Anonymity Analysis of the Umbra Stealth Address Scheme on Ethereum

ALEX M. KOVÁCS* and ISTVÁN ANDRÁS SERES, Eötvös Loránd University, Hungary

Stealth addresses are a privacy-enhancing technology that provides recipient anonymity on blockchains. In this work, we investigate the recipient anonymity and unlinkability guarantees of Umbra, the most widely used implementation of the stealth address scheme on Ethereum and on its three off-chain scalability solutions, e.g., Arbitrum, Optimism, and Polygon. We define and evaluate four heuristics to uncover the real recipients of stealth payments. We find that for the majority of Umbra payments, it is straightforward to establish the recipient, hence nullifying the benefits of using Umbra. Specifically, we find the real recipient of 48.5%, 25.8%, 65.7%, and 52.6% of all Umbra transactions on the Ethereum main net, Polygon, Arbitrum, and Optimism networks, respectively. Finally, we suggest easily implementable countermeasures to evade our deanonymization and linking attacks.

CCS Concepts: • **Security and Privacy** → Privacy-preserving protocols.

Additional Key Words and Phrases: Ethereum, decentralized finance, recipient anonymity, unlinkability, stealth address scheme.

1 INTRODUCTION

Decentralized finance (DeFi) offers a radically new financial system, i.e., it is an open, trustless, composable, and transparent financial system built on top of blockchains. The transparent nature of DeFi aids trustlessness and public verifiability. However, the lack of financial privacy is undesirable in most applications. Therefore, the DeFi ecosystem on Ethereum offers several deployed privacy-enhancing solutions, such as mixers (e.g., Tornado Cash [Pertsev et al. 2019]), confidential transactions (e.g., AZTEC [Williamson 2018]) or stealth addresses (e.g., Umbra [DiFrancesco and Solomon 2021]. Previous work has thoroughly analyzed the anonymity guarantees provided by Tornado Cash [Béres et al. 2021; Wang et al. 2023; Wu et al. 2022]. In this work, we study the recipient anonymity and unlinkability guarantees provided by the Umbra stealth address scheme.

Stealth addresses (or seldom Elliptic-curve Diffie–Hellman (ECDH) addresses) are a prevalent privacy-enhancing technology for cryptocurrencies. They were first proposed by Peter Todd on the Bitcoin mailing list in 2014 [Todd 2014]. Stealth addresses allow a payment sender to transfer assets *non-interactively* to a recipient in a recipient anonymous way. In a nutshell, senders generate a computationally random-looking address, i.e., a stealth address. Subsequently, the sender can pay the recipient by transferring funds to the freshly generated stealth address. Specifically, no party other than the transacting parties can establish the actual recipient of a transaction using stealth addresses. However, the recipient can detect on the blockchain that a specific stealth address belongs to them, and crucially they are the only ones who can spend the funds at the stealth address. In conclusion, stealth addresses allow the sender to obfuscate its economic relationship with the recipient.

Stealth addresses are standardized for Bitcoin in the Bitcoin Improvement Proposal (BIP) 47 [Ranvier 2015] and Ethereum in the Ethereum Improvement Proposal (EIP) 5564 [Wahrstätter et al. 2022]. Even though the BIP47 did not gain significant traction in the Bitcoin ecosystem [Möser and Böhme 2017], stealth addresses remain a popular privacy-enhancing technique for public blockchains as they are implemented and deployed in Monero [Noether and Noether 2014] and Ethereum [Wood et al. 2014] as well.

This paper aims to analyze the recipient anonymity guarantees achieved by the most popular stealth address implementation on Ethereum: Umbra[1]. Umbra is currently deployed on Ethereum and its numerous layer-2 scalability solutions such as Polygon, Arbitrum, and Optimism. Umbra is the second most popular privacy-enhancing technology used on Ethereum and its layer-2 systems after the AZTEC protocol with 413 daily users versus 1866 daily users on March 20th, 2023[2,3]. In this work, our goal is to identify erroneous idioms of use that allow any blockchain analysts to reduce the theoretical recipient anonymity and unlinkability guarantees provided by the stealth address cryptographic scheme.

In this work, we make the following contributions.

- We identify four heuristics enabled by user behavior that can reduce or nullify the recipient anonymity or unlinkability guarantees of the Umbra stealth address scheme, see Section 6.
- We evalaute the efficacy of the identified heuristics in uncovering and linking Umbra stealth payment recipients, see Section 7. Our results are reproducible as our code is open-source. It is available at the following repository: https://github.com/alekszkovacs/UmbraAnonymityAnalysis.
- We suggest counter-measures to our proposed heuristics. We make suggestions to stealth wallet providers and developers.

