [8]	✓	✗ ^b	✓
StealthHub [9]	✓	✓	✗ ^c
PrivGas (ours)	✓ ^d	✓ ^e	✓

^aRelies on a semi-trusted blind-signature aggregator with no reported mainnet deployment.

^bSponsorship is bound to a visible, persistent Soulbound Token.

^cFunds are held as UTXO notes in a tumbler, not at an independently executable stealth account.

^dUnder Eq. (1).

^ePrevents known-wallet gas top-up linkage and hides the redeemed commitment index. Timing, account-level, and usage side channels are outside the guarantee.

B. ERC-4337 Account Abstraction and Paymasters

ERC-4337 [6] replaces externally owned account (EOA)-submitted transactions with *UserOperations*, which are collected by bundlers and executed through a singleton *EntryPoint* contract. Each *UserOperation* specifies the executing *sender* and its execution parameters. It introduces *Paymasters*, which may sponsor gas for a *UserOperation*. Sponsorship is decided during the restricted *validatePaymasterUserOp* phase. If validation succeeds, the paymaster covers the native gas cost, allowing the *sender* to execute without holding native currency. ERC-7562 [10] defines validation-scope rules that restrict opcode and storage access during *UserOperation* validation while allowing a staked paymaster to access its own storage.

C. Intersecting Stealth Addresses and ERC-4337

For ERC-4337 compatibility, the stealth destination *addr* must be a smart account with *validateUserOp* logic rather than an EOA. We model it as a counterfactual *CREATE2* address derived from a canonical factory, a derivation-dependent salt, and an *initCode* template whose owner is the derived stealth public key. The sender transfers assets to this address before deployment. The recipient later derives the owner key from the announcement and deploys the account through the factory in their first ERC-4337 *UserOperation*. The factory, *initCode* hash, and implementation are public constants, while the salt and owner key vary per stealth derivation and do not reveal the reusable meta-address under the ERC-5564 derivation.

D. Zero-Knowledge Proofs and Shielded Pools

Zero-knowledge succinct non-interactive arguments of knowledge (zk-SNARKs) prove computational statements without revealing private inputs [11]. Shielded pools use commitments, Merkle membership proofs, and nullifiers to support private one-time spending [12]. A user deposits a cryptographic commitment into an incremental Merkle tree and later proves knowledge of a secret corresponding to a valid tree leaf without revealing its index. A *nullifier* is a deterministic public value derived from the secret. Recording

it on-chain allows the verifier to reject reuse without revealing the corresponding commitment.

III. PROBLEM FORMULATION AND THREAT MODEL

This section formalizes PrivGas’s privacy and security goals and defines the on-chain adversary model.

A. System Setting and Goals

We assume a recipient holds stealth addresses $\text{addr}_1, \dots, \text{addr}_n$, each created by a possibly-different sender via a distinct *Announcement* event a_i ($1 \leq i \leq n$). Each addr_i holds an asset but no ETH to pay for transaction gas. Our goal is to design a paymaster $\mathcal{P}$ such that, for any *UserOperation* originating from addr_i , $\mathcal{P}$ sponsors gas only when the operation is backed by an unspent protocol credit created through a legitimate, value-bearing announcement. PrivGas must satisfy three primary requirements:

- **(G1) Sybil resistance:** An adversary cannot manufacture a usable sponsorship credit for less than the non-refundable economic cost required by PrivGas.
- **(G2) Gas top-up unlinkability:** An adversary observing on-chain state and bundler-visible fields cannot link the sponsored spend to a known recipient-controlled funding wallet. The zero-knowledge (ZK) pool hides the redeemed commitment index from the paymaster, but does not hide the public stealth account address, Bootstrap participation, timing, or target-call behavior.
- **(G3) One-credit redemption:** A given deposited credit cannot sponsor more than one later operation.

To satisfy G1, PrivGas must enforce an economic bound over total sponsored exposure. Let F be a strictly non-refundable fee, B_{boot} and B_{spend} be the maximum sponsorship costs accepted by the Bootstrap and Credit Paymasters, and $\kappa > 1$ be the economic safety margin. PrivGas must ensure:

$$F > \kappa \cdot (B_{\text{boot}} + B_{\text{spend}}) \quad (1)$$

B. Adversary Model

The adversary can observe all on-chain state and bundler-visible *UserOperation* fields, submit arbitrary announcements, and operate custom bundlers. However, they cannot break underlying cryptographic assumptions, such as the discrete-log problem or zero-knowledge proof (ZKP) soundness. Since the blockchain cannot cryptographically verify honest stealth address derivation, PrivGas security relies strictly on the economic constraint defined in Eq. (1).

