BaseSAP: Modular Stealth Address Protocol for Programmable Blockchains

Anton Wahrstätter, Matthew Solomon, Ben DiFrancesco, Vitalik Buterin, and Davor Svetinovic

***Abstract*—Stealth addresses represent an approach to enhancing privacy within public and distributed blockchains, such as Ethereum and Bitcoin. Stealth address protocols employ a distinct, randomly generated address for the recipient, thereby concealing interactions between entities. In this study, we introduce BaseSAP, an autonomous base-layer protocol for embedding stealth addresses within the application layer of programmable blockchains. BaseSAP expands upon previous research to develop a modular protocol for executing unlinkable transactions on public blockchains. BaseSAP allows for the development of additional stealth address layers using different cryptographic algorithms on top of the primary implementation, capitalizing on its modularity. To demonstrate the effectiveness of our proposed protocol, we present simulations of an advanced Secp256k1-based dual-key stealth address protocol. This protocol is developed on top of BaseSAP and deployed on the Ethereum test network as the first prototype implementation. Furthermore, we provide cost analyses and underscore potential security ramifications and attack vectors that could affect the privacy of stealth addresses. Our study highlights the flexibility of the BaseSAP protocol and provides insights into the broader implications of stealth address technology in the realm of blockchain privacy.**

***Index Terms*—Blockchain, Privacy, Security, Confidentiality, Ethereum, Stealth Address**

I. Introduction

STEALTH addresses have gained importance in blockchain technology due to their potential to improve confidentiality and privacy in transactions on blockchains. In the context of public blockchains, all transactions are recorded transparently, making it possible to track the transaction history of a particular pseudonymous user. This traceability could occur unintentionally, as the parties involved in a transaction may not have consciously aimed to establish a linkable record. Nonetheless, since blockchains like Bitcoin and Ethereum are transparent and publicly accessible, third parties can analyze the data and potentially identify the participants in a particular transaction.

A. Wahrstätter is with the Research Institute for Cryptoeconomics, Department of Information Systems and Operations Management, Vienna University of Economics and Business, Vienna, Austria. E-Mail: anton.wahrstaetter@wu.ac.at

M. Solomon and B. DiFrancesco are with ScopeLift, Broomall, Pennsylvania, United States. E-Mail: {matt, ben}@scopelift.co

V. Buterin is with the Ethereum Foundation, Singapore, Singapore. E-Mail: vitalik@ethereum.org

D. Svetinovic is with the Center for Secure Cyber-Physical Systems, Department of Computer Science, Khalifa University, Abu Dhabi, UAE, and the Research Institute for Cryptoeconomics, Department of Information Systems and Operations Management, Vienna University of Economics and Business, Vienna, Austria.E-mail: dsve@acm.org (Corresponding author.)

Manuscript received June 2023.

Stealth address protocols (SAPs) offer a solution to the privacy challenges faced on public blockchains by enabling users to interact confidentially without allowing external observers to link the parties involved in a transaction. At their core, SAPs empower the shielding of recipient information in peer-to-peer (P2P) transactions [1], [2].

Stealth addresses can have numerous applications, including but not limited to donations and payroll payments. They can be used in any P2P interaction where privacy concerns demand concealing the connection between two parties. Users can transfer funds using SAPs while protecting the recipient's identity.

The first stealth addresses were initially introduced in the Bitcoin ecosystem and have since undergone continuous refinement. In 2013, Nicolas van Saberhagen described the CryptoNote protocol, which utilized stealth addresses to enhance the privacy of blockchain transactions [1]. Peter Todd subsequently built upon this concept in 2014 and further improved it [2]. Ultimately, stealth addresses were integrated into the Monero blockchain when it launched in 2014 [3].

On the Ethereum blockchain, stealth addresses can significantly improve confidentiality. They allow users to autonomously generate a unique, one-time address for each transaction instead of relying on a static, publicly identified address. The programmable nature of the Ethereum blockchain facilitates the development of SAPs on top of it, hence leveraging the decentralization and trust attributes of the underlying blockchain [4].

While stealth addresses offer considerable potential, they currently exhibit multiple limitations affecting their anticipated implementation and effectiveness across various blockchain platforms. In the context of privacy-focused blockchains like Monero, stealth addresses are inherently incorporated within the core protocol. Conversely, popular programmable blockchains, notably Ethereum, lack inherent, comprehensive privacy protections at the base protocol level, thereby requiring additional measures at the application layer to achieve similar levels of privacy assurance [3].

The potential for SAPs on blockchains such as Ethereum becomes evident within this context. They can introduce innovative privacy features by leveraging the Turing-complete environment of Ethereum and the modularity of the base blockchain protocol. This interoperability could be advantageously employed in various applications, including Smart Contract wallets, public goods funding, Decentralized Finance (DeFi) systems, or the Non-Fungible Token (NFT) landscape, thereby broadening the potential application areas for these

protocols.

However, it is crucial to address the scalability challenges inherent in conventional SAPs. These must be resolved to facilitate the mainstream acceptance of stealth addresses while guaranteeing a secure user experience.

To overcome the aforementioned challenges, we developed BaseSAP. BaseSAP is a fully open and reusable SAP that can reliably offer stealth addresses at the application layer of programmable blockchains like Ethereum. The protocol aims to provide a lightweight mechanism for users to generate stealth addresses, maintaining complete backward compatibility and requiring no modifications to the core blockchain. Our proposed base protocol is agnostic towards various cryptographic schemes and holds the potential to substantially improve user interactions with stealth addresses in the context of programmable blockchains. BaseSAP comprises a foundational implementation, which includes the reusable functionality required for most trustless SAPs.

We designed BaseSAP to be fully extendable, thereby enabling the creation of unique SAPs based on particular cryptographic schemes on top of it. Examples of such extensions include stealth addresses derived from the Secp256k1 elliptic curve, SAPs based on elliptic curve pairings [5]–[7] or generated using lattice-based cryptography [8], [9]. The protocol design ensures compatibility and proactively accommodates future quantum-resistant cryptographic schemes that require larger key sizes.

Beyond the base protocol contribution, we create the first practical implementation on top of BaseSAP. We implement an improved dual-key protocol that relies on the Secp256k1 elliptic curve and employs "view tags" to improve parsing efficiency compared to conventional Dual-Key Stealth Address Protocols (DKSAPs). The key contributions of this work are:

- We present a detailed analysis of the current state of research and deployment of stealth address technology, examining its application and development across diverse blockchains.
- We identify and address substantial challenges associated with interoperable SAPs, emphasizing privacy concerns and Denial-of-Service (DoS) attack vulnerabilities.
- We design and develop BaseSAP as a fully open, cohesive, and extendable SAP to be integrated into Ethereum [10] in active collaboration with the Ethereum development community.
- We illustrate the inherent modularity of our protocol, accentuating the significant potential of such approaches when implemented at the application layer of programmable blockchains.
- We develop a preliminary stealth address prototype that leverages the Secp256k1 elliptic curve and exhibits superior performance in terms of parsing time when compared to existing SAPs [11].

We publish the code base created for this work under an open-source license to ensure reproducibility and transparency [11]. Additionally, we propose the described protocol as an ERC (Ethereum Request for Comment) [10] to facilitate the adoption of stealth addresses on programmable and decentralized blockchains.

II. Related Work

Prior research has established the basis for the development of the proposed protocol. This section focuses on the most relevant literature and provides an overview of the current state of the art concerning stealth addresses.

Stealth addresses were first introduced to the blockchain domain by an anonymous entity dubbed "bytecoin" in April 2011 [12]. Subsequently, van Saberhagen and Todd put forward more refined SAPs in 2013 and 2014, respectively [1], [2]. These protocols formed the basis for the DKSAP implemented in the Monero blockchain upon its launch in 2014 [3]. Since DKSAP's inception, numerous researchers have sought to extend the capabilities of this SAP by introducing additional features and functionality.

Courtois and Mercer [13] provide an overview of the development history of stealth addresses. Furthermore, the authors introduce multiple different spending keys to the DKSAP, improving its resistance to attacks such as the "bad random attack" or compromised keys. Their proposed protocol comes at the cost of requiring the users to manage multiple different spending keys.

Fan [14] improves the DKSAP, enabling sender and receiver pairs to use their generated Diffie-Hellman secret multiple times with an increasing counter, enabling a faster parsing process for users. This approach is based on a similar idea as TLS and achieves performance gains of at least 50% compared to the standard DKSAP.

Fan *et al.* [7] improve the DKSAP by halving the number of required keys from two to one, significantly reducing storage costs while employing bilinear mapping to preserve the protocol's desired properties. The authors demonstrate notable efficiency gains through this reduction in stored key pairs.

Liu *et al.* [8] implement stealth addresses together with ring signatures to define a confidential layer within a cryptocurrency system. The authors use a lattice-based protocol to fully shield the information about the sender and recipient of a transaction.

Feng *et al.* [15] propose a SAP that does not require additional information to be published with every stealth address transaction, allowing such transactions to look like common blockchain transactions. The authors use the number of transactions between certain peers instead of generating a Diffie-Hellman secret for the stealth address generation process. This comes with the requirement for users to parse every transaction recorded on the blockchain.

Lee and Song [16] use a SAP and ring signatures to implement confidential transactions on an Ethereum private network. The authors focus on the exchange of healthcare information and further analyze the security of their protocol using threat models.

AbdulKader and Kumar [17] build on top of the protocol of [15] and use the transaction ID of the most recent P2P transaction between two entities instead of the number of transactions in the stealth address generation process. The authors argue that without the need to attach information to the stealth address transaction, related protocols become more censorship-resistant and lightweight.