The rest of this paper is organized as follows. In Section 2, we describe the cryptographic details of stealth addresses and, in particular, the relevant implementation details of the Umbra stealth address scheme. In Section 3, we introduce our system and threat models. In Section 4, we detail our data collection methods. We present empirical Umbra usage statistics in Section 5. In Section 6, we define several heuristics aimed at deanonymizing Umbra recipients. We evaluate the efficacy of these heuristics in Section 7. Finally, we conclude our paper in Section 8.

*
_____

Authors' address: Alex M. Kovács, kovcsaleex0104@gmail.com; István András Seres, seresistvanandras@gmail.com, Eötvös Loránd University, Pázmány Péter stny. 1/C, Budapest, Pest, Hungary, 1117.

_____
[1]See: https://app.umbra.cash/.
[2]See: https://dune.com/intake/umbra-protocol
[3]See: https://dune.com/gm365/aztec-v2.

2 BACKGROUND

Hereby, we describe the stealth address protocol used by Umbra.

2.1 Notations

When we uniformly at random sample r from a set S, we write $r \in_R S$. We assume the existence of a prime-order cyclic group $\mathbb{G}$ ($|\mathbb{G}| = p$), where the discrete logarithm and the (decisional and computational) Diffie-Hellman assumptions hold. The finite group $\mathbb{G}$ is generated by G, i.e., $\mathbb{G} = \langle G \rangle$. In our applications, $\mathbb{G}$ is an elliptic curve group over a finite field $\mathbb{F}_q$ in which we denote the group operation additively. Secret keys are typically sampled from $\mathbb{F}_q^*$, while public keys are elliptic curve points in $\mathbb{G}$. A secure cryptographic hash function is denoted as $H(\cdot)$. Ethereum addresses are obtained by hashing the corresponding public key. Ethereum addresses are typed in bold; sender, recipient, registrant, and stealth addresses are denoted as s, r, reg, st, respectively.

2.2 Stealth Addresses

Umbra applies the dual-key stealth address protocol; see Figure 1. In the applied stealth address scheme, the recipients use two key pairs: a spending and a viewing public/private key pair. The benefit of this approach is that recipients can separate the concerns of detecting incoming stealth payments and spending them. For instance, this allows recipients to accept stealth payments while keeping their spending private key off-chain, e.g., in cold storage. Additionally, this architecture enables recipients to outsource the detection of stealth payments to a trusted party since the detection of payments only relies on the secret viewing key and the spending public key. However, the trusted third party cannot spend the funds, as one also needs the spending private key to redeem the funds at the stealth address. On the other hand, this scanning approach does not provide privacy against the trusted third party.

Recently, several novel cryptographic schemes were proposed to improve the linear scanning problem of stealth addresses [Beck et al. 2021; Liu and Tromer 2022; Madathil et al. 2021]. Nonetheless, this problem is still not entirely settled, as current solutions are unsatisfactory. Specifically, Fuzzy Message Detection [Beck et al. 2021] does not provide sufficient levels of privacy [Seres et al. 2022]. Private Signaling assumes non-colluding scanning servers or trusted execution environments [Jakkamsetti et al. 2023; Madathil et al. 2021]. Both of them are strong assumptions. Oblivious Message Retrieval (OMR) and its latest variant apply fully homomorphic encryption in an elegant way [Liu and Tromer 2022; Liu et al. 2023]. Therefore, OMR uses large ($\approx$ 1GB) detection keys that are impractical for resource-constrained devices. At the time of writing, OMR is being developed and soon to be deployed by the Zcash Foundation.

3 UMBRA: SYSTEM AND THREAT MODELS

3.1 Umbra: a system model

Umbra, Ethereum's most popular stealth address scheme, consists of the following five system components.

(1) **Senders**: look up viewing and spending keys $(\mathrm{pk}_r^{view}, \mathrm{pk}_r^{spend})$ of their recipients r in a registry and generate stealth addresses for their recipients, see Figure 1. Subsequently, senders

---

The dual-key stealth address generation algorithm

Recipient: generates $s, v \in_R \mathbb{F}_q$. Publishes $(\mathrm{pk}_r^{view}, \mathrm{pk}_r^{spend}) = (vG, sG)$.

Sender: publishes $\mathrm{pk}_s \in \mathbb{G}$ on the blockchain.