IV. PRIVGAS SYSTEM DESIGN

Figure 1 illustrates the three-stage PrivGas architecture supported by four contracts and a non-recoverable fee sink. In Stage 1 (Fund), the sender binds a non-refundable fee to the stealth-address announcement, establishing sponsorship eligibility. In Stage 2 (Bootstrap), PrivGas sponsors the recipient’s deposit of a commitment into the Shielded Credit Pool. In Stage 3 (Spend), the recipient redeems this credit using a Merkle-membership proof and domain-separated nullifier, allowing the Credit Paymaster to sponsor the target operation without revealing the redeemed commitment index.

Algorithm 1 Registry Admission and Funding Phase

Require: stealthAddr , ephemeralPubKey , msg.value

Ensure: Forwards value, routes the non-refundable fee, and grants sponsorship eligibility

```

1: {Constants:  $V_{\min}$  and  $F$ , with  $F$  chosen to satisfy Eq. (1)}
2: {Enforce minimum transfer and fee}
3: if  $\text{msg.value} < V_{\min} + F$  then
4:   return INSUFFICIENT_FUNDS
5: end if
6:  $\text{feeAmount} \leftarrow F$ 
7:  $\text{forwardAmount} \leftarrow \text{msg.value} - F$ 
8: {Authorize Bootstrap sponsorship}
9:  $\text{eligible}[\text{stealthAddr}] \leftarrow \text{true}$ 
10:  $\text{BootstrapPaymaster.eligible}[\text{stealthAddr}] \leftarrow \text{true}$ 
11: {Transfer value and emit announcement}
12: Transfer  $\text{feeAmount}$  to the non-recoverable fee sink
13: Transfer  $\text{forwardAmount}$  to  $\text{stealthAddr}$ 
14:  $\text{ERC5564.announce}(\text{stealthAddr}, \text{ephemeralPubKey})$ 
15: return SUCCESS

```

A. Announcement Registry

The Registry functions as a *payable* wrapper around the standard ERC-5564 *Announcer*. To establish eligibility for PrivGas sponsorship, a sender must invoke *announceAndFund(...)* and provide a native value covering both a minimum transfer amount ($V_{\min}$) and a non-refundable protocol fee (F).

This atomic execution, detailed in Algorithm 1, forwards the residual $\text{msg.value} - F$, which is at least $V_{\min}$, to the derived stealth address, routes F to a non-recoverable fee sink, broadcasts the standard announcement event, and records the stealth address as eligible, with eligibility mirrored into paymaster-associated storage for validation-scope compliance. The Registry enforces the minimum admissible fee at deployment and on updates. The strict separation of F from the forwarded value is the cornerstone of PrivGas’s Sybil resistance.

B. Bootstrap Paymaster and Shielded Credit Pool

Once an address is marked eligible, the Bootstrap Paymaster sponsors exactly one operation per eligible address: a deposit into the Shielded Credit Pool. Because the *EntryPoint* executes *callData* on the *sender*, the Bootstrap *UserOperation* invokes the stealth smart account’s *execute* function with the Credit Pool as target, zero ETH value, and *deposit(uint256)* as the inner call. The paymaster validates this call shape and enforces its sponsorship-cost and gas-price bounds. The recipient deposits a locally generated Semaphore identity commitment C into the pool’s incremental Merkle tree. The Bootstrap deposit is public, while the later membership proof hides which commitment is redeemed.

C. Credit Paymaster

The Credit Paymaster sponsors the final *UserOperation*, such as transferring assets from the stealth account. Sponsorship requires a Semaphore membership proof, a domain-separated nullifier N , and a message $D = \text{uint256}(\text{userOpHash})$. The *EntryPoint* encoding excludes the appended paymaster proof from the operation hash, avoiding a circular dependency. Since *userOpHash* commits to the operation fields, including *userOp.sender*, the proof is bound