TABLE I
SUMMARY OF RELATED WORK ON STEALTH ADDRESSES AND COMPATIBILITY WITH BASESAP

Author	Year	Technique	Nr. of Keys	Extra Data Requirement	BaseSAP Compatible
Bytecoin [12]	2011	Elliptic Curve Diffie-Hellman key exchange (ECDH)	One	Yes	Yes
Van Saberhagen [1]	2013	ECDH + Dual-Key Stealth Address Protocol (DKSAP)	Two	Yes	Yes
Todd [2]	2014	ECDH	One	Yes	Yes
Monero [3]	2014	ECDH + DKSAP	Two	Yes	Yes
Courtois and Mercer [13]	2017	ECDH + DKSAP with multiple key pairs	Multiple	Yes	Yes
Fan [14]	2018	ECDH + DKSAP with improved parsing	Two	Yes	Yes
Fan *et al.* [7]	2019	Bilinear Mapping	One	No	N/A
Liu *et al.* [8]	2019	Lattice-based SAP	Two	No	N/A
Feng *et al.* [15]	2020	ECDH + DKSAP with improved parsing	Two	No	N/A
Lee and Song [16]	2021	ECDH	One	Yes	Yes
Feng *et al.* [5]	2021	Bilinear Mapping	Two	Yes	Yes
Mohideen and Kumar [17]	2022	ECDH + DKSAP with improved parsing	Two	No	N/A

In summary, previous research underscores a substantial acceptance of the DKSAP. Several studies focused on reducing parsing time for recipients by introducing efficient strategies, such as deterministic rules that dictate the computation of stealth addresses between two parties based on an initially generated Diffie-Hellman secret or adopt sophisticated cryptographic algorithms such as bilinear mappings [7], [14].

Furthermore, mitigating the problem of detectability in stealth address transactions can be achieved by refraining from publishing any extra information alongside stealth address transactions. However, this approach entails a significant drawback for blockchains that handle a large volume of transactions, in addition to stealth address transactions, as it requires parsing each transaction recorded.

In Table I, we provide an overview of existing SAPs, focusing on two key aspects: the use of multiple keys (commonly known as scanning and spending keys) and the requirement to publish supplementary information with a stealth address transaction. Protocols such as [7], [15], and [17] don't require the publication of additional data, enabling enhanced privacy. These stealth address transactions blend seamlessly with regular transactions, making them indistinguishable. However, these protocols have a major limitation: users have to scan every single transaction on the blockchain. This is because any transaction could potentially be related to a stealth address. This requirement makes such protocols less practical and relevant for blockchains such as Ethereum. Protocols that do not require additional information for stealth address transactions are unsuitable for integration with the BaseSAP framework. While this extra information makes stealth address transactions detectable, it offers a significant advantage in parsing. It narrows down the number of transactions that recipients need to scrutinize, thereby streamlining the identification of relevant stealth address transactions.

III. BACKGROUND ON BLOCKCHAIN PRIVACY

Privacy remains a primary concern within the realm of public blockchains. The inherent transparency of these systems may jeopardize users' privacy when conducting financial transactions or other sensitive interactions. To address this issue, blockchain developers have attempted to develop privacy-enhancing protocols that provide unlinkability and untraceability or focus exclusively on the former. In this context, we adhere to the definitions established in the CryptoNote whitepaper to define "unlinkability" and "untraceability." Per this reference, unlinkability is characterized as the inability to verify that two outgoing transactions are directed to the same recipient. Untraceability, on the other hand, is the inability to pinpoint the sender of a transaction from a group of potential senders [1].

ZK-SNARKs. There have been numerous efforts to bring confidential transactions to public ledgers such as Bitcoin and Ethereum, including the use of ZK-SNARKs ("Zero-Knowledge Succinct Non-Interactive Argument of Knowledge") [18]–[21]. ZK-SNARKs enable a user to prove certain information without disclosing that information, which allows for the possibility of depositing funds into a Smart Contract using one pseudonym and then withdrawing those funds by proving the deposit under a different pseudonym without disclosing which deposit was referenced for the withdrawal. In addition to promoting enhanced scalability in the blockchain, this technology is implemented on Ethereum through privacy-enhancing tools like Tornado Cash or Privacy Pools and in Zero-Knowledge rollup platforms such as Aztec. This technology ensures both untraceability and unlinkability, thereby offering a robust means of preserving privacy. [22]–[24].

CoinJoin. Chaumian CoinJoin is a privacy-enhancing technology used on UTXO-based blockchains that ensures the untraceability and unlinkability of transactions. CoinJoin is a process in which multiple users combine their unspent transaction outputs (UTXOs) into a single, larger transaction. This consolidation complicates the task of an external observer who tries to correlate input addresses (the senders) with output addresses (the recipients) [25], [26]. CoinJoins have been implemented in applications such as Wasabi Wallet, Samurai Wallet, and JoinMarket [27]. Blind signatures are employed to guarantee that the central coordinator cannot link the input and output addresses of the participants. Applied correctly, CoinJoins prevent the central coordinator or any other third

party from tracing the flow of funds and de-anonymizing users [28].

Stealth Addresses. Stealth addresses are a privacy-enhancing solution that hides the recipient of a transaction, thereby preventing third parties from linking the interacting parties. By enabling senders to create a new stealth address for the recipient in a non-interactive manner, such protocols can provide unlinkability. While stealth addresses can be implemented on the application layer of programmable blockchains, some projects, such as Monero [3], opted to integrate them into the core protocol. Furthermore, stealth addresses can be employed on UTXO and account-based blockchain protocols. One key difference between stealth addresses and ZK-SNARK-based solutions is the extent of privacy that can be achieved. ZK-SNARK-based privacy applications are commonly used to prove ownership of an asset without necessarily possessing that asset at the time. This allows for commingling funds with those of other users and consequently eliminating discernible on-chain traces. Stealth addresses obfuscate the recipient's ownership within a transaction by employing newly generated pseudonymous addresses. The funds remain traceable as funds are not commingled with those of third parties. This distinction is essential to consider when evaluating the suitability of these privacy-enhancing solutions for different use cases.

Another distinction is the computational overhead: by the time of writing, ZK-SNARKs typically have a more significant overhead regarding computational resources and setup, while stealth addresses can be implemented with minimal impact using existing tools that modern blockchain platforms already provide.

Unlike CoinJoins or mixing pools, stealth addresses do not aim to obfuscate the on-chain visible flow of funds but instead hide the interaction between an identified sender and recipient. Consequently, external observers can trace the flow of funds to specific stealth addresses. However, observers cannot identify the individual or entity behind those recipient addresses. In contrast, CoinJoins provide an additional privacy layer by allowing users to conceal their identities within an anonymous group of CoinJoin participants. Therefore, CoinJoins offer more comprehensive anonymity than stealth addresses but rely on more user interaction and coordination.

Despite the mentioned variations, we assert that employing stealth addresses through elliptic curve mathematics offers a more lightweight and interoperable approach, making it accessible to a larger audience. Additionally, the inherent decentralization of stealth addresses contributes to the protocol's robustness and further promotes the principles of autonomy and user privacy. This can potentially enhance the adoption of privacy features on public blockchains. Numerous applications, such as donations or payroll transactions, may not demand the high anonymity offered by notable ZK-SNARK-based tools or CoinJoin wallets. In particular, in situations that require Know Your Customer (KYC) procedures, stealth addresses present a more suitable solution. Moreover, the lack of commingling funds means that users do not inadvertently help malicious parties anonymize ill-gotten assets by contributing to an extended anonymity set. This property makes stealth addresses particularly suitable for interactions where it is desired to refrain from helping malicious parties.

IV. Definition

The following sections outline the various components of our proposed protocol, BaseSAP. Given that our first implementation is based on elliptic curve (EC) cryptography, we introduce the foundational principles of elliptic curves. We then define our SAP, divided into *address generation* and *parsing* sections.

A. Elliptic Curve Cryptography

We define an elliptic curve E over a finite field F_p where p is a 256-bit prime and present it in Weierstrass form as

$$y^2 = x^3 + ax + b \mid x, y \in F_p$$

with a and b representing constants that determine the shape and position of the respective curve. The coordinates (x, y) are points on the elliptic curve that can take any value within F_p and form an Abelian group. This group structure allows us, given two points, e.g., P and Q to solve for R by performing a binary operation called point addition, such that $R = P+Q$. For any points P and Q on the curve, we know $P + Q$ must also be on the curve.

We denote lower and uppercase letters to scalars and points on the curve, respectively. The scalars p and q are random integers of size n, such that $p,\ q \in \{0,1\}^n$. It is common for private keys, such as those used in the Secp256k1 curve, to have a size of 256 bits [29]. An EC multiplication of the point P by the scalar n can be done by repeatedly performing additions of the point along the curve, such that $n \times P = P_i + \ldots P_{i+n}$. Another property of the point addition operation on an EC group is that it is commutative, meaning for all points, e.g., P and Q, $Q+P = P+Q$. The EC has a generator point G, representing a fixed curve point. A public key can be derived by multiplying a scalar with the generator point, $P = p \times G$.

We denote the "point at infinity" $\mathcal{O}$ as the identity element of the EC arithmetic, such that $\mathcal{O} + \mathcal{O} = \mathcal{O}$ and $P + \mathcal{O} = P$. Finally, for every point P on the elliptic curve, there exists an inverse point such that $(-P) + P = \mathcal{O}$.

The Standards for Efficient Cryptography (SEC) is a set of standardized elliptic curves proposed for use in cryptography. These curves are designated as "SEC curves" and are intended to provide a standard set of curves for use in various cryptographic applications.

One of the most well-known SEC curves is Secp256k1, which is defined by the equation $y^2 = x^3 + 7 \pmod p$, where $p = 2^{256} - 2^{32} - 977$. This curve has a prime order n of approximately 2^{256}, and it is used as the basis for the Bitcoin Elliptic Curve Digital Signature Algorithm (ECDSA) [29].

Secp256k1 has several attractive properties, including a large prime order and efficient arithmetic, making it well-suited for use in cryptocurrencies. It has been widely adopted in various applications, including blockchain technologies, IoT, and secure communication protocols [30]–[32].

Our proposed stealth addresses protocol is designed to be agnostic towards the specific elliptic curve employed, although the initial implementation utilizes the Secp256k1 curve from the SEC set.

B. Stealth Address Protocol

The following section is divided into different parts. First, we detail how users generate a stealth address using the Improved Stealth Address Protocol (ISAP), described by Todd [2], when they want to perform a transaction. Second, we cover the DKSAP, described by van Saberhagen [1], primarily focusing on the parsing process that the receiver or third-party parsing providers can perform.