Input: $(\mathrm{pk}_s, (\mathrm{pk}_r^{view}, \mathrm{pk}_r^{spend}))$.

(1) The sender $\mathcal{S}$ generates an ephemeral key pair $(r, R) = (r, rG)$, where $r \in_R \mathbb{F}_p$.

(2) $\mathcal{S}$ computes shared secret $c = H(r \cdot \mathrm{pk}_r^{view})$.

(3) $\mathcal{S}$ sends funds to the stealth address st derived from the public key $\mathrm{pk}^{stealth} := cG + \mathrm{pk}_r^{spend}$.

Return: $(R, \mathrm{pk}^{stealth})$.

A naïve (linear) stealth payment detection algorithm

Input: $(\mathrm{pk}_r^{view}, \mathrm{pk}_r^{spend}), \{(R_i, \mathrm{pk}_i^{stealth})\}_{i=1}^n$.

(1) Let incomingPayments := {}

(2) Recipient checks $H(vR_i)G + \mathrm{pk}_r^{spend} \stackrel{?}{=} \mathrm{pk}_i^{stealth}$ for each $i \in [1, n]$.
   - If it holds incomingPayments.*append*$(\mathrm{pk}_i^{stealth})$.

Return: incomingPayments.

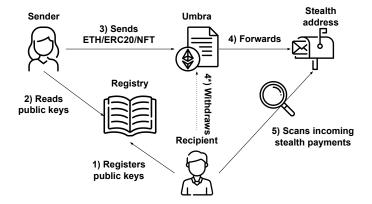Fig. 1. The dual-key stealth address protocol.

Fig. 2. A schematic depiction of the Umbra stealth address scheme. A recipient first registers its viewing and spending public keys in the Stealth Key Registry. Later, the sender can read these public keys from the registry and send assets to a freshly generated pseudorandom stealth address. Finally, the recipient scans the blockchain for any incoming stealth payments and subsequently withdraws the received assets to an address they own.

transfer crypto assets, e.g., ETH, or ERC20 tokens, to the stealth address for their recipients.

(2) **Recipients**: write their viewing and spending keys to a registry contract. Later, occasionally, they scan the blockchain for incoming stealth payments. Currently, they need to check every stealth address to see whether it belongs to them. Alternatively, recipients can outsource this computation.

(3) **Registry of stealth public keys**: it is a smart contract that stores all the recipients' viewing and spending keys.

(4) **Umbra smart contract**: facilitates the asset transfer between sender and recipient. Additionally, for every stealth

payment, it emits a so-called *Announcement* event containing $(R, \text{pk}^{stealth})$, cf. Figure 1. Later these events can be scanned by recipients to detect incoming stealth payments.

(5) **Relayers**: in the special cases of ERC20 tokens, relayers are paid to withdraw assets from the stealth addresses. This is because those stealth addresses do not hold ether, hence cannot send transactions as they cannot pay for the occurring withdrawal transaction fees.

The Umbra stealth address scheme entails the following steps.

(1) **Recipient registers public keys**: recipients register their stealth public viewing and spending keys from an Ethereum address reg in the registry smart contract.

(2) **Sender reads registry**: senders read viewing and spending public keys $(\text{pk}_r^{view}, \text{pk}_r^{spend})$ of their recipients in the stealth public key registry contract.

(3) **Sender transfers assets**: senders generate a pseudorandom stealth address st given the viewing and spending public keys $(\text{pk}_r^{view}, \text{pk}_r^{spend})$ and subsequently transfer assets to the derived stealth address st.

(4) **Recipient scans the blockchain**: recipients intermittently scan the *Announcement* events emitted by the Umbra contract for incoming stealth payments. Typically, users do not leave assets residing at stealth address st, rather, they withdraw them to a recipient address r.

(5) **Recipient withdraws assets**: either the Umbra smart contract forwards assets to the stealth address or stores assets in the case of ERC20 tokens. In the latter case, recipients use relayers (not depicted in Figure 2) to withdraw funds held by the Umbra smart contract. Finally, recipients withdraw funds from the stealth address st to the recipient address r.

3.2 Threat model

The adversary aims to link the Umbra stealth payment recipients r to entities previously registered from some address reg in the Registry contract. We assume the adversary can access the Ethereum blockchain, recording all transactions and smart contracts. Specifically, the adversary can read the code of the Umbra and Registry smart contracts and all transactions that call these smart contracts. However, we do not assume that the adversary can deanonymize Umbra users off-chain, e.g., by running a relayer node. A potent adversary on the peer-to-peer layer could certainly link IP addresses or other identifying information to Umbra users whenever they withdraw their assets from their stealth addresses. Nonetheless, our adversary solely relies on the public blockchain data to decrease the recipient anonymity and unlinkability guarantees of Umbra.