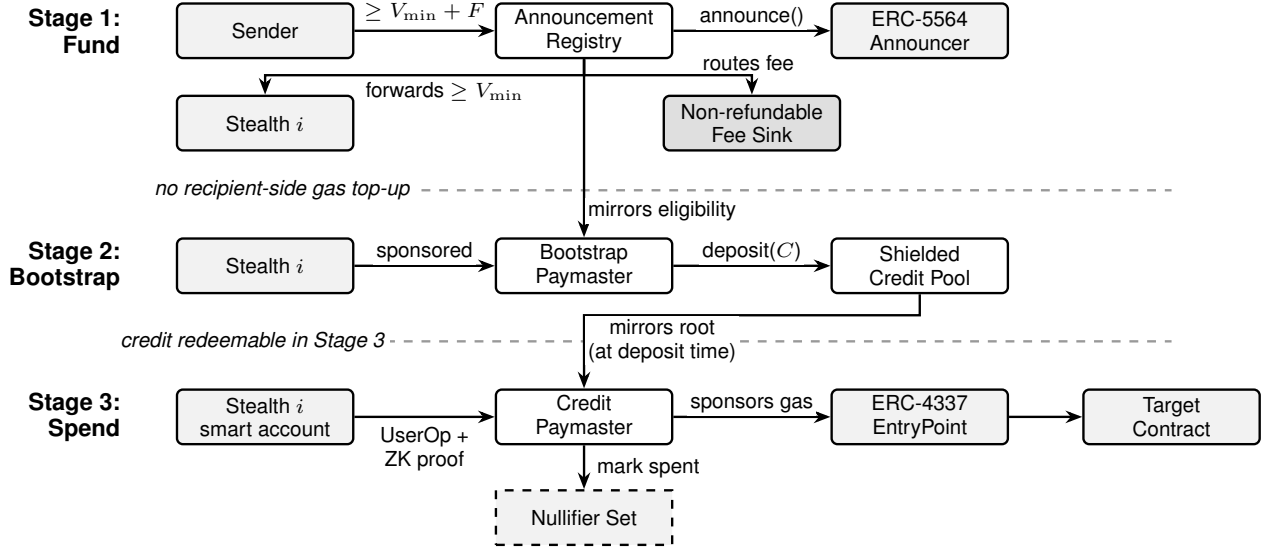


Fig. 1. PrivGas architecture and three-stage protocol flow.

Algorithm 2 Credit Paymaster Validation Phase

Require: $userOp$, $userOpHash$, $maxCost$
Ensure: Validation success or failure

- 1: Extract π , root R , nullifier N , message D , and scope S
- 2: **if** $maxCost > B_{spend} \vee gasUnits > G_{max} \vee maxFeePerGas > p_{max}$ **then**
- 3: **return** *REJECT*
- 4: **end if**
- 5: **if** $R \neq currentRoot \vee S \neq S_{credit} \vee D \neq uint256(userOpHash)$ **then**
- 6: **return** *REJECT*
- 7: **end if**
- 8: **if** $spentNullifiers[N]$ **then**
- 9: **return** *REJECT*
- 10: **end if**
- 11: $isValid \leftarrow Verify_{Semaphore}(R, N, D, S, \pi)$
- 12: **if** $\neg isValid$ **then**
- 13: **return** *REJECT*
- 14: **end if**
- 15: $spentNullifiers[N] \leftarrow true$
- 16: **return** *VALIDATION_SUCCESS*

to the sponsored operation. During validation, Algorithm 2 checks the current mirrored root, fixed scope, message, nullifier, and resource bounds before invoking the Semaphore verifier. Recording the nullifier enforces one-time redemption while hiding the deposit index. Validation state remains in paymaster-associated storage under the ERC-7562 model [10]. If execution fails, the credit remains consumed because the paymaster has committed to covering gas.

V. SECURITY AND PRIVACY ANALYSIS

This section evaluates how PrivGas’s economic and cryptographic constraints satisfy our primary security requirements by formalizing them into three core lemmas.

Lemma 1 (Sybil Resistance): *If every credit requires a non-refundable fee F and the Bootstrap and Spend sponsorship exposures are bounded by B_{boot} and B_{spend} , then an attacker’s self-dealing profit is negative when Eq. (1) holds.*

A recoverable-value gate is insufficient because an attacker could withdraw the forwarded ETH and retain a sponsorship

credit. Our design separates recoverable forwarded value from the unrecoverable fee F . Since Eq. (1) prices F above the combined Bootstrap and Spend exposure, credit farming remains economically irrational even for a self-dealing attacker who controls the sender, recipient, and target contract simultaneously, mirroring the worst-case colluding adversary considered in prior Sybil-resistant paymaster designs.