Stealth Address Generation. For the stealth address generation, we define two independent parties, the sender $\mathcal{C}$ (cf. *caller*) and the recipient R, who both have access to a cryptographic key pair, (p, P) and (r, R). We assume the public key of the recipient R is published and known to the respective sender. Furthermore, it is important to note that the sender uses an ephemeral key pair, (p, P), randomly generated for each transaction using the SAP instead of using a key pair with a public key directly linked to their identity.

The sender generates a shared secret using the Elliptic Curve Diffie-Hellman (ECDH) protocol to derive a stealth address for interaction with the recipient. A stealth address is generated by adding the point obtained from multiplying the Diffie-Hellman (DH) secret with the generator point to the recipient's public key. The sender performs the following steps for this process:

1) Generate an ephemeral key pair (p, P) and publish the coordinates P.
2) Multiply the randomly generated ephemeral private key with the recipient's public key: $k = p \times R$. This creates the DH secret, such that $k = r \times P = p \times R = r \times p \times G$.
3) Hash the shared DH secret $k_h = h(k)$, where h represents a cryptographic hash function. In this context, let k denote the input domain of h, which consists of an EC point representing the DH secret between the sender and recipient. The output domain, represented as k_h, is a scalar value. This transformation from k to k_h is crucial as it enables further cryptographic operations by converting the EC point into a scalar.
4) Multiply the hashed shared secret with the generator point of the elliptic curve $\mathcal{K}_h = k_h \times G$.
5) Add the result of (4) to the recipient's public key: $R_{st} = \mathcal{K}_h + R$.

Let $R_{st} \in E(F)$ denote the point which party $\mathcal{C}$ uses as a stealth address for R. It is important to note that there is no direct link between the R and the derived stealth address, and to external observers, it appears as if $\mathcal{C}$ is interacting with a random account unrelated to R. Furthermore, the protocol ensures that only the owner of r can access R_{st} by deriving the private key r_{st} through parsing.

Stealth Address Parsing. The stealth address parsing process allows potential transaction recipients to locate their stealth address and obtain the private key required to access it.

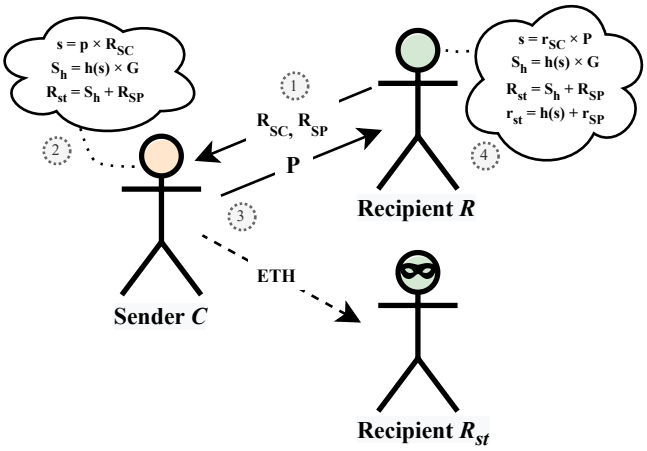

Fig. 1. ISAP + DKSAP: (1) sender obtains public keys of the recipient, (2) generates the stealth address, and (3) sends to the stealth address and publishes an announcement; (4) recipient uses the announcement to derive the private key that unlocks the stealth address.

This procedure requires that every potential recipient conduct parsing across the complete set of published ephemeral public keys, denoted by $\mathcal{A} = \{P_i, \ldots, P_n\}$. The total number of unique stealth addresses generated via the protocol is represented by $|\mathcal{A}|$. To execute parsing, a potential recipient must first gather the set of all existing ephemeral public keys and then perform the subsequent steps on each $P \in \mathcal{A}$:

1) Multiply P with the private key r: $k = r \times P$.
2) Hash the derived shared secret $k_h = h(k)$.
3) Add the result of (2) to the own private key: $r_{st} = k_h + r$.
4) Multiply the result of (3) with the generator point to derive the stealth public key: $R_{st} = r_{st} \times G$.
5) Hash the stealth public key and take the least significant 20 bytes to derive the Ethereum address: $R_{st}^{addr} = h(R_{st})$[-20 :].

Upon deriving the point R_{st}, the recipient can determine whether R_{st}^{addr} has been the recipient of the transaction or whether R_{st}^{addr} received any assets. If the check is successful, the recipient may store the private key r_{st}.

To conclude, the protocol leverages the fact that $k_h \times G + P = (k_h + p) \times G$. This allows for deriving a stealth address through two different paths, while only the recipient can generate the private key for the stealth address.

Dual-key scheme. Stealth addresses on Ethereum require the recipient to use their private key r during the process. This has important implications for both security and user experience. First, users may encounter situations where they must use their private keys for operations outside of their cold storage, which poses significant security risks. Second, users cannot delegate the parsing process to a third-party service as it involves sharing the private key and compromising its confidentiality. Therefore, users must perform the parsing process themselves in a local environment.

Researchers have developed a solution to these issues, the

DKSAP, which improves both the user experience and security [13]–[15]. The DKSAP is an extension of the ISAP and introduces an additional key pair exclusively used for the parsing process. Recipients have two key pairs — scanning and spending keys — represented as $(r_{\mathcal{SC}}, R_{\mathcal{SC}})$ and $(r_{\mathcal{SP}}, R_{\mathcal{SP}})$, respectively. Equipping the recipient with two separate key pairs, the scanning key pair, which is still used in the DH secret generation, can be partially separated from the stealth address generation. To use the DKSAP, the sender needs to follow these steps:

1) Multiply the randomly generated ephemeral private key with the scanning public key of the recipient: $k = p \times R_{\mathcal{SC}}$.
2) Hash the shared secret $k_h = h(k)$ and multiply the result with the generator point $\mathcal{K}_h = k_h \times G$.
3) Add the result of (2) to the recipient's spending public key: $R_{st} = \mathcal{K}_h + R_{\mathcal{SP}}$.

Henceforth, the recipient has two options for deriving the respective stealth address public key R_{st}. First, the recipient can compute the DH secret by multiplying the scanning private key $r_{\mathcal{SC}}$ with the ephemeral public key $P \in \mathcal{A}$. Having the DH secret, the recipient can derive the stealth address public key by hashing it, multiplying the result with the generator point, and adding the result to the spending public key. Second, the recipient can add the hashed DH secret to the spending private key and multiply the result with the generator point for deriving the stealth address public key R_{st}:

$$R_{\mathcal{SP}} + h(r_{\mathcal{SC}} \times P) \times G = (r_{\mathcal{SP}} + h(r_{\mathcal{SC}} \times P)) \times G.$$

It is important to note that the recipient can share the scanning private key $r_{\mathcal{SC}}$ with a third-party parsing provider without compromising the spending private key. Using the scanning key, the parsing provider can take on the parsing task and notify users when an incoming stealth address transaction occurs. However, without access to the spending private key $r_{\mathcal{SP}}$, parsing providers cannot access the stealth address.

V. BaseSAP Protocol

In the following, we describe our proposed SAP in detail. This involves on-chain key management solutions, stealth address transaction routing, and specific efficiency improvements we propose to the DKSAP.

Our protocol is designed to operate on the application layer of programmable blockchains such as Ethereum and does not require integration with the core protocol layer of a blockchain. After deployment, BaseSAP operates autonomously, eliminating the possibility of user interference or censorship of any party.

The proposed protocol can serve as a foundation for various implementations to build upon it and leverage the modular basis. BaseSAP comprises a single singleton contract, the *Announcer* contract, which enables users to publish the ephemeral public keys at a central place.

For the initial implementation of an improved version of a DKSAP, built on top of BaseSAP, we propose a *Registry* contract that serves as a central repository for "stealth meta-addresses" associated with registered users.

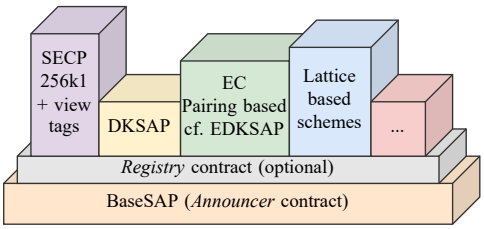

Fig. 2. Modular property: BaseSAP enables different stealth address schemes to build on top of it and leverage the underlying foundational protocol's modularity, interoperability, and trust.

We practically implement our proposed protocol on both the Ethereum Goerli and Sepolia testnets to analyze the performance of the proposed protocol under practical conditions and enable the community to engage with it.

A. BaseSAP Announcer contract

To interact with a user's stealth address, the sender must first obtain the recipient's public key and then generate a stealth address. This process involves the sender using their own randomly generated ephemeral private key and the recipient's public key.

To enable recipients to detect their stealth addresses, senders must publicly announce their ephemeral public keys. Adversaries cannot exploit this announcement to compromise a recipient's privacy, as they cannot recreate the necessary DH secret for generating the stealth address.

The *Announcer* contract emits announcements from a central location to which users can subscribe. Unlike [15], where the objective was to have stealth address transactions mimic regular transactions, providing additional information — particularly the ephemeral public key — in conjunction with each stealth address transaction is necessary. Without attaching additional information to a stealth address interaction, the protocol would require users to parse the entire ledger [15]. However, by emitting events containing that information, users can utilize the existing Bloom filters on the Ethereum blockchain to identify the set of transactions related to stealth address interactions.

The *Announcer* contract is designed to be agnostic towards the cryptographic scheme used, enabling different implementations of various cryptographic schemes to share the same source of event emissions. This means that the same *Announcer* contract can be used for multiple distinct implementations without needing modification or adjustment. This essential characteristic is illustrated in Figure 2.

Announcements. Employing BaseSAP, the asset transfer transaction serves a dual purpose. It transfers assets and broadcasts an announcement containing the ephemeral public key to the public, thus making this information accessible to the recipient. To facilitate this feature, a lightweight *Announcer* contract is used (refer to Listing 1). The *Announcer* contract can be called by anyone to emit additional information along with a transaction.