3.3 Anonymity notions

Stealth address schemes provide two types of privacy notions: recipient anonymity and recipient unlinkability. Informally, recipient anonymity guarantees that an observer cannot uncover the identity of a recipient better than randomly guessing. A weaker notion is recipient unlinkability, which dictates that it must be indistinguishable whether or not two stealth payments are sent to the same user. For a formal, game-based definition of these privacy notions, we refer the reader to [Backes et al. 2013]. Hereby, due to space constraints,

---

> **The recipient unlinkability $\mathcal{G}_{\mathcal{A},\Pi}^{RU}(\lambda)$ game**
> (1) Adversary $\mathcal{A}$ selects target recipients r_0, r_1 and a target sender s.
> (2) Challenger C instructs sender s to send a stealth payment to r_c for $c \xleftarrow{\$} \{0, 1\}$.
> (3) C generates randomly a challenge bit $b \xleftarrow{\$} \{0, 1\}$. If $b = 0$, C instructs s to send a stealth payment to r_c. Otherwise, instructs s to send a stealth payment to r_{1-c}.
> (4) $\mathcal{A}$ observes the blockchain and outputs b'.
> **Return**: 1, iff. $b = b'$, otherwise 0.

Fig. 3. The security game for the anonymity notion of recipient unlinkability.

we only provide a definition for recipient unlinkability; recipient anonymity can be defined analogously.

Definition 3.1 (Recipient unlinkability (RU)). An anonymous communication protocol Π satisfies recipient unlinkability if for all PPT adversaries $\mathcal{A}$ there is a negligible function $\text{negl}(\cdot)$ such that

$$\Pr[\mathcal{G}_{\mathcal{A},\Pi}^{RU}(\lambda) = 1] \leq \frac{1}{2} + \text{negl}(\lambda), \tag{1}$$

where the privacy game $\mathcal{G}_{\mathcal{A},\Pi}^{RU}(\lambda)$ is defined in Figure 3.

4 DATA COLLECTION

We were primarily interested in three main types of transactions in relation to the Umbra stealth address scheme on a given network.

- **Stealth key registrations.** We collected every transaction that registered public keys in the stealth key registry.
- **Umbra sends and withdraws.** Every incoming (sent stealth payments) and outgoing (withdrawn stealth payments) transactions of the Umbra smart contract had been collected.
- **Registrant, sender, and withdrawer transactions.** For every address that interacted with either the Stealth Key Registry or the Umbra smart contract, we collected every transaction that was sent or received by these addresses.

We obtained these transactions using the APIs of the Etherscan blockchain explorers for the investigated four blockchains, i.e., Ethereum L1 [4], Arbitrum [5], Polygon [6], and Optimism [7]. Additionally, the measurements of Section 5 were obtained from the https://dune.com blockchain analytics website.

5 UMBRA ACTIVITY

At the time of writing, Umbra facilitated the transfer of more than 155 million U.S. dollars worth of cryptoassets (ether and ERC20s) across all chains where it is deployed[8]. Interestingly, Umbra became more popular on Layer 2s, such as Arbitrum, Optimism, or Polygon, than it is on mainnet Ethereum, see Figure 4. We observe a sharp increase in Umbra usage after the previously most popular on-chain privacy-enhancing technology, i.e., Tornado Cash, was banned in

---

[4]See: https://etherscan.io/
[5]See: https://arbiscan.io/
[6]See: https://polygonscan.com/
[7]See: https://optimistic.etherscan.io/
[8]See: https://dune.com/intake/umbra-protocol.

the U.S. As of July 2023, there are nearly 70, 000 registered users in the stealth key registries of Umbra across all chains.
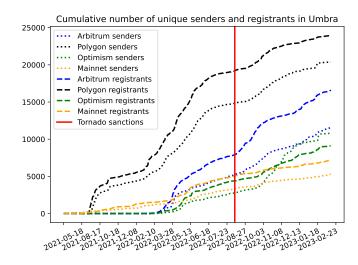

Fig. 4. Increasing popularity of the Umbra Stealth Address scheme on Ethereum and various Layer-2 systems. Notably, Layer-2 deployments of Umbra surpass the mainnet deployment in demand. The red vertical line shows the date when Tornado Cash contracts were sanctioned by OFAC.