Lemma 2 (Gas Top-Up and Commitment-Index Privacy): *Assuming standard zero-knowledge soundness and commitment-hiding properties, PrivGas avoids known-wallet recipient gas top-ups and prevents the paymaster from learning the redeemed commitment index.*

The recipient never funds the stealth account from a known wallet before spending, removing the direct gas top-up heuristic. Separately, the Shielded Credit Pool proof reveals only that some valid, unspent commitment exists under the current mirrored root. The verifier learns the public *UserOperation*, the nullifier, and the proof message, but not the Merkle leaf being redeemed. This guarantee is limited to proof-level commitment-index privacy and known-wallet top-up avoidance. It does not hide Bootstrap participation, timing, network metadata, target behavior, credit denomination, or cold-start anonymity limits.

Lemma 3 (One-Credit Redemption): *Assuming deterministic nullifier derivation within the zero-knowledge circuit, a single commitment cannot be redeemed twice.*

PrivGas integrity ensures that N is derived deterministically from the deposited identity under a fixed credit-spending domain S_{credit} . Because the scope is a fixed domain separator rather than a per-operation value, the same credit always reveals the same nullifier. The proof is also bound to $D = uint256(userOpHash)$, preventing transplantation to a different sponsored operation. Any second proof using the same identity and scope generates the same nullifier and is rejected.

TABLE II
PRIVGAS PROTOTYPE GAS EVALUATION

Operation	Gas (max)	Measurement Type
<i>announceAndFund()</i>	175,951	Foundry gas report
Bootstrap validation	52,543	Gas report, direct call
<i>CreditPool.deposit()</i>	216,905	Foundry gas report
Credit validation	284,527	Gas report, direct call
Groth16 verification	225,733	Measured verifier call

TABLE III
SMART CONTRACT FUNCTIONAL CORRECTNESS MATRIX

Test Scenario (UserOperation Input)	Expected	Observed
Valid proof, unspent nullifier, correct sender	Accept	Accept
Valid proof, previously spent nullifier	Reject	Reject
Proof bound to a different sender	Reject	Reject
Proof bound to a different operation	Reject	Reject
Invalid current Merkle root	Reject	Reject
$maxCost > B_{spend}$	Reject	Reject

Spend-phase tests via *EntryPoint.handleOps* with Groth16 proofs.

VI. EVALUATION AND DISCUSSION

All experiments ran in local Foundry simulation (Solidity v0.8.28, via-IR optimizer), not on a testnet or mainnet. We used an *EntryPoint* implementation with v0.9 semantics, a Semaphore v4 verifier [13], and Groth16 proof calldata. The *CreditPaymaster* validates a Spend request at a peak cost of 284,527 gas, while isolated Groth16 verification requires 225,733 gas (Table II), below the 500,000-gas ERC-4337 verification limit. Integration tests execute both Bootstrap and Spend through *EntryPoint.handleOps*. The Foundry suite contains 65 passing tests, including the six Spend-phase cases in Table III and the Bootstrap-sponsored deposit path. Both paymasters cap sponsorship at 0.005 ETH and gas price at 10 gwei. With $\kappa = 2$, $F = 0.021$ ETH satisfies $F > 2(0.005 + 0.005) = 0.020$ ETH. This deters credit farming while remaining viable for high-value transfers.

ERC-7562 compatibility was preliminarily verified with the local *simulateValidation* remote procedure call (RPC) harness [10]. While a local simulation is not equivalent to a live bundler opcode trace, it provides an initial check against disallowed validation-state access. Mirroring eligibility into paymaster storage avoids registry reads during validation and keeps mirrored-root reads and nullifier writes inside associated storage. Full bundler-level opcode tracing remains future work.

PrivGas eliminates known-wallet gas top-ups, mitigating the primary deanonymization vector observed in live ERC-5564 deployments such as Umbra [5]. The paymaster absorbs the ZK-verification overhead. Proof-level index privacy does not hide timing or cold-start anonymity limitations, and the current prototype accepts only the latest mirrored Merkle root. Randomized delays between Bootstrap and Spend and a pre-populated pool may mitigate these limitations.