External Owned Accounts (EOAs) can call the *Announcer* contract using regular transactions, while contracts can interact

with the *Announcer* through internal calls to execute the *announce* function. As shown in Listing 1, the caller can provide four parameters to the function:

1) Scheme ID — to specify the cryptographic scheme that was used. In BaseSAP, the Secp256k1 implementation is assigned the identifier number 1. An incrementing number is assigned for subsequent implementations.
2) Stealth address — the address of the transaction recipient.
3) Ephemeral public key — compressed public key derived from the randomly generated ephemeral private key.
4) Metadata — arbitrary information that may be helpful to the recipient in identifying the particular interaction between the sender and the stealth address. For ERC-20 tokens, the metadata field may include the four most significant bytes of the Method ID, 20 bytes for the (token) contract address, and 32 bytes carrying the amount transferred. For ERC-721/ERC-1155 contracts, the metadata field may include the token ID instead of the amount transferred. For regular ETH transfers, the metadata field may remain empty.

Incorporating an extra metadata field within the announcement enables recipients to verify receipt of an asset, including the amount transferred and the specific token involved. Moreover, by incorporating the Method ID in the metadata, recipients can pinpoint the contract interaction involving their stealth addresses. As a result, well-known interactions, such as token approvals or mints that carry the right to execute on a certain state, are compatible with stealth addresses. Consequently, the recipient is not obligated to perform extra Remote Procedure Calls (RPCs) to obtain information about the nature or quantity of an asset or right received.
Considering the metadata field's dynamic size, it can also be employed to incorporate additional features for enhancing parsing at a later stage or to include more information that might be needed for future token standards.

Costs. Executing the *announce* function with the parameters used for an ERC-20 transfer consumes approximately 35,492 units of gas on the Ethereum blockchain. If the metadata field is left empty, as it would be in the case of Ether transfers, the gas usage is reduced to 34,057 units of gas. Assuming a gas price of 10 $gwei$ and a price for 1 Ether (ETH) of 2,000 US dollars, with 1 ETH = $10e9$ gwei = $10e18$ wei, the cost of calling the *announce* function for Ether transfers is approximately 0.68 USD, or 0.00034057 ETH. On layer-2 rollup platforms such as Optimism[1] or Arbitrum[2], the costs associated with the announcement are effectively negligible.
Compressing the ephemeral public key to 33 bytes can reduce the gas consumption to 35,064 units. Without using the metadata field, the emission of the announcement consumes 33,629 units of gas. From these figures, we can deduce that applying public key compression results in cost savings of 1.21% for non-empty metadata emissions and 1.26% for ETH transactions that do not require metadata.

[1]https://www.optimism.io/
[2]https://arbitrum.io/

```solidity
pragma solidity ^0.8.0;

/// @notice Announcer emitting the Announcement event.
interface BaseSAPAnnouncer {

  /// @notice Emitted when interacting with a stealth
       address.
  event Announcement (
    uint256 indexed schemeId,
    address indexed stealthAddress,
    address indexed caller,
    bytes ephemeralPubKey,
    bytes metadata
  );

  /// @notice To be called when interacting with a
       stealth address.
  function announce (
    uint256 schemeId,
    address stealthAddress,
    bytes memory ephemeralPubKey,
    bytes memory metadata
  )
    external
  {
    emit Announcement(
      schemeId,
      stealthAddress,
      msg.sender,
      ephemeralPubKey,
      metadata
    );
  }
}
```

Listing 1. Announcer Contract Interface

B. Stealth Meta-Address Format

In the design of the DKSAP, the recipient has two separate key pairs, the spending $\mathcal{SP}$ and the scanning keys $\mathcal{SC}$. We combine the two public keys to generate the "stealth meta-address", enabling a more intuitive way for users to interact with each other. In Secp256k1, the public keys PKs can be compressed to 33 bytes each, denoted as PK_{comp}. Consequently, our proposed protocol uses the following format for the stealth meta-address:

$$st : \langle chainID \rangle \ : \ 0x \langle PK^{\mathcal{SP}}_{comp} \rangle \langle PK^{\mathcal{SC}}_{comp} \rangle$$

The "st" prefix indicates that the following address refers to a stealth meta-address. The "chainID" parameter distinguishes blockchain-specific addresses with which the corresponding recipient can interact via stealth addresses. Within the Ethereum ecosystem, chain IDs have been formalized in the ERC-3770.

To compress the public key, we store only the x-coordinate of the public key point and prefix it with either 0x2 or 0x3, depending on whether the y-coordinate is positive or negative, respectively. The stealth meta-address can be shared through off-chain communication channels or made publicly available on the blockchain. This way, any individual can generate stealth addresses on behalf of the user, thereby facilitating their interaction.

The uncompressed public keys can be used instead for cryptographic schemes where compressing the public key is not feasible.

C. Secp256k1 Implementation

In the following, we propose efficiency improvements to the DKSAP to make it more viable for implementation on blockchain platforms like Ethereum. Our analysis of the existing protocols highlights two deficiencies that impede their practical usage. First, we observe that the parsing process required by potential recipients to decode every announcement can be excessively time-consuming. Second, we note that the announcement, which merely contains the ephemeral public key, does not offer sufficient information for recipients to identify the relevant assets and rights in a stealth interaction.

To remedy these limitations, we focus on enhancing the efficiency and flexibility of the recipients by modifying the announcement and publication process and introducing a "view tag" approach. We intend to enhance the overall functionality of the protocol, thereby facilitating its application in blockchain environments.

Announcement. The set of announcements, denoted by $\mathcal{A} \mid a \in \mathcal{A}$, contains information that enables prospective recipients to identify themselves as the intended recipient of a stealth address transaction and locate the corresponding stealth address. Generally, the recipient utilizes the ephemeral public key disclosed by the sender to compute the stealth address and validate that it corresponds with the recipient address of the transaction. This procedure allows users to confirm that they are indeed the rightful recipients of the transaction. This process involves two RPC requests — one for obtaining the announcement and another for retrieving the transaction recipient.

To refine the parsing process and enhance its flexibility, we propose integrating the stealth address R_{st}^{addr} into the announcement a that is emitted for every stealth address transaction. This enables a direct comparison between the derived stealth address R_{st}^{addr} and the address listed in the announcement R_{st}^{addr}', enabling the recipient to determine if they are the intended recipient without the need to initiate supplementary RPC requests to obtain transaction recipients or query balances for different assets on the derived stealth address. The condition $R_{st}^{addr} == R_{st}^{addr}$' confirms whether the parsing user is the intended recipient. By consolidating all the vital information the recipient needs within the announcement, we obviate the need for the recipient to initiate an extra RPC call.

View Tag. View tags represent a technique employed within the Monero blockchain protocol, allowing recipients of stealth address transactions to bypass certain steps in the parsing process under specific conditions [3], [33]. Rather than computing the stealth address and comparing it to the address in the announcement, the recipient can hash the DH secret and compare the most significant n bytes with the view tag documented in the announcement. In this case, n can be kept very small, so setting $n = 1$ means that a full derivation of the stealth address must only be attempted 1/256 of the time at the cost of only a single-byte view tag. To construct the view tag, senders use their ephemeral key pair (p, P) to compute the hashed DH secret $k_h = h(p \times R_{\mathcal{SC}})$ and select the most significant n bytes of k_h. The resulting view tag $\mathcal{Q}$, where $\mathcal{Q} = k_h[:n]$, is disclosed alongside the stealth address transaction.

The recipient, possessing the scanning key pair $(r_{\mathcal{SC}}, R_{\mathcal{SC}})$, can also compute the view tag by following the same procedure, $\mathcal{Q}' = h(r_{\mathcal{SC}} \times P)[:n]$, and compare it to the view tag listed in the announcement, $\mathcal{Q} == \mathcal{Q}'$. If the view tags do not match, the parsing user can omit every subsequent operation for the current announcement and advance to the next one. It is worth noting that exposing n bytes of the hashed DH secret affects the users' privacy, as attackers may attempt to brute-force a user's stealth address by applying the view tag to potential recipients. Nevertheless, such attacks are likely to be successful only for sufficiently large n.

The parsing process then involves the following steps on each announcement $(P, R_{st}^{addr}, \mathcal{Q}) \in a \mid \forall a \in \mathcal{A}$:

1) Multiply P with the scanning private key $r_{\mathcal{SC}}$: $k = r_{\mathcal{SC}} \times P$.
2) Hash the derived shared secret $k_h = h(k)$.
3) Derive the view tag $\mathcal{Q}$'$= k_h[:n]$
4) Compare the derived view tag with the one in the announcement $\mathcal{Q} == \mathcal{Q}'$.
5) Only if the view tag matches, the recipient continues to compute the stealth address and compare it to the address logged R_{st}^{addr}'$= h((k_h + r_{\mathcal{SP}}) \times G)[\text{-}20 :] = R_{st}^{addr}$.

Like Monero, we use view tags that are 1 byte in size and insert them into the metadata field, taking up the first byte.

D. Public Key Management

Considering the usage of dual-key mechanisms, we advocate for integrating a key management solution that facilitates blockchain users to store their stealth meta-addresses publicly in a predefined location. Absent a dual-key configuration, a central repository for storing public keys would not be necessary. Users could alternatively derive another user's public key by extracting it from a transaction that the latter has signed. It is crucial to mention that the SAP could still be employed even without a central repository. Thus, any key management solution can be built atop the fundamental protocol.

We design a fully autonomous and lightweight registry contract to maintain a record of registered users and their corresponding stealth meta-addresses. This contract predominantly consists of getter and setter methods that assist users in registering their stealth meta-addresses on the blockchain or retrieving those of others. Moreover, the registry permits users to register a stealth meta-address on behalf of another user by providing a valid signature from the respective registrant. Finally, an event is broadcasted each time a user registers a new stealth meta-address.