We analyzed the usage patterns of Umbra users. First, we had suspected that certain users might use Umbra as a recipient anonymous payment processor. This could have been justified if, for instance, there are Umbra payments occurring in large numbers on a specific day of each month. Interestingly, we could not find such behavior on any of the deployed chains. See two typical activity heatmaps of Umbra users in Figure 5. We found that numerous Umbra sender addresses are very active (i.e., send dozens of Umbra transactions) in a short time period (e.g., weeks or a few days) and afterward they do not interact with Umbra anymore.
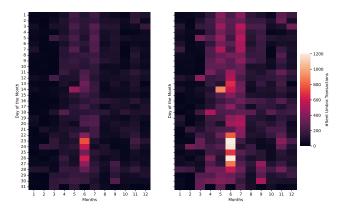

Fig. 5. Umbra activity heatmaps for two Umbra users on the Polygon network. Observe that most of their activity is concentrated in a short time period. The user on the left sent 650 Umbra transactions, while the user on the right sent 34, 081 Umbra transactions.
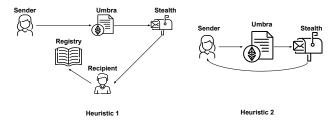
Fig. 6. A schematic depiction of Heuristic 6.1 (Registrant address reuse), and Heuristic 6.2 (Same sender and receiver).

6 UNCOVERING UMBRA PAYMENT RECIPIENTS

Here, we define our heuristics to identify Umbra payment recipients. We remark that our ultimate goal is to link stealth addresses to their actual recipients, i.e., addresses registered to the Umbra stealth key registry. Additionally, we provide heuristics that only reduce the anonymity guarantees of Umbra but do not uncover the actual recipient of an Umbra stealth payment deterministically.

6.1 Registrant address reuse

Users typically withdraw their funds (Ether or ERC20 tokens) from stealth address st to recipient address r. If the recipient address r is an address already interacted with the stealth key registry, then we can heuristically link the stealth address st to the registrant address reg. In some cases, we can also determine the ENS (Ethereum Name Registry) address(es) owned by the recipient since, in the early days of Umbra, only ENS users could register as recipients.

HEURISTIC 6.1. *If a recipient address* r *is also a registrant address* reg*, we link the corresponding stealth address* st *to* reg.

To refine this heuristic, we only consider recipient addresses r where the whole amount was transferred from the stealth address st. If this is not the case, i.e., multiple transactions originate from the stealth address potentially with different recipients, then we cannot confidently tell which recipient address (if at all) is the same entity as the owner of the stealth address. Note, we only need to check multiple outgoing withdraw transactions in the case of ether stealth transactions because, for ERC20 tokens, Umbra dictates the withdrawal of the entire stealth payment amount.

6.2 Same sender and receiver

When the sender s of a stealth payment is the same as the recipient r to where funds are withdrawn, then we heuristically link the corresponding stealth address st as being the same entity as the s (and hence the r) address. We speculate that these stealth payments are primarily issued for testing purposes.

HEURISTIC 6.2. *If the sender* s *of a stealth payment and the recipient address* r *of a withdraw transaction are the same, then we link the stealth address* st *and* s *as being the same entity.*

Similarly, in this heuristic, we check whether the whole amount was withdrawn from the stealth address st to r. If yes, then we apply the heuristic and conclude that s, st, and r addresses are owned by the same entity. Otherwise, we do not apply the heuristic. If there are
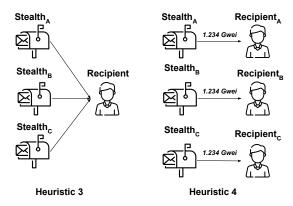
Fig. 7. A schematic depiction of Heuristic 6.3 (Collector pattern), and Heuristic 6.4 (Unique `maxPriorityFeePerGas`). Both of these heuristics forfeit the unlinkability guarantee provided by Umbra, as these heuristics enable one to link the recipients of multiple stealth payments.

multiple withdraw transactions from a stealth address with possibly multiple recipient addresses, then we cannot confidently link the actual owner of the stealth address st to the sender address s even if one of the recipient addresses is the same as the s.

6.3 Collector pattern

A stealth address scheme should provide recipient anonymity and *recipient unlinkability*. Recipient unlinkability dictates that the adversary should not be able to tell whether or not two messages (or payments) are sent to the same user. An anonymous communication system satisfies recipient unlinkability if the adversary cannot do better than randomly guessing whether two messages are sent to the same user [Backes et al. 2013]. The following heuristic aims to break the recipient unlinkability guarantees of Umbra.