VII. CONCLUSION AND FUTURE WORK

This work characterized ERC-5564 stealth-address gas funding as a Sybil-resistant admission-control problem re-

quiring the prevention of recipient-side gas top-up linkage and proof-level commitment-index privacy. We proposed PrivGas, which separates value-bearing announcement, shielded credit creation, and later proof-based redemption. By pricing the non-refundable fee above total sponsored gas exposure and using domain-separated nullifiers, PrivGas prevents credit farming and double spending under the stated threat model. In local Foundry simulation (not testnet or mainnet), the PrivGas prototype validates a full Semaphore Groth16 proof at 284,527 gas in ERC-4337 paymaster validation, 43.1% below the 500,000-gas ERC-4337 verification limit. Together, these results demonstrate that proof-based sponsorship can be integrated with stealth smart accounts without requiring recipient-side gas top-ups. Future work will evaluate PrivGas on a public testnet, conduct bundler-level ERC-7562 opcode tracing, and quantify timing-correlation and cold-start anonymity-set risks.

REFERENCES

- [1] N. van Saberhagen, “CryptoNote v 2.0,” Oct. 2013, doi: 10.5281/zenodo.6496896. Accessed: Oct. 6, 2026. [Online]. Available: <https://zenodo.org/records/6496896>
- [2] A. Wahrstätter, M. Solomon, B. DiFrancesco, V. Buterin, and D. Svetinovic, “BaseSAP: Modular stealth address protocol for programmable blockchains,” *IEEE Trans. Inf. Forensics Security*, vol. 19, pp. 3539–3553, Feb. 2024, doi: 10.1109/TIFS.2024.3364081.
- [3] T. Wahrstätter, M. Solomon, B. DiFrancesco, and V. Buterin, “ERC-5564: Stealth addresses,” *Ethereum Improvement Proposals*, no. 5564, Aug. 2022. [Online]. Available: <https://eips.ethereum.org/EIPS/eip-5564>
- [4] V. Buterin, “An incomplete guide to stealth addresses,” Jan. 20, 2023. Accessed: Oct. 6, 2026. [Online]. Available: <https://vitalik.eth.limo/general/2023/01/20/stealth.html>
- [5] A. M. Kovács and I. A. Seres, “Anonymity analysis of the Umbra stealth address scheme on Ethereum,” in *Companion Proc. ACM Web Conf. 2024 (WWW '24 Companion)*, May 2024, pp. 1768–1775, doi: 10.1145/3589335.3651963.
- [6] V. Buterin, Y. Weiss, D. Tirosch, S. Nacson, A. Forshtat, K. Gazso, and T. Hess, “ERC-4337: Account abstraction using alt mempool,” *Ethereum Improvement Proposals*, no. 4337, Sep. 2021. [Online]. Available: <https://eips.ethereum.org/EIPS/eip-4337>
- [7] Q. Wang and S. Chen, “Account abstraction, analysed,” in *Proc. IEEE Int. Conf. Blockchain (Blockchain)*, Dec. 2023, pp. 323–331, doi: 10.1109/Blockchain60715.2023.00057.
- [8] H. Jiao and N. Udomlertsakul, “SuperPaymaster: Eliminating centralized signer authority via asset-oriented abstraction to reconcile usability and decentralization in account abstraction,” May 2026, *arXiv:2605.05774*, doi: 10.48550/arXiv.2605.05774.
- [9] H. Guo, Y. Feng, C. Wu, Z. Li, and J. Xu, “StealthHub: UTXO-based stealth address protocol,” in *Proc. IEEE Int. Conf. Web Services (ICWS)*, Jul. 2025, pp. 996–1003, doi: 10.1109/ICWS67624.2025.00131.
- [10] Y. Weiss, D. Tirosch, A. Forshtat, and S. Nacson, “ERC-7562: Account abstraction validation scope rules,” *Ethereum Improvement Proposals*, no. 7562, Sep. 2023. [Online]. Available: <https://eips.ethereum.org/EIPS/eip-7562>
- [11] N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer, “From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again,” in *Proc. 3rd Innov. Theor. Comput. Sci. Conf. (ITCS)*, Jan. 2012, pp. 326–349, doi: 10.1145/2090236.2090263.
- [12] A. Pertsev, R. Semenov, and R. Storm, “Tornado Cash privacy solution version 1.4,” Dec. 2019. Accessed: Oct. 6, 2026. [Online]. Available: <https://berkeley-defi.github.io/assets/material/Tornado%20Cash%20Whitepaper.pdf>
- [13] K. Gurkan, W. J. Koh, and B. Whitehat, “Community proposal: Semaphore: Zero-knowledge signaling on Ethereum,” Feb. 2020. Accessed: Oct. 6, 2026. [Online]. Available: <https://semaphore.pse.dev/whitepaper-v1.pdf>