Our proposed registry contract includes dynamic size storage slots for the stealth meta-address to ensure compatibility with various elliptic curves and cryptographic schemes. This allows for the construction of supplementary stealth address implementations on top of the existing framework, capitalizing on the benefits of a shared registry. Users can register distinct stealth meta-addresses for different cryptographic schemes by

```solidity
pragma solidity ^0.8.0;

interface BaseSAPRegistry {

  /// @dev Emitted when a registrant updates their
        stealth meta-address.
  event StealthMetaAddressSet(
    bytes indexed registrant,
    uint256 indexed scheme,
    bytes stealthMetaAddress
  );

  /// @notice Maps a registrant's identifier to the
        scheme to the stealth meta-address.
  mapping(bytes => mapping(uint256 => bytes)) public;

  /// @notice Sets the caller's stealth meta-address for
        the given stealth address scheme.
  function registerKeys(
    uint256 scheme,
    bytes memory stealthMetaAddress
  ) external;

  /// @notice Sets the `registrant`s stealth
        meta-address for the given scheme.
  function registerKeysOnBehalf(
    address registrant,
    uint256 scheme,
    bytes memory signature,
    bytes memory stealthMetaAddress
  ) external;
}
```

Listing 2. Registry Contract Interface

specifying a scheme ID. For instance, a user could register one stealth meta-address with an elliptic curve $E(F)$ and another for the curve $E(F') \mid F \neq F'$, thus avoiding conflicts. This provision ensures compatibility with future cryptographic methods.

Algorithm 1 — Register Stealth Meta-Address

Input I: Scheme ID id

Input II Stealth Meta-Address SMA in byte-format

Input III: Caller/Signer $\mathcal{C}$

Input IV: (optional): Signature sig

$R^x_{\mathcal{SC}}, R^x_{\mathcal{SP}} \leftarrow parse\_compressed\_pubkeys(SMA)$
$R^{pre}_{\mathcal{SC}}, R^{pre}_{\mathcal{SP}} \leftarrow R^x_{\mathcal{SC}}[0], R^x_{\mathcal{SP}}[0]$
$R^x_{\mathcal{SC}}, R^x_{\mathcal{SP}} \leftarrow R^x_{\mathcal{SC}}[1{:}], R^x_{\mathcal{SP}}[1{:}]$
assert $R^{pre}_{\mathcal{SP}} == (2|3)$ & $R^{pre}_{\mathcal{SC}} == (2|3)$
assert $on\_curve(R^x_{\mathcal{SP}}, id)$ & $on\_curve(R^x_{\mathcal{SC}}, id)$
if sig **then**
    registerKeysOnBehalf$(id, \mathcal{C}, SMA, sig)$
else
    registerKeys$(id, \mathcal{C}, SMA)$

The *registerKeys* function accepts the scheme ID, the stealth meta-address, and, optionally, a signature. The stealth meta-address includes a variable-sized field, allowing for managing diverse public key formats and sizes. As illustrated in Algorithm 1, the registration process involves parsing the stealth meta-address to confirm that the two public keys are on the specified elliptic curve. This validation can be performed off-chain, hence circumventing unnecessary on-chain computations. For other cryptographic methods, other validation methods can be used.

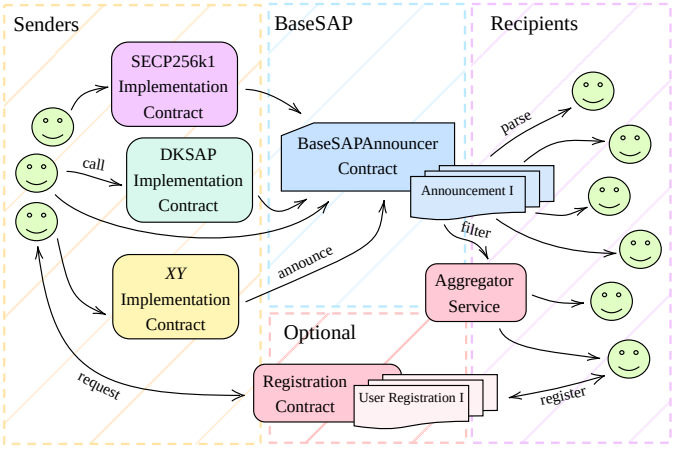

Fig. 3. System Model: A schematic representation showcasing the pivotal *Announcer* contract and optional registration contract, surrounded by implementation contracts and auxiliary services enhancing protocol security and efficiency.

Our proposed *registerKeys* function registers the public keys through a mapping that associates the scheme ID with the stealth meta-address. This mapping is subsequently linked to the registrant's address in another mapping. Senders can interact with the registry to retrieve the stealth meta-address of another user by providing a scheme ID and the recipient's address.

E. Summarized System Model

In Figure 3, we illustrate the integration of the *Announcer* contract in a comprehensive framework. Both senders and recipients involved in stealth address transactions can directly interact with the *Announcer* contract or engage through intermediary contracts. These intermediaries aim to enhance aspects like user experience, security, or efficiency.

For senders, there are two pathways to publish their announcements: through the *Announcer* contract or via an auxiliary implementation contract. This latter option can offer added security features, including the verification of user-provided information such as the scheme ID, stealth address, ephemeral public key, or metadata. Furthermore, intermediary contracts can streamline the process by combining the announcement with the asset transfer into a single transaction. This integration means users don't need separate transactions for transferring funds and conveying SAP-required information.

The design of the *Announcer* contract is highly flexible, allowing various SAPs to connect with it. This feature benefits these protocols by providing a centralized location for announcement publication. The implementation contracts designed for this purpose can be tailored specifically for individual operators/companies or created in a generic manner to cater to the fundamental requirements of multiple entities.

For recipients, there are two pathways to parse announcements: they can either directly extract announcements from the *Announcer* contract or potentially utilize intermediary

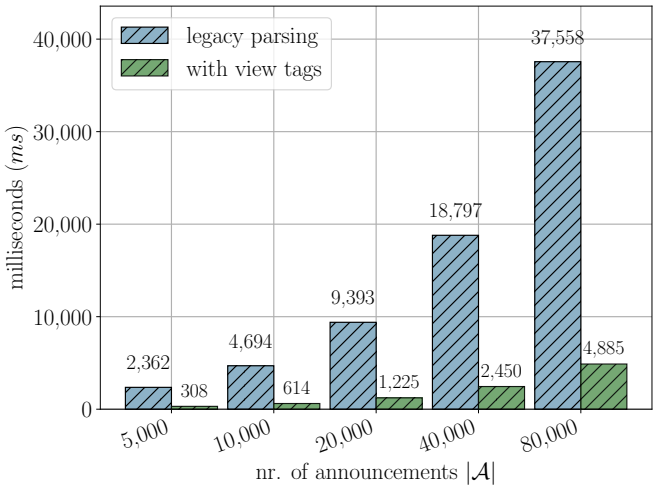

Fig. 4. View tag improvements: Conventional parsing (without employing view tags) versus an upgraded variant using view tags. In summary, the view tags approach operates approx. 7.6 times more efficient concerning parsing time than conventional parsing. Both algorithms have a complexity of $O(n)$.

parsing providers. These providers may offer services ranging from simple spam filtering to more sophisticated sorting for efficiency enhancement. In some cases, equipped with viewing private keys, these intermediaries can completely handle parsing, alerting users upon receiving funds.

Figure 3 also displays the role of the optional registry contract, designed to serve as a centralized repository for stealth meta-addresses used by senders in the transaction process. The contract facilitates two main functions. First, senders have the opportunity to query the registry contract to retrieve the stealth meta-address of a specific user. This feature streamlines the process for senders, providing easy access to the necessary recipient details for initiating transactions. Second, users have the option to register or update their own stealth meta-address within this contract for different cryptographic implementations. By doing so, they signal their readiness to receive transactions via the registered stealth address method. This aspect of the contract allows for dynamic updating and maintenance of stealth meta-addresses, ensuring users can manage their transaction-receiving capabilities effectively across different cryptographic SAP.

VI. Experiments

Figure 4 presents simulations that juxtapose the efficiency gains achieved by implementing view tags, as measured by parsing time. We compare the view tag approach to the conventional DKSAP method (cf. "legacy parsing"). The experiments were carried out on a machine with a 10-core CPU Apple M1 Max chip using Node.js/JavaScript, with the *elliptic.js*[3] and *js-sha3.js*[4] libraries employed for EC operations and hash functions, respectively. No efficiency improvements through multiprocessing were leveraged.

[3]https://github.com/indutny/elliptic
[4]https://github.com/emn178/js-sha3

Legacy parsing requires the recipients to perform the following operations to ascertain if they were the recipients of stealth address transactions:

- 2x ecMUL — DH secret & DH secret to EC point,
- 2x HASH — hash of DH secret & address derivation,
- 1x ecADD — deriving the stealth address

Adopting *view tags* significantly decreases parsing time by approximately 86.84%. In most cases, users are only required to perform a single EC multiplication operation (ecMUL) and a single hash operation (HASH), thus eliminating the necessity for an additional ecMUL, ecADD, and HASH. With a 1-byte view tag, the likelihood that users can bypass the remaining computations after hashing the shared secret is $255/256$. This suggests that users can almost certainly bypass the aforementioned three operations for most announcements. The realized reduction in parsing time comes with a significant positive impact on the user experience. As displayed in Figure 4, for $80{,}000$ announcements, view tags enable the reduction of the parsing time from 37.56 to 4.89 seconds.

TABLE II
Timing Test Results

Operation	Sym.	Iterations	Time (ms)
privKey $\rightarrow$ pubKey	ϕ	1,000	381.442
privKey $\times$ pubKey	μ	1,000	816.732
pubKey + pubKey	α	1,000	48.573
keccak256	κ	1,000	3.209

Table II shows the results of a series of performance tests conducted to measure the time taken for different cryptographic operations. These operations include converting a private key to a public key, multiplying a private key with a public key, adding two public keys, and executing the keccak256 hash function. Each operation was performed 1,000 times to ensure measurement accuracy and consistency. The times, presented in milliseconds, provide valuable insights into the computational overhead of these fundamental cryptographic operations required in most SAPs.

In the following comparison of SAPs, we focus on those that enhance privacy by not requiring the publication of an announcement with the transaction. Specifically, the protocol by Feng *et al.* [15] generates the DH secret based on the transaction count between two addresses, while the protocol by Abdulkader and Kumar [17] utilizes the most recent transaction ID for this purpose. Both approaches contrast with protocols that need additional published information and are thus compatible with BaseSAP. Our analysis compares these with a basic DKSAP implementation, using the results from Table II. We omit the pubkey to address operation, as it remains consistent across all protocols and does not affect the comparative analysis.