Umbra users can forfeit the recipient unlinkability guarantees the stealth address scheme provides. If a set of stealth addresses $\{st_i\}_{i=0}^n$ have sent their funds directly to the same recipient address r, then we can cluster the stealth addresses $\{st_i\}_{i=0}^n$ as they are likely owned by the same entity. This implies that the recipients of these stealth payments are likely to be the same entity. Note that this heuristic is not able to link a recipient r or stealth address st to a registrant address reg in the Umbra stealth key registry. Nonetheless, it already uncovers important information about a set of stealth addresses.

HEURISTIC 6.3. *If a user withdraws funds from a set of stealth addresses $\{st_i\}_{i=0}^n$ to the very same recipient address r, then we cluster all these addresses ($\{st_i\}_{i=0}^n$ and r) together as they are likely controlled by the same entity.*

We acknowledge the possibility of false positives output by Heuristic 6.3. For instance, it might be that the recipient address r is a business that provides services in exchange for ether or ERC20 tokens, and Umbra users with stealth addresses are paying for them. We refine this heuristic as follows in order to avoid false positives. First, we only consider stealth addresses from where there is only a single withdrawal transaction, i.e., the whole amount is withdrawn to a recipient address in one transaction. If there were multiple transactions from a stealth address, then it is more likely that they are

rather payment transactions and not withdrawal transactions. We remark that one only needs to check this condition in the case of ether transactions because, for ERC20 tokens, users must withdraw all the amount from the Umbra contract in one transaction.

6.4 Unique max priority fee

In contrast to the previous Heuristic 6.3 (collector pattern), it is possible to link stealth addresses even if a user withdraws from a stealth address st_i to a different recipient address r_i for $i \in [1, n]$.

Ethereum uses a novel transaction fee mechanism called EIP-1559 [Buterin et al. 2019] that was deployed on August 5, 2021, as part of the London hard fork update. For our discussion, the only relevant part of the EIP-1559 transaction fee mechanism design is the `maxPriorityFeePerGas` field of Ethereum transactions. This is the only part of an Ethereum transaction set by the user (wallet) concerning transaction fees. The larger the transaction fee `maxPriorityFeePerGas` is, the sooner a validator includes the transaction in a new block. Typically, users (rather their wallet softwares) set `maxPriorityFeePerGas` automatically. However, users can modify the `maxPriorityFeePerGas` field manually and leave a chosen setting for an extended period of time. Hence, if a wallet owns multiple addresses and users apply "unique" `maxPriorityFeePerGas` fees, then the sender addresses of these transactions can be clustered together as they are likely owned by the same entity.

HEURISTIC 6.4. *If two or more withdraw transactions from $st_i \rightarrow r_i$ uses the same "unique" `maxPriorityFeePerGas`, then we link all these stealth addresses $\{st_i\}_{i=1}^n$ together.*

To make this Heuristic sensible, we need to choose when we consider a `maxPriorityFeePerGas` value "unique" in our dataset. Obviously, if a value is encountered dozens of times, then we cannot consider it unique, e.g., this is the case with round number as `maxPriorityFeePerGas` values. To limit the number of false positives, we considered a `maxPriorityFeePerGas` values "unique" if not more than five transactions apply that `maxPriorityFeePerGas` value. Note that this Heuristic does not apply to Umbra stealth payments that transfer ERC-20 tokens. Tokens are withdrawn by relayers on behalf of Umbra recipients. Therefore, token withdrawal transactions' `maxPriorityFeePerGas` field is not set by Umbra recipients but rather by the relayers.

7 EVALUATION

In this section, we evaluate the effectiveness of our heuristics in identifying Umbra payment recipients.

7.1 Heuristic 6.1

The first heuristic is the most successful in identifying the true recipients of Umbra payments; 48, 25% of Umbra payments (both ether and ERC20 token transfers) on the main Ethereum network can be trivially linked to stealth key registry transactions as the withdrawer address, and the registrant addresses are the same. We observe similarly high percentages for the other networks, i.e., 25, 79%, 65, 61%, 52, 57% for Polygon, Arbitrum, and Optimism, respectively.

7.2 Heuristic 6.2

This heuristic (same sender and withdrawer address) performs worse than the first heuristic. However, it still identifies a handful of Umbra withdraw transactions, i.e., 2, 61%, 1, 15%, 1, 87%, and 0, 85% of Umbra payments are deanonymized on the Ethereum L1, Polygon, Arbitrum, and Optimism networks, respectively, thanks to this heuristic.

7.3 Heuristic 6.3

The collector pattern heuristic breaks the recipient unlinkability guarantee of Umbra payments. As Figure 8 shows, several Umbra users withdraw multiple incoming payments to a *single* withdraw address. The largest whale is a user on the Arbitrum network who collected 481 stealth payments to a unique withdrawer address.

	Mainnet	**Polygon**	**Arbitrum**	**Optimism**
Heuristic 6.3	8.48 bits	9.86 bits	8.86 bits	8.83 bits
Entropy	12.53 bits	14.86 bits	13.36 bits	13.29 bits

Table 1. Most users withdraw multiple incoming stealth payments to a single address. This user behavior, formulated as "collector pattern" in Heuristic 6.3, greatly reduces the Shannon entropy of stealth payment recipients.

If n stealth payments are occurring on a network, an adversary ideally has $\log_2(n)$ bits of uncertainty about the recipients of stealth payments. However, with the help of the "collector pattern" heuristic, we can heuristically conclude that several stealth payments are likely received by the same user, hence, decreasing the entropy of this distribution, see Table 1. For instance, in the case of Arbitrum, naïvely, there is a 13, 36 bits of entropy in the recipients' distribution. The application of Heuristic 6.3 reduces this to 8.86 bits.
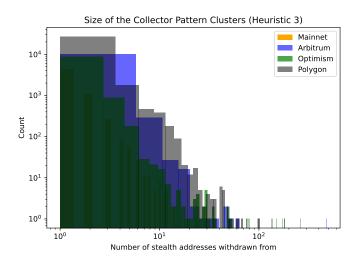

Fig. 8. This figure displays the number of withdrawer addresses with a given number of withdraw transactions from stealth addresses. Users tend to reuse withdraw addresses to collect stealth payments from multiple stealth addresses. Note that this is a log-log figure.

7.4 Heuristic 6.4

Previous work had already successfully applied the heuristic of unique transaction fees in the context of linking Tornado cash deposit and withdraw transactions [Béres et al. 2021]. Interestingly, in our case, we did not find Umbra withdraw transactions with the same unique priority fees. We attribute this phenomena as one of the beneficial outcomes of the EIP-1559 transaction fee mechanism.

	Mainnet	**Polygon**	**Arbitrum**	**Optimism**
Heuristic 6.1.	4671	15075	12488	8391
Heuristic 6.2.	253	670	356	135
Total linked	4696	15084	12513	8403
Total withdrawn	9680	58454	19033	15963

Table 2. Performance of the identified heuristics for uncovering Umbra payment recipients. Three of our heuristics could link the 48.5%, 25.8%, 65.7%, and 52.6% of all Umbra transactions on the Ethereum L1, Polygon, Arbitrum, and Optimism networks, respectively.

7.5 Countermeasures

A common root cause of Heuristic 6.1, 6.2, and 6.3 is the ability to reuse user addresses. Generally speaking, one should always avoid the reuse of addresses even in an account-based cryptocurrency as it facilitates user profiling and address clustering [Möser and Narayanan 2022]. The use of unique addresses is especially important when one uses privacy-enhancing technologies [Béres et al. 2021; Wang et al. 2023; Wu et al. 2022]. One must use different addresses for different types of on-chain activities. Developers of stealth address wallets should never allow users to reuse addresses or to withdraw funds from a stealth address to a registrant address.

8 CONCLUSION AND FUTURE DIRECTIONS

In this work, we analyzed the recipient anonymity and unlinkability guarantees provided by the second most used privacy-enhancing overlay in the Ethereum DeFi ecosystem. We identified four heuristics that enabled us to link the majority of Umbra recipients on the Ethereum mainnet. Finally, we suggested several countermeasures that would prevent one from our identified heuristics. Future work should investigate the interplay between Ethereum's cross-chain ecosystem and the deployed privacy-enhancing technologies on these chains. For instance, Ethereum and its L2s use the same address format. We also observed that users tend to reuse their addresses across different networks. We suspect that this address reuse could lead to cross-chain heuristics that could further decrease the recipient anonymity guarantees of Umbra. Another venue for future work could be to apply the heuristics of [Victor 2020] to enhance our deanonymization and linking capabilities.