For practical insights, we examine "Umbra," a stealth address implementation operational on the Ethereum mainnet since August 2021. By assessing Umbra, we estimate the volume of stealth address transactions. This involves comparing the number of transactions involving announcement publications within the Umbra contract against the total num-

ber of Ethereum transactions within the same period. This comparison helps understand the additional transaction load a user would incur if stealth address transactions were not filterable by specific Announcement events.

From September 1st to November 30th, 2023, Umbra recorded 1,752 stealth address transactions, while Ethereum logged 94,402,727 transactions in total. Consequently, the ratio of non-stealth to stealth address transactions is approximately 53,883 to 1, translating to a stealth address transaction share of 0.001856%.

Utilizing the time measurements from Table II, we analyze the efficiency of the PDKSAP (described in [15]) compared to the standard DKSAP. The PDKSAP requires users to parse significantly more transactions (53,883 times more) than an approach allowing event-based filtering of non-stealth transactions. Standard DKSAP processing of a single announcement requires $\mu+\phi+\alpha+2\kappa$, while PDKSAP operates with $\phi+\alpha+2\kappa$ per transaction, saving one EC multiplication.

Considering a scenario of 5,000,000 total transactions, which include approximately 93 stealth address transactions, the parsing time for PDKSAP is calculated as $5*10^6(\phi+\alpha+2\kappa)$. In contrast, DKSAP requires $93(\mu+\phi+\alpha+2\kappa)$. This translates to a parsing time of 2,182,165 ms (36.37 minutes) for PDKSAP versus 116.544 ms (0.117 seconds) for standard DKSAP, without factoring in potential efficiency gains from employing view tags.

Next, we determine the upper limit for the share of stealth address transactions sT, in relation to the total transactions T, for DKSAP to maintain higher efficiency over PDKSAP:

$$sT(\mu+\phi+\alpha+2\kappa) > T(\phi+\alpha+2\kappa)$$

Solving for s yields:

$$s > \frac{\phi+\alpha+2\kappa}{\phi+\mu+\alpha+2\kappa}$$

Using the measurements of Table II, the share of stealth address transactions must exceed 34,83% for PDKSAP to become more efficient than DKSAP, excluding any optimizations from employing view tags or the time required for PDKSAP to derive the transaction count between two addresses, which would require additional RPC requests.

VII. Security Implications

In this section, we provide an overview of the security implications of our proposed protocol. We focus on DoS attack vectors and privacy implications, particularly user de-anonymization risks.

Threat Model: Our threat model considers two attacks that could potentially compromise the security, efficiency, and privacy of the SAP. We address each threat by implementing appropriate countermeasures:

- **DoS Attacks via Announcement Spamming:** Our protocol is susceptible to DoS attacks where malicious actors could flood the network with excessive announcements, leading to resource exhaustion in the parsing process. We further analyze two primary mechanisms to mitigate these DoS attacks: a *toll* system and a *staking* system.
- **User De-Anonymization:** The risk of de-anonymization in our protocol stems from the potential linkage of stealth addresses to users' real identities. Our protocol counters this by ensuring that each transaction uses a newly generated address for the recipient, rendering transaction linkability and user identification extremely challenging.

Adversarial Model: In general, we assume that an adversary may either try to attack the parsing process by spamming announcements or try to compromise the privacy of users by establishing links between "doxxed" accounts and the stealth address. An adversary can observe the network traffic and might know the identity behind certain addresses already used (e.g., an exchange with Know-Your-Customer processes in place). However, we assume an adversary cannot deanonymize users using the DH secret between sender and recipient based on the hardness of the elliptic curve discrete log problem. Furthermore, our threat model excludes consideration of metadata such as IP addresses, RPC nodes used, and browser characteristics. This exclusion is based on the assumption that such data falls outside the scope of the blockchain protocol itself and is typically addressed through complementary privacy-enhancing technologies such as VPNs, Tor, mixnets, or private gateways.

Security and Privacy Properties: The primary goal of our protocol is to ensure unlinkable interactions between users, thereby anonymizing transaction recipients. Furthermore, the parsing process for users must not introduce significant costs for the user.

DoS and De-anonymization countermeasures: In the following, we further elaborate on DoS attack vectors that may compromise our protocol's efficiency and introduce a cost function to guide the selection of appropriate DoS attack countermeasures. We focus on two DoS attack prevention mechanisms: a *toll* system and a *staking* system. After that, we focus on the privacy implications of the protocol, specifically emphasizing the risks of user de-anonymization.

A. Stealth Addresses and DoS attacks

Minimizing the time and CPU resources required for the parsing process is crucial. As noted earlier, this process involves executing several EC operations off-chain, thereby circumventing the gas costs associated with on-chain transactions. However, this exposes the protocol to the risk of DoS attacks. In such scenarios, attackers can flood the network with announcements, compelling users to perform unnecessary and resource-intensive EC operations on these spurious messages. This not only leads to the wasteful consumption of computational resources but also creates an imbalance where the cost of issuing an announcement may be significantly lower than the cost of processing it. Such a discrepancy can cause operational inefficiencies and degrade the user experience by extending the duration of the parsing process.

It is worth mentioning that the dual-key setup of our proposed SAP enables users to share their private scanning keys with third-party entities specializing in the parsing process. These parties can offer their services for a market price and come equipped with defense measures against DoS attacks. These measures can be based on specific heuristics that help

identify spammers. As a result, third-party parsing providers can provide additional protection against DoS attacks targeting users, thereby ensuring the effectiveness and reliability of the parsing process.

The subsequent section focuses on two different approaches for mitigating DoS attacks. We will discuss the merits of each and explain why a staking-based mechanism is more compatible with our objectives, offering a more robust solution for maintaining the integrity and efficiency of the parsing process.

Toll. The DoS attack vector can be addressed by introducing a *toll* system, which accounts for the computational costs incurred during the parsing process. In particular, the *toll* $\mathcal{T}$ is designed to increase the expense associated with emitting announcements and may be proportionate to the parsing costs c, which are accrued until the hashed DH secret is derived. These costs comprise an EC multiplication $c_{mul} \in c$ and a hashing operation $c_{hash} \in c$. The *toll* can be attached to the transaction and paid by the sender, who is responsible for initiating announcements and thus contributing to the parsing load. This strategy ensures that the parsing costs associated with announcements are not exclusively shouldered by the recipients but also shared by the senders.
For the toll, we assume that:

$$\mathcal{T} \geq c_{mul} + c_{hash},$$

while the costs without using view tags can be described as $2(c_{mul} + c_{hash}) + c_{add}$. To ensure that adversaries pay at least for the effort imposed on a single stealth address recipient, the SAP must charge a fee of up to $\mathcal{T}$.
The keccak-256 opcode, used with a 64 bytes input, costs 42 gas and is therefore negligible. Taking the EIP-196 [34] precompiled contracts for the alt_bn128 curve as a reference for the cost of EC operations, we assume a gas usage of $40{,}000$ units for EC multiplications. Therefore, a toll of $40{,}000$ gas units might be suitable. Based on a gas price of 10 gwei, the toll would amount to 0.0004 ETH.
The primary function of the *toll* within our system is to create an economic deterrent against spam, aiming to render spamming activities financially impractical. It is not designed to offset the total costs incurred during the parsing process. As such, we can significantly lower the *toll* and still effectively discourage DoS attacks. The specific value of the *toll* should be carefully determined based on various factors such as prevailing network conditions, the overall cost structure of the protocol, and the desired level of protection against spamming. Striking the right balance is key to developing an economically viable strategy that efficiently counters DoS attacks while also ensuring that legitimate users are not burdened with undue financial costs.
There are various ways to utilize the collected toll, but ensuring it does not directly return to the originator is essential to maintain the DoS attack protection. One option for the proposed protocol is to send it to the coinbase address of the respective block, which is the address of the block proposer. This approach would distribute the *toll* among block proposers, giving them an additional incentive to include stealth address transactions in blocks. Proposers then have an additional source of extractable value, encouraging them to prioritize stealth address transactions in block creation. However, the initiators of stealth address transactions may offset the expenses associated with the *toll* by reducing the gas price. Since block proposers are indifferent to whether the paid fee originates from transaction fees or payments to a block's coinbase, this strategy would effectively impede the *toll's* spam prevention. In blockchain networks similar to Ethereum, each transaction requires a minimum base fee. This fee is fundamental for including transactions in a block and acts as a safeguard against users' total evasion of transaction costs. This base fee ensures that the implementation of the *toll* system, as described, does not lead to significant issues. However, an alternate approach becomes necessary for blockchain networks that lack such a minimum transaction fee requirement. In these scenarios, redirecting the *toll* to a different beneficiary, like the commonly used burn account at address(0), presents itself as a more appropriate solution for handling these fees.
Additional research and analysis are essential to ascertain the ideal value of the *toll* across diverse network conditions and under various scenarios. This is equally true for determining how to utilize the *toll* effectively without inadvertently introducing elements of trust or centralization.

Staking. A staking system can be implemented to provide users and third-party parsing providers with an additional tool for managing spam and Sybil attacks. The *Announcer* contract, or implementation contracts interacting with the *Announcer* contract, may permit users to stake an arbitrary amount of ETH and lock it within the contract. Users or third-party parsing providers can then confirm if the sender of a stealth address transaction has staked the required minimum collateral. If not, they can deprioritize the respective announcements of that sender in the parsing process.
Analogous to the ERC-4337 standard, a *MIN_STAKE_VALUE* and a *MIN_UNSTAKE_DELAY* variable are established. The latter could be directly encoded into contracts communicating with the *Announcer* contract and may be set to one day. The *MIN_STAKE_VALUE* can be agreed upon by network participants off-chain and may change over time. Theoretically, every parsing provider may independently set the minimum stake required for prioritization.
We define the staking system as follows:

- Let $\mathcal{A}$ be the set of all announcements, where
$$A = \{a_1, a_2, a_3, \ldots, a_n\}.$$
- Let U be the set of all users, where
$$U = \{u_1, u_2, u_3, \ldots, u_n\}.$$
- For each user $u \in U$, let $D(u)$ be the amount of ETH staked by user u.
- Let F be the function that maps a prioritization factor PF to users $F : u_i \rightarrow PF$.

We can define two priority factors, one based on the amount of ETH staked ($PF1$) and the other based on the number of announcements made by a user ($PF2$).

1) **PF1: Staking Priority Factor**
For each user $u_i \in U$ and their corresponding staked ETH amount $D(u_i)$, we can define the staking priority factor ($PF1$) as:

$$PF1(u_i) = min(D(u_i),\ MIN\_STAKE\_VALUE).$$

Users staking more than the *MIN_STAKE_VALUE* are assigned a first prioritization factor equaling the *MIN_STAKE_VALUE*

2) **PF2: Announcement Count Priority Factor**
For each user $u_i \in U$ and the number of announcements made by u_i, the announcement count priority factor can be defined as:

$$n(u_i) = |\{a_j \in \mathcal{A} : a_j \text{ is made by } u_i\}|$$

To discourage spamming, we want to assign a higher priority to users who made fewer announcements. We can define the announcement count priority factor ($PF2$) as:

$$PF2(u_i) = \frac{1}{n(u_i)}$$

Higher values of $PF2$ indicate higher priority for a user's announcements.

Now, we can combine these two priority factors into a single prioritization factor (PF) for each user:

$$PF(u_i) = w_1 \cdot PF1(u_i) + w_2 \cdot PF2(u_i)$$

where w_1 and w_2 are weights assigned to $PF1$ and $PF2$, respectively, to balance their importance in determining the overall priority. For example, if we want to prioritize ETH staked over the number of announcements made, we could set $w_1 > w_2$. Initially, w_1 and w_2 may be set to 1.

Finally, parsing providers can order the list of announcements based on each user's computed PF values, prioritizing announcements made by users with higher PF values.

The proposed mechanism guarantees that the announcements from staking users are prioritized in the parsing process over those from users who did not stake. Furthermore, announcements from users with fewer announcements are given precedence.

It is worth noting that the staking-based DoS attack prevention is implemented on the parsing side, allowing parsing users and third-party parsing providers to manage spam more effectively. Spamming users can be disregarded or deprioritized when serving their announcements to parsing users. By requiring a stake for prioritization, Sybil attacks become inefficient, and the stake of known spammers can be traced. This prevents spammers from switching addresses to evade deprioritization. This method gives preference to users who engage in fewer stealth address transactions, ranking those with a greater volume of such transactions lower in terms of priority. While this might seem unfair, especially in the context of exchanges that may handle a large number of stealth address transactions, it's important to note that exchanges are anticipated to establish their own parsing systems. These dedicated systems will likely incorporate advanced caching strategies and could be designed to prioritize transactions involving the exchange's own addresses above others. This approach balances the needs of individual users and large entities like exchanges, ensuring an efficient and equitable system.

Beyond its role in thwarting spam through Sybil attacks, the staking approach offers the advantage of not incurring additional user costs. Users can lock the required minimum stake directly in the contract that interacts with the *Announcer* contract. This integration facilitates a smooth and uninterrupted user experience, aligning with the system's overall efficiency and user-friendliness.

Based on the reasons discussed, we deem the staking approach superior for our specific case instead of requiring a *toll* for every stealth address transaction.

B. Privacy Guarantees and De-anonymization

Stealth addresses offer an extra layer of privacy, enabling users to engage with each other discreetly without public disclosure of their interactions. However, maintaining privacy as a recipient of stealth addresses requires careful consideration of several factors. Errors, such as the compromise of private keys, can lead to de-anonymization or, worse, the loss of user funds. Therefore, it is essential for users to be vigilant and well-informed to safeguard their anonymity and assets effectively.

Commingled Funds. Ethereum de-anonymization studies commonly employ various heuristics to cluster addresses based on user activity [35]–[38]. The use of stealth addresses on Ethereum has the potential to improve privacy by shielding the identity of the recipient of funds. However, commingling these funds, i.e., mixing them with other assets, poses a risk to this privacy enhancement. Particularly, when stealth address funds become intermingled with "doxxed" funds — assets already associated with a specific individual or entity through public records or other means — the initial privacy benefits are compromised. This intermingling can occur during transactions involving withdrawals from a stealth address. Users who lack a comprehensive understanding of the privacy enhancements provided by stealth addresses may inadvertently transfer funds from a stealth address to an identifiable, or "doxxed," address. This practice can effectively erode the anonymity of the stealth recipient. Therefore, it is paramount for users to thoughtfully assess the risks associated with commingling funds destined for a stealth address to maintain the intended level of privacy.

Transaction fee funding. To manage the transaction fees required for activities like spending an ERC-20 token or executing approval rights on the Ethereum blockchain, recipients need to supply their stealth addresses with a small amount of ETH. This is essential to avoid reliance on a publicly identifiable address. To address this, senders can include a nominal amount of ETH with each stealth address transaction, thereby equipping recipients with the necessary funds to cover gas costs for on-chain actions. Alternatively, recipients can fund their stealth addresses with anonymized ETH retrieved through another privacy tool, such as Tornado Cash, or acquired via a trusted centralized exchange.

Another solution to the recipient's transaction fee funding problem involves entrusting specialized transaction aggrega-

tors, often known as "searchers" in the Miner Extractable Value (MEV) context. These intermediaries can provide users with the option of a one-time payment in exchange for a batch of "tickets," which are subsequently used to cover the on-chain inclusion costs of transactions. When a user intends to initiate a transaction from a stealth address, they present the aggregator with one such ticket. This ticket is encoded using a Chaumian blinding scheme, a protocol widely employed in the privacy-focused e-cash systems first proposed by David Chaum in 1983 [28]. Upon receipt of the ticket, the aggregator funds the recipient's account bundles the transaction with others and includes it within a block. Given that the funds involved in this process are minimal and exclusively used for covering transaction fees, the trust prerequisites are significantly lower than those associated with a full-scale implementation of privacy-preserving e-cash. This approach has significant potential to bridge the gap between privacy and functionality of stealth addresses.

Stealth Address Detection. A critical balance between privacy and detectability must be considered when addressing stealth address transactions. A public on-chain announcement is made within the proposed protocol whenever a stealth address transaction occurs, potentially enabling blockchain forensics to discern related transactions. To mitigate this issue, it is possible to broadcast the announcement through channels different from the stealth address transaction or via a separate transaction, breaking the link between the announcements and the actual transactions.

Detectability is not an issue exclusive to Ethereum-based privacy tools. Prominent Bitcoin CoinJoin wallets, such as Wasabi and Samurai Wallet, integrate techniques enabling users to mix their funds and obscure their origins. Despite efforts, the resulting transactions may still be identifiable using certain heuristic techniques [26], [39], [40]. Similarly, Tornado Cash, a popular privacy tool within the Ethereum ecosystem, confronts a comparable challenge, as any deposit and withdrawal can be identified through publicly available event logs. However, it is crucial to recognize that external observers cannot de-anonymize the recipient of a stealth address interaction without access to the shared DH secret between the sender and the recipient.

VIII. Conclusion

Stealth addresses have significant potential to enhance the privacy of programmable blockchain interactions. This work proposes BaseSAP as a blockchain-based foundation-layer protocol for stealth addresses compatible with different cryptographic schemes.

BaseSAP is designed to function entirely autonomously, leveraging the immutable nature of Smart Contracts to deliver the required functionality for deploying interoperable stealth addresses on programmable blockchains. Compared to the previous solutions, the protocol's modularity not only encourages the evolution of cohesive auxiliary layers on top of its core implementation but also underscores its flexibility in supporting various user applications, such as programmable wallets, public goods funding, Decentralized Finance (DeFi), Non-Fungible Tokens (NFTs), and more.

Through the simulations of an optimized Secp256k1-based stealth address protocol, we demonstrated the operational effectiveness of BaseSAP, the results of which we validated on the Goerli and Sepolia test networks via our preliminary prototype implementations.

Additionally, we conducted thorough cost analyses and identified potential security vulnerabilities and attack vectors that could undermine the privacy offered by SAPs. These insights underscore the critical need to address privacy issues prevalent in public and distributed blockchains.

In conclusion, our research provides the basis for implementing stealth address technology on programmable blockchains. It effectively demonstrates the efficacy of BaseSAP in augmenting the privacy of public blockchain transactions. Furthermore, it underlines the significant potential of such protocols, especially when applied on the application layer of programmable blockchains, for boosting interoperability across various aspects of blockchain technology. The code base created for this work is available under an open-source license to ensure reproducibility and transparency [11]. The described protocol is available as an ERC (Ethereum Request for Comment) [10].

References

[1] N. van Saberhagen, "Cryptonote v 2.0," Oct 2013. [Online]. Available: https://web.archive.org/web/20201028121818/https://cryptonote.org/whitepaper.pdf

[2] P. Todd, "[bitcoin-development] stealth addresses," Jan 2014. [Online]. Available: https://web.archive.org/web/20220209182020/https://www.mail-archive.com/bitcoin-development@lists.sourceforge.net/msg03613.html

[3] "The monero project." [Online]. Available: https://www.getmonero.org/

[4] V. Buterin, "Ethereum white paper: A next generation smart contract & decentralized application platform," 2013. [Online]. Available: https://ethereum.org/en/whitepaper/

[5] C. Feng, L. Tan, H. Xiao, X. Qi, Z. Wen, and Y. Liu, "Edksap : Efficient double-key stealth address protocol in blockchain," in *2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, 2021, pp. 1196–1201.

[6] W. Li, Z. Lin, and Q. Chen, "A hybrid design of linkable ring signature scheme with stealth addresses," *Security and Communication Networks*, vol. 2022, 2022.

[7] J. Fan, Z. Wang, Y. Luo, J. Bai, Y. Li, and Y. Hao, "A new stealth address scheme for blockchain," in *Proceedings of the ACM Turing Celebration Conference - China*, ser. ACM TURC '19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3321408.3321573

[8] Z. Liu, K. Nguyen, G. Yang, H. Wang, and D. S. Wong, "A lattice-based linkable ring signature supporting stealth addresses," in *Computer Security – ESORICS 2019*, K. Sako, S. Schneider, and P. Y. A. Ryan, Eds. Cham: Springer International Publishing, 2019, pp. 726–746.

[9] M. F. Esgin, R. Steinfeld, and R. K. Zhao, "Matrict+: More efficient post-quantum private blockchain payments," Cryptology ePrint Archive, Paper 2021/545, 2021, https://eprint.iacr.org/2021/545. [Online]. Available: https://eprint.iacr.org/2021/545

[10] A. Wahrstätter, M. Solomon, B. DiFrancesco, and V. Buterin, "ERC-5564: Stealth Addresses," https://eips.ethereum.org/EIPS/eip-5564, 2022, [Online; accessed 19.06.2023].

[11] A. Wahrstätter and M. Solomon, "EIP-Stealth-Address-ERC: Stealth Addresses for Ethereum," https://github.com/nerolation/EIP-Stealth-Address-ERC, 2023.

[12] bytecoin, "Untraceable transactions which can contain a secure message are inevitable," https://bitcointalk.org/index.php?topic=5965.0, 2011, accessed: [insert date of access].

[13] N. T. Courtois. and R. Mercer., "Stealth address and key management techniques in blockchain systems," in *Proceedings of the 3rd International Conference on Information Systems Security and Privacy - ICISSP,*, INSTICC. SciTePress, 2017, pp. 559–566.
[14] X. Fan, "Faster dual-key stealth address for blockchain-based internet of things systems," in *Blockchain – ICBC 2018*, S. Chen, H. Wang, and L.-J. Zhang, Eds. Cham: Springer International Publishing, 2018, pp. 127–138.
[15] C. Feng, L. Tan, H. Xiao, K. Yu, X. Qi, Z. Wen, and Y. Jiang, "Pdksap : Perfected double-key stealth address protocol without temporary key leakage in blockchain," in *2020 IEEE/CIC International Conference on Communications in China (ICCC Workshops)*, 2020, pp. 151–155.
[16] D. Lee and M. Song, "Mexchange: A privacy-preserving blockchain-based framework for health information exchange using ring signature and stealth address," *IEEE Access*, vol. 9, pp. 158 122–158 139, 2021.
[17] M. M. AbdulKader and S. G. Kumar, "A privacy-preserving data transfer in a blockchain-based commercial real estate platform using random address generation mechanism," *The Journal of Supercomputing*, pp. 1–27, 2022.
[18] E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin," 2014. [Online]. Available: http://zerocash-project.org/media/pdf/zerocash-extended-20140518.pdf
[19] S. Bowe, A. Gabizon, and I. Miers, "Scalable multi-party computation for zk-snark parameters in the random beacon model," *Cryptology ePrint Archive*, 2017.
[20] A. Banerjee, M. Clear, and H. Tewari, "Demystifying the role of zk-snarks in zcash," in *2020 IEEE Conference on Application, Information and Network Security (AINS)*, 2020, pp. 12–19.
[21] Z. Guan, Z. Wan, Y. Yang, Y. Zhou, and B. Huang, "Blockmaze: An efficient privacy-preserving account-model blockchain based on zk-snarks," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 3, pp. 1446–1463, 2022.
[22] A. Pertsev, R. Semenov, and R. Storm, "Tornado cash privacy solution version 1.4," 2019.
[23] Z. J. Williamson, "The aztec protocol," *URL: https://github.com/AztecProtocol/AZTEC*, 2018.
[24] V. Buterin, J. Illum, M. Nadler, F. Schär, and A. Soleimani, "Blockchain privacy and regulatory compliance: Towards a practical equilibrium," *Available at SSRN*, 2023.
[25] G. Maxwell, "CoinJoin: Bitcoin privacy for the real world," Aug. 2013. [Online]. Available: https://bitcointalk.org/index.php?topic=279249.0
[26] Ádám Ficsór, "Zerolink the bitcoin fungibility framework," 2017. [Online]. Available: https://github.com/nopara73/ZeroLink
[27] D. Deuber and D. Schröder, "Coinjoin in the wild," in *European Symposium on Research in Computer Security*. Springer, 2021, pp. 461–480.
[28] D. Chaum, "Blind signatures for untraceable payments," in *Advances in Cryptology*, D. Chaum, R. L. Rivest, and A. T. Sherman, Eds. Boston, MA: Springer US, 1983, pp. 199–203.
[29] Certicom Research, "secp256k1," http://www.secg.org/sec2-v2.pdf, 2010.
[30] A. Takahashi and M. Tibouchi, "Degenerate fault attacks on elliptic curve parameters in openssl," in *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, 2019, pp. 371–386.
[31] S. Zhai, Y. Yang, J. Li, C. Qiu, and J. Zhao, "Research on the application of cryptography on the blockchain," in *Journal of Physics: Conference Series*, vol. 1168, no. 3. IOP Publishing, 2019, p. 032077.
[32] M. A. Mehrabi, C. Doche, and A. Jolfaei, "Elliptic curve cryptography point multiplication core for hardware security module," *IEEE Transactions on Computers*, vol. 69, no. 11, pp. 1707–1718, 2020.
[33] UkoeHB, "Reduce scan times with 1-byte-per-output 'view tag' #73," 2020. [Online]. Available: https://github.com/monero-project/research-lab/issues/73
[34] C. Reitwiessner. (2017, February) Eip-196: Precompiled contracts for addition and scalar multiplication on the elliptic curve alt_bn128. Ethereum Improvement Proposals. [Online; accessed 23.06.2023]. [Online]. Available: https://eips.ethereum.org/EIPS/eip-196
[35] W. Chan and A. Olmsted, "Ethereum transaction graph analysis," in *2017 12th International Conference for Internet Technology and Secured Transactions (ICITST)*, 2017, pp. 498–500.
[36] T. Chen, Y. Zhu, Z. Li, J. Chen, X. Li, X. Luo, X. Lin, and X. Zhange, "Understanding ethereum via graph analysis," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, 2018, pp. 1484–1492.
[37] D. Lin, J. Wu, Q. Yuan, and Z. Zheng, "Modeling and understanding ethereum transaction records via a complex network approach," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 11, pp. 2737–2741, 2020.
[38] A. Kovács and I. A. Seres, "Anonymity analysis of the umbra stealth address scheme on ethereum," 2023.
[39] T. Tironsakkul, M. Maarek, A. Eross, and M. Just, "The unique dressing of transactions: Wasabi coinjoin transaction detection," in *Proceedings of the 2022 European Interdisciplinary Cybersecurity Conference*, ser. EICC '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 21–28. [Online]. Available: https://doi.org/10.1145/3528580.3528585
[40] A. Wahrstätter, J. Gomes, S. Khan, and D. Svetinovic, "Improving cryptocurrency crime detection: Coinjoin community detection approach," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–11, 2023.

Anton Wahrstätter received his B.Sc. in Economics and LL.B. in Business Law from the University of Innsbruck, Austria, in 2018 and 2019, respectively. He completed his M.Sc. in Digital Business from the University of Innsbruck, Austria, in 2020 and an M.Sc. in Blockchain and Digital Currency from the University of Nicosia, Cyprus, in 2022. He has been contributing to the Bitcoin and Ethereum community as an open-source developer since 2016, primarily focusing on privacy and quantitative data analysis. He is currently affiliated with the Institute for Distributed Ledgers and Token Economy at the Vienna University of Economics and Business, and the Research Institute for Cryptoeconomics in Vienna, Austria. His research focuses on blockchain privacy and trust, as well as the application of data science techniques to blockchain data.

Matthew Solomon received his Bachelor of Science (B.S.) in Aerospace Engineering from the University of Miami, US in 2014 and an M.S. in Aerospace, Aeronautical, and Astronautical/Space Engineering from the University of Maryland, US in 2016. He then spent several years at Lockheed Martin as an aerospace engineer and developed a keen interest in the field of cryptocurrencies. His academic and professional pursuits have led him to cultivate a deep interest in the intersections of open-source software development, privacy, and blockchain technology. Presently, Matt contributes to these domains, focusing on the design and implementation of privacy-oriented smart contract mechanisms.

Ben DiFrancesco earned his B.S. degree in Aerospace Engineering and began his career at Boeing as an aerospace engineer. In 2013, around the same time he discovered Bitcoin, he founded ScopeLift, a company initially focused on native mobile development. As his interest in cryptocurrencies grew, Ben pivoted the focus of ScopeLift towards the emerging crypto ecosystem. An engineer at heart, Ben has fostered a culture of technical excellence at ScopeLift. Currently, he is deeply involved in enhancing privacy on blockchains like Ethereum, particularly through the implementation of stealth address protocols.

Vitalik Buterin is a renowned computer scientist and programmer, best known for co-founding Ethereum, a pioneering platform in blockchain technology. His engagement in the world of cryptocurrency began in 2011 when he co-founded Bitcoin Magazine. His most significant contribution came in 2015 with the launch of Ethereum. Buterin has written numerous influential papers and is a highly cited researcher in the field of blockchain technology. His research interests include blockchain technology, cryptoeconomics, consensus protocols, privacy and scalability solutions, and the security of blockchain systems and smart contracts. His pioneering work on Ethereum has significantly impacted the landscape of blockchain technology and the broader field of cryptocurrency.

Davor Svetinovic (SM'16) is a professor of computer science at the Department of Computer Science, Khalifa University, Abu Dhabi, and the Department of Information Systems and Operations Management, Vienna University of Economics and Business, Austria (on leave), where he is the head of the Institute for Distributed Ledgers and Token Economy, and the Research Institute for Cryptoeconomics. He received his doctorate in computer science from the University of Waterloo, Waterloo, ON, Canada, in 2006. Previously, he worked at TU Wien, Austria, and Lero – the Irish Software Engineering Center, Ireland. He was a visiting professor and a research affiliate at MIT and MIT Media Lab, MIT, USA. Davor has extensive experience working on complex multidisciplinary research projects. He has published over a hundred papers in leading journals and conferences and is a highly cited researcher in blockchain technology. His research interests include cybersecurity, blockchain technology, cryptoeconomics, trust, and software engineering. His career has furthered his interest and expertise in developing advanced research capabilities and institutions in emerging economies. He is a Senior Member of IEEE and ACM (Lifetime) and an affiliate member of the Mohammed Bin Rashid Academy of Scientists.