REFERENCES

Michael Backes, Aniket Kate, Praveen Manoharan, Sebastian Meiser, and Esfandiar Mohammadi. 2013. AnoA: A framework for analyzing anonymous communication

protocols. In *2013 IEEE 26th Computer Security Foundations Symposium*. IEEE, 163–178.

Gabrielle Beck, Julia Len, Ian Miers, and Matthew Green. 2021. Fuzzy message detection. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 1507–1528.

Ferenc Béres, István A Seres, András A Benczúr, and Mikerah Quintyne-Collins. 2021. Blockchain is watching you: Profiling and deanonymizing ethereum users. In *2021 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS)*. IEEE, 69–78.

Vitalik Buterin, Eric Conner, Rick Dudley, Matthew Slipper, Ian Norden, and Abdelhamid Bakhta. 2019. EIP-1559: Fee market change for ETH 1.0 chain. *Published online* (2019).

Benjamin DiFrancesco and Matthew Solomon. 2021. The Umbra protocol. *URL: https://github.com/ScopeLift/umbra-protocol* (2021).

Sashidhar Jakkamsetti, Zeyu Liu, and Varun Madathil. 2023. Scalable Private Signaling. *Cryptology ePrint Archive* (2023).

Zeyu Liu and Eran Tromer. 2022. Oblivious message retrieval. In *Annual International Cryptology Conference*. Springer, 753–783.

Zeyu Liu, Eran Tromer, and Yunhao Wang. 2023. Group Oblivious Message Retrieval. *Cryptology ePrint Archive* (2023).

Varun Madathil, Alessandra Scafuro, István András Seres, Omer Shlomovits, and Denis Varlakov. 2021. Private Signaling. *IACR Cryptol. ePrint Arch.* 2021 (2021), 853.

Malte Möser and Rainer Böhme. 2017. Anonymous alone? measuring bitcoin's second-generation anonymization techniques. In *2017 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. IEEE, 32–41.

Malte Möser and Arvind Narayanan. 2022. Resurrecting address clustering in Bitcoin. In *International Conference on Financial Cryptography and Data Security*. Springer, 386–403.

Shen Noether and Sarang Noether. 2014. Monero is not that mysterious. *Technical report* (2014).

Alexey Pertsev, Roman Semenov, and Roman Storm. 2019. Tornado Cash Privacy Solution Version 1.4. (2019).

Justus Ranvier. 2015. Bip47: Reusable Payment Codes for Hierarchical Deterministic Wallets. *https://github.com/bitcoin/bips/blob/master/bip-0047.mediawiki* (2015).

István András Seres, Balázs Pejó, and Péter Burcsi. 2022. The Effect of False Positives: Why Fuzzy Message Detection Leads to Fuzzy Privacy Guarantees?. In *International Conference on Financial Cryptography and Data Security*. Springer, 123–148.

Peter Todd. 2014. Stealth Addresses. 2014. *URL: https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2014-January/004020.html (visited on 2017-02-10). APPENDIX Multisignature transactions (←) Overall share (in%)(→) per block 0.2 0.4 0.6 0.8* 1, 50 (2014), 100.

Friedhelm Victor. 2020. Address clustering heuristics for Ethereum. In *Financial Cryptography and Data Security: 24th International Conference, FC 2020, Kota Kinabalu, Malaysia, February 10–14, 2020 Revised Selected Papers 24*. Springer, 617–633.

Toni Wahrstätter, Matt Solomon, Ben DiFrancesco, and Vitalik Buterin. 2022. EIP5564: Non-Interactive Stealth Address Generation. *https://github.com/ethereum/EIPs/blob/master/EIPS/eip-5564.md* (2022).

Zhipeng Wang, Stefanos Chaliasos, Kaihua Qin, Liyi Zhou, Lifeng Gao, Pascal Berrang, Benjamin Livshits, and Arthur Gervais. 2023. On how zero-knowledge proof blockchain mixers improve, and worsen user privacy. In *Proceedings of the ACM Web Conference 2023*. 2022–2032.

Zachary J Williamson. 2018. The aztec protocol. *URL: https://github.com/AztecProtocol/AZTEC* (2018).

Gavin Wood et al. 2014. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper* 151, 2014 (2014), 1–32.

Mike Wu, Will McTighe, Kaili Wang, Istvan A Seres, Nick Bax, Manuel Puebla, Mariano Mendez, Federico Carrone, Tomás De Mattey, Herman O Demaestri, et al. 2022. Tutela: An open-source tool for assessing user-privacy on ethereum and tornado cash. *arXiv preprint arXiv:2201.06811* (2